RX23W グループ

BLE モジュール Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT) を使用した、BLE モジュールについて説明します。本モジュールは Bluetooth® Low Energy 通信の制御を行います。以降、本モジュールを BLE 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 Specification (https://www.bluetooth.com)
RX23W グループユーザーズマニュアル　ハードウェア編 (R01UH0823)
Firmware Integration Technology ユーザーズマニュアル (R01AN1833)
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 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 Renesas FreeRTOS (R01AN4307)
Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)
RX23W グループ Bluetooth 専用クロック周波数の調整手順 (R01AN4762)
Bluetooth Low Energy プロファイル開発者ガイド (R01AN6459)
Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205)
RX23W グループ Bluetooth Low Energy アプリケーション開発者ガイド(R01AN5504)
Bluetooth Low Energy MCU Bluetooth Test Tool Suite 操作説明書(R01AN4554)

目次

1. 概要

BLE FIT モジュールは Bluetooth SIG が規定する Bluetooth Core Specification version 5.0 に準拠した Bluetooth Low Energy (以下、Bluetooth LE) 機能を提供します。以下に本モジュールがサポートする機能を示します。

Bluetooth 5.0 機能
- LE 2M PHY
- LE Coded PHY
- LE Advertising Extensions
- LE Channel Selection Algorithm #2
- High Duty Cycle Non-Connectable Advertising

Bluetooth 4.2 機能
- LE Secure Connections
- Link Layer privacy
- Link Layer Extended Scanner Filter policies
- LE Data Packet Length Extension

Bluetooth 4.1 機能
- LE L2CAP Connection Oriented Channel Support
- Low Duty Cycle Directed Advertising
- 32-bit UUID Support in LE
- LE Link Layer Topology
- LE Ping

1.1 使用方法

BLE FIT モジュールは API として、プロジェクトに組み込んで使用します。BLE FIT モジュールの組み込み方については、「2.12 FIT モジュールの追加方法」を参照してください。

1.2　ソフトウェア構成

BLE FIT モジュールのソフトウェア構成について図 1-1 に示します。

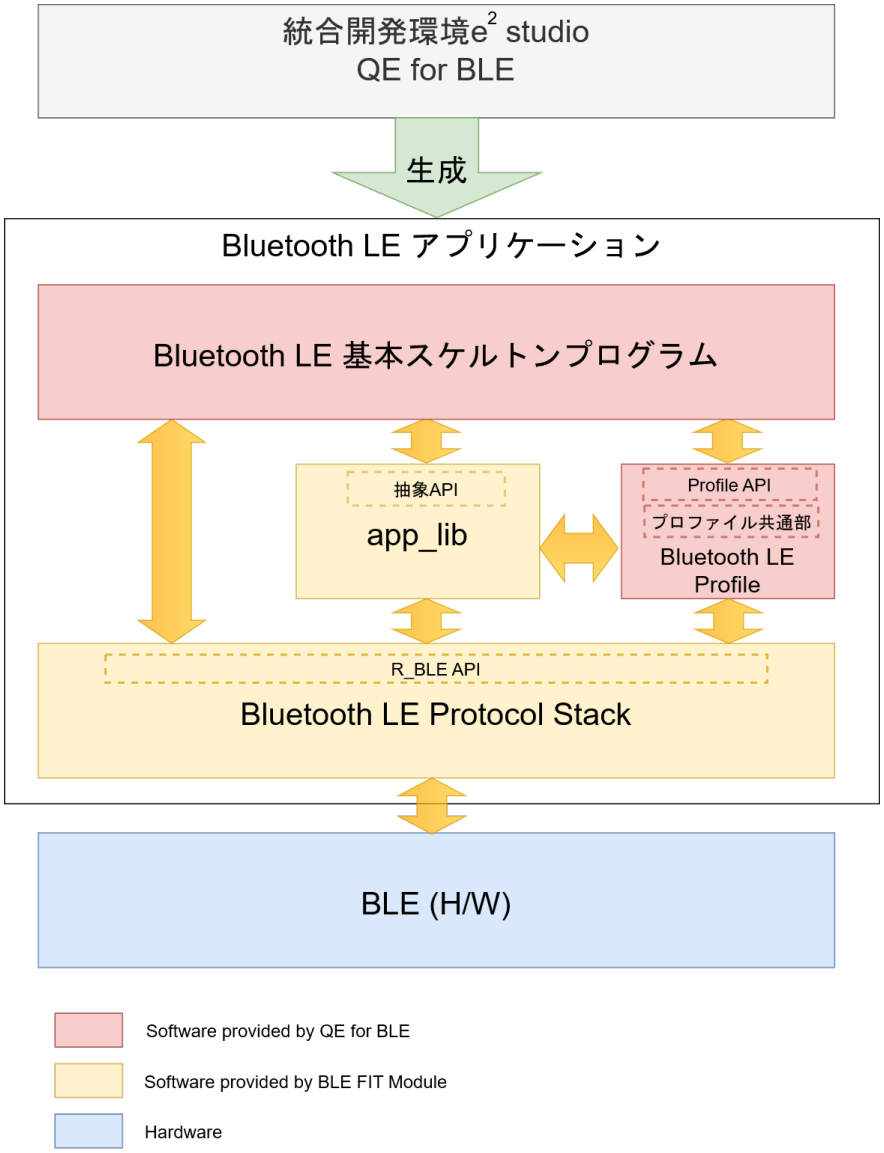


図 1-1 : BLE FIT モジュールのソフトウェア構成

BLE FIT モジュールは Bluetooth LE Protocol Stack と app_lib から構成されています。
Bluetooth LE Protocol Stack が提供する R_BLE API 関数をコールすることにより、Bluetooth LE アプリケーションは Bluetooth LE 機能の使用が可能となります。
app_lib は Bluetooth LE アプリケーションが利用可能な補助機能を提供します。そのひとつの抽象化した R_BLE API(抽象 API)を使うことで、よく使う Bluetooth LE 機能を簡単に使用することができます。
QE for BLE はアプリケーションとプロファイル開発用のスケルトンプログラムを出力します。
BLE FIT v2.50 以降はプロファイル共通部を含みません。QE for BLE V1.6.0 以降で提供されます。
ルネサスでは、QE for BLE を使用した、Bluetooth LE アプリケーション開発を推奨しています。

1.3 ディレクトリ／ファイル構成

BLE FIT モジュールのディレクトリ／ファイル構成を表 1.1 に示します。

表 1.1 : BLE FIT モジュールのディレクトリ／ファイル構成

ディレクトリ/ファイル構成					概要
r_ble_rx23w	doc	en	r01an4860ejXXXX-rx23w-ble.pdf		アプリケーションノート(英語)
		jp	r01an4860jjXXXX-rx23w-ble.pdf		アプリケーションノート(日本語)
		r_ble_api_spec.chm			R_BLE API ドキュメント
	lib	ble_fit_lib_selector.bat			ライブラリ選択バッチファイル
		ble_fit_lib_selector.sh			ライブラリ選択シェルスクリプト
		lib_ble_ps_ccrx_a.lib			Bluetooth LE Protocol Stack (ALL Features)
		lib_ble_ps_ccrx_b.lib			Bluetooth LE Protocol Stack (Balance)
		lib_ble_ps_ccrx_c.lib			Bluetooth LE Protocol Stack (Compact)
		lib_ble_ps_hci_ccrx_a.lib			HCI モードライブラリ(ALL Features)
		lib_ble_ps_hci_ccrx_b.lib			HCI モードライブラリ(Balance)
		lib_ble_ps_hci_ccrx_c.lib			HCI モードライブラリ(Compact)
	src	app_lib	abs		抽象 API(GAP)
			board		LED and Switch 制御
			cli		コマンドライン (ターミナルソフトへの入出力処理)
			cmd		コマンドライン (コマンド実装部分)
			logger		ロガー
			rtos		BLE タスク制御部
			sec_data		セキュリティデータ管理
			timer		ソフトウェアタイマ
		platform	driver	dataflash	BLE 用 Data Flash 制御ドライバ
			r_ble_pf_config_private.h		BLE コンフィギュレーション制御部
			r_ble_pf_configs.c		
			r_ble_pf_functions.c		RF ドライバプラットフォーム依存部
			r_ble_pf_lowpower.c		MCU 消費電力低減機能プログラム
	r_ble_rx23w_if.h				BLE インタフェース定義ファイル
	readme.txt				readme
r_config	r_ble_rx23w_config.h				コンフィギュレーションオプションファイル

BLE FIT v2.5.0 以降は app_lib にプロファイル共通部(profile_cmn, discovery)を含みません。
XXXX にはリビジョン番号が入ります。

2. API 情報

BLE FIT モジュールの API はルネサスの API の命名基準に従っています。

2.1 ハードウェア要件

BLE FIT の Bluetooth LE Protocol Stack は表 2.1 の RX23W のハードウェア機能を常に使用します。そのため、これらのハードウェア機能はユーザアプリ等では使用できません。

表 2.1　BLE FIT が使用する RX23W のハードウェア機能

ハードウェア機能
BLE
CMT: CMT2, CMT3
割り込み: BLEIRQ, BLE.ERI/RXI/TXI/TEI

BLEIRQ, BLE.ERI/RXI/TXI/TEI の優先レベルは 14 です。

BLE FIT のオプションは表 2.2 の RX23W のハードウェア機能を使用します。そのため、オプションで選択したハードウェア機能はユーザアプリ等では使用できなくなります。各オプションの詳細については「2.9 app_lib の機能」をご参照ください。

表 2.2　オプション機能が使用する RX23W のハードウェア機能

ハードウェア機能	オプション	備考
CMT: CMT0 or CMT1	ソフトウェアタイマ	ベアメタル版の場合、CMT FIT がソフトウェアタイマ用に CMT0 or CMT1 を動的に割り当てます。 FreeRTOS 版の場合、ソフトウェアタイマは FreeRTOS のタイマ API を使用します。
割り込み	コマンドライン	SCIn.ERI/RXI/TXI/TEI を使用します。 (n はコマンドラインが使用する SCI のチャネル番号。)
	LED and Switch 制御	Switch 用に使用します。使用するボードに合わせてスマート・コンフィグレータ上で設定します。詳細については表 4.1 をご参照ください。
Data Flash	デバイス固有データ	それぞれのオプションが使用する Data Flash の領域はスマート・コンフィグレータから選択します。 詳細については「4.5.3 セキュリティデータ管理機能/デバイス固有データ(データ領域)」をご参照ください。
	セキュリティデータ管理	
SCI	コマンドライン	使用する SCI のチャネルはスマート・コンフィグレータから選択します。詳細については「4.5.2 コマンドライン機能」をご参照ください。
ポート機能	LED and Switch 制御	使用するポートはスマート・コンフィグレータから選択します。詳細については表 4.1 をご参照ください。

2.2　ソフトウェア要件

BLE FIT モジュールは以下の FIT モジュールに依存しています。

- BSP(r_bsp)
  ※BLE FIT モジュール v1.01 以降は BSP v5.40 以降をご使用ください。
  　RX23W モジュールの型番 R5F523W8CxLN, R5F523W8DxLN を選択する場合、BSP v5.62 以降をご使用ください。
- CMT (r_cmt_rx, v4.10 以降)
  ※Bluetooth LE Protocol Stack は CMT2, CMT3 を使用します。v4.50 より古い場合、「4.6.1 CMT の変更」に説明する CMT の変更を行ってください。v4.50 以降は、「4.6.1 CMT の変更」の変更は必要ありません。app_lib/timer のソフトウェアタイマ機能を使用する場合は、さらに 1ch 使用します。

- LPC(r_lpc_rx)

また、以下の FIT モジュールは、コンフィギュレーションオプションの設定によって必要となります。

コマンドライン機能使用時(BLE_CFG_CMD_LINE_EN=1 の場合) :
- SCI(r_sci_rx)
- byte queues/circular buffers (r_byteq_rx)

セキュリティデータ管理機能使用時(BLE_CFG_EN_SEC_DATA=1 の場合)またはデバイス固有データ(データ領域)使用時(BLE_CFG_DEV_DATA_DF_BLOCK=0～7 の場合) :
- Data Flash (r_flash_rx)

LED and Switch 制御使用時(BLE_CFG_BOARD_LED_SW_EN=1 の場合) :
- GPIO (r_gpio_rx)
  RX23W モジュールの型番 R5F523W8CxLN, R5F523W8DxLN を選択する場合、GPIO v3.80 以降をご使用ください。
- IRQ (r_irq_rx)

2.3 制限事項

2.3.1 ペリフェラルロールでの接続パラメータに関する制限事項

- BLE FIT v2.40 以前のバージョンでは以下の制限事項があります。

RX23W がペリフェラルロールとして接続したときに Connection Interval が 66.25ms 以下である場合、Supervision Timeout による切断後から再接続するまでの間に RF スリープモードに遷移できなくなる場合があります。

本現象を回避するために、Supervision Timeout 時間を Connection Interval の 300 倍以下に設定してください。

例：Connection Interval: 20ms の場合、 Supervision Timeout: 20ms * 300 = 6sec 以下となるようにセントラルの接続パラメータを設定

もしセントラルから上記条件を超える Supervision Timeout 値を設定された場合、ペリフェラル側から R_BLE_GAP_UpdConn()による接続パラメータの更新を行ってください。

- BLE FIT v2.50 以降のバージョンでは以下の制限事項があります。

RX23W がペリフェラルロールとして接続され Connection Interval が 66.25ms 以下に設定されたとき、セントラルロールが設定した Supervision Timeout よりも早く切断する場合があります。

接続を維持する時間の上限は、接続時に通知されるセントラルの Sleep Clock Accuracy (SCA)によって変化します。
表 2.3 に設定された Connection Interval とセントラルの SCA 値による接続維持時間の上限値を示します。

表 2.3 Connection Interval と SCA 値による接続維持時間の上限値

Connection Interval	Central Sleep Clock Accuracy							
	0 (500ppm)	1 (250ppm)	2 (150ppm)	3 (100ppm)	4 (75ppm)	5 (50ppm)	6 (30ppm)	7 (20ppm)
7.5 ms	3390 ms	5670 ms	5670 ms	8520 ms	8520 ms	8520 ms	8520 ms	8520 ms
15.0 ms	8160 ms	12870 ms	15000 ms	18030 ms	22560 ms	22560 ms	22560 ms	22560 ms
20.0 ms	11320 ms	17000 ms	21260 ms	24300 ms	28360 ms	28360 ms	32000 ms	32000 ms
30.0 ms	18360 ms	26970 ms	31980 ms	31980 ms	31980 ms	31980 ms	31980 ms	31980 ms
40.0 ms	24600 ms	32000 ms	32000 ms	32000 ms	32000 ms	32000 ms	32000 ms	32000 ms
50.0 ms	30800 ms	32000 ms	32000 ms	32000 ms	32000 ms	32000 ms	32000 ms	32000 ms
60.0 ms	31980 ms	31980 ms	31980 ms	31980 ms	31980 ms	31980 ms	31980 ms	31980 ms
66.25 ms	31800 ms	31800 ms	31800 ms	31800 ms	31800 ms	31800 ms	31800 ms	31800 ms
67.5 ms ~ 4 sec	制限なし	制限なし	制限なし	制限なし	制限なし	制限なし	制限なし	制限なし

RX23W のペリフェラル動作において、これらの上限値を超えた Supervision Timeout の接続を受け付けますが、セントラルのパケットが受信出来ない状態が上限値時間に到達した時点で切断と判断し、アプリケーションへ切断イベント(BLE_GAP_EVENT_DISCONN_IND)を通知します。
Supervision Timeout が上限値を超えない場合は従来どおり Supervision Timeout で指定された時間の到達で切断と判断します。

RX23W のセントラル動作では本制限事項は適用されません。

2.3.2 Resolvable Private Address (RPA) 機能を有効にした場合の制限事項

- BLE FIT v2.50 以前のバージョンでは以下の制限事項があります。

R_BLE_GAP_EnableRpa()により RPA 機能を有効にした状態で、暗号化された接続でのデータ通信を繰り返すアプリケーションにおいて、以下のいずれかの動作中に暗号化データのパケット送受信と RPA 解決(注 1)もしくは RPA 生成(注 2)のハードウェア処理タイミングが重なった場合に、R_BLE_Execute()から処理が戻らない現象が発生する可能性があります。

- R_BLE_GAP_StartAdv() / R_BLE_GAP_StartPerdAdv()により開始したアドバタイジング動作
- R_BLE_GAP_StartScan() / R_BLE_GAP_CreateConn()により開始したスキャン動作

(注 1) RPA 解決のハードウェア処理が行われるタイミングは以下となります。
- アドバタイジング動作時の SCAN_REQ または CONNECT_IND / REQ パケットの受信
- スキャン動作時のアドバタイジングパケットの受信

(注 2) RPA 生成のハードウェア処理はデフォルトの 900 秒(15 分)もしくは R_BLE_GAP_SetRpaTo()で指定した時間で周期的に実行されます。

この現象はウォッチドッグタイマ(WDT)などで検出し MCU リセットを実行することが必要となります。
本現象が問題となる場合は、恒久回避策として BLE FIT v2.60 以降 へ更新することを推奨します。

2.4　サポートされているツールチェーン

BLE FIT モジュールは「9.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.5　ヘッダファイル

すべての API 呼び出しと使用されるインタフェース定義は r_ble_rx23w_if.h に記載しています。

2.6　整数型

このプロジェクトは 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。

2.7　コンパイル時の設定

BLE FIT モジュールのコンフィギュレーションオプションの設定は r_ble_rx23w_config.h で行います。Smart Configurator を使用する場合は、ソフトウェアコンポーネント設定画面でコンフィギュレーションオプションを設定できます。設定値はモジュールを追加する際に、自動的に r_ble_rx23w_config.h に反映されます。オプション名および設定値に関する説明を以下に示します。

表 2.4 : コンフィギュレーションオプション

コンフィギュレーションオプション（r_ble_rx23w_config.h）	
BLE_CFG_LIB_TYPE ※デフォルト値は"0"	Bluetooth LE Protocol Stack のタイプを設定します。 0: All features 1: Balance 2: Compact "0"～"2"の範囲で設定してください。 各タイプがサポートする Bluetooth LE の機能については「2.8 Bluetooth LE Protocol Stack の機能」をご参照ください。
BLE_CFG_RF_DBG_PUB_ADDR ※デフォルト値は "{0xFF,0xFF,0xFF,0x50,0x90,0x74}"	BLE FIT モジュールに設定されるパブリックアドレスの初期値を設定します。このパブリックアドレスはデータフラッシュ、コードフラッシュのパブリックアドレスの領域が ALL 0x00 or 0xFF の場合に採用されます。ただし、設定した値が ALL 0x00 or 0xFF の場合、74:90:50:FF:FF:FF がパブリックアドレスとして採用されます。
BLE_CFG_RF_DBG_RAND_ADDR ※デフォルト値は "{0xFF,0xFF,0xFF,0xFF,0xFF,0xFF} "	BLE FIT モジュールに設定されるスタティックアドレスの初期値を設定します。このオプションのスタティックアドレスはデータフラッシュ、コードフラッシュのスタティックアドレスの領域が ALL 0x00 or 0xFF の場合に採用されます。ただし、設定した値が ALL 0x00 or 0xFF の場合、デバイス固有のスタティックアドレスが採用されます。
BLE_CFG_RF_CONN_MAX ※デフォルト値は"7"	リモートデバイス接続情報管理テーブルの最大エントリ数を設定します。具体的には同時に接続する最大デバイス数、または、保持するボンディング情報の最大数のいずれか大きい方の値を設定します。なお、この値の増減に伴い、Bluetooth LE Protocol Stack が使用するヒープメモリが増減します。 "1"～"7"の範囲で設定してください。
BLE_CFG_RF_CONN_DATA_MAX ※デフォルト値は"251"	最大パケットデータサイズ(バイト)を設定します。 "27"～"251"の範囲で設定してください。
BLE_CFG_RF_ADV_DATA_MAX ※デフォルト値は"1650"	最大 Advertising データサイズ(バイト)を設定します。 "31"～"1650"の範囲で設定してください。 Bluetooth LE Protocol Stack のタイプが"0: All features"以外の場合は"31"固定となります。
BLE_CFG_RF_ADV_SET_MAX ※デフォルト値は"4"	最大 Advertising セット数を設定します。 "1"～"4"の範囲で設定してください。 Bluetooth LE Protocol Stack のタイプが"0: All features"以外の場合は"1"固定となります。
BLE_CFG_RF_SYNC_SET_MAX ※デフォルト値は"2"	最大 Periodic Sync セット数を設定します。 "1"～"2"の範囲で設定してください。 Bluetooth LE Protocol Stack のタイプが"0: All features"以外の場合はこの値は使用しません。

コンフィギュレーションオプション（r_ble_rx23w_config.h）	
BLE_CFG_EVENT_NOTIFY_CONN_START ※デフォルト値は"0"	接続イベントの開始割り込み発生通知機能の有効／無効を設定します。 0: Disable 1: Enable 開始通知は割り込みを契機としていることから、実際の RF イベントの事後通知となります。
BLE_CFG_EVENT_NOTIFY_CONN_CLOSE ※デフォルト値は"0"	接続イベントの完了割り込み発生通知機能の有効／無効を設定します。 0: Disable 1: Enable 各動作の停止コマンドを実行した場合、完了イベントは通知されません。
BLE_CFG_EVENT_NOTIFY_ADV_START ※デフォルト値は"0"	Advertising イベントの開始割り込み発生時通知機能の有効／無効を設定します。 0: Disable 1: Enable 下記のタイミングで通知します。 ➢ Primary Adv Ch の開始 ➢ Secondary Adv Ch(AdvExt)の開始 ➢ Periodic Adv の開始 ※対応する AdvExt が Enable であることが前提 開始通知は割り込みを契機としていることから、実際の RF イベントの事後通知となります。
BLE_CFG_EVENT_NOTIFY_ADV_CLOSE ※デフォルト値は"0"	Advertising イベントの完了割り込み発生時通知機能の有効／無効を設定します。 0: Disable 1: Enable 下記のタイミングで通知します。 ➢ Primary Adv Ch の完了 ➢ Secondary Adv Ch(AdvExt)の完了 ➢ Periodic Adv の完了 ※対応する AdvExt が Enable であることが前提 各動作の停止コマンドを実行した場合、完了イベントは通知されません。
BLE_CFG_EVENT_NOTIFY_SCAN_START ※デフォルト値は"0"	Scan イベントの開始割り込み発生通知機能の有効／無効を設定します。 0: Disable 1: Enable Interval=Window の場合、通知されません。 開始通知は割り込みを契機としていることから、実際の RF イベントの事後通知となります。

コンフィギュレーションオプション（r_ble_rx23w_config.h）	
BLE_CFG_EVENT_NOTIFY_SCAN_CLOSE ※デフォルト値は"0"	Scan イベントの完了割り込み発生通知機能の有効／無効を設定します。 0: Disable 1: Enable Interval=Window の場合、通知されません。 各動作の停止コマンドを実行した場合、完了イベントは通知されません。
BLE_CFG_EVENT_NOTIFY_INIT_START ※デフォルト値は"0"	Initiator イベントの Scan 開始割り込み発生通知機能の有効／無効を設定します。 0: Disable 1: Enable Interval=Window の場合、通知されません。 開始通知は割り込みを契機としていることから、実際の RF イベントの事後通知となります。
BLE_CFG_EVENT_NOTIFY_INIT_CLOSE ※デフォルト値は"0"	Initiator イベントの Scan 完了割り込み発生通知機能の有効／無効を設定します。 0: Disable 1: Enable Interval=Window の場合、通知されません。 各動作の停止コマンドを実行した場合、完了イベントは通知されません。
BLE_CFG_EVENT_NOTIFY_DS_START ※デフォルト値は"0"	RF_DEEP_SLEEP start 通知機能の有効／無効を設定します。 0: Disable 1: Enable
BLE_CFG_EVENT_NOTIFY_DS_WAKEUP ※デフォルト値は"0"	RF_DEEP_SLEEP wakeup 通知機能の有効／無効を設定します。 0: Disable 1: Enable
BLE_CFG_RF_CLVAL ※デフォルト値は"6"	32MHz 水晶振動子の調整値をボード環境に応じて設定します。 "0"～"15"の範囲で設定してください。 関連ドキュメント「RX23W グループ Bluetooth 専用クロック周波数の調整手順」(R01AN4762)を参照してください。 RX23W モジュールの型番 R5F523W8CxLN, R5F523W8DxLN を選択した場合、7 に設定してください。
BLE_CFG_RF_DDC_EN ※デフォルト値は"0"	RF 部の DC-DC の有効／無効を設定します。 0: Disable 1: Enable Target Board をご使用の場合は、0 を設定してください。

コンフィギュレーションオプション（r_ble_rx23w_config.h）	
BLE_CFG_RF_EXT32K_EN ※デフォルト値は"0"	RF 部への slow clock source を設定します。 "0"～"1"の範囲で設定してください。 0:　RF_LOCO 使用 1: 外部 32.768kHz 使用 1 にする場合、Smart Configurator のクロック設定画面(図 4-11)でサブクロックにチェックを入れ、有効にする必要があります。
BLE_CFG_RF_MCU_CLKOUT_PORT ※デフォルト値は"0"	MCU CLKOUT のポートを設定します。 "0"～"1"の範囲で設定してください。 0: PE3 1: PE4 BLE_CFG_RF_EXT32K_EN が 0 の場合、この値は無視されます。
BLE_CFG_RF_MCU_CLKOUT_FREQ ※デフォルト値は"0"	MCU の CLKOUT からの出力周波数を設定します。 "0"～"1"の範囲で設定してください。 0: MCU CLKOUT frequency 32.768kHz 1: MCU CLKOUT frequency 16.384kHz BLE_CFG_RF_EXT32K_EN が 0 の場合、この値は無視されます。
BLE_CFG_RF_SCA ※デフォルト値は"250"	RF slow clock 用の Sleep Clock Accuracy(SCA)を設定します。 "0"～"500"の範囲で設定してください。 BLE_CFG_RF_EXT32K_EN が 0 の場合、SCA は 250ppm 以上に固定され、この値は無視されます。
BLE_CFG_RF_MAX_TX_POW ※デフォルト値は"1"	最大送信パワーを設定します。 "0"～"1"の範囲で設定してください。 0: max +0dBm 1: max +4dBm
BLE_CFG_RF_DEF_TX_POW ※デフォルト値は"0"	デフォルトの送信パワーを設定します。 "0"～"2"の範囲で設定してください。 デフォルト送信パワーは BLE_CFG_RF_MAX_TX_POW の設定に依存します。 BLE_CFG_RF_MAX_TX_POW が 0(0dBm)の場合、 BLE_CFG_RF_DEF_TX_POW は以下のようになります。 0(High) : 0dBm 1(Mid) : 0dBm 2(Low) : -18dBm BLE_CFG_RF_MAX_TX_POW が 1(+4dBm)の場合、 BLE_CFG_RF_DEF_TX_POW は以下のようになります。 0(High) : +4dBm 1(Mid) : 0dBm 2(Low) : -20dBm

コンフィギュレーションオプション（r_ble_rx23w_config.h）	
BLE_CFG_RF_CLKOUT_EN ※デフォルト値は"0"	CLKOUT_RF 出力を設定します。 以下のいずれかの値を選択してください。 0: No output 5: 4MHz output 6: 2MHz output 7: 1MHz output
BLE_CFG_RF_DEEP_SLEEP_EN ※デフォルト値は"1"	RF Deep Sleep 機能の有効／無効を設定します。 0: Disable 1: Enable
BLE_CFG_MCU_MAIN_CLK_KHZ ※デフォルト値は"4000"	MCU メインクロック周波数(kHz)を設定します。 ボード環境に合わせて設定してください。 HOCO の場合、この値による設定は無視されます。 メインクロックの場合、"1000"から"20000"の範囲で設定してください。 PLL Circuit の場合、"4000"から"20000"の範囲で設定してください。 Smart Configurator の”クロック”画面で入力した周波数を設定してください。
BLE_CFG_DEV_DATA_CF_BLOCK ※デフォルト値は"16"	デバイス固有データを格納する Code Flash(ROM)の Block を設定します。 "-1"～"255"の範囲で設定してください。 "-1"が設定された場合、Code Flash のデバイス固有データを使用しません。 "0"から"15"は Start-Up Program Protection block となっていますので、Start-Up Program Protection 機能を使用する場合は、”0”から”15”を指定しないでください。
BLE_CFG_DEV_DATA_DF_BLOCK ※デフォルト値は"-1"	デバイス固有データを格納する Data Flash の Block を設定します。 "-1"～"7"の範囲で設定してください。 "-1"が設定された場合、E2 Data Flash のデバイス固有データを使用しません。 BLE_CFG_SECD_DATA_DF_BLOCK で指定した Block と別の Block を指定してください。
BLE_CFG_GATT_MTU_SIZE ※デフォルト値は"247"	GATT 通信で使用する MTU サイズ(バイト)を設定します。 "23"～"247"の範囲で設定してください。リモートデバイスから MTU 交換要求を受信すると、プロファイル共通部の中で BLE_CFG_GATT_MTU_SIZE の値を返します。
BLE_CFG_NUM_BOND ※デフォルト値は"7"	セキュリティデータ管理機能で保存するボンディング情報の数を設定します。 "1"～"7"の範囲で設定してください。 この値を一度決定してコード生成した後、デバッグ等で値を変更すると、変更前にセキュリティデータ管理機能によって管理されていたボンディング情報が Data Flash に残り、再ペアリングができない場合があります。これを避けるため、ファームウェア書き込み後、R_BLE_GAP_DeleteBondInfo()、または”gap auth del remote all” コマンドにより、ボンディング情報を削除してください。
BLE_CFG_EN_SEC_DATA ※デフォルト値は"0"	セキュリティデータ管理機能を有効にします。この機能はペアリング成功時に BLE_CFG_SECD_DATA_DF_BLOCK で示す Block にボンディング情報を保存します。 0: Disable 1: Enable この機能を有効にする場合、Data Flash モジュールのコンポーネントを有効にしてください。

コンフィギュレーションオプション（r_ble_rx23w_config.h）	
BLE_CFG_SECD_DATA_DF_BLOCK ※デフォルト値は"0"	セキュリティデータ管理機能で Data Flash へのボンディング情報の保存機能で使用する Data Flash Block を設定します。 "0"～"7"の範囲で設定してください。 BLE_CFG_DEV_DATA_DF_BLOCK で指定した Block と別の Block を指定してください。
BLE_CFG_CMD_LINE_EN ※デフォルト値は"0"	コマンドライン機能の有効／無効を設定します。 0: Disable 1: Enable この機能を有効にする場合、SCI FIT モジュールのコンポーネントを有効にしてください。
BLE_CFG_CMD_LINE_CH ※デフォルト値は"1"	コマンドライン機能で使用する SCI のチャネルを設定します。 SCI FIT モジュールの設定にて、コマンドライン機能が使用する SCI のチャネルを有効にしてください。 BLE_CFG_CMD_LINE_EN が 0 の場合、この値は無視されます。
BLE_CFG_BOARD_LED_SW_EN ※デフォルト値は"0”	ボードの LED と SW の制御コードの有効／無効を設定します。 0: Disable 1: Enable この機能を有効にする場合、IRQ FIT モジュールと GPIO FIT モジュールのコンポーネントを有効にしてください。
BLE_CFG_BOARD_TYPE ※デフォルト値は”0”	ボードのタイプを設定します。 "0"～"3"の範囲で設定してください。 0 : Customer board 1 : Target Board 2 : RSSK 3 : Evaluation Board
BLE_CFG_LOG_LEVEL ※デフォルト値は"3"	ログレベルを設定します。 "0"～"3"の範囲で設定してください。 0 : disable 1 : Error 2 : Error & Warning 3 : Error & Warning & Debug
BLE_CFG_ABS_API_EN ※デフォルト値は"1"	抽象 API の有効／無効を設定します。 0: Disable 1: Enable
BLE_CFG_SOFT_TIMER_EN ※デフォルト値は"1"	app_lib が提供するソフトウェアタイマの有効／無効を設定します。 0: Disable 1: Enable 抽象 API を使用する場合は、この機能を有効に設定してください。

コンフィギュレーションオプション（r_ble_rx23w_config.h）	
BLE_CFG_MCU_LPC_EN ※デフォルト値は"1"	MCU の低消費電力機能の有効／無効を設定します。 0: Disable 1: Enable
BLE_CFG_HCI_MODE_EN ※デフォルト値は"0"	HCI モードでの起動を設定します。 0: 通常起動 1: HCI モードでの起動

2.8 Bluetooth LE Protocol Stack の機能

BLE FIT モジュールの Bluetooth LE Protocol Stack が提供する機能はコンフィギュレーションファイル (r_ble_rx23w_config.h)の BLE_CFG_LIB_TYPE の設定値により変わります。表 2.5 に Bluetooth LE Protocol Stack の各タイプがサポートする機能を示します。各機能の説明については「Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205) 3.1.2 Bluetooth LE Protocol Stack ライブラリ」をご参照ください。

表 2.5 : Bluetooth LE Protocol Stack の各タイプがサポートする機能

Bluetooth LE Feature	Bluetooth LE Protocol Stack のタイプ		
	All features	Balance	Compact
LE 2M PHY	Yes	Yes	No
LE Coded PHY	Yes	Yes	No
LE Advertising Extensions	Yes	No	No
LE Channel Selection Algorithm #2	Yes	Yes	No
High Duty Cycle Non-Connectable Advertising	Yes	Yes	Yes
LE Secure Connections	Yes	Yes	Yes
Link Layer privacy	Yes	Yes	Yes
Link Layer Extended Scanner Filter policies	Yes	Yes	No
LE Data Packet Length Extension	Yes	Yes	Yes
LE L2CAP Connection Oriented Channel Support	Yes	No	No
Low Duty Cycle Directed Advertising	Yes	Yes	Yes
LE Link Layer Topology	Yes	Yes	No
LE Ping	Yes	Yes	Yes
GAP Role	Central Peripheral Observer Broadcaster	Central Peripheral Observer Broadcaster	Peripheral Broadcaster
GATT Role	Sever Client	Sever Client	Sever Client
32-bit UUID Support in LE	Yes	Yes	Yes

2.9　app_lib の機能

Bluetooth LE アプリケーションが利用可能な補助機能を app_lib で提供します。
app_lib が提供する機能について以下に説明します。

2.9.1　抽象 API

抽象 API は Bluetooth LE Protocol Stack でよく使う機能をより簡単に使うための API です。抽象 API の詳細な仕様については R_BLE API ドキュメント(r_ble_api_spec.chm)をご参照ください。この機能を使用する場合、BLE FIT モジュールのコンポーネント設定から BLE_CFG_ABS_API_EN と BLE_CFG_SOFT_TIMER_EN を”1”に設定してください。
抽象 API のコードは変更しないでください。

2.9.2　ソフトウェアタイマ

ソフトウェアタイマはベアメタル版と Renesas FreeRTOS 版で実装が異なります。

[ベアメタル版]
ソフトウェアタイマは MCU のコンペアマッチタイマ(CMT)を使用します。ソフトウェアタイマを使用する場合、CMT FIT モジュールをアプリケーションに組み込んでください。使用する CMT のチャネルは FIT により、動的に割り当てられます。

[Renesas FreeRTOS 版]
ソフトウェアタイマは FreeRTOS の API を使用します。

この機能を使用する場合、BLE FIT モジュールのコンポーネント設定から BLE_CFG_SOFT_TIMER_EN を”1”に設定してください。

2.9.3　セキュリティデータ管理

ペアリング成功時に Data Flash にボンディング情報を自動的に保存するためのインタフェースを提供します。この機能を使用する場合、BLE FIT モジュールのコンポーネント設定から BLE_CFG_EN_SEC_DATA を”1”に設定してください。使用する Data Flash のブロックは BLE_CFG_SECD_DATA_DF_BLOCK で指定します。

2.9.4　プロファイル共通部

プロファイル共通部は Bluetooth LE Profiles で共通する処理です。QE for BLE が生成したプロファイルのソースコードからコールされます。プロファイル共通部、および、プロファイル開発の詳細については、「Bluetooth Low Energy プロファイル開発者ガイド(R01AN6459)」をご参照ください。
BLE FIT v2.50 以降はプロファイル共通を含みません。QE for BLE(V1.6.0 以降)でのコード生成時にプロジェクトに追加されます。

2.9.5　ロガー

メッセージを出力します。出力レベルはコンフィギュレーションオプションの BLE_CFG_LOG_LEVEL で設定します。CCRX の C++をご使用される場合、ロガーはサポート範囲外となります。

2.9.6　コマンドライン

コマンドラインはターミナルソフトから入力したコマンドを介して Bluetooth LE 機能を使用する機能です。本機能は MCU のシリアルコミュニケーションインタフェース(SCI)を使用します。

1)コンポーネントの設定

コマンドラインを使用する場合、SCI FIT モジュールをアプリケーションに組み込んでください。
使用するチャネルは以下の手順で設定します。

1. Smart Configurator から SCI FIT モジュールを組み込み、コンポーネント設定から使用するチャネルを選択します。
2. BLE_CFG_CMD_LINE_EN オプションの値を”1”にします。
3. BLE_CFG_CMD_LINE_CH オプションに SCI FIT モジュールのコンポーネント設定で有効にしたチャネルを指定します。

2)ターミナルソフトの設定

ボードと接続するコンピュータのターミナルソフトに以下の項目を設定します。

表 2.6 : ターミナルソフトの設定

項目	設定
New-line (Receive)	LF
New-line (Transmit)	CR
Terminal Mode	VT100
Baud rate	115200
Data	8bit
Parity	none
Stop bits	1bit
Flow Control	none

3)サポートするコマンド
コマンドライン機能で使用可能なコマンドを表 2.7 に示します。

表 2.7 : サポートするコマンド一覧

コマンド名	サブコマンド名	コマンドの内容
gap	adv	Advertising を開始します。
	scan	Scan を開始します。
	conn	接続します。
	disconn	切断します。
	device	接続中のデバイスを表示します。
	priv	ローカルデバイスのプライバシー機能を ON にします。
	conn_cfg	接続の設定を行います。
	wl	White List にリモートデバイスを登録します。
	auth	ペアリング／暗号化を行います。
	sync	Periodic Sync を確立します。
	ver	バージョン情報を表示します。
vs	txp	送信パワーの設定、取得を行います。
	scheme	Coded PHY の Coding Scheme を設定します。
	test	DTM の操作を行います。
	addr	BD_ADDR の設定、取得を行います。
	rand	乱数を生成します。
sys	stby	ソフトウェアスタンバイモードの操作を行います。
ble	reset	Bluetooth LE Protocol Stack のリセットを行います。
	close	Bluetooth LE Protocol Stack を停止します。

gap コマンドや vs コマンドを使用する場合、BLE FIT モジュールのコンポーネント設定から BLE_CFG_ABS_API_EN を”1”に設定してください。

各コマンドの詳細な仕様については、「Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205)　5.1 コマンドライン」をご参照ください。

2.9.7　LED and Switch 制御

ボード上の LED, SW の制御を行う機能です。LED and Switch 制御を行うための BLE, IRQ FIT のコンフィギュレーションオプションの設定については「4.5.4 LED and Switch 制御機能」を参照してください。

2.9.8　BLE タスク制御

プロジェクト生成時に RTOS に[FreeRTOS]を選択した場合に有効となります。他のタスクから BLE タスク制御を行うときに使用します。

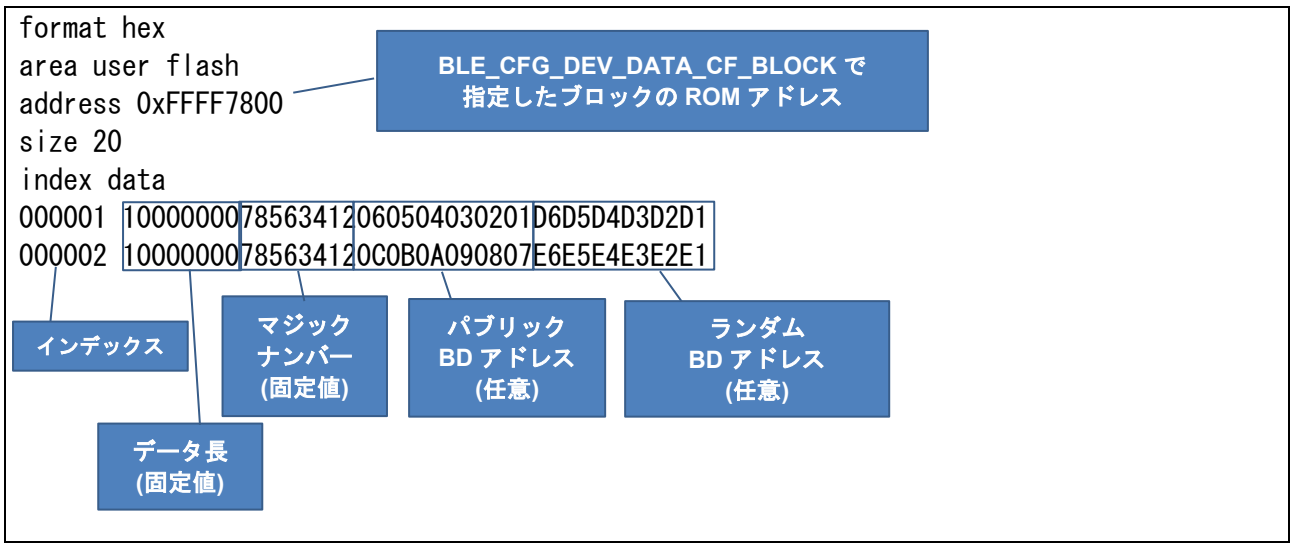
2.10 デバイス固有データ

BLE FIT が使用する Bluetooth Device Address(以降は BD アドレスと記載)はフラッシュメモリのユーザ領域(ROM)またはデータ領域(E2 データフラッシュ)に書き込むことができます。
書き込み先はコンフィギュレーションオプション（r_ble_rx23w_config.h）の
BLE_CFG_DEV_DATA_CF_BLOCK および BLE_CFG_DEV_DATA_DF_BLOCK で指定できます。
ユーザ領域(ROM)に書き込む場合、プログラムコードでは使用しないブロックを指定する必要があります。
また、指定したブロックの先頭アドレスに書き込む必要があります。
書き込むデータのフォーマットを表 2.8 に記載します。

表 2.8 : ユーザ領域(ROM)に書き込むデータのフォーマット

オフセット	サイズ[byte]	概要
0	4　(uint32_t 型)	マジックナンバー以降のデータ長(16 固定)
4	4　(uint32_t 型)	マジックナンバー(0x12345678 固定)
8	6　(uint8_t[6]型)	パブリック BD アドレス
14	6　(uint8_t[6]型)	ランダム BD アドレス

データは各ブロックごとにリトルエンディアンで書き込む必要があります。
データの書き込みは Renesas Flash Programmer(RFP)のユニークコード機能を使用することで複数の RX23W に異なる BD アドレスを連続して書き込むことができます。
以下に RFP のユニークコードファイル(ruc)の設定例を記載します。



なお、BD アドレスは以下の順番で検索し、全て 0xFF または 0x00 でない BD アドレスを採用します。

(1) データ領域(E2 データフラッシュ)の指定ブロック
(2) ユーザ領域(ROM)の指定ブロック
(3) ファームウェア初期値
(BLE_CFG_RF_DBG_PUB_ADDR または BLE_CFG_RF_DBG_RAND_ADDR)

RX23W のフラッシュメモリは、デフォルトではフラッシュメモリプロテクト機能が無効状態となっています。フラッシュメモリプロテクト機能が無効の場合、Renesas Flash Programmer(RFP)などを使用してシリアルプログラマ接続を行った際にすべてのブロックを消去します。
そのため、書き込んだ BD アドレスを保持したまま新たなファームウェアを書き込むためにはフラッシュメモリプロテクト機能を有効にする必要があります。
プロテクト機能を有効にするために、以下の章に記載した r_bsp のコンフィギュレーションオプションとデバッグの ID Code の設定を行ってください。

- 4.5.1 BSP(r_bsp)
- 4.8.1 Flash の ID Code

2.11 コードサイズ

表 2.9 に BLE FIT モジュールの Bluetooth LE Protocol Stack のコードサイズを示します。 表 2.9 は「9.1 動作確認環境」で記載したツールチェーンを使用し、最適化レベル 2、およびコードサイズ重視の最適化を前提としたサイズとなります。

BLE FIT モジュールの Bluetooth LE Protocol Stack の ROM と RAM のサイズは、コンフィギュレーションヘッダファイル内の設定値により決まります。コードサイズを算出時のコンフィギュレーションオプションの設定値は表内に記載しています。

表 2.9 : Bluetooth LE Protocol Stack のコードサイズ

デバイス	コンパイラ	分類	Bluetooth LE Protocol Stack のタイプ		
			All Features	Balance	Compact
RX23W	CC-RX	ROM	191K バイト	149K バイト	133K バイト
		RAM	38K バイト	23K バイト	23K バイト

コンフィギュレーションオプション

共通 :

BLE_CFG_RF_CONN_MAX 7

BLE_CFG_RF_CONN_DATA_MAX 251

BLE_CFG_NUM_BOND 7

All Features :

BLE_CFG_LIB_TYPE 0

BLE_CFG_RF_ADV_DATA_MAX 1650

BLE_CFG_RF_ADV_SET_MAX 4

BLE_CFG_RF_SYNC_SET_MAX 2

Balance :

BLE_CFG_LIB_TYPE 1

Compact :

BLE_CFG_LIB_TYPE 2

[Note]
上記のコードサイズには app_lib と Bluetooth LE Profiles は含まれていません。
CCRX、IAR の開発環境共に CCRX でコンパイルした Bluetooth LE Protocol Stack ライブラリを使用します。

2.12 FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、スマート・コンフィグレータを使用した(1)、(3)、(5)の追加方法を推奨しています。ただし、スマート・コンフィグレータは、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1) e^2 studio 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
e^2 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、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: e^2 studio 編 (R20AN0451)」を参照してください。

(2) e^2 studio 上で FIT コンフィグレータを使用して FIT モジュールを追加する場合
e^2 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、アプリケーションノート「RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3) CS+上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
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、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」を参照してください。

(4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「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) IAREW 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
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、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.13 for 文、while 文、do while 文について

FIT モジュールでは、レジスタの反映待ち処理等で for 文、while 文、do while 文（ループ処理）を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合は、「WAIT_LOOP」で該当の処理を検索できます。

以下に記述例を示します。

```
while 文の例：
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for 文の例：
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while 文の例：
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. R_BLE API 関数

API 関数の詳細については R_BLE API ドキュメント(r_ble_api_spec.chm)をご参照ください。

r_ble_api_spec.chm ファイルが開けない場合、ファイルを右クリックし、プロパティ選択後、[許可する]をチェックしてください。

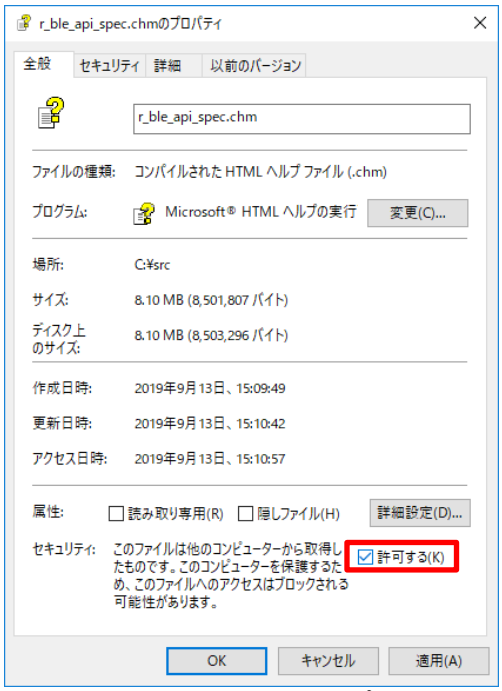


図 3-1 : r_ble_api_spec.chm のプロパティ画面

4. BLE FIT モジュールのプロジェクト

本章は e² studio 上で Smart Configurator により、新規プロジェクトから BLE FIT モジュールを使用して、アプリケーション開発環境を構築する方法について説明します。

4.1 新規プロジェクトの作成

[ファイル]メニューから[C/C++ Project]を選択します。[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、[Next]ボタンをクリックします。

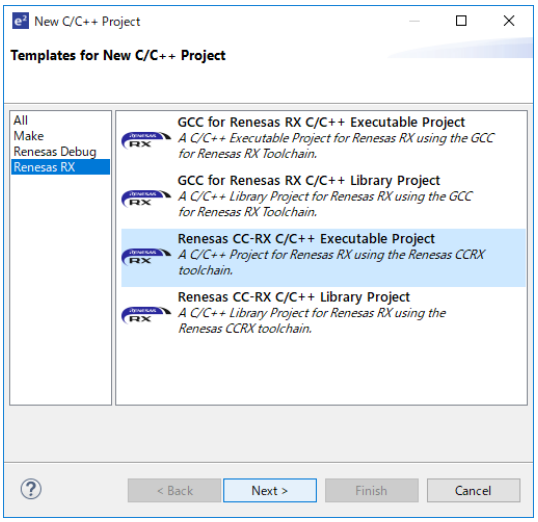


図 4-1： プロジェクトテンプレート選択画面

プロジェクト名を入力し、[Next]ボタンをクリックします。

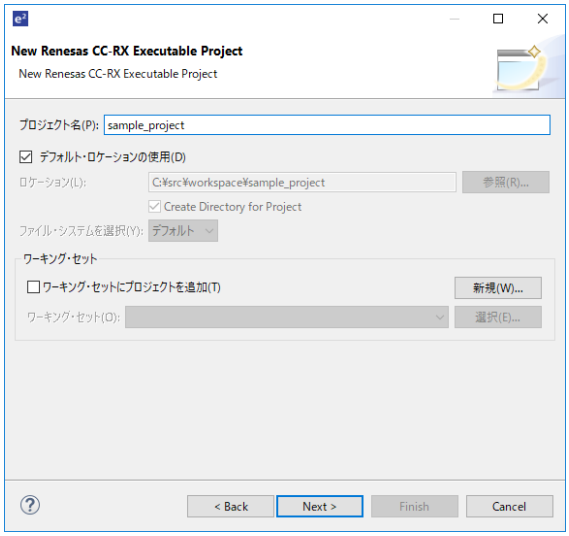


図 4-2： 新規プロジェクト設定画面

[Device Settings]のエンディアンを[Little]にします。
ターゲットデバイスは[RX200]→[RX23W]から使用する board に合わせて、RX23W のデバイスの型名を選択します。

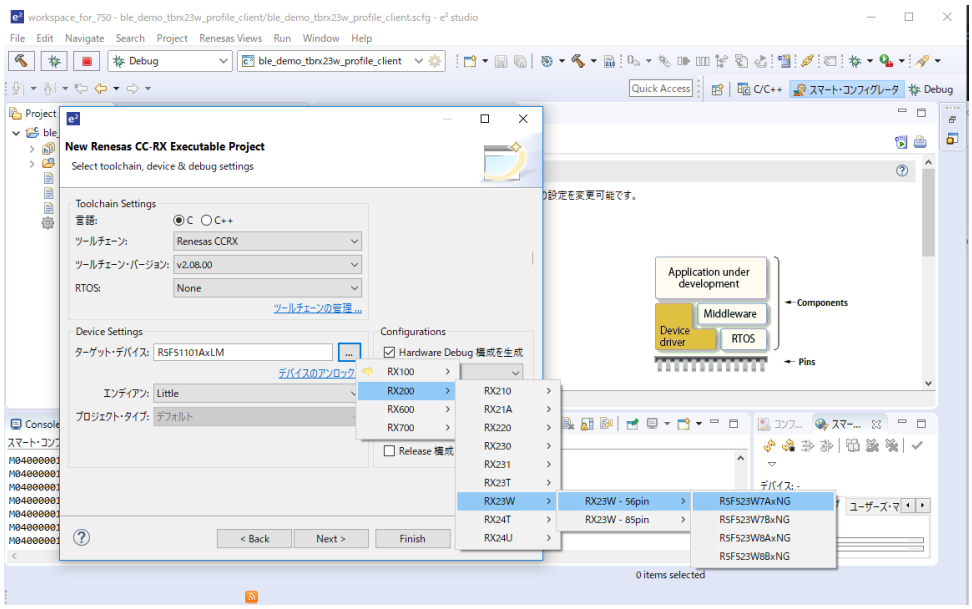


図 4-3 : ツールチェーン、デバイス、デバッグ設定画面

RX23W の型名とメモリサイズ・パッケージについて図 4-4 に示します。

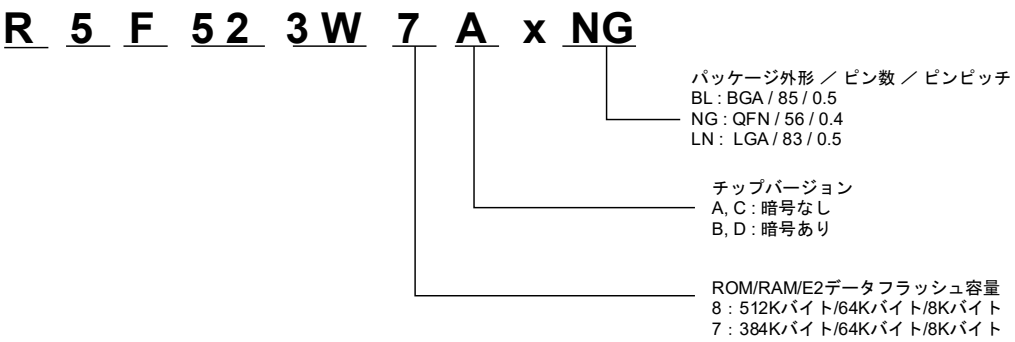


図 4-4 : RX23 の型名とメモリサイズ・パッケージ

RX23W の型名の詳細な説明については関連ドキュメント「 RX23W グループユーザーズマニュアル　ハードウェア編(R01UH0823)」の「1.2 製品一覧」をご参照ください。

Renesas FreeRTOS を使用する場合、RTOS に[FreeRTOS (kernel only)]を選択し、RTOS Version に使用するバージョンを指定します。

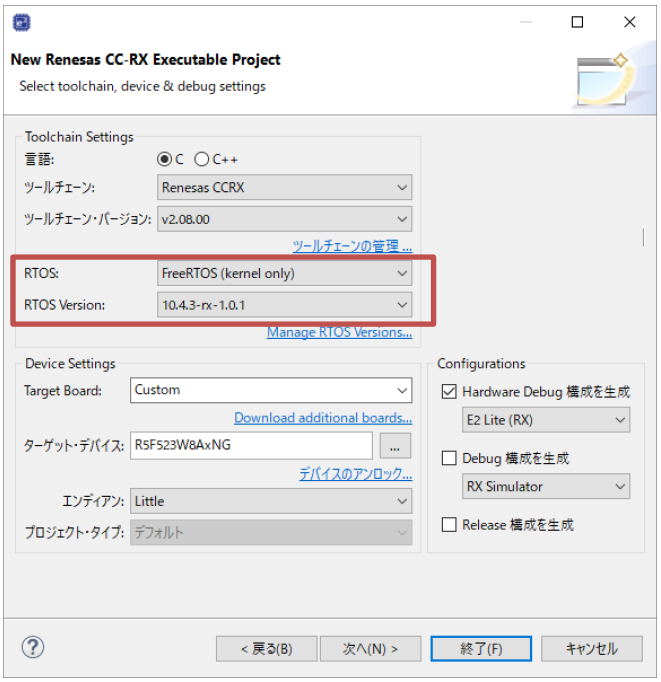


図 4-5 : RTOS と RTOS version

最新の Renesas FreeRTOS については'Manage RTOS Versions…"を押し、"RTOS Module Download"画面からダウンロード可能です。

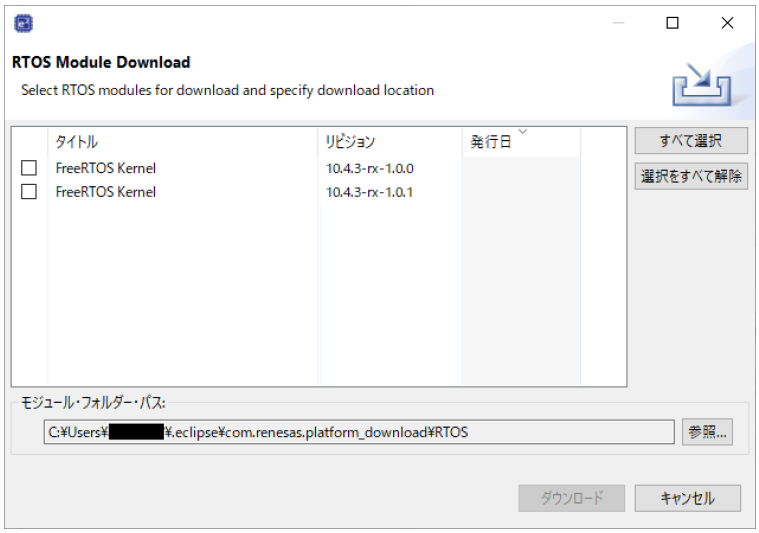


図 4-6 : Renesas FreeRTOS のダウンロード画面

RTOS を使用しない場合、RTOS に[None]を選択します。

[次へ]ボタンを押すと、[コーディング・アシストツールの選択]ダイアログが表示されます。

[コーディング・アシストツールの選択]ダイアログでは、[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、[Finish]ボタンをクリックします。

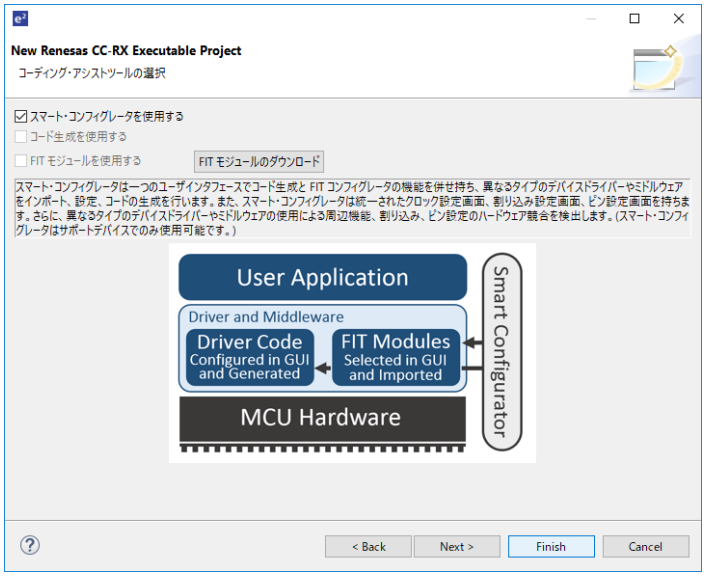


図 4-7 : コーディング・アシストツールの選択画面

しばらくすると、プロジェクトが生成されます。

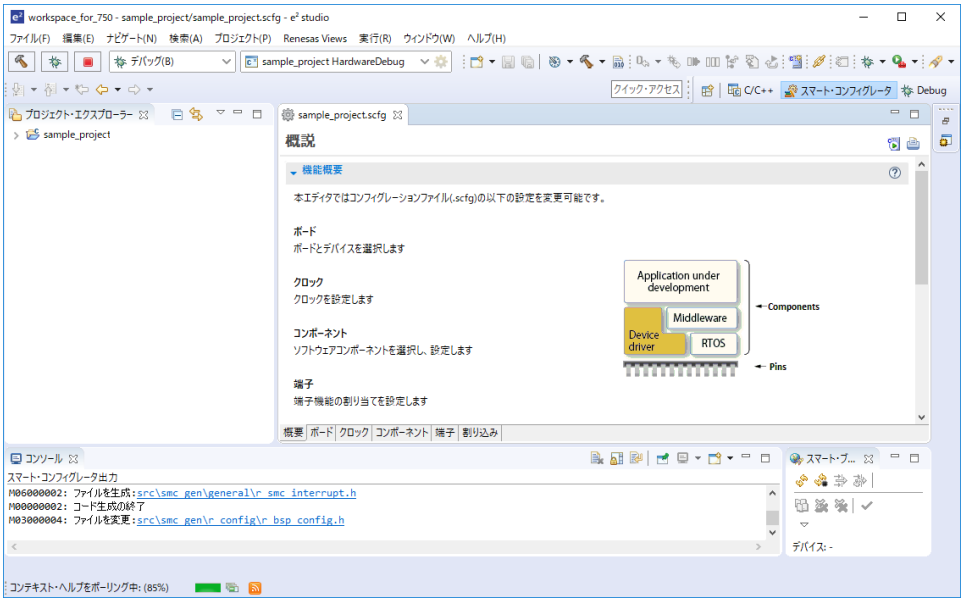


図 4-8 : プロジェクト生成後の画面

Smart Configurator を使用した、e² studio での新規プロジェクト作成の詳細については、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」をご参照ください。

4.2　BSP バージョン確認

プロジェクト作成後にソフトウェアコンポーネント設定画面のコンポーネントタブから r_bsp を右クリックし、[バージョン変更]を選択します。

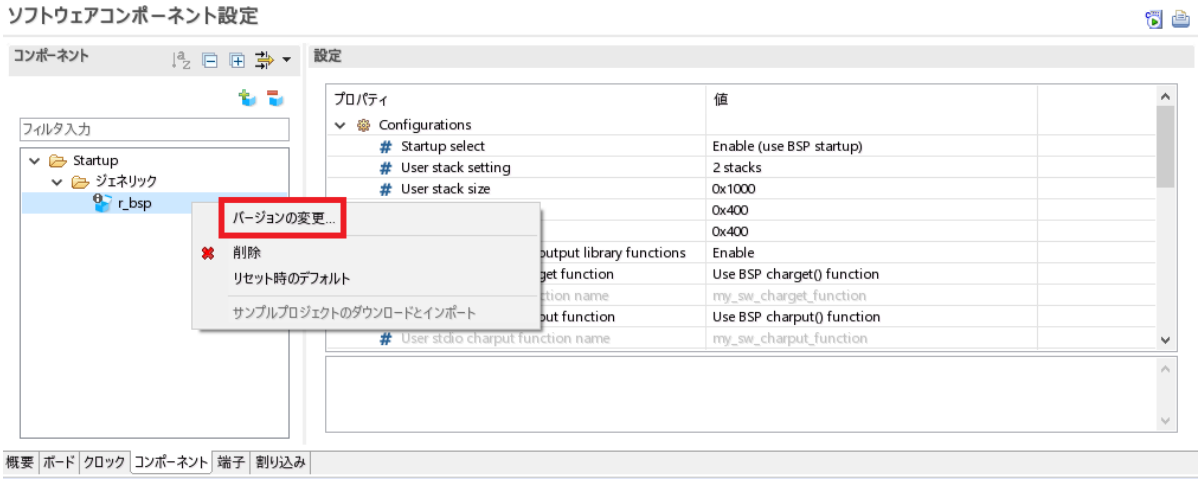


図 4-9: バージョンの変更

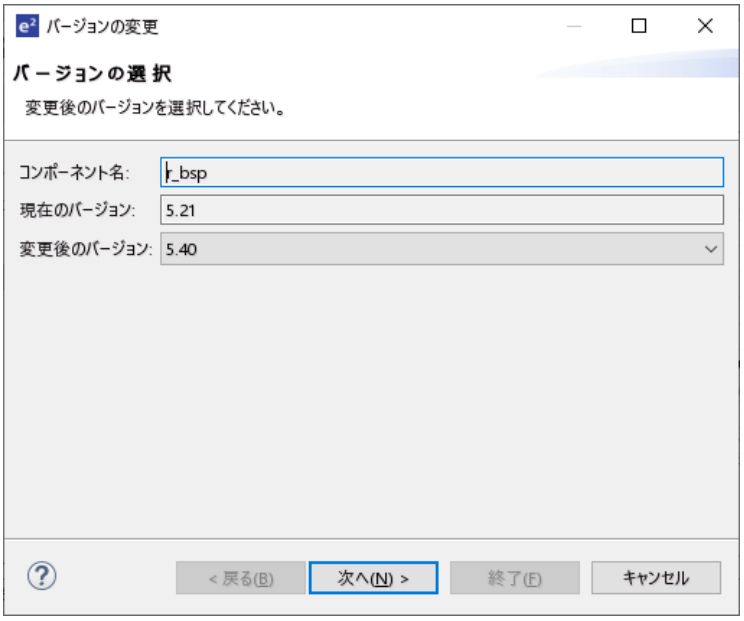


図 4-10: バージョンの変更ダイアログ

バージョンの変更ダイアログで表示している r_bsp の[現在のバージョン]をご確認ください。
BLE FIT モジュールのバージョン 1.01 以降を使用する場合、r_bsp のバージョンを 5.40 以降への変更が必要となります。この変更を行わない場合、Target Board にて SCI を使用すると、ビルド時に以下のエラーが発生します。

../src/smc_gen/r_sci_rx/src/targets/rx23w/r_sci_rx23w_data.c(153):E0520020:Identifier "IR_SCI8_ERI8" is undefined

RX23W の型番として、R5F523W8CxLN, R5F523W8DxLN を選択した場合、BSP v5.62 以降をご使用ください。

4.3　クロック設定

プロジェクト内の scfg ファイルをクリックし、Smart Configurator の[クロック]タブ上でクロック設定を行います。BLE を使用する場合、下記の範囲となるように、クロックの選択、周波数の設定を行ってください。

- 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

Bluetooth LE Protocol Stack は ICLK, PCLKB 周波数 32MHz で最適化を行っています。
BLE の性能を引き出すために ICLK, PCLKB の周波数が 32MHz となるように、クロックの設定を行うことを推奨します。

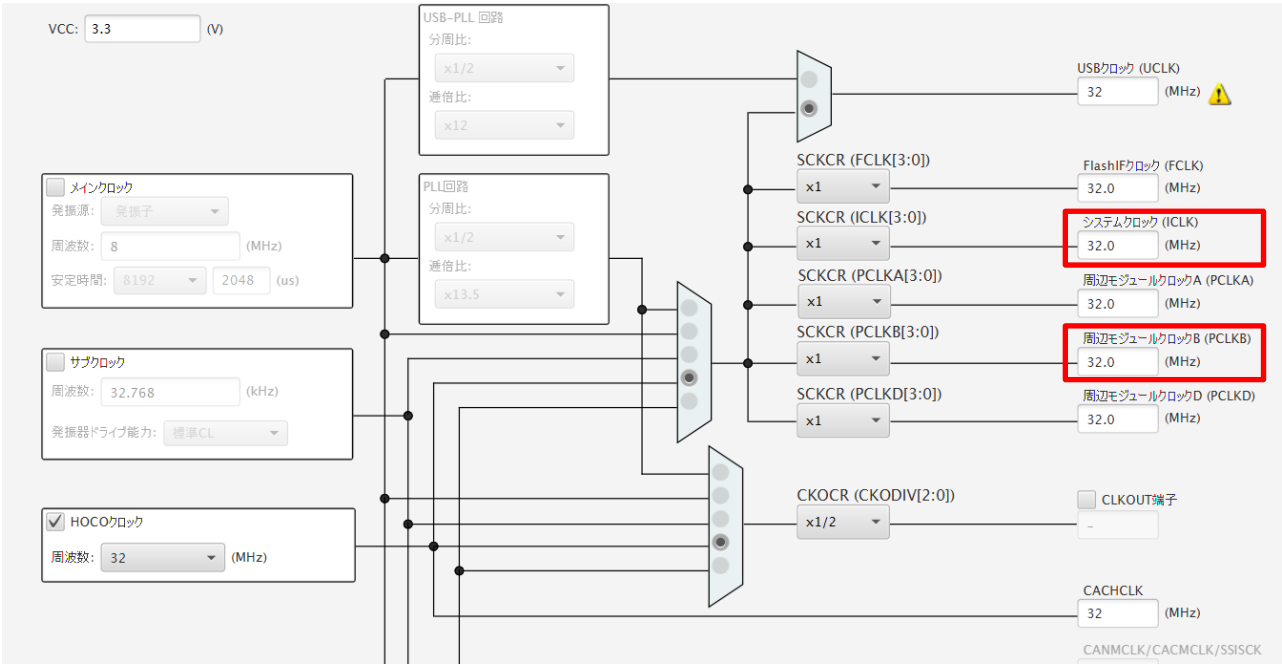


図 4-11: クロック設定画面

4.4　コンポーネントの追加

プロジェクト内の scfg ファイルをクリックし、Smart Configurator の[コンポーネント]タブ上で Bluetooth LE アプリケーションに必要なコンポーネントの追加を行います。

「2.2 ソフトウェア」に記載した、

- コマンドライン機能
- セキュリティデータ管理機能
- LED and Switch 制御機能

を使用する場合を例として説明します。

[コンポーネントの追加]ボタン をクリックし、
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。

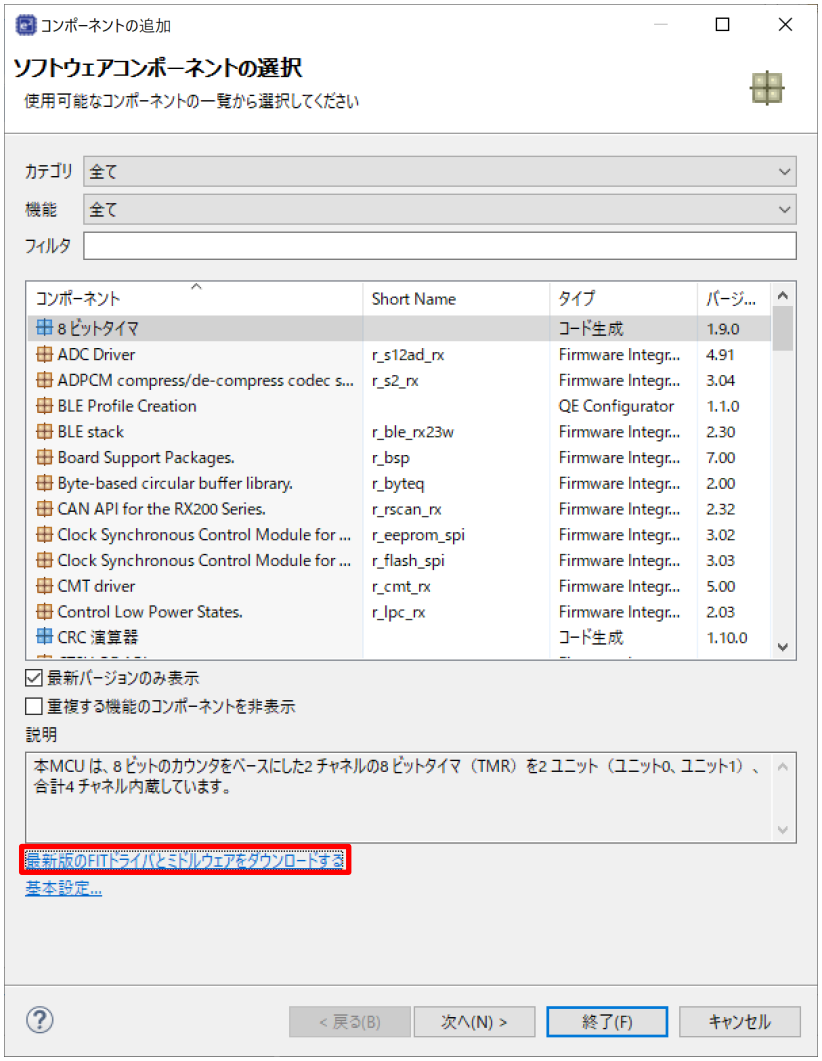


図 4-12：ソフトウェアコンポーネントの選択画面

注 : 必要な FIT モジュールが見つからない場合は、[最新版の FIT ドライバとミドルウェア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。

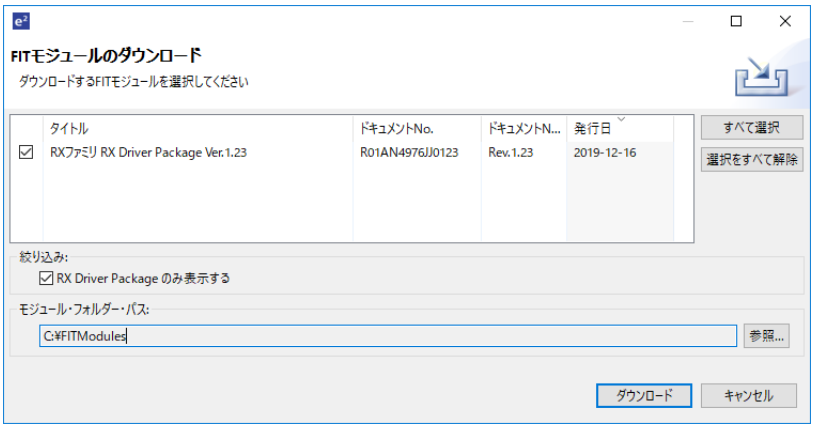


図 4-13 : FIT モジュールのダウンロード

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、[ソフトウェアコンポーネントの選択]画面の[重複する機能のコンポーネントを非表示]のチェックを外します。

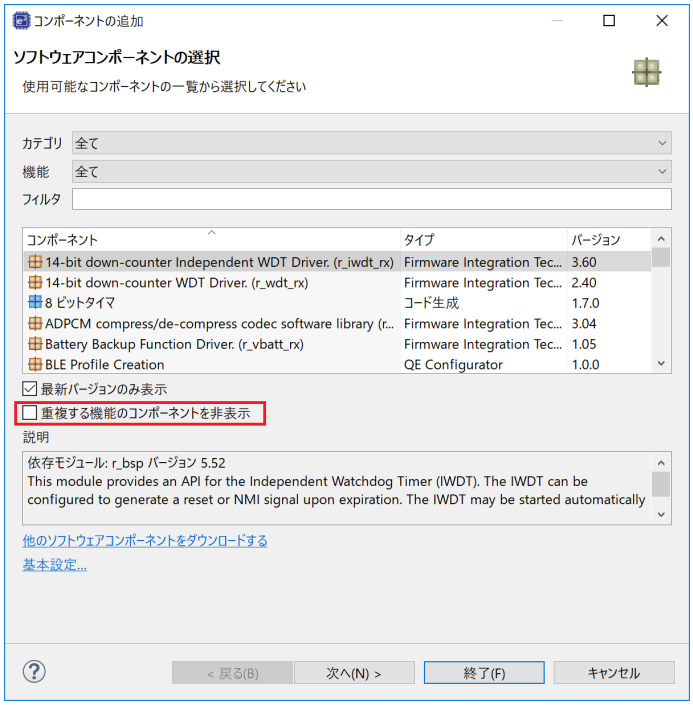


図 4-14 : ソフトウェアコンポーネントの選択画面

しばらくすると、選択したコンポーネントが追加されます。

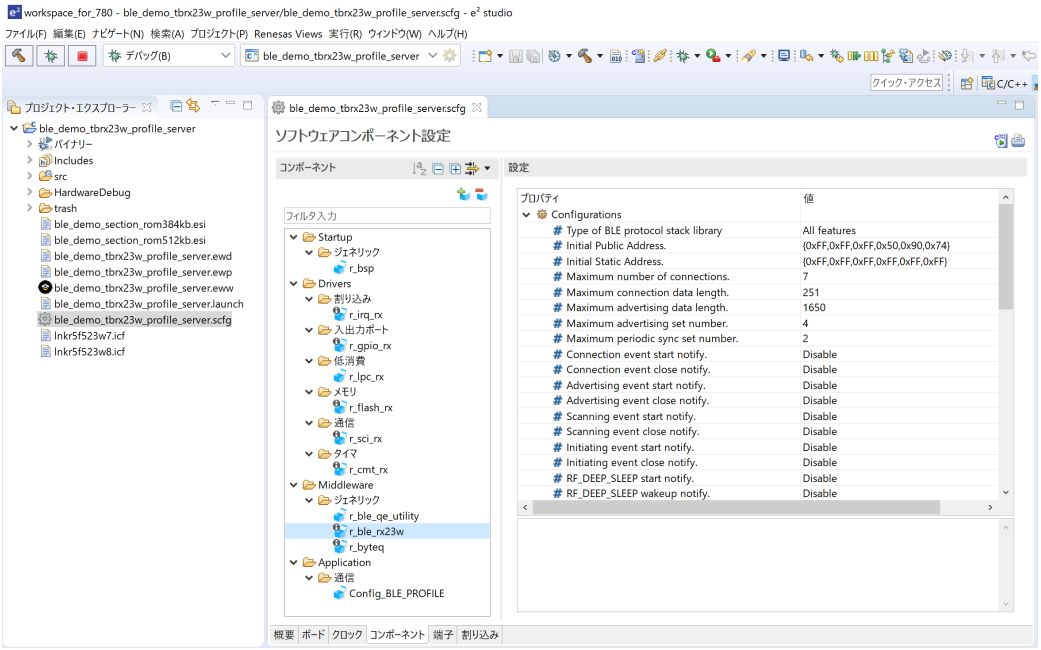


図 4-15 : ソフトウェアコンポーネント設定画面

4.5　コンフィギュレーションオプションの設定

コンポーネント追加後、使用する BLE FIT の機能に応じて、各 FIT モジュールのコンフィギュレーションオプションを設定します。

4.5.1　BSP(r_bsp)

「2.10 デバイス固有データ」に記載したフラッシュメモリプロテクト機能を有効にするために ID Code の設定を行います。Smart Configurator の[コンポーネント]タブから[r_bsp]を選択し、[ID code 1]コンフィギュレーションオプションを”0x45FFFFFF”に設定します。

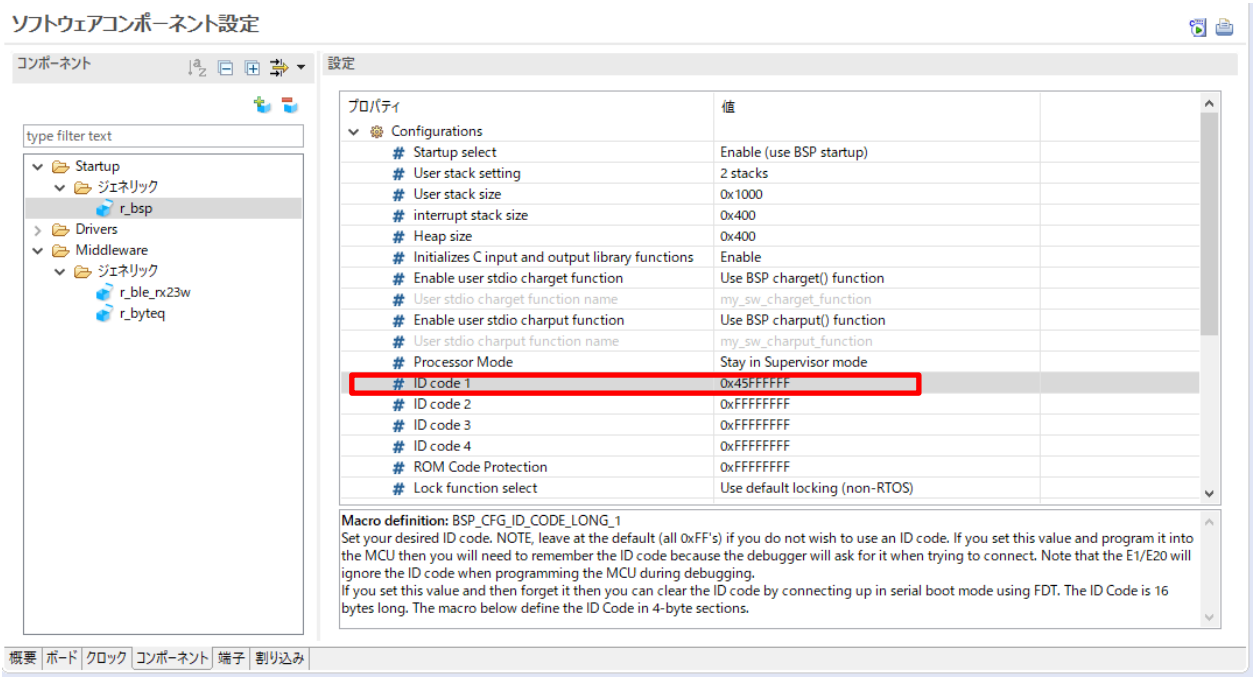


図 4-16 : BSP のコンフィギュレーションオプションの設定画面

4.5.2　コマンドライン機能

BLE(r_ble_rx23w), SCI(r_sci_rx)のコンフィギュレーションオプションの設定を行います。byte queues/circular buffers (r_byteq_rx)については設定は必要ありません。

(1) BLE(r_ble_rx23w)
Smart Configurator の[コンポーネント]タブから[r_ble_rx23w]を選択し、[Enabled/Disabled command line function]コンフィギュレーションオプションを”Enabled”に設定します。
使用する SCI のチャネル番号は[SCI CH for command line function] コンフィギュレーションオプションに設定します。Target Board(TB)と RSSK の場合、SCI8 を使用するため、”8”に設定します。

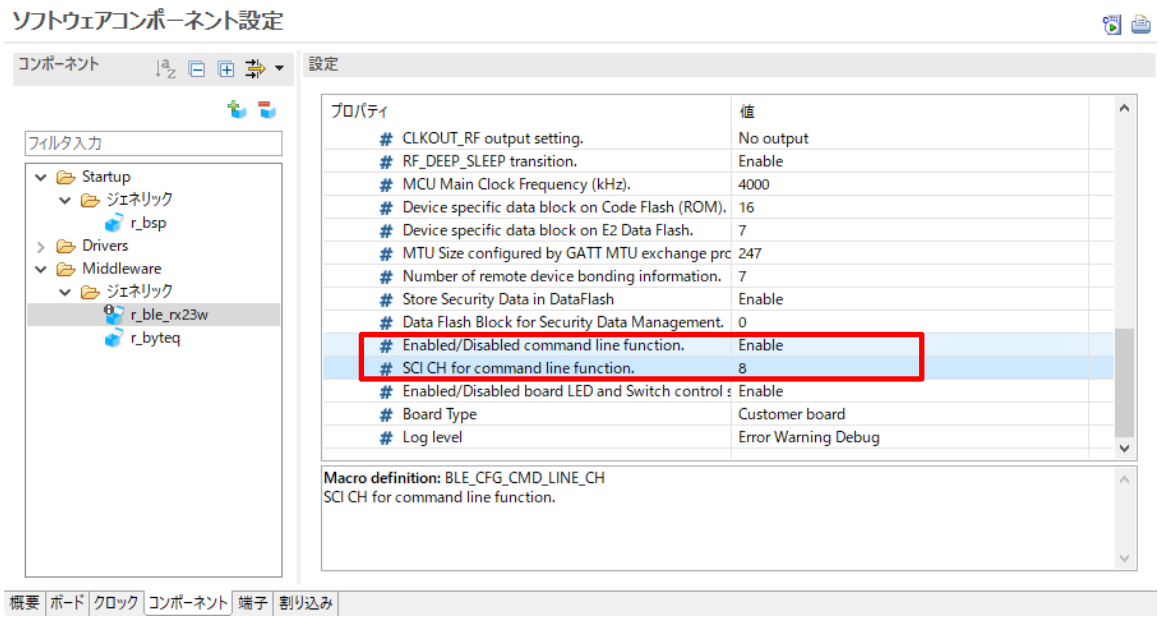


図 4-17: BLE のコンフィギュレーションオプションの設定画面(1)

(2) SCI(r_sci_rx)
Smart Configurator の[コンポーネント]タブから[r_sci_rx]を選択し、以下のオプションを設定します。

- 使用するチャネル番号の[Include software support for channel n]コンフィギュレーションオプションを”Include”に設定します。

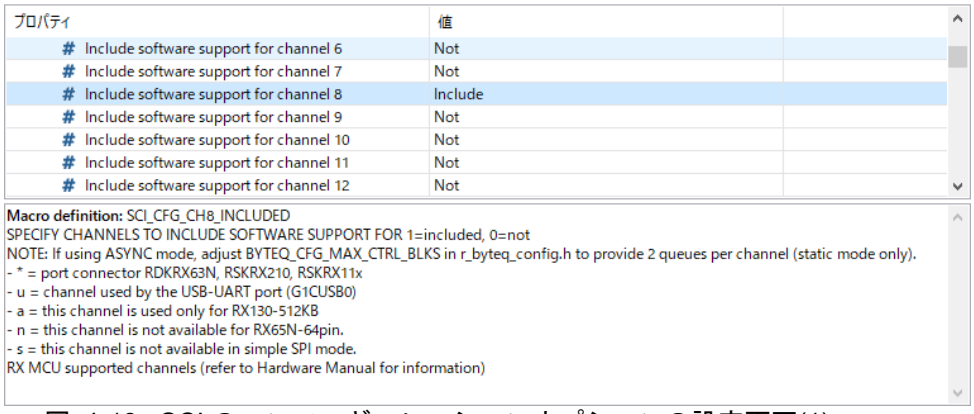


図 4-18 : SCI のコンフィギュレーションオプションの設定画面(1)

- 使用するチャネル番号の[ASYNC mode TX queue buffer size for channel n]コンフィギュレーションオプションを”160”に設定します。

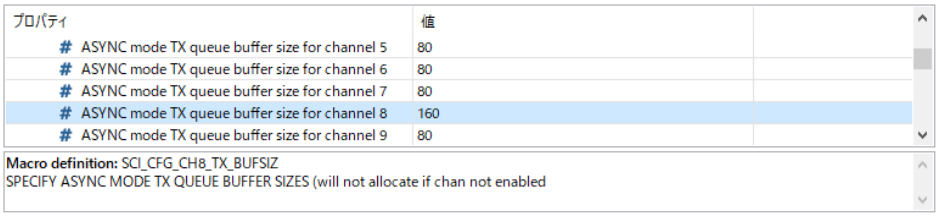


図 4-19 : SCI のコンフィギュレーションオプションの設定画面(2)

- [Transmit end interrupt]コンフィギュレーションオプションを”Enable”に設定します。

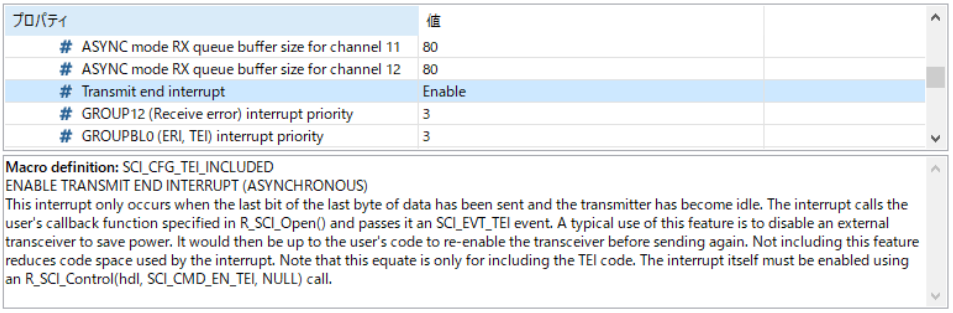


図 4-20 : SCI のコンフィギュレーションオプションの設定画面(3)

- [リソース]の[SCI]から使用するチャネルを選択します。選択したチャネルの[RXDn/SMISOn 端子], [TXDn/SMOSIn 端子]を”使用する”に設定します。

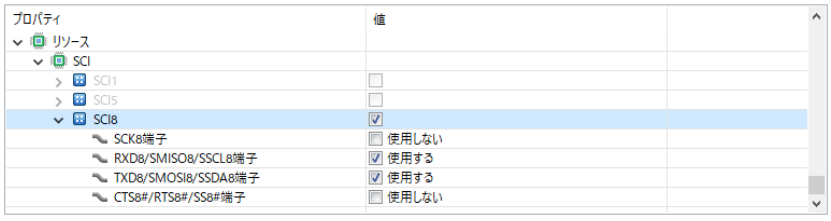


図 4-21 : SCI のコンフィギュレーションオプションの設定画面(4)

4.5.3　セキュリティデータ管理機能/デバイス固有データ(データ領域)

BLE(r_ble_rx23w)のコンフィギュレーションオプションの設定を行います。Data Flash(r_flash_rx)については設定する必要はありません。

(1) BLE(r_ble_rx23w)

Smart Configurator の[コンポーネント]タブから[r_ble_rx23w]を選択し、[Store Security Data in DataFlash] コンフィギュレーションオプションを”Enabled”に設定します。
使用する Data Flash の Block を[Data Flash Block for Security Data Management] コンフィギュレーションオプションに設定します。
また、デバイス固有データで使用する Data Flash の Block を[Device Specific data block on E2 Data Flash] コンフィギュレーションオプションに設定します。

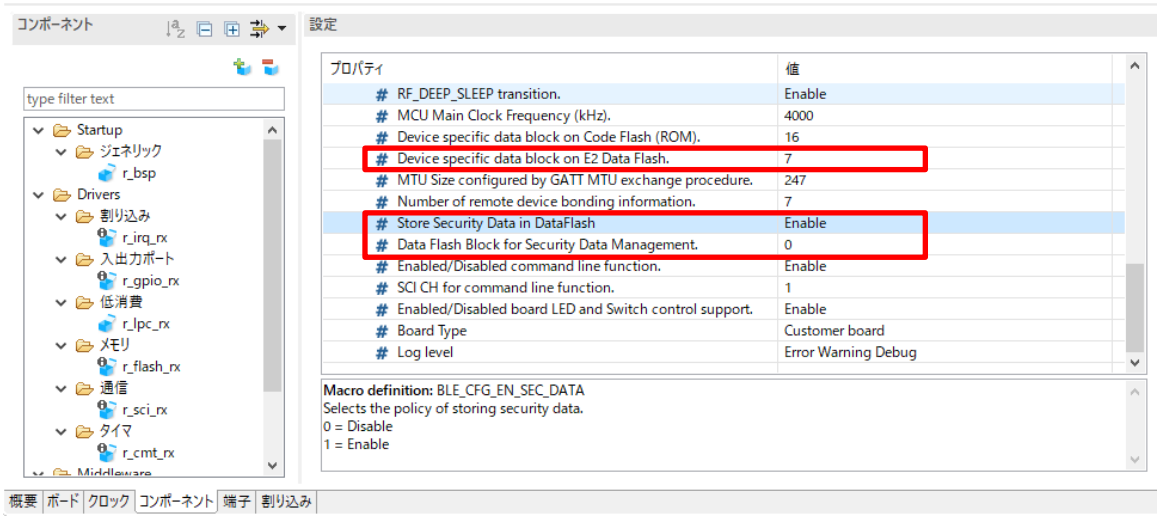


図 4-22: BLE のコンフィギュレーションオプションの設定画面(2)

4.5.4　LED and Switch 制御機能

BLE(r_ble_rx23w)と IRQ(r_irq_rx)のコンフィギュレーションオプションの設定を行います。
GPIO(r_gpio_rx)については設定する必要はありません。
Target Board と RSSK の LED, Switch が接続されている端子を表 4.1 に示します。表 4.1 の LED1, LED2, SW1, SW2 は r_ble_board.c の表記を使用しています。Customer board をご使用の場合は設定値をボードに合わせて変更します。

表 4.1 : Target Board と RSSK の LED, Switch の端子

Board	LED	Switch
Target Board*1	LED1 : PC0 LED2 : PB0	SW1 : IRQ5
RSSK	LED1 : P42 LED2 : P43	SW1 : IRQ1 SW2 : IRQ0
Customer board	以下を任意に変更 LED1 : PC5(初期値) LED2 : PC6(初期値)	以下を任意に変更 SW1 : IRQ7(初期値) SW2 : IRQ5 (初期値)

*1 : Target Board の LED 名は LED0, LED1 となります。それぞれ LED1, LED2 に対応しています。

(1)BLE(r_ble_rx23w)
Smart Configurator の[コンポーネント]タブから[r_ble_rx23w]を選択し、[Enabled/Disabled board LED and Switch control support]コンフィギュレーションオプションを”Enabled”に設定します。
使用する board のタイプを[Board Type] コンフィギュレーションオプションに設定します。[Board Type]に”Customer board”を選択した場合、「4.6.2 Customer board 用の LED and Switch 制御」に記載の Customer board 用の変更を行ってください。

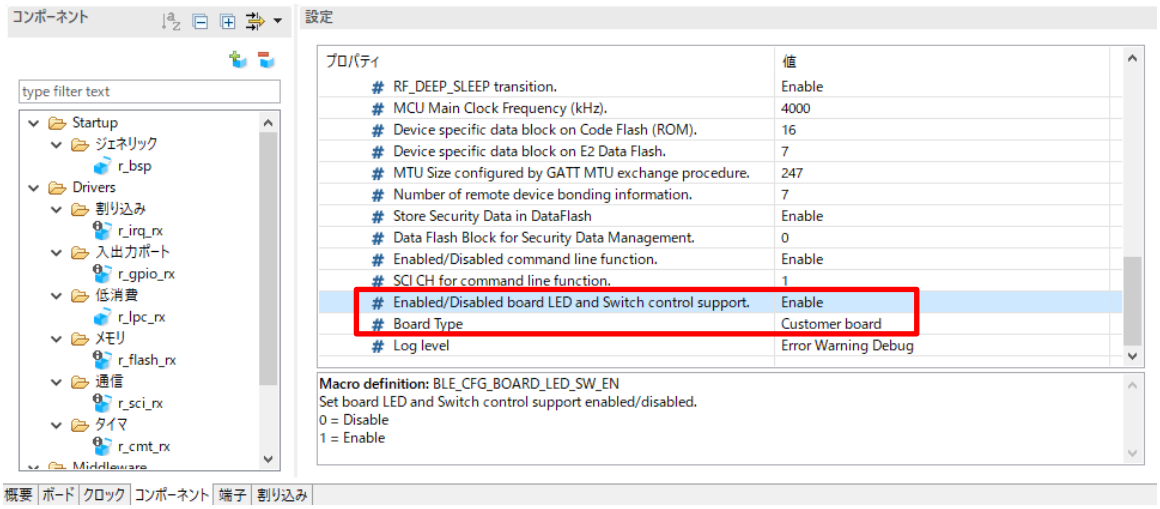


図 4-23: BLE のコンフィギュレーションオプションの設定画面(3)

(2)IRQ(r_irq_rx)
Smart Configurator の[コンポーネント]タブから[r_irq_rx]を選択し、以下のコンフィギュレーションオプションを設定します。

- 各ボードの Switch 用の IRQ 端子の[Filter for IRQn]コンフィギュレーションオプションを”Enabled”に、[Filter clock divisor for IRQn] コンフィギュレーションオプションを”Divisor1”に設定します。図 4-24 は Customer Board の初期値での設定を示しています。ご使用のボードの IRQ 端子に合わせて[Filter for IRQn], [Filter clock divisor for IRQn] コンフィギュレーションオプションを設定します。

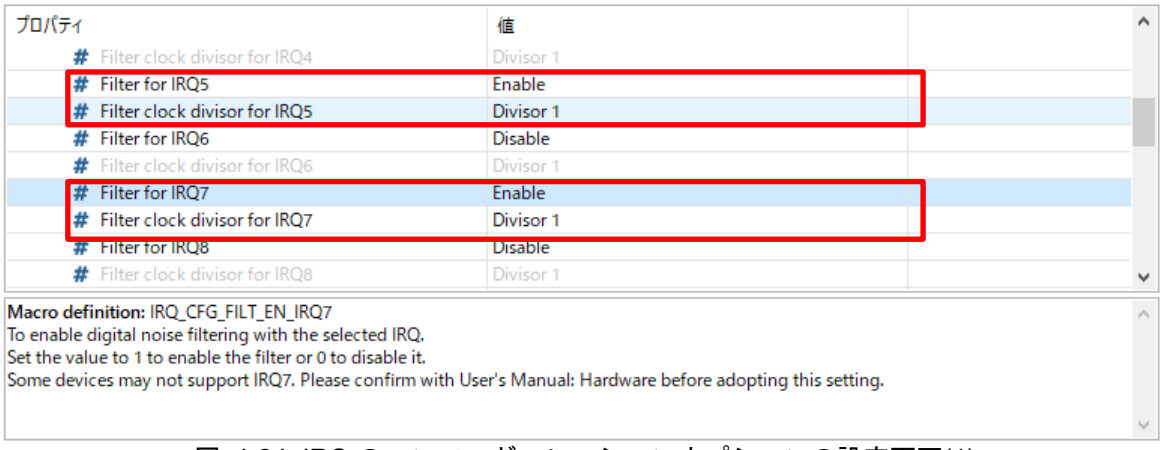


図 4-24: IRQ のコンフィギュレーションオプションの設定画面(1)

- [リソース]の[ICU]から使用するボード上の Switch と接続されている IRQ 端子を選択し、”使用する”に設定します。図 4-25 は Customer Board の初期値での設定を示しています。ご使用のボードの IRQ 端子に合わせてチェックボックスを設定します。

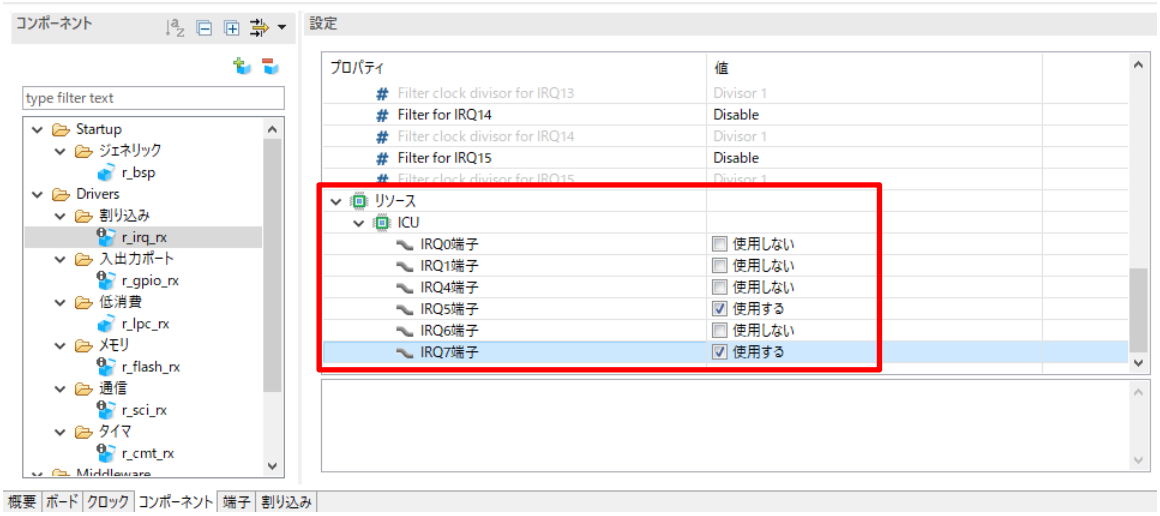


図 4-25: IRQ のコンフィギュレーションオプションの設定画面(2)

4.5.5　Renesas FreeRTOS

(1)　FreeRTOS Kernel

Smart Configurator の[コンポーネント]タブから[FreeRTOS_Kernel]を選択し、FreeRTOS Kernel の設定を行います。Bluetooth LE 通信を行う場合、以下のコンフィギュレーションオプションをデフォルトから変更します。

RF 割り込み(BLEIRQ)が最大の優先度(14)を持つため、以下の変更を行います。

- configMAX_PRIORITIES (Maximum number of priorities to the application task)
  7->16 に設定します。

- configMAX_SYSCALL_INTERRUPT_PRIORITY (Maximum syscall interrupt priority)
  4->15 に設定します。

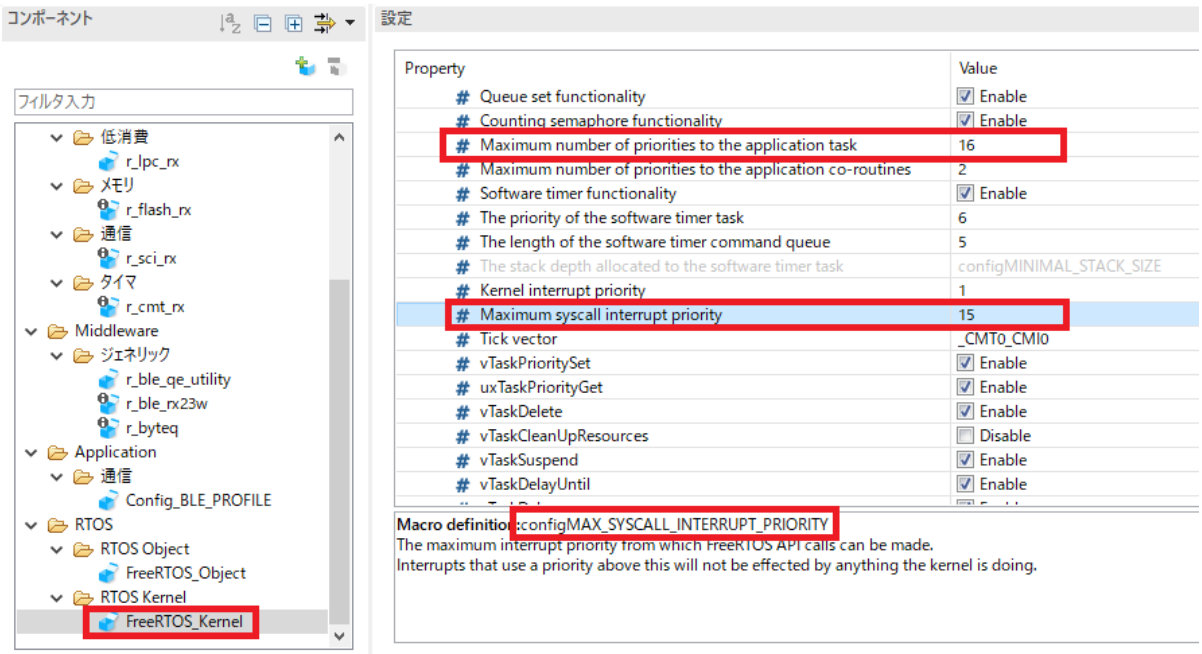


図 4-26 : FreeRTOS Kernel の設定画面(1)

ヒープサイズについては以下のコンフィギュレーションオプションを変更します。Bluetooth LE 通信を行う場合、10 程度に設定することを推奨します。

- configTOTAL_HEAP_SIZE_N (The configTOTAL_HEAP_SIZE_N)

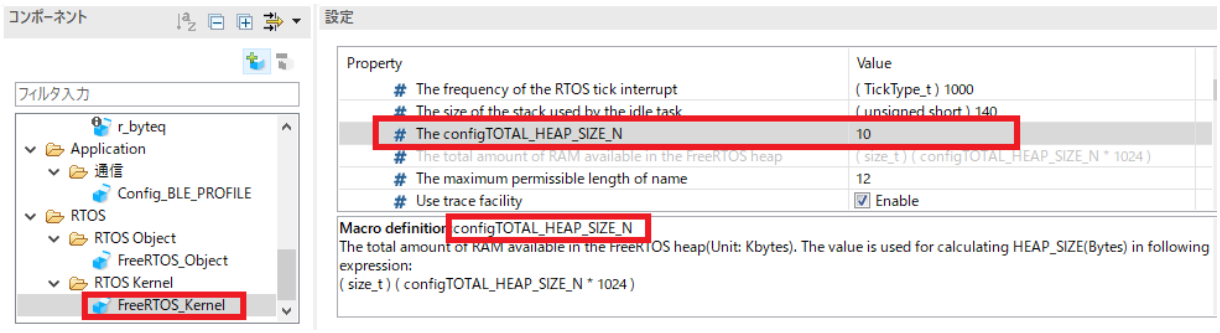


図 4-27 : FreeRTOS Kernel の設定画面(2)

以下の項目はデフォルト設定のままご使用ください。

- configUSE_MUTEXES(Mutex functionality)
  1(Enable)

- configUSE_TASK_NOTIFICATIONS (Use task notifications)
  1(Enable)

その他の FreeRTOS Kernel のコンフィギュレーションオプションについては
Smart Configurator の"Macro definition"に記載されている、"configXXX"の名称をもとに、

- FreeRTOS customization : https://www.freertos.org/a00110.html
- 「RX ファミリ Renesas FreeRTOS (R01AN4307) 3.2 カスタム設定 」

をご参照ください。

この設定画面で設定した内容はコード生成時に

src/frtos_config/FreeRTOSConfig.h

に出力されます。

(2) サブタスクの宣言
メインタスク以外のタスク(以降はサブタスクと記載)を生成する場合、Smart Configurator の[コンポーネント]タブから[FreeRTOS_Object]を選択し、アプリケーションで使用するサブタスク、セマフォなどのリソースの宣言を行います。ここではサブタスクの宣言について説明します。

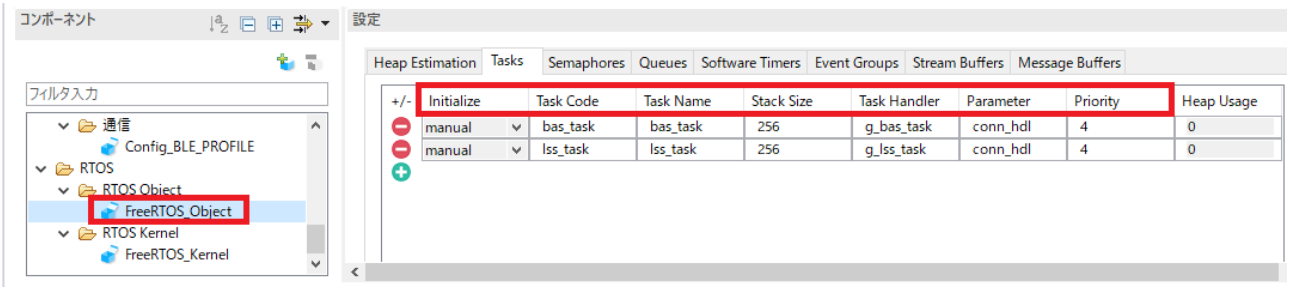


図 4-28 : FreeRTOS Object の設定画面

FreeRTOS Object の設定画面(図 4-28)のサブタスクのパラメータについて表 4.2 に示します。

表 4.2 : サブタスク関連のパラメータ

パラメータ	説明
Initialize	宣言したタスクをどこで生成するのかを指定します。 "manual"は freertos_object_init.c の Object_init_manual()内でサブタスクを生成します。この関数はアプリケーションからコールする必要があります。 "kernel"は freertos_object_init.c の Kernel_Object_init()内でサブタスクを生成します。この関数はメインタスクの生成前に Renesas FreeRTOS のコードからコールされます。
Task Code	タスク関数名を指定します。
Task Name	タスク名を指定します。デバッグ用に使用します。
Stack Size	タスクのスタックサイズを指定します。 このパラメータ x 4 のバイト数のサイズのスタックが用意されます。
Task Handler	タスク生成時にタスクハンドルを保持するタスクハンドル名を指定します。

Parameter	タスク時にタスク関数に渡すパラメータを指定します。
Priority	タスクの優先度を指定します。 Bluetooth LE 通信を行う場合は、BLE タスクの優先度よりも低い値を設定してください。

FreeRTOS Object 設定画面でサブタスクを宣言すると、コード生成時に表 4.3 のコードが出力されます。

表 4.3 : コード生成時に出力されるサブタスク関連のコード

ファイル名	説明
src/frtos_skeleton/"サブタスク名".c	サブタスクのテンプレート
src/frtos_startup/freertos_object_init.c	サブタスク生成関数を含むコード

4.6　コード生成後の設定

コンフィギュレーションオプションの設定後、Smart Configurator のコード生成ボタン をクリックし、各 FIT モジュールのコードを生成します。コード生成後、以下を設定します。

4.6.1　CMT の変更(CMT FIT モジュールのバージョンが 4.50 より古い場合に必要)

BLE FIT モジュールは CMT のチャネルを 2 つ使用するため、CMT FIT モジュールの r_cmt_rx.c 内の CMT_RX_NUM_CHANNELS の定義の後に以下を追加してください。(図 4-29 参照)
CMT FIT モジュールが v4.50 以降の場合、この変更は必要ありません。

```
#if defined(BSP_MCU_RX23W)
#undef CMT_RX_NUM_CHANNELS
#define CMT_RX_NUM_CHANNELS           (2)
#endif /* BSP_MCU_RX23W */
```

```
/***********************************************************************************************************************
Macro definitions
***********************************************************************************************************************/
/* Define the number of CMT channels based on MCU type. */
#if defined(BSP_MCU_RX62_ALL) || defined(BSP_MCU_RX63_ALL) || defined(BSP_MCU_RX21_ALL) || \
    defined(BSP_MCU_RX61_ALL) || defined(BSP_MCU_RX64_ALL) || defined(BSP_MCU_RX113)    || \
    defined(BSP_MCU_RX71_ALL) || defined(BSP_MCU_RX231)    || defined(BSP_MCU_RX23_ALL) || \
    defined(BSP_MCU_RX24_ALL) || defined(BSP_MCU_RX65_ALL) || defined(BSP_MCU_RX66_ALL) || \
    defined(BSP_MCU_RX72_ALL)
    #define CMT_RX_NUM_CHANNELS       (4)
#elif defined(BSP_MCU_RX111)  || defined(BSP_MCU_RX110)    || defined(BSP_MCU_RX130)
    #define CMT_RX_NUM_CHANNELS       (2)
#else
    #error "Error! Number of channels for this MCU is not defined in r_cmt_rx.c"
#endif

#if defined(BSP_MCU_RX23W)
#undef CMT_RX_NUM_CHANNELS
#define CMT_RX_NUM_CHANNELS           (2)
#endif /* BSP_MCU_RX23W */
```

図 4-29 : r_cmt_rx.c の変更内容

4.6.2　Customer board 用の LED and Switch 制御

Customer board を使用する場合、BLE FIT モジュールの app_lib/board/r_ble_board.c の以下の部分を変更してください。

(1)LED, Switch のマクロ定義
IRQ の端子番号、GPIO の pin 番号を定義している、
　　BLE_BOARD_SW1_IRQ
　　BLE_BOARD_SW2_IRQ
　　BLE_BOARD_LED1_PIN
　　BLE_BOARD_LED2_PIN
を Customer board の設定に合わせて変更します。

```
#if (BLE_CFG_BOARD_TYPE == 1)   /* for RX23W Target Board(TB) */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_C_PIN_0)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_B_PIN_0)
#elif (BLE_CFG_BOARD_TYPE == 2) /* for RX23W RSSK board */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_1)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_0)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_4_PIN_2)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_4_PIN_3)
#else /* BLE_CFG_BOARD_TYPE */  /* for Customer board */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_7)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_C_PIN_5)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_C_PIN_6)
#endif /* BLE_CFG_BOARD_TYPE */
```

この部分を
Customer board の設定に
合わせて変更。

図 4-30 : LED, SW マクロの定義の変更箇所

SW1 に IRQ7 端子, SW2 に IRQ5 端子が接続され、LED1 の pin は PortC の pin5, LED2 の pin は PortC の pin6 が接続されている場合、図 4-30 のような変更を行います。

(2)irq_pin_set()内のレジスタの設定
Customer board の Switch に接続されている IRQ 端子用に、irq_pin_set()関数内のレジスタ設定の部分を変更します。

```
#if (BLE_CFG_BOARD_TYPE == 1)   /* for RX23W Target Board(TB) */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_C_PIN_0)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_B_PIN_0)
#elif (BLE_CFG_BOARD_TYPE == 2) /* for RX23W RSSK board */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_1)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_0)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_4_PIN_2)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_4_PIN_3)
#else /* BLE_CFG_BOARD_TYPE */  /* for Custom board */
#define BLE_BOARD_SW1_IRQ   (IRQ_NUM_7)
#define BLE_BOARD_SW2_IRQ   (IRQ_NUM_5)
#define BLE_BOARD_LED1_PIN  (GPIO_PORT_C_PIN_5)
#define BLE_BOARD_LED2_PIN  (GPIO_PORT_C_PIN_6)
#endif /* BLE_CFG_BOARD_TYPE */
```

この部分を
Customer board の設定に
合わせて変更。

図 4-31 : irq_pin_set()の変更箇所

4.6.3　アプリケーションの追加

プロジェクト作成時にアプリケーションのソースコード([プロジェクト名.c])が出力されます。このファイルの main()から QE for BLE が出力する app_main.c の app_main()を呼ぶように変更します。Renesas FreeRTOS を使用する場合は main_task 関数から app_main()関数を呼ぶように変更します。「6.2.1 メインタスク」の「(2)app_main()のコール」を参照してください。以下にサンプルを示します。

```
#include "r_smc_entry.h"

void main(void);
void app_main(void);

void main(void)
{
        app_main();
}
```

図 4-32 : main()内での app_main()のコール

- QE for BLE

QE for BLE から使用するプロファイル、サービス、キャラクタリスティックの選択と設定を行い、アプリケーションとプロファイルのベースとなるコードを出力します。
QE for BLE の使用方法の詳細については、関連ドキュメント「Bluetooth Low Energy プロファイル開発者ガイド(R01AN6459)」をご参照ください。

4.6.4　プロファイル共通部のコード生成

プロファイル共通部は BLE FIT v2.50 以降では含まず、QE for BLE(V1.6.0 以降)でコード生成を行います。以下の組み合わせでご使用になる場合はプロファイル共通部がコード生成されないため、ビルド時にエラーとなります。

- BLE FIT v2.50 以降のバージョン
- QE for BLE V1.6.0 より前のバージョン

このような場合、下記のいずれかによりプロファイル共通部の追加が必要となります。

1. QE for BLE V1.6.0 以降にバージョンアップ後にコード生成
   Bluetooth Low Energy 対応開発支援ツール QE for BLE にある、QE for BLE[RA, RE, RX] V1.6.0 のリリースノートの"本製品のインストール方法"の手順にて、QE for BLE のバージョンアップを行います。バージョンアップ後、QE for BLE 上でコード生成を行うと、プロファイル共通部が追加されます。

2. trash からプロファイル共通部をコピー
   BLE FIT v2.50 より前のバージョンから v2.50 以降にバージョンアップした場合、プロジェクトの trash に以前のバージョンの BLE FIT が移動します。以前のバージョンのプロファイル共通部(profile_cmn, discovery)をプロジェクトの qe_gen/ble にコピーします。

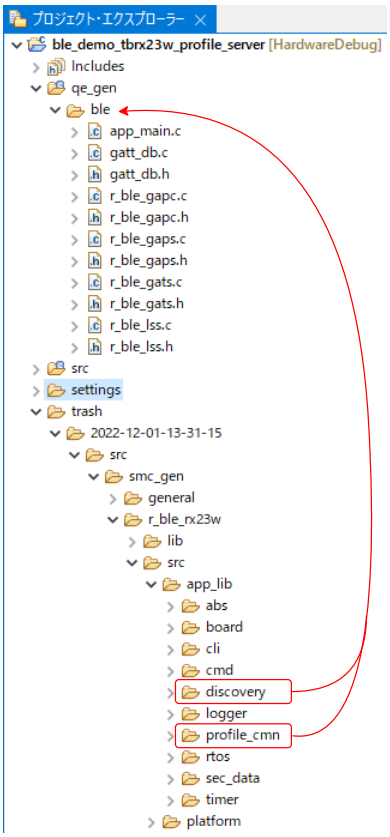


図 4-33: プロファイル共通部のコピー

4.7　リンカ設定

4.7.1　セクション配置

BLE 用のセクション配置を行います。Bluetooth LE Protocol Stack のセクション名を表 4.4 に示します。

表 4.4 : Bluetooth LE Protocol Stack のセクション名

No.	Name	Default Section Name	Bluetooth LE Protocol Stack Section Name	Description
1	Program area (ROM)	P	BLE_P	Stores machine code
2	Constant area (ROM)	C	BLE_C	Stores const type data
		C_2	BLE_C_2	
		C_1	BLE_C_1	
3	Initialized data area (ROM)	D	BLE_D	Stores data with initial values in ROM
		D_2	BLE_D_2	
		D_1	BLE_D_1	
4	Initialized data area (RAM)	R	BLE_R	Stores data with initial values in RAM
		R_2	BLE_R_2	
		R_1	BLE_R_1	
5	Uninitialized data area (RAM)	B	BLE_B	Stores data without initial values
		B_2	BLE_B_2	
		B_1	BLE_B_1	
6	Switch statement branch table area (ROM)	W	BLE_W	Stores branch tables for switch statements
		W_2	BLE_W_2	
		W_1	BLE_W_1	
7	Literal area (ROM)	L	BLE_L	Stores string literals and initializers used for dynamic initialization of aggregates

表 4.4 のセクション名を以下の手順で Bluetooth LE アプリケーションのプロジェクトに反映してください。

(1) セクション名の追加

[プロジェクト]メニューの[プロパティ]から[C/C++ Build]の[Settings]の[Tool Settings]タブ内の[Linker]→[セクション]を選択し、[...]ボタン押下で表示される、セクション設定画面において図 4-34 のように

RAM

BLE_B*
BLE_R*

ROM

BLE_C*
BLE_D*
BLE_W*
BLE_L
BLE_P

を追加します。

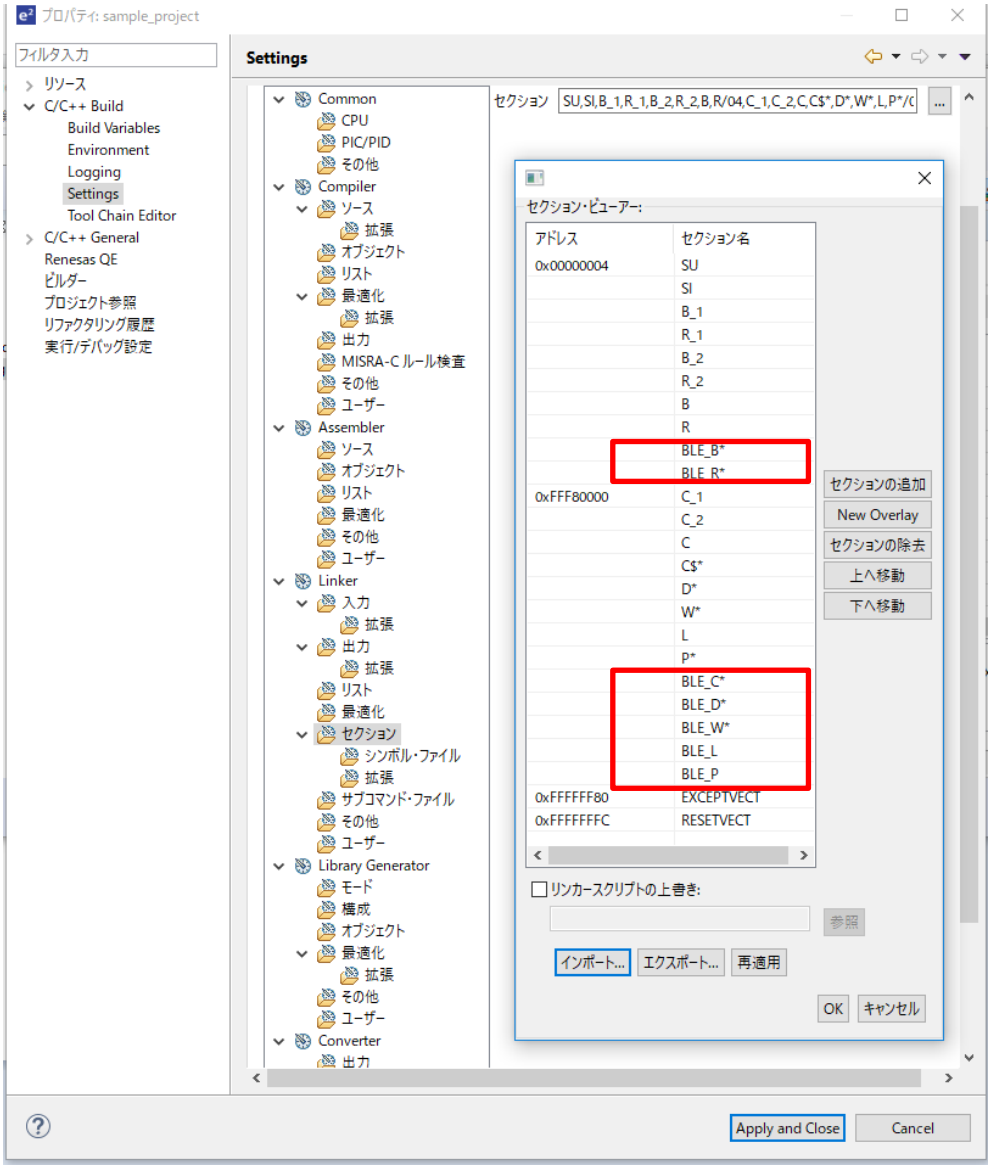


図 4-34： プロジェクトのセクションの設定画面

(2) ROM to RAM mapped section の設定
[Settings]の[Tool Settings]タブ内の[Linker]→[セクション]→[シンボル・ファイル]を選択し、[ROM から RAM へマップするセクション]に図 4-35 のように、

BLE_D=BLE_R
BLE_D_1=BLE_R_1
BLE_D_2=BLE_R_2

を追加します。

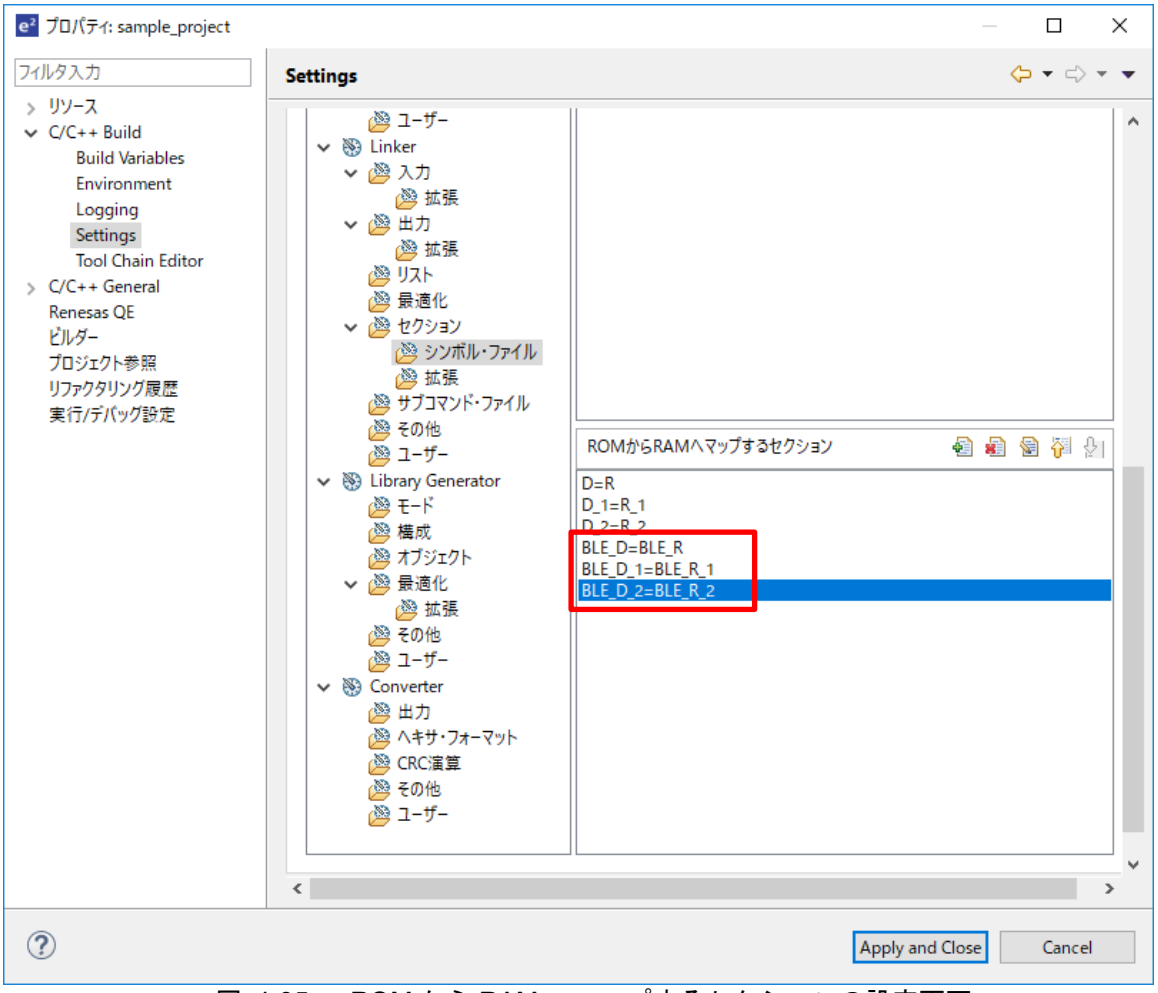


図 4-35　: ROM から RAM へマップするセクションの設定画面

[Note]
BLE 初期化を行う、R_BLE_Open() API で BLE セクションは初期化されるため、セクション初期化用テーブルである、DTBL/BTBL テーブルを変更する必要はありません。

4.7.2　Bluetooth LE Protocol Stack ライブラリ

Bluetooth LE Protocol Stack はスタティックライブラリとして提供されます。BLE FIT モジュールの lib ディレクトリ内に表 4.5 のライブラリを用意しています。

表 4.5 : Bluetooth LE Protocol Stack のライブラリ

Bluetooth LE Protocol Stack のタイプ	ライブラリ
All features	lib_ble_ps_ccrx_a.lib
Balance	lib_ble_ps_ccrx_b.lib
Compact	lib_ble_ps_ccrx_c.lib

e² studio のプロジェクトから Bluetooth LE Protocol Stack ライブラリを参照するために、以下の設定を行ってください。

1)Linker の入力

[プロジェクト]メニューの[プロパティ]から[C/C++ Build]の[Settings]の[Tool Settings]タブ内の[Linker] →[入力]を選択し、[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]にて、以下が追加されていることを確認します。登録されていない場合は以下を追加してください。(図 4-36 参照)

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

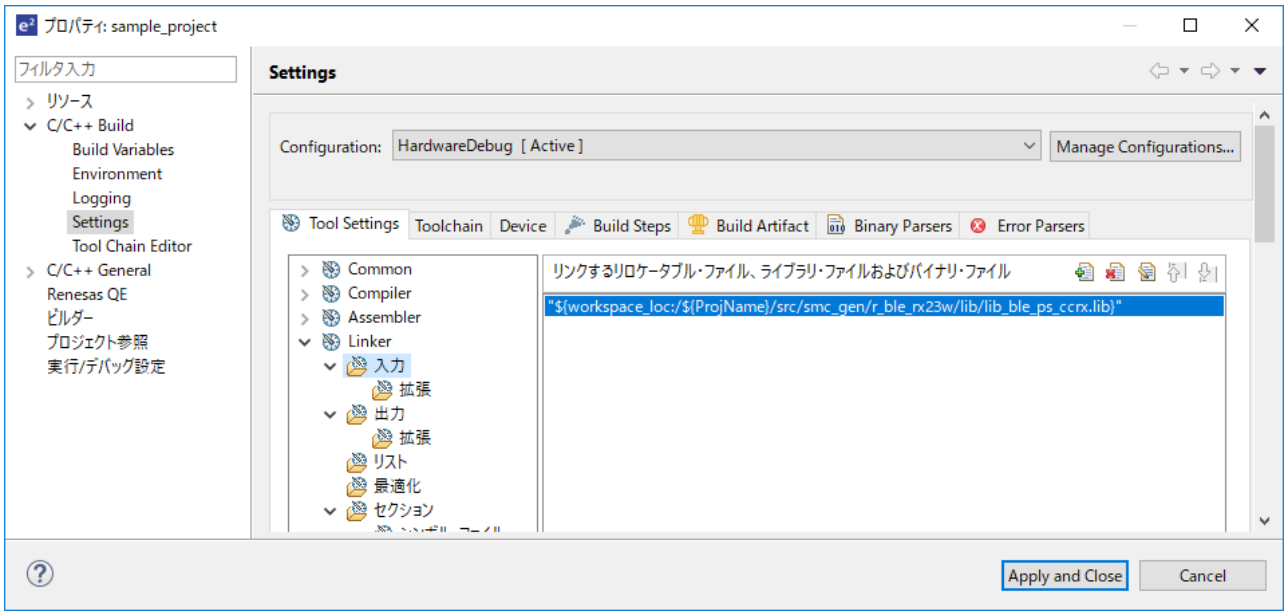


図 4-36 : Linker の入力設定画面

2)ライブラリ選択バッチファイルの登録
BLE FIT のコンフィギュレーションオプションと連携して、参照するライブラリを切り替えるために、ライブラリ選択のバッチファイルを登録します。
[プロジェクト]メニューの[プロパティ]から[C/C++ ビルド]の[設定]の[ビルド・ステップ]タブ内の[ビルド前のステップ]の[コマンド:]に以下を入力します。(図 4-37 参照)

${ProjDirPath}/src/smc_gen/r_ble_rx23w/lib/ble_fit_lib_selector.bat

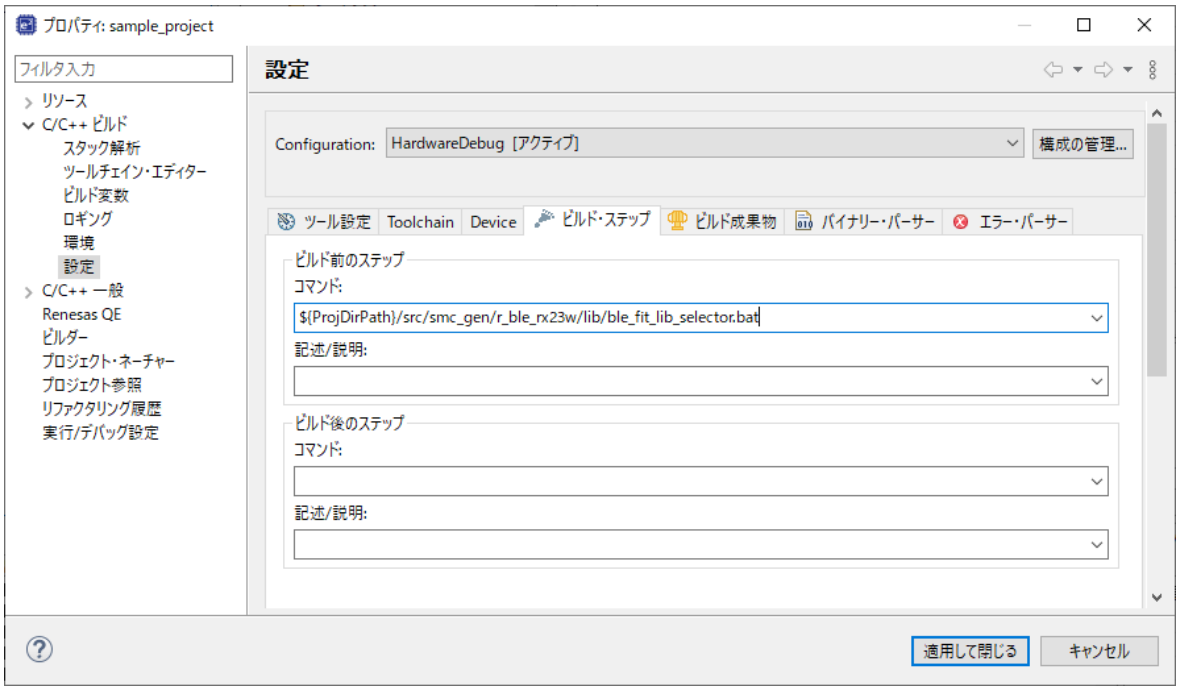


図 4-37 : ビルド・ステップの入力設定画面

4.8　デバッグ設定

e² studio の[実行]メニューの[デバッグの構成]にて、[Renesas GDB Hardware Debugging]からアプリケーションのプロジェクトを選択し、以下の項目の設定を行います。これらの項目の設定後、プロジェクトをビルドします。ビルドが成功したら、ファームウェアを board に書き込みます。

4.8.1　Flash の ID Code

「2.10 デバイス固有データ」に記載したフラッシュメモリプロテクト機能を有効にするために ID Code の設定を行います。[Debugger]タブの[Connection Settings]タブの[フラッシュ]の[ID コード]に”45FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF”を入力します。ファームウェアのライトに失敗した場合、[ID コード]の内容をご確認ください。

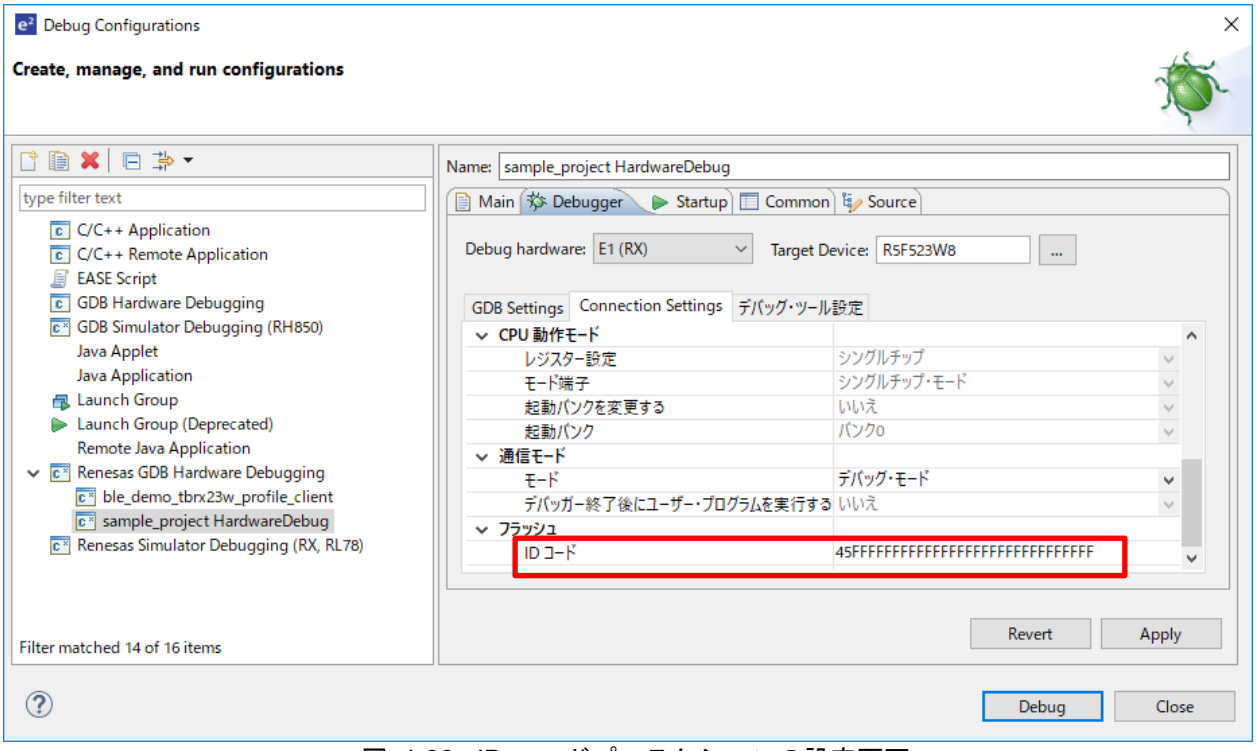


図 4-38 : ID コードプロテクションの設定画面

4.8.2　電源

[電源]の[エミュレーターから電源を供給する(MAX 200mA)]を”いいえ”に設定します。

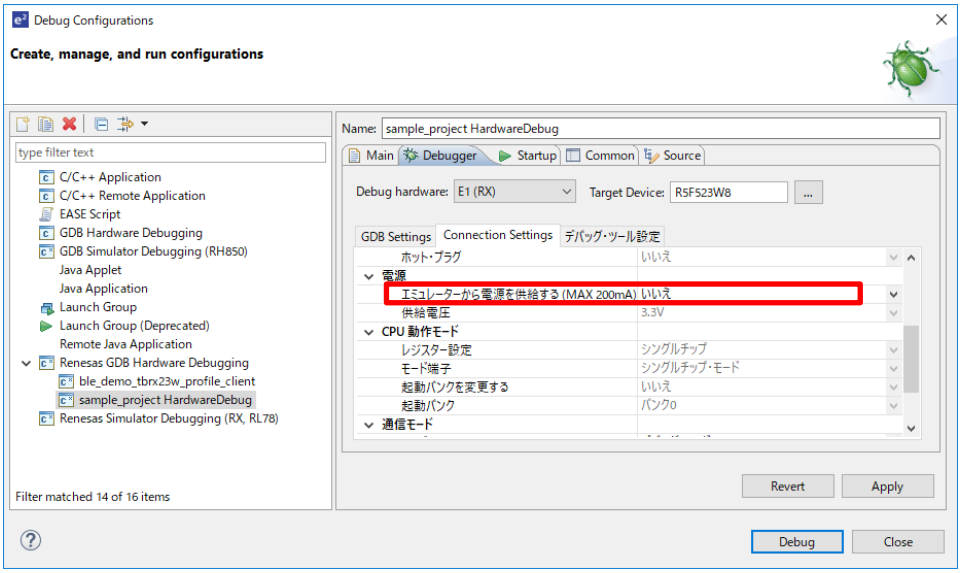


図 4-39 : エミュレーターからの電源供給の設定画面

4.9 IAR 開発環境

本章では IAR Embedded Workbench for Renesas RX 上でのプロジェクト作成について説明します。
BLE プロジェクトに Renesas FreeRTOS を含む場合、IAR 開発環境に対応していません。

4.9.1 プロジェクト作成

4.1 から 4.7 までの手順にて e^2 studio 上でベースとなる CCRX プロジェクトを作成します。

4.9.2 IAR 用 iodefines.h の追加

プロジェクトに追加する IAR 用の iodefines.h を(1), (2)のいずれかの方法により生成し、CCRX プロジェクトへ配置します。4.9.3 プロジェクト変換の前に CCRX プロジェクトへの iodefines.h の追加が必要となります。

(1) RX Smart Configurator 単体版から生成

- https://www.renesas.com/software-tool/smart-configurator
  に公開されている RX Smart Configurator 単体版をインストールします。
- RX Smart Configurator 単体版を起動します。
- “ファイル>>新規…”を選択し、”新規スマート・コンフィグレータファイル”ウィンドウ上で、
  “プラットフォーム” : RX23W のデバイス
  “ツールチェーン” : “IAR EWRX Toolchain”
  “ファイル名” : プロジェクト名
  を設定し、”終了”ボタンを押します。

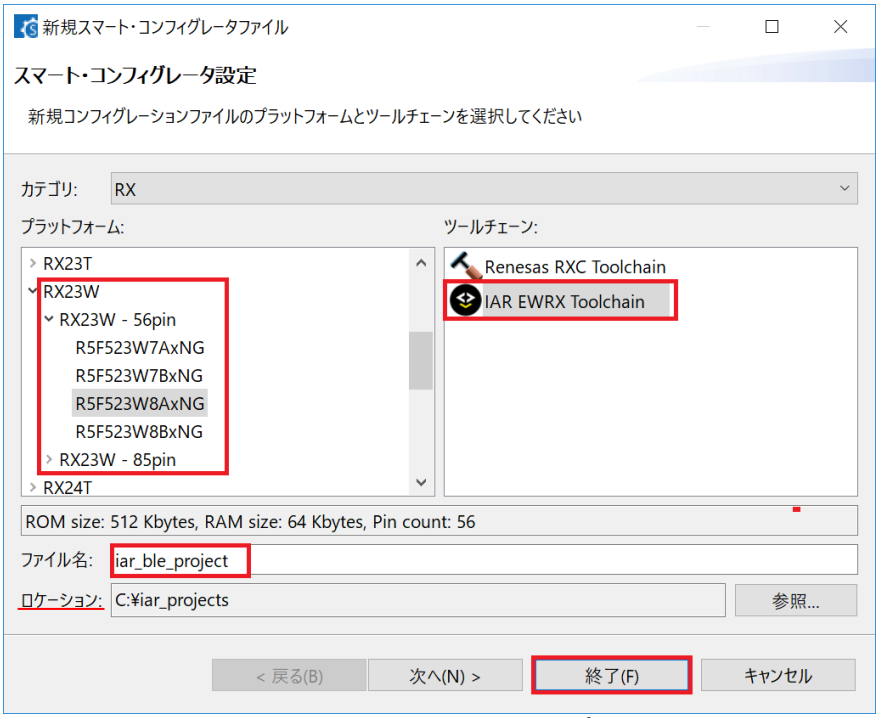


図 4-40 : RX Smart Configurator 単体版の新規プロジェクトの作成画面

- 生成したプロジェクトの“コンポーネント”タブ上で”コンポーネントの追加”ボタンを押し、”コンポーネントの追加”ウィンドウ上から”r_bsp”を選択し、”終了”ボタンを押します。

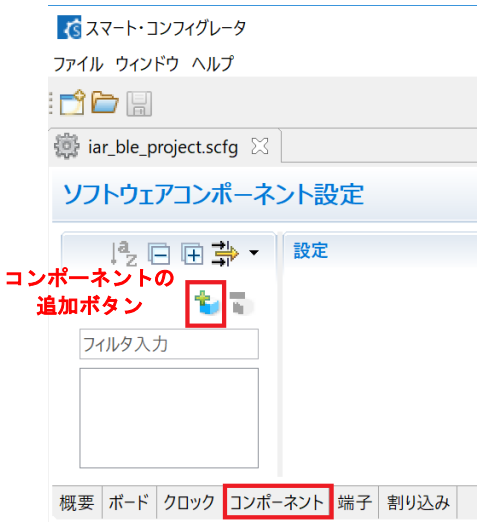


図 4-41： コンポーネントの追加(1)

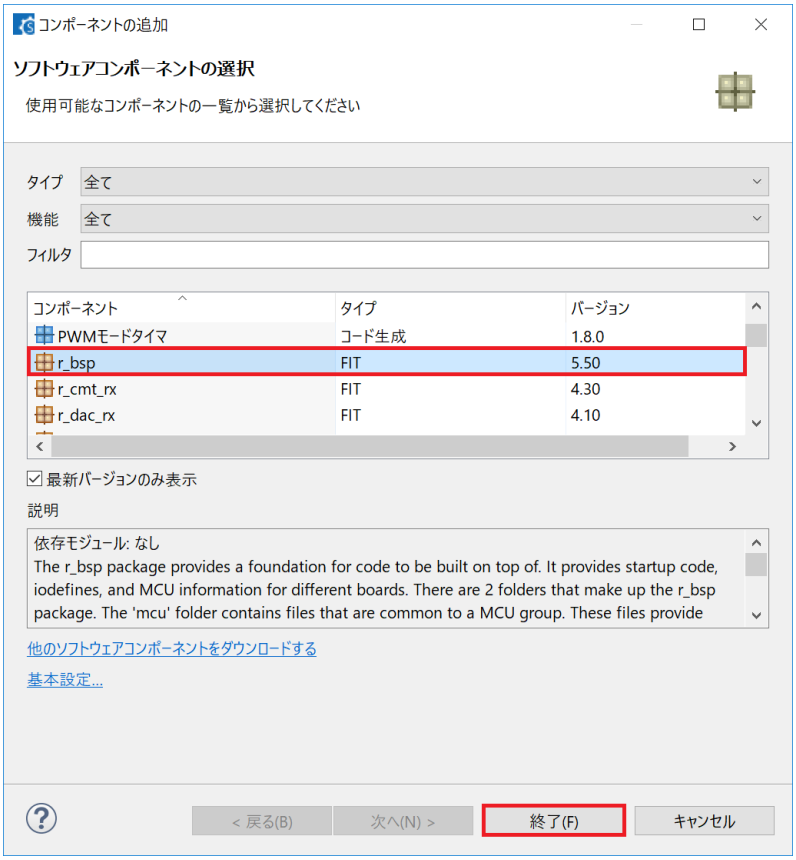


図 4-42： コンポーネント追加(2)

”r_bsp”がない場合は、CS+を起動し、CS+からスマート・コンフィグレータを起動して、”r_bsp”のインストールを行います。詳細については「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」をご参照ください。

- コード生成ボタンを押します。

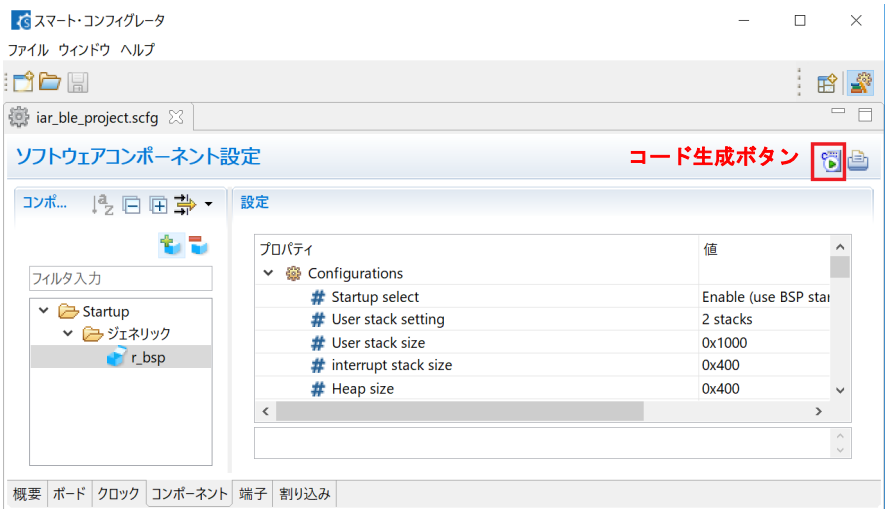


図 4-43 : コード生成

- RX Smart Configurator 単体版プロジェクトのロケーション(図 4-40 参照)ディレクトリの
¥src¥smc_gen¥r_bsp¥mcu¥rx23w¥register_access
にある、iccrx ディレクトリをコピーし、4.9.1 で生成した CCRX プロジェクトの
¥src¥smc_gen¥r_bsp¥mcu¥rx23w¥register_access
に配置します。

(2) デモプロジェクトからコピー
表 8.1 のデモプロジェクトの
<<デモプロジェクト名>>¥src¥smc_gen¥r_bsp¥mcu¥rx23w¥register_access
にある、iccrx ディレクトリをコピーし、4.9.1 で生成した CCRX プロジェクトの
¥src¥smc_gen¥r_bsp¥mcu¥rx23w¥register_access
に配置します。

4.9.3　プロジェクト変換

以下の手順で CCRX プロジェクトを IAR のプロジェクトに変換します。

(1) IAR Embedded Workbench for Renesas RX を起動します。
(2) “ツール” >> “Convert To IAR for RX…”をクリックします。
(3) Root directory of source project に 4.9.1 で作成した CCRX プロジェクトを指定します。
(4) Project file conversion で”Enable”のチェックボックスを選択します。
(5) Project type で”Renesas e2studio for RX”を選択します。
(6) Source code substitution で”Enable”のチェックボックスのチェックを外します。
(7) “Execute”をクリックします。

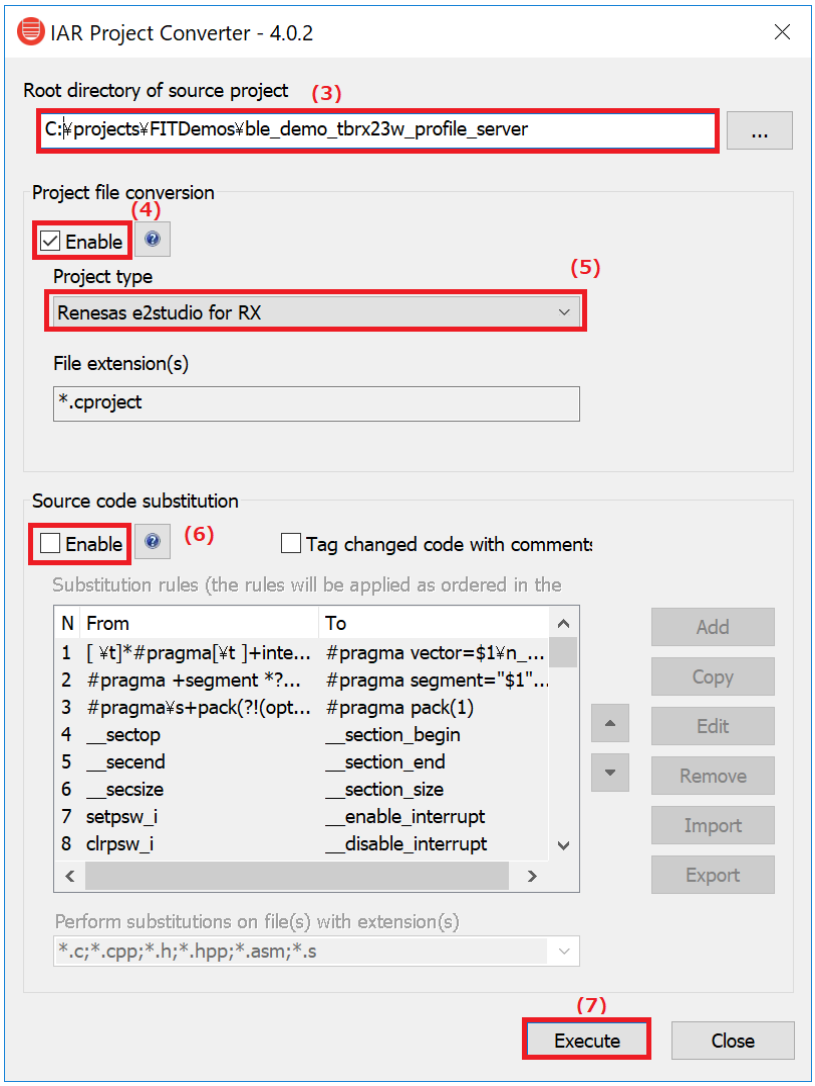


図 4-44 : IAR Project Converter の設定画面(1)

(8) Choose destination directory のウィンドウで”Operate on a copy of the source code in the following directory”のラジオボタンを選択します。

(9) 変換先の IAR プロジェクトを置くディレクトリを指定し、”OK”をクリックします。

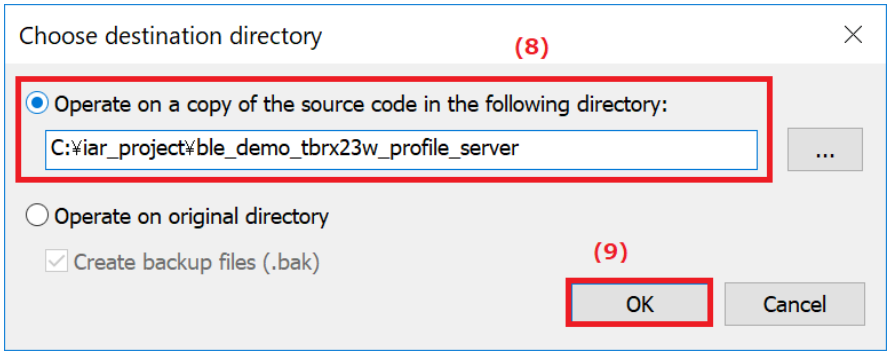


図 4-45 : IAR Project Converter の設定画面(2)

(10) レポートのウィンドウで”OK”をクリックします。

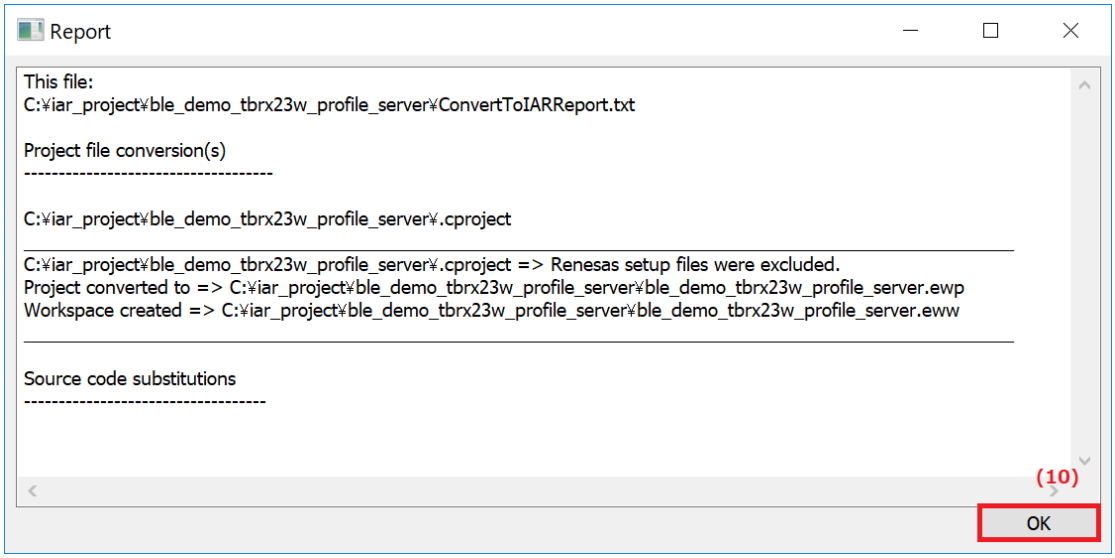


図 4-46 : IAR Project Converter の設定画面(3)

4.9.4　プロジェクトオプション設定

IAR Embedded Workbench for Renesas RX の“プロジェクト>> 既存プロジェクトの追加”をクリックし、4.9.3 で変換したプロジェクトの.ewp ファイルを開きます。”プロジェクト >> オプション”から以下のオプションを変更します。

(1) スタック／ヒープ
IAR プロジェクトの”一般オプション”カテゴリの”スタック／ヒープ”のサイズを CCRX プロジェクトのスタック／ヒープ(r_bsp のコンフィギュレーションオプション)と同じサイズに設定します。CCRX プロジェクトと IAR プロジェクトのスタック／ヒープサイズを示すオプションの対応関係を表 4.6、図 4-47 に示します。

表 4.6 : プロジェクトのスタック／ヒープのオプション名

CCRX プロジェクト	IAR プロジェクト
User stack size	ユーザーモードスタックサイズ
interrupt stack size	スーパーバイザモードスタックサイズ
Heap size	ヒープサイズ

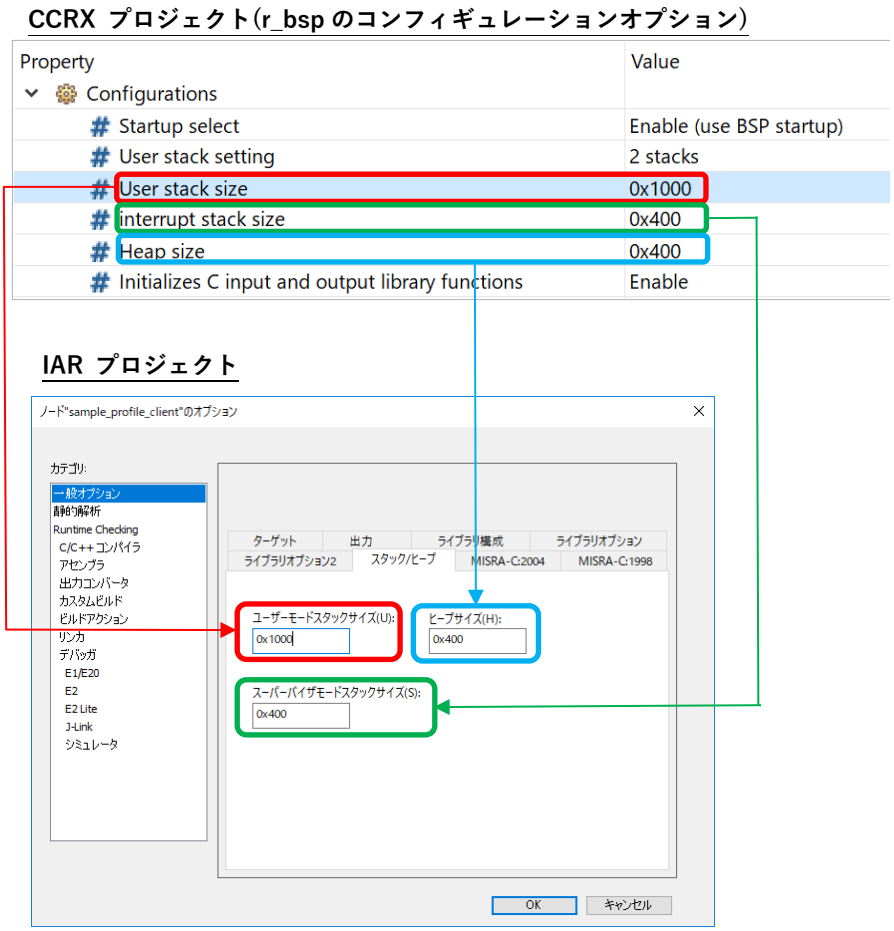


図 4-47 : CCRX プロジェクトの IAR プロジェクトのスタック／ヒープの対応関係

(2) ターゲット

”一般オプション”カテゴリの”ターゲット”の”デバイス”から”RX23W Group”のデバイスを選択します。
“浮動小数点数”の”タイプ’double’のサイズ”に”32 ビット”を選択します。

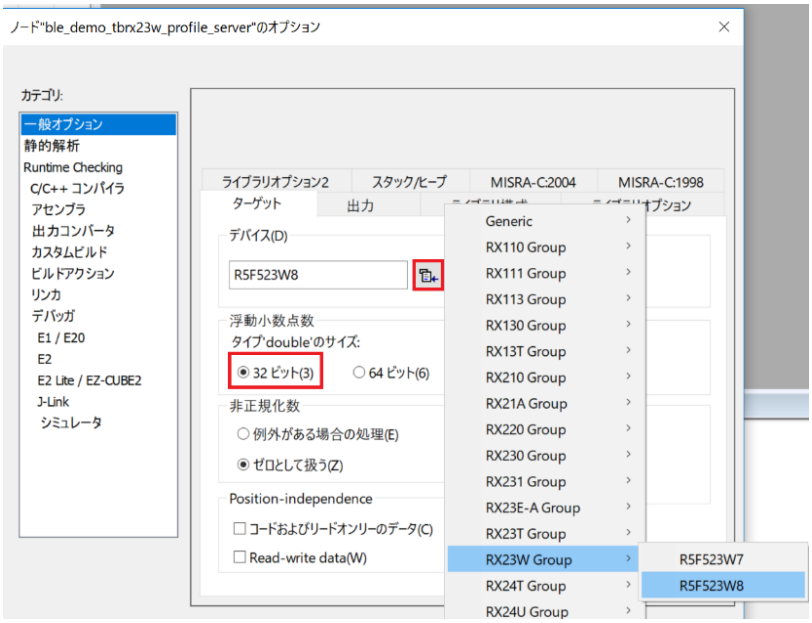


図 4-48： ターゲット設定

(3) ライブラリ構成

”一般オプション”カテゴリの”ライブラリ構成”のライブラリを”通常の DLIB”に設定します。

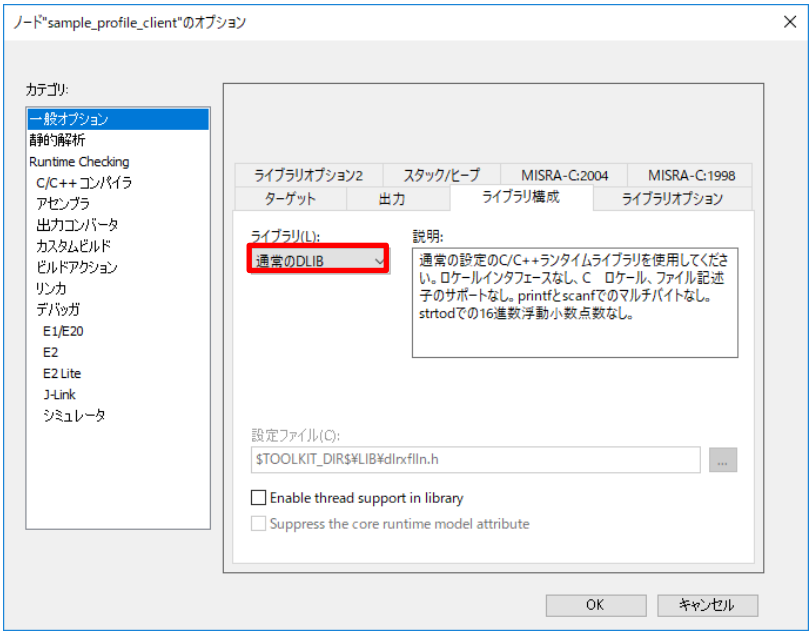


図 4-49： ライブラリ構成の設定

(4) プリプロセッサ

“C/C++コンパイラ”カテゴリの”プリプロセッサ”の”追加インクルードディレクトリ”に CCRX プロジェクトと同じインクルードパスが設定されていることを確認します。

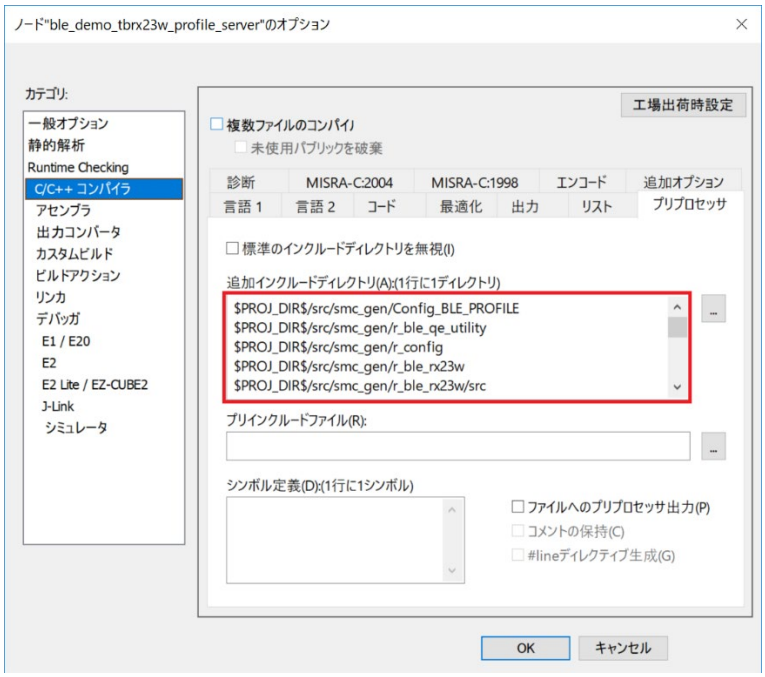


図 4-50： 追加インクルードディレクトリ設定

(5) 最適化

“C/C++コンパイラ”カテゴリの”最適化”にてプロジェクトの最適化のレベルを設定します。デバッグを行わない場合は、レベルを”高”, “サイズ”に設定することで、ROM サイズの削減を行うことが可能です。

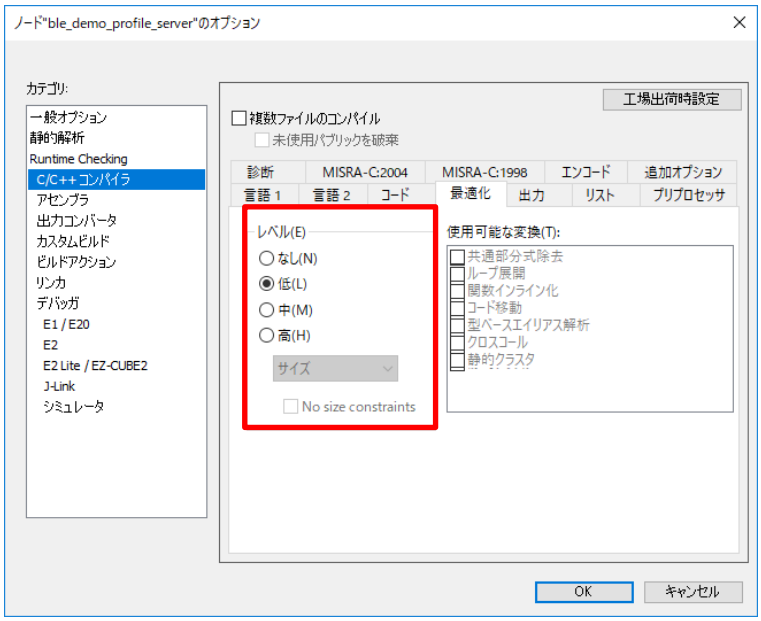


図 4-51： 最適化設定

(6) 追加出力の生成
”出力コンバータ”カテゴリの”追加出力の生成”にチェックを入れ、”出力形式”を”Motorola S-records”に設定します。

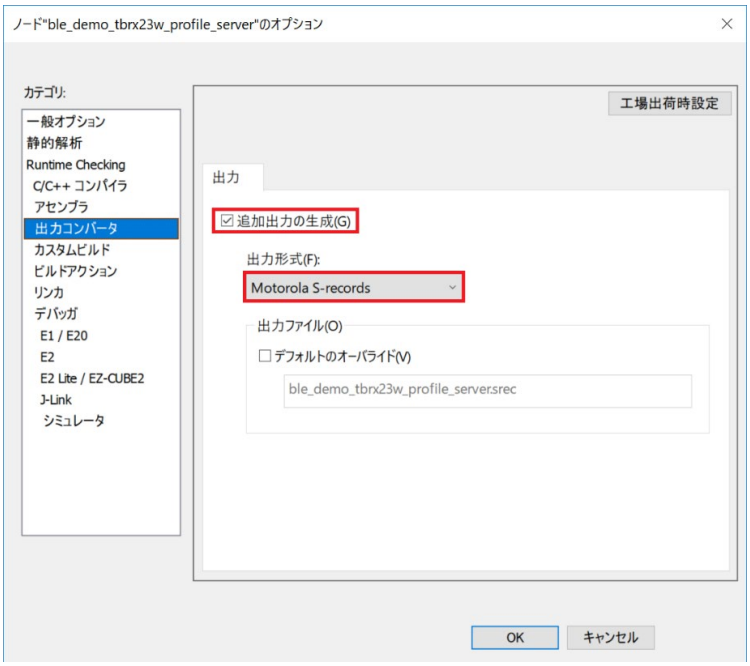


図 4-52 : 出力形式の設定

(7) プリビルドコマンドライン
“ビルドアクション”カテゴリの”プリビルドコマンドライン”に
ble_fit_lib_selector.bat のパスを指定します。

設定例.
ライブラリ選択バッチファイルのパスが
$PROJ_DIR$¥src¥smc_gen¥r_ble_rx23w¥lib¥ble_fit_lib_selector.bat
の場合、下記のように入力します。

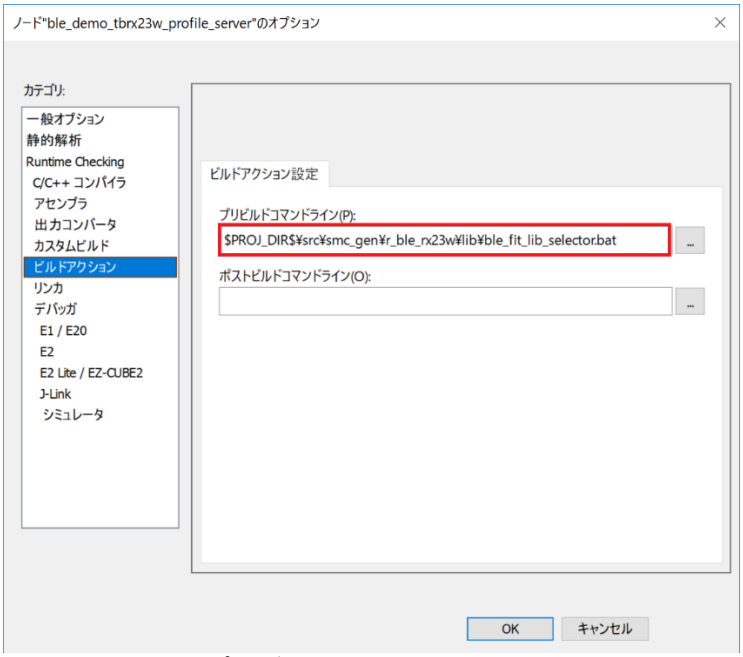


図 4-53 : プリビルドコマンドラインの設定

(8) リンカ設定ファイル

“リンカ”カテゴリの“デフォルトのオーバライド”にチェックを入れ、lnkr5f523w8.icf、または、lnkr5f523w7.icf ファイルを指定します。

設定例.
リンカ設定ファイルのパスが
$TOOLKIT_DIR$¥CONFIG¥lnkr5f523w8.icf
の場合、下記のように入力します。

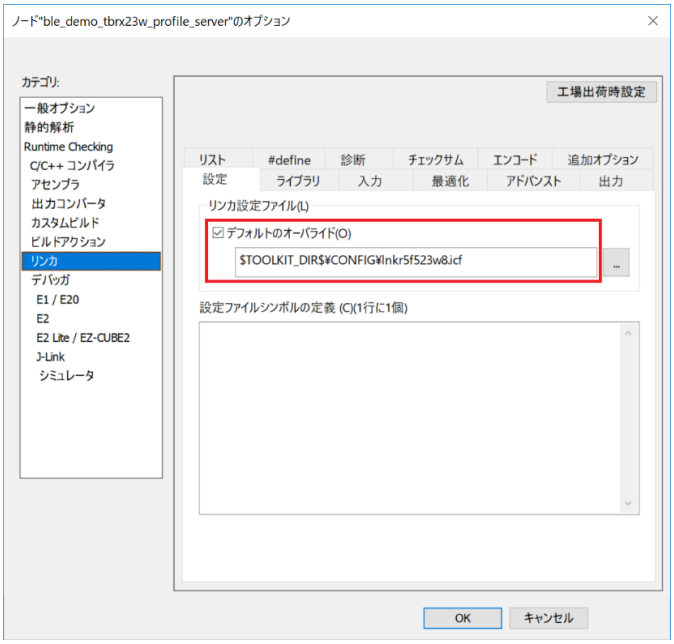


図 4-54： リンカ設定ファイル

(9) 追加ライブラリ

“リンカ”カテゴリの”追加ライブラリ”に lib_ble_ps_ccrx.lib のパスを指定します。

設定例.
Bluetooth LE Protocol Stack のパスが
$PROJ_DIR$/src/smc_gen/r_ble_rx23w/lib/lib_ble_ps_ccrx.lib
の場合、下記のように入力します。

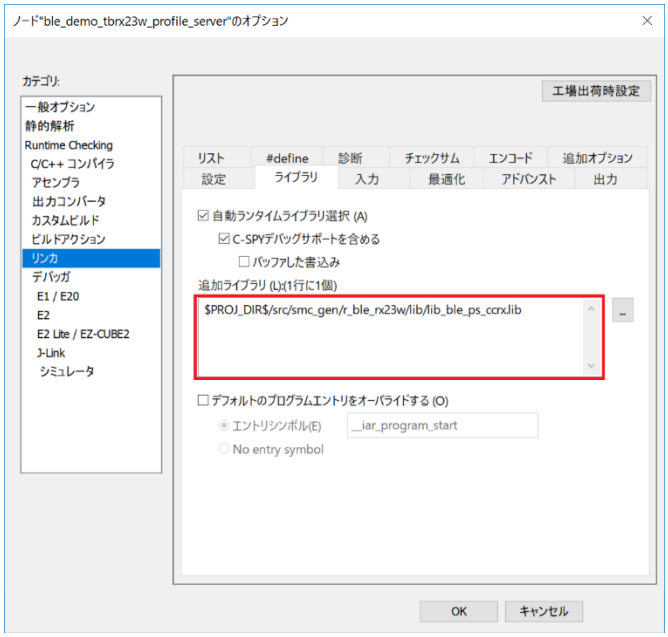


図 4-55： 追加ライブラリの設定

(10) デバッガ

“デバッガ”カテゴリの”設定”の”ドライバ”に使用するエミュレーターを指定します。

設定例.

エミュレーターに E2 Lite を使用する場合、下記のように”E2 Lite/EZ-CUBE2”を選択します。

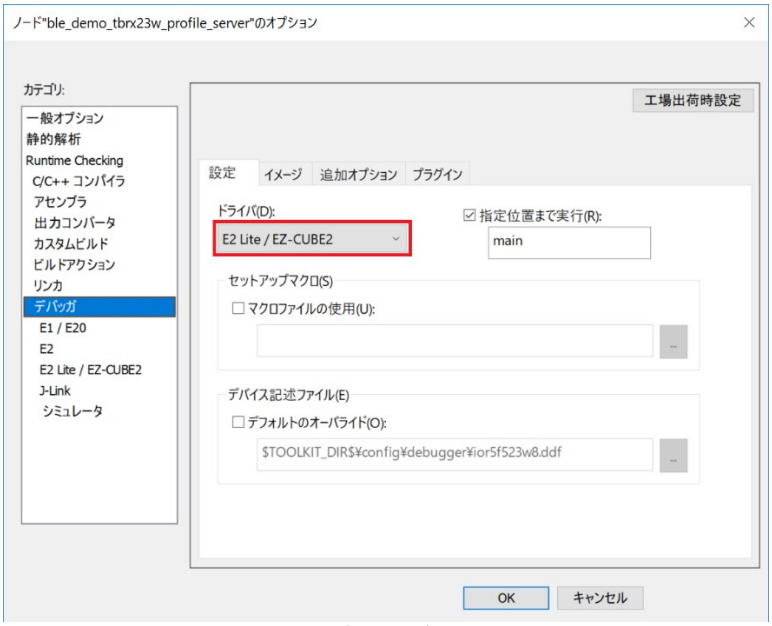


図 4-56 : デバッガの設定

4.9.5　ビルド対象からの除外

プロジェクト変換時にビルド対象となった下記ディレクトリをビルド対象から除外します。

(1) BLE FIT モジュールの lib ディレクトリ
src/smc_gen/r_ble_rx23w/lib 上で右クリックし、”オプション”を選択します。
オプションウィンドウ上の”ビルドから除外”をチェックします。

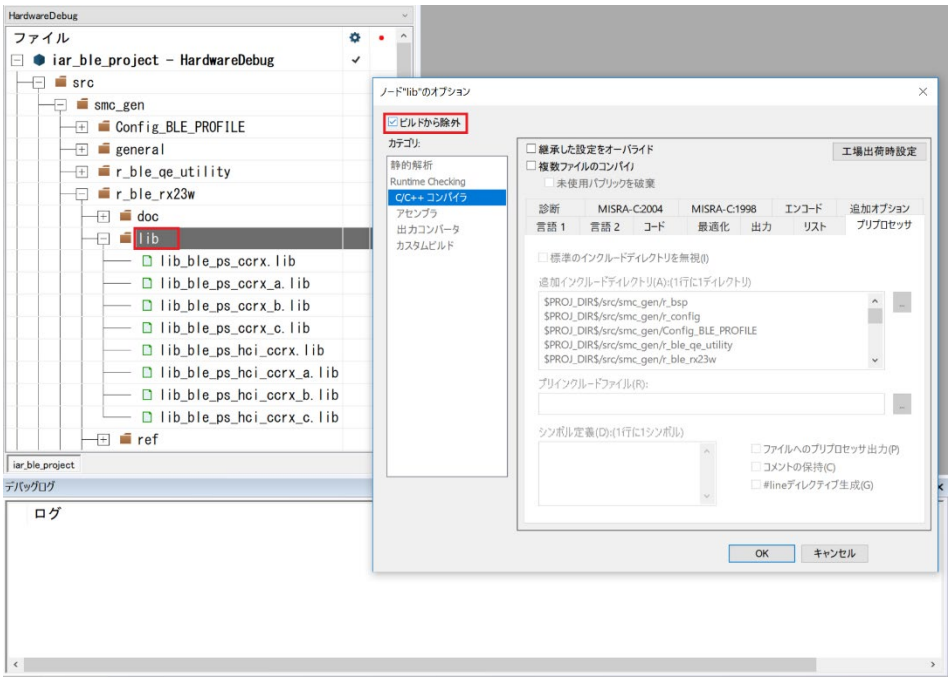


図 4-57 : lib ディレクトリのオプション

(2) trash ディレクトリ
プロジェクト内に”trash”ディレクトリがある場合、右クリックし、”オプション”を選択します。
オプションウィンドウ上の”ビルドから除外”をチェックします。

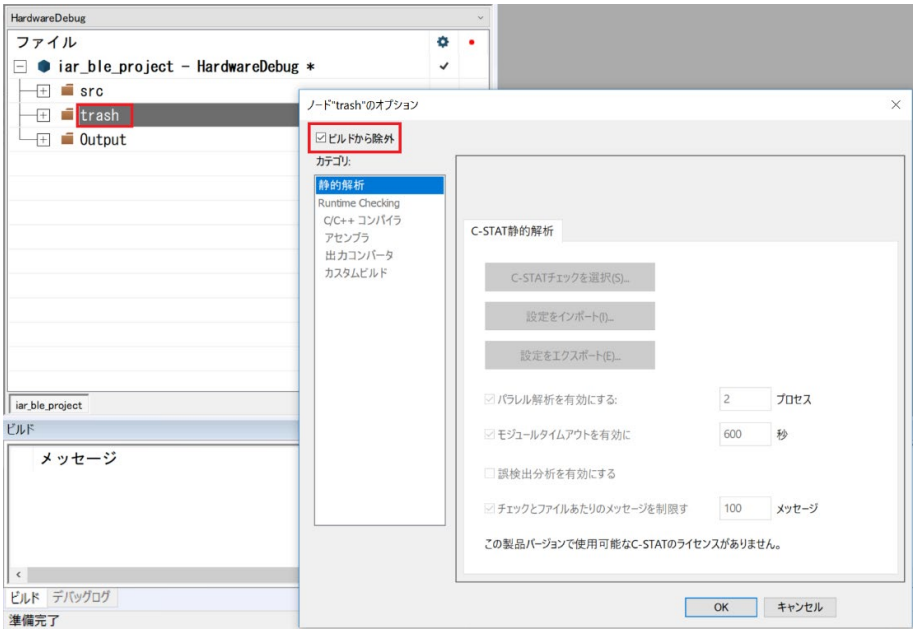


図 4-58 : trash のオプション

4.9.6　プロジェクトのビルドとファームウェアのダウンロード

“プロジェクト >> すべてを再ビルド”をクリックし、ビルドを行います。
ビルド完了後、”E2/E2 Lite >> ハードウェア設定”を選択します。

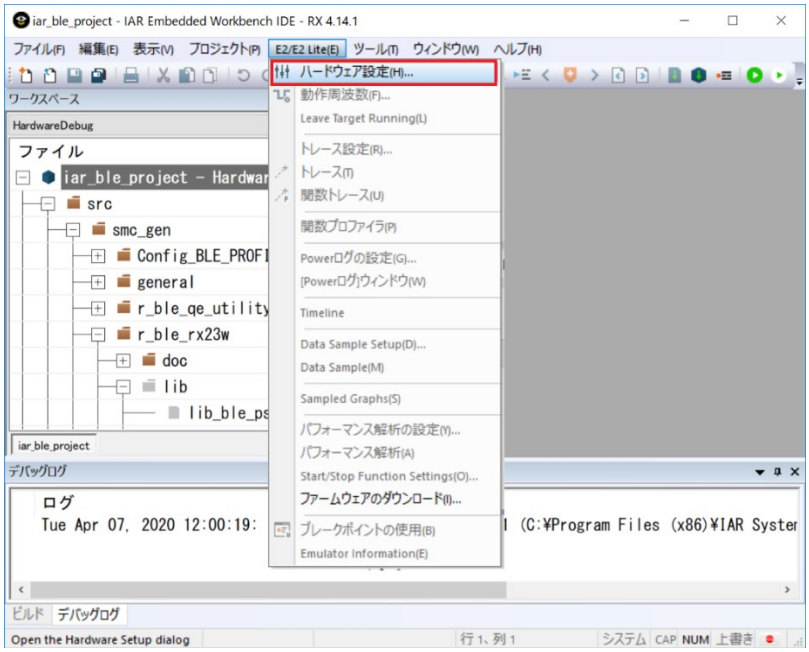


図 4-59： ハードウェア設定(1)

“ハードウェア設定”ウィンドウで設定を行います。デフォルトから変更がなければ、そのまま”OK”ボタンを押します。

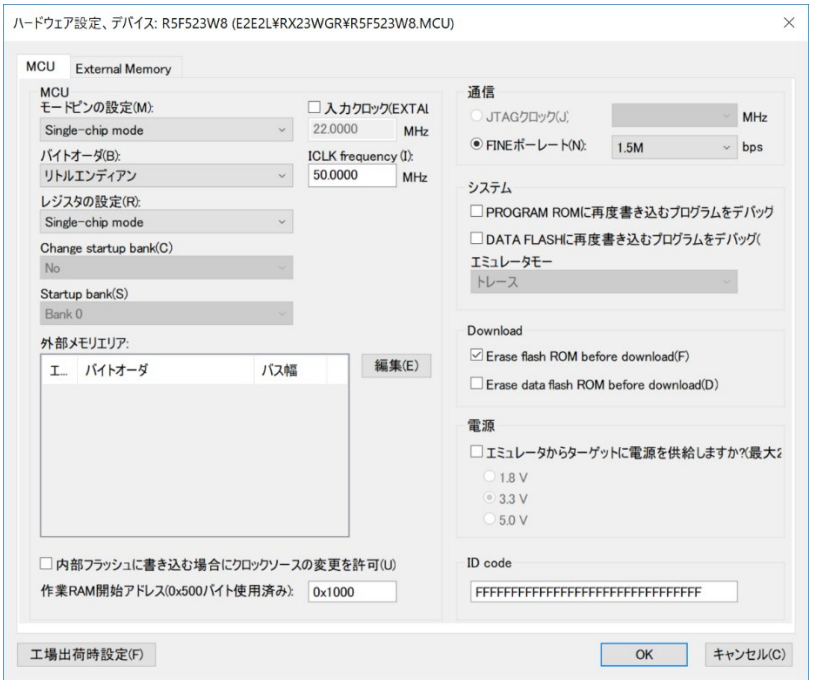


図 4-60： ハードウェア設定(2)

“プロジェクト >> ダウンロードしてデバッグ”をクリックし、ファームウェアをダウンロードします。

5. アプリケーションの作成方法

アプリケーションの作成方法については「RX23W グループ Bluetooth Low Energy アプリケーション開発者ガイド(R01AN5504)」をご参照ください。

6. Renesas FreeRTOS の BLE タスク

Renesas FreeRTOS のアプリケーションは図 6-1 のタスク構成を想定しています。BLE タスクは Bluetooth LE コアスタック機能および Bluetooth LE 通信制御とコールバック処理のアプリケーションで構成されます。BLE GATT アプリタスクは GATT サービスのアプリケーションで構成されます。

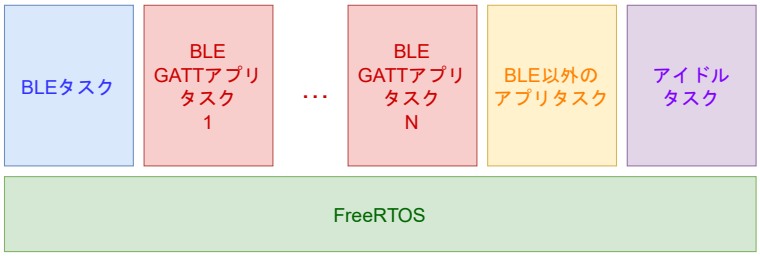


図 6-1 : Renesas FreeRTOS 利用時のタスク構成

6.1 コール可能な API

BLE タスクと BLE GATT アプリタスクとでは Bluetooth LE 通信用にコール可能な API が異なります。

- BLE タスクからコール可能な API
  - GAP API (R_BLE_GAP_XXX)
  - GATT Server API (R_BLE_GATTS_XXX)
  - GATT Client API (R_BLE_GATTC_XXX)
  - Vendor Specific API (R_BLE_VS_XXX)
  - L2CAP API (R_BLE_L2CAP_XXX)
  - 抽象 API (R_BLE_ABS_XXX)
  - Discovery API (R_BLE_DISC_XXX)
  - R_BLE_[GATT サービス略称名 + S(サーバ選択時) or C(クライアント選択時)]_YYY

- BLE GATT アプリタスクからコール可能な API
  - R_BLE_[GATT サービス略称名 + S(サーバ選択時) or C(クライアント選択時)]_YYY

- どちらかのタスクからのみコール可能な API
  - R_BLE_CLI_Printf

BLE_LOG_XXX や BLE 以外の FIT を使用する API(R_BLE_LPC_XXX、R_BLE_TIMER_XXX、R_BLE_SECD_XXX、R_BLE_CLI_XXX(R_BLE_CLI_Printf 以外)、R_BLE_CMD_XXX、R_BLE_BOARD_XXX)はすべてのタスクでコール可能です。

なお、BLE GATT アプリタスクで GATT サービスの API がコールされると、実際には BLE タスクのコアスタック機能で処理されます。図 6-2 は BLE タスク、BLE GATT アプリタスクを含むマルチタスク環境において、2 つの BLE GATT アプリタスクが GATT サービスを制御し、各サービスの Characteristic を Notification で送信する場合の Bluetooth LE 通信を示しています。

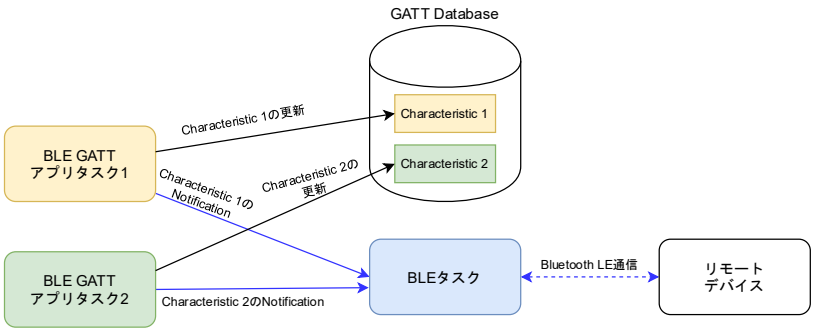


図 6-2 : BLE タスクと BLE GATT アプリタスク

6.2 BLE タスク生成

BLE タスクの生成はメインタスクとして生成する場合とメインタスク以外のタスク(以降サブタスクと記載)として生成する方法があります。

6.2.1 メインタスクとして生成する場合

BLE タスクをメインタスクとして生成する場合、以下のコード変更を行います。

(1) メインタスク生成のパラメータ変更

src/frtos_startup/freertos_start.c の Processing_Before_Start_Kernel()関数内に xTaskCreate()によるメインタスク生成コードが出力されます。freertos_start.c 内で BLE タスク制御のヘッダである、rtos/r_ble_rtos.h をインクルードし、メインタスク生成の以下のパラメータ(表 6.1)を図 6-3 のように変更します。

表 6.1 : BLE タスク(メインタスク)生成時に変更するパラメータ

パラメータ	設定値
スタックサイズ	1024 スタックサイズは x4 したバイト数(4096 バイト)となります。
優先度	6 (サブタスクよりも高い値)
タスクハンドル	g_ble_task_hdl

```
#include "rtos/r_ble_rtos.h"

/* 省略*/

/************** task creation ****************************/
/* Main task. */
ret = xTaskCreate(main_task, "MAIN_TASK", 1024, NULL, 6, &g_ble_task_hdl);
if (pdPASS != ret)                              スタックサイズ  優先度   タスクハンドル
{
    while(1)
    {
        /* Failed! Task cannot be created. */
    }
}
```

図 6-3 : BLE タスク生成のコード

(2) app_main()のコール

コード生成で出力される src/[プロジェクト名.c]の main_task()内で app_main()をコールするように変更します。

```
#include "FreeRTOS.h"
#include "task.h"

void app_main(void);

void main_task(void *pvParameters)
{
    app_main();

    vTaskDelete(NULL);
}
```

図 6-4 : main_task()内の app_main()のコール

(3) app_main.c の変更

FreeRTOS の API をコールできるように FreeRTOS.h と task.h をインクルードし、app_main()内のメインループで Bluetooth LE Protocol Stack が処理を行っていない場合は、他のタスクに制御を渡すため、以下のように変更します。

```
#include "FreeRTOS.h"
#include "task.h"

/* 省略*/

/* main loop */
while (1)
{
    /* 省略 */

    /* Process Event */
    R_BLE_Execute();

    if(0 != R_BLE_IsTaskFree())
    {
       ulTaskNotifyTake(pdFALSE, portMAX_DELAY);
    }
}
```

図 6-5 : メインループの変更

(4) サブタスクの生成

「4.5.5(2) サブタスクの宣言」の方法でサブタスクを生成する場合、Object_init_manual()を[プロジェクト名.c]、または app_main.c などのアプリケーションのコードからコールします。

```
void app_main(void)
{
    /* 省略 */

    extern void Object_init_manual (void);
    Object_init_manual();

    /* main loop */
    while (1)
    {
        /* 省略 */
    }
}
```

図 6-6 : サブタスク生成の例

6.2.2　サブタスクとして生成する場合

BLE タスクをサブタスクとして「4.5.5(2) サブタスクの宣言」の方法で生成する場合について説明します。

(1)　BLE タスクの登録

FreeRTOS Object 設定画面で登録する場合、図 6-7 のように表 6.1 のパラメータを設定してください。スタックサイズは x4 したバイト数(4096 バイト)となります。

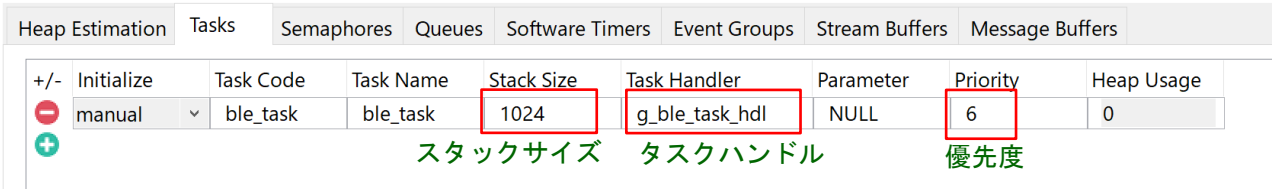


図 6-7 : FreeRTOS Object 設定画面での BLE タスクの登録

(2)　app_main()のコール

コード生成で出力される src/ rtos_skeleton/ble_task.c 内で app_main()をコールするように変更します。また、ble_task 内で FreeRTOS の API をコールできるように FreeRTOS.h と task.h をインクルードします。以下に ble_task.c のサンプルを示します。

```
/* Start user code for import. Do not edit comment generated here */
#include "FreeRTOS.h"
#include "task.h"

void app_main(void);

/* End user code. Do not edit comment generated here */

void ble_task(void * pvParameters)
{
    /* Start user code for function. Do not edit comment generated here */
    app_main();

    vTaskDelete(NULL);
    /* End user code. Do not edit comment generated here */
}
```

図 6-8 : ble_task.c のサンプル

(3)　app_main.c の変更

「6.2.1(3) app_main.c の変更」を参照してください。

(4) BLE タスク制御ヘッダのインクルードとタスクハンドルの削除
src/frtos_startup/freertos_object_init.c 内の BLE タスクハンドルの定義を削除し、BLE タスク制御ヘッダである、rtos/r_ble_rtos.h をインクルードします。

```
#include "FreeRTOS.h"
#include "freertos_start.h"
#include "../frtos_skeleton/task_function.h"
#include "task.h"
/* BLE タスク制御ヘッダをインクルード */
#include "rtos/r_ble_rtos.h"

/* 省略*/

void Kernel_Object_init (void);
void Object_init_manual (void);
BaseType_t ret;
/* BLEタスクハンドルの定義を削除 */
/* TaskHandle_t g_ble_task_hdl = NULL; */
```

図 6-9 : freertos_object_init.c の変更

(5) サブタスクの生成
Object_init_manual()を[プロジェクト名.c]などのメインタスクのコードからコールします。以下はサブタスクを起動してメインループで何も処理しないメインタスクのサンプルです。

```
#include "FreeRTOS.h"
#include "task.h"

void main_task(void *pvParameters)
{
    /* Create all other application tasks here */
    void Object_init_manual (void);
    Object_init_manual();
    while(1)
    {
    }
}
```

図 6-10 : サブタスク生成の例

6.3 BLE GATT アプリタスクの生成

BLE GATT アプリタスクを「4.5.5(2) サブタスクの宣言」の方法で生成する場合について説明します。

(1) BLE GATT アプリタスクの登録
FreeRTOS Object 設定画面での BLE GATT アプリタスクの登録のサンプルを示します。

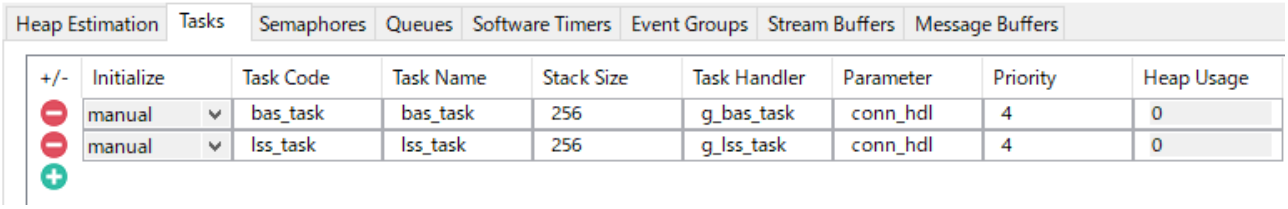


図 6-11 : FreeRTOS Object 設定画面での BLE GATT アプリタスクの登録

(2) BLE GATT アプリタスクでインクルードするヘッダファイル
コード生成で出力される src/ rtos_skeleton/[Task Code 名].c 内に BLE GATT アプリタスクを実装します。BLE GATT タスクアプリでは、以下のヘッダのインクルードが必要となります。

- GATT サービスの API のヘッダ　../Config_BLE_Profile/r_ble_[GATT サービス略称].h
- BLE タスク制御用のヘッダ　rtos/r_ble_rtos.h
- FreeRTOS の API のヘッダ　FreeRTOS.h と task.h
- サブタスク用のヘッダ　../ frtos_skeleton/task_function.h

```
/* FreeRTOS APIヘッダ */
#include "FreeRTOS.h"
#include "task.h"
/* GATTサービス APIヘッダ */
#include "../Config_BLE_Profile/r_ble_bas.h"
/* BLEタスク制御用ヘッダ */
#include "rtos/r_ble_rtos.h"
```

図 6-12 : BLE GATT アプリタスクでインクルードするヘッダファイル

なお、サブタスク用ヘッダ(task_function.h)には、サブタスクとタスク間通信するためのイベント定義などを実装します。以下にサブタスク用ヘッダのサンプルを示します。

```
/* 省略*/

/* bas */
#define BAS_WAIT_EN_CCCD        (0x0001)
/* lss */
#define LSS_NOTIF_DIS_CCCD      (0x0000)
#define LSS_WAIT_EN_CCCD        (0x0001)
#define LSS_WAIT_DIS_CCCD       (0x0002)
#define LSS_WAIT_PUSH_SW        (0x0004)
#define LSS_WAIT_WR_BLINK       (0x0008)
```

図 6-13 : サブタスク用ヘッダのサンプル

(3)　生成用 API(Object_init_manual)の編集
(1)でタスクを生成するときにパラメータを渡す場合は、static 変数にパラメータを用意した後に、xTaskCreate で指定するように編集します。以下は接続確立時にコネクションハンドルを渡してタスク生成するサンプルです。

```
static uint16_t gs_conn_hdl;

/* 省略 */
void Object_init_manual (uint16_t conn_hdl)
{
    gs_conn_hdl = conn_hdl;

    /************** task creation ***************************/
    ret = xTaskCreate(bas_task, "bas_task", 256, &gs_conn_hdl, 4, &g_bas_task);
    /* 省略 */
    ret = xTaskCreate(lss_task, "lss_task", 256, &gs_conn_hdl, 4, &g_lss_task);
    /* 省略 */
} /* End of function Object_init_manual()*/
```

図 6-14：生成したタスクにパラメータを渡す場合のサンプル

(4)　生成用 API(Object_init_manual)のコール
「6.2.1(4) サブタスクの生成」を参照してください。

6.4　他のタスクからの BLE タスクの起動

BLE タスクに他のタスクからタスクスイッチする場合、BLE タスク制御ヘッダである、rtos/r_ble_rtos.h をインクルードし、R_BLE_RTOS_WakeTask()をコールしてください。以下は BLE GATT アプリタスク(LED Switch Service)から BLE タスクへタスクスイッチするサンプルです。

```
/* BLE タスク制御ヘッダをインクルード */
#include "rtos/r_ble_rtos.h"

/* 省略 */
static uint16_t gs_conn_hdl;

void lss_task(void * pvParameters)
{
    uint8_t push_state;
    ble_status_t retval;

    /* 省略 */

    retval = R_BLE_LSS_NotifySwitchState(gs_conn_hdl, &push_state);
    if(BLE_SUCCESS == retval)
    {
        R_BLE_CLI_Printf("R_BLE_LSS_NotifySwitchState ¥n");
        R_BLE_RTOS_WakeTask();
    }
    /* 省略 */
}
```

図 6-15 : BLE タスクへタスクスイッチするサンプル

6.5　RF 以外の割り込みからの BLE タスクの起動

BLE FIT モジュールではコンソール機能使用時の SCI の割り込み、LED & SW 使用時の IRQ 割り込みで登録したタスクを起動するためのしくみを用意しています。

6.5.1　コマンド入力で起動するタスクの登録

コマンド入力で発生した SCI 割り込みでタスクを起動したい場合、rtos/r_ble_rtos.h をインクルードし、アプリケーションの初期化時に R_BLE_CLI_RegisterEventCb() API により、タスク起動を行うコールバックを登録してください。以下のサンプルではコマンド入力により、Bluetooth 通信が行えるように BLE タスクを ISR から起動可能な関数を登録しています。

```
#include "rtos/r_ble_rtos.h"

/* 省略 */

void app_main(void)
{
    /* Initialize BLE */
    R_BLE_Open();

    /* 省略 */

    /* Configure CommandLine */
    R_BLE_CLI_Init();
    R_BLE_CLI_RegisterCmds(gsp_cmds, ARRAY_SIZE(gsp_cmds));
    /* コマンド入力で起動するタスクの登録 */
    R_BLE_CLI_RegisterEventCb(R_BLE_RTOS_WakeTaskFromIsr);
    R_BLE_CMD_SetResetCb(ble_host_stack_init);
```

図 6-16 : コマンド入力で起動するタスクの登録の例

6.5.2　LED & SW 使用時の BLE タスクの起動

SW 押下時に発生する IRQ 割り込みでタスクを起動したい場合、rtos/r_ble_rtos.h をインクルードし、アプリケーションの初期化時に R_BLE_BOARD_RegisterSwitchEventCb() API により、タスク起動を行うコールバックを登録してください。以下のサンプルでは SW 押下により、Bluetooth 通信が行えるように BLE タスクを ISR から起動可能な関数を登録しています。

```
#include "rtos/r_ble_rtos.h"

/* 省略 */

void app_main(void)
{
    /* Initialize BLE */
    R_BLE_Open();

    /* Configure the board */
    R_BLE_BOARD_Init();
    R_BLE_BOARD_RegisterSwitchCb(BLE_BOARD_SW2, sw_cb);
    /* SW押下で起動するタスクの登録 */
    R_BLE_BOARD_RegisterSwitchEventCb(R_BLE_RTOS_WakeTaskFromIsr);
```

図 6-17 : SW 押下で起動するタスク登録の例

7. ツール

Renesas の RX ファミリ用 Bluetooth Low Energy プロトコルスタックの Web ページ (https://www.renesas.com/software-tool/bluetooth-low-energy-protocol-stack-rx-family)からダウンロード可能なパッケージ内に図 7-1 の HCI モードのファームウェア向けのツールを用意しています。HCI モードの説明については「Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205), 6. HCI モード」をご参照ください。

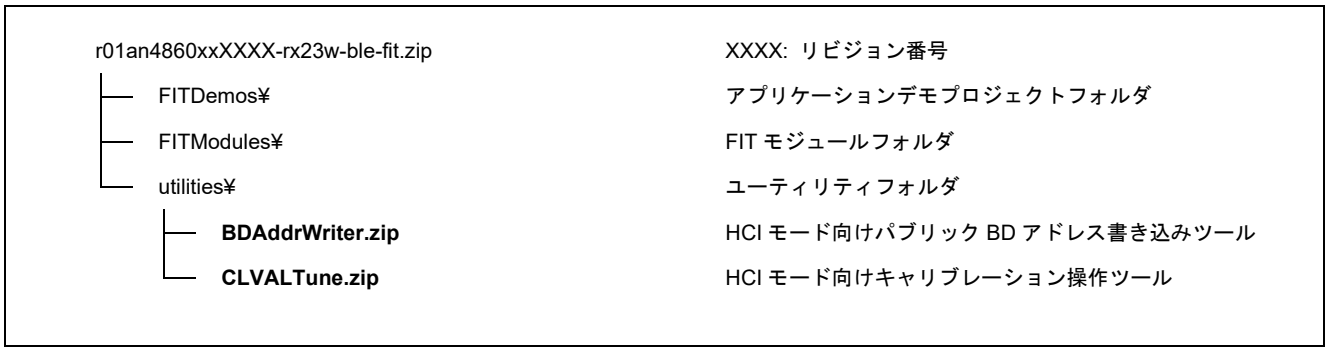


図 7-1 : HCI モード向けツールの場所

7.1 BDAddrWriter

BDAddrWriter はパブリック BD アドレス設定ツールです。HCI モードのファームウェアをボードに書き込んだ後、BDAddrWriter により、パブリック BD アドレスを設定することが可能です。詳細な使用方法については「Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205), 5.6.4.2 BDAddrWriter による書き込み」をご参照ください。

7.2 CLVALTune

CLVALTune はキャリブレーション操作プログラムです。詳細については、「RX23W グループ Bluetooth 専用クロック周波数の調整手順 (R01AN4762)」をご参照ください。

8. デモプロジェクト

Bluetooth LE の機能を使用する表 8.1 のデモプロジェクトを用意しています。デモプロジェクトには、FIT モジュールとそのモジュールが依存するモジュール（例：r_bsp）を使用する main()関数が含まれます。

表 8.1　デモプロジェクト

デモプロジェクト	説明
ble_demo_rsskrx23w_profile_client	RSSKRX23W 用の GATT Client のデモです。
ble_demo_rsskrx23w_profile_server	RSSKRX23W 用の GATT Server のデモです。
ble_demo_rsskrx23w_uart_hci	RSSKRX23W 用の HCI モードのデモです。
ble_demo_tbrx23w_profile_client	Target Board for RX23W 用の GATT Client のデモです。
ble_demo_tbrx23w_profile_server	Target Board for RX23W 用の GATT Server のデモです。
ble_demo_tbrx23w_uart_hci	Target Board for RX23W 用の HCI モードのデモです。
ble_demo_tbrx23wmodule_profile_client	Target Board for RX23W module 用の GATT Client のデモです。
ble_demo_tbrx23wmodule_profile_server	Target Board for RX23W module 用の GATT Server のデモです。
ble_demo_tbrx23wmodule_uart_hci	Target Board for RX23W module 用の HCI モードのデモです。
ble_demo_tbrx23w_FreeRTOS_multi_services	Target Board 用の Renesas FreeRTOS 上での GATT Server のデモです。

上記の FITDemo に含まれる、r_ble_rx23w のバージョンは本ドキュメントと同じバージョンとなります。その他の FIT については、RX Driver Package v1.42 (https://www.renesas.com/software-tool/rx-driver-package)に含まれるバージョンを使用しています。プロジェクト内で使用している QE for BLE のバージョンは V1.6.0 となります。

GATT Server と GATT Client デモプロジェクトはコマンドラインインタフェースをサポートしています。board と PC をシリアル接続し、PC 上のターミナルソフトからコマンドの入力やログ出力の確認が行えます。ターミナルソフトの設定は表 2.6 の内容となります。

8.1 GATT Server デモプロジェクト

GATT Server デモプロジェクトは以下のような動作を行います。

- 起動後、GATT Server デモプロジェクトは Advertising を開始し、コマンド入力待ちの状態となります。
- リモートデバイスから Scan を行うと、GATT Server デモプロジェクトは”RBLE-DEV”という名前で検出されます。

図 8-1： リモートデバイスの Scan 画面

- 接続すると、Advertising を停止します。
- リモートデバイスから GATT サービス検索を行うと、以下が検出されます。
  - LED Switch サービス(LSS, UUID : 58831926-5F05-4267-AB01-B4968E8EFCE0)
  - Switch State キャラクタリスティック(UUID : 58837F57-5F05-4267-AB01-B4968E8EFCE0)
  - LED Blink Rate キャラクタリスティック(UUID : 5883C32F-5F05-4267-AB01-B4968E8EFCE0)

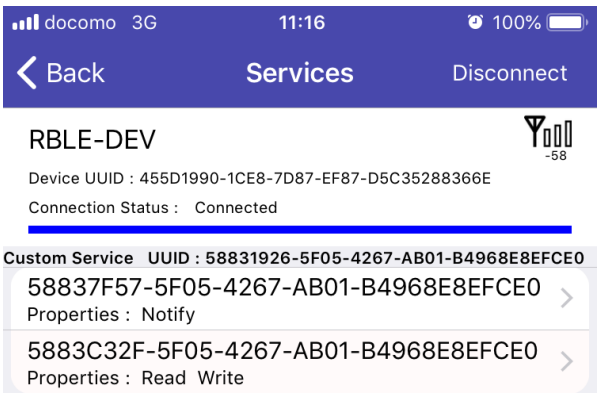


図 8-2： リモートデバイスのサービス検出画面

- gatt_db.c の gs_gatt_service 内の LED Switch サービスの設定にて、2 番目のパラメータを BLE_GATT_DB_SER_SECURITY_UNAUTH | BLE_GATT_DB_SER_SECURITY_ENC に設定すると、リモートデバイスから LED Switch サービスのキャラクタリスティックへアクセスする場合にペアリングが必要となります。0 の場合、ペアリングは必要ありません。

```
static const st_ble_gatts_db_serv_cfg_t gs_gatt_service[] =
{

    /* 省略 */

    /* LED Switch */
    {
        /* Num of Services */
        {
            1,
        },
        /* Description */
        BLE_GATT_DB_SER_SECURITY_UNAUTH | BLE_GATT_DB_SER_SECURITY_ENC,
        /* Service Start Handle */
        0x0010,
        /* Service End Handle */
        0x0015,
        /* Characteristic Start Index */
        6,
        /* Characteristic End Index */
        7,
    },
};
```

ペアリング必要

```
static const st_ble_gatts_db_serv_cfg_t gs_gatt_service[] =
{

    /* 省略 */

    /* LED Switch */
    {
        /* Num of Services */
        {
            1,
        },
        /* Description */
        0,
        /* Service Start Handle */
        0x0010,
        /* Service End Handle */
        0x0015,
        /* Characteristic Start Index */
        6,
        /* Characteristic End Index */
        7,
    },
};
```

ペアリング必要なし

図 8-3 : LED Switch サービスへのアクセス時のセキュリティ設定

- リモートデバイスから Switch State キャラクタリスティックの Notification を有効にした後、board 上の SW1 を押すと、Notification を送信します。
- リモートデバイスから LED Blink Rate キャラクタリスティックに値を書き込むと数値 x 100ms の間隔で LED の点滅が始まります。0 を書き込むと、LED が消灯します。
- 切断すると、Advertising を再開します。

GATT Server デモプロジェクトとリモートデバイスの使用例を図 8-4 に示します。

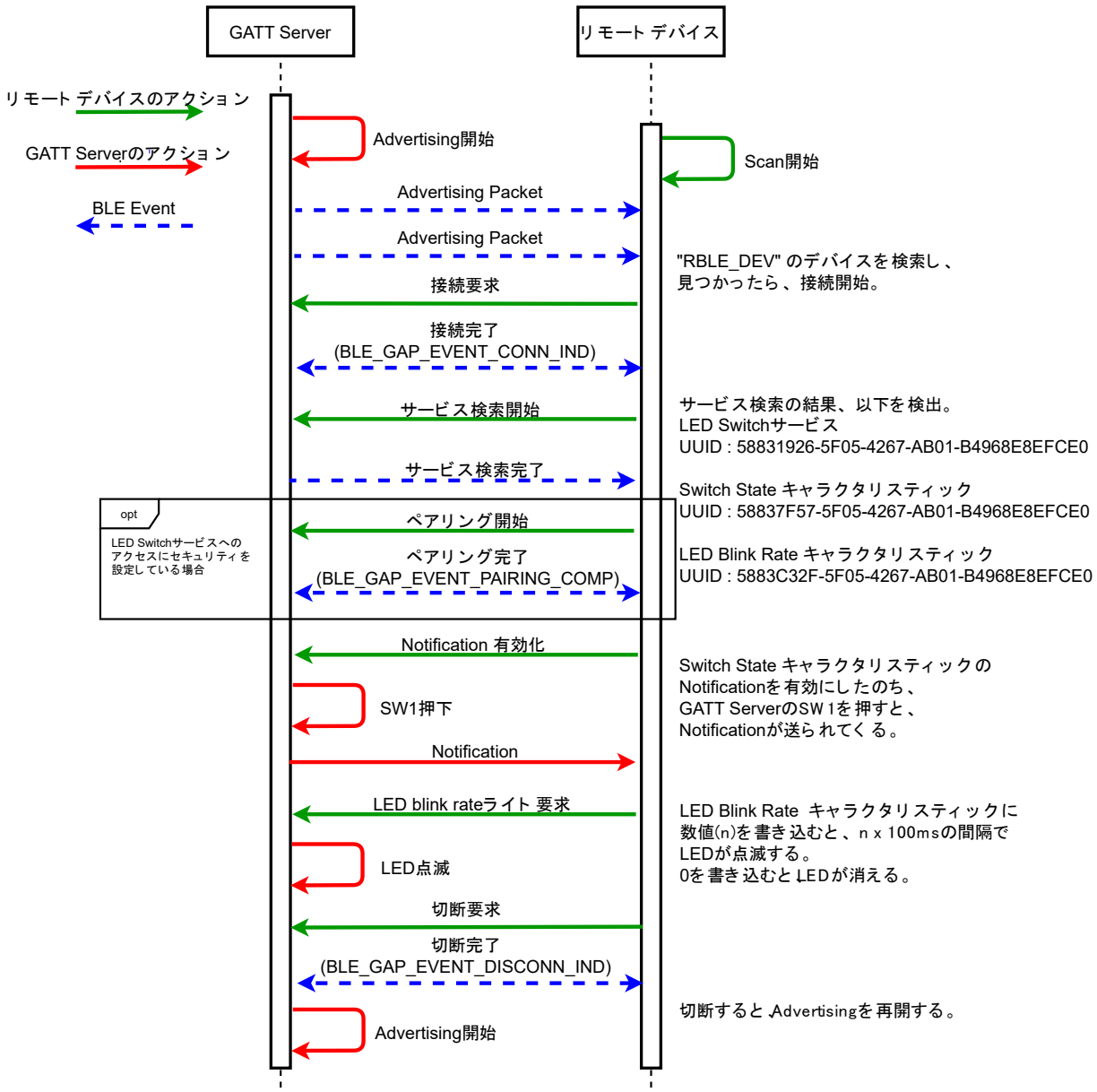


図 8-4 : GATT Server デモプロジェクトとリモートデバイスの使用例

8.2 GATT Client デモプロジェクト

GATT Client デモプロジェクトは以下のような動作を行います。

- 起動後、コマンド入力待ちの状態となります。
- GATT Server の起動を確認する場合、ターミナルソフトより gap scan コマンドを入力し、GATT Server の Advertising を受信します。gap scan コマンドの詳細については「Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205), 5.1.1.1 (2)Scan コマンド」をご参照下さい。GATT Server から Advertising を受信したら、ターミナルソフトより CTRL + C キーを入力し、Scan を停止します。
- GATT Server との接続のため、ターミナルソフトから gap conn コマンドを入力します。gap conn コマンドの詳細については「Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205), 5.1.1.1 (3)接続コマンド」をご参照下さい。
- 接続完了後、パケット長更新を行います。
- パケット長更新後、MTU 変更要求をリモートデバイスに送信します。
- リモートデバイスから MTU 変更要求への応答を受信すると、GATT サービス検索を開始します。
- GATT Server デモプロジェクトがペアリングを要求する設定 (BLE_GATT_DB_SER_SECURITY_UNAUTH | BLE_GATT_DB_SER_SECURITY_ENC 図 8-3 参照) になっている場合、LED Switch クライアントコマンド実行前にペアリングが必要となります。ペアリング開始のため、gap auth コマンドを入力します。gap auth コマンドの詳細については「Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205), 5.1.1.1 (9)セキュリティコマンド」をご参照下さい。
- 上記の処理が完了した後、下記の LED Switch クライアントコマンドが可能となります。

  - Notification の有効化／無効化
    lsc　set_switch_state_ntf　[コネクションハンドル]　[0(無効) or 1(有効)]

  - LED blink rate の書き込み
    lsc　write_led_blink_rate　[コネクションハンドル]　[blink_rate]
    [blink_rate] x 100ms の間隔で LED が点滅します。範囲は 0-255 です。
    0 を書き込んだ場合、消灯します。
- GATT Server と切断する場合、ターミナルソフトより gap disconn コマンドを入力します。gap disconn コマンドの詳細については「Bluetooth Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205), 5.1.1.1 (4)切断コマンド」をご参照下さい。

GATT Client デモプロジェクトと GATT Server デモプロジェクトの使用例を図 8-5 に示します。

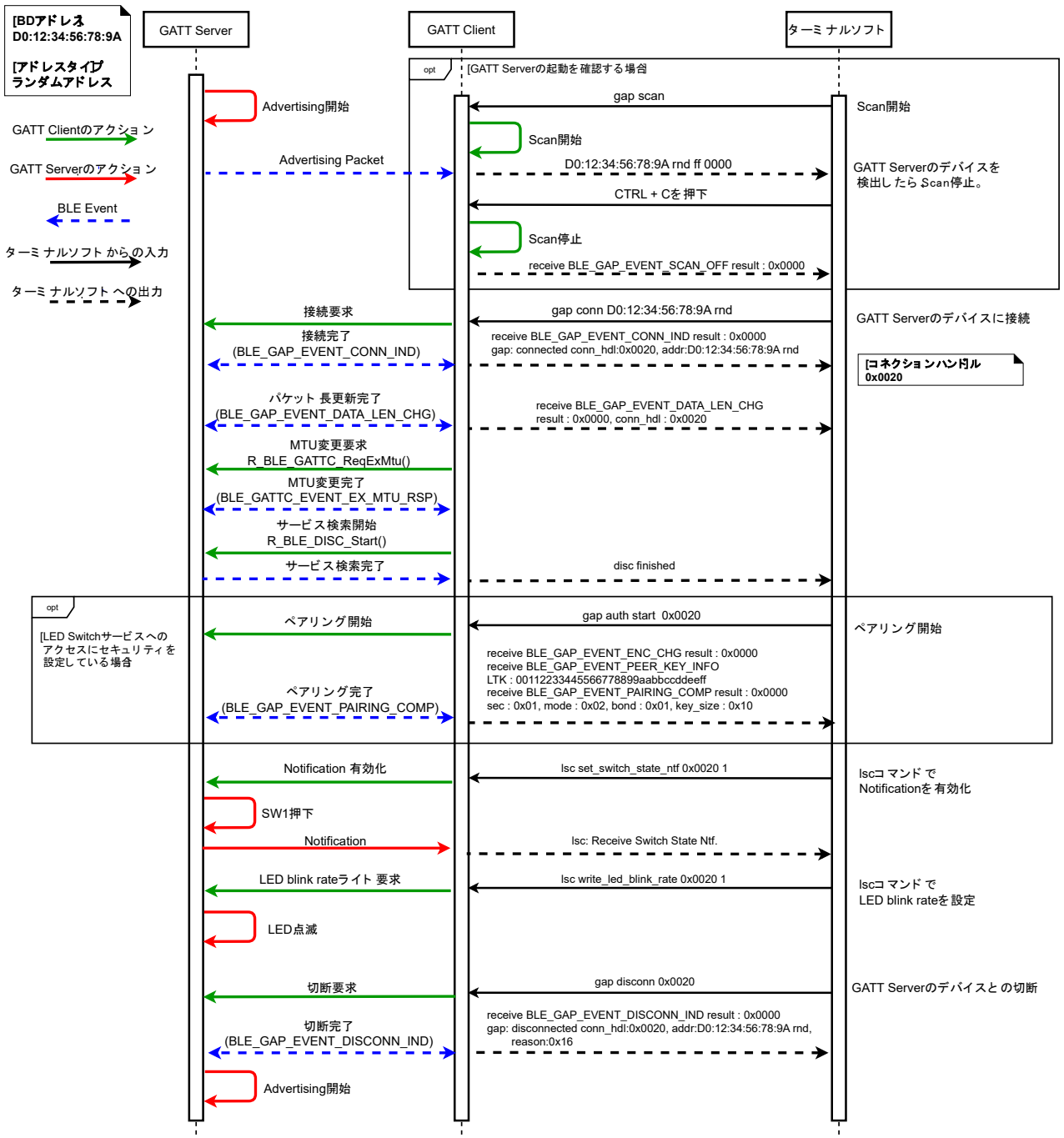


図 8-5 : GATT Client デモプロジェクトと GATT Server デモプロジェクトの使用例

8.3 Renesas FreeRTOS BLE デモプロジェクト

Renesas FreeRTOS BLE デモプロジェクトは以下のような動作を行います。このプロジェクトでは Battery Service 用のタスク(BAS タスク)、LED & SW 用のタスク(LSS タスク)と BLE タスクで構成されています。

- 起動後、Renesas FreeRTOS BLE デモプロジェクトは Advertising を開始し、コマンド入力待ちの状態となります。
- リモートデバイスから Scan を行うと、下記のように”RBLE-DEV”という名前で検出されます。

図 8-6： リモートデバイスの Scan 画面

- 接続すると、Advertising を停止します。
- リモートデバイスから GATT サービス検索を行うと、以下が検出されます。
  - Battery サービス(BAS, UUID : 180F)
  - Battery Level キャラクタリスティック(UUID : 2A19)

  - LED Switch サービス(LSS, UUID : 58831926-5F05-4267-AB01-B4968E8EFCE0)
  - Switch State キャラクタリスティック(UUID : 58837F57-5F05-4267-AB01-B4968E8EFCE0)
  - LED Blink Rate キャラクタリスティック(UUID : 5883C32F-5F05-4267-AB01-B4968E8EFCE0)

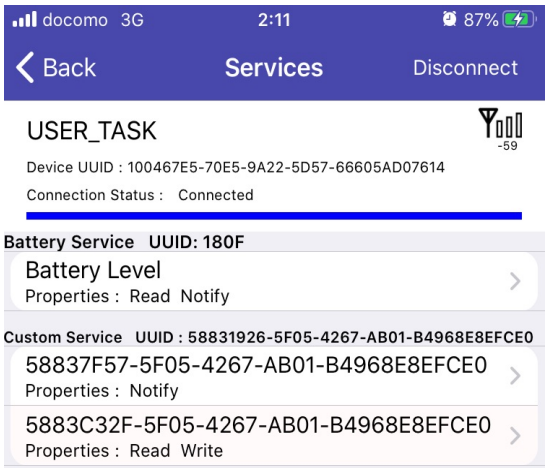


図 8-7： リモートデバイスのサービス検出画面

- Battery Level キャラクタリスティックの Notification を有効にした後、RX23W からダミーの Battery Level を 1s ごとに送信します。
- Switch State キャラクタリスティックの Notification を有効にした後、board 上の SW1 を押すと、Notification を送信します。
- リモートデバイスから LED Blink Rate キャラクタリスティックに値を書き込むと数値 x 100ms の間隔で LED の点滅が始まります。0 を書き込むと、LED が消灯します。
- 切断すると、Advertising を再開します。

Renesas FreeRTOS BLE デモプロジェクトとリモートデバイスの使用例を図 8-8 に示します。

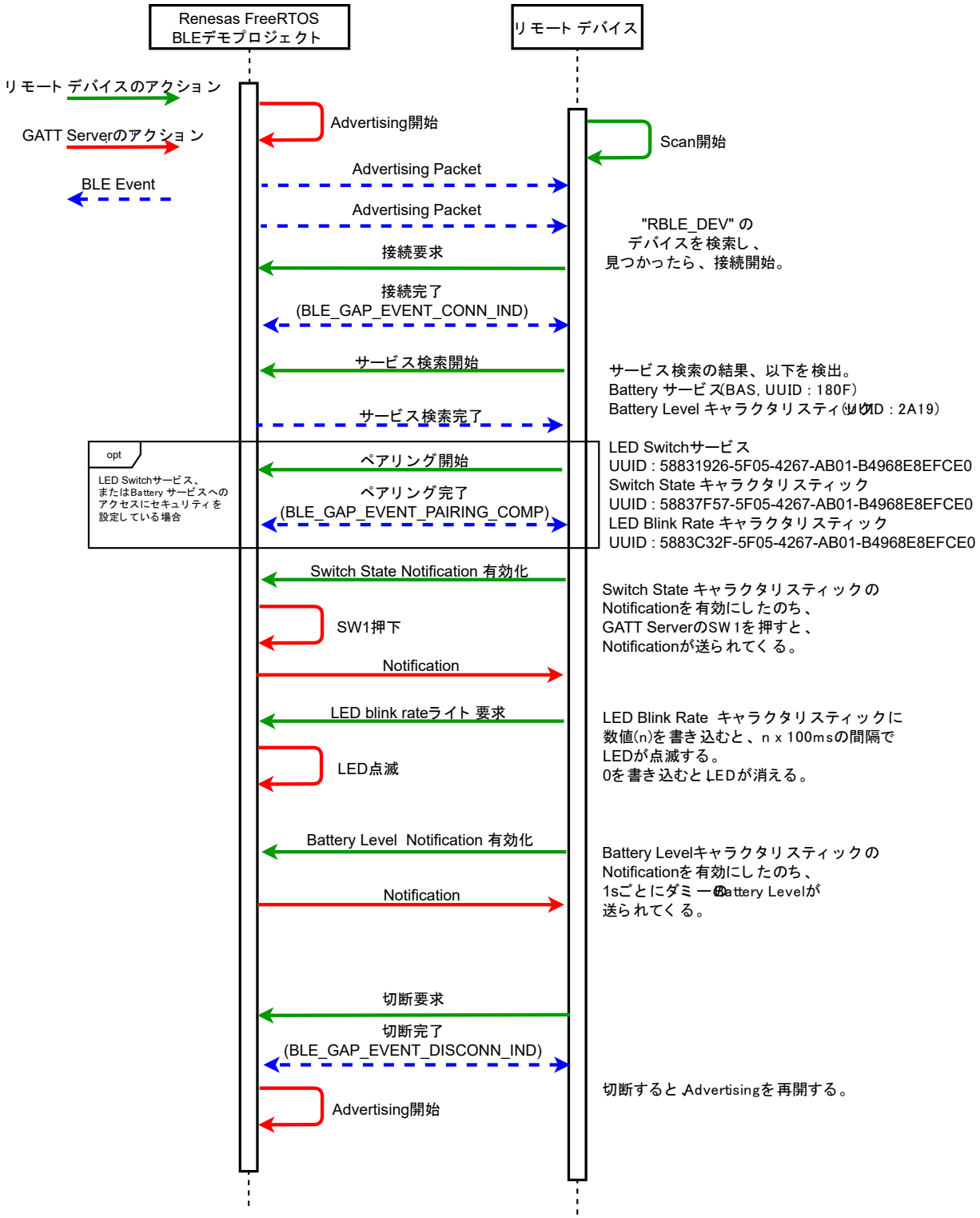


図 8-8 : Renesas FreeRTOS BLE デモプロジェクトとリモートデバイスの使用例

8.4　HCI モードデモプロジェクト

HCI モードデモプロジェクトは起動後、HCI コマンド待ちの状態となります。RF 特性評価用の HCI コマンドを入力するか、BTTS(Bluetooth Test Tool Suite : R01AN4554)と接続してご使用ください。

8.5　CS+上での動作確認

デモプロジェクト内の Renesas 共通プロジェクトファイル(rcpc ファイル)により、CS+上においてもデモプロジェクトの動作確認が可能です。動作確認に使用した CS+のバージョンは V8.05 です。
下記の手順で CS+上でデモプロジェクトをオープンします。

1. CS+を起動します。
2. “ファイル >> ファイルを開く”画面上で拡張子を“MCU Simulator Online / e² studio 用プロジェクト・ファイル(*.rcpc)”に設定し、rcpc ファイルを選択します。

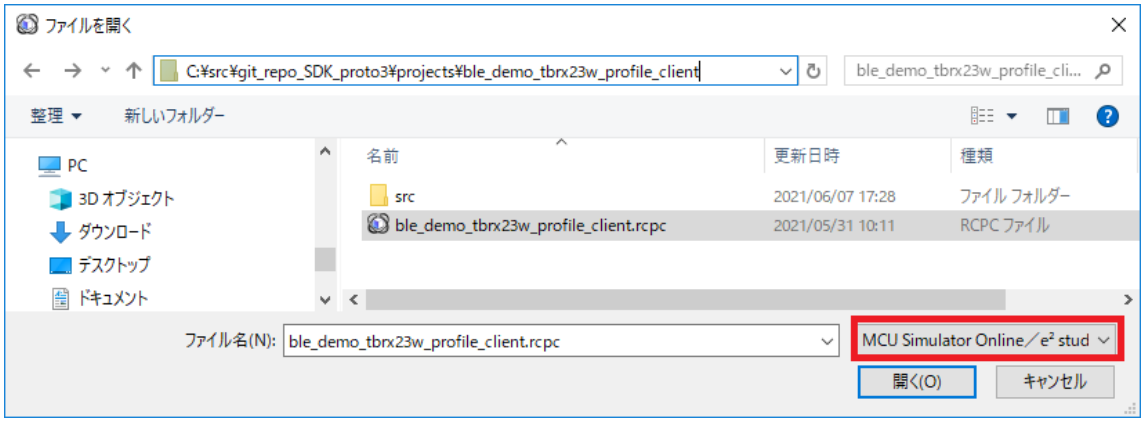


図 8-9 : rcpc ファイルのオープン

3. “プロジェクト変換設定”画面で”OK”を押します。

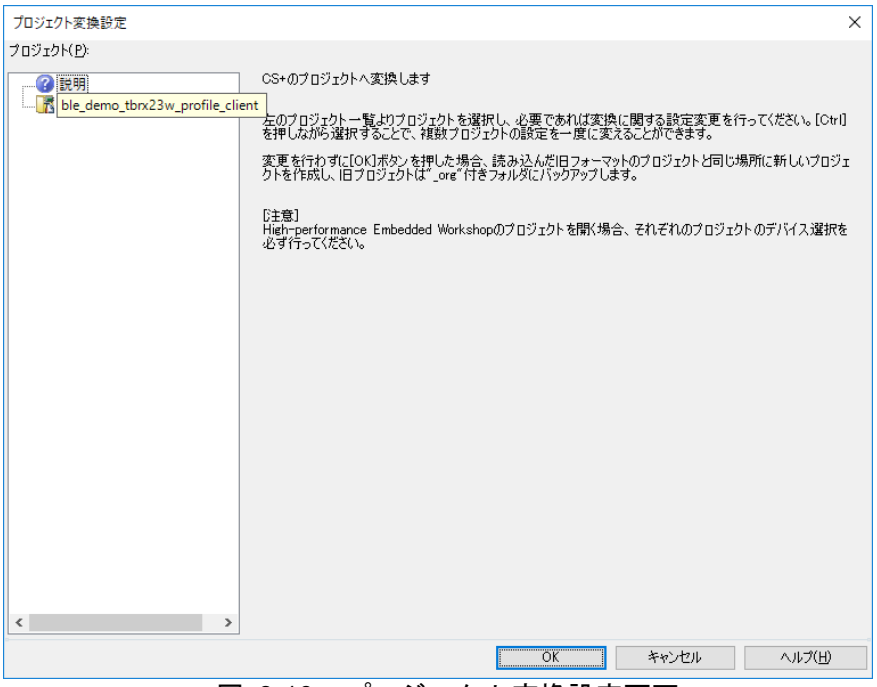


図 8-10 : プロジェクト変換設定画面

4. プロジェクトの変換が完了すると、CS+上でデモプロジェクトの動作確認が可能となります。
CS+の操作については以下 URL のユーザーズマニュアルをご参照ください。
https://www.renesas.com/software-tool/cs

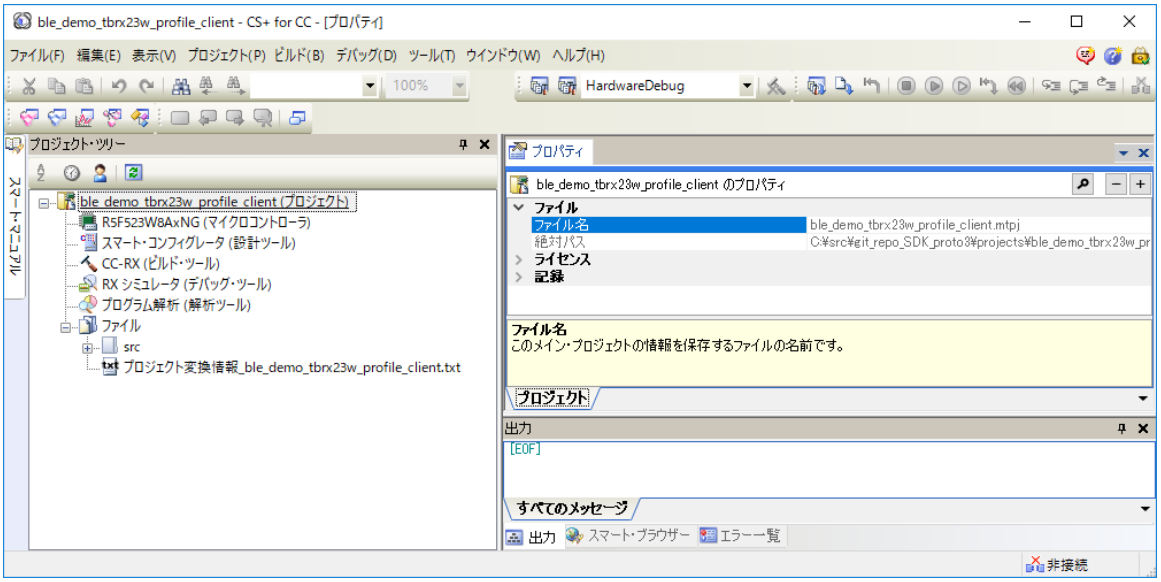


図 8-11 : CS+上でのデモプロジェクト

8.6　追加方法

e² studio にデモプロジェクトを追加する方法を以下に説明します。

(1) 4.1 新規プロジェクトの作成と 4.4 コンポーネントの追加を参照し、BLE FIT モジュール(r_ble_rx23w)を含む RX23W のプロジェクトを作成します。または、BLE FIT モジュールがコンポーネントとして含まれている RX23W の既存のプロジェクトを用意します。
(2) “コンポーネント”タブを選択し、”r_ble_rx23w”を右クリックし、”サンプルプロジェクトのダウンロードとインポート”をクリックします。

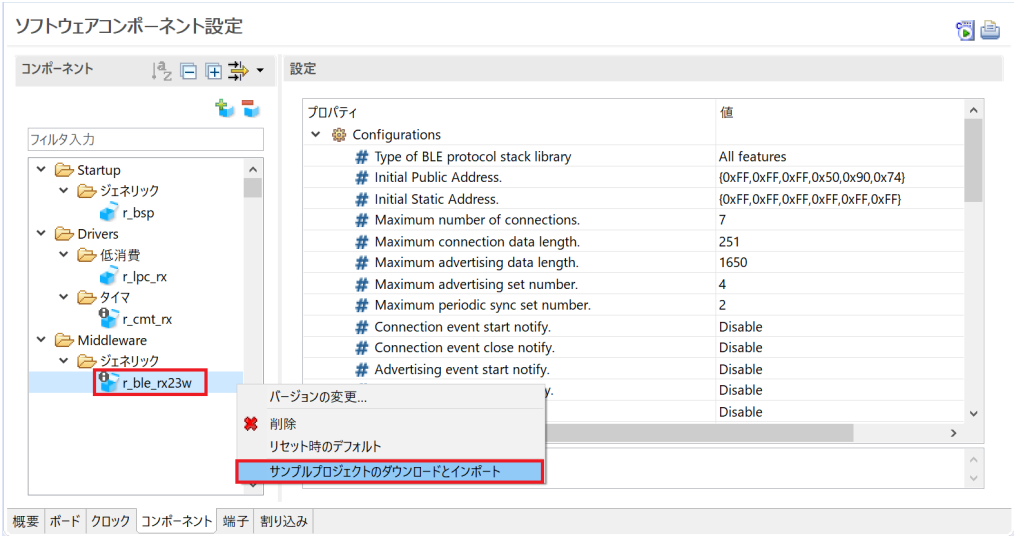


図 8-12 : サンプルプロジェクトのダウンロードとインポート(1)

(3) スマート・ブラウザー上に BLE FIT モジュールのアプリケーションノート(タイトル : RX23W グループ BLE Module Firmware Integration Technology アプリケーションノート, ドキュメント No : R01AN4860EJYYYY (YYYY はバージョン))が表示されるので、これを右クリックし、”サンプル・コード(プロジェクトのインポート)”を選択します。

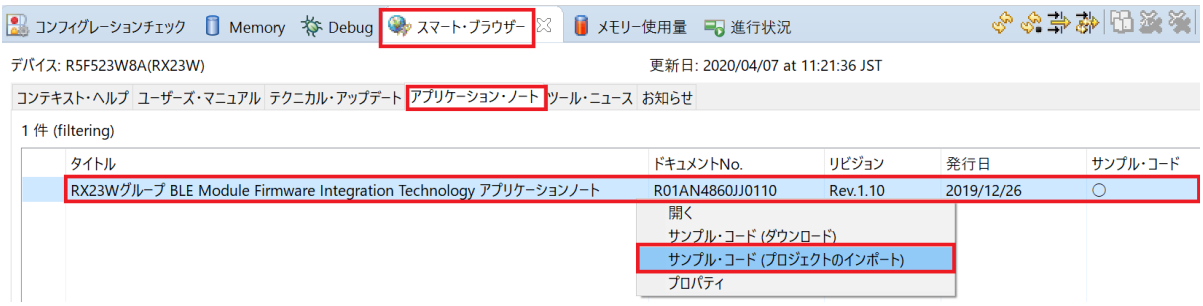


図 8-13 : サンプルプロジェクトのダウンロードとインポート(2)

(4) r01an4860xxYYYY-rx23w-ble-fit.zi(YYYY はバージョン)をダウンロードする場所を指定し、ダウンロードします。ダウンロード完了後、”インポートするパッケージの選択”ウィンドウが表示されます。ここでインポートするプロジェクトを選択します。

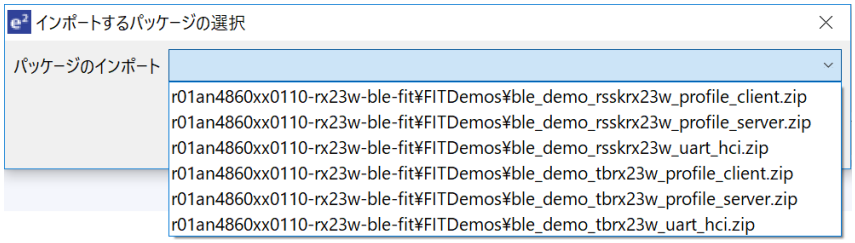


図 8-14 : サンプルプロジェクトのダウンロードとインポート(3)

(5) “インポート”ウィンドウ上で終了ボタンを押すと、デモプロジェクトがインポートされます。

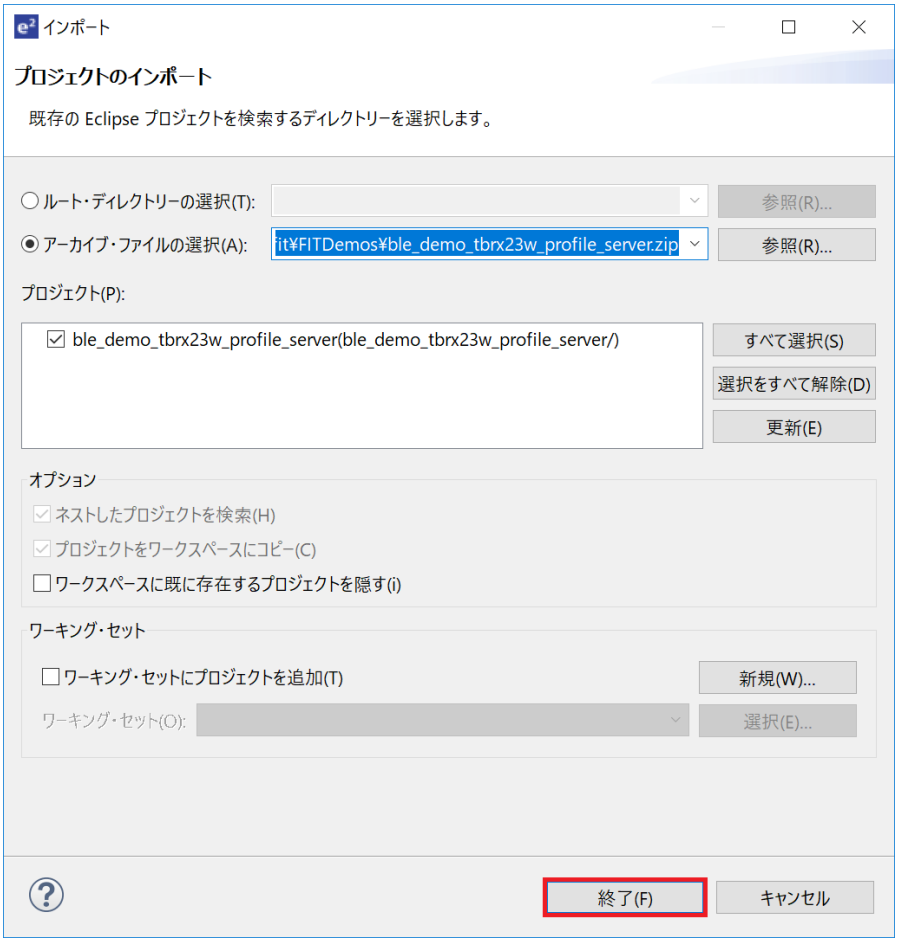


図 8-15 : サンプルプロジェクトのダウンロードとインポート(4)

9. 付録

9.1 動作確認環境

BLE FIT モジュールの動作確認環境を以下に示します。

表 9.1 動作確認環境 (Rev.2.60)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio 2024-01.1 (64bit 版) IAR Embedded Workbench for Renesas RX 4.20.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V2.08.00 IAR C/C++ Compiler for Renesas RX version 4.20.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	リトルエンディアン
モジュールのリビジョン	Rev 2.60
RTOS	FreeRTOS (kernel only) version 10.4.3-rx-1.0.6
使用ボード	Target Board for RX23W (RTK5RX23W0C00000BJ) Target Board for RX23W module (RTK5RX23W0C01000BJ) Renesas Solution Starter Kit for RX23W (RTK5523W8AC00001BJ)

9.2　トラブルシューティング

(1) Q：本FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A：FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。 本FIT モジュールを使用する場合、ボードサポートパッケージFIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

(2) Q：本FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_ble_rx23w module.」エラーが発生します。

A：追加したFIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加したFIT モジュールの対象デバイスを確認してください。

(3) Q：本FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「コンフィグ設定が間違っている場合のエラーメッセージ」エラーが発生します。

A：“r_ble_rx23w_config.h”ファイルの設定値が間違っている可能性があります。
“r_ble_rx23w_config.h” ファイルを確認して正しい値を設定してください。詳細は「2.7 コンパイル時の設定」を参照してください。

(4) Q：アプリケーションのビルドを実行すると「Could not open source file 」エラーが発生します。

A：アプリケーションのソースコードのパスが長くなっており、e^2 studioの最大パス長を超えている可能性があります。アプリケーションのソースコードのパス名が短くなるように設定してください。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2019.08.23	—	初版発行
1.01	2019.10.31	20	“2.8.6 コマンドライン”の各コマンドの説明を”Bluetooth® Low Energy プロトコルスタック基本パッケージ ユーザーズマニュアル (R01UW0205)”に移動。
		26	“4. 使用方法”の更新
		58	“7. デモプロジェクト”にデモアプリの使い方を追記。
		ライブラリ	Mesh 対応を追加。
		ライブラリ	R_BLE_VS_SetBdAddr()で BLE_VS_ADDR_AREA_REG(0x00)でのアドレス変更時のペアリングに対応。
1.10	2019.12.26	-	以下のコンパイラをサポート。 - IAR C/C++ Compiler for Renesas RX
		ライブラリ	- ボンディング情報を最大数保持した状態で再起動後にペアリングに失敗する現象の修正。 - ローカルデバイスのみがボンディング情報を削除した状態で R_BLE_GAP_StartPairing ()をコールすると、BLE_GAP_EVENT_PAIRING_COMP イベントが通知されない現象の修正。 - 再起動後にペアリング済みでアドレス解決を行った 2 台目以降のリモートデバイスとの接続が失敗する現象の修正。 - R_BLE_GAP_DeleteBondInfo()の remote パラメータに” BLE_GAP_SEC_DEL_REM_NOT_CONN(0x02)”が指定された場合に、ボンディング情報が削除されない現象の修正。 - Resolving List / White List / Periodic Advertiser List をクリア中に別の操作が要求された場合にエラーを返すように修正。 - 高負荷パケット送信中に対向デバイスから別の要求を受け付けた場合に切断する現象の修正。 - RSSI 誤差の修正。 - MTU 交換時に指定可能な最小のサイズを 23 バイトに変更。 - Balance ライブラリ使用時に advertising 実施中に接続要求パケットを発行できるように改善。
		プログラム	- 割り込み関数の宣言、セクションの宣言について BSP マクロ定義を使用。 - Advertising 用の抽象 API にローカルデバイスのアドレスタイプとして、スタティックアドレスが指定できるように修正。 - 接続パラメータ更新要求への応答処理をコマンドライン機能からアプリ層に移動。 - BLE_CFG_CMD_LINE_EN に 0 が設定された場合に cmd ディレクトリのコードを無効化。 - BLE_CFG_RF_ADV_SET_MAX が 4 以外に設定した場合の不正なメモリアクセスを修正。 - スキャンフィルタ使用時に AD type : 0 を指定された場合の処理を追加。 - 他の FIT モジュールへの依存関係を解消し、以下のコンフィギュレーションオプションのデフォルトを無効に変更。 BLE_CFG_EN_SEC_DATA (r_flash_rx) BLE_CFG_CMD_LINE_EN (r_sci_rx, r_byteq_rx) BLE_CFG_BOARD_LED_SW_EN (r_gpio_rx, r_irq_rx) BLE_CFG_DEV_DATA_DF_BLOCK (r_flash_rx) - 以下のコンフィギュレーションオプションを追加。 BLE_CFG_ABS_API_EN (抽象 API の有効／無効) BLE_CFG_SOFT_TIMER_EN (ソフトウェアタイマの有効／無効)

			BLE_CFG_MCU_LPC_EN(MCU 低消費電力機能の有効／無効) BLE_CFG_HCI_MODE_EN(HCI モードの有効／無効)
2.00	2020.7.27	7, 43	- BSP のバージョンについての注記を追加。 - CMT FIT モジュールが v4.50 以降の場合、ソースコードを変更する必要がないことを追記。
		13	- BLE_CFG_NUM_BOND 変更時の注意を追記。
		16	- 表 2.2.の各機能の説明をユーザーズマニュアルへ移行。
		29	- “4.2 BSP バージョン確認”を追加。
		31	- SCI FIT モジュール名の修正。 r_sic_rx --> r_sci_rx
		32	- FIT モジュールが見えない場合の対応方法を追記。
		35-36	- コマンドライン使用時の SCI コンフィギュレーションオプションの設定内容を追記。
		38-39	- 表 4.1 に Customer board の初期値を追記。 - IRQ コンフィギュレーションオプションの設定を追記。
		46	- “4.6.3 アプリケーションの追加”のコード変更内容を修正。
		49	- DTBL/BTBL について補足を追記。
		54-66	- “4.9 IAR 開発環境”の章の構成を変更。
		67	- “5. アプリケーションの作成方法”をアプリケーション開発者ガイド(R01AN5504)に移行。
		77	- “7. ツール”に HCI モード向けツールの場所を追記。 - “7.1 BDAddrWriter”にユーザーズマニュアル(R01UW0205)への参照を追加。 - “7.2 CLVALTune”の参照にダウンロード番号を追記。
		82-83	- GATT Client デモプロジェクトの説明と図 8.5 を修正。
		87-88	- “8.5 追加方法”を Smart Configurator からダウンロード・インポートする方法に変更。
		90	- “9.2 (1)”の回答から FIT コンフィグレータによるプロジェクトへの追加方法を削除。
		全体	- コードサンプルを e^{2}studio のエディタ表示に変更。 - API 仕様書名の修正。 r_ble_spec.chm --> r_ble_api_spec.chm
		ライブラリ	- Scan チャネルマップ API を追加。 - Balance/Compact での RF イベント通知機能を修正。 - R_BLE_VS_SetTxPower()による送信パワー変更の修正。 - Long Characteristic の書き込み完了通知を修正。 - 不正なパケットに対する脆弱性の問題を修正。 - Slave Latency に関する問題を修正。 - ボンディングに関する問題を修正。
		プログラム	- Renesas FreeRTOS 対応を追加。 - ソフトウェアスタンバイに遷移する条件の見直し。 - GATT 通信の Notification, Indication の時の CCCD の確認方法を変更。
2.10	2020.12.24	6	- BLE FIT モジュールのソフトウェア構成図の修正
		ライブラリ	- R_BLE_Execute()内の API 実行処理と RF ウェイクアップ割り込みが限定的なタイミング(数サイクル)で競合した場合に、RF ウェイクアップ処理が正常に行えない現象の修正。 - Scan 実行中に Advertising の開始・停止を繰り返した場合に、タイミングによっては Scan が継続できなくなる現象の修正。 - ソフトウェアタイマの R_BLE_TIMER_UpdateTimeout()を連続実行すると指定時間でタイムアウトしなくなる現象の修正。

			- Central(Master)が RPA を使用しない場合に、RPA を使用している Peripheral(Slave)に対して RPA 解決できる条件でも接続できない現象の修正。 - QE for BLE 上で Included Service に Encryption をつけると、Included Service discovery 時に Insufficient Authentication エラーを返してしまう現象の修正。

		プログラム	- 抽象 API の Scan, 接続要求時にローカルデバイスのアドレスタイプを指定できるように変更。
			- FreeRTOS 使用時にソフトウェアタイマのコールバックで BLE タスクをウェイクアップするように修正。
			- セキュリティデータ管理のボンディング情報の上書き処理を修正。
			- コマンドライン機能に以下を追加。
			  - adv, scan, conn, priv コマンドにローカルデバイスのアドレスタイプを示すオプションを追加。
			  - adv, scan, conn コマンドにホワイトリストを指定するオプションを追加。
			  - adv コマンドに advertising set を削除するオプションを追加。
			  - priv コマンドに Resolving List の登録を削除するオプションを追加。

2.11	2021.3.30	-	- RX23W モジュール(R5F523W8CxLN, R5F523W8DxLN)をサポート。
		8	- RX23W モジュール選択時の BSP、GPIO FIT のバージョンについて追記。
		12	- RX23W モジュール使用時の BLE_CFG_RF_CLVAL の設定値追加。 - Target Board 使用時の BLE_CFG_RF_DDC_EN の設定値を追加。
		18	- CCRX の C++を使用する場合は、ロガーがサポート外となることを追記。
		20	- gap コマンド,vs コマンドを使用時の注記を追加。 - 2.8.7 LED and Switch 制御を追加。
		26	- RX23W モジュール向けのプロジェクトを作成する場合は、e^{2}studio 2021-01 以降のバージョンを使うことを追記。
		27	- 図 4.4 に RX23W モジュールの型番の情報を追記。
		30	- RX23W モジュール選択時の BSP のバージョンについて追記。
		32	- BLE Profile Creation, r_ble_qe_utility を選択するコンポーネントに追加。
		33-34	- ダウンロードした FIT モジュールが見つからない場合に変更するスマート・コンフィグレータの設定を追記。
		78	- BLE FIT APN の URL を変更。 - パッケージ名を変更。
		79	- 表 8.1 に RX23W モジュールの FITDemo を追加。 - FITDemo に含まれる FIT のバージョンを追記。

		92	- 表 9.1 に以下を追加。
			  - 統合開発環境に e^2 studio 2021-01 (64bit 版)を追加。
			  - 使用ボードに Target Board for RX23W module、Renesas Solution Starter Kit for RX23W を追加。

		ライブラリ	- スタティックアドレスを途中で変えた場合に advertising に反映されるように変更。 - RPA 機能使用時の Scan Request 受信イベント (BLE_GAP_EVENT_SCAN_REQ_RECV)内に含まれる scanner のアドレスを修正。

			- Coded PHY のみで scan を実行後、1M PHY のみで scan を実行した場合に BLE_GAP_EVENT_ADV_REPT_IND イベントが通知されないことがある現象を修正。 - ノンマスカブル割り込みステータスレジスタ(NMISR)が 0x00 でない場合に R_BLE_Execute() API から戻らない現象の修正。

		プログラム	- CCRX の C++使用時に可変長マクロを使用しないように変更。
			- コマンドライン機能の以下を修正。
			  - passkey entry によるペアリング時に 0 から始まる passkey を 10 進数として識別できるように修正。
			  - gap scan コマンドのフィルタに 0x00 を含めて設定できるように修正。

2.20	2021.6.30	88	- “8.5 CS+上での動作確認”を追加。
		ライブラリ	- Impersonation in the Passkey Entry Protocol (CVE-2020-26558)への対応
2.30	2021.10.15	14	- 表 2.4 の BLE_CFG_MCU_MAIN_CLK_KHZ の PLL Circuit 使用時の設定範囲を修正。
		52	- 2)ライブラリ選択バッチファイルの登録のビルド前のステップに設定するパスを修正。
		ライブラリ	- 特定の条件において、BLE 接続動作開始時に BLEIRQ 割り込みが発生し続け、メインルーチンが動作しない問題の修正。 - PLL 回路有効時のクロックの範囲チェックを修正。
2.31	2022.4.15	-	- BLE Profile Creation , r_ble_qe_utility についての記載を削除。
		8	- BLE FIT が使用する RX23W のハードウェア機能の表を追加。
		11	- BLE_CFG_RF_CONN_MAX の説明を修正。
		15	- BLE_CFG_NUM_BOND の説明を修正。
		22	- “2. 9 Flash Memory Protection”を”2.9 デバイス固有データ”に変更。
		29	- Renesas FreeRTOS の新規プロジェクトの作成の説明を更新。
		70	- R_BLE_CLI_Printf はどちらかのタスクからのみコール可能であることを追記。
		79	- パッケージの URL の修正。 - 図 7.1 の zip ファイル名の修正。
		80	- FITDemo が QE for BLE v1.40 を使用していることを明記。
		93	- 表 9.1 から e^2 studio V7.8.0 (32bit 版)を削除。
		ライブラリ	- 初期化時の動作安定化のため R_BLE_Open API 内のタイミングを調整。 - R_BLE_GAP_UpdConn のパラメータチェックを修正。
		プログラム	- app_lib の軽微な修正。 - FITDemo が e^2 studio 2022-04 以降に対応。
2.40	2022.6.30	10	- “2.3.1 ペリフェラルロールでの接続パラメータに関する制限事項”を追加。
		20	- 抽象 API のコードを変更しないよう追記。
		ライブラリ	- R_BLE_Open の初期化安定待ち時間の調整。 - Connection Latency が有効な接続におけるペリフェラル通信動作の安定化。
		プログラム	- BD アドレスを指定したボンディング情報削除後にデータフラッシュ内のボンディング情報が削除できない問題の改修。

2.50	2022.12.27	-	- BLE FIT プロファイル共通部を BLE FIT から QE for BLE に移行。
		8	- 表 2.1 に割り込み優先レベルについて追記。
		10	- RF スリープモードに遷移できなくなる問題を回避するための Supervision Timeout 短縮補正について追記。
		52	- 前バージョンから BLE FIT のみ更新した場合、プロジェクトのフォルダ構成が変わり、ビルドエラーが発生するため、“4.6.4 プロファイル共通部のコード生成“を追加。
		57	- ライブラリ選択バッチファイルのファイルパスの区切り文字を変更。
		ライブラリ	- Peripheral 動作の時、特定の条件で切断が発生すると RF スリープモードに遷移できなくなる問題の修正。 - R_BLE_GAP_SetPhy API で Coded PHY に変更後、Coded PHY の Coding scheme(S=2/S=8)の切り替えが行えないことがある問題を修正。 - MTU よりサイズ大きい可変長キャラクタリスティックの read /write に対する GATT Database の処理を修正。 - エラー発生時の prepare write queue の処理の修正。 - BLE_CFG_LIB_TYPE で Balance または Compact を指定した場合の White List に登録可能なエントリ数の修正。 - R_BLE_GATTC_ReadCharUsingUuid のイベント ID の修正。 - R_BLE_GAP_CreateConn API で White List フィルタを有効にした場合に、BLE_GAP_EVENT_CONN_IND イベントパラメータの local_rpa に使用していない値を通知することがある問題を修正。 - Peripheral 動作での接続後、接続イベントが実行できず 41 秒後に BLE_GAP_EVENT_DISCONN_IND イベントが通知されることがある問題を修正。 - 接続パラメータ変更要求に対してリモートデバイスが拒否した場合のエラーコードを修正。
		プログラム	- app_lib から profile_cmn, discovery を削除。 - ライブラリ選択シェルスクリプトを追加。 - セキュリティデータ管理のエラー処理の修正。
2.60	2024.5.17	11	- “2.3.2 Resolvable Private Address (RPA) 機能を有効にした場合の制限事項”を追加。
		27	- “2.12 FIT モジュールの追加方法”の記載内容を更新。
		28	- “2.13 for 文、while 文、do while 文について”を追加。
		ライブラリ	- “2.3.2 Resolvable Private Address (RPA) 機能を有効にした場合の制限事項”に関する問題を修正。
		プログラム	- ソースコードの for 文、while 文、do while 文箇所に /* WAIT_LOOP */のコメントを追加。
2.61	2025.3.31	プログラム	- 免責事項を更新。
2.62	2026.3.13	パッケージ	- spdx ファイルと json ファイルをパッケージに同梱

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。