Renesas Synergy™ プラットフォーム

BLE フレームワーク

R30AN0309JU0104
Rev.1.04
2018.7.25

本資料は英語版を翻訳した参考資料です。内容に相違がある場合には英語版を優先します。資料によっては英語版のバージョンが更新され、内容が変わっている場合があります。日本語版は、参考用としてご使用のうえ、最新および正式な内容については英語版のドキュメントを参照ください。

要旨 (Introduction)

本書では、Bluetooth® Low Energy (BLE) フレームワークモジュール(Framework Module)の活用方法について説明しています。本ガイドの内容を理解することにより、ユーザ独自のBLEフレームワークモジュールを追加やターゲットアプリケーションへの組込み、また付属のアプリケーションサンプルコードを参考にしたプログラムコードの作成が可能になります。他のアプリケーションプロジェクトの詳細なAPI記述や提案事項に関する参考情報は、*Synergyソフトウェアパッケージ (SSP) ユーザーズマニュアル (Synergy Software Package (SSP) Users' s Manual)* に記載されています (「BLEフレームワークの次の手順」の項を参照してください) 。これは、本モジュールの高度な利用方法について解説しており、より複雑な設計を行う上で重要な情報になります。

現時点では、RL78G1D BLEモジュール向けのBLEフレームワークが実装およびテストされています。その他のBLEモジュールのサポートは、今後のリビジョンで提供される予定です。

BLEフレームワークはBLEアプリケーションに高度なAPIを提供し、sf_ble_rl78g1dとして実装されます。BLEフレームワークはSynergyソフトウェアパッケージ (SSP) 通信フレームワーク(communication framework)を使用しており、この通信フレームワークはBLEモジュールとの通信に対してUARTドライバを有効にします。また、BLEフレームワークは、BLEプロファイルとの統一インタフェースを提供する汎用BLEプロファイルフレームワーク (g_sf_ble_onboard_profile) も統合します。RL78G1D BLEハードウェアモジュール(hardware module)の場合、汎用BLEプロファイルはBLEモジュールファームウェアによって実装されます。

必要リソース (Required Resources)

BLEフレームワークアプリケーションサンプルをビルドして実行するには、以下のものが必要です。

- Renesas SK-S7G2 Synergy MCUグループキットまたはPK-S5D9 Synergy MCUグループキット
- e² studio ISDE v5.4.0.023以降またはRenesas Synergy™用IAR組込みWorkbench® v7.71.3以降
- Synergyソフトウェアパッケージ (SSP) 1.4.0以降またはSynergy スタンドアロンコンフィグレータ (SSC) 5.4.0.023以降
- Segger J-link® USBドライバ
- マイクロUSBケーブル
- USB 2.0フラッシュドライブ
- **BLE Scanner APK**がインストールされたAndroid携帯
- 必要なすべてのRenesasソフトウェアをRenesas Synergy™ウェブ (https://www.renesas.com/ja-jp/products/synergy.html) からダウンロードします。

前提条件と対象ユーザ (Prerequisites and Intended Audience)

本アプリケーションノートは、Renesas Synergy e² studio ISDEおよびSynergyソフトウェアパッケージ (SSP) に関して、ある程度経験があるユーザを前提としています。本アプリケーションノートの手順に進む前に、*SSPユーザーズマニュアル(SSP Users Manual)*の手順に従って**Blinky**プロジェクトをビルドおよび実行してください。そうすることで、e² studioおよびSSPに慣れ、ボードへのデバッグ接続が適切に機能していることを確認できるようになります。また、本アプリケーションノートは、BLEとその通信プロトコルに関して、ある程度知識があるユーザを前提としています。

対象ユーザは、Renesas Synergy™ S3、S5、S7 MCUシリーズを使用してBLEインタフェースでアプリケーションを開発しようとしているユーザです。

目次

1. BLEフレームワークの概要 (BLE Framework Overview)

Bluetooth® Low Energy (BLE) は**Bluetooth Smart**と呼ばれることもあります。これはClassic Bluetoothのライトウェイトサブセットであり、Bluetooth 4.0コア仕様の一環として導入されました。Classic Bluetoothとは対照的に、BLEは消費電力を大幅に低減するように設計されています。BLEにより、近接したIoTデバイス間でのデータ転送が可能になります。

アプリケーション開発者は、BLEスタック(stack)がそのAPIを使用して提供する機能にアクセスします。各ベンダーから提供されるBLEスタックAPIは標準化されていません。このため、アプリケーション開発者は異なるBLEスタックに移植する際にコードを更新する必要があります。

アプリケーションからベンダー固有BLEスタックコードを直接呼び出さないBLEスタック向け汎用インタフェースの提供により、Synergy BLEフレームワークはこの問題に対処します。汎用APIを使用すると、アプリケーション開発が単純で移植可能になります。

1.1 サポートされる機能 (Supported feature)

Synergy BLEフレームワークは以下の機能をサポートします。

- ThreadX® RTOS対応およびスレッドセーフ(thread safe)
- Bluetooth v4.2準拠のフレームワーク(framework)
- Generic Access Profile (GAP) 機能
  — ユーザ定義アドバタイジングデータ(advertising date)
  — セキュリティモード(security mode)1および2
  — ペリフェラルロール(peripheral role)およびセントラルロール(central role)
  — ホワイトリストサポート(white list support) (最大6デバイス)
  — ボンディングサポート(bonding support)
- Generic Attribute Profile (GATT) 機能
  — GATTクライアントおよびサーバ
- Generic Attribute Profile (GATT) API
- Generic Access Profile (GAP) API
- 汎用オンボードプロファイルAPI

2. BLEフレームワークモジュールの動作概要 (BLE Framework Module Operational Overview)

本項では、Synergy BLEフレームワークソフトウェアアーキテクチャの概要を示し、ユーザのアプリケーションレベルの動作フローシーケンスで、BLEフレームワークの一環として使用される主要なSSPモジュールを説明します。

2.1 BLEフレームワークアーキテクチャの概要 (BLE Framework architecture overview)

BLEフレームワークは、アプリケーションの共通インタフェース(common interface)を提供します。このインタフェースの実装はモジュールごとに固有です。現在、Synergy BLEフレームワークはRL78G1D BLEモジュール向けに実装されたインタフェースを定義しています。各実装は対応するBLEデバイスドライバと連携します。BLEデバイスドライバ(device driver)は、SSP通信フレームワーク(ssp communication framework) (g_sf_comms) を使用しています。この通信フレームワークは、SSP HALコンポーネントである汎用非同期送受信回路 (UART) ドライバ、データトランスファコントローラ (DTC) ドライバ、汎用PWMタイマ (GPT) ドライバと連携して、BLEモジュールと通信します。

以下の図は、SSPのBLEフレームワークに関するハイレベルのソフトウェアアーキテクチャの概要を示しています。

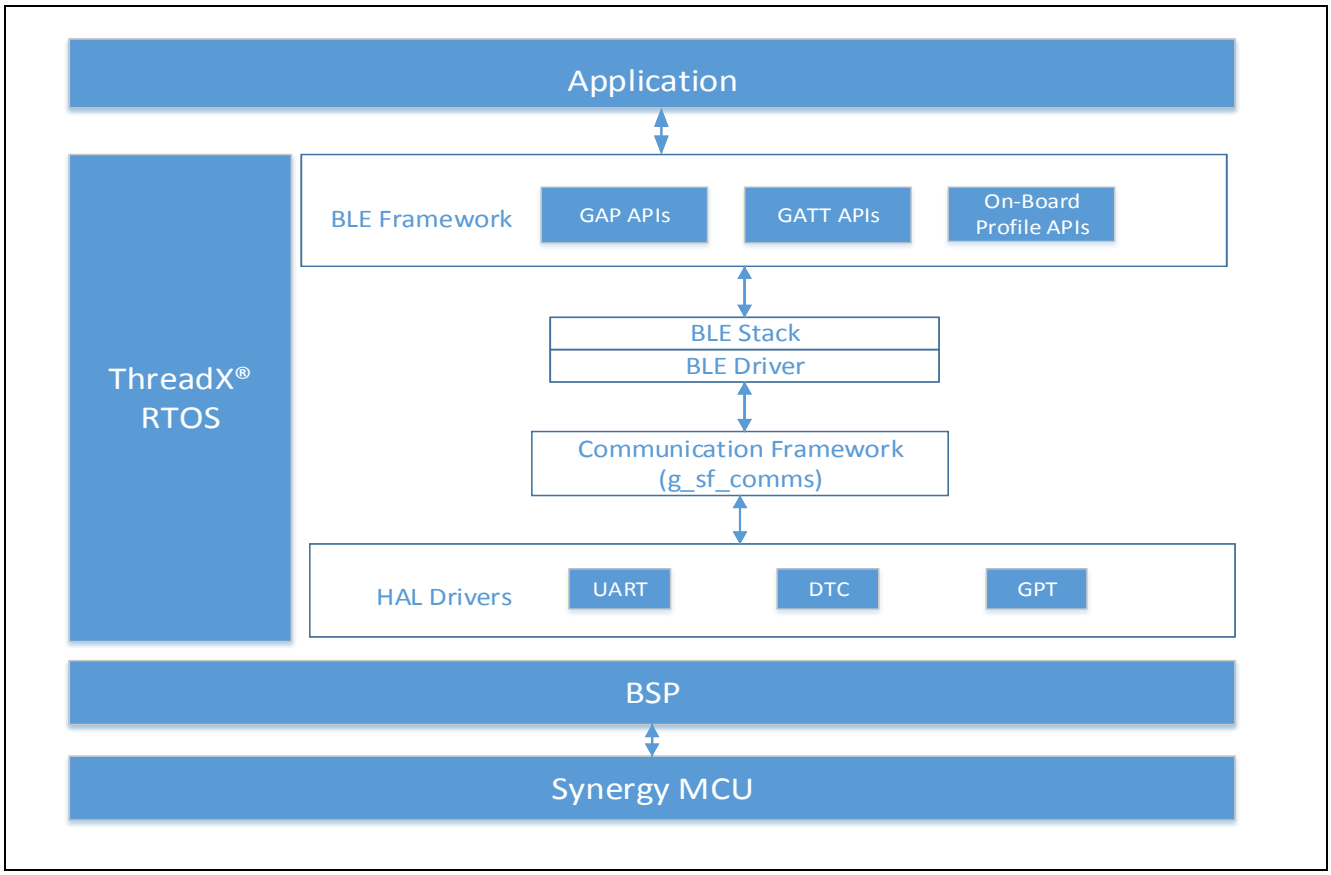


図1　標準的なBLEモジュールアーキテクチャのタイプ

Synergy BLEフレームワークは以下のブロックで構成されます。

- GAPおよびGATT API
- オンボードプロファイルAPI(On-Board profile API)
- BLEスタック(BLE stack)

GAPおよびGATT API (GAP and GATT APIs)

BLEフレームワークは、アプリケーションがBLEモジュールを設定(configure)およびプロビジョニング(provisioning)するための汎用インタフェースを提供します。BLEモジュールには、Bluetooth Smart規格ファミリによって規定された様々な設定パラメータ(configuration parameter)があります。個別のデバイスドライバやBLEモジュールは必ずしもすべての設定パラメータをサポートしていない可能性があります。プロビジョニングAPI(provisioning API)は、BLEインタフェースの動作モード、セキュリティモード(security mode)、セキュリティ鍵(security key)、およびボンディングモード(bonding mode)を設定するために必要最小限のメカニズムを提供します。また、GAP/GATT層(GAP/GATT layer)向けのAPIも提供します。

オンボードプロファイルAPI (On-Board profile API)

オンボードプロファイルAPIは、BLEモジュールファームウェアによって実装されるBLEプロファイル(BLE profile)との統一インタフェースを提供します。

BLEスタック (BLE Stacks)

通常、BLEモジュールホストスタック(BLE module host stack)はBLEモジュールベンダーから提供されます。BLEモジュールは、ホストMCUとBLEモジュールの間のHW/SW分割に応じて、通常は3種類に分かれます。RL78G1D BLEモジュールはネットワークコントローラ実装アーキテクチャ(Network Controller Implementation)の一部です。このアーキテクチャでは、BLEリンク層(BLE link layer)、GAP、GATT、およびオンボード

プロファイルの実装がすべてBLEチップセットに組み込まれています。本モジュールは、SSPによって提供されるsf_commsフレームワークでMCUとインタフェースをとります。

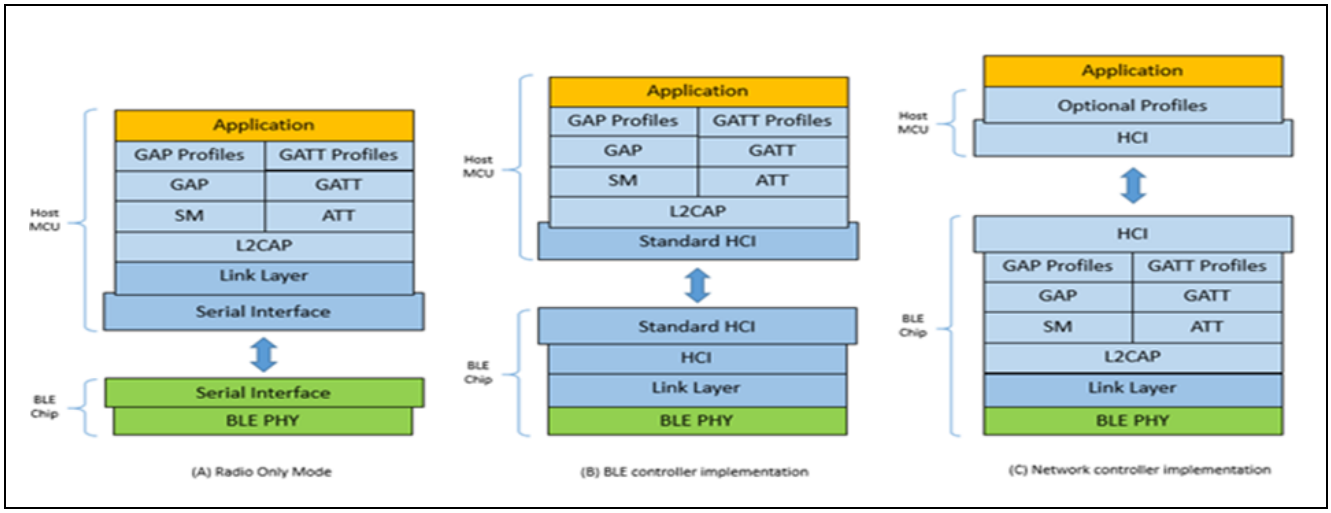


図2　BLEモジュールアーキテクチャのタイプ

1. BLE無線専用モード(BLE radio-only mode)：

   リンク層、L2CAP、GATT、GAPの各層、プロファイル、およびアプリケーションはホストMCUで動作します。物理層はBLEチップセットで動作します。

2. BLEコントローラ実装(BLE controller implementation)：

   リンク層はBLEチップセット、L2CAP、および上位のBLEプロトコル（GATT、GAP）の各層で動作します。プロファイルおよびアプリケーションはホストMCUで動作します。

3. ネットワークコントローラ実装(Network controller implementation)：

   リンク層、L2CAP、GATT、GAPの各層、および汎用プロファイルはBLEチップセットで動作します。オプションのプロファイルおよびアプリケーションはホストプロセッサで動作します。

2.2　BLEフレームワークインスタンス（BLE framework instance）

インスタンスは以下のいずれかへのポインタを含む構造体です。

- BLEフレームワーク制御構造体(BLE Framework control structure)
- BLEフレームワーク設定構造体(BLE Framework configuration structure)
- BLEフレームワークAPI構造体(BLE Framework APIs structure)
- オンボードプロファイルAPI構造体(On-board profiles APIs structure)

```
/** BLE instance */
typedef struct st_sf_ble_instance
{
  sf_ble_ctrl        * p_ctrl;     ///<  Pointer to the control structure
                                   for this instance
  sf_ble_cfg   const * p_cfg;      ///<  Pointer to the configuration
                                   structure for this instance
  sf_ble_api_t const * p_api;      ///<  Pointer to the API structure for
                                   this instance
} sf_ble_instance_t;
```

以下の構造体はSynergy BLEフレームワークインスタンスです。

BLEフレームワーク制御構造体：

この構造体はすべてのBLEフレームワークAPIで使用されます。

```
/** BLE Framework control structure */
typedef struct sf_ble_ctrl
{

void * p_driver_handle; ///< Storage for information needed for each BLE
device driver in the system
} sf_ble_ctrl_t;
```

この構造体にはドライバハンドル(driver handle)へのポインタ(pointer)が含まれています。これは、BLEデバイスドライバに必要な情報を格納するためにフレームワークで使用されます。

BLEフレームワーク設定構造体：

この構造体はopen () APIに渡され、ユーザはこの構造体を使用してBLEモジュールを設定することが可能です。この設定は、初期化（openなど）またはプロビジョニング（provisioningSetなど）のいずれかの実行時に適用されます。BLEモジュールでサポートされていない設定パラメータ(configuration parameter)はフレームワークで無視されます。

```
/** BLE configuration information */
typedef struct sf_ble_cfg
{
    uint8_t             bd_addr[SF_BLE_ADDR_LEN];   ///< BLE address
    sf_ble_addr_type_t  own_addr_type;              ///< self address type
    uint8_t             max_slaves;                 ///< Maximum slaves
allowed to be connected

uint8_t             update_bd_addr;         ///< Set this to true to
                                                update bluetooth address
                                                during SF_BLE_Open

    uint16_t            scan_interval;          ///< BLE scan interval
for receiving advertisement

    uint16_t            scan_window;          ///< Period of time during
                                              which advertising data is
                                              received at the scan interval.

uint16_t            disc_time;            ///< Duration for which
    the device remain discoverable

    uint16_t            con_interval;         ///< Interval for transmitting
                                              and receiving data periodically
                                              after connection establishment

    uint16_t            slave_latency;        ///< Period of time during
                                        which data is transmitted
              and received at the connection
              interval

    uint16_t            sup_timeout;          ///< Link loss time-out
    void const          * p_extend;           ///<  Instance specific
configuration
} sf_ble_cfg_t;
```

BLEフレームワークAPI構造体

この構造体には、所定のモジュールに固有のBLEフレームワークAPIへのポインタが含まれています。これらのAPIの詳細については、3項「BLEフレームワークモジュールAPIの概要」を参照してください。

2.3 BLEフレームワークモジュールの動作フロー (BLE framework module operational flow)

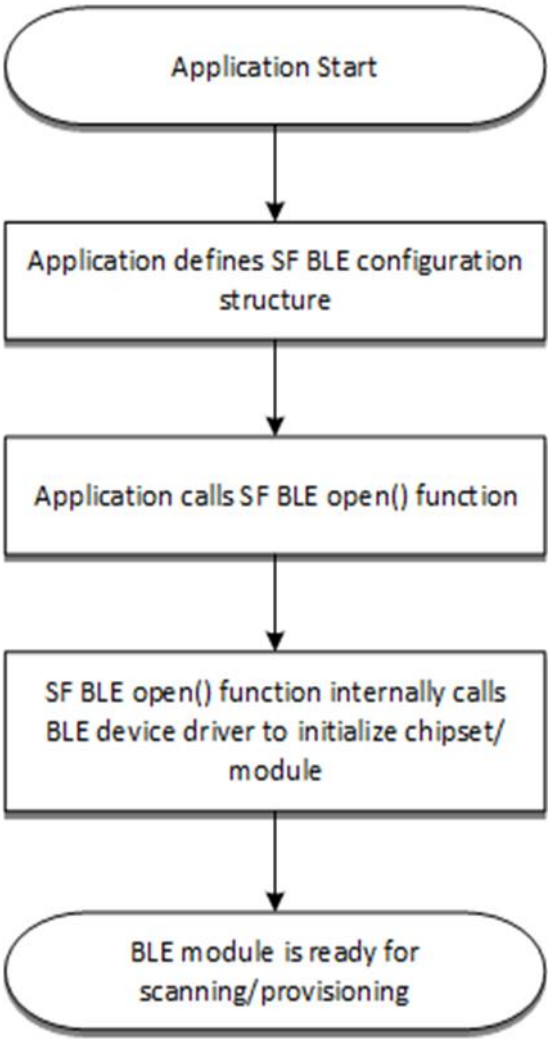
アプリケーションでBLEフレームワークモジュールを使用する手順は以下のとおりです。

1. BLEハードウェアモジュールを初期化します。
2. GATT層ロール（GATTクライアント、GATTサーバなど）を選択します。スレーブ（ペリフェラル）デバイスはGATTサーバになり、マスタ（セントラル）デバイスはGATTクライアントになるのが最も一般的です。

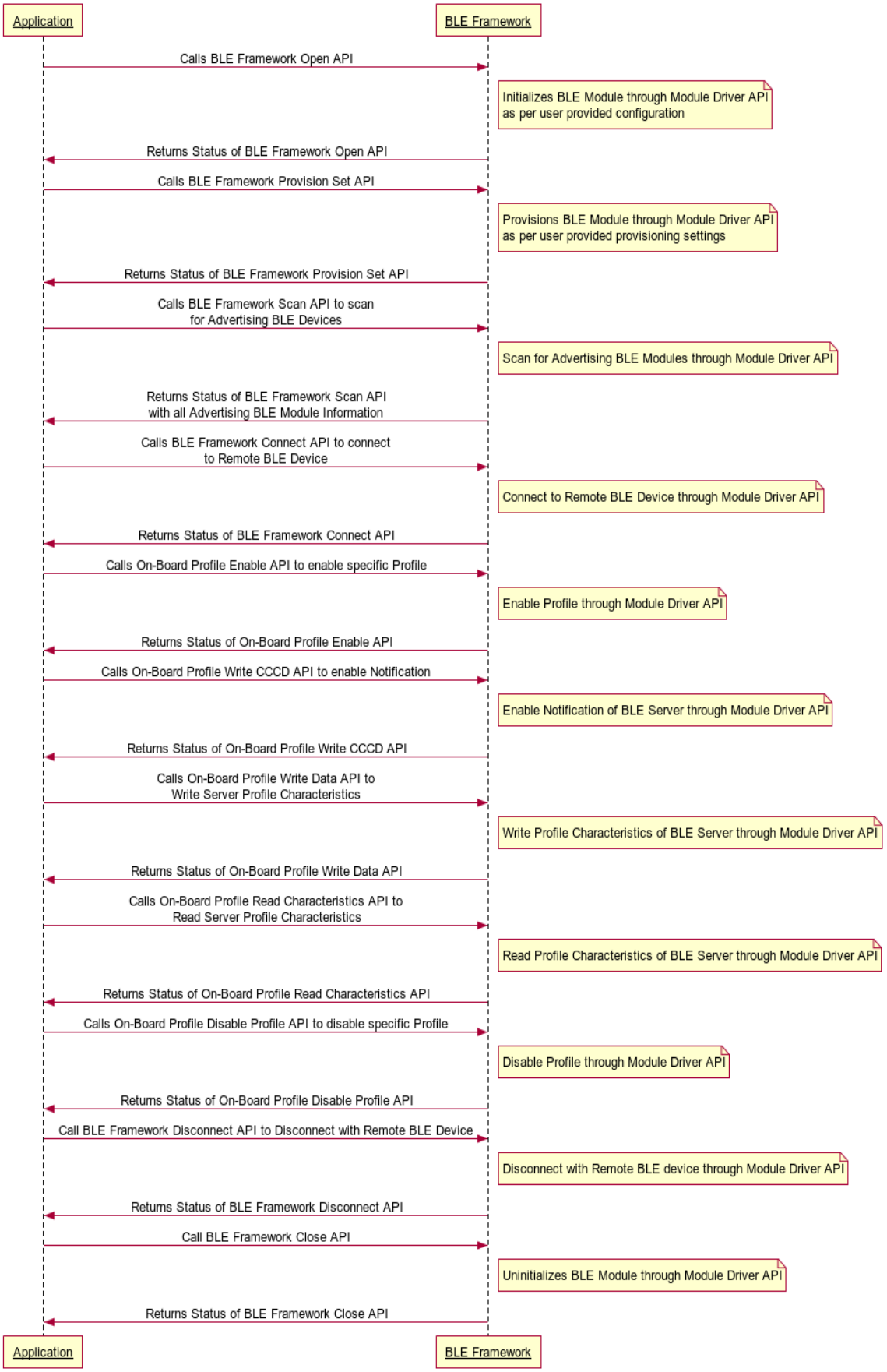
汎用（オンボード）プロファイルAPIまたはGAP/GATT APIを使用してアプリケーションを開発します。

2.3.1 BLEモジュールの初期化フローシーケンス (BLE module initialization flow sequence)

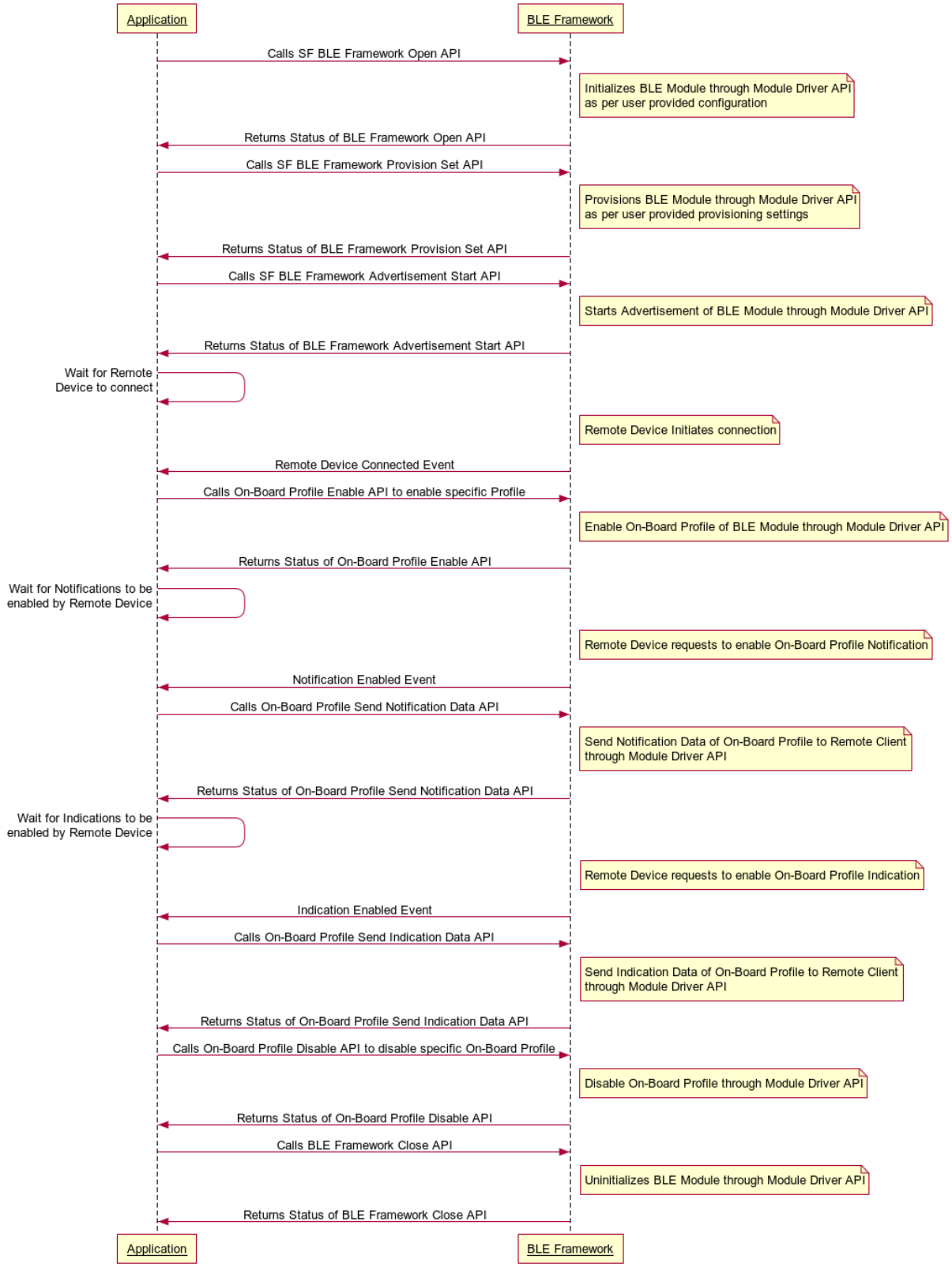
以下に示すBLEモジュールの初期化シーケンスはSynergy自動生成コード(auto-generated)の一部です。



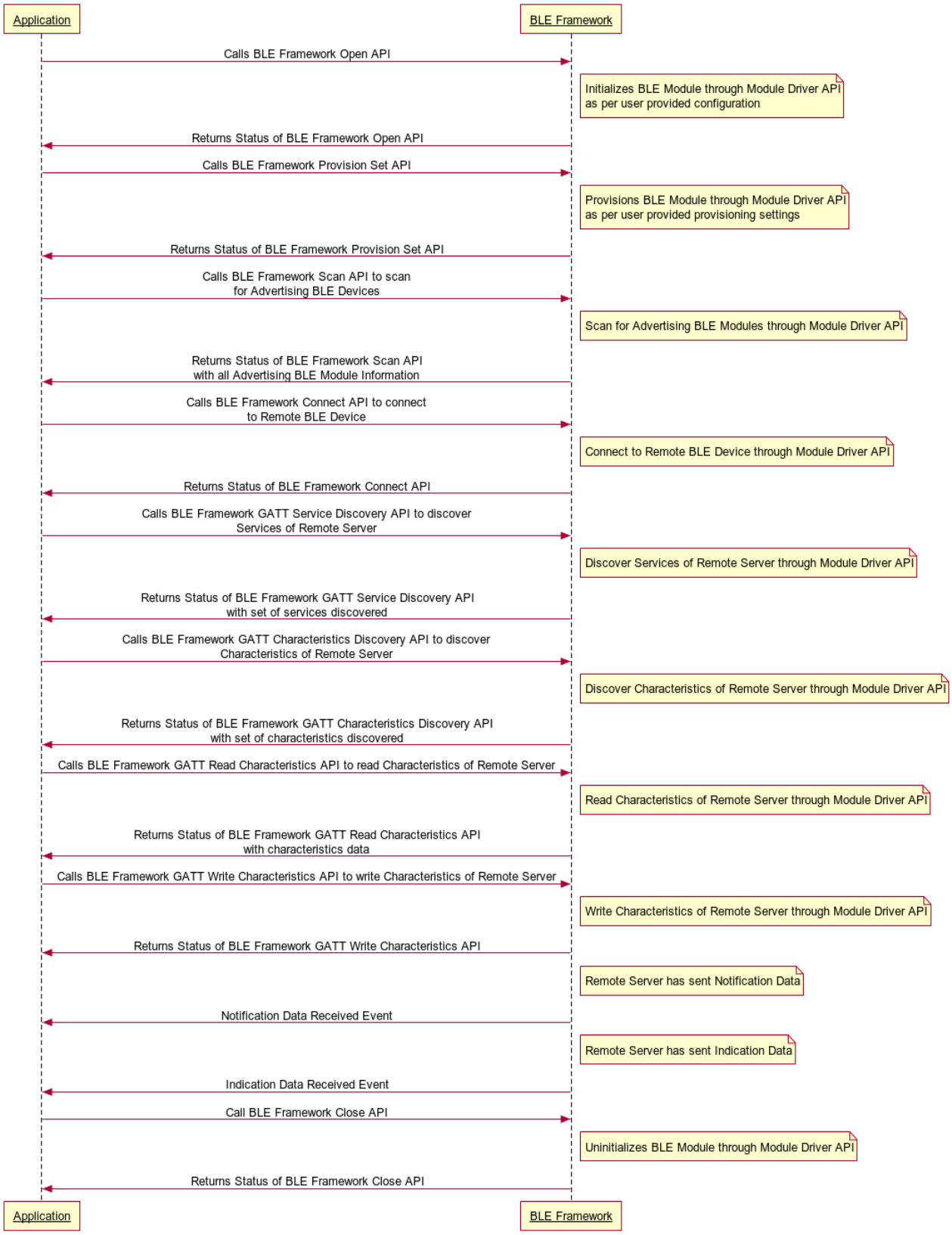
2.3.2 オンボードプロファイルに基づくクライアントアプリケーションのフローシーケンス (On-Board Profile based client application flow sequence)



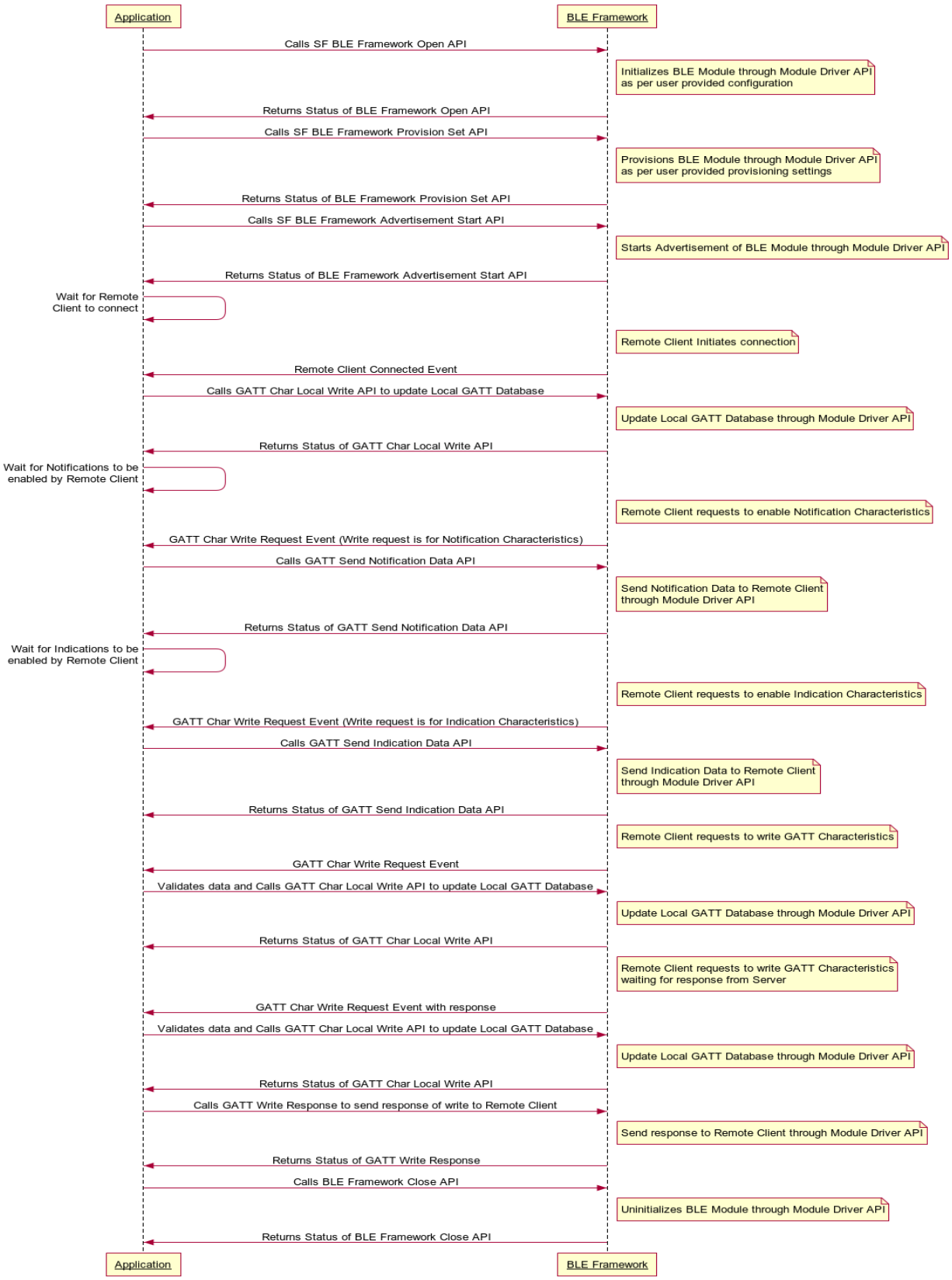
2.3.3 オンボードプロファイルに基づくサーバアプリケーションのフローシーケンス (On-board Profile based server appliation flow sequence)



2.3.4 GAP/GATTに基づくクライアントアプリケーションのフローシーケンス (GAP/GATT based client application flow sequence)



2.3.5 GAP/GATTに基づくサーバアプリケーションのフローシーケンス (GAP/GATT based server application sequence)



2.4 BLEフレームワークセキュリティ (BLE framework security)

Security Managerは、通信リンクの暗号化に使用されるセキュリティ鍵を生成および交換する機能をBLEプロトコルスタックに提供します。Security Managerには以下の2つの機能があります。

- **イニシエータ(Initiator)**
  これはGAPマスタ／セントラルデバイス(Maseter/Central device)です。
- **レスポンダ(Responder)**
  これはGAPスレーブ／ペリフェラルデバイス(Slave/Peripheral device)です。

イニシエータはセキュリティ手順を開始するマスタデバイスですが、スレーブデバイスはセキュリティ手順を開始するようにイニシエータに非同期で要求することが可能です。

2.4.1 BLEセキュリティモード (BLE security mode)

BLEセキュリティは各モードに対応するレベルに合わせてモードを提供します。セキュリティモードおよびレベルは、認証済み(authenticated)または未認証(unauthenticated)であるのかの、暗号化(encryption)またはデータ署名(signing) であるのかのペアリングに対するサポートの組み合わせです。ペアリングは各種セキュリティ要件を満たす必要があります。以下に示す2つのタイプのペアリングが利用可能です。

- 認証済みペアリング(authenticated paring)。このペアリングでは、デバイスがMITM（中間者）攻撃から保護されます。
- 未認証ペアリング(unauthenticated paring)。このペアリングでは、デバイスがMITMから保護されません。

セキュリティモード1

セキュリティレベル1：セキュリティなし

セキュリティレベル2：未認証ペアリング（暗号化あり）

セキュリティレベル3：認証済みペアリング（暗号化あり）

セキュリティレベル4：認証済みLEセキュア接続ペアリング（暗号化あり）

セキュリティモード2

セキュリティレベル1：未認証ペアリング（データ署名あり）

セキュリティレベル2：認証済みペアリング（データ署名あり）

注： RL78G1D BLEモジュールはセキュリティモード1のセキュリティレベル4をサポートしていません。

2.4.2 BLEセキュリティ手順 (BLE security procedure)

BLEセキュリティには以下の手順があります。

- **ペアリング(paring)**
  この手順は、一時的な暗号鍵(temporary encryption key)を生成して通信リンク(communication link)を暗号化するために使用されます。
  この暗号化された通信リンクで恒久的な暗号鍵(permanent encryption key)を別の通信のために共有することが可能です。

- **ボンディング(bonding)**
  これは恒久鍵のペアリングと格納の組み合わせです。ペアリングの後に、恒久鍵は2つのデバイス間で恒久的な連結(permanent bond)が作られ、不揮発性メモリに格納されます。後続の通信では、デバイスでボンディング手順(bonding procedure)を実行する必要はありません。

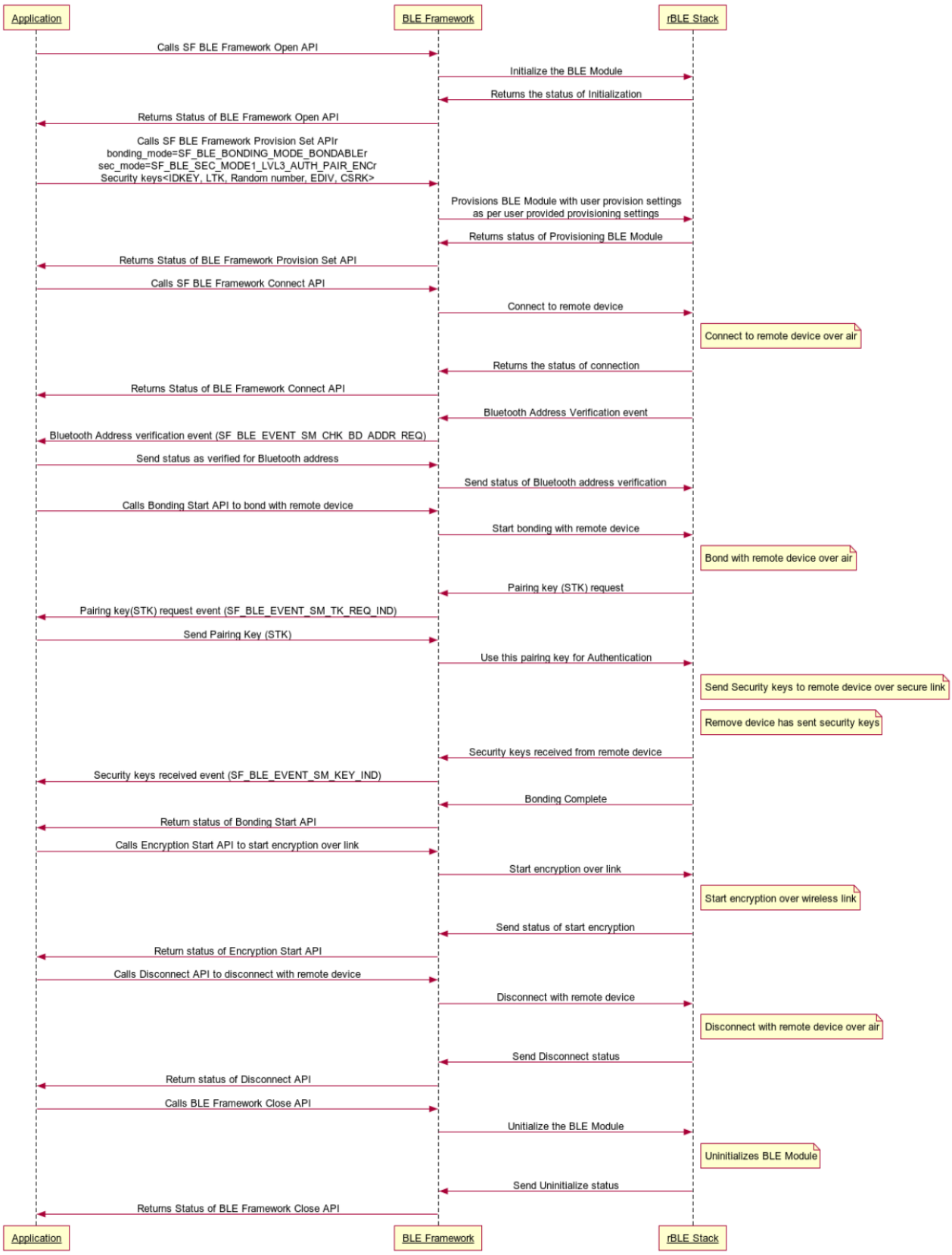
- **暗号化の確立(Encryption Establishment)**
  通信は恒久鍵で暗号化されます。
  ペアリングでは、接続の存続期間にわたって継続するセキュアリンク(secure link)が作成されます。これに対し、ボンディングでは、ボンドと呼ばれる恒久的な対応関係が作成されます。

2.4.3 BLEセキュリティ段階 (BLE security phases)

BLEセキュリティは、後述の図に示すように3つの段階(phase)を通ります。2つのデバイスがGAP接続手順(GAP conection procedure)で接続を確立した後、セキュア通信リンクを確立するための3つの段階が続きます。

- 段階1 (ペアリング段階、情報の共有)
  最初に段階1では、一時的な鍵の生成に必要なすべての情報が2つのデバイス間で共有されます。
- 段階2 (ペアリング段階、一時鍵の共有)
  この段階では、一時的な暗号鍵 (短期鍵 (STK)) が両方のデバイスで生成されます。これは接続の暗号化に使用されます。この暗号化リンクは別の通信に使用することが可能です。ピアデバイス(peer device)が接続状態を維持している限り、この通信リンクは暗号化されたままです。
- 段階3 (恒久鍵のボンディング、共有、および格納)
  ボンディング(bonding)が必要な場合、デバイスはこの段階に入ります。この段階では、段階2で2つの一時鍵を使用して確立された暗号化リンクにより、恒久鍵 (長期鍵 (LTK)) が2つのデバイス間で交換されます。その後、これらの恒久鍵は不揮発性メモリに格納され、各接続でデバイスに利用できるようになります。

2.4.4 BLEフレームワークの認証フローシーケンス (BLE framework authentication flow sequence)



2.5 BLEフレームワークの制限 (BLE framework limitations)

1. BLEフレームワークはRL78G1D BLEハードウェアモジュールのみでテストされています。その他のBLEモジュールのサポートは、今後のバージョンで追加される予定です。
2. RL78G1Dを使用するBLEフレームワークにはコンパイル警告(complilation warning)があります。これらの警告はすべてサードパーティ製のRL78G1Dドライバコードに含まれています。BLEフレームワークファイルに警告は含まれていません。これらの警告はユーザアプリケーションには影響しません。
3. BLEフレームワークのカスタムプロファイルサポート(custom profile support)は、RL78G1DタイプのBLEハードウェアモジュールのみに制限されています。
4. HIDプロファイルクライアントモード(HID profile client mode)は、RL78G1D BLEハードウェアモジュールでサポートされていません。結果として、HIDプロファイルのBLEフレームワーク実装もHIDプロファイルクライアントモードをサポートしません。RL78G1DのBLEフレームワークを使用するアプリケーションは、クライアントモードでHIDプロファイルを使用することはできません。
5. 複数のスレーブBLEデバイス(slave BLE device)は、RL78G1D BLEモジュールに接続することはできません。

3. BLEフレームワークモジュールAPIの概要 (BLE Framework Module API Overview)

本項では、利用可能なAPIのリストを示し、各APIの機能、パラメータ、戻り値(return value)などについて簡単に説明します。詳細については、*SSPユーザーズマニュアル(SSP User's Manual)*のAPIリファレンスの項を参照してください。

3.1 BLE GAP API

3.1.1 open

説明：

本APIはデータ転送のインタフェースを初期化します。ドライバの初期設定、ドライバリンクの有効化、割り込みの許可、およびデータ転送に向けたデバイスの準備を行います。

パラメータ：

名前	方向	内容
p_ctrl	In, Out	BLEモジュールの制御部へのポインタ (sf_ble_ctrlを参照)
p_cfg	In	BLE設定構造体sf_ble_cfg_tへのポインタ (sf_ble_cfgを参照)

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*open)(sf_ble_ctrl_t * const p_ctrl,
const sf_ble_cfg_t * p_cfg);
```

3.1.2 close

説明：

本APIはインタフェースを終了(de-initialize)、BLEモジュールを低消費電力モードにするか電源オフにすることが可能です。また、BLEモジュールドライバにおいて、ドライバを閉じ、ドライバリンクを無効にし、割り込みを禁止します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*close)(sf_ble_ctrl_t * const p_ctrl);
```

3.1.3 infoGet

説明：

本APIはBLEモジュール情報（チップセット情報、RSSI値など）を取得します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_handle	In	接続ハンドルへのポインタ
p_ble_info	Out	モジュール情報へのポインタ

戻り値：

BLEモジュールから取得された以下の情報を返します。

- チップセット／ドライバ情報文字列
- RSSI値（符号なし16ビット整数）

関数プロトタイプ：

```
ssp_err_t (*infoGet)(sf_ble_ctrl_t * const p_ctrl, sf_ble_conn_handle_t * p_handle, sf_ble_info_t * p_ble_info);
```

3.1.4 provisionGet

説明：

provisionGet()関数は、BLE GAPプロビジョニング情報を取得します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_ble_provisioning	Out	現在のプロビジョニング情報

戻り値：

以下のパラメータを返します。

- GAP名(Gap name)
- ブロードキャストモードフラグ(Broadcast mode flag)
- ボンディングモード(Bonding mode)
- セキュリティモード(Security mode)
- GAPロール(GAP role)（セントラル／マスタまたはペリフェラル／スレーブ）
- GAPユーザイベントコールバック(GAP user event callback)

関数プロトタイプ：

```
ssp_err_t (*provisionGet)(sf_ble_ctrl_t * const p_ctrl,  sf_ble_provisioning_t * p_ble_provisioning);
```

3.1.5 provisionSet

説明：

provisionSet()関数は、BLEモジュールをプロビジョニングします。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_ble_provisioning	In	BLEプロビジョニング構造体へのポインタ

戻り値：

SSPエラーステータス(SSP Error status)

関数プロトタイプ：

```
ssp_err_t (*provisionSet)(sf_ble_ctrl_t * const p_ctrl, const sf_ble_provisioning_t * p_ble_provisioning);
```

3.1.6 scan

説明：

本APIは利用可能なBLEデバイスをスキャンし、呼び出し元にリストを返します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_scan	Out	スキャン構造体へのポインタ
P_cnt	In, Out	スキャンされたBLEデバイス数へのポインタ
P_scan_info	In	スキャン情報構造体へのポインタ

戻り値：

scan()関数は、BLEモジュールによって以下のパラメータでスキャンされたBLEデバイスのリストを返します。

- 48ビットBluetoothアドレス
- RSSI
- スキャンデータ

関数プロトタイプ：

```
ssp_err_t (*scan)(sf_ble_ctrl_t * const p_ctrl, sf_ble_scan_t * p_scan, uint8_t * p_cnt, sf_ble_scan_info_t * p_scan_info);
```

3.1.7 advertisementStart

説明：

advertisementStart()関数は、アドバタイジング(advertisement)を開始します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_advt_info	In	アドバタイジング情報構造体へのポインタ

戻り値：

SSPエラーステータス(SSP Error status)

関数プロトタイプ：

```
ssp_err_t (*advertisementStart)(sf_ble_ctrl_t * const p_ctrl, sf_ble_adv_info_t * const p_advt_info);
```

3.1.8 advertisementStop

説明：

advertisementStop()関数は、アドバタイジングを停止します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*advertisementStop)(sf_ble_ctrl_t * const p_ctrl);
```

3.1.9 whitelistAdd

説明：

whitelistAdd()関数は、アドバタイジング、スキャン、および接続の各要求に関するホワイトリスト(white list)にデバイスを追加します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_bd_addr	In	BLEアドレスへのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*whitelistAdd)(sf_ble_ctrl_t * const p_ctrl, const uint8_t * p_bd_addr);
```

3.1.10 whitelistDel

説明：

whitelistDel()関数は、アドバタイジング、スキャン、および接続の各要求に関するホワイトリストからデバイスを削除します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_bd_addr	In	BLEアドレスへのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*whitelistDel)(sf_ble_ctrl_t * const p_ctrl, const uint8_t *
p_bd_addr);
```

3.1.11 bondingStart

説明：

bondingStart()関数は、リモートデバイス(remote device)とのボンディングを開始します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_bd_addr	In	BLEアドレスへのポインタ
p_handle	In	接続ハンドルへのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*bondingStart)(sf_ble_ctrl_t * const p_ctrl,
                          sf_ble_conn_handle_t * p_handle,
  const uint8_t *p_bd_addr,
  sf_ble_bonding_start_t *p_bonding_start);
```

3.1.12 bondingResponse

説明：

bondingResponse()関数は、ボンディング要求に応答します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_bd_addr	In	BLEアドレスへのポインタ
p_handle	In	接続ハンドルへのポインタ
P_bonding_resp	In	ボンディングアドレスへのポインタ

戻り値：SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*bondingResponse)(sf_ble_ctrl_t * const p_ctrl,
                             sf_ble_conn_handle_t * p_handle,
                             const uint8_t * p_bd_addr,
                             sf_ble_bonding_response_t * p_bonding_resp);
```

3.1.13 connect

説明：

connect()関数は、リモートデバイスに接続します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_conn	In	接続情報へのポインタ
p_handle	Out	接続ハンドルへのポインタ

戻り値：

接続ハンドルを返します。

関数プロトタイプ：

```
ssp_err_t (*connect)(sf_ble_ctrl_t * const p_ctrl, sf_ble_connection_t
const * const p_conn,              sf_ble_conn_handle_t * p_handle);¥
```

3.1.14 disconnect

説明：

disconnect()関数は、リモートデバイスから切断します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
p_handle	Out	接続ハンドルへのポインタ

戻り値：

接続ハンドルを返します。

関数プロトタイプ：

```
ssp_err_t (*disconnect)(sf_ble_ctrl_t * const p_ctrl,
                        sf_ble_conn_handle_t * p_handle);
```

3.1.15 listen

説明：

listen()関数は、リモートデバイスからの接続要求をリッスンします。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）

戻り値：

接続ハンドルを返します(Returns the connection handle)

関数プロトタイプ：

```
ssp_err_t (*listen)(sf_ble_ctrl_t * const p_ctrl);
```

3.2 BLE GATT API

3.2.1 gattCharWriteLocal

説明：

gattCharWriteLocal()関数は、ローカルGATTデータベースを更新します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
Char_handle	In	特性ハンドル
Data_length	In	書き込むデータ長
P_data	In	データへのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (* gattCharWriteLocal)(sf_ble_ctrl_t * const p_ctrl, uint16_t
                          char_handle, uint16_t data_length,
                          uint8_t * const p_data);
```

3.2.2 gattServiceDiscovery

説明：

gattServiceDiscovery()関数は、リモートデバイスのGATTサービスを発見します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル
P_sf_ble_svc_dscv_req	In	サービス発見要求へのポインタ
P_sf_ble_svc_dscv_rsp	Out	サービス発見応答へのポインタ
P_rsp_cnt	In, Out	入力サイズ（応答に格納される可能性があるサービス発見結果の最大数を指定）、出力（応答に格納されるサービス発見結果の数を指定）

戻り値：

サービス発見応答へのポインタ、出力（応答に格納されるサービス発見結果の数を指定）を返します。

関数プロトタイプ：

```
ssp_err_t (* gattServiceDiscovery)(sf_ble_ctrl_t * const p_ctrl,
                                   sf_ble_conn_handle_t * p_handle,
                                   sf_ble_service_discovery_req_t const
                                   * const p_sf_ble_svc_dscv_req,
                                   sf_ble_service_discovery_rsp_t *
                                   const p_sf_ble_svc_dscv_rsp,
                                   uint32_t * const p_rsp_cnt);
```

3.2.3 gattCharDiscovery

説明：

gattCharDiscovery()関数は、リモートデバイス(remote device)のGATT特性を発見します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル(Connection handle)
P_sf_ble_char_dscv_req	In	特性発見要求へのポインタ(Pointer to characteristics discovery request)
P_sf_ble_char_dscv_rsp	Out	特性発見応答へのポインタ(Pointer to characteristics discovery response)
P_rsp_cnt	In, Out	入力サイズ（応答に格納される可能性があるサービス発見結果の最大数を指定）、出力（応答に格納されるサービス発見結果の数を指定）

戻り値：

特性発見応答(characteristics discovery response)へのポインタ、出力（応答に格納される特性発見結果の数を指定）を返します。

関数プロトタイプ：

```
ssp_err_t (* gattCharDiscovery)(sf_ble_ctrl_t * const p_ctrl,
                                sf_ble_conn_handle_t * p_handle,
                                sf_ble_char_discovery_req_t const *
                                const p_sf_ble_char_dscv_req,
                                sf_ble_char_discovery_rsp_t * const
                                p_sf_ble_char_dscv_rsp, uint32_t * const
                                p_rsp_cnt);
```

3.2.4 gattCharDescDiscovery

説明：

gattCharDescDiscovery()関数は、リモートデバイスのGATT特性ディスクリプタ(GATT characteristic descriptor)を発見します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル
Start_handle	In	発見で使用されるハンドル範囲のセットの開始ハンドル
End_handle	In	発見で使用されるハンドル範囲のセットの終了ハンドル
P_sf_ble_chardesc_dscv_rsp	Out	特性ディスクリプタ発見応答へのポインタ
P_rsp_cnt	In, Out	入力サイズ（応答に格納される可能性があるサービス発見結果の最大数を指定）、出力（応答に格納されるサービス発見結果の数を指定）

戻り値：

特性ディスクリプタ発見応答(characteristics discovery response)へのポインタを返します。

関数プロトタイプ：

```
ssp_err_t (* gattCharDescDiscovery)(sf_ble_ctrl_t * const p_ctrl,
                                    sf_ble_conn_handle_t * p_handle,
                    uint16_t start_handle, uint16_t end_handle,
                    sf_ble_char_desc_discovery_rsp_t * const
        p_sf_ble_chardesc_dscv_rsp, uint32_t * const p_rsp_cnt);
```

3.2.5 gattCharWrite

説明：

gattCharWrite()関数は、リモートデバイスのGATT特性(GATT characteristics)を書き込みます。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル
P_char_write_req	In	特性書き込み要求へのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (* gattCharWrite)(sf_ble_ctrl_t * const p_ctrl,
                            sf_ble_conn_handle_t * p_handle,
                            sf_ble_char_write_req_t const * const
                            p_char_write_req);
```

3.2.6 gattCharRead

説明：

gattCharRead()関数は、リモートデバイスのGATT特性を読み出します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル
P_char_read_req	In	特性読み出し要求へのポインタ
P_char_read_rsp	Out	特性読み出し応答へのポインタ

戻り値：

特性読み出し応答へのポインタを返します。

関数プロトタイプ：

```
ssp_err_t (* gattCharRead)(sf_ble_ctrl_t * const p_ctrl,
                           sf_ble_conn_handle_t * p_handle,
                           sf_ble_char_read_req_t const * const
                           p_char_read_req, sf_ble_char_read_rsp_t *
                           const p_char_read_rsp);
```

3.2.7 gattCharExecuteWrite

説明：

gattCharExecuteWrite()関数は、リモートデバイスのGATT特性に対する書き込み（コミット:commit）を実行します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル
Execute_flag	In	保留中の書き込みを実行するか取り消すかを指定するフラグ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (* gattCharExecuteWrite)(sf_ble_ctrl_t * const p_ctrl,
                                   sf_ble_conn_handle_t * p_handle,
                                   sf_ble_execute_write_t execute_flag);
```

3.2.8 gattSendNotify

説明：

gattSendNotify()関数は、ローカルGATTサーバからリモートGATTクライアントにノーティフィケーション(notification)を送信します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル
Char_handle	In	通知される値を持つ特性ハンドル

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (* gattSendNotify)(sf_ble_ctrl_t * const p_ctrl,
                             sf_ble_conn_handle_t * p_handle,
                                    uint16_t char_handle);
```

3.2.9 gattSendIndicate

説明：

gattSendIndicate()関数は、ローカルGATTサーバからリモートGATTクライアントにインディケーション(indication)を送信します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル
Char_handle	In	インディケートされる値を持つ特性ハンドル

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (* gattSendIndicate)(sf_ble_ctrl_t * const p_ctrl,
                               sf_ble_conn_handle_t * p_handle,
                                    uint16_t char_handle);
```

3.2.10 gattWriteResponse

説明：

gattWriteResponse()関数は、リモートGATTクライアントからの書き込み特性値要求(characteristics value request)に応答します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル
handle	In	書き込み操作に使用される特性ハンドル
Error_code	In	応答で送信される特性書き込み操作エラーコード

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (* gattWriteResponse)(sf_ble_ctrl_t * const p_ctrl,
                                sf_ble_conn_handle_t * p_handle,
                                uint16_t handle,
                                sf_ble_attribute_error_code_t
                                error_code);
```

3.3 オンボードプロファイルAPI (On-Board Profiles APIs)

3.3.1 open

説明：

本APIはデータ転送のインタフェースを初期化します。

パラメータ名	方向	内容
p_ctrl	In, Out	BLEモジュールの制御部(control block)へのポインタ (sf_ble_ctrlを参照)
P_cfg	In	BLE設定構造体(BLE configuration structure)へのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*open)(sf_ble_onboard_profile_ctrl_t * const p_ctrl, const
                  sf_ble_onboard_profile_cfg_t *p_cfg);
```

3.3.2 close

説明：

本APIはインタフェースを終了(de-initialize)、低消費電力モードにするか電源オフにすることが可能です。ドライバを閉じ、ドライバリンクを無効にし、割り込みを禁止します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部(control block)へのポインタ (sf_ble_ctrlを参照)

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*close)(sf_ble_onboard_profile_ctrl_t * const p_ctrl);
```

3.3.3 onbpEnable

説明：

onbpEnable()関数は、サーバモードまたはクライアントモードのプロファイルを有効にします。

パラメータ名	方向	内容
p_ctrl	In, Out	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドルへのポインタ
Profile	In	有効にするプロファイルタイプ
P_prf_cb	In	プロファイルのユーザコールバック
Sec	In	プロファイルのセキュリティタイプ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*onbpEnable)(sf_ble_onboard_profile_ctrl_t * const p_ctrl,
                        sf_ble_conn_handle_t * p_handle,
                        sf_onbp_t profile, sf_ble_profile_callback_t
                        p_prf_cb, sf_ble_prf_sec_t sec);
```

3.3.4 onbpServerWriteData

説明：

onbpServerWriteData()関数は、ローカルデータベース(local database)で特性の値を更新します(update the value of the characteristics)。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドル(connection handle)へのポインタ
Profile	In	プロファイルタイプ(Profile Type)
characteristics	In	プロファイル特性(Profile characteristics)
P_data	In	データへのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*onbpServerWriteData)(sf_ble_onboard_profile_ctrl_t * const
                                 p_ctrl, sf_ble_conn_handle_t *
                                 p_handle, sf_onbp_t profile,
                                 sf_ble_onbp_char_t characteristics,
                                 const void * p_data);
```

3.3.5 onbpServerSendNotification

説明：

onbpServerSendNotification()関数は、ノーティフィケーション(notification)を送信します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドルへのポインタ
Profile	In	プロファイルタイプ
characteristics	In	プロファイル特性
P_data	In	データへのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*onbpServerSendNotification)(sf_ble_onboard_profile_ctrl_t *
                            const p_ctrl, sf_ble_conn_handle_t *
                            p_handle, sf_onbp_t profile,
                            sf_ble_onbp_char_t characteristics, const
                            void * p_data);
```

3.3.6 onbpServerSendIndication

説明：

onbpServerSendIndication()関数は、インディケーション(indication)を送信します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドルへのポインタ
Profile	In	プロファイルタイプ
characteristics	In	プロファイル特性
P_data	In	データへのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*onbpServerSendIndication)(sf_ble_onboard_profile_ctrl_t *
                            const p_ctrl,sf_ble_conn_handle_t *
                            p_handle, sf_onbp_t profile,
                            sf_ble_onbp_char_t characteristics, const
                            void * p_data);
```

3.3.7 onbpClientWriteCCCD

説明：

onbpClientWriteCCCD()関数は、リモートデバイスのクライアント設定制御ディスクリプタ(Client Configuration Control Descriptor)を設定します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドルへのポインタ(Pointer to connection handle)
Profile	In	プロファイルタイプ(Profile type)
Cccd_char	In	CCCDコード
Cccd_val	In	CCCDの設定データ(Configuration data of CCCD)

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*onbpClientWriteCCCD)(sf_ble_onboard_profile_ctrl_t * const
                                 p_ctrl, sf_ble_conn_handle_t *
                           p_handle, sf_onbp_t profile,
                           sf_ble_onbp_char_t cccd_char,
                                 sf_ble_cccd_val_t cccd_val);
```

3.3.8 onbpDisable

説明：

onbpDisable()関数は、サーバモードおよびクライアントモードのプロファイルを無効(disable)にします。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドルへのポインタ
Profile	In	無効にするプロファイルタイプ(Profile type to disable)

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*onbpDisable)(sf_ble_onboard_profile_ctrl_t * const p_ctrl,
      sf_ble_conn_handle_t * p_handle,
            sf_onbp_t profile);
```

3.3.9 onbpClientReadChar

説明：

onbpClientReadChar()関数は、プロファイルまたはサービスに対応するGATT特性(GATT characteristics)を読み出します。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドルへのポインタ
Profile	In	プロファイルタイプ
Characteristics	In	プロファイル特性

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*onbpClientReadChar)(sf_ble_onboard_profile_ctrl_t * const
                                p_ctrl, sf_ble_conn_handle_t *
                                p_handle, sf_onbp_t profile,
                                sf_ble_onbp_char_t characteristics);
```

3.3.10 onbpClientWriteChar

説明：

onbpClientWriteChar()関数は、プロファイルまたはサービスに対応するGATT特性を書き込みます。

パラメータ名	方向	内容
p_ctrl	In	BLEモジュールの制御部へのポインタ（sf_ble_ctrlを参照）
P_handle	In	接続ハンドルへのポインタ
Profile	In	プロファイルタイプ
Characteristics	In	GATT特性コード
P_data	In	データへのポインタ

戻り値：

SSPエラーステータス

関数プロトタイプ：

```
ssp_err_t (*onbpClientWriteChar)(sf_ble_onboard_profile_ctrl_t * const
        p_ctrl, sf_ble_conn_handle_t *
        p_handle, sf_onbp_t profile,
        sf_ble_onbp_char_t characteristics,
        const void * p_data);
```

4. アプリケーションへのBLEフレームワークの組み込み（Including BLE Framework in an Application）

本項は、Renesas e² studio ISDEおよびSynergyソフトウェアパッケージ（SSP）に関して、ある程度経験があるユーザを前提としています。本項の手順に進む前に、*SSP 1.4.0ユーザーズマニュアル(SSP 1.4.0 User's Manual)*の手順に従ってBlinkyプロジェクトをビルドおよび実行してください。そうすることで、e2 studio ISDEおよびSSPに慣れることができます。

*SSP 1.4.0ユーザーズマニュアル*はRenesas Synergy™ ウェブ（https://www.renesas.com/ja-jp/products/synergy/software/ssp.html）からダウンロードすることが可能です。

以下の手順は、e² studio ISDEを使用するアプリケーションにSynergy BLEフレームワークを組み込むために使用されます。

手順1：RTOSを組み込んだ新規プロジェクトの作成

1. 「File」->「New」->「Synergy C Project」をクリックして、新規Synergyプロジェクトを作成します。
2. プロジェクト名を入力し、Synergyライセンスファイル(license file)を設定します。
3. ボード（たとえば、SK-S7G2、S7G2 SKなど）を選択します。
4. 「Project Template Selection」ウィンドウで「BSP」オプションを選択します。

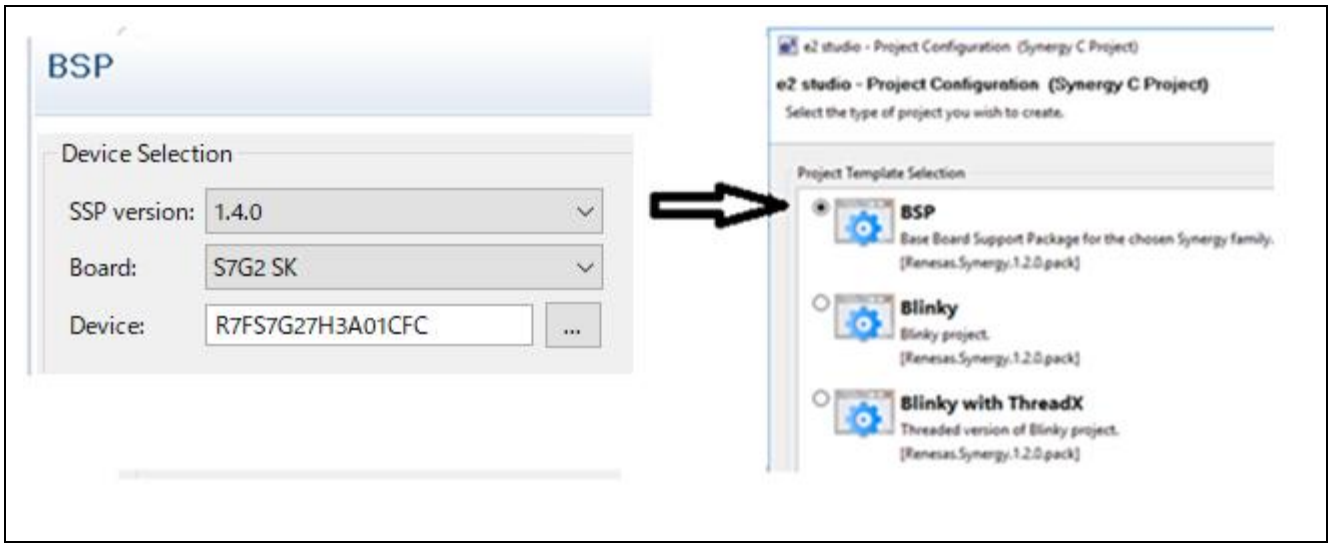


図3　Synergyプロジェクトの作成

手順2：新規スレッドの作成とBLEフレームワークの組み込み

1. BLEフレームワークはThreadXに準拠しています。アプリケーションにBLEフレームワークを組み込むには、新規スレッド(new thread)を作成し、BLEフレームワークモジュール(BLE framework module)を組み込みます。
2. 「Thread」タブを選択し、「+」記号をクリックして新規スレッドを作成します。
3. **スレッドプロパティ**については、以下の表を参照してください。

表1　スレッドプロパティ(Thread Properties)

プロパティ	デフォルト値	内容
シンボル	New_thread0	スレッドのシンボル名。この名前はスレッドファイルの作成に使用されます。シンボルがテンプレートに設定されている場合、スレッドファイルはtemplate_entry.cとして作成されます。
名前	New_thread	作成されたスレッドの名前
スタックサイズ	1024	このスレッドのスタックサイズ（バイト単位）
優先度	1	このスレッドの優先度
オートスタート	有効	有効の場合、スレッドは作成後に実行を開始します。 無効の場合、スレッドは作成後に実行されません。ユーザは必要に応じて開始する必要があります。
タイムスライス間隔（tick数）	1	スレッド実行間隔（tick単位）

4. 以下の図は、BLE**スレッド**の作成方法および**プロパティ**の更新方法に関する例を示しています。

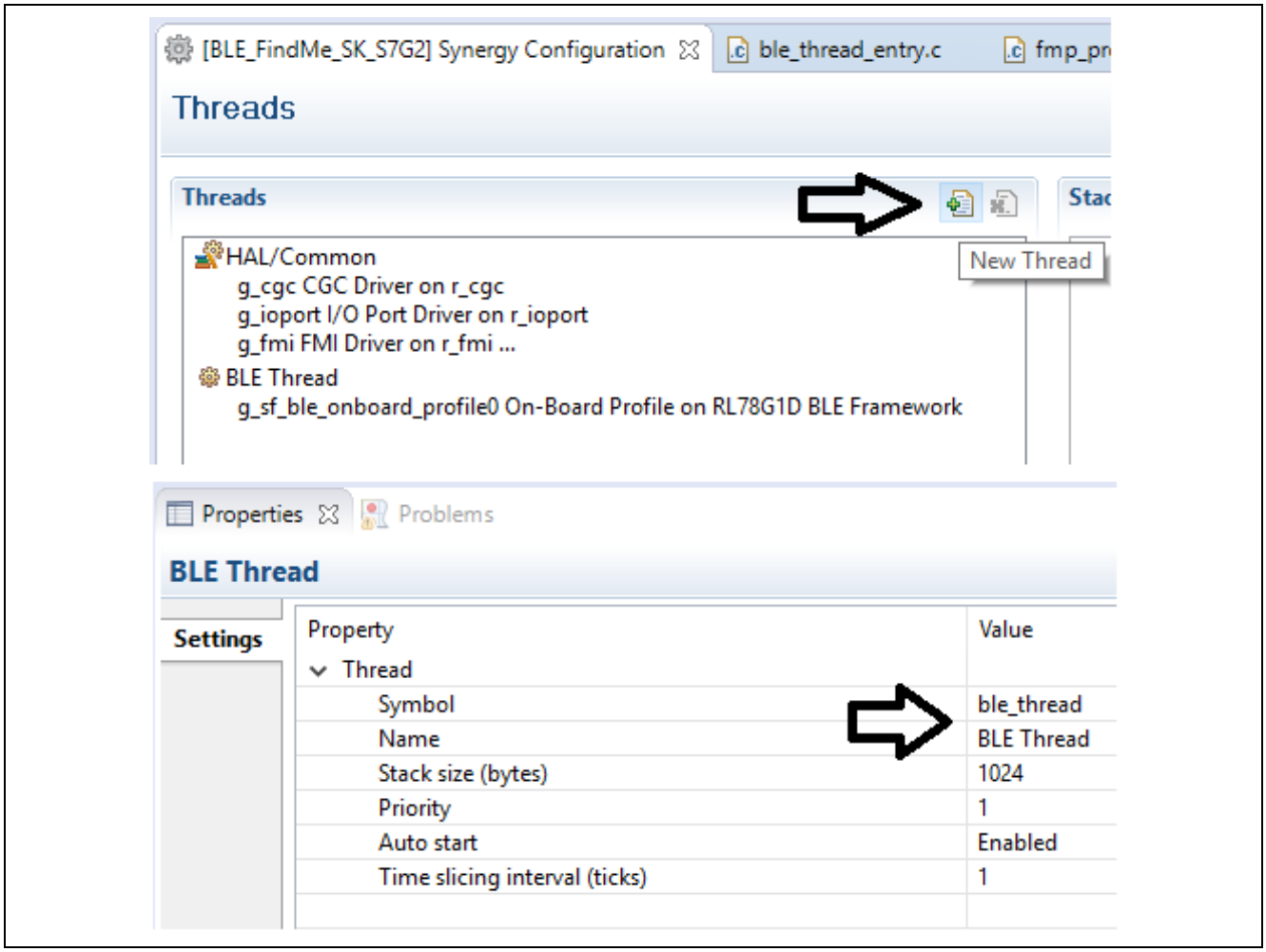


図4　BLEスレッドの作成と「Properties」タブ

手順3：BLEフレームワークの追加

1. 新規作成されたBLEスレッドをクリックします。「BLE Thread Stacks」ウィンドウで、「+」記号をクリックしてBLEフレームワークを追加します。
2. 「Framework」->「Networking」->「BLE」->「On-Board Profile on RL78G1D BLE Framework」を選択します。

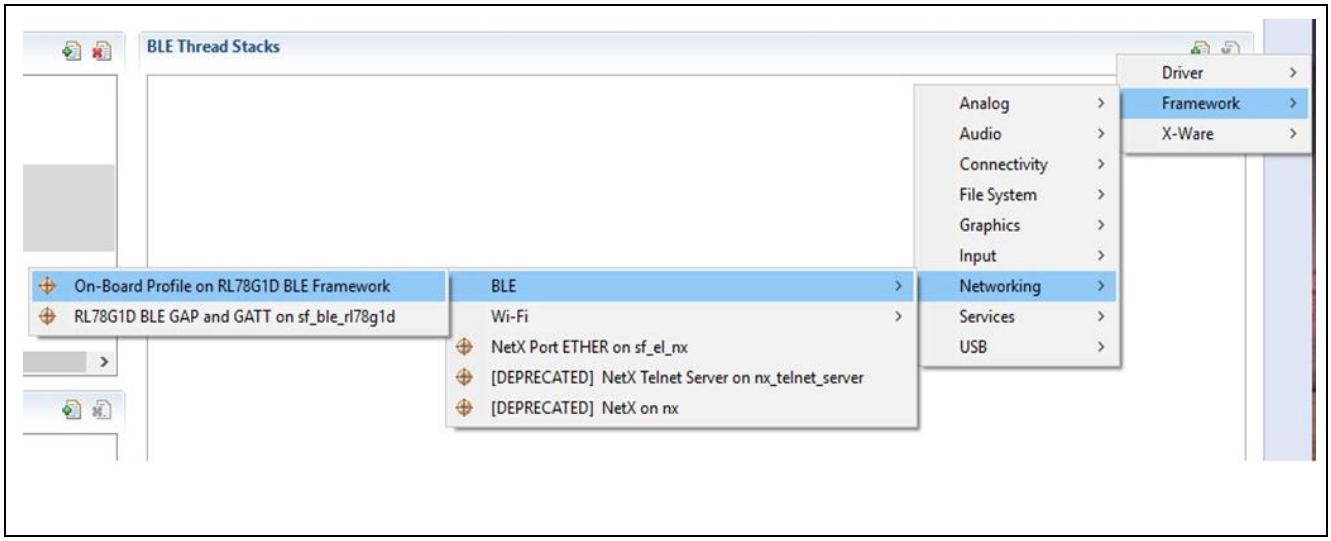


図5　BLEフレームワークの追加

3. BLEフレームワークはSSP通信フレームワークモジュールを使用して、BLEハードウェアモジュールの下層部と通信します。通信では、BLEハードウェアモジュール下層部との通信にUART/USBを使用します。

4. 「Add Communication Framework」ボックス->「New」をクリックし、「Communications Framework on sf_uart_comms」を選択します。

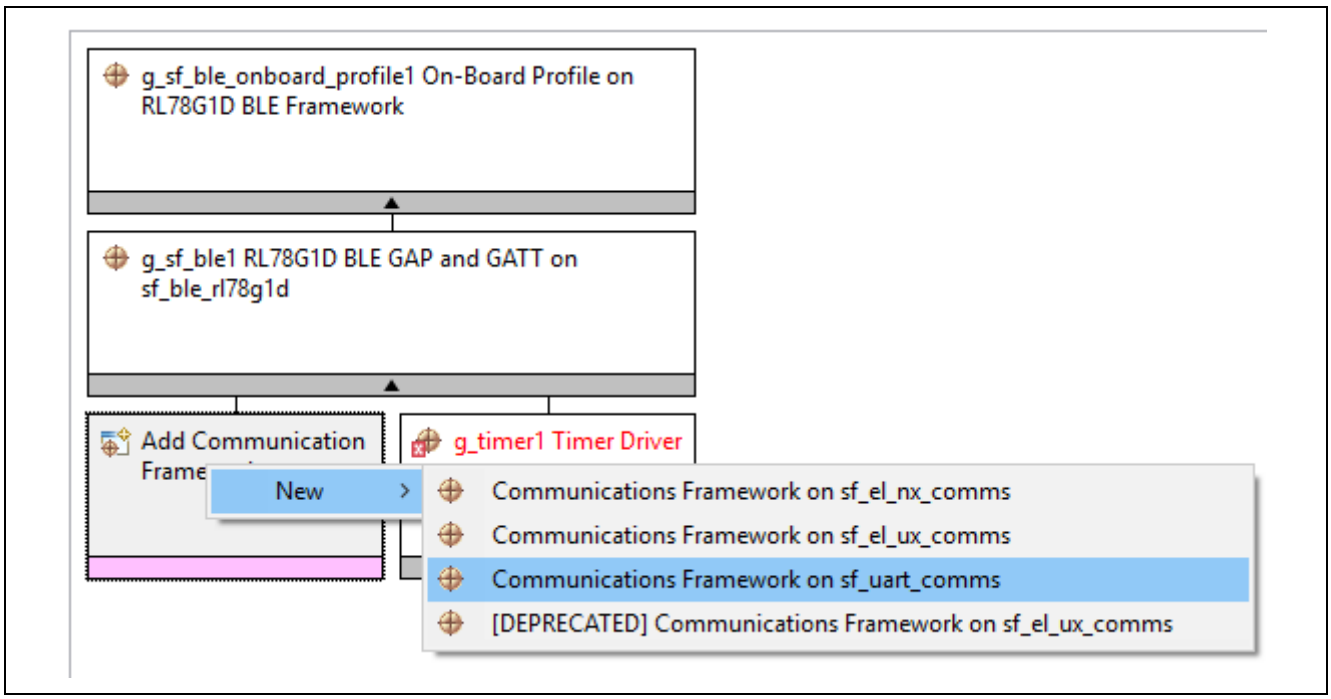


図6　通信フレームワークの追加

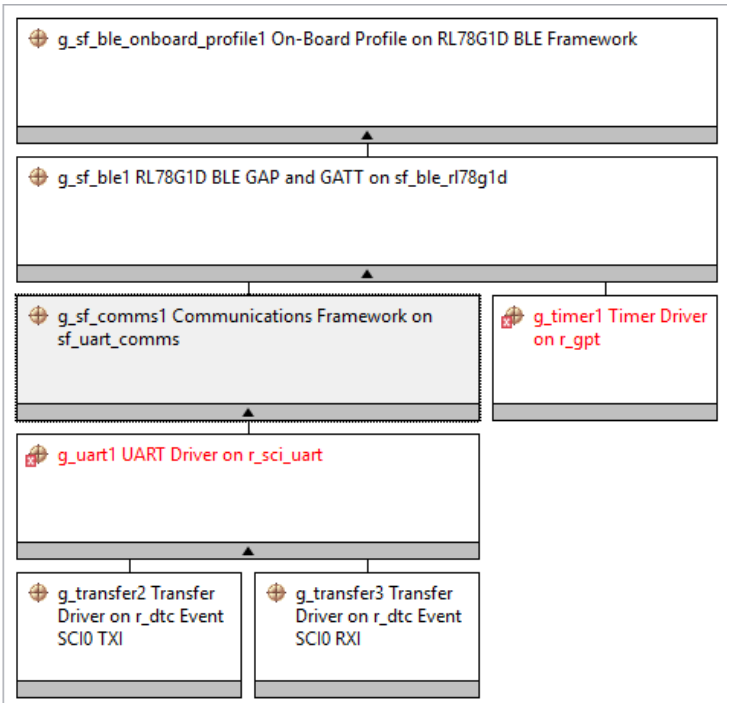


図7　追加された通信フレームワーク

5. BLEフレームワークモジュールの設定（Configurating BLE Framework Module）

本項では、BLEフレームワークモジュールに対応する設定パラメータについて詳しく説明します。

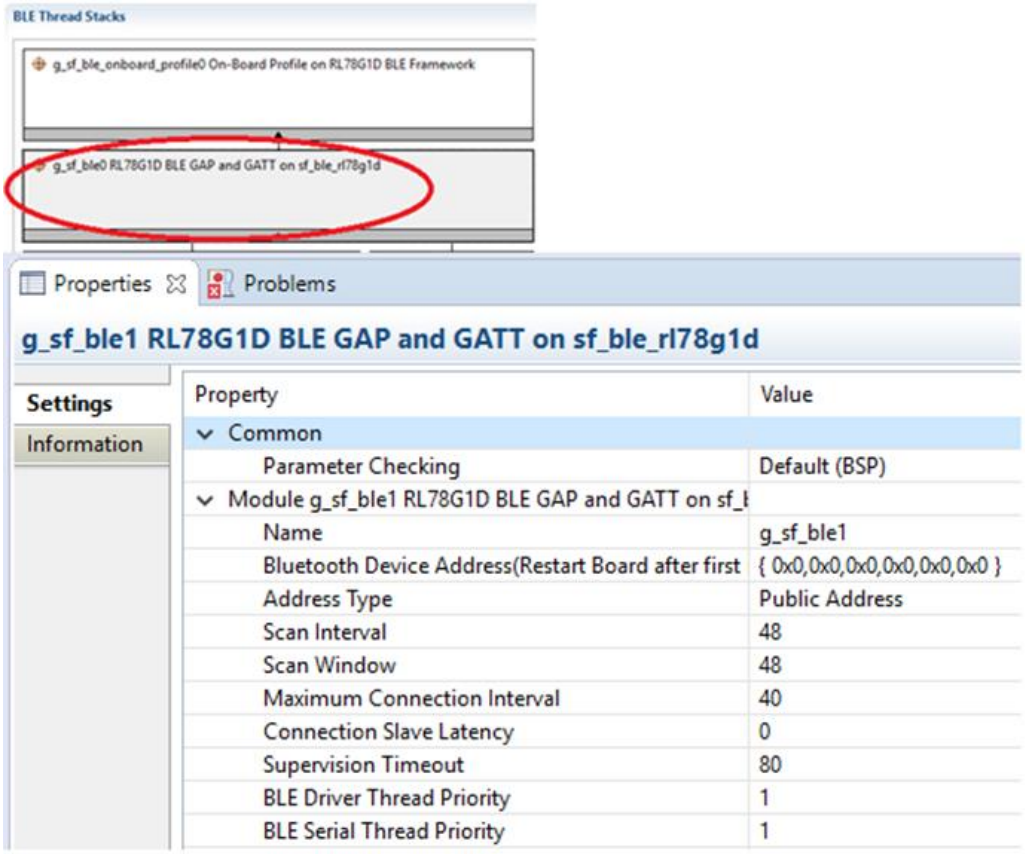


表2 BLEフレームワークモジュールの設定プロパティ

プロパティ	デフォルト値	内容
名前	g_sf_ble0	BLEフレームワークインスタンス(BLE framework instance)
Bluetoothデバイスアドレス	{0x0, 0x0, 0x0, 0x0, 0x0, 0x0}	Bluetoothデバイスアドレス(Bluetooth device address)
アドレスタイプ	パブリックアドレス	アドレスタイプはランダムアドレス(Random address)またはパブリックアドレス(Public Address)のいずれかになります。 パブリックアドレス：これは、IEEEに登録された割り当て済みの24ビットOUI（Organizationally Unique Identifier）を含むアドレスです。 ランダムアドレス：これは、乱数で構成され、以下に示す3つのカテゴリのいずれかに属するアドレスです。 静的アドレス(Static Address) 解決不能プライベートアドレス (Non-Resolvable Private Address) 解決可能プライベートアドレス (Resolvable Private Address)
スキャン間隔	48	これはアドバタイジングデータ(advertising data)を受信する間隔であり、0.625ms単位です。許容範囲は2.5～10240msです。
スキャン期間	48	これは、アドバタイジングデータがスキャン間隔で受信される期間です。0.625ms単位です。許容範囲は2.5～10240msです。
最大接続間隔	40	これは、接続確立の後にデータを定期的に送受信する間隔です。1.25ms単位です。許容範囲は7.5ms～4sです。
接続スレーブレイテンシ	0	これは、データが接続間隔で送受信される期間です。許容範囲は0～500msです。
スーパービジョンタイムアウト	80	これは、ピアデバイス(peer device)からの応答が受信されないときにリンクが失われたと見なされるタイムアウト間隔です。10ms単位です。許容範囲は100ms～32sです。
BLEドライバスレッド優先度	1	BLEドライバスレッド優先度 (BLE driver thread priority)
BLEシリアルスレッド優先度	1	BLEシリアルスレッド優先度 (BLE serial thread priority)

以下のスクリーンショットは、オンボード汎用BLEプロファイルフレームワーク(on-board generic BLE profile framework)の設定プロパティ(configuration properties)を示しています。

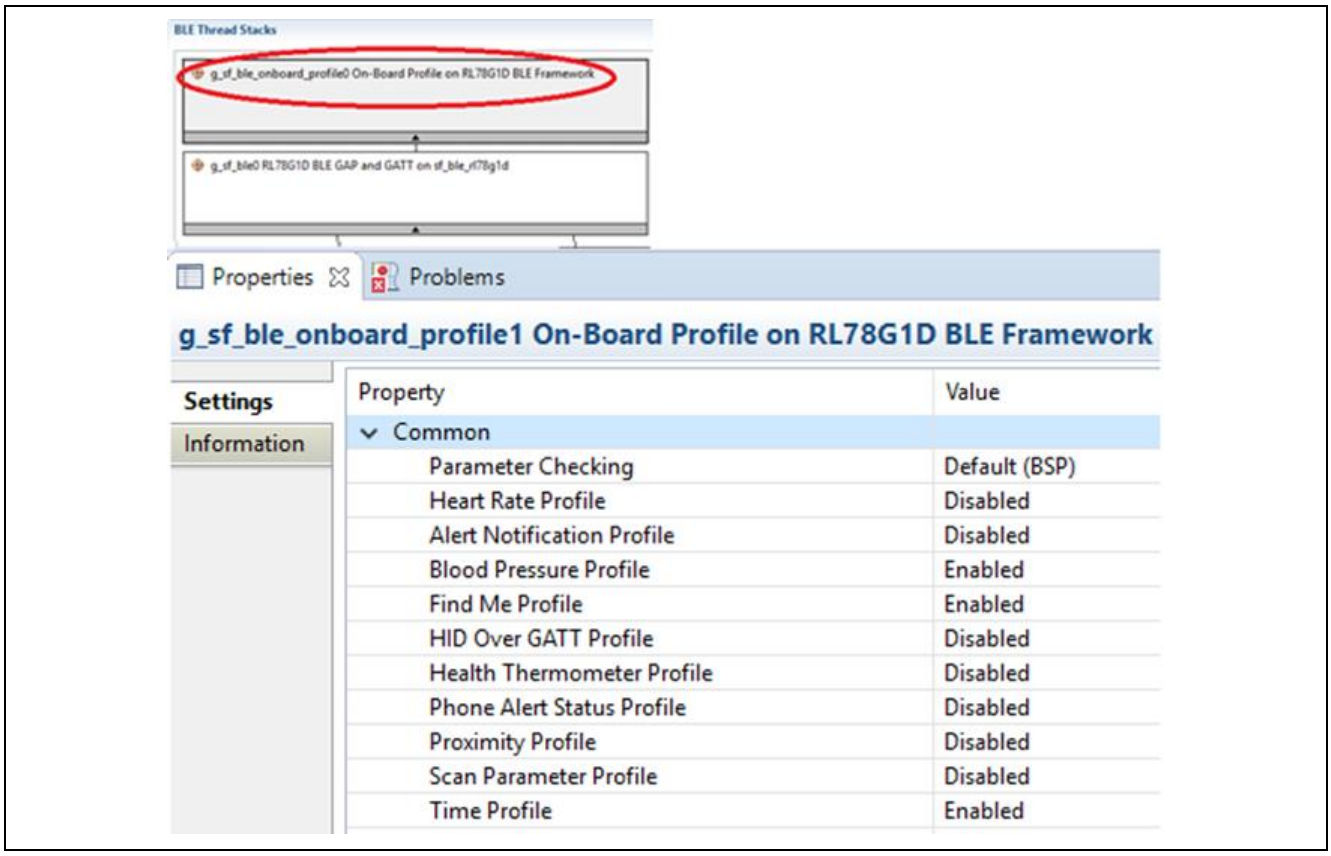


表3　オンボードプロファイルフレームワークの設定プロパティ

プロパティ	デフォルト値	内容
心拍数プロファイル(Heart Rate profile)	無効(Disabled)	BLE心拍数プロファイルは、必要に応じて有効または無効にすることが可能です。
アラートノーティフィケーションプロファイル(Alert Notification profile)	無効(Disabled)	BLEアラートノーティフィケーションプロファイルは、必要に応じて有効または無効にすることが可能です。
血圧プロファイル(Blood Pressure Profile)	無効(Disabled)	BLE血圧プロファイルは、必要に応じて有効または無効にすることが可能です。
ファインドミープロファイル(Find meProfile)	有効(Enabled)	BLEファインドミープロファイルは、必要に応じて有効または無効にすることが可能です。
HID over GATTプロファイル	無効(Disabled)	BLE HID over GATTプロファイルは、必要に応じて有効または無効にすることが可能です。
体温計プロファイル(Hearth Thermometer profile)	無効(Disabled)	BLE体温計プロファイルは、必要に応じて有効または無効にすることが可能です。
フォンアラートステータスプロファイル(Phone Alert Status profile)	無効(Disabled)	BLEフォンアラートステータスプロファイルは、必要に応じて有効または無効にすることが可能です。
近接プロファイル(Proximity profile)	無効(Disabled)	BLE近接プロファイルは、必要に応じて有効または無効にすることが可能です。
スキャンパラメータプロファイル(Scan Parameter profile)	無効(Disabled)	BLEスキャンパラメータプロファイルは、必要に応じて有効または無効にすることが可能です。
時刻プロファイル(Time profile)	無効(Disabled)	BLE時刻プロファイルは、必要に応じて有効または無効にすることが可能です。

6. BLEフレームワークモジュールアプリケーション例 (BLE Framework Module Aplication Example)

6.1　概要 (Overview)

本アプリケーションプロジェクト例は、Synergy BLEフレームワークのファインドミープロファイル動作(Find Me Profile)を提示します。ファインドミーターゲットは即時アラートサービス(Immediate Alert Service)の1つのインスタンス(instance)でファインドミープロファイルを利用して、クライアントがデバイスの設定を変更した場合にアラートを表示します。ファインドミーターゲットは、ファインドミーロケータプロファイル(Find Me Locator Profile)を実装する他のデバイスと連動します。

ファインドミープロファイルは以下の2つの役割 (role) を定義します。

- ファインドミーターゲットはGATTサーバです。
- ファインドミーロケータはGATTクライアントです。

以下の図は、サービスとプロファイルロールの関係を示しています。

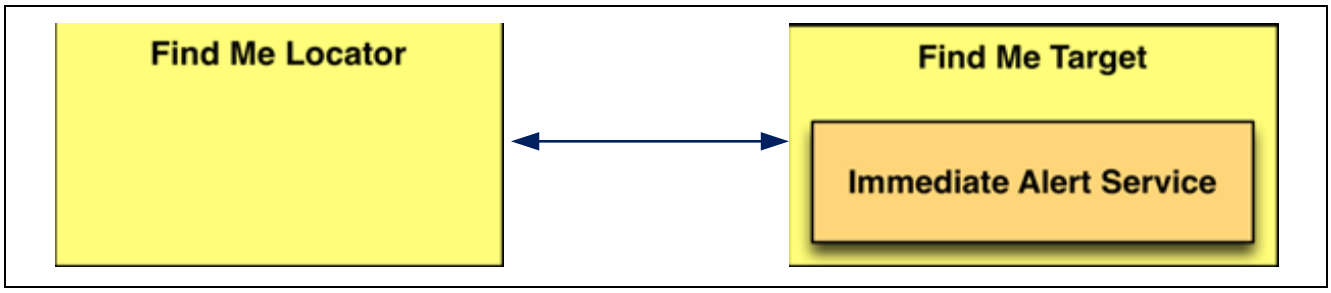


図8　ロールとサービスの関係

ファインドミーターゲット(Find Me Target)には即時アラートサービスのインスタンスが含まれています。本BLEアプリケーション例では、SK-S7G2キットはファインドミーターゲットとして機能し、Android携帯で動作するBLE Scanner APK、iPhoneで動作するLightBlue APKなどのアプリケーションはファインドミーロケータとして機能します。

6.2　BLEアプリケーションソフトウェアアーキテクチャの概要（BLE application software architecture overview）

本項では、BLEフレームワークアプリケーション例のソフトウェアアーキテクチャの概要を説明します。

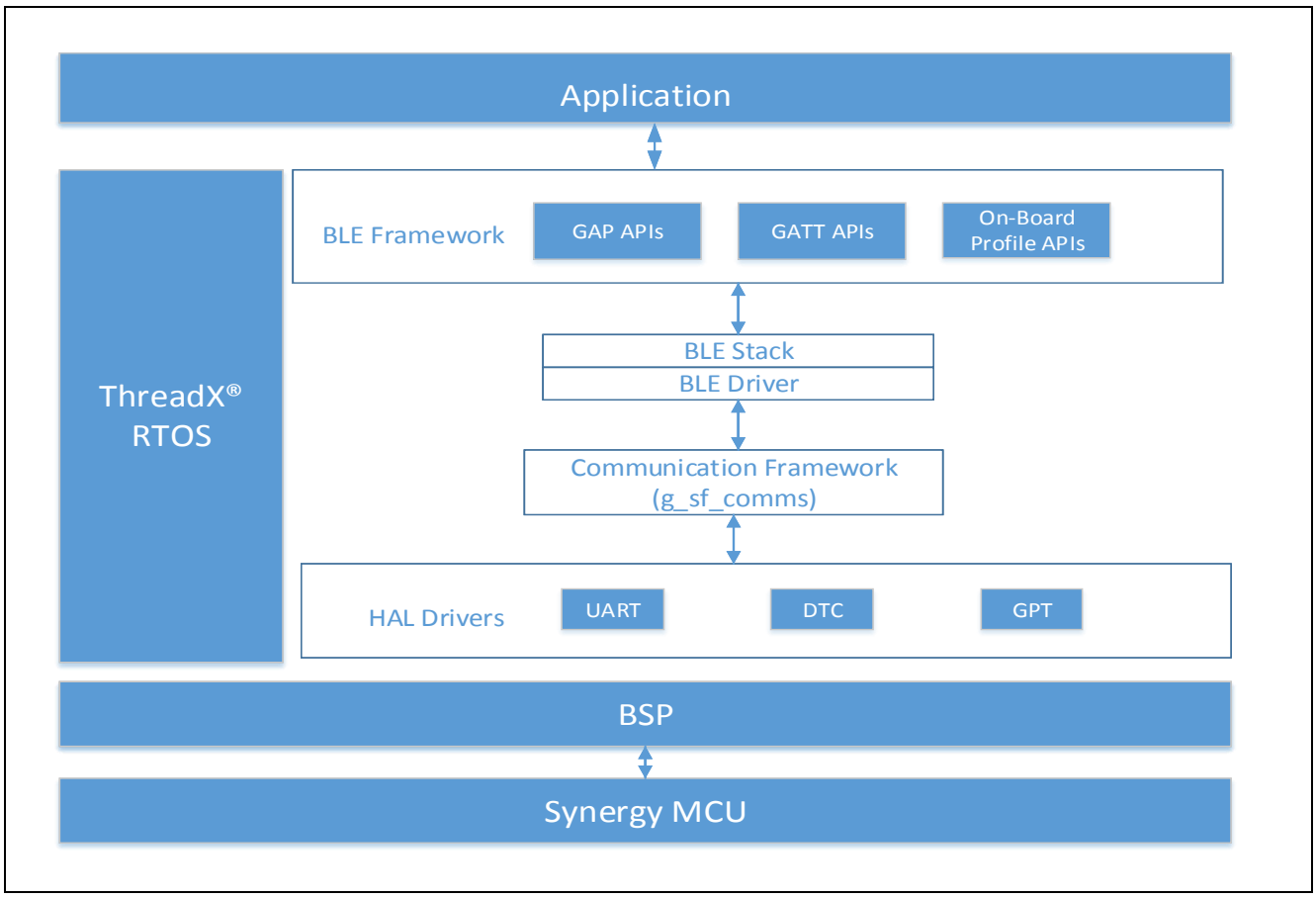


図9　BLEファインドミーターゲットアプリケーションのソフトウェアアーキテクチャ

BLEファインドミーターゲットアプリケーションの主要なソフトウェアコンポーネントは以下のとおりです。

- BLEスレッド(BLE thread)
- BLEフレームワーク(BLE framework)
- 通信フレームワーク(Communication framework)

本BLEアプリケーション例はSynergy BLEフレームワークのコア機能を提示します。この機能はRL78G1D BLEモジュールのファインドミープロファイル(Finf Me Profile)を使用して設定されています。RL78G1D BLEモジュールはファインドミーターゲットとして機能します。本プロジェクトには、ペリフェラル専用モードとして設定されたGAPロール(GAP role)があります。

BLEスレッドには、RL78G1DモジュールのSynergy BLEフレームワークおよびそれに対応するBLEスタックとBLEデバイスドライバに加えて、2つのBLEデバイス間のデータ転送をサポートするSSPモジュール（UART、DTC、GPTなど）が含まれています。BLEスレッドは、BLEフレームワーク（初期化、プロビジョニング、スキャニング、アドバタイジング、2つのBLEデバイス間のデータ転送など）を使用してすべてのBLE通信を処理します。

Callbacks：

以下の2つのユーザコールバック(user callbacks)は、BLEフレームワークに登録されています。

1. User_ble_callback
2. Fmpt_callback

User_ble_callback：

初期化時に、BLEスレッドはBLEフレームワークにコールバックを登録し、イベント（接続、切断、ボンディング、BLEフレームワークからのGATTノーティフィケーション／インディケーションなど）に関する通知を受信します。

Fmpt_callback：

ファインドミープロファイルターゲット(Find Me Profile Target)の有効時に、BLEスレッドはfmpt_callbackを登録して、ファインドミープロファイル固有のイベント（アラートレベルの変更など）に関する通知(notification)をBLEフレームワークから受信します。

BLEスレッドは以下の状態のステートマシンで動作します。ある任意の時点で、BLEスレッドは以下のいずれかの状態になります。

1. 初期状態(Init State)
2. 接続状態(Connect state)
3. 起動プロファイル(Activate state)
4. 処理プロファイルイベント(Handle profile events)
5. 切断状態(Disconnect state)

初期状態(Init state)：

アプリケーション設計に基づいて、sf_ble_provisioning_t構造体およびsf_ble_adv_info_t構造体を手動で作成する必要があります。

初期化時に、BLEスレッドは上記の構造体をBLEフレームワークに渡すことで、アドバタイジング(advertisement)をプロビジョニング(provisioning)して開始します。user_ble_callbackコールバック関数は、BLEフレームワークからノーティフィケーションを受信するために登録されます。

初期化後に、BLEスレッドは接続状態に設定されます。

接続状態(Connect State)：

デバイスはすでにアドバタイジングを開始しており、BLEクライアント(BLE client)が接続を開始するのを待機しています。この状態では、BLEスレッドはBLEフレームワークからの接続イベントを待機しています。

接続イベントが受信されると、BLEスレッドは起動プロファイル状態に設定されます。

起動プロファイル(Activate State)：

デバイスはすでにBLEクライアントデバイスに接続されています。BLEスレッドはファインドミープロファイル(Find Me Profile)を有効にし、ファインドミープロファイル固有のノーティフィケーションを受信するためにfmpt_callbackルーチンを登録します。

プロファイルが起動されると、BLEスレッドは処理プロファイルイベント状態に設定されます。

処理プロファイルイベント(Handle profile events)：

BLEスレッドはファインドミープロファイルイベント(Find Me Profile events)を処理します。BLEスレッドは、BLEフレームワークから切断イベントを受信するまでこの状態を維持します。その後、切断状態(disconnect state)に遷移します。

ファインドミープロファイルは、デバイスでボタンが押されてピアデバイス(peer device)で即時アラートが発生するときの動作を定義します。これは、置き忘れたデバイス(devices that are misplaced)を探すために利用することが可能です。

LED2はアラートレベルの実証に使用されます。アラートレベルがクライアントからMILD_ALERTに設定された場合、LED2は点滅を開始します。アラートレベルがクライアントからHIGH_ALERTに設定された場合、LED2はオンになります。これらのアラートをクリアするには、クライアントから要求を送信してアラートレベル特性をNO_ALERTに設定します。クライアントとの接続が解除または切断された場合も、これらのアラートはクリアされます。

切断状態(disconnect state):

この状態では、BLEデバイスはBLEクライアントから切断され、BLEフレームワークから切断イベントを受信しています。

BLEスレッドはファインドミープロファイル(Find Me Profile)を無効にし、ユーザLEDをオフにして状態を初期状態に設定します。

6.3 設定 (Configuration)

以下の手順は、e² studio ISDEを使用する本アプリケーション例でSynergy BLEフレームワークモジュール(BLE Framework module)を設定するために使用されます。

本項では、SK-S7G2 Synergy MCUグループボードを標準キットとして利用しており、ハードウェアプラットフォームに関連する設定はSK-S7G2 Synergy MCUグループボード向けに設定されています。

1. 図に示すように、g_sf_ble0のプロパティを設定します。希望のBluetoothアドレスを設定することが可能です。変更したアドレスを確認するには、最初の実行後にボードを再起動します。

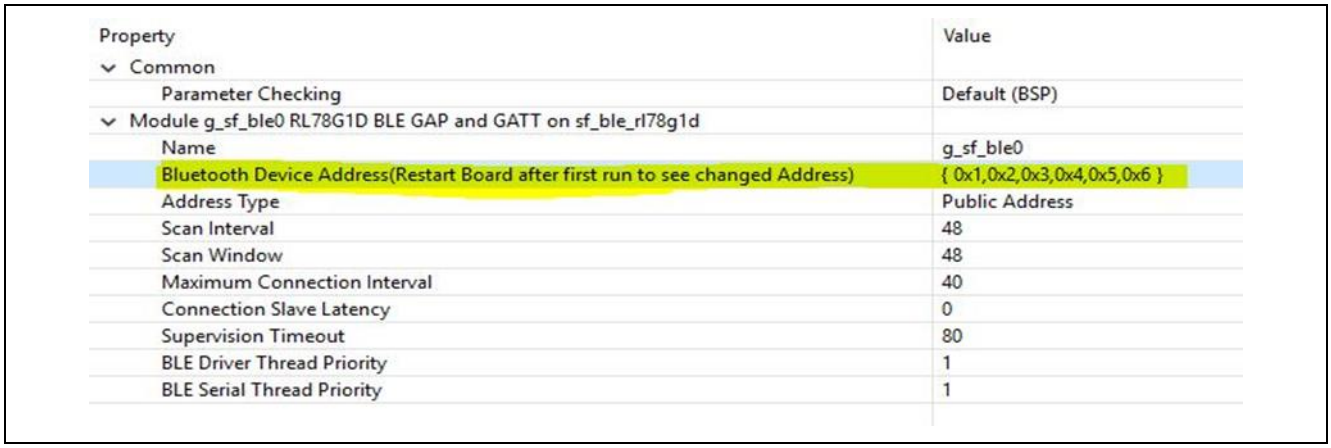


図10 g_sf_ble0のプロパティ設定

2. 本プロジェクトでは、ファインドミープロファイル(Find Me Profile)を使用してSynergy BLEフレームワーク機能を提示します。以下の図に示すように、g_sf_ble_onboard_profile0プロパティウィンドウから「Find Me Profile」を有効にします。

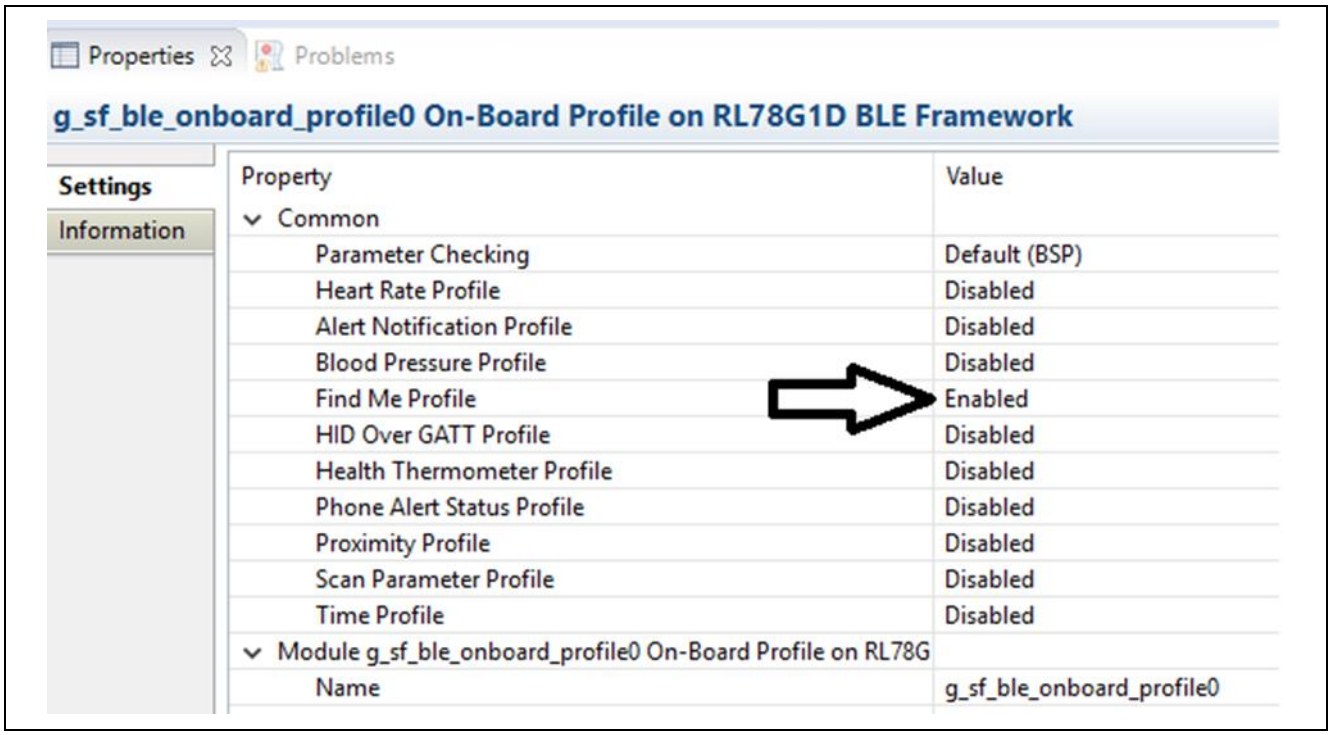
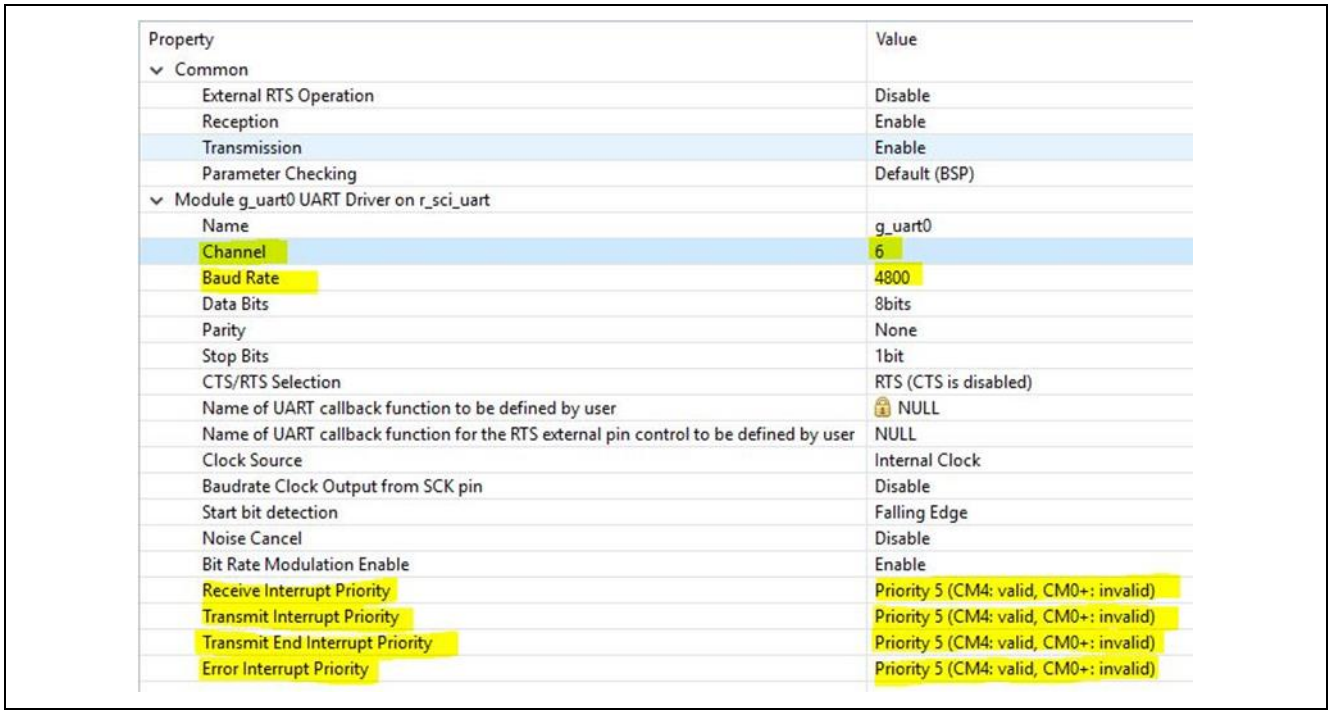


図11　g_sf_ble_onboard_profileのプロパティ設定

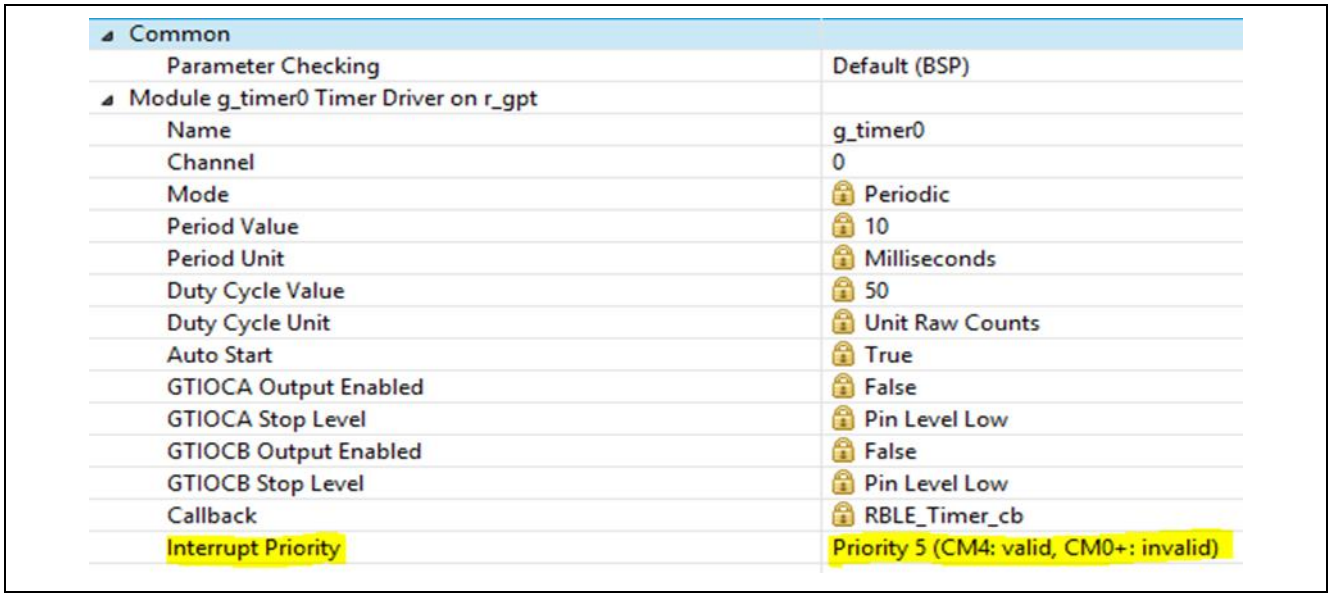
3. 以下の図に示すように、r_sci_uartに対して「Property」ウィンドウで「Channel」を6に設定し、「Baud Rate」を4800に設定します。UARTドライバのプロパティの詳細については、*UARTモジュールガイド(UART Module Guide)*を参照してください。*UARTモジュールガイド*をダウンロードするには、このリンクを使用します。



Property	Value
Common	
External RTS Operation	Disable
Reception	Enable
Transmission	Enable
Parameter Checking	Default (BSP)
Module g_uart0 UART Driver on r_sci_uart	
Name	g_uart0
Channel	6
Baud Rate	4800
Data Bits	8bits
Parity	None
Stop Bits	1bit
CTS/RTS Selection	RTS (CTS is disabled)
Name of UART callback function to be defined by user	NULL
Name of UART callback function for the RTS external pin control to be defined by user	NULL
Clock Source	Internal Clock
Baudrate Clock Output from SCK pin	Disable
Start bit detection	Falling Edge
Noise Cancel	Disable
Bit Rate Modulation Enable	Enable
Receive Interrupt Priority	Priority 5 (CM4: valid, CM0+: invalid)
Transmit Interrupt Priority	Priority 5 (CM4: valid, CM0+: invalid)
Transmit End Interrupt Priority	Priority 5 (CM4: valid, CM0+: invalid)
Error Interrupt Priority	Priority 5 (CM4: valid, CM0+: invalid)

図12　g_uart0のプロパティ設定

4. 以下の図に示すように、g_timer0のプロパティを設定します。GPTドライバのプロパティの詳細については、*GPTモジュールガイド(GPT Module Guide)*を参照してください。*GPTモジュールガイド*をダウンロードするには、このリンクを使用します。



Common	
Parameter Checking	Default (BSP)
Module g_timer0 Timer Driver on r_gpt	
Name	g_timer0
Channel	0
Mode	Periodic
Period Value	10
Period Unit	Milliseconds
Duty Cycle Value	50
Duty Cycle Unit	Unit Raw Counts
Auto Start	True
GTIOCA Output Enabled	False
GTIOCA Stop Level	Pin Level Low
GTIOCB Output Enabled	False
GTIOCB Stop Level	Pin Level Low
Callback	RBLE_Timer_cb
Interrupt Priority	Priority 5 (CM4: valid, CM0+: invalid)

図13　g_timerのプロパティ設定

端子設定(Pin Configuratio)：

「Pins」タブに移動し、SK-S7G2ボードに対して以下の端子設定を行います。

SCI端子(SCI pins)

1. SK-S7G2ボードの場合、SCI6が使用されます。
2. 「Pins」タブから、「Pin Selection」セクションに移動します。

3. 「Peripherals」->「Connectivity: SCI」->「SCI6」に移動します。
4. 「Operation Mode」を「Asynchronous UART」に設定します。以下の図に示すように、SCI用に「P304」と「P305」を選択します。

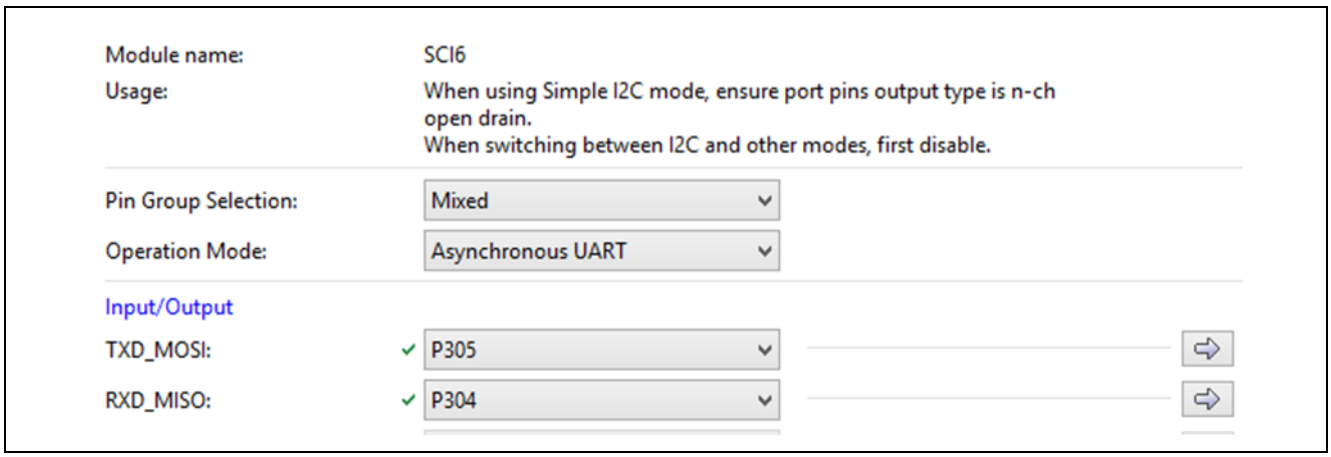


図14　SK-S7G2ボードに対するSCI6端子設定

SK-S7G2に対してリセット端子を設定します。

1. 「Pins」タブから、「Pin Selection」セクションに移動します。
2. 「Ports」->「P3」->「P309」に移動します。

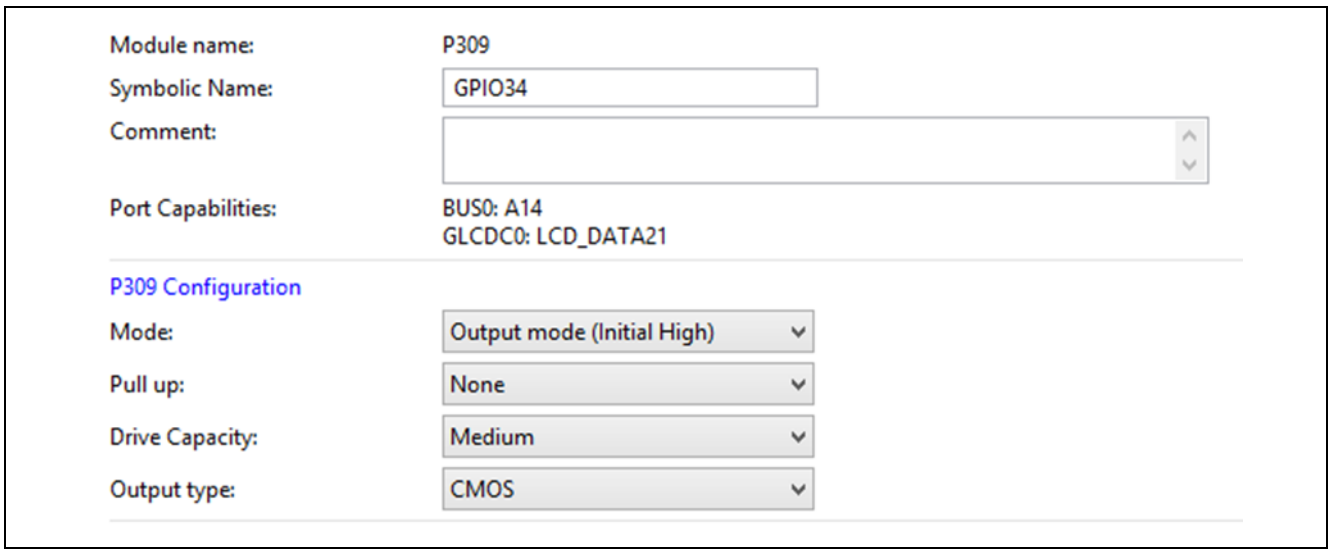


図15　SK-S7G2ボードに対するリセット端子設定

アプリケーションプロジェクトの設定が終了したら、「Generate Project Content」をクリックしてプロジェクトの内容を生成します。これにより、選択した設定オプションでプロジェクトファイル(project file)が生成されます。

e² studio ISDEによりアプリケーションプロジェクトのファイルが選択した設定で生成されたら、プロジェクトのプロジェクトエクスプローラーウィンドウ(project explorer window)に移動し、srcフォルダを開いて、本アプリケーションプロジェクト用に生成された関連ファイルを確認します。

これらのファイルは、ユーザアプリケーションコードを追加するプレースホルダ(place holder)です。独自のアプリケーション機能を記述することが可能です。あるいは、BLE_FindMe_SK_S7G2デモアプリケーションプロジェクトから既存のソースファイルをコピーして、このデモを再作成することも可能です。

メニューバーからハンマーアイコンをクリックして、アプリケーションプロジェクトをビルドします。

7. BLEフレームワークモジュールアプリケーション例の実行 (Running the BLE Framework Module Application Example)

7.1 ボードの電源投入 (Powering up the board)

本項では、電源とボード、J-LinkデバッガとPC、ボードとPC USBポートをそれぞれ接続する方法、およびデバッグアプリケーションを実行する方法について説明します。

ボードに接続するには：

1. 付属のUSBケーブルのマイクロUSB端子をSK-S7G2ボードJ19コネクタ (DEBUG_USB) に接続します。

注： キットにはSEGGER J-Link® On-board (OB) が含まれています。J-Linkは、SK-S7G2ボード用の全デバッグ機能およびプログラミング機能を備えています。

2. USBケーブルのもう一方の端子をPCのUSBポートに接続します。

7.2 RL78G1Dファームウェアの準備 (RL78G1D firmware programming)

BLEアプリケーションデモを実行する前に、ユーザアプリケーションに基づいてRL78G1D BLEモジュールを準備する必要があります。アプリケーションに応じて以下の.hexファイルのいずれかをUSBフラッシュドライブに手動でコピーする必要があります。

以下の手順は、オンボードRL78G1Dのファームウェアを SK-S7G2ボードに書き込む方法を示しています。

1. BLE_FindMe_SK_S7G2プロジェクトのコンパイルが成功した後：

   a) 「debug」->「debug configurations」->「Renesas GDB Hardware Debugging」->プロジェクトデバッグ「**BLE_FindMe_SK_S7G2 Debug**」に移動します。

   b) 以下の図に示すように、「Browse」ボタンをクリックし、PCに格納されているProgrammer.hexファイルを選択します。本ファイルは、BLEフレームワークモジュールアプリケーションサンプルパッケージ(BLE Framework Module Application Example package)の一部として提供されます。

2. 「debugger」に移動し、対象デバイスをR7FS7G2として設定します。

   a) 「Synergy」->「Synergy/CM4」->「R7FS7G2」

3. 「debug」ボタンをクリックします。

4. 画像がフラッシュ(flashed)されたら、プロジェクトを終了します。

5. デバイスを再起動します。ボードの表示が「**Looking for a USB device**」になったら、ファームウェアファイルRL78_G1D_IM (FMP).hexがロードされたUSBを接続します。表示の指示に従って、GATTのhexファイルまたは特定のオンボードプロファイルを書き込み(flash)ます。

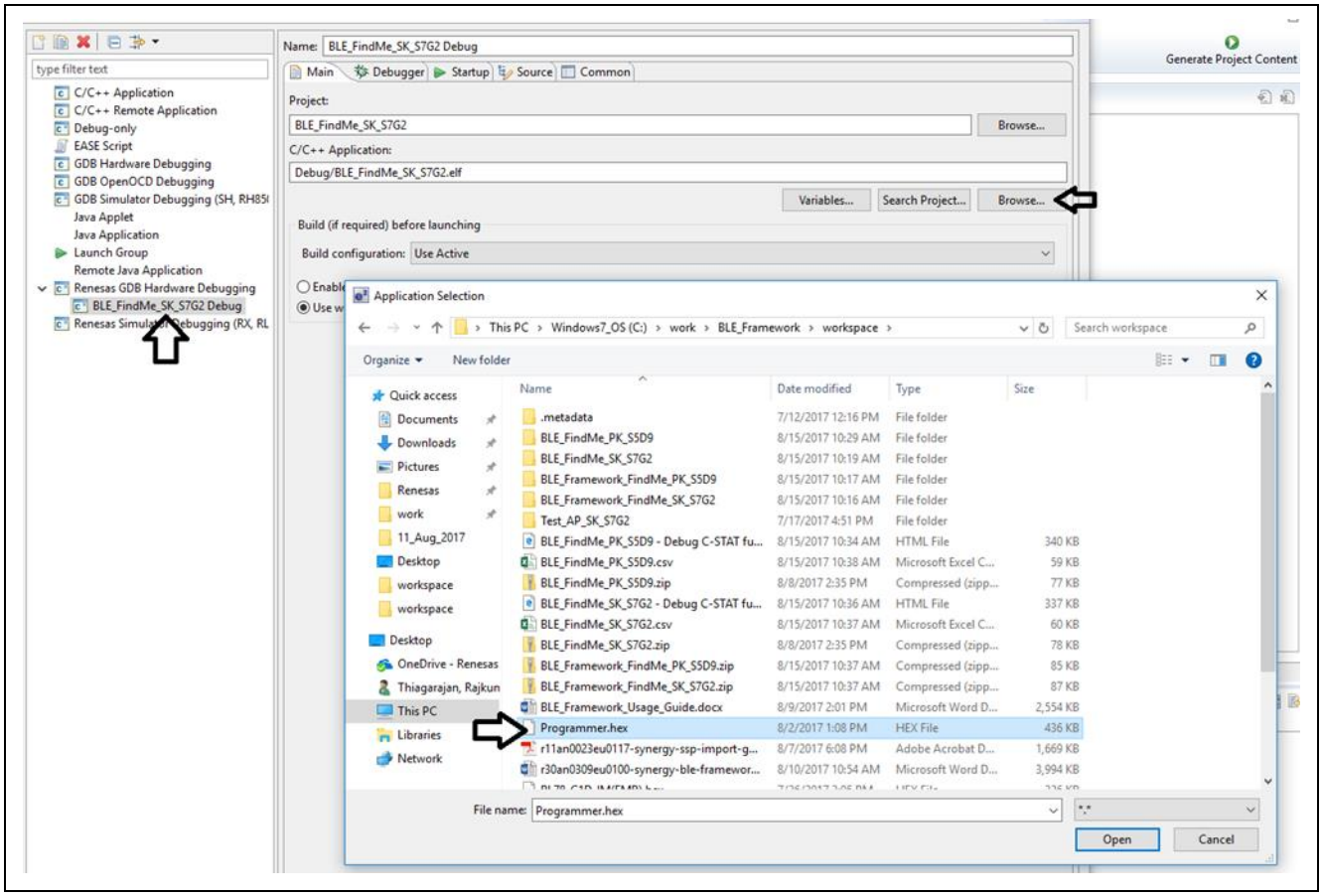


図16 RL78G1D BLEハードウェアモジュールのプログラミング

GATTおよびオンボードプロファイルの場合、別のファイルが書き込まれ(flashed)ます。

1. GATTの場合、**RL78_G1D_IM (SCP).hex**ファイルをフラッシュします。RL78_G1D_IM (SCP).hexファイルがロードされたUSBデバイスを接続します。

2. オンボードプロファイルの場合、以下のhexファイルが表示されます。一度に1つのhexファイルしか書き込む(flashed)ことができません。これらのファイルは、BLEフレームワークモジュールアプリケーションサンプルの一部として提供されます。

RL78_G1D_IM (GLP, PASP, TIP).hex
本ファイルは以下のプロファイルを兼ね備えています。
- グルコースプロファイル(Gluecose profile)
- フォンアラートステータスプロファイル(Phone Alert Status profile)

RL78_G1D_IM (HOGP, ScPP).hex
本ファイルは以下のプロファイルを兼ね備えています。
- HID over GATTプロファイル(HID over GATT profile)
- スキャンパラメータプロファイル(Scan parameter profile)

RL78_G1D_IM (HTP, BLP, HRP).hex
本ファイルは以下のプロファイルを兼ね備えています。
- 体温計プロファイル(Health Thermometer profile)
- 血圧プロファイル(Blood Pressure profile)
- 心拍数プロファイル(Heart Beat profile)

RL78_G1D_IM (PXP, FMP, ANP).hex
本ファイルは以下のプロファイルを兼ね備えています。
- 近接プロファイル(Proximity profile)
- ファインドミープロファイル(Fine Me profile)
- アラートノーティフィケーションプロファイル(Alert Notification Profile)

3. コードが書き込まれたら(flashed)、USBデバイスを抜いてボードを再起動します。

注： これらの.hexファイルは、複数のプロファイルおよびそれらに対応する必須サービスをサポートしています。これらの.hexファイルのいずれかをプログラミングし、サポートされるプロファイルのいずれかを有効にすると、.hexファイルに含まれる他のプロファイルに対応するサービスもすべてデフォルトで有効になります。

7.3 プロジェクトのインポート、ビルド、および実行 (Importing, building, and running the Project)

プロジェクトをe2 studioにインポートしてビルド／実行する手順については、*SSPインポートガイド* (英語版：r11an0023eu0119-synergy-ssp-import-guide.pdf, 日本語版：r11an0023ju0116-synergy-ssp-import-guide.pdf (参考資料)) を参照してください。

注： デバッグには、「GDB Hardware Debugging」設定の**「BLE_FindMe_SK_S7G2 Debug」**を選択する必要があります。
注： SSPインポートガイドは更新されている場合があります。その場合は Renesas Synergy WEBから最新のものを入手してください。

7.4　デモの検証（Verifying the demonstration）

5.2.1～5.2.3の各項を参照し、手順に従って、SK-S7G2ボードの電源を投入し、RL78G1Dファームウェアの書込みを行った後、既存のBLEフレームワークアプリケーション例プロジェクトを実行します。

クライアントデバイスは、ファインドミーロケータアプリケーション(Find Me Locator applicatgion)を実行する別のボードになる場合があります。あるいは、標準BLEアプリケーション（AndroidデバイスまたはIOSデバイスで動作するBLE Scannerなど）になる場合もあります。本書では、Androidデバイスで動作するBLE Scanner APKをBLEクライアントデバイスとして使用しています。

BLEアプリケーションがSK-S7G2ボードで動作している場合、Android携帯でBLE Scannerアプリケーションを開き、デバイスをスキャンします。以下の図に示すように、SK-S7G2ボードはSynergy BLEデバイスとして表示されます。

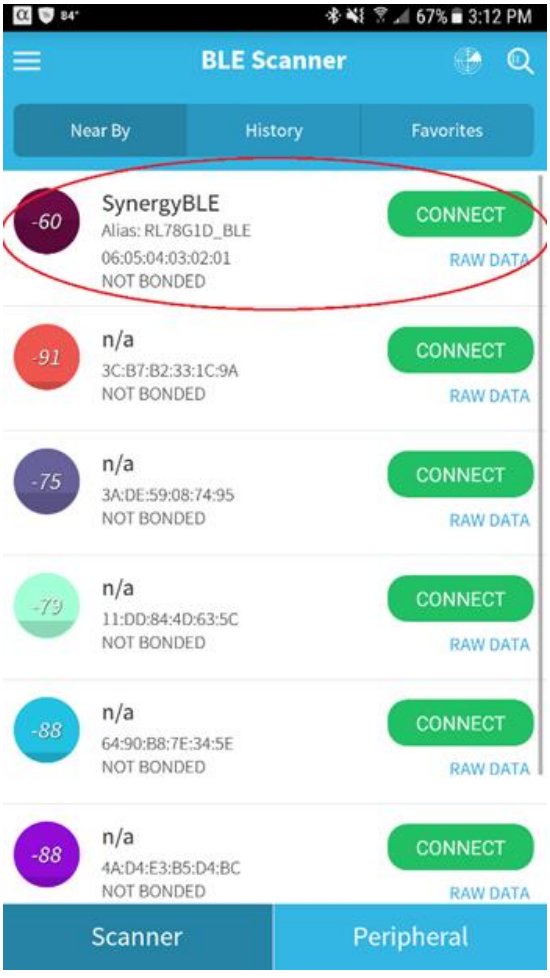


図17　BLEスキャンウィンドウ

Synergy BLEデバイスがウィンドウに表示されたら、「CONNECT」ボタンをクリックしてデバイスに接続します。接続に成功すると、以下の図に示すように、BLE Scanner APKは新しいウィンドウを開き、このプロファイルでサポートされているサービスのリストを表示します。

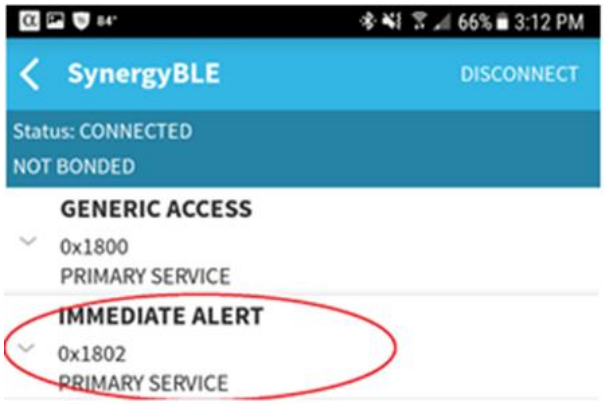


図18　BLEファインドミープロファイルサービス

IMMEDIATE_ALERTサービスを展開するには、サービスの横にある下矢印をクリックします。そのサービスに対応するプロパティ(properties)が表示されます。以下の図に示すように、「W」ボタンをクリックしてアラートレベルをBLEサーバ（SK-S7G2ボード）に送信します。

図19　BLEファインドミープロファイルアラートレベルのトリガ

以下の図に示すように、ドロップダウンメニューにアラートレベルが表示されます。

図20　BLEファインドミープロファイルアラートレベル

アラートレベルを選択し、「OK」ボタンを押します。BLEクライアントは、SK-S7G2ボードで動作しているBLEサーバアプリケーションにアラートレベルを送信します。

表4に示すように、LED2はアラートレベルに応じて点滅します。

表4　BLEファインドミープロファイルアラートレベル

アラートレベル(Alert Level)	LED2ステータス
アラートなし(No Alert)	オフ
マイルドアラート(Mild Alert)	継続的な点滅
ハイアラート(High Alert)	オン

8.　BLEフレームワークの次の手順（Next Steps）

1. https://www.renesas.com/ja-jp/products/synergy.htmlにアクセスして、開発ツールおよびユーティリティの詳細を確認してから、ダウンロードします。

2. 以下の内容について詳細を確認します。
   - Synergyキット：http://www.renesassynergy.com/kits
   - Synergyマイクロコントローラ：http://www.renesassynergy.com/microcontrollers
   - Synergyソフトウェア：http://www.renesassynergy.com/software
   - Synergyソリューション：http://www.renesassynergy.com/solution

3. RL78G1D BLEモジュールを入手します。

   RL78G1D BLEモジュールはSK-S7G2/PK-S5D9キットに実装されています。

4. Renesas Synergyモジュールガイド付随リンク

   https://www.renesas.com/ja-jp/products/synergy/tools-kits.html#sampleCodes

9.　BLEフレームワークの参考資料（References）

1. *SSP 1.4.0ユーザーズマニュアル*はRenesas Synergy™ WEBの”SSPについて学ぶ”（https://www.renesas.com/ja-jp/products/synergy/software/ssp.html）からダウンロードすることが可能です。

2. *BLE Find My Profile Specification*

ホームページとサポート窓口

サポート： https://synergygallery.renesas.com/support

テクニカルサポート：

- アメリカ： https://www.renesas.com/en-us/support/contact.html
- ヨーロッパ： https://www.renesas.com/en-eu/support/contact.html
- 日本： https://www.renesas.com/ja-jp/support/contact.html

改訂履歴

		改訂内容	
Rev.	発行日	ページ	ポイント
1.04	2018.7.25	-	第1.04版 発行 英文版（資料番号r30an0309eu0104-synergy-ble-framework、Rev1.04、発効日2018年3月22日）を翻訳 Renesas Synergy ウェブを日本語版に変更 アプリケーションノート参考例リンク先に日本語版を追加

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.3.0-1 November 2016)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0