

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(Bluetooth® Low Energy)モジュールについて説明します。本モジュールは BLE 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 e²studio 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)

Renesas e²studio 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 (R01AN4553)

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)

Bluetooth® のワードマークおよびロゴは、Bluetooth SIG,Inc. が所有する登録商標であり、ルネサス エレクトロニクス株式会社はこれらのマークをライセンスに基づいて使用しています。その他の商標および登録商標は、それぞれの所有者に帰属します。

目次

| | |
|---|----|
| 1. 概要 | 4 |
| 1.1 使用方法 | 4 |
| 1.2 ソフトウェア構成 | 5 |
| 1.3 ディレクトリ/ファイル構成 | 6 |
| 2. API 情報 | 7 |
| 2.1 ハードウェア要件 | 7 |
| 2.2 ソフトウェア要件 | 7 |
| 2.3 サポートされているツールチェーン | 8 |
| 2.4 ヘッダファイル | 8 |
| 2.5 整数型 | 8 |
| 2.6 コンパイル時の設定 | 9 |
| 2.7 BLE Protocol Stack の機能 | 15 |
| 2.8 app_lib の機能 | 18 |
| 2.8.1 抽象 API | 18 |
| 2.8.2 ソフトウェアタイマ | 18 |
| 2.8.3 セキュリティデータ管理 | 18 |
| 2.8.4 プロファイル共通部 | 18 |
| 2.8.5 ロガー | 18 |
| 2.8.6 コマンドライン | 19 |
| 2.9 Flash Memory Protection | 21 |
| 2.10 コードサイズ | 23 |
| 2.11 FIT モジュールの追加方法 | 24 |
| 3. R_BLE API 関数 | 25 |
| 4. BLE FIT モジュールのプロジェクト | 26 |
| 4.1 新規プロジェクトの作成 | 26 |
| 4.2 クロック設定 | 29 |
| 4.3 コンポーネントの追加 | 30 |
| 4.4 コンフィギュレーションオプションの設定 | 32 |
| 4.4.1 BSP(r_bsp) | 32 |
| 4.4.2 コマンドライン機能 | 33 |
| 4.4.3 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) | 35 |
| 4.4.4 LED and Switch 制御機能 | 36 |
| 4.5 コード生成後の設定 | 38 |
| 4.5.1 CMT の変更 | 38 |
| 4.5.2 Customer board 用の LED and Switch 制御 | 39 |
| 4.5.3 アプリケーションの追加 | 41 |
| 4.6 リンカ設定 | 42 |
| 4.6.1 セクション配置 | 42 |
| 4.6.2 BLE Protocol Stack ライブラリ | 45 |
| 4.7 デバッグ設定 | 47 |
| 4.7.1 Flash の ID Code | 47 |

| | | |
|-------|----------------------------|----|
| 4.7.2 | 電源 | 48 |
| 4.8 | IAR 開発環境でのプロジェクト作成 | 49 |
| 5. | アプリケーションの作成方法 | 53 |
| 5.1 | main 関数 | 53 |
| 5.1.1 | main loop | 54 |
| 5.1.2 | ホストスタック、プロファイルの初期化 | 55 |
| 5.2 | コールバックと登録関数 | 57 |
| 5.3 | イベント通知機能 | 58 |
| 5.4 | GATT Database | 59 |
| 5.5 | 低消費電力状態 | 60 |
| 5.6 | Data Flash のブロック | 61 |
| 6. | ツール | 62 |
| 6.1 | BDAAddrWriter | 62 |
| 6.2 | CLVALTune | 62 |
| 7. | デモプロジェクト | 63 |
| 7.1 | GATT Server デモプロジェクト | 64 |
| 7.2 | GATT Client デモプロジェクト | 67 |
| 7.3 | HCI モードデモプロジェクト | 69 |
| 7.4 | 追加方法 | 70 |
| 8. | 付録 | 72 |
| 8.1 | 動作確認環境 | 72 |
| 8.2 | トラブルシューティング | 73 |
| | 改訂記録 | 74 |
| | 製品ご使用上の注意事項 | 76 |
| | ご注意書き | 77 |

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 BLE 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、「2.11 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。

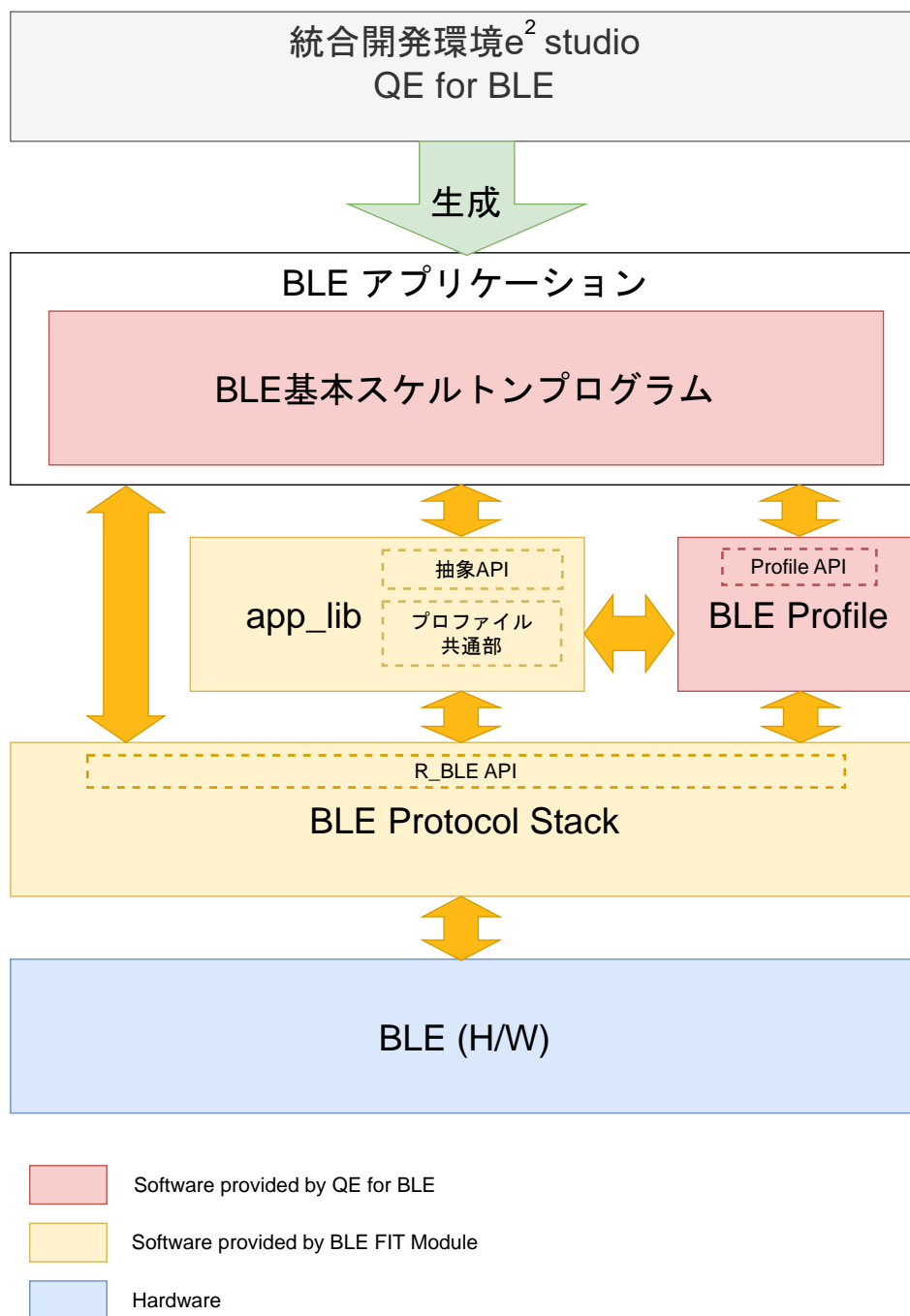


図 1.1 : BLE FIT モジュールのソフトウェア構成

BLE FIT モジュールは BLE Protocol Stack と app_lib から構成されています。

BLE Protocol Stack が提供する R_BLE API 関数をコールすることにより、BLE アプリケーションは BLE 機能の使用が可能となります。

app_lib は BLE 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 BLE 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。

ルネサスでは、QE for BLE を使用した、BLE 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 | | r01an4860ej0110-rx23w-ble.pdf | アプリケーションノート(英語) | |
| | | jp | | r01an4860jj0110-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 | |
| | | lib_ble_ps_ccrx_a.lib | | | BLE Protocol Stack(ALL Features) | |
| | | lib_ble_ps_ccrx_b.lib | | | BLE Protocol Stack(Balance) | |
| | | lib_ble_ps_ccrx_c.lib | | | BLE 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) | |
| | ref | r_ble_rx23w_config_reference.h | | | コンフィグレーションリファレンスファイル | |
| | src | app_lib | abs | | 抽象 API(GAP) | |
| | | | board | | ボード上の LED, SW 制御 | |
| | | | cli | | コマンドライン(入出力) | |
| | | | cmd | | コマンドライン(コマンド) | |
| | | | discovery | | プロファイル共通部(discovery 機能) | |
| | | | logger | | ロガー | |
| | | | profile_cmnn | | プロファイル共通部 | |
| | | | 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 コンフィグレーション制御部 |
| | | | 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 | | | コンフィグレーションオプションファイル | | |

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

ご使用になる MCU が以下の機能をサポートしている必要があります。

BLE

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)
- CMT (r_cmt_rx, version 4.10 以降)
※BLE Protocol Stack は CMT2, CMT3 を使用します。そのため、「4.5.1 CMT の変更」に説明する CMT の変更を行ってください。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、コンフィグ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)
- IRQ (r_irq_rx)

2.3 サポートされているツールチェーン

BLE FIT モジュールは「8.1 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。

2.4 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。

2.5 整数型

このプロジェクトは ANSI 99 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。

2.6 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 2.1: 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" | BLE 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 BLE の機能については「2.7 BLE 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" | 同時最大接続数を設定します。 "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" | 最大アドバタイジングデータサイズ(バイト)を設定します。 "31"~"1650"の範囲で設定してください。 BLE 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" | 最大アドバタイジングセット数を設定します。 "1"~"4"の範囲で設定してください。 BLE 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。 BLE 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" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>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。</p> |
| BLE_CFG_EVENT_NOTIFY_CONN_CLOSE ※デフォルト値は"0" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>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。</p> |
| BLE_CFG_EVENT_NOTIFY_ADV_START ※デフォルト値は"0" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>下記のタイミングで通知します。</p> <ul style="list-style-type: none"> ➢ 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 <p>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。</p> |
| BLE_CFG_EVENT_NOTIFY_ADV_CLOSE ※デフォルト値は"0" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>下記のタイミングで通知します。</p> <ul style="list-style-type: none"> ➢ 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 <p>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。</p> |
| BLE_CFG_EVENT_NOTIFY_SCAN_START ※デフォルト値は"0" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>Interval=Window の場合、通知されません。</p> <p>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。</p> |

| 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" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>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。</p> |
| BLE_CFG_EVENT_NOTIFY_INIT_START ※デフォルト値は"0" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>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。</p> |
| BLE_CFG_EVENT_NOTIFY_INIT_CLOSE ※デフォルト値は"0" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>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。</p> |
| BLE_CFG_EVENT_NOTIFY_DS_START ※デフォルト値は"0" | <p>RF_DEEP_SLEEP start 通知機能の有効／無効を設定します。</p> <p>0: Disable 1: Enable</p> |
| BLE_CFG_EVENT_NOTIFY_DS_WAKEUP ※デフォルト値は"0" | <p>RF_DEEP_SLEEP wakeup 通知機能の有効／無効を設定します。</p> <p>0: Disable 1: Enable</p> |
| BLE_CFG_RF_CLVAL ※デフォルト値は"6" | <p>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」を参照してください。</p> |
| BLE_CFG_RF_DDC_EN ※デフォルト値は"0" | <p>RF 部の DC-DC の有効／無効を設定します。</p> <p>0: Disable 1: Enable</p> |
| BLE_CFG_RF_EXT32K_EN ※デフォルト値は"0" | <p>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。</p> <p>0: RF_LOCO 使用 1: 外部 32.768kHz 使用</p> <p>1にする場合、Smart Configurator のクロック設定画面(図 4.7)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。</p> |

| コンフィギュレーションオプション (r_ble_rx23w_config.h) | |
|---|---|
| BLE_CFG_RF_MCU_CLKOUT_PORT ※デフォルト値は"0" | <p>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。</p> <p>0: PE3 1: PE4</p> <p>BLE_CFG_RF_EXT32K_EN が 0 の場合、この値は無視されます。</p> |
| BLE_CFG_RF_MCU_CLKOUT_FREQ ※デフォルト値は"0" | <p>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。</p> <p>0: MCU CLKOUT frequency 32.768kHz 1: MCU CLKOUT frequency 16.384kHz</p> <p>BLE_CFG_RF_EXT32K_EN が 0 の場合、この値は無視されます。</p> |
| BLE_CFG_RF_SCA ※デフォルト値は"250" | <p>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。</p> <p>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。</p> |
| BLE_CFG_RF_MAX_TX_POW ※デフォルト値は"1" | <p>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。</p> <p>0: max +0dBm 1: max +4dBm</p> |
| BLE_CFG_RF_DEF_TX_POW ※デフォルト値は"0" | <p>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。</p> <p>BLE_CFG_RF_MAX_TX_POW が 0(0dBm)の場合、 BLE_CFG_RF_DEF_TX_POW は以下ようになります。</p> <p>0(High) : 0dBm 1(Mid) : 0dBm 2(Low) : -18dBm</p> <p>BLE_CFG_RF_MAX_TX_POW が 1(+4dBm)の場合、 BLE_CFG_RF_DEF_TX_POW は以下ようになります。</p> <p>0(High) : +4dBm 1(Mid) : 0dBm 2(Low) : -20dBm</p> |
| BLE_CFG_RF_CLKOUT_EN ※デフォルト値は"0" | <p>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。</p> <p>0: No output 5: 4MHz output 6: 2MHz output 7: 1MHz output</p> |

| コンフィギュレーションオプション (r_ble_rx23w_config.h) | |
|--|---|
| 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"12500"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。 |
| BLE_CFG_NUM_BOND ※デフォルト値は"7" | 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。 |
| 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。 |
| 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。 |

| コンフィギュレーションオプション (r_ble_rx23w_config.h) | |
|---|--|
| BLE_CFG_CMD_LINE_CH ※デフォルト値は"1" | <p>コマンドライン機能で使用する SCI のチャンネルを設定します。 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 のチャンネルを有効にしてください。</p> <p>BLE_CFG_CMD_LINE_EN が 0 の場合、この値は無視されます。</p> |
| BLE_CFG_BOARD_LED_SW_EN ※デフォルト値は"0" | <p>ボードの LED と SW の制御コードの有効／無効を設定します。</p> <p>0: Disable 1: Enable</p> <p>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。</p> |
| BLE_CFG_BOARD_TYPE ※デフォルト値は"0" | <p>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。</p> <p>0: Customer board 1: Target Board 2: RSSK 3: Evaluation Board</p> |
| BLE_CFG_LOG_LEVEL ※デフォルト値は"3" | <p>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。</p> <p>0: disable 1: Error 2: Error & Warning 3: Error & Warning & Debug</p> |
| BLE_CFG_ABS_API_EN ※デフォルト値は"1" | <p>抽象 API の有効／無効を設定します。</p> <p>0: Disable 1: Enable</p> |
| BLE_CFG_SOFT_TIMER_EN ※デフォルト値は"1" | <p>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。</p> <p>0: Disable 1: Enable</p> <p>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。</p> |
| BLE_CFG_MCU_LPC_EN ※デフォルト値は"1" | <p>MCU の低消費電力機能の有効／無効を設定します。</p> <p>0: Disable 1: Enable</p> |
| BLE_CFG_HCI_MODE_EN ※デフォルト値は"0" | <p>HCI モードでの起動を設定します。</p> <p>0: 通常起動 1: HCI モードでの起動</p> |

2.7 BLE Protocol Stack の機能

BLE FIT モジュールの BLE 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 2.2 に BLE Protocol Stack の各タイプがサポートする機能を示します。

表 2.2 : BLE Protocol Stack の各タイプがサポートする機能

| BLE Feature | BLE 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 |

- LE 2M PHY
2 Msym/s PHY での BLE 通信をサポートします。
- LE Coded PHY
Coded PHY での BLE 通信をサポートします。
1M PHY, 2M PHY より長い距離での通信が可能となります。
- LE Advertising Extensions
Advertising の拡張機能です。本機能の特徴は以下の通りです。
 - 4 つまでの独立した Advertising の同時実行が可能。
(コンフィグレーションオプションの BLE_CFG_RF_ADV_SET_MAX で同時実行する Advertising の数を設定します。)
 - Advertising Data/Scan Response Data のサイズを最大 1650 バイトまで拡張。
(コンフィグレーションオプションの BLE_CFG_RF_ADV_DATA_MAX で最大サイズ(バイト)を設定します。)
 - Periodic Advertising が可能。
- LE Channel Selection Algorithm #2
Version 5.0 で追加されたホッピングチャネルを選択するアルゴリズムにより、チャネルを選択する機能です。
- High Duty Cycle Non-Connectable Advertising
最小のインターバルが 20 msec までの non-connectable の Advertising をサポートする機能です。
- LE Secure Connections
Elliptic curve Diffie-Hellman 鍵共有方式(ECDH)により、passive eavesdropping に対応したペアリングをサポートします。
- Link Layer privacy
定期的に Bluetooth デバイスアドレスを変更することにより、他の BLE デバイスからの追跡を回避する機能です。
- Link Layer Extended Scanner Filter policies
Resolvable private address に対応した Scan Filter をサポートします。
- LE Data Packet Length Extension
BLE データ通信のパケットサイズを拡張する機能です。
251 バイトまで拡張することが可能です。
- LE L2CAP Connection Oriented Channel Support
L2CAP の credit based flow control チャネルを使った通信をサポートする機能です。
- Low Duty Cycle Directed Advertising
既知のデバイスとの再接続用に Low Duty Cycle の advertising をサポートする機能です。
- LE Link Layer Topology
Master, Slave の両方のロールをサポートし、あるリモートデバイスとの接続では Master として、別のリモートデバイスとの接続では Slave として動作できる機能です。

- LE Ping
リンク暗号化後、MIC を含むパケットの送信要求により、リンクが維持されているかどうかをチェックする機能です。
- GAP Role
GAP Role として、以下をサポートします。
 - Central : 接続要求を Peripheral デバイスに送信するデバイスです。
 - Peripheral : Central からの接続要求を受け入れ、接続を確立するデバイスです。
 - Observer : Advertising をスキャンするデバイスです。
 - Broadcaster : Advertising を送信するデバイスです。
- GATT Role
GATT Role として、以下をサポートします。
 - Server : GATT Database にサービスが提供する Characteristic を用意し、Client からの要求に応答するデバイスです。
 - Client : Server が提供するサービスに対して、要求を発行するデバイスです。
- 32-bit UUID Support in LE
32-bit の UUID をサポートします。GATT で使用する場合は、128-bit に拡張して使用します。

2.8 app_lib の機能

BLE 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。

2.8.1 抽象 API

抽象 API は BLE 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_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。

2.8.2 ソフトウェアタイマ

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 のチャンネルは FIT により、動的に割り当てられます。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。

2.8.3 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。

2.8.4 プロファイル共通部

プロファイル共通部は BLE Profile 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(R01AN4553)をご参照ください。

2.8.5 ロガー

メッセージを出力します。出力レベルはコンフィグレーションオプションの BLE_CFG_LOG_LEVEL で設定します。

2.8.6 コマンドライン

シリアルから入力したコマンドを介して BLE 機能を使用する機能です。

コマンドラインは 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。

使用するチャン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ンネルを選択します。
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ンネルを指定します。

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 2.3：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 2.4 に示します。

表 2.4: 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 | BLE Protocol Stack のリセットを行います。 |
| | close | BLE Protocol Stack を停止します。 |

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)」をご参照ください。

2.9 Flash Memory Protection

BLE 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 2.5 に記載します。

表 2.5: 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。

```
format hex
area user flash
address 0xFFFF7800
size 20
index data
000001 1000000078563412060504030201D6D5D4D3D2D1
000002 10000000785634120C0B0A090807E6E5E4E3E2E1
```

BLE_CFG_DEV_DATA_CF_BLOCK で指定したブロックの ROM アドレス

マジックナンバー (固定値)

パブリック BD アドレス (任意)

ランダム BD アドレス (任意)

データ長 (固定値)

インデックス

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。

- 4.4.1 BSP(`r_bsp`)
- 4.7.1 Flash の ID Code

2.10 コードサイズ

表 2.6 に BLE FIT モジュールの BLE Protocol Stack のコードサイズを示します。表 2.6 は「8.1 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 BLE 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 2.6 : BLE Protocol Stack のコードサイズ

| デバイス | コンパイラ | 分類 | BLE Protocol Stack のタイプ | | |
|--------------------------------|-------|-----|-------------------------|----------|----------|
| | | | All Features | Balance | Compact |
| RX23W | CC-RX | ROM | 187K バイト | 146K バイト | 130K バイト |
| | | RAM | 38K バイト | 23K バイト | 23K バイト |
| コンフィグレーションオプション <u>共通</u> : | | | | | |
| BLE_CFG_RF_CONN_MAX 7 | | | | | |
| BLE_CFG_RF_CONN_DATA_MAX 251 | | | | | |
| BLE_CFG_NUM_BOND 7 | | | | | |
| <u>All Features</u> : | | | | | |
| 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 | | | | | |
| <u>Balance</u> : | | | | | |
| BLE_CFG_LIB_TYPE 1 | | | | | |
| <u>Compact</u> : | | | | | |
| BLE_CFG_LIB_TYPE 2 | | | | | |

[Note]

上記のコードサイズには app_lib と BLE Profile は含まれていません。

CCR、IAR の開発環境共に CCR でコンパイルした BLE Protocol Stack ライブラリを使用します。

2.11 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、Smart Configurator を使用した(1)、(3)の追加方法を推奨しています。

- (1) e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator 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「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。
- (2) e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² studio の FIT Configurator 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 e² 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 Smart Configurator 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 Smart Configurator 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「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」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。

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_spec.chm)をご参照ください。

r_ble_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_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。

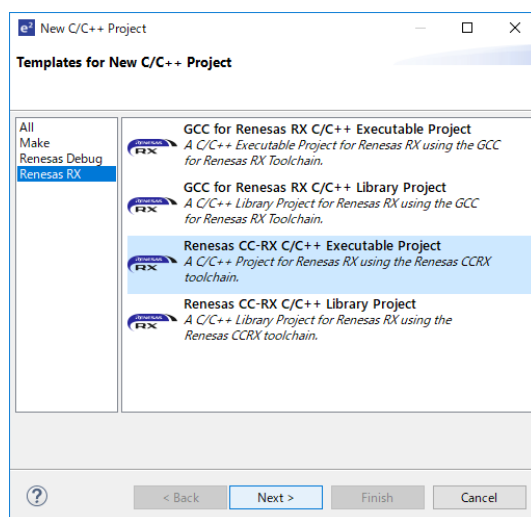


図 4.1：プロジェクトテンプレート選択画面

プロジェクト名を入力し、[Next]ボタンをクリックします。

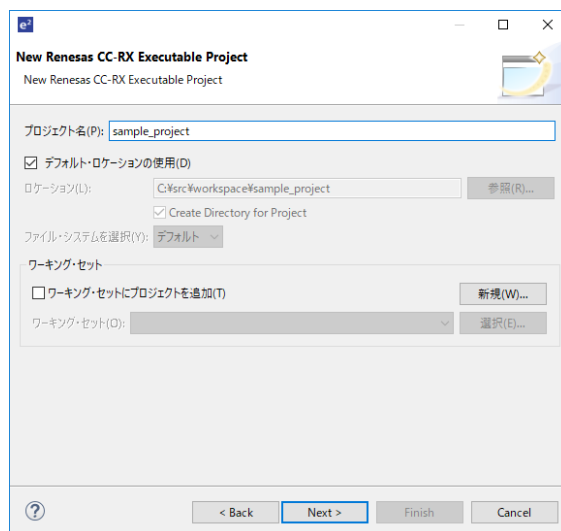


図 4.2：新規プロジェクト設定画面

[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。

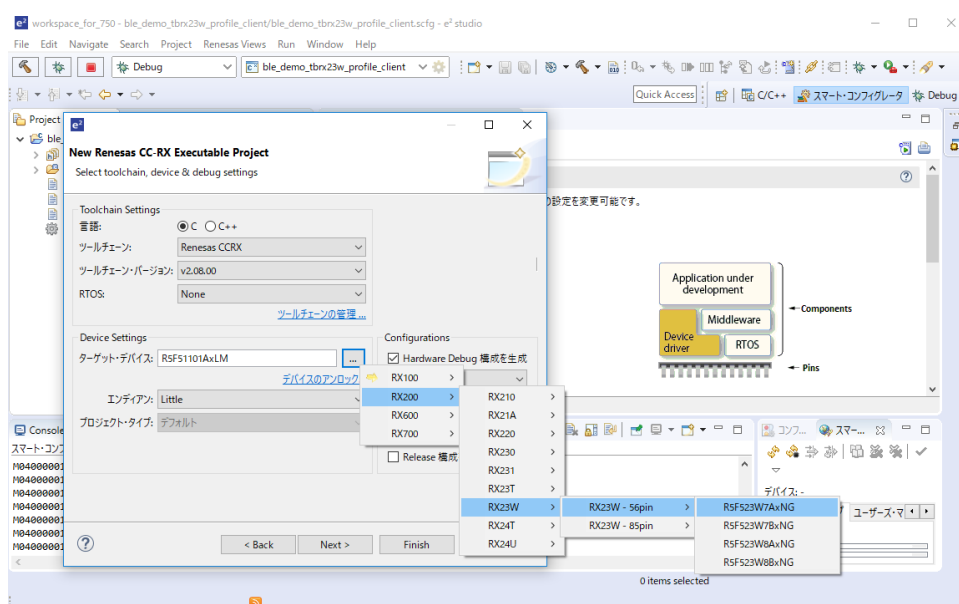


図 4.3：ツールチェーン、デバイス、デバッグ設定画面

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。

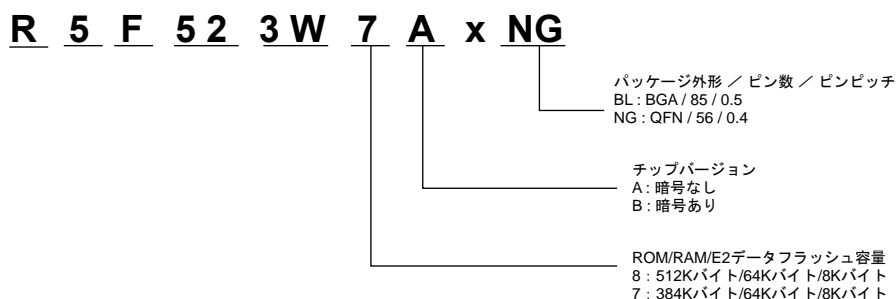


図 4.4：RX23 の型名とメモリサイズ・パッケージ

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。

[Next]ボタンをクリックすると、[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。
[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。

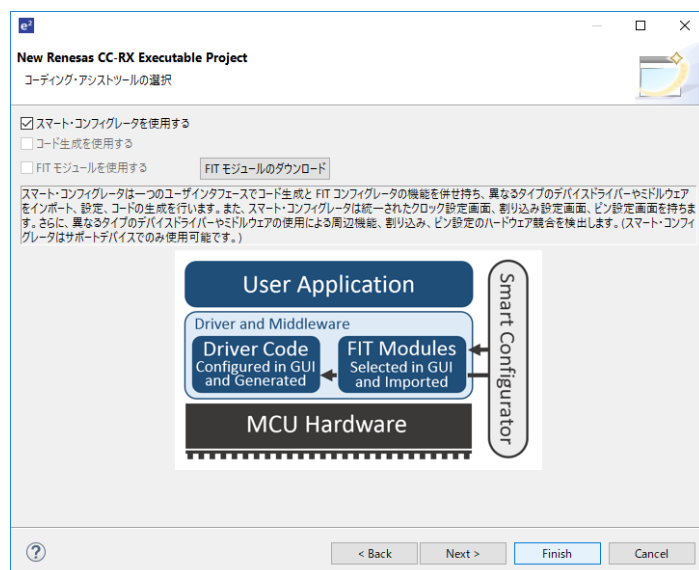


図 4.5：コーディング・アシストツールの選択画面

しばらくすると、プロジェクトが生成されます。

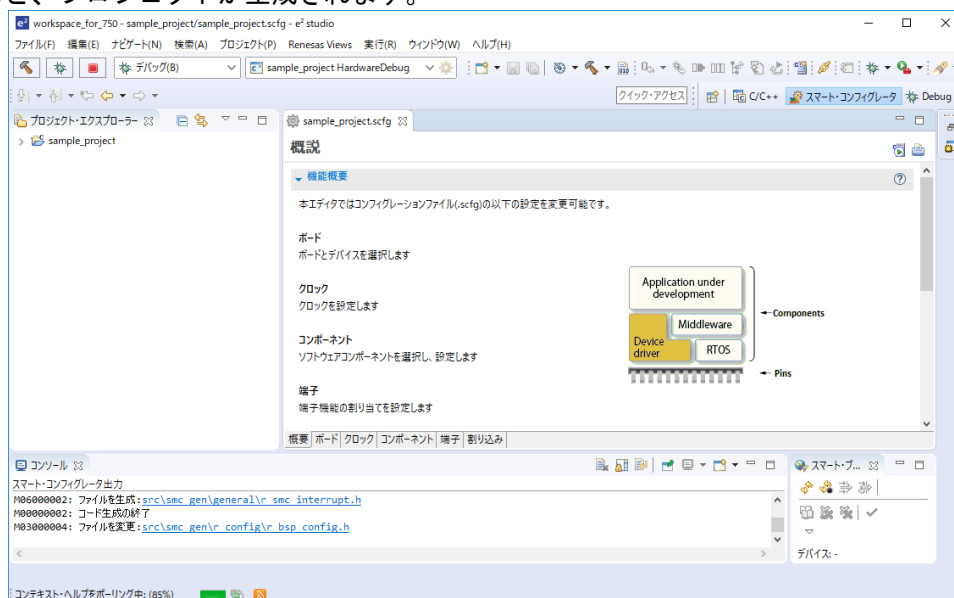


図 4.6：プロジェクト生成後の画面

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 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

BLE 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。

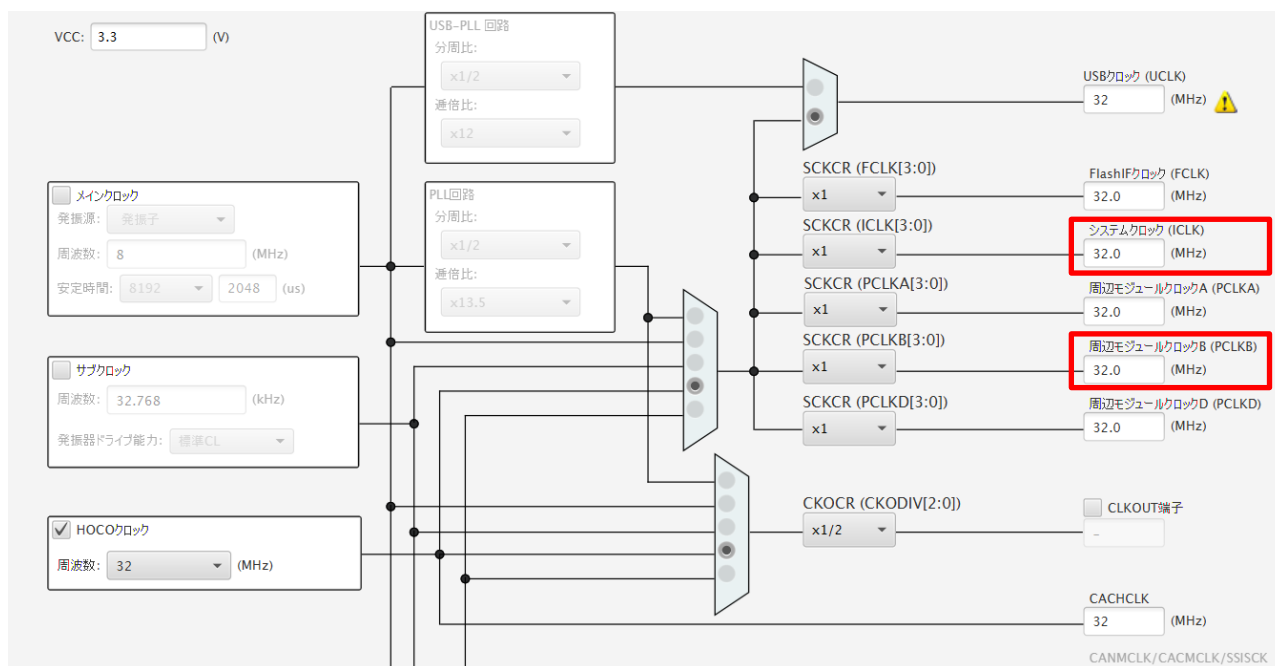


図 4.7: クロック設定画面


4.3 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 BLE 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_sic_rx を選択し、[Finish]ボタンをクリックします。

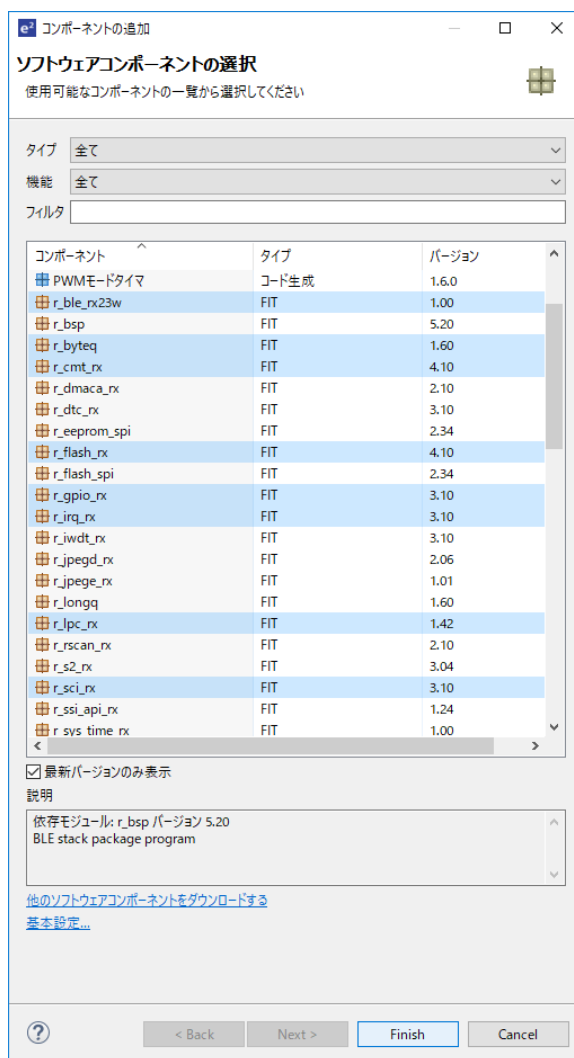


図 4.8：ソフトウェアコンポーネントの選択画面

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。

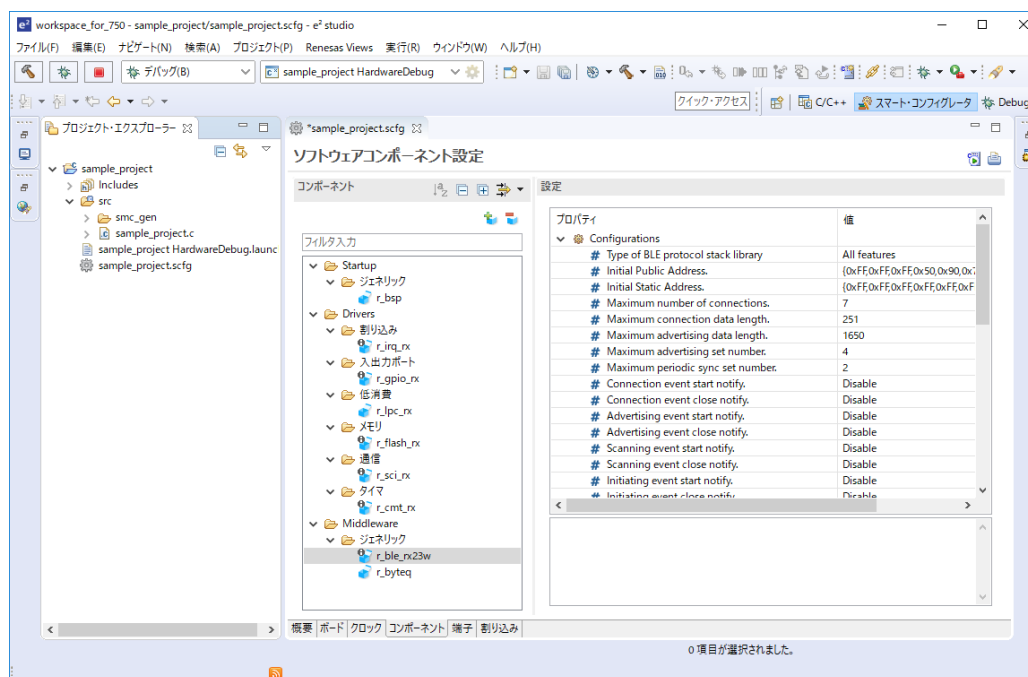


図 4.9: ソフトウェアコンポーネント設定画面

4.4 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 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。

4.4.1 BSP(r_bsp)

“2.9 Flash Memory Protection”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。

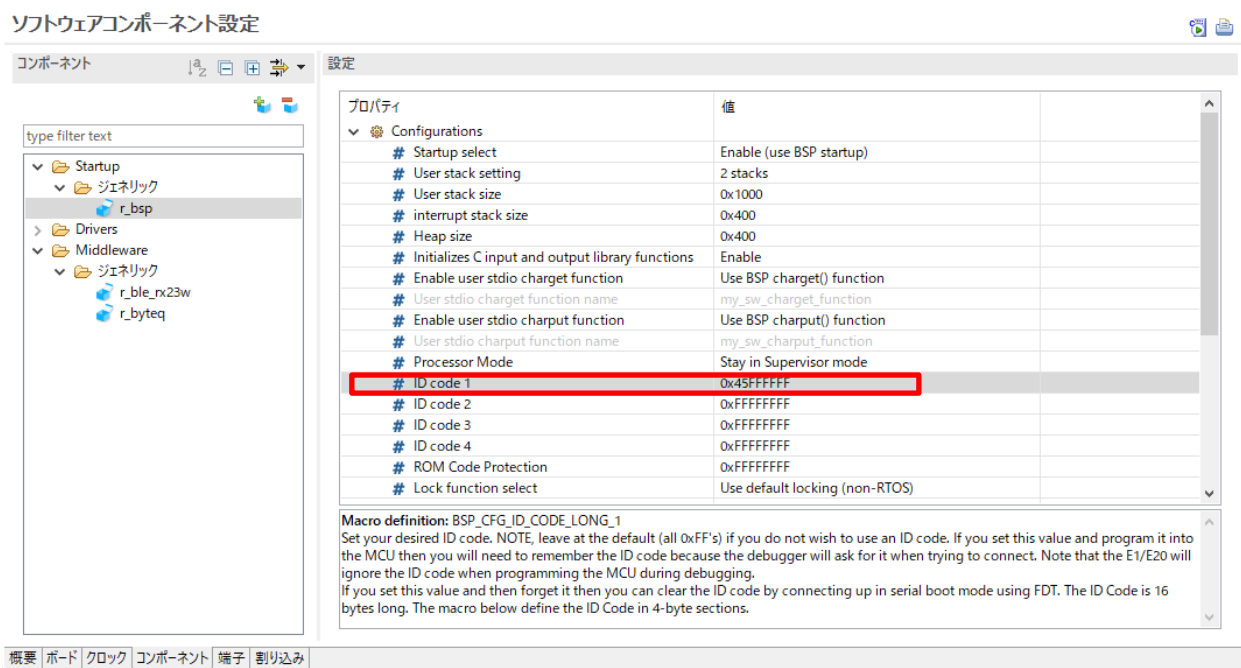


図 4.10 : BSP のコンフィギュレーションオプションの設定画面

4.4.2 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 のチャンネル番号は[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 は SCI8 を使用するため、"8"に設定します。

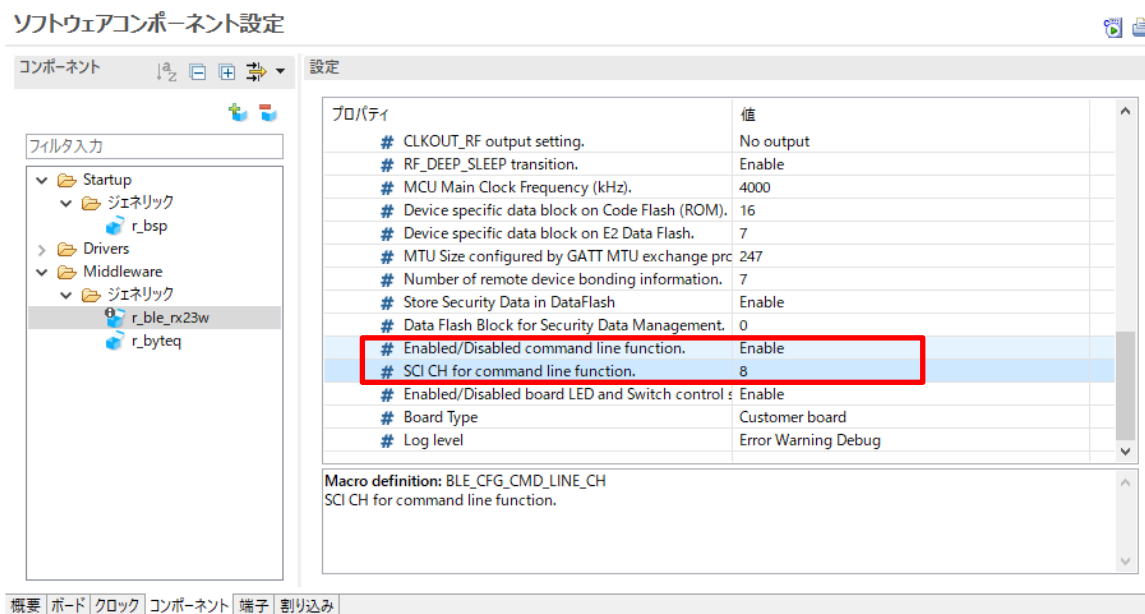


図 4.11: BLE のコンフィギュレーションオプションの設定画面

(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、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。

[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。

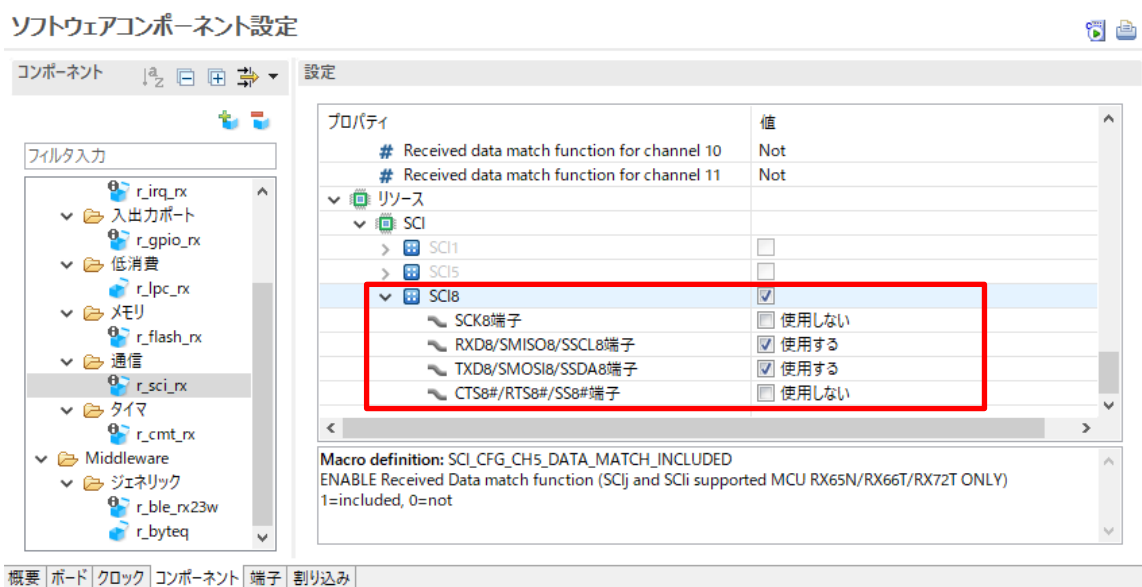
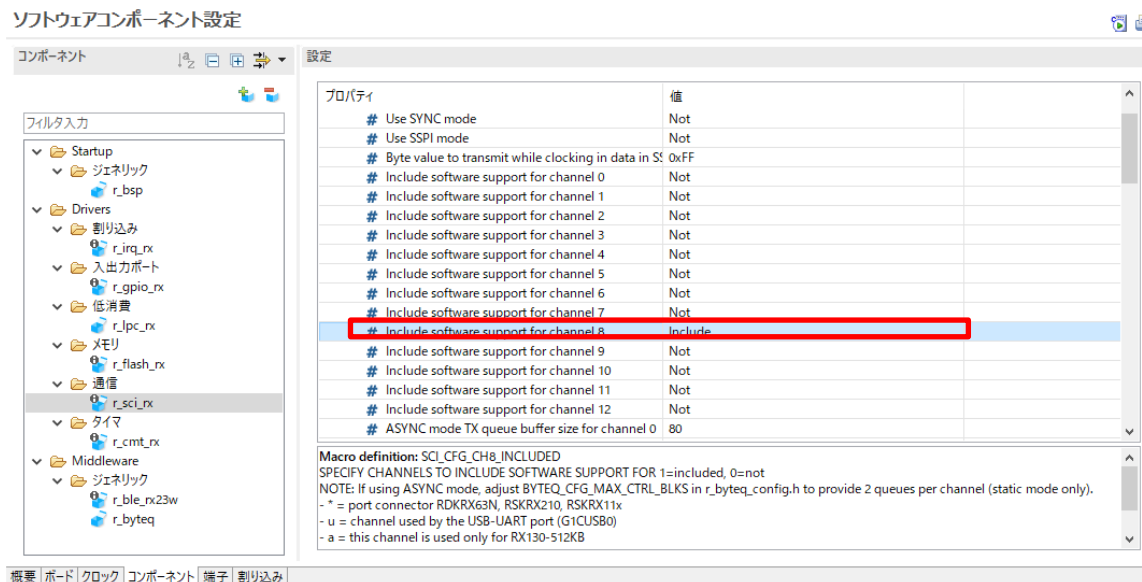


図 4.12 : SCI のコンフィギュレーションオプションの設定画面

4.4.3 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。

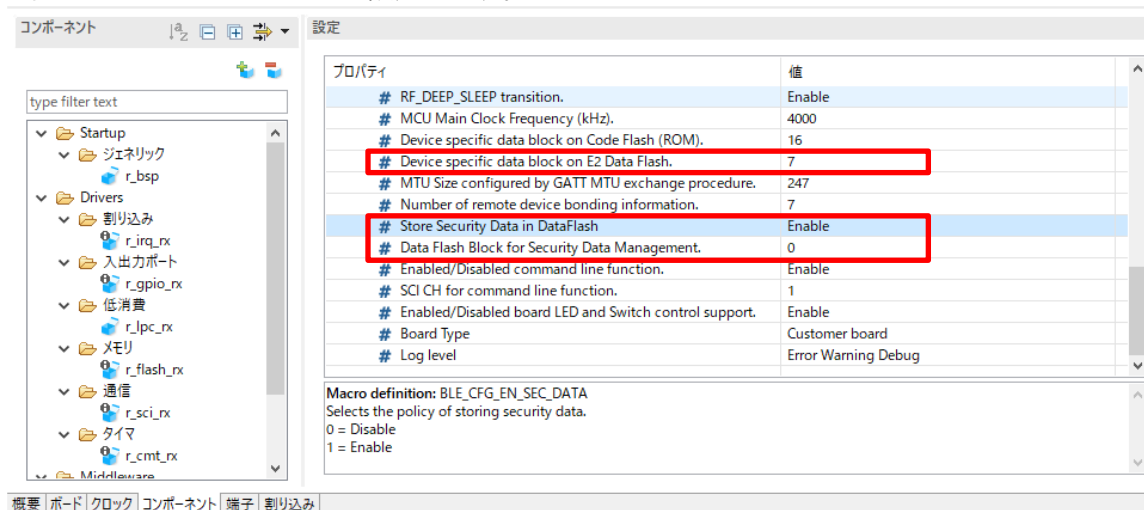


図 4.13: BLE のコンフィギュレーションオプションの設定画面

4.4.4 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 : Target Board と RSSK の LED, Switch の端子

| Board | LED | Switch |
|--------------|--------------------------|--------------------------|
| Target Board | LED1 : PC0 LED2 : PB0 | SW1 : IRQ5 |
| RSSK | LED1 : P42 LED2 : P43 | SW1 : IRQ1 SW2 : IRQ0 |

(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、"4.5.2 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。

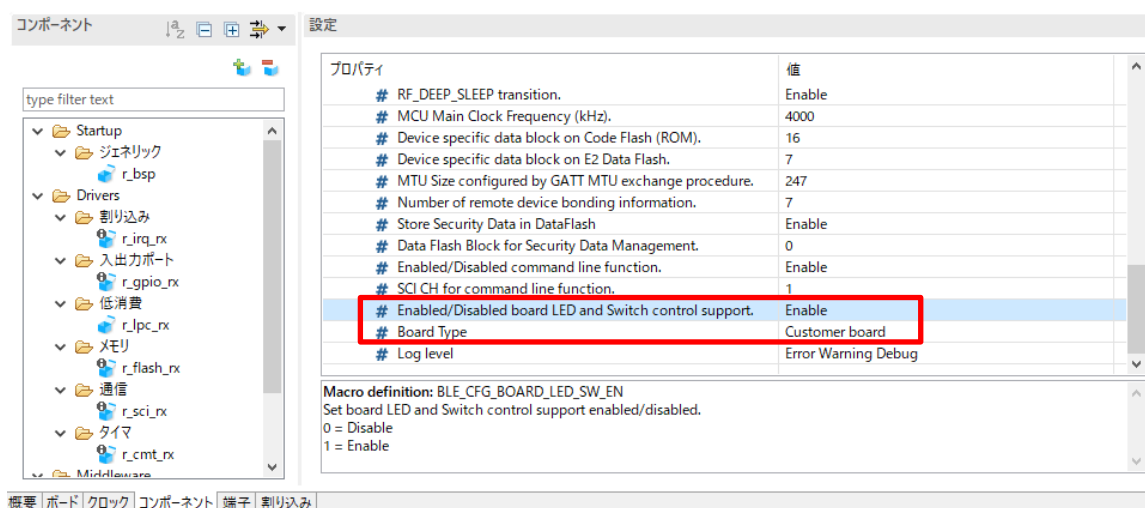


図 4.14: BLE のコンフィギュレーションオプションの設定画面

(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、[リソース]の[ICU]から 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。

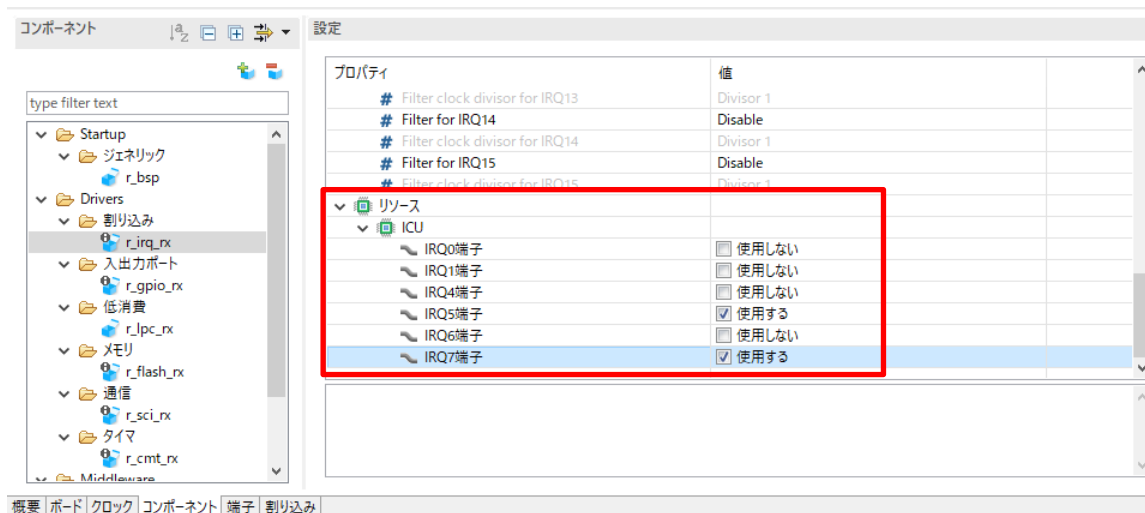


図 4.15: IRQ のコンフィグレーションオプションの設定画面

4.5 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。

4.5.1 CMT の変更

BLE FIT モジュールは CMT のチャンネルを 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 4.16 参照)

```
#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 4.16 : `r_cmt_rx.c` の変更内容

4.5.2 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 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 4.17 : 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 4.17 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)
    /*Set IRQ5 pin */
    PORT1.PMR.BIT.B5 = 0U;
    PORT1.PDR.BIT.B5 = 0U;
    MPC.P15PFS.BYTE = 0x40U;
#elif (BLE_CFG_BOARD_TYPE == 2)
    /*Set IRQ0 pin */
    PORT3.PMR.BIT.B0 = 0U;
    PORT3.PDR.BIT.B0 = 0U;
    MPC.P30PFS.BYTE = 0x40U;

    /*Set IRQ1 pin */
    PORT3.PMR.BIT.B1 = 0U;
    PORT3.PDR.BIT.B1 = 0U;
    MPC.P31PFS.BYTE = 0x40U;
#else /* (BLE_CFG_BOARD_TYPE == x) */
    /*Set IRQ5 pin */
    PORT1.PMR.BIT.B5 = 0U;
    PORT1.PDR.BIT.B5 = 0U;
    MPC.P15PFS.BYTE = 0x40U;
    /*Set IRQ7 pin */
    PORT1.PMR.BIT.B7 = 0U;
    PORT1.PDR.BIT.B7 = 0U;
    MPC.P17PFS.BYTE = 0x40U;
#endif /* (BLE_CFG_BOARD_TYPE == x) */
```

この部分を
Customer board の
設定に合わせて変更。

図 4.18 : irq_pin_set()の変更箇所

4.5.3 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])が出力されますが、実装は空となっています。そのため、このファイルを削除し、下記いずれかの方法でアプリケーション開発のベースを追加することを推奨します。アプリケーションの実装内容の詳細については”5. アプリケーションの作成方法”をご参照ください。

(1)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(R01AN4553)”をご参照ください。

(2)デモプロジェクト

デモプロジェクトで使用している、app_main.c, gatt_db.c, gatt_db.h と services ディレクトリのプロファイルのコードをコピーし、[src]にペーストします。

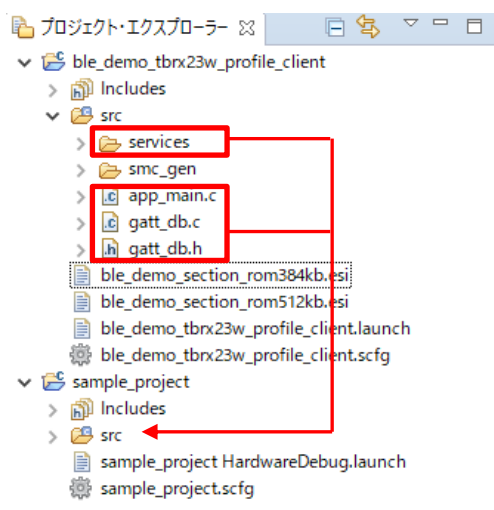


図 4.19：プロジェクト・エクスプローラー画面

追加したコードがビルド対象に含まれるように、[プロジェクト]メニューの[プロパティ]を選択し、[C/C++ build]の[Settings]ダイアログの[Tool Settings]タブを選択し、[Compiler]→[ソース]の[インクルード・ファイルを検索するフォルダ]に追加したコードを配置したディレクトリを登録します。

4.6 リンカ設定

4.6.1 セクション配置

BLE 用のセクション配置を行います。BLE Protocol Stack のセクション名を表 4.2 に示します。

表 4.2 : BLE Protocol Stack のセクション名

| No. | Name | Default Section Name | BLE 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 4.2 のセクション名を以下の手順で BLE 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 4.20 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。

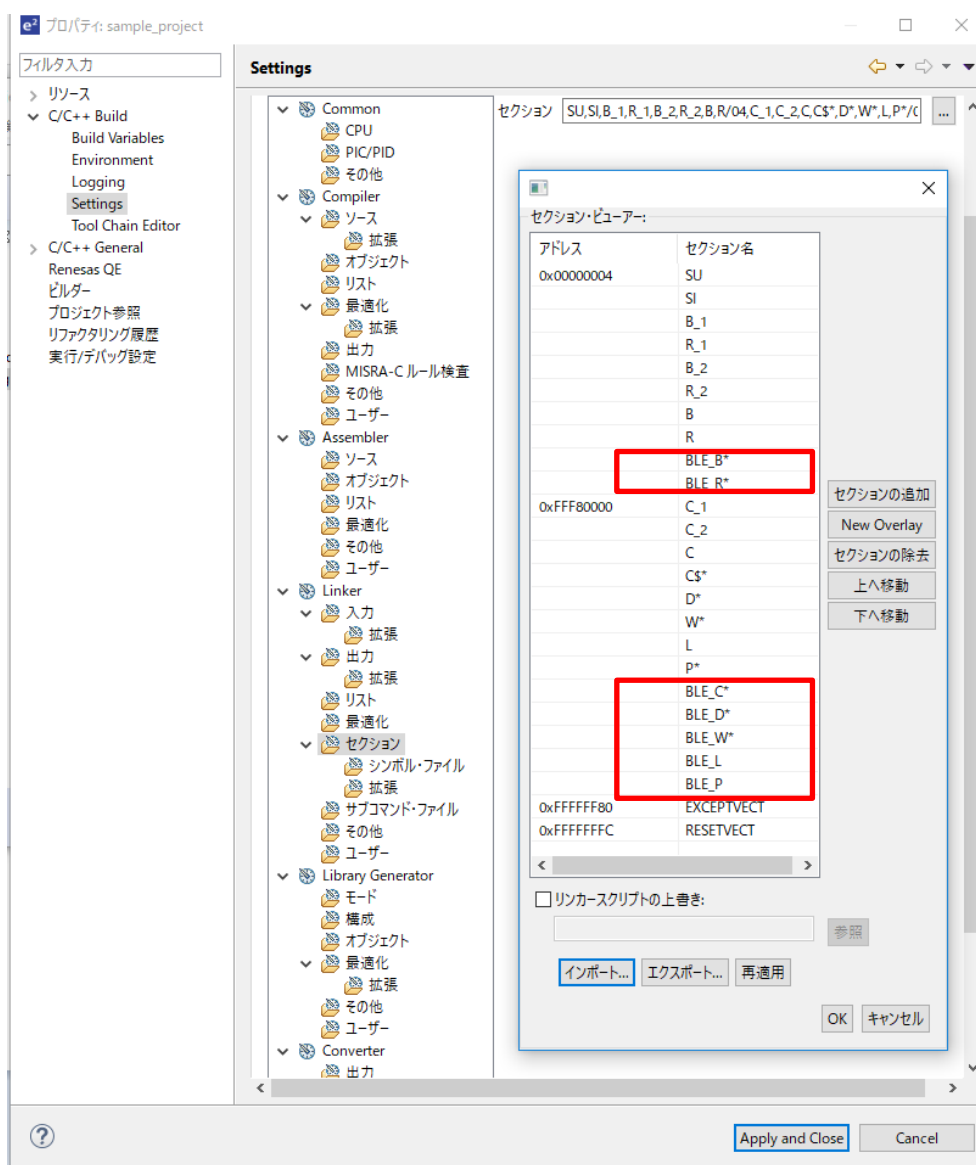


図 4.20 : プロジェクトのセクションの設定画面

(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 4.21 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。

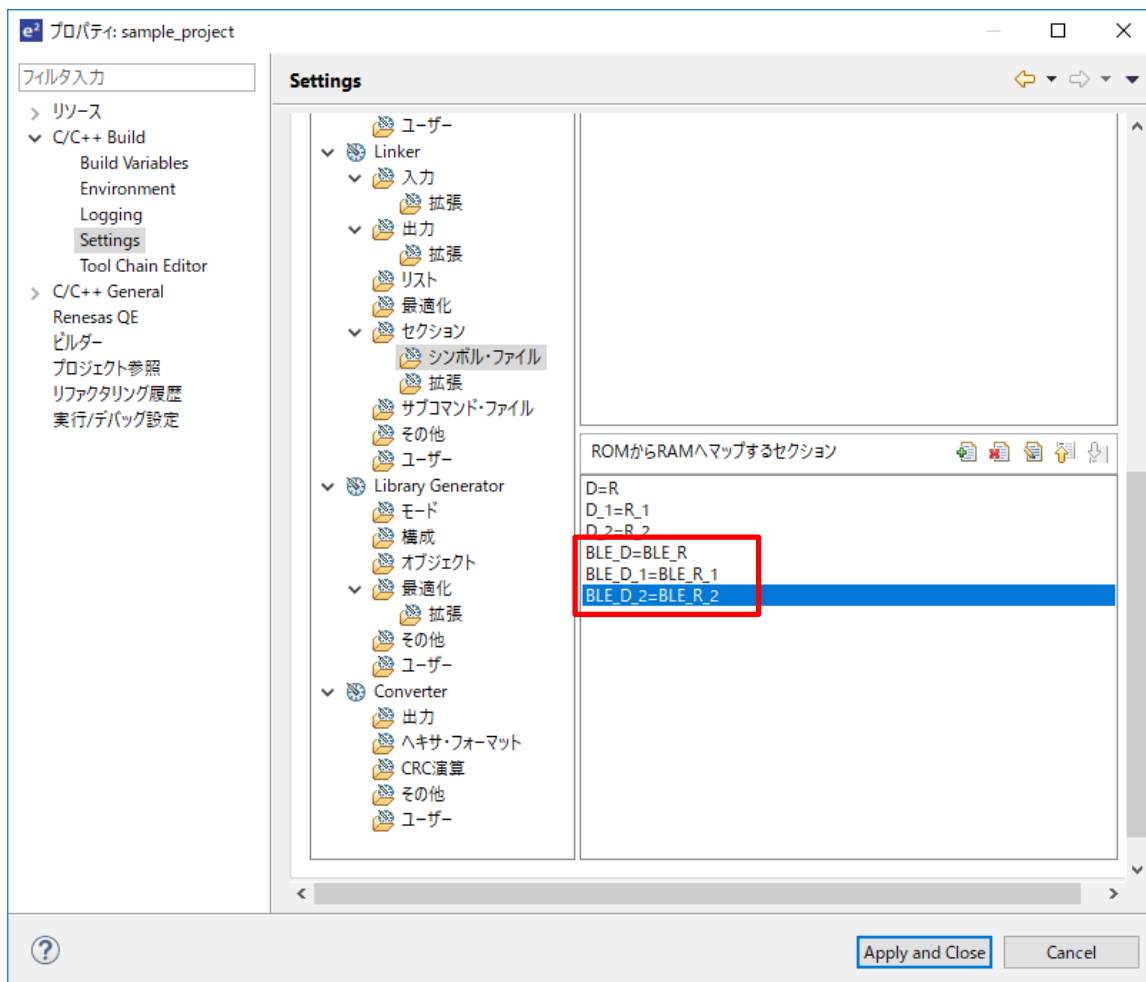


図 4.21 : ROM から RAM へマップするセクションの設定画面

[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、DTBL/BTBL テーブルを変更する必要はありません。

4.6.2 BLE Protocol Stack ライブラリ

BLE Protocol Stack はスタティックライブラリとして提供します。BLE FIT モジュールの lib ディレクトリ内に表 4.3 のライブラリを用意しています。

表 4.3 : BLE Protocol Stack のライブラリ

| BLE 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 BLE 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 4.22 参照)

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

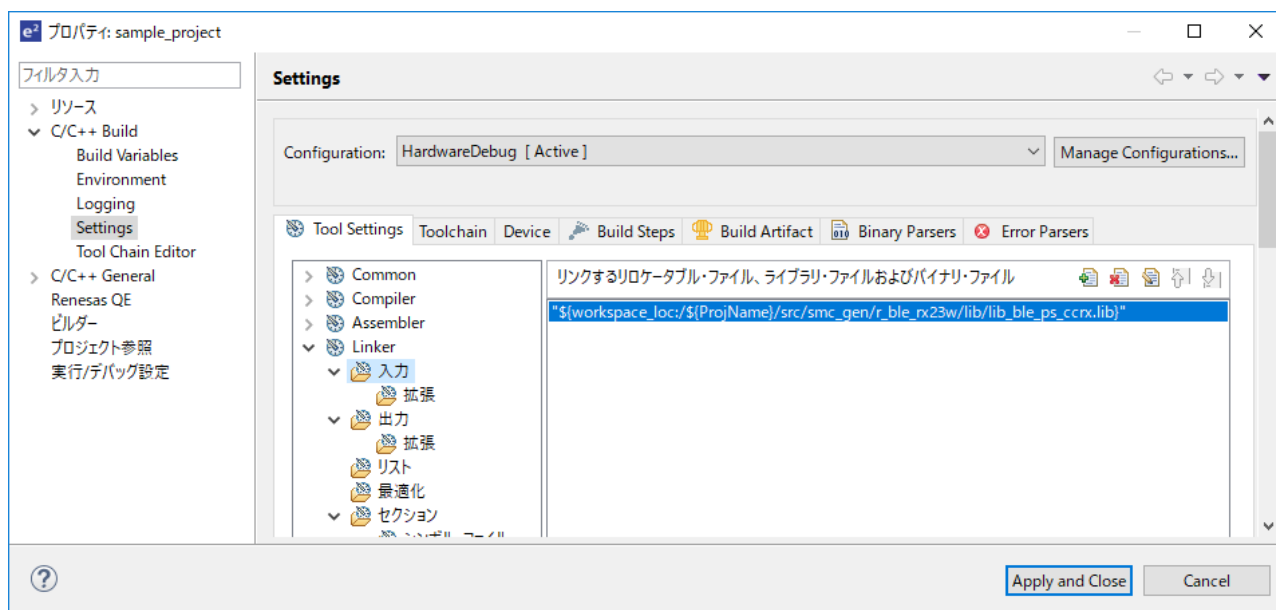


図 4.22 : Linker の入力設定画面

2) ライブラリ選択バッチファイルの登録

BLE のコンフィグ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++ Build]の[Settings]の[Build Steps]タブ内の[Pre-build steps]の[Command(s):]に以下を入力します。(図 4.23 参照)

```
..%src%smc_gen%r_ble_rx23w%lib%ble_fit_lib_selector.bat
```

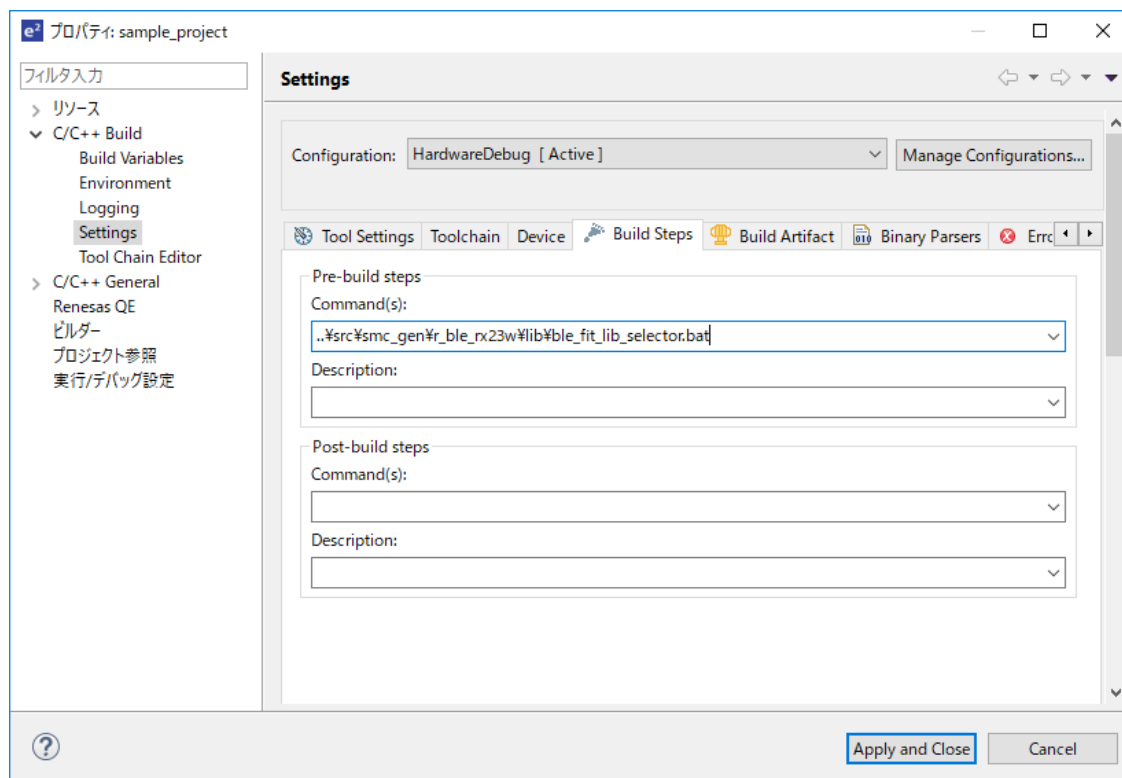


図 4.23 : Build Steps の入力設定画面

4.7 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。

4.7.1 Flash の ID Code

“2.9 Flash Memory Protection”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“45FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF”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。

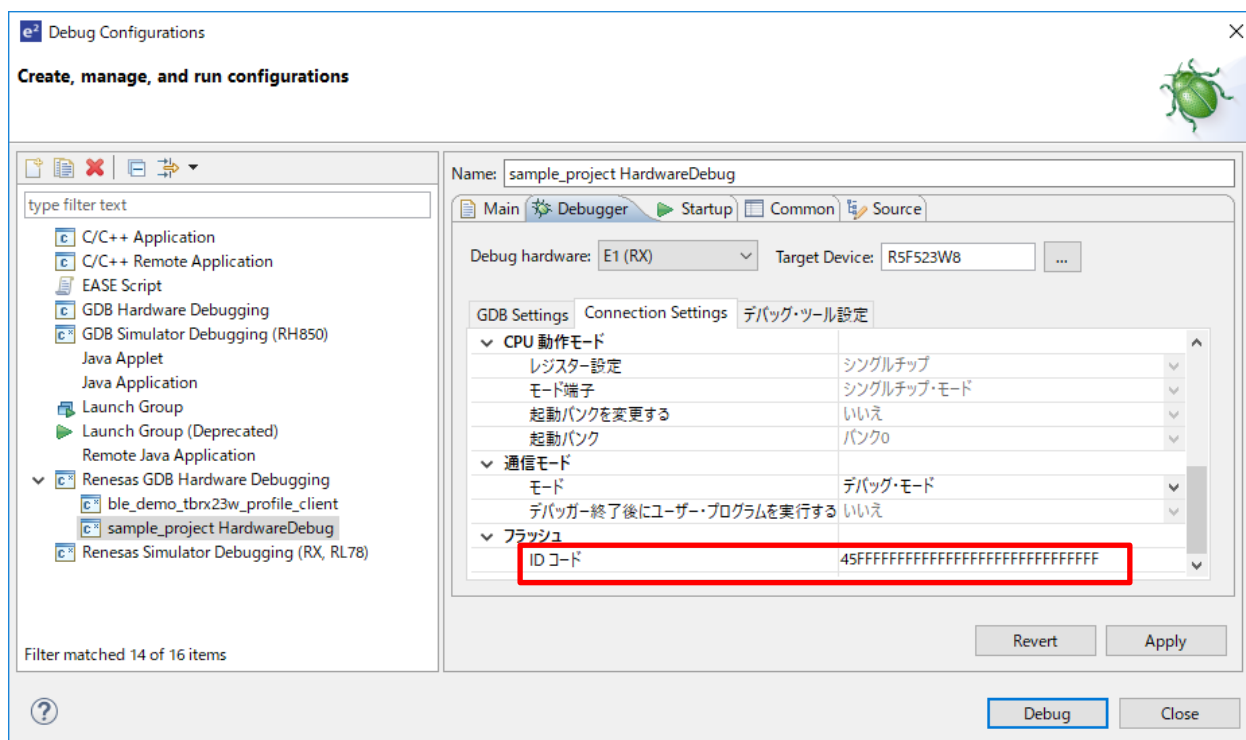


図 4.24 : ID コードプロテクションの設定画面

4.7.2 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。

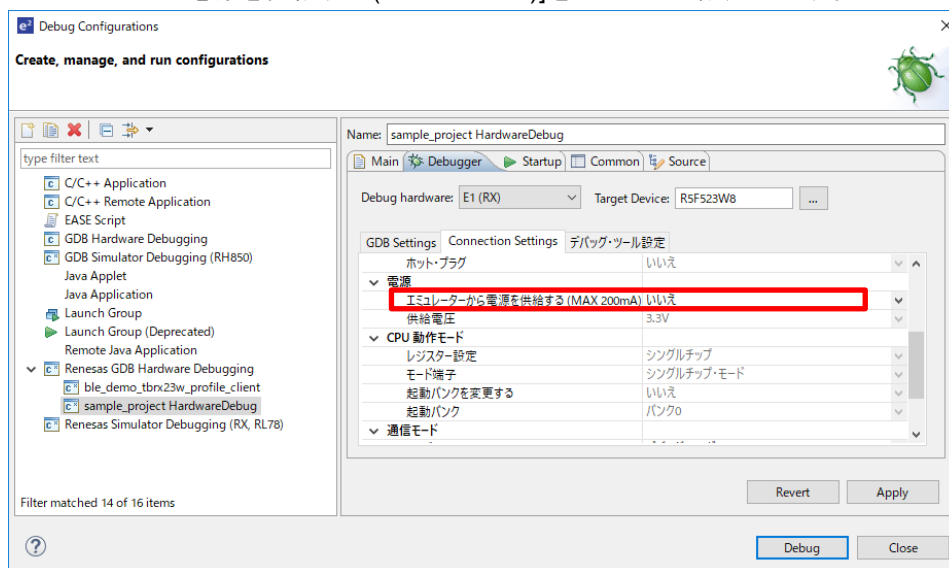
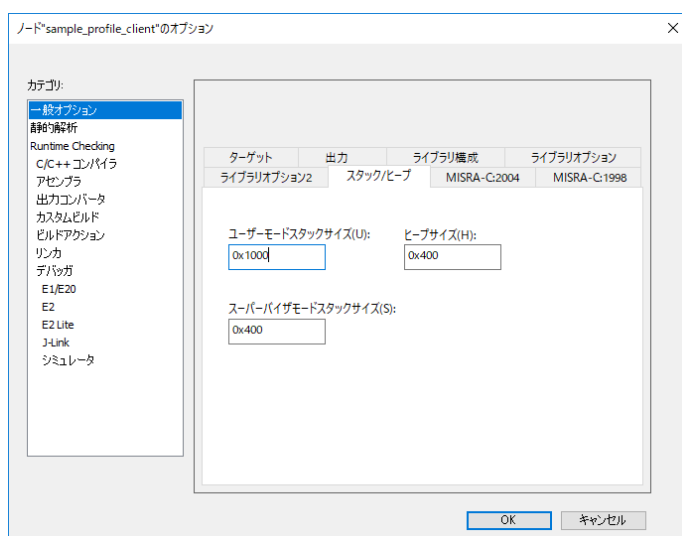


図 4.25：エミュレーターからの電源供給の設定画面

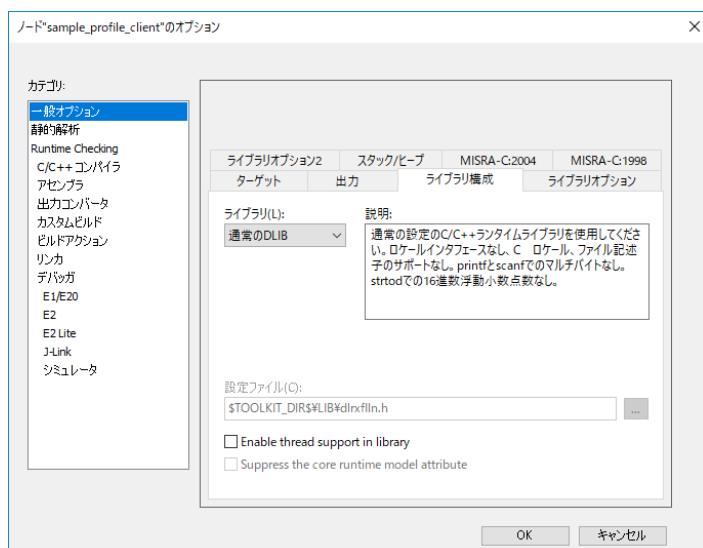
4.8 IAR 開発環境でのプロジェクト作成

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。

- 4.1 から 4.6 までの手順にて e² studio 上でベースとなるプロジェクトを作成します。
 - "RX ファミリー ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)" (以下、R01AN1685)の「7.2 IAR プロジェクトへの FIT モジュールの追加方法」に従って、1.で作成した e² studio のプロジェクトを IAR のプロジェクトに変換します。
 - IAR Embedded Workbench for Renesas RX を起動し、変換した IAR プロジェクト内の eww ファイルをオープンし、"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。
- **スタック／ヒープ**
R01AN1685 の「2.8 スタック領域とヒープ領域」、「表 3.2 スタックサイズとヒープサイズの定義」を参考に、"一般オプション"カテゴリの"スタック／ヒープ"のサイズを e² studio プロジェクトと同じサイズに設定します。

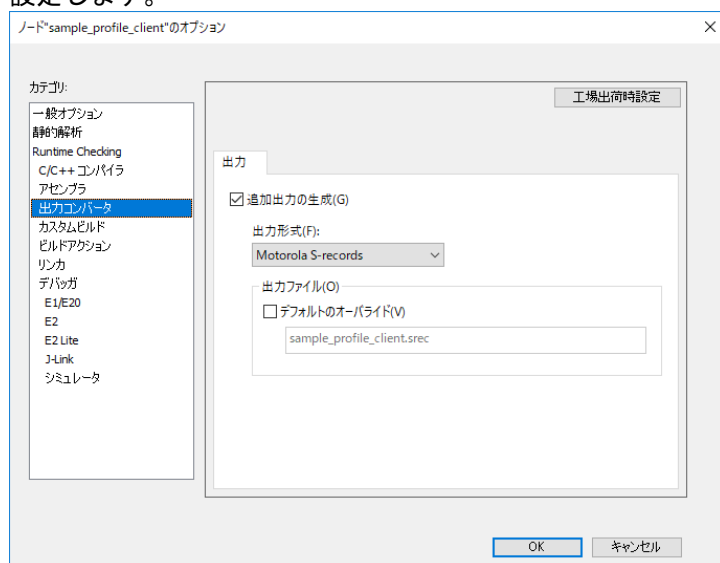


- **ライブラリ構成**
"一般オプション"カテゴリの"ライブラリ構成"のライブラリを"通常の DLIB"に設定します。



- 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。



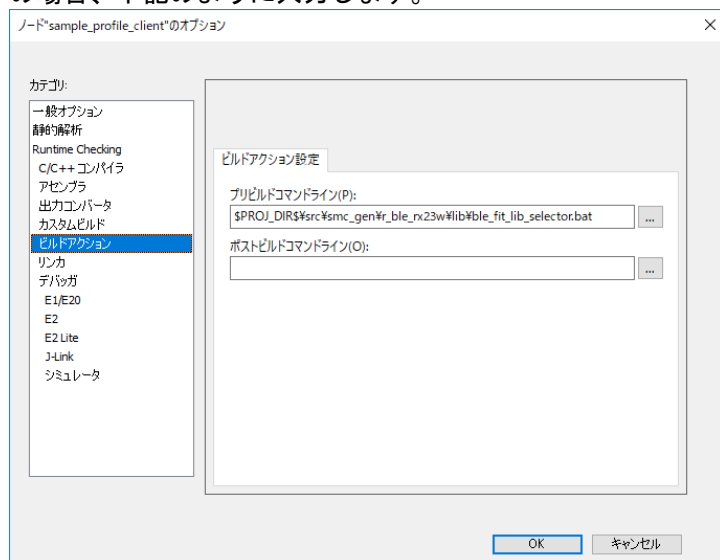
- プリビルドコマンドライン

”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。



- リンカ設定ファイル

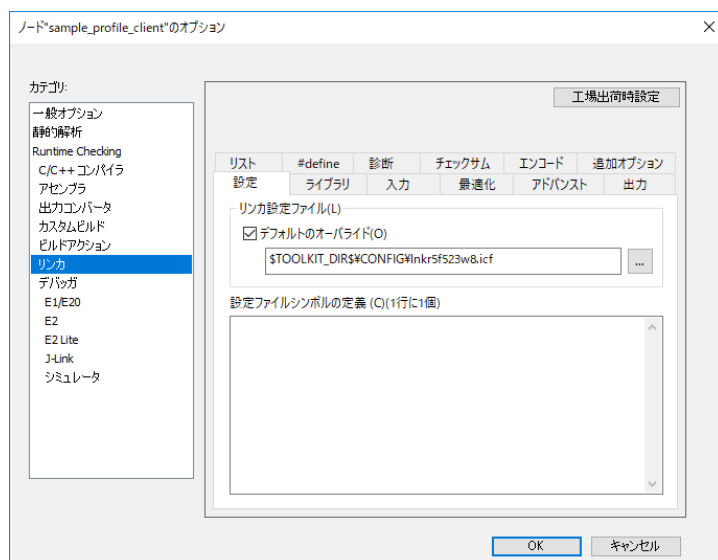
“リンカ”カテゴリの“デフォルトのオーバーライド”にチェックを入れ、Inkr5f523w8.icf、または、Inkr5f523w7.icf ファイルを指定します。

設定例.

リンカ設定ファイルのパスが

\$TOOLKIT_DIR\$¥CONFIG¥Inkr5f523w8.icf

の場合、下記のように入力します。



- 追加ライブラリ

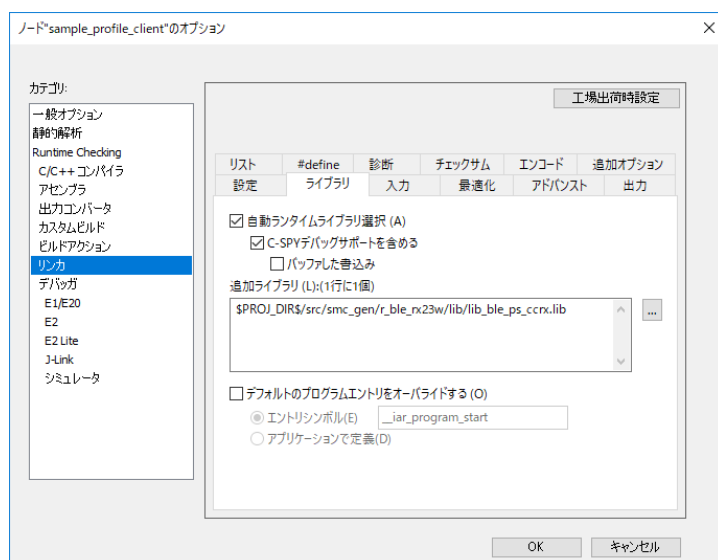
“リンカ”カテゴリの“追加ライブラリ”に lib_ble_ps_ccrx.lib のパスを指定します。

設定例.

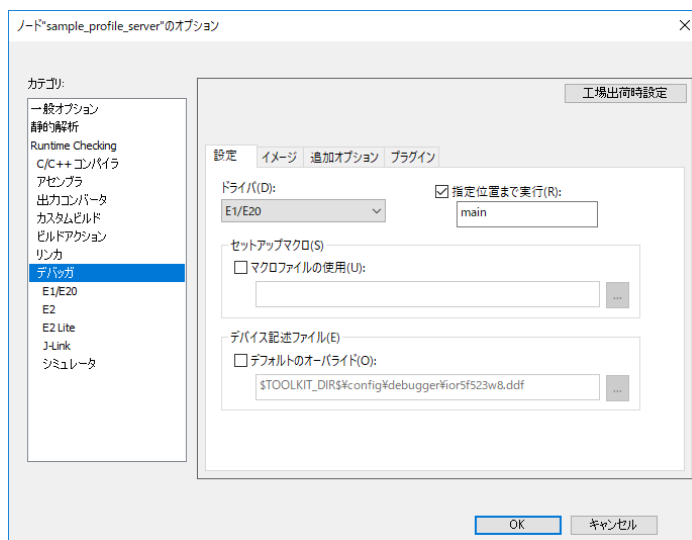
BLE プロトコルスタックのパスが

\$PROJ_DIR\$/src/smc_gen/r_ble_rx23w/lib/lib_ble_ps_ccrx.lib

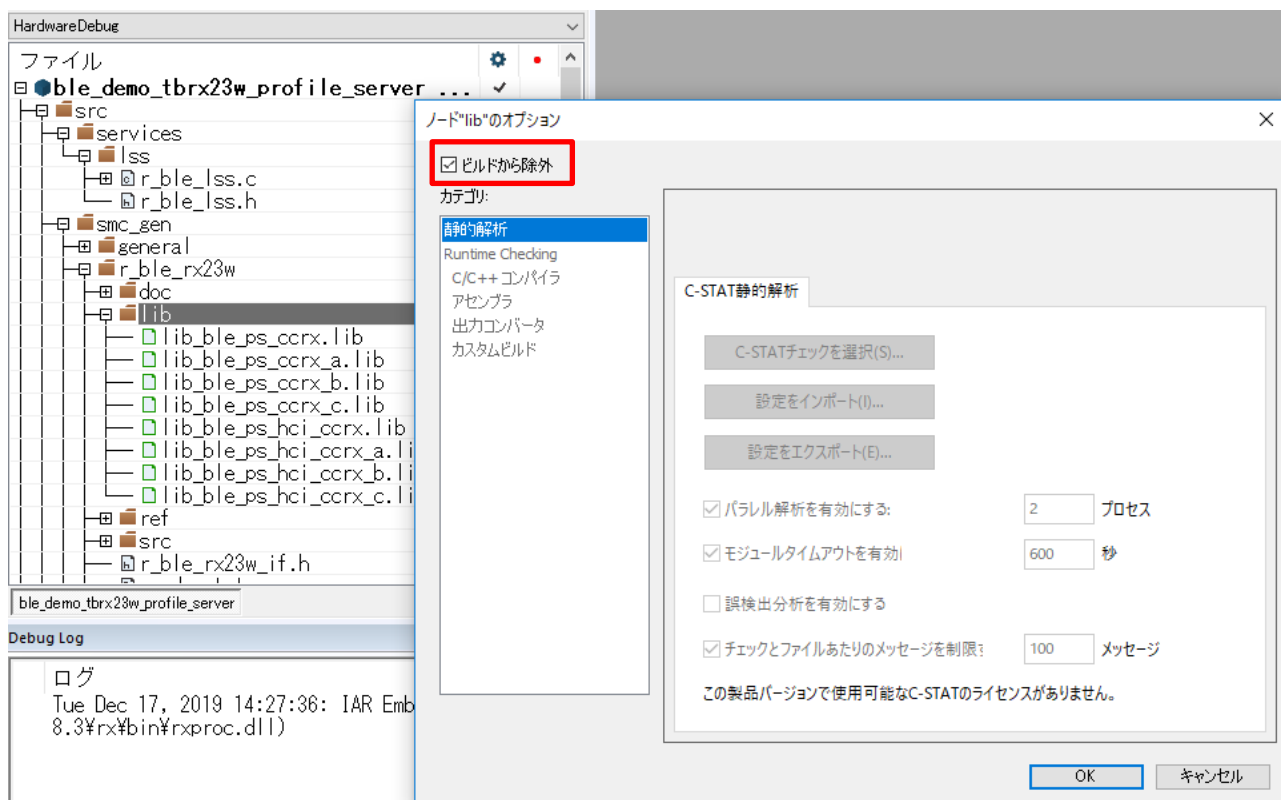
の場合、下記のように入力します。



- 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を指定します。
設定例。
エミュレータに E1 を使用する場合、下記のように“E1/E20”を選択します。



4. BLE FIT モジュールの lib ディレクトリをビルド対象から除外します。
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で“ビルドから除外”のチェックボックスを選択します。



5. 上記を変更後、ビルドを実行します。ビルド完了後、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

5.1 main 関数

BLE アプリケーションの処理を実装するファイルに main 関数を用意します。

main 関数内では、初期化処理、main loop の実装を行います。BLE を使用する場合、以下を必ず実装してください。

- R_BLE_Open()による、BLE 機能の初期化
R_BLE_Open()は必ず、main 関数の最初でコールしてください。
- ble_app_init()による BLE ホストスタック、プロファイルの初期化
- main loop

また、タイマ、ボード設定、コマンドラインなどを使用する場合、main 関数で初期化処理を行ってください。

例として、

- BLE の初期化
- ボードの初期化
- 低消費電力制御機能の初期化
- BLE アプリケーション用の timer の初期化
- BLE ホストスタック、プロファイルの初期化
- コマンドラインの初期化
- main loop
- 低消費電力状態への移行

から構成される main 関数を図 5.1 に示します。

```
/* *****  
 * Function Name: main  
 * Description  : The main loop  
 * Arguments   : none  
 * Return Value : none  
 ***** */  
void main(void)  
{  
    /* Initialize BLE */  
    R_BLE_Open();  
  
    /* Configure the board */  
    R_BLE_BOARD_Init();  
    R_BLE_BOARD_RegisterSwitchCb(BLE_BOARD_SW2, sw_cb);  
  
    /* Initialize the Low Power Control function */  
    R_BLE_LPC_Init();  
  
    /* Initialize timer for LED blink */  
    R_BLE_TIMER_Init();  
  
    /* Configure CommandLine */  
    R_BLE_CLI_Init();  
    R_BLE_CLI_RegisterCmds(gsp_cmds, ARRAY_SIZE(gsp_cmds));  
  
    /* Initialize BLE host stack and profiles */  
    ble_app_init();  
  
    /* main loop */  
    while (1)  
    {  
        /* Process Command Line */  
        R_BLE_CLI_Process();  
        /* Process Event */  
        R_BLE_Execute();  
        /* Enter the Lower Power Mode */  
        R_BLE_LPC_EnterLowPowerMode();  
    }  
  
    /* Terminate timer for LED blink */  
    R_BLE_TIMER_Terminate();  
  
    /* Terminate CommandLine */  
    R_BLE_CLI_Terminate();  
  
    /* Terminate BLE */  
    R_BLE_Close();  
}
```

図 5.1 : main 関数のサンプル

5.1.1 main loop

BLE ソフトウェアではイベント処理のためにスケジューラを使用します。 スケジューラを動作させるために、 main loop 内にて R_BLE_Execute() をコールしてください。(図 5.2 参照)

```
while (1)  
{  
    R_BLE_Execute();  
}
```

図 5.2 : R_BLE_Execute() の実行例

発生したイベントは登録したコールバック関数で処理されます。コールバック関数については、5.2 コールバックと登録関数をご参照ください。

5.1.2 ホストスタック、プロファイルの初期化

ホストスタック、プロファイルの初期化は アプリケーションの ble_app_init() で実装します。

ホストスタックの初期化は

- R_BLE_GAP_Init()
- R_BLE_ABS_Init()

のいずれかで行います。パラメータの詳細については R_BLE API ドキュメント(r_ble_spec.chm)をご参照ください。

プロファイルの初期化は、作成するプロファイルのロールによって、表 5.1 の初期化が必要となります。QE for BLE からプロファイルを生成する場合、表 5.1 の初期化を実装したソースコードが出力されます。

表 5.1 : GATT の初期化

| アプリケーションのタイプ | ble_app_init()内でコールする関数 | 備考 |
|--------------|---|--------------------------|
| GATT Server | R_BLE_GATTS_SetDBInst() | GATT Database の初期化 |
| | R_BLE_SERVS_Init()と R_BLE_XXX_Init() (XXX は GATT Server 名) | GATT Server の初期化 |
| GATT Client | R_BLE_SERVC_Init()と R_BLE_XXX_Init() (XXX は GATT Client 名) | GATT Client 機能の初期化 |
| | R_BLE_DISC_Init() | Service Discovery 機能の初期化 |

ble_app_init()のサンプルを図 5.3 に示します。

```
/* *****  
 * Function Name: ble_app_init  
 * Description  : Initialize host stack and profiles.  
 * Arguments   : none  
 * Return Value : BLR_SUCCESS - SUCCESS  
 *              BLE_ERR_INVALID_OPERATION -  
 *              Failed to initialize host stack or profiles.  
 * *****/  
static ble_status_t ble_app_init(void)  
{  
    ble_status_t status;  
  
    gs_conn_hdl = BLE_GAP_INVALID_CONN_HDL;  
    gs_timer_hdl = BLE_TIMER_INVALID_HDL;  
  
    /* Initialize host stack */  
    status = R_BLE_ABS_Init(&gs_abs_init_param);  
    if (BLE_SUCCESS != status)  
    {  
        return BLE_ERR_INVALID_OPERATION;  
    }  
  
    /* Initialize GATT Database */  
    status = R_BLE_GATTS_SetDbInst(&g_gatt_db_table);  
    if (BLE_SUCCESS != status)  
    {  
        return BLE_ERR_INVALID_OPERATION;  
    }  
  
    /* Initialize GATT Server */  
    status = R_BLE_SERVS_Init();  
    if (BLE_SUCCESS != status)  
    {  
        return BLE_ERR_INVALID_OPERATION;  
    }  
  
    /* Initialize LED and Switch Service */  
    status = R_BLE_LSS_Init(lss_cb);  
    if (BLE_SUCCESS != status)  
    {  
        return BLE_ERR_INVALID_OPERATION;  
    }  
  
    /* Create timer for LED blink */  
    status = R_BLE_TIMER_Create(&gs_timer_hdl, 1, BLE_TIMER_PERIODIC, timer_cb);  
    if (BLE_SUCCESS != status)  
    {  
        return BLE_ERR_INVALID_OPERATION;  
    }  
  
    return status;  
}
```

図 5.3 : ble_app_init()のサンプル

5.2 コールバックと登録関数

アプリケーションにコールバック関数を登録することで、各種イベントの受信タイミングで処理を行うことが可能です。各機能ブロックのコールバック登録関数を表 5.2 に示します。

表 5.2：コールバック登録関数

| 機能ブロック | 登録関数 | 用途 |
|--------------------------|---|--|
| GAP | R_BLE_GAP_Init() or R_BLE_ABS_Init() | Advertising, Scan, Connection 確立などのイベント受信時 |
| GATT | Server : R_BLE_GATTS_RegisterCb() | Server : Client からアクセス時の処理 |
| | Client : R_BLE_GATTC_RegisterCb() or R_BLE_ABS_Init() | Client : リクエストの応答が返った時の処理 |
| Vendor Specific | R_BLE_VS_Init() or R_BLE_ABS_Init() | Vendor 固有のイベント受信時の処理 |
| L2CAP | R_BLE_L2CAP_RegisterCbPsm() | L2CAP Credit-Based Flow Control のリクエストの応答が返った時の処理 L2CAP Credit-Based Flow Control のイベント受信時の処理 |
| board | R_BLE_BOARD_RegisterSwitchCb() | Board の Switch が押下された場合の処理 |
| Timer | R_BLE_TIMER_Create() | 指定した時間が経過した場合の処理 |
| Profile(Service, Client) | R_BLE_XXX_Init() (XXX は Service, Client の名前) | Service : Client からアクセスされた場合の処理 Client : リクエストの応答が返った時の処理 |
| Service Discovery | R_BLE_DISC_Start() | サービスディスカバリーが完了したときの処理 |

5.3 イベント通知機能

イベント通知機能は BLE ソフトウェアのスケジューラにイベントを設定します。BLE Protocol Stack は次の R_BLE_Execute()時にイベントの状態を確認し、イベントが設定されている場合は、イベント設定時に登録したコールバック関数を呼び出します。コールバック関数は R_BLE_SetEvent()により登録します。

本機能は主に以下の場合に使用します。

- 割り込みコンテキストで時間のかかる処理を割り込みコンテキスト外で実行する。
- 割り込みコンテキスト内で実行できない関数を割り込みコンテキスト外で呼び出す。

図 5.4 に LED Switch サービスからの Notification 受信時に R_BLE_SetEvent()によるイベント通知を行い、コールバック関数内で、board の LED を点灯・消灯を行うサンプルを示します。

```
static void sw_ntf_rcv_event(void)
{
    R_BLE_BOARD_ToggleLEDState(BLE_BOARD_LED1);
}

static void lsc_cb(uint16_t type, ble_status_t result, st_ble_servc_evt_data_t
*p_data)
{
    ...

    switch (type)
    {
        case BLE_LSC_EVENT_SWITCH_STATE_HDL_VAL_NTF:
        {
            pf("lsc: Receive Switch State Ntf.¥n");
            R_BLE_SetEvent(sw_ntf_rcv_event);
        } break;

        default:
            break;
    }
}
```

図 5.4：イベント通知の例

5.4 GATT Database

BLE アプリケーションでは、GATT サービスアプリケーションを作成する場合、

- gatt_db.c
- gatt_db.h

に GATT Server が提供するデータベースを実装します。

上記のファイルは指定された GATT Service に応じて QE for BLE が出力します。

生成された GATT Database は R_BLE_GATTS_SetDbInst()により登録されます。

5.5 低消費電力状態

MCU の低消費電力状態への移行は BLE 機能を使用時においても可能です。

低消費電力状態への移行の基本方針は以下の通りです。

- R_BLE_Execute()の実行完了後、次の R_BLE_Execute()を実行するまでの期間に、BLE Protocol Stack は MCU の低消費電力状態への移行を阻害しません。
- BLE 機能を含め、使用するすべてのコンポーネントが MCU を低消費電力状態に移行しても問題ないことを確認した上で、アプリケーションが MCU を低消費電力状態に移行します。

低消費電力のサンプルとして、以下の機能を持ったプログラム(r_ble_pf_lowpower.c)を提供します。

- MCU の低消費電力状態への移行は、LPC FIT モジュールを使用します。
- 低消費電力状態として、スリープモード、ディープスリープモード、ソフトウェアスタンバイモードを用意しています。
- 低消費電力機能の初期化は R_BLE_LPC_Init()で行います。
- 低消費電力状態への移行は R_BLE_LPC_EnterLowPowerMode()で行います。
この関数は R_BLE_Execute()の実行完了後にコールし、以下を実行します。
 - 割り込みの無効化
 - 各コンポーネントが低消費電力状態に移行しても問題ないことを確認
 - 各コンポーネントの低消費電力状態への移行処理を実施
 - MCU を低消費電力状態に移行
 - MCU が低消費電力状態から復帰後、各コンポーネントの低消費電力状態への復帰処理を実施
- BLE 通信が発生した場合、RF からの割り込みによりソフトウェアスタンバイモードから復帰

また、コマンドライン機能では、MCU の低消費電力状態を制御するコマンドとして、sys stby コマンドを用意しています。

5.6 Data Flash のブロック

アプリケーションにてユーザが独自のデータを Data Flash 内に保持する場合、以下のコンフィギュレーションオプションで指定したブロック以外をご使用ください。

- BLE_CFG_DEV_DATA_DF_BLOCK
- BLE_CFG_SECD_DATA_DF_BLOCK (セキュリティデータ管理機能が有効な場合のみ)

6. ツール

HCI モードのファームウェア向けに以下のツールを提供します。

6.1 BDAAddrWriter

BDAAddrWriter は 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、BDAAddrWriter により、BD アドレスを設定することが可能です。

6.2 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」をご参照ください。

7. デモプロジェクト

BLE の機能を使用する表 7.1 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 7.1 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 用の GATT Client のデモです。 |
| ble_demo_tbrx23w_profile_server | Target Board 用の GATT Server のデモです。 |
| ble_demo_tbrx23w_uart_hci | Target Board 用の HCI モードのデモです。 |

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 上のシリアルターミナルからコマンドの入力やログ出力の確認が行えます。

7.1 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、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 を行くと、下記のように"RBLE-DEV"、または、"RBLE"という名前で検出されます。



図 7.1：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 キャラクターリスティック(UUID : 58837F57-5F05-4267-AB01-B4968E8EFCE0)
 - LED Blink Rate キャラクターリスティック(UUID : 5883C32F-5F05-4267-AB01-B4968E8EFCE0)

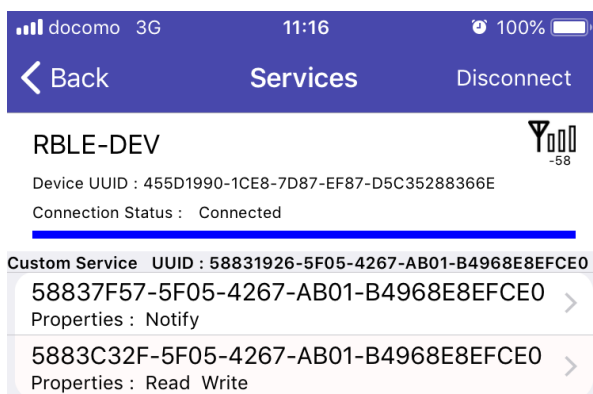


図 7.2：リモートデバイスのサービス検出画面

- 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 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ー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[] =
{
    ...
    /* LED Switch */
    {
        /* Num of Services */
        {
            1,
        },
        /* Description */
        BLE_GATT_DB_SER_SECURITY_UNAUTH,
        /* 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[] =
{
    ...
    /* 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 7.3 : LED Switch サービスへのアクセス時のセキュリティ設定

- Switch State キャラクタースティックの 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 キャラクター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 7.4 に示します。

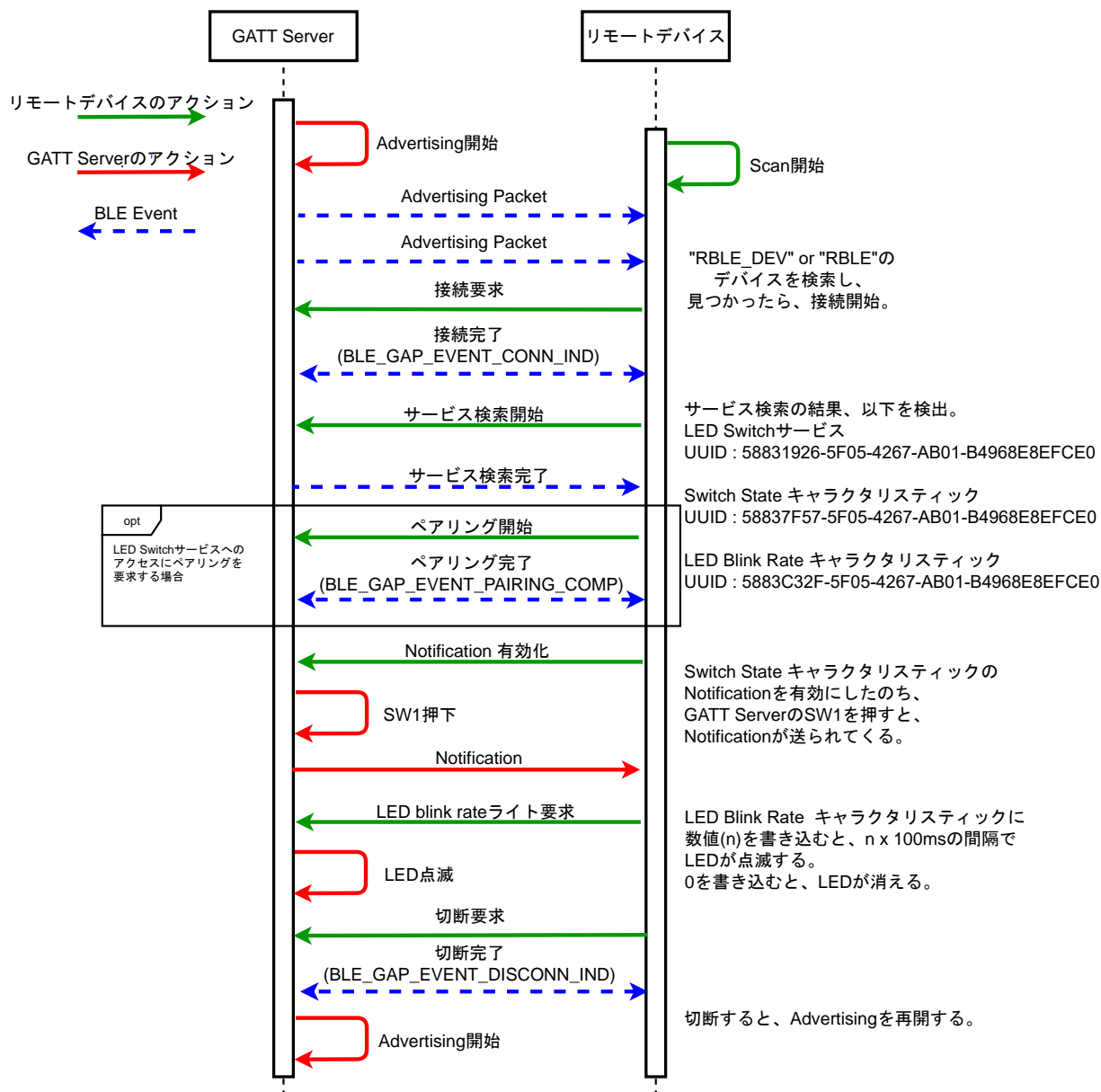


図 7.4 : GATT Server デモプロジェクトとリモートデバイスの使用例

7.2 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。
- 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 デモプロジェクトと接続する場合、接続後、下記の LED Switch クライアントコマンドが可能となります。GATT Server デモプロジェクトがペアリングを要求する設定になっている場合、LED Switch クライアントコマンド実行前に gap auth コマンドによるペアリングが必要となります。
 - Notification の有効化／無効化
lsc cmd_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 cmd_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 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 7.5 に示します。

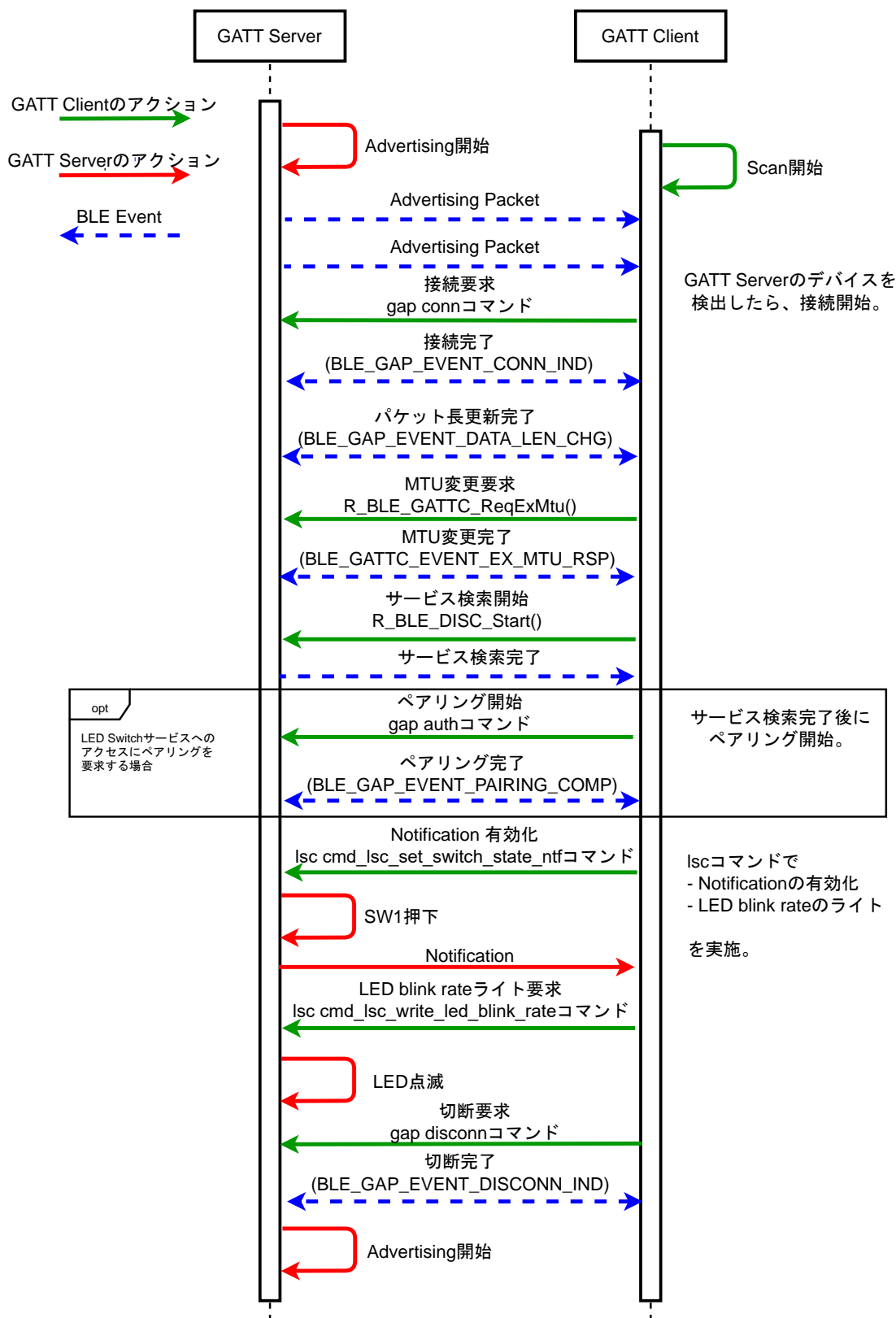


図 7.5 : GATT Client デモプロジェクトと GATT Server デモプロジェクトの使用例

7.3 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 Trial Tool Suite : R01AN4554)と接続してご使用ください。

7.4 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) [ファイル]メニューから[インポート]を選択します。

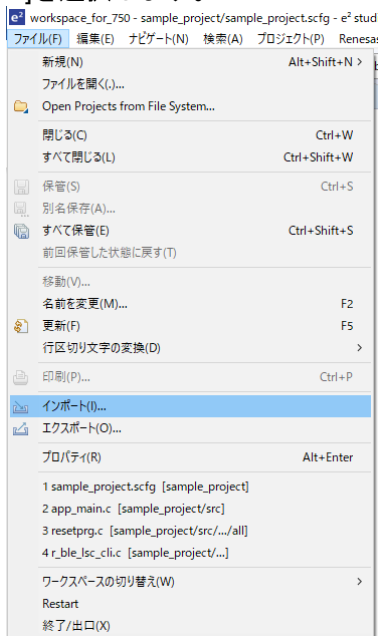


図 7.6：ファイルメニュー

(2) [インポート]ダイアログから[一般]の[既存プロジェクトをワークスペースへ]を選択して [Next]ボタンをクリックします。

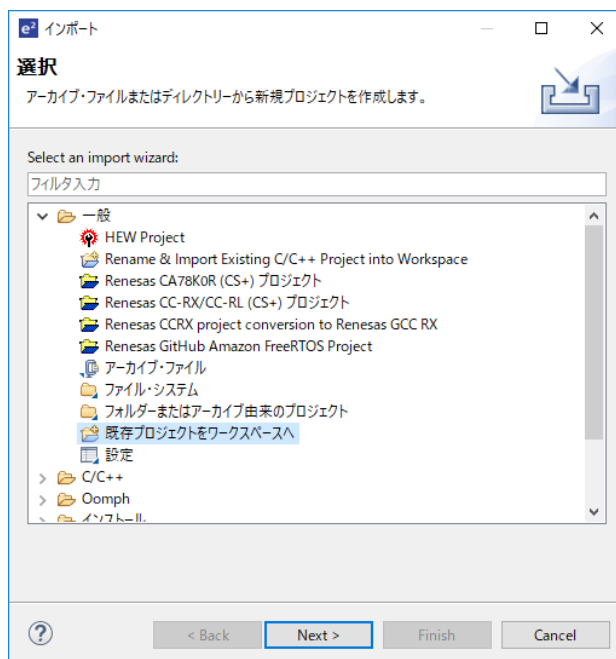


図 7.7：インポートの選択画面

- (3) [インポート]ダイアログで[アーカイブ・ファイルの選択]ラジオボタンを選択し、[参照]ボタンをクリックし、デモの zip ファイルを選択します。[Finish]ボタンをクリックすると、デモプロジェクトがインポートされます。

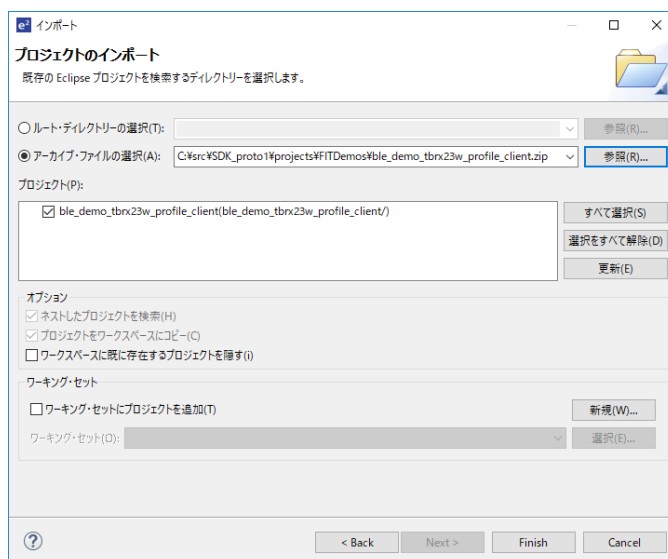


図 7.8：インポートするプロジェクトの選択画面

8. 付録

8.1 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 8.1 動作確認環境 (Rev.1.10)

| 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 V7.6.0 IAR Embedded Workbench for Renesas RX 4.12.1 |
| 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 4.12.1 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 1.10 |
| 使用ボード | Target Board for RX23W (RTK5RX23W0C00000BJ) RSSK RX23W(RTK5523W8AC00001BJ) |

8.2 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。

- ・ CS+を使用している場合
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)」
- ・ e² studio を使用している場合
アプリケーションノートRX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、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「2.6 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、e² 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 | <ul style="list-style-type: none"> - 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 | <ul style="list-style-type: none"> - 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) |

| | | | |
|--|--|--|---|
| | | | <p>BLE_CFG_BOARD_LED_SW_EN (r_gpio_rx, r_irq_rx)</p> <p>BLE_CFG_DEV_DATA_DF_BLOCK (r_flash_rx)</p> <p>- 以下のコンフィギュレーションオプションを追加。</p> <p>BLE_CFG_ABS_API_EN (抽象 API の有効／無効)</p> <p>BLE_CFG_SOFT_TIMER_EN (ソフトウェアタイマの有効／無効)</p> <p>BLE_CFG_MCU_LPC_EN (MCU 低消費電力機能の有効／無効)</p> <p>BLE_CFG_HCI_MODE_EN (HCI モードの有効／無効)</p> |
|--|--|--|---|

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、ハインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、 V_{IL} (Max.) から V_{IH} (Min.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、 V_{IL} (Max.) から V_{IH} (Min.) までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違くと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、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には、お客様の責任において行ってください。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まれます。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、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、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。
 5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
高品質水準： 輸送機器（自動車、電車、船舶等）、交通管制（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
 6. 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。
 7. 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。
 8. 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。
 9. 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。
 10. 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いたします。
 11. 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。
 12. 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.4.0-1 2017.11)

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。