RX23W グループ
高速通信用サンプルプログラム

要旨

本アプリケーションノートでは、Bluetooth® Low Energy を使用して高速通信を実現するためのサンプルプログラムを提供します。高速通信を実現するためには、適切な GAP パラメータの設定と連続的にデータを送信することが必要になります。本アプリケーションノートでは、この二つのプログラム例とその仕組みを解説します。

動作確認デバイス

Target Board for RX23W

関連ドキュメント

- RX23W グループ Target Board for RX23W クイックスタートガイド(R20QS0014)
- RX23W グループ ユーザーズマニュアル ハードウェア編 (R01UH0823)
- Bluetooth Low Energy プロトコルスタック 基本パッケージ ユーザーズマニュアル(R01UW0205)
- RX23W グループ BLE Module Firmware Integration Technology アプリケーションノート (R01AN4860)
- RX23W グループ Bluetooth Low Energy プロファイル開発者ガイド アプリケーションノート (R01AN4553)

目次

1. 高速通信用サンプルプログラム

高速通信用サンプルプログラムは、Target Board for RX23W(ターゲットボード)2 台を接続し、高速通信を実現するために GAP パラメータを最適な値に設定します。その後、双方向の連続的な GATT 通信のスループットを計測します。

1.1 プロジェクト

本アプリケーションノートはセントラルとペリフェラル用のプログラムファイルとプロジェクトを用意しています。それぞれのプログラムの Bluetooth Low Energy 通信でのロールを表 1.1 に示します。プロジェクトを e²studio のワークスペースへ追加する方法は、「第 5.2 節 e²studio のワークスペースへのプロジェクト追加方法」をご覧ください。表 1.2 に本プロジェクトの動作確認環境を示します。

表 1.1 サンプルプログラムのプロジェクトとロール

プロジェクト名	GAP ロール	GATT ロール
tbrx23w_throughput_service_client	セントラル	クライアント
tbrx23w_throughput_service_server	ペリフェラル	サーバー

表 1.2 動作確認環境

ソフトウェア	バージョン
e²studio	7.6.0
CC-RX コンパイラ	2.08

1.2 動作環境セットアップ

サンプルプログラムはターゲットボード 2 枚を対向接続し通信します。

各ターゲットボードに、セントラル用ファイル(tbrx23w_throughput_service_client.mot)と、ペリフェラル用ファイル(tbrx23w_throughput_service_server.mot)をそれぞれ書き込みます。ターゲットボードへのプログラムファイルの書き込み方法は「第 5.1 節ターゲットボードへのプログラムファイルの書き込み方法」をご覧ください。

セントラル用プログラムに実装された BLE プロトコルスタックの Command Line Interface (CLI)機能を使用して端末エミュレータからプログラムの動作を制御します。セントラル用プロブラムを書き込んだデバイスは、USB で PC と接続します。ターゲットボードはシリアルポートとして認識されます。コンピュータの端末エミュレータに表 1.3 の項目を設定します。ペリフェラル用プログラムを書き込んだデバイスは、CLI 機能を利用しないため、PC のポートか電源に接続します(図 1.1)。

表 1.3 端末エミュレータの設定

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

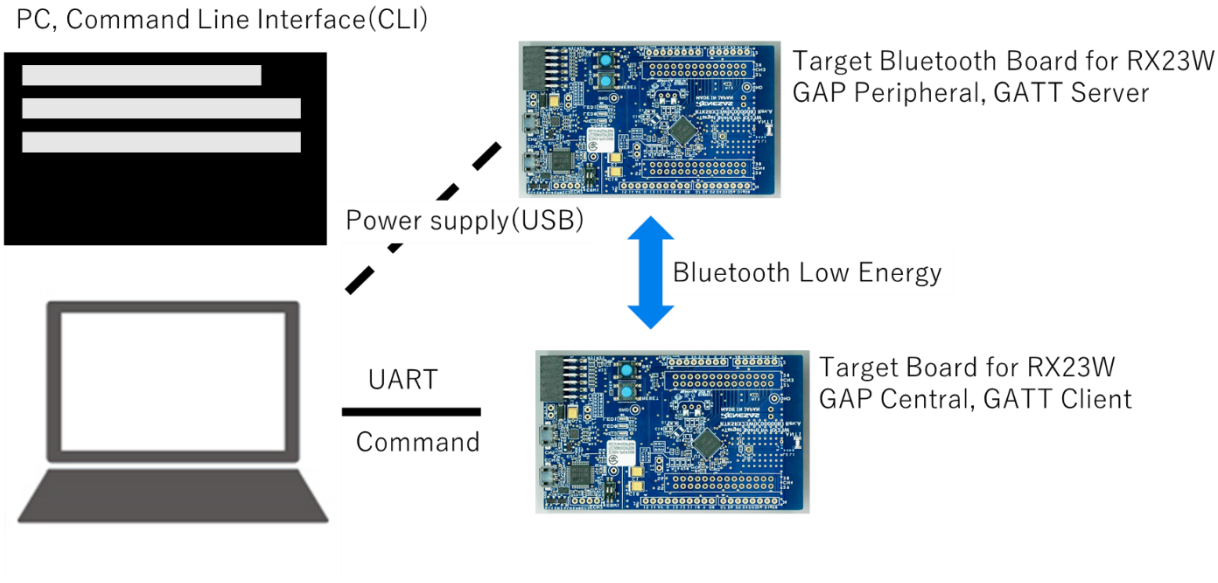

図 1.1 サンプルプログラムの動作セットアップ

1.3　プログラムの動作

USB ケーブルをマイコン側の USB 端子に繋ぎ変え、ターゲットボードのエミューレータスイッチ 2 を OFF にします(図 1.2)。

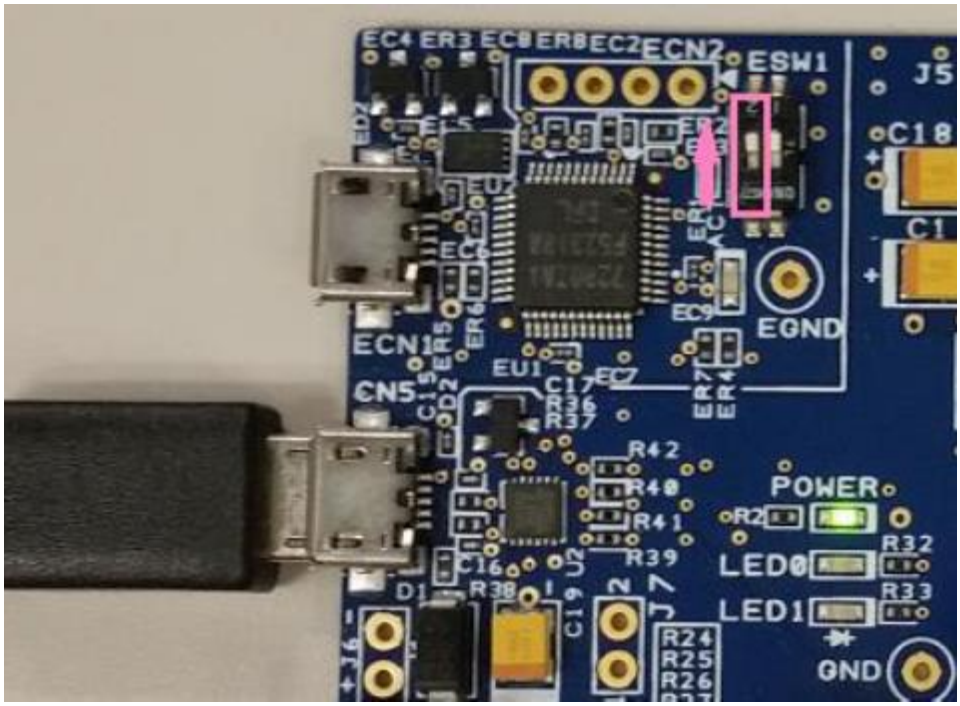

図 1.2 プログラムの実行

プログラムが実行されると、セントラル、ペリフェラルはそれぞれスキャニング、アドバタイズを行い、自動で接続されます。接続確立後は、GATT 通信を行うためのサービスディスカバリと、高速通信実現のために GAP パラメータと MTU が変更されます(図 1.3)。高速通信実現のために設定されるパラメータとその値を表 1.4 に示します。

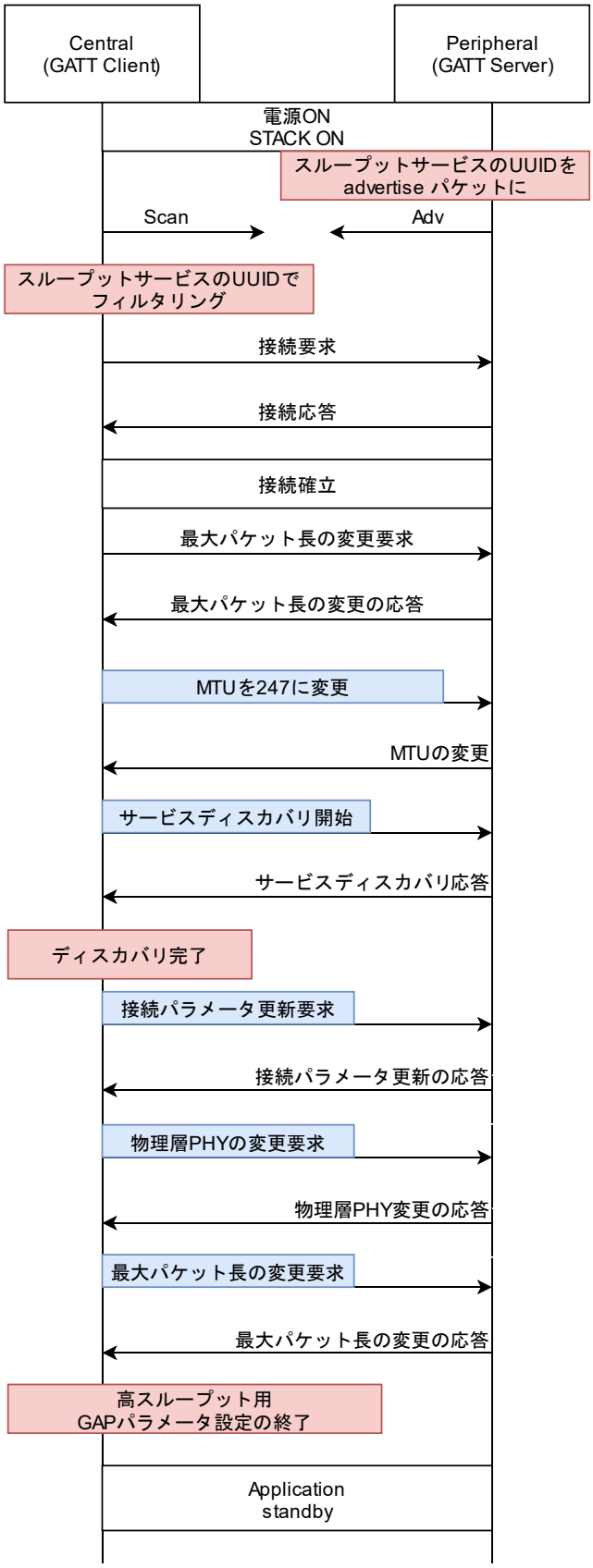

図 1.3 サンプルプログラムの立ち上がり動作

表 1.4 サンプルプログラム立ち上げ時に設定されるパラメータとその値

パラメータ	値
コネクションインターバル	50 (msec)
物理層 PHY	2M PHY
最大パケット長	251 (byte)
MTU	247 (byte)

高速通信実現のためのパラメータ設定が完了すると、CLI 画面に”Enter command and start Throughput measurement”と表示されます(図 1.4)。CLI 画面にコマンドを入力して、スループットサービスによる GATT 通信を開始します。サンプルプログラムでは、セントラルがスループットサービスのクライアント、ペリフェラルがスループットサービスのサーバーとして動作します。

デバイスアドレス
スキャン停止
接続確立

MTUの変更完了
最大パケット長の変更
最大送信バイト数 251 byte
最大送信時間2.120 msec
最大受信バイト数 251 byte
最大受信時間2.120 msec
サービスディスカバリ完了
接続パラメータの更新
コネクションインターバル 50msec
スレーブレイテンシ 0
スーパービジョンタイムアウト 32000msec
物理層PHYの設定
物理層PHYの更新
送信PHY 2M PHY
受信PHY 2M PHY
最大パケット長の設定
最大パケット長の更新
最大送信バイト数 251 byte
最大送信時間 1.064 msec
最大受信バイト数 251 byte
最大受信時間 1.064 msec
GAP設定の完了
コマンドを入力してください

```
74:90:50:0F:0F:0F pub ff 0000
receive BLE_GAP_EVENT_SCAN_OFF result : 0x0000
receive BLE_GAP_EVENT_CONN_IND result : 0x0000
gap: connected conn_hdl:0x0020, addr:74:90:50:0F:0F:0F pub
ATT_MTU has been changed 247
receive BLE_GAP_EVENT_DATA_LEN_CHG result : 0x0000, conn_hdl : 0x0020
tx_octets : 0x00fb
tx_time   : 0x0848
rx_octets : 0x00fb
rx_time   : 0x0848
Service Discovery Completed!
receive BLE_GAP_EVENT_CONN_PARAM_UPD_COMP result : 0x0000, conn_hdl : 0x0020
conn_intv    : 0x0028
conn_latency : 0x0000
sup_to       : 0x0c80
receive BLE_GAP_EVENT_PHY_SET_COMP result : 0x0000, conn_hdl : 0x0020
receive BLE_GAP_EVENT_PHY_UPD result : 0x0000, conn_hdl : 0x0020
tx_phy : 0x0002
rx_phy : 0x0002
receive BLE_GAP_EVENT_SET_DATA_LEN_COMP result : 0x0000, conn_hdl : 0x0020
receive BLE_GAP_EVENT_DATA_LEN_CHG result : 0x0000, conn_hdl : 0x0020
tx_octets : 0x00fb
tx_time   : 0x0428
rx_octets : 0x00fb
rx_time   : 0x0428
High-speed GAP Configuration has been completed.
Enter command and start Throughput measurement.
```

図 1.4 セントラルデバイスの電源投入後の CLI 画面

サンプルプログラムは、サーバーからクライアント、クライアントからサーバーの双方向の GATT 通信を行います。サポートする GATT 動作の種類と制御コマンドを表 1.5 に示します。これらのコマンドは Throughput client (thc)コマンドとして CLI 機能に登録されています。コマンドを実行すると GATT 通信におけるスループットの計測が開始されます。データ送信の方向によって、送信方法とスループット測定方法が異なります。

表 1.5 GATT 動作の実行コマンド

GATT 動作	開始コマンド	停止コマンド
Notification	thc start notification	thc stop receive
Indication	thc start indication	thc stop receive
Write Without Response	thc start write_without_response	thc stop transmit
Write	thc start write	thc stop transmit

1.3.1　サーバーからクライアントにデータを送信する(Notification, Indication)場合

コマンドが実行されるとクライアントは、サーバーの CCCD に書き込み、Notification/ Indication 動作を有効にします。Notification の場合は、クライアントの応答を待たないので連続してサーバーから送信します(図 1.5)。Indication はクライアントの応答を待つため Confirmation パケットを待ってから送信します(図 1.6)。

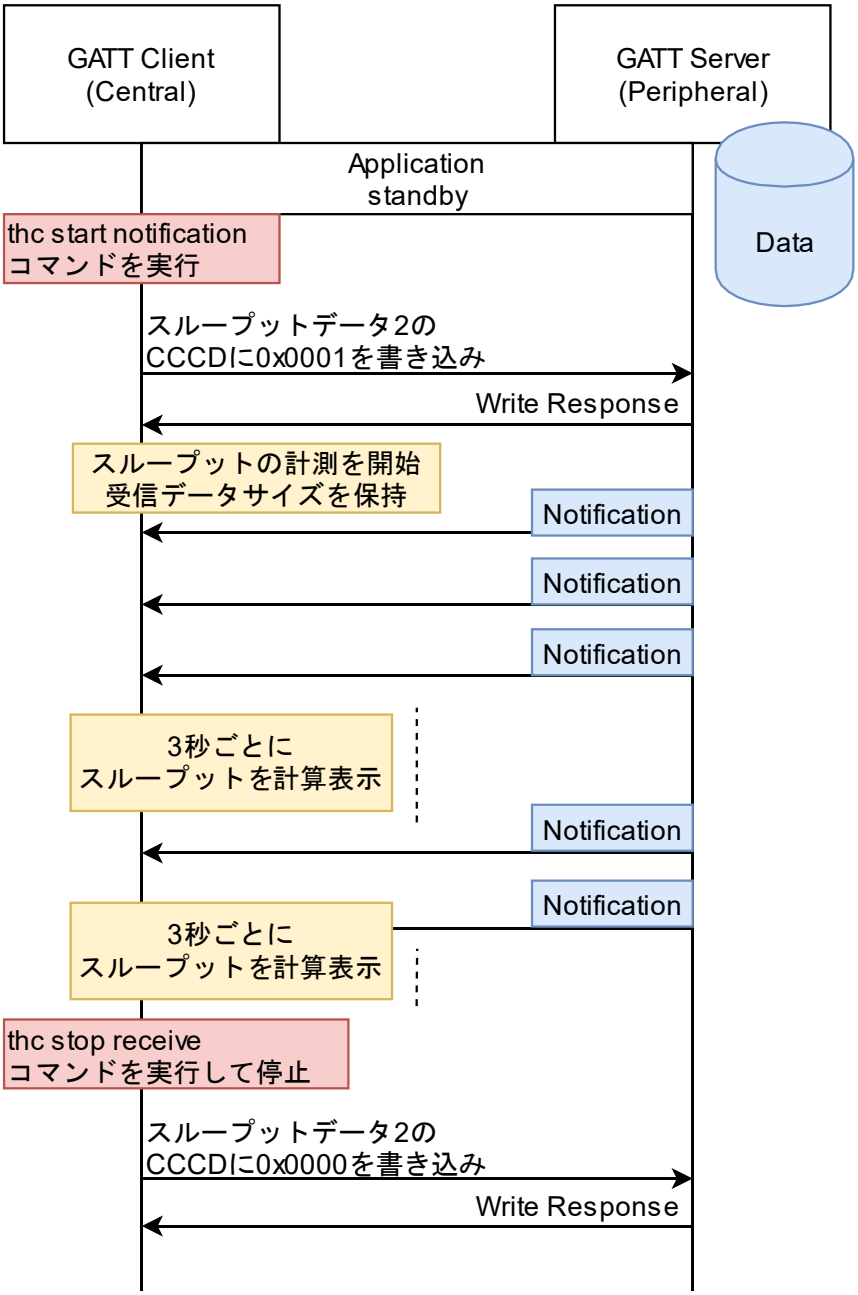

図 1.5 Notification 実行時サンプルプログラムの動作

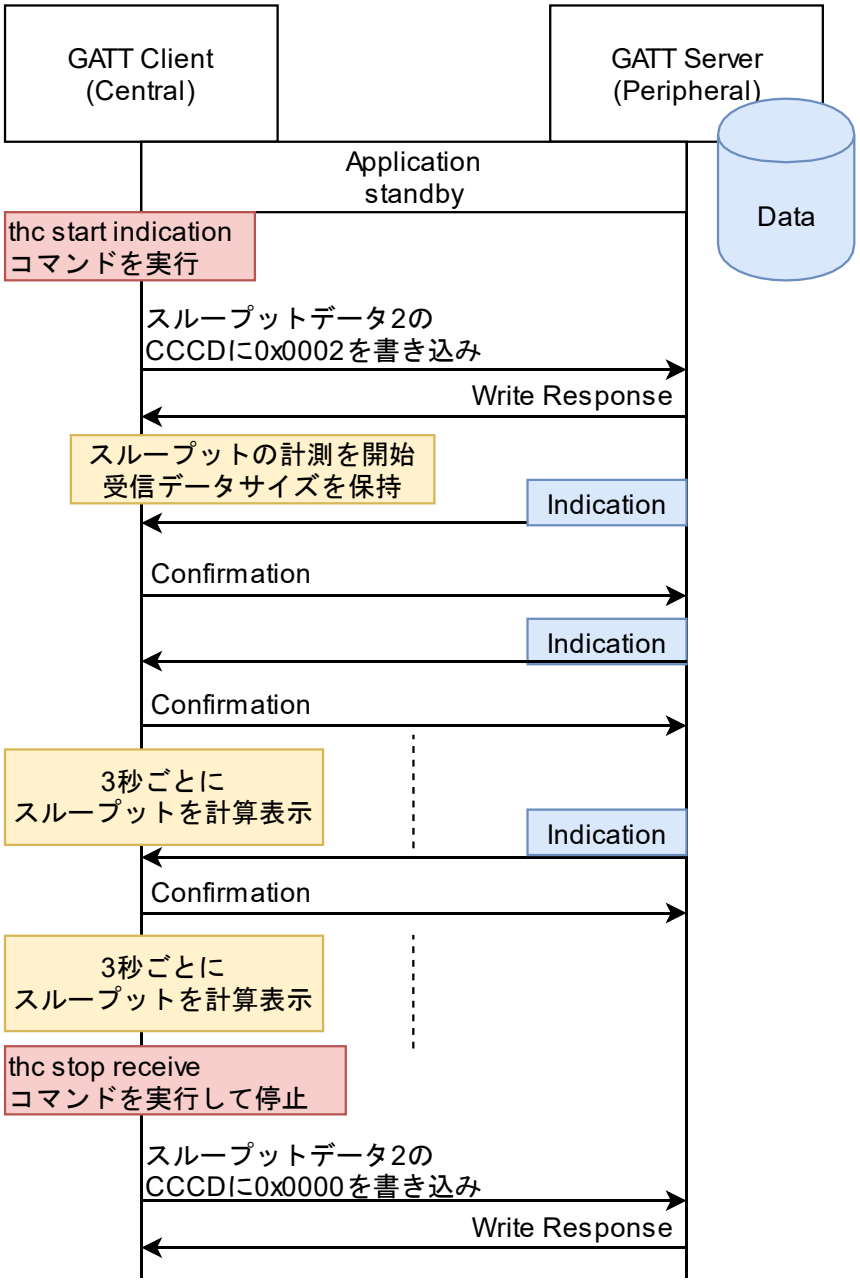

図 1.6 Indication 実行時のサンプルプログラムの動作

クライアントは 3 秒毎に、受信した Notification/Indication のデータサイズの合計と経過時間(3sec)からスループットを計算します。サーバーは、Notification/ Indication が停止されるまで、データを送信し続けます。Notification/Indication の動作を行った場合の画面を図 1.7、図 1.8 に示します。

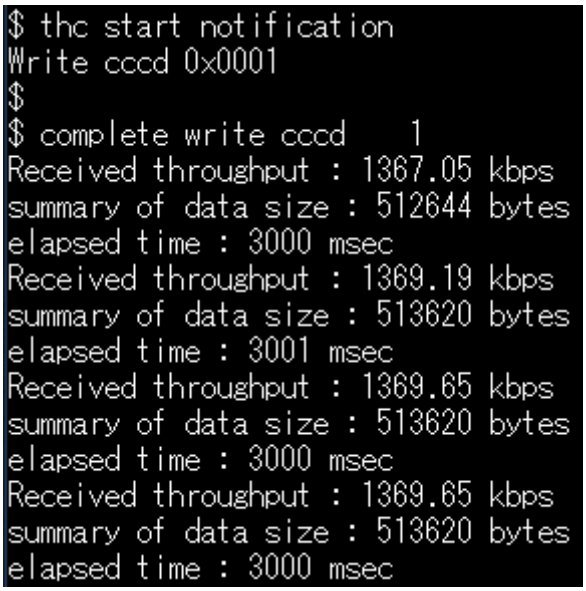

図 1.7 Notification 実行時の CLI 画面

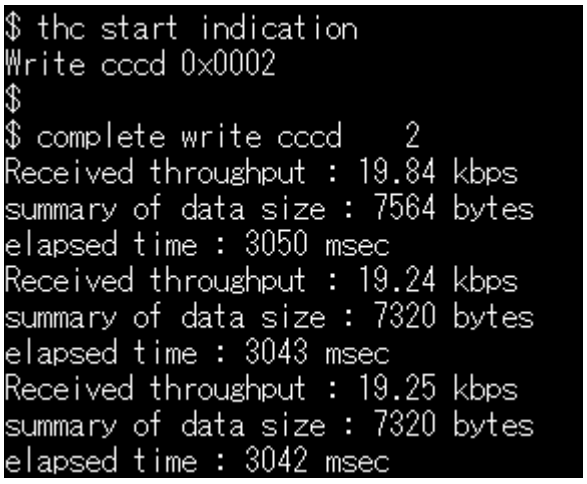

図 1.8 Indication 実行時の CLI 画面

1.3.2　クライアントからサーバーにデータを送信する(Write Without Response, Write)場合

Write Without Response, Write コマンドが実行された時のサンプルプログラムの動作をそれぞれ図 1.9、図 1.10 に示します。コマンドが実行されるとクライアントは、ROM 上に配置された 5k byte のデータを、MTU 毎に区切って送信します。5k byte のデータ全てを送信し終えると、スループットを表示します。スループットの計測は、コマンド実行から BLE プロトコルスタックの送信バッファに全てのデータが格納されるまでの時間を用いて計算されます。そのため、実際の送信スループットよりも少し大きい値が表示されます。

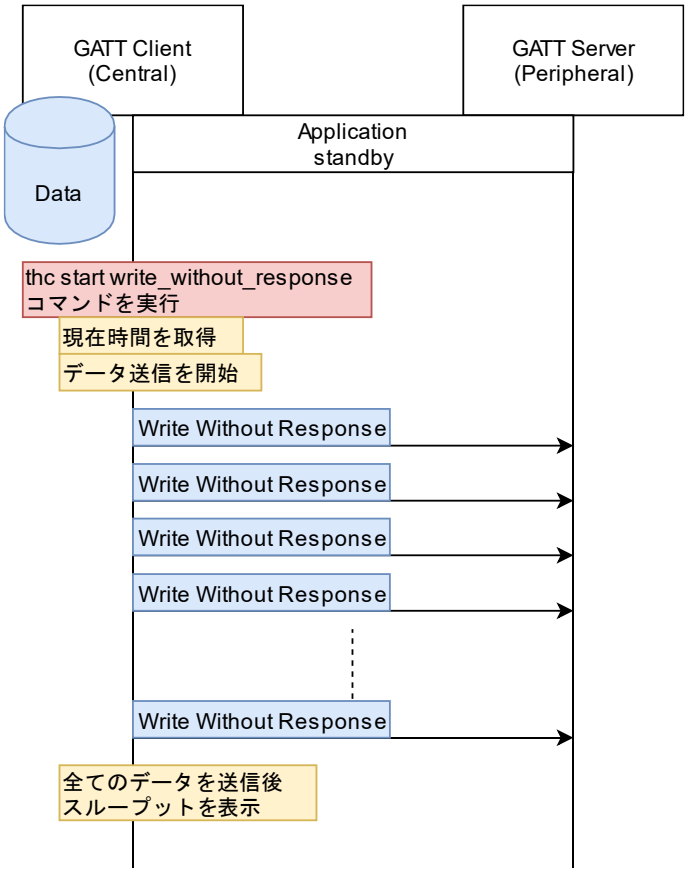

図 1.9 Write Without Response 実行時のサンプルプログラムの動作

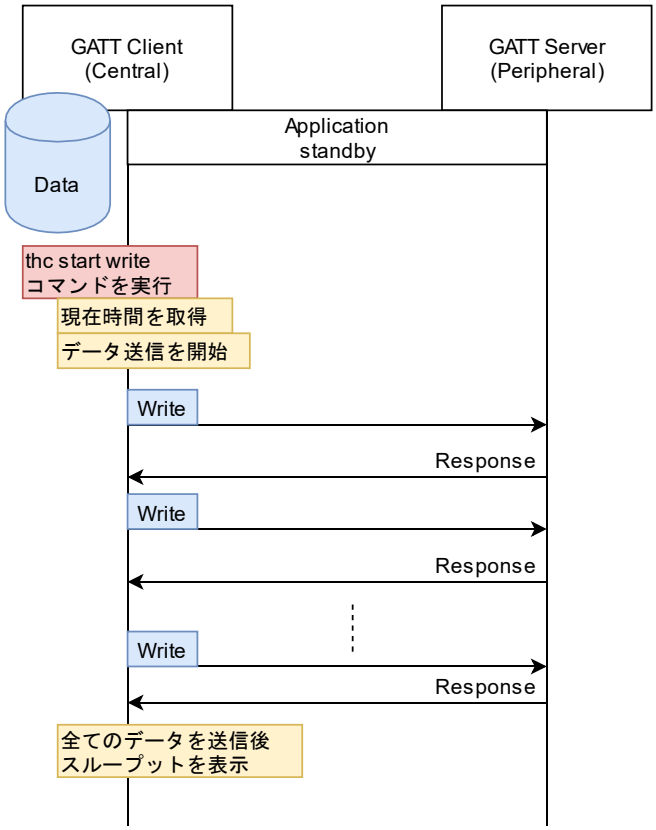

図 1.10 Write 実行時のサンプルプログラムの動作

```
$ thc start write_without_response
Start write without response
$
$ Transmit throughput : 1365.33 kbps
summary of data size : 5120 bytes
elapsed time : 30 msec

$
$ thc start write
Start write
$
$ Transmit throughput : 20.64 kbps
summary of data size : 5120 bytes
elapsed time : 1984 msec
```

図 1.11 クライアントからサーバーへの送信動作時の CLI 画面

1.3.3　GAP パラメータの変更

サンプルプログラムは、GATT 動作の制御だけでなく、スループットに影響を与える GAP パラメータを変更することができます。サンプルプログラムの CLI から変更可能なパラメータとコマンドを表 1.6 に示します。これらのコマンドは BLE プロトコルスタックの CLI 機能を使用しています。パラメータの入力方法などは、「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)」の[5.1.1.1 GAP コマンド]を参照してください。

表 1.6 変更可能なパラメータとコマンド

GAP パラメータ	コマンド	パラメータ
コネクションインターバル	gap conn_cfg update	connection handle connection interval slave latency supervision timeout
物理層 PHY	gap conn_cfg phy	connection handle tx phy rx phy coded scheme
最大パケット長	gap conn_cfg data_len	connection handle max tx packet length max tx time
通信の暗号化	gap auth start	connection handle

1.4　独自(スループット)サービス

サンプルプログラムで送受信されるアプリケーションデータは表 1.7 に示す独自(スループット)サービスを使用しています。これらのキャラクタリスティックのデータは、アプリケーションプログラム上では BLE FIT モジュールが提供する app_lib の Profile Common Library が定義する st_ble_seq_data_t 構造体として表現されています。

表 1.7 スループットサービス

名前	UUID	Properties
Throughput Service	9CEF3D10-7FAB-49DC-AB89-762C9079FE96	---
Throughput Data 1 Characteristic	9CEF3D11-7FAB-49DC-AB89-762C9079FE96	Write Write Without Response
Throughput Data 2 Characteristic	9CEF3D12-7FAB-49DC-AB89-762C9079FE96	Indicate Notify
Client Characteristic Configuration Descriptor	0x2902	Read Write

1.5　ソフトウェア構成

サンプルプログラムは、QE for BLE と BLE FIT モジュールから成る BLE ソリューションツールを使用して作成されています。サンプルプログラムのコンポーネント図を図 1.12 に示します。アプリケーションには R_BLE_API や他のモジュール、サービス API からのイベントや通知を受け取るためのコールバック関数がそれぞれ実装されています。

サンプルプログラムでは、連続的な送信動作の実現のために TxFlow 制御プログラムを使用しています。また、Highspeed GAP Config プログラムを使用して、高速通信実現のための GAP 設定を行っています。

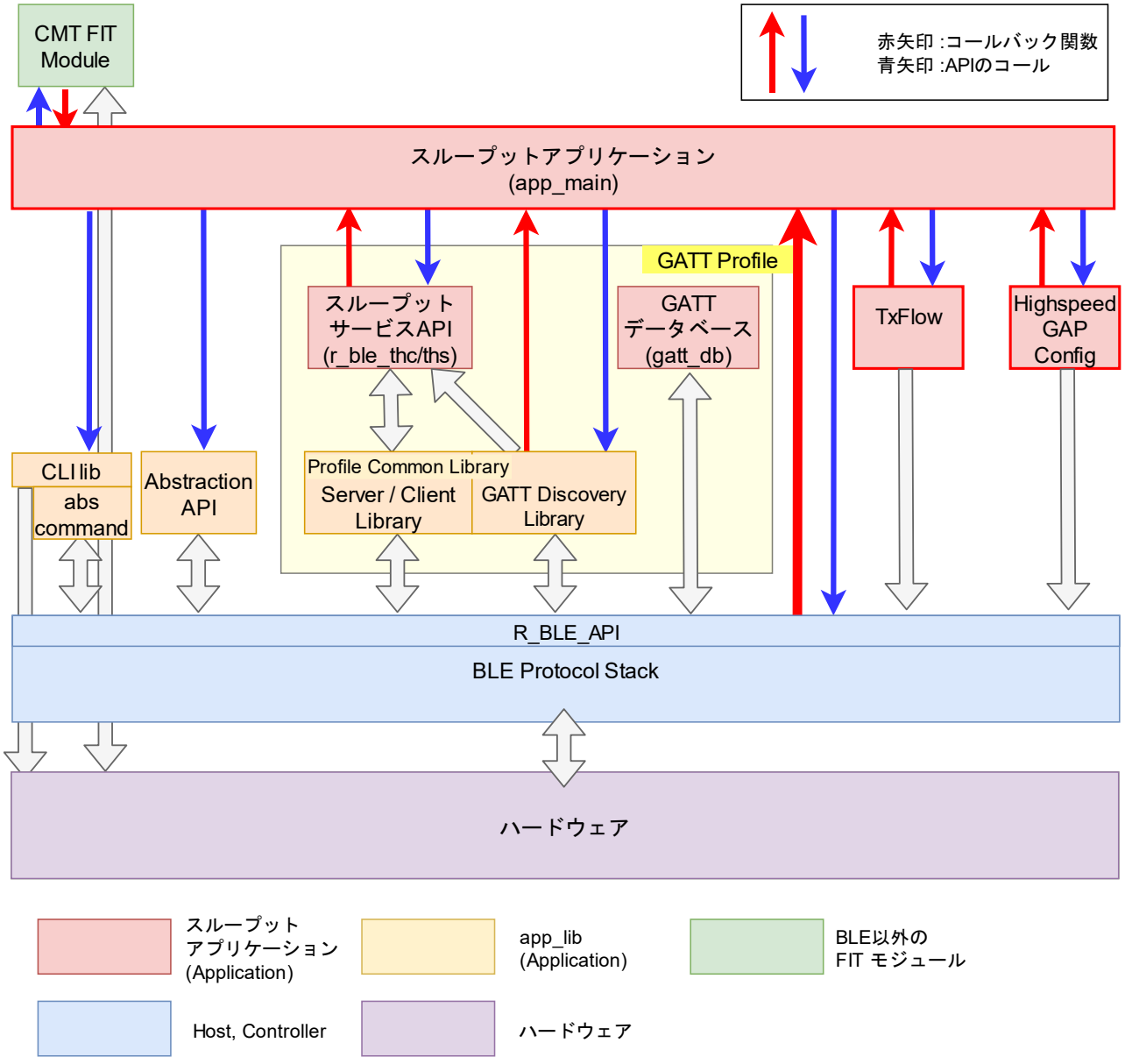

図 1.12 サンプルプログラムのコンポーネント図

1.6　ファイル構成

サンプルプログラムのファイル構成を図 1.13 に示します。サンプルプログラムは、e²studio の Smart Configurator を使用して作成されています。ファイル名の右側の説明は以下の内容を示します。

(A)：サンプルプログラムが新たに作成、追加したファイル

(QE for BLE)：QE for BLE によって生成させるコード

(SC)：その他の Smart Configurator によって生成されるフォルダ

(Server)：サーバープログラムに実装されているファイル

(Client)：クライアントプログラムに実装されているファイル

```
-- Config_BLE_PROFILE (QE for BLE)
        -- app_main.c                          (QE for BLE) (A) (Server, Client)
           アプリケーションフレームワーク
        -- gatt_db.c                           (QE for BLE) (Server, Client)
        -- gatt_db.h                           (QE for BLE) (Server, Client)
           GATT データベース
        -- r_ble_gapc.c                        (QE for BLE) (Server, Client)
        -- r_ble_gapc.h                        (QE for BLE) (Server, Client)
        -- r_ble_gaps.c                        (QE for BLE) (Server, Client)
        -- r_ble_gaps.h                        (QE for BLE) (Server, Client)
        -- r_ble_gatc.c                        (QE for BLE) (Client)
        -- r_ble_gatc.h                        (QE for BLE) (Client)
        -- r_ble_gats.c                        (QE for BLE) (Server)
        -- r_ble_gats.h                        (QE for BLE) (Server)
           GAP,GATT サービス API
        -- r_ble_thc.c                         (QE for BLE) (Client)
        -- r_ble_thc.h                         (QE for BLE) (Client)
        -- r_ble_ths.c                         (QE for BLE) (Server)
        -- r_ble_ths.h                         (QE for BLE) (Server)
           スループット測定用独自サービス API
        -- r_ble_highspeed_gap_config.c        (A) (Client)
        -- r_ble_highspeed_gap_config.h        (A) (Client)
           高速通信用 GAP 設定プログラム
        -- r_ble_txflow.c                      (A) (Server, Client)
        -- r_ble_txflow.h                      (A) (Server, Client)
           TxFlow 制御プログラム
-- general                          (SC)
-- r_ble_qe_utility                 (SC)
-- r_ble_rx23w                      (SC)
        BLE FIT Module
-- r_bsp                            (SC)
-- r_byteq                          (SC)
-- r_cmt_rx                         (SC)
-- r_config                         (SC)
-- r_flash_rx                       (SC)
-- r_gpio_rx                        (SC)
-- r_irq_rx                         (SC)
-- r_lpc_rx                         (SC)
-- r_pincfg                         (SC)
-- r_sci_rx                         (SC)
```

図 1.13 サンプルプログラムのファイル構成

2.　高速通信実現のためのプログラム

Bluetooth Low Energy で高速通信を実現するためには、一つに More Data による通信のために連続的な送信要求を行うこと、二つに GAP の設定を最適化することが必要です。

本アプリケーションノートでは、これらの動作を簡単に行うための 2 つのプログラムを提供します。サンプルプログラムはこれらのプログラムを利用して高速通信を実現しています。

2.1　TxFlow 制御プログラム

TxFlow 制御プログラムは、BLE プロトコルスタックが提供する Vendor Specific API の送信フロー制御機能を利用します。本プログラムは送信バッファに空きがある場合に送信用コールバック関数を呼び出します。BLE プロトコルスタックの送信フロー制御機能については、「RX23W グループ BLE Module Firmware Integration Technology アプリケーションノート(R01AN4860)」に同梱されている R_BLE API ドキュメント(r_ble_api_spec.chm)をご覧ください。

アプリケーションは、TxFlow 制御プログラムからの送信可能イベントを受けて連続的な送信要求を行うことで、高速通信を実現します。

本プログラムの型定義と関数をそれぞれ表 2.1、表 2.2 に示します。

表 2.1 TxFlow 制御プログラムの型定義

```
typedef void(*r_txflow_cb_t)(void)
```

表 2.2 TxFlow 制御プログラムの関数一覧

関数名	動作
R_BLE_TxFlow_Init	プログラムを初期化して、送信可能イベントを受け取るコールバック関数を設定します。
R_BLE_TxFlow_Start	送信可能イベントの通知を開始します。
R_BLE_TxFlow_Stop	送信可能イベントの通知を停止します。
R_BLE_TxFlow_TxFlowChg	Vendor Specific API のコールバック関数内で呼びだし、送信フロー制御機能によるイベントを渡します。

本プログラムを利用した連続送信を行うアプリケーションの動作例を図 2.1 に示します。

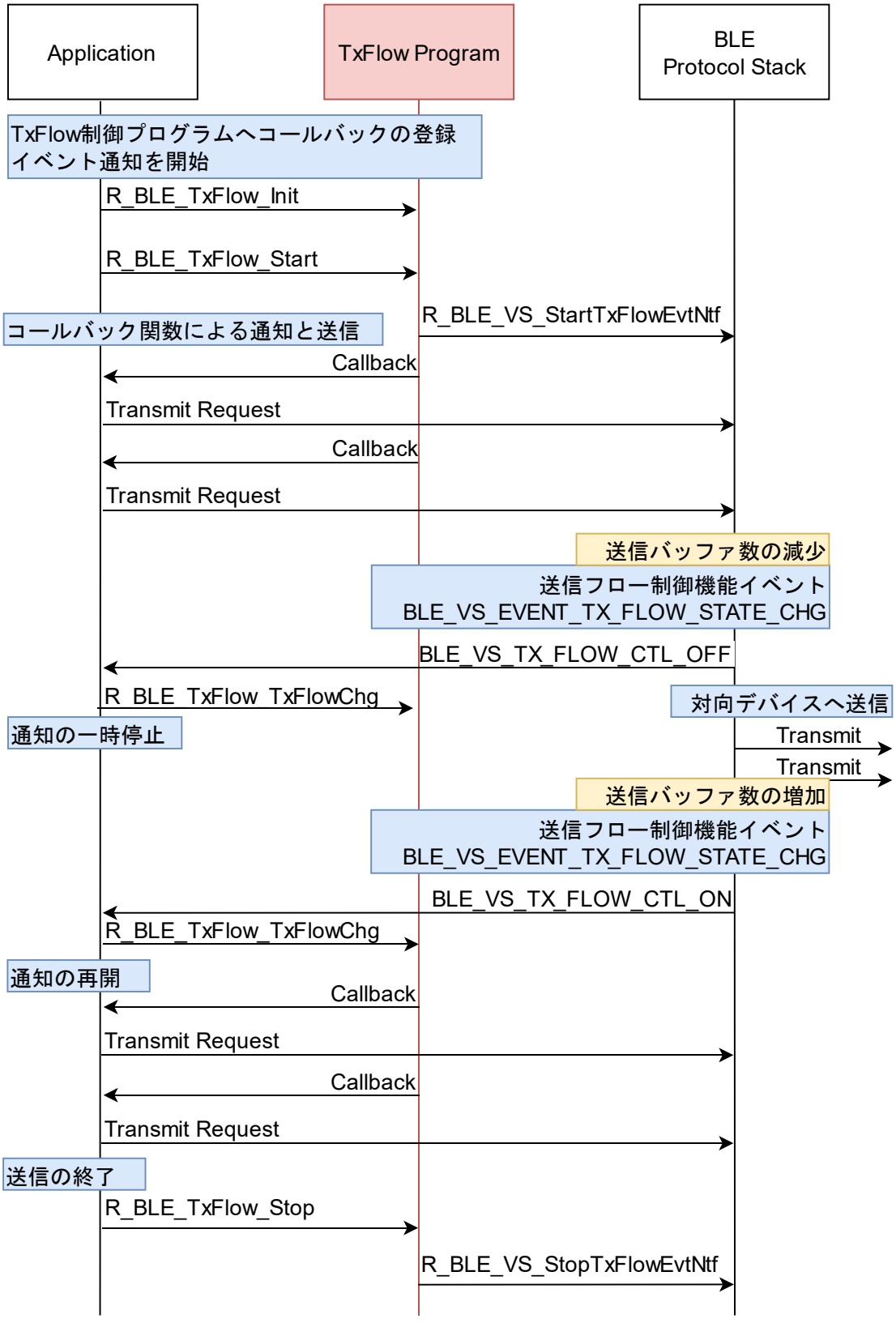

図 2.1 TxFlow 制御プログラムの動作例

2.1.1　API リファレンス

TxFlow 制御プログラムが持つ関数の詳細を表 2.3 に示します。

表 2.3 TxFlow 制御プログラムの API リファレンス

R_BLE_TxFlow_Init(r_txflow_cb_t cb)		
プログラムの初期化と、コールバック関数を登録します。 ここで登録したコールバック関数は、BLE プロトコルスタックの送信バッファに空きがあり、送信可能な場合に呼び出され続けます。 *登録したコールバック関数は R_BLE_TxFlow_Stop もしくは、R_BLE_TxFlow_TxFlowChg によって通知が停止されるまで、1 回の R_BLE_Execute()で繰り返し呼び出されます。		
Parameters:		
r_txflow_cb_t	*cb*	送信可能イベントを通知するコールバック関数。
Return:		
r_ble_status_t	*BLE_SUCCESS*	正常終了。

R_BLE_TxFlow_Start(void)		
送信可能イベントの通知を開始します。		
Parameters:		
None		
Return:		
r_ble_status_t	*BLE_SUCCESS*	正常終了。
	BLE_ERR_INVALID_OPERATION	既に通知が開始されています。
	BLE_ERR_INVALID_PTR	コールバック関数が未登録です。

R_BLE_TxFlow_Stop(void)		
送信可能イベントの通知を停止します。		
Parameters:		
None		
Return:		
None		

R_BLE_TxFlow_TxFlowChg(st_ble_vs_tx_flow_chg_evt_t* evt_data)		
Vendor Specific API の送信フロー制御機能のイベント(BLE_VS_EVENT_TX_FLOW_STATE_CHG)に通知されるデータ変数を受け取ります。 通常は、Vendor specific API に登録したコールバック関数内で呼び出されます。 この関数を利用して、送信フロー制御イベントを取得しない場合、TxFlow 制御プログラムは正常に動作しません。		
Parameters:		
st_ble_vs_tx_flow_chg_evt_t*	*evt_data*	Vendor Specific API の BLE_VS_EVENT_TX_FLOW_STATE_CHG イベントのデータ。
Return: r_ble_status_t		
r_ble_status_t	*R_BLE_SUCCESS*	正常終了。

2.2　Highspeed GAP Config プログラム

Highspeed GAP Config プログラムは、高速通信実現のために GAP パラメータの設定を最適値へ変更します。本プログラムを実行することで、高速通信に最適な GAP 設定に変更することができます。

本プログラムによって変更される GAP パラメータと設定される値を表 2.4 に示します。

表 2.4 高速通信実現のための GAP 設定

パラメータ	値
コネクションインターバル	50 (msec)
物理層 PHY	2M PHY
最大パケット長	251 (byte)

本プログラムの型定義と関数をそれぞれ表 2.5、表 2.6 に示します。

表 2.5 Highspeed GAP Config プログラムの型定義

```
typedef void(*r_highspeed_confgap_comp_cb_t)(uint16_t , ble_status_t);
```

表 2.6 Highspeed GAP Config プログラムの関数一覧

関数名	動作
R_BLE_Highspeed_ConfigGapConnPram	完了通知を受け取るコールバック関数を登録し、GAP 設定を開始します。 コールバック関数は、GAP 設定が終了するか、途中で切断されるとコールされます。
R_BLE_Highspeed_GapCb	GAP API のコールバック関数内で呼び出し、GAP のイベントを渡します。

図 2.2 に本プログラムの動作シーケンスを示します。コネクションインターバル、物理層 PHY、最大パケット長の順に変更します。最大パケット長の設定が終了すると、R_BLE_Highspeed_ConfigGapConnPram に登録したコールバック関数に BLE_SUCCESS が渡されコールされます。設定途中で接続が切断されるとコールバック関数に BLE_ERR_DISCONNECTED が渡されコールされます。

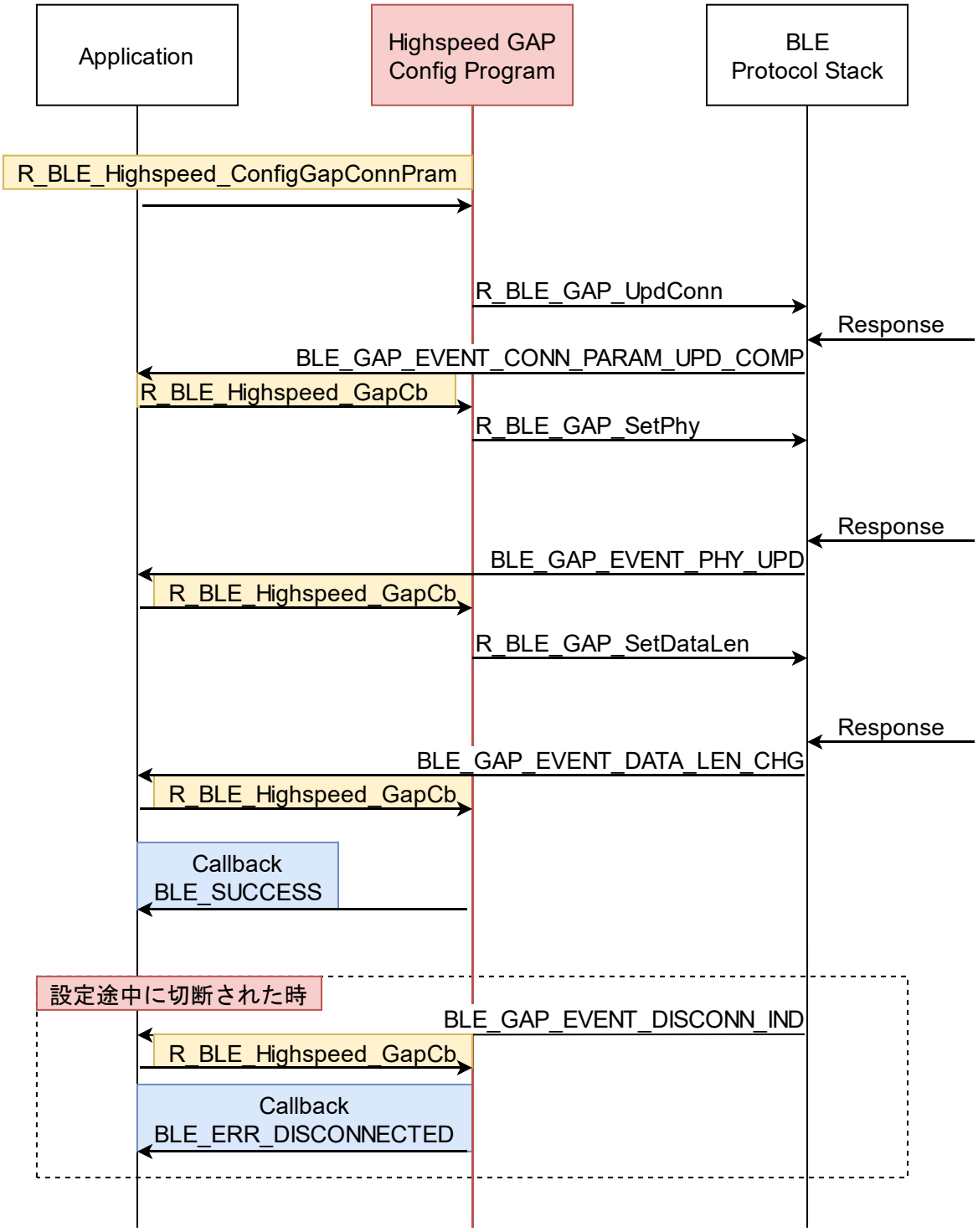

図 2.2 Highspeed GAP Config プログラムの動作

2.2.1　API リファレンス

Highspeed GAP Config プログラムが持つ関数の詳細を表 2.7 に示します。

表 2.7 Highspeed GAP Config プログラムの API リファレンス

R_BLE_Highspeed_ConfigGapConnParam(uint16_t conn_hdl, r_highspeed_confgap_comp_cb_t cb)		
完了通知を受け取るコールバック関数を登録し、GAP 設定を開始します。		
Parameters:		
uint16_t	*conn_hdl*	GAP 設定を行う接続のコネクションハンドルです。
r_highspeed_confgap_comp_cb_t	*cb*	GAP 設定が終了した通知を受け取るコールバック関数を登録します。 GAP 設定が正常に終了した場合は BLE_SUCCESS が、途中で切断された場合は BLE_ERR_DISCONNECTED が渡されます。
Return:		
r_ble_status_t	*BLE_SUCCESS*	正常終了。
	BLE_ERR_INVALID_OPERATION	既に GAP 設定が開始されています。
	BLE_ERR_INVALID_PTR	コールバック関数が NULL です。

R_BLE_Highspeed_GapCb(uint16_t type, ble_status_t result, st_ble_evt_data_t *p_data)		
GAP API からのイベントを受け取ります。		
Parameters:		
uint16_t	*type*	GAP API で定義される GAP のイベントコードです。
ble_status_t	*result*	GAP API で定義されるイベントの結果です。
st_ble_evt_data_t *	*p_data*	GAP API で定義されるイベントに対応したデータです。
Return:		
None		

3.　Bluetooth Low Energy　とスループット

本章では、Bluetooth Low Energy 通信の仕組みとスループットとの関係をごく簡単に説明します。通信規格の詳細は Bluetooth の仕様書でご確認ください。Bluetooth Low Energy は、大きく分けて 3 つのレイヤーを持ちます。Controller と Host は、Host Controller Interface(HCI) によって接続されます。Application と Host は API(BLE FIT Module では R_BLE_API)によって接続されます(図 3.1)。

Bluetooth Low Energy では、実際の通信経路や送受信間隔などはコントローラー層の Link Layer が制御します。高速通信を実現するためには、この Link Layer の動作が重要になります。Link Layer の動作は Host 層の GAP によって設定されます。

一方で、アプリケーションとして意味のあるデータを送信する場合には、ホスト層の GATT を利用します。GATT では、プロファイルによってアプリケーションデータの送信手順や送受信されるデータ構造が決められています。高速通信を実現するためには、このプロファイルの設計も重要になります。

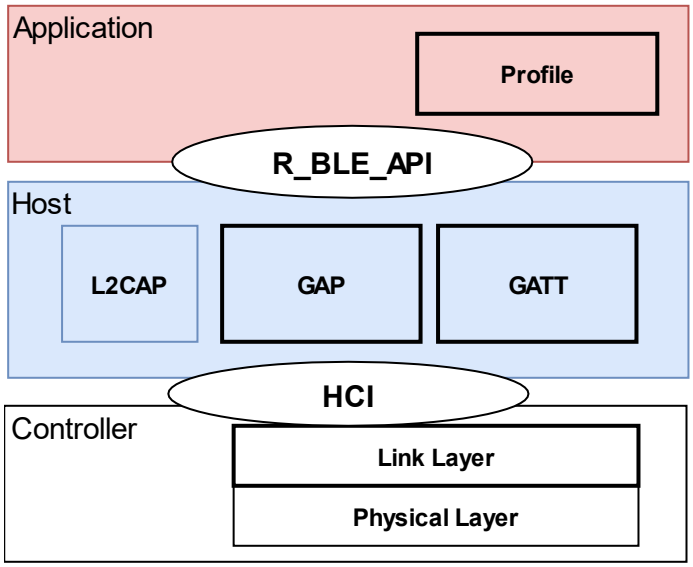

図 3.1 Bluetooth Low Energy の三つのレイヤー

3.1　Generic Access Profile (GAP)

接続するデバイスの検出や、接続確立などの手順は Generic Access Profile(GAP)によって決められています。GAP は Link Layer の動作設定を行い、これらの手順を実現します。

3.1.1　デバイスの検出と接続確立

Bluetooth Low Energy では、一方のデバイスが自身のデバイス情報を発信（アドバタイジング）し、他方のデバイスがデバイスの検出（スキャニング）と接続要求（イニシエーティング）を実行することで、接続が確立されます。このとき、スキャニングと接続要求を行うデバイスがセントラル、アドバタイズを行うデバイスがペリフェラルです。接続確立後の周波数マップや通信間隔（コネクションインターバル）などの接続維持に関わるパラメータは、セントラルが決定します。GAP では、

- コネクションインターバル
- 物理層の設定
- 最大パケット長
- ペアリング情報

などの情報が管理されます。

3.1.2　接続確立後の通信

Bluetooth Low Energy では接続の確立後、デバイスはコネクションインターバル毎に発生するコネクションイベントで無線フレームの交換を行います。Link Layer ではセントラルがマスター、ペリフェラルがスレーブとなります。無線フレームは、事前に共有されたコネクションイベントに合わせてマスターから送信されます。(図 3.2)

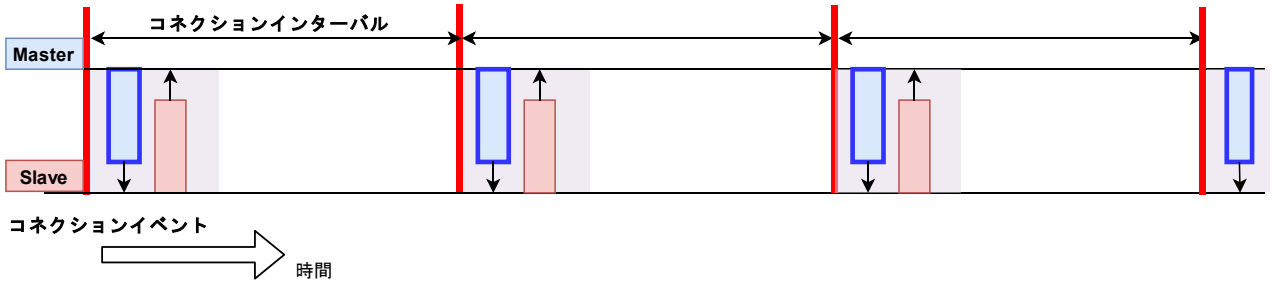

図 3.2 コネクションイベントと無線フレームの交換

コネクションイベントで無線フレームが交換されることで接続は維持されます。どちらかのデバイスに追加で送信するデータがある場合には、無線フレーム内の More Data Bit がセットされ、コネクションイベントが延長されます。コネクションイベントは、お互いの More Data Bit がセットされなくなるか、受信パケットの異常が発生すると終了します。一度コネクションイベントが終了すると、次のコネクションイベントまで無線フレームの交換は行われません(図 3.3)。高速通信を実現するためには、この More Data による通信を行うことが重要になります。

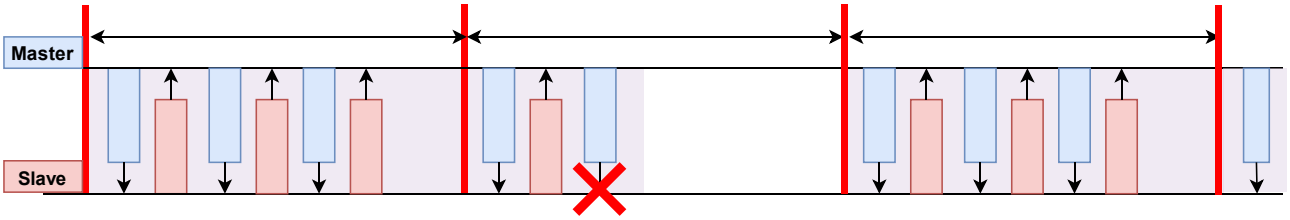

図 3.3 More Data による通信

3.1.3　コネクションインターバルの設定

コネクションインターバルが変更された場合の Link Layer 動作の模式図を図 3.4 に示します。コネクションインターバルが変更されても、More Data による通信が安定している場合には、スループットへの大きな変化はありません。コネクションインターバルを極端に短くするとコネクションイベント直前の待機時間のオーバヘッドがコネクションイベント毎に発生し、スループットが低下します。

図 3.5 に通信環境が良好で、フレームの交換が必ず成功すると仮定した場合のコネクションインターバルとスループットの関係を示します。GAP、GATT の設定は PHY:2M PHY、最大パケット長 251 byte、MTU 247 byte で、244 byte のアプリケーションデータを常に Notification している場合の値です。コネクションインターバルが 7.5msec の場合は 1040kbps 程度になります。

コネクションインターバルあたりのスループットは、コネクションイベント直前の待機時間 $T_{overhead}$、無線フレームが往復する最小転送時間 T_{frame}、アプリケーションデータ長(L_{data})から計算されます。パケット長が 251 バイトの場合 T_{frame} は約 1.408msec になります。

$$Throughput\ (kbps) = floor\left(\frac{connection\ interval - T_{\mathrm{overhead}}}{T_{\mathrm{frame}}}\right) * 8 * L_{\mathrm{data}} * \frac{1}{connection\ interval}$$

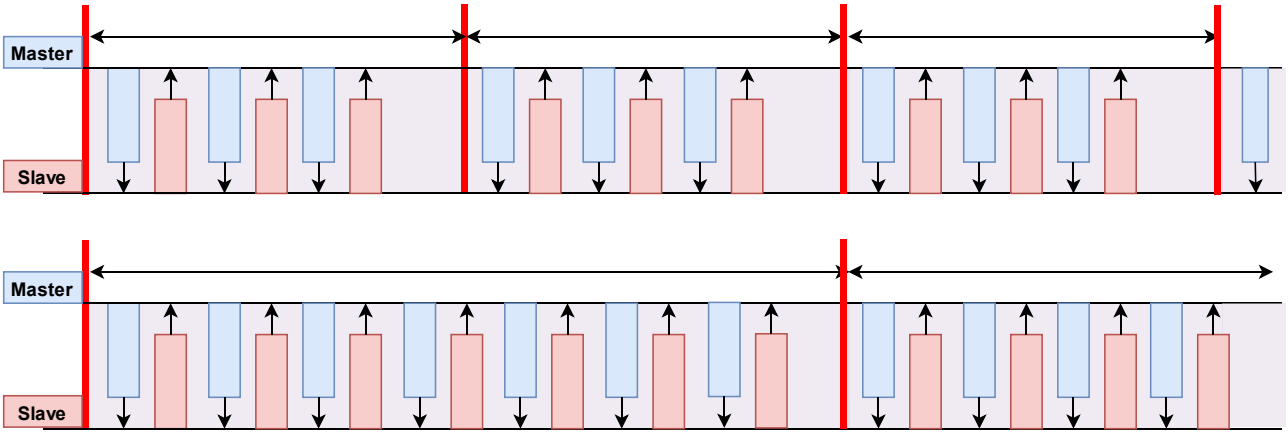

図 3.4 コネクションインターバルの変化とフレーム数

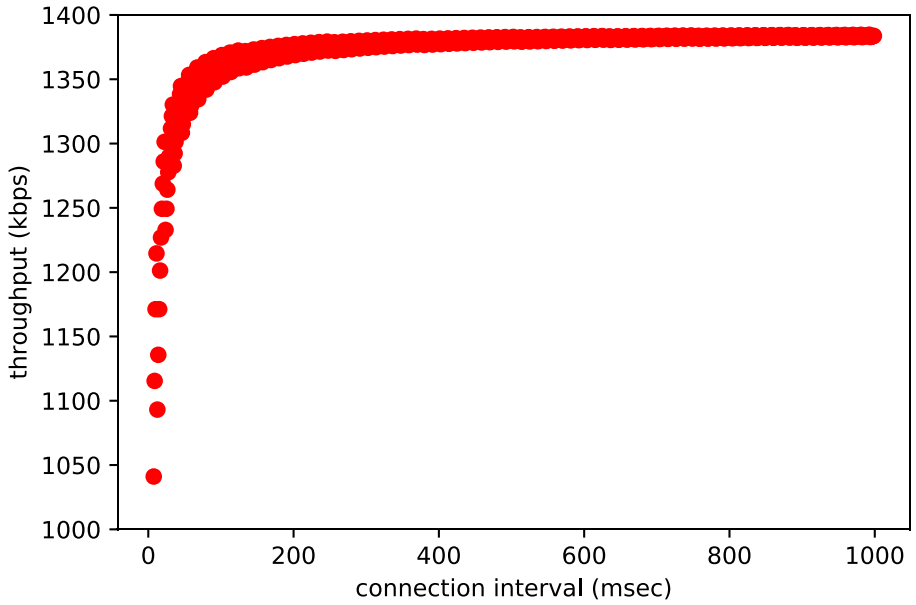

図 3.5 コネクションインターバルとスループットの関係

通信環境が良好の場合には、コネクションインターバルが長くなってもスループットに影響はありませんが、通信エラーにより More Data による通信が途切れる場合には、コネクションインターバルの違いが大きく影響します(図 3.6)。通信エラーが発生すると、互いのデバイスは次のコネクションイベントまで待機するため、コネクションインターバルが長いほどスループットが低下します。

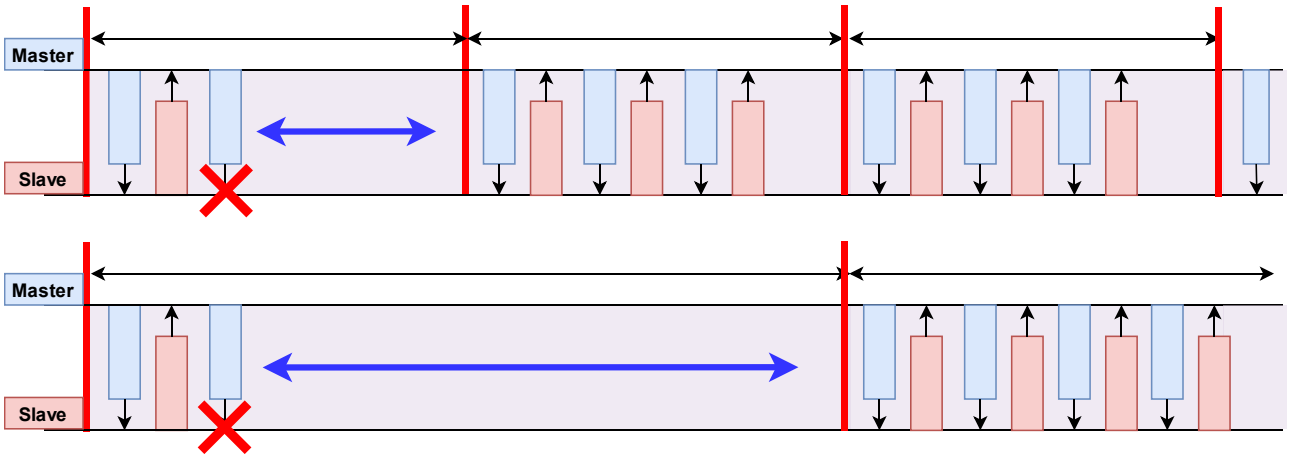

図 3.6 通信エラー発生時のコネクションインターバルとスループット

コネクションインターバルと、フレーム交換が失敗する確率とスループットの関係を図 3.7 に示します。GAP の設定は PHY:2M PHY、最大パケット長 251 byte, MTU 247byte で、244byte のアプリケーションデータを常に Notification している場合の値です。コネクションインターバルあたりのスループットの期待値をプロットしています。

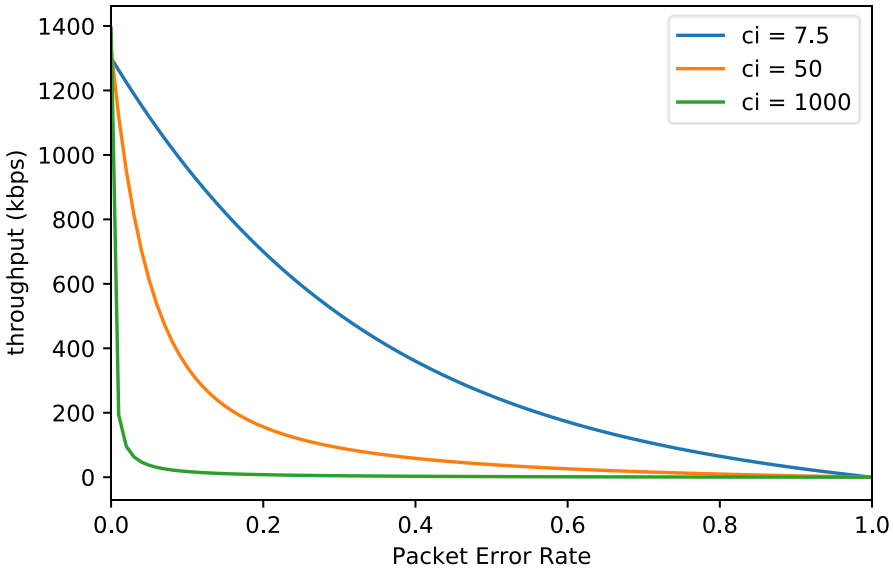

図 3.7 フレーム交換失敗確率とスループットの関係

コネクションインターバルの変更は、gap conn_cfg update コマンドを実行してください。R_BLE_API から変更する場合には、R_BLE_GAP_UpdConn を使用します。API についての詳細は「RX23W グループ BLE Module Firmware Integration Technology アプリケーションノート(R01AN4860)」に同梱されている「R_BLE API ドキュメント(r_ble_api_spec.chm)」を確認してください。

3.1.4　物理層 PHY の設定

物理層 PHY の設定を変更した場合の Link Layer の動作の模式図を図 3.8 に示します。物理層 PHY が変更されると、無線フレームの空中占有時間が変化します。データ長が同じ場合、1M PHY を基準に 2M PHY では空中占有時間が約半分になります。1 フレームの空中占有時間が短ければ単位時間当たりに送受信されるパケット数は増加するためスループットは向上します。

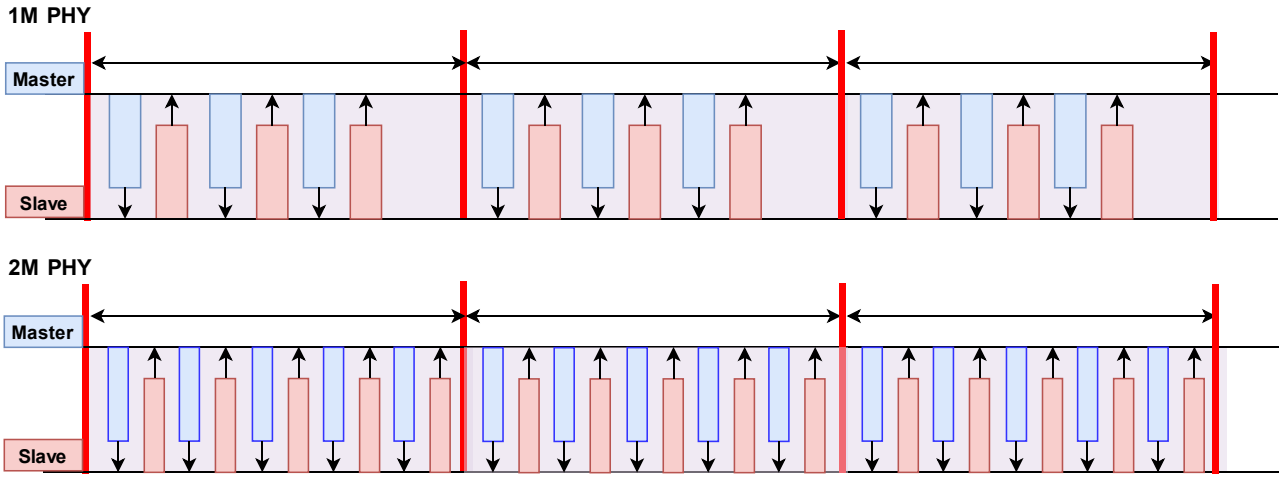

図 3.8 2M PHY 使用時の模式図

物理層の変更は gap conn_cfg phy コマンドを実行してください。R_BLE_API から変更する場合には、R_BLE_GAP_SetPhy を使用します。API についての詳細は「RX23W グループ BLE Module Firmware Integration Technology アプリケーションノート(R01AN4860)」に同梱されている「R_BLE API ドキュメント(r_ble_api_spec.chm)」を確認してください。

3.1.5　最大パケット長の設定

最大パケット長を大きく設定し、パケット長が大きいデータを送信した場合の Link Layer の動作の模式図を図 3.9 に示します。無線フレームのヘッダ情報と、送受信インターバルを最小にしてアプリケーションデータを効率よく送信することができます。

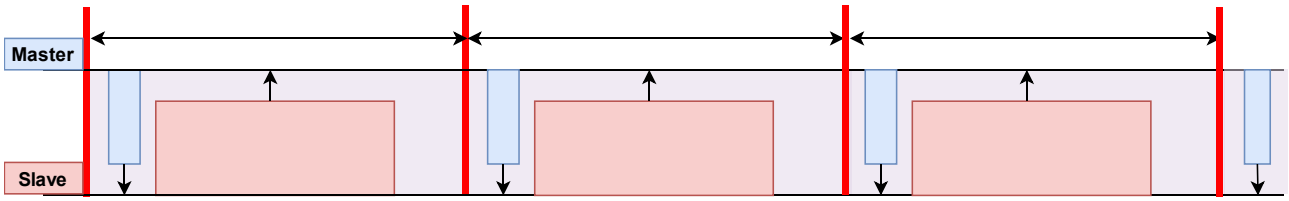

図 3.9 パケット長変更時の Link Layer 模式図

最大パケット長の変更は gap conn_cfg data_len コマンドを実行してください。R_BLE_API から変更する場合は R_BLE_GAP_SetDataLen を使用します。API についての詳細は「RX23W グループ BLE Module Firmware Integration Technology アプリケーションノート(R01AN4860)」に同梱されている「R_BLE API ドキュメント(r_ble_api_spec.chm)」を確認してください。

3.1.6　通信の暗号化の設定

通信を暗号化した場合の Link Layer の動作の模式図を図 3.10 に示します。暗号化によって、パケットの完全性をチェックするためのデータ(4bytes)が無線フレームに乗るため、スループットが低下することがあります。

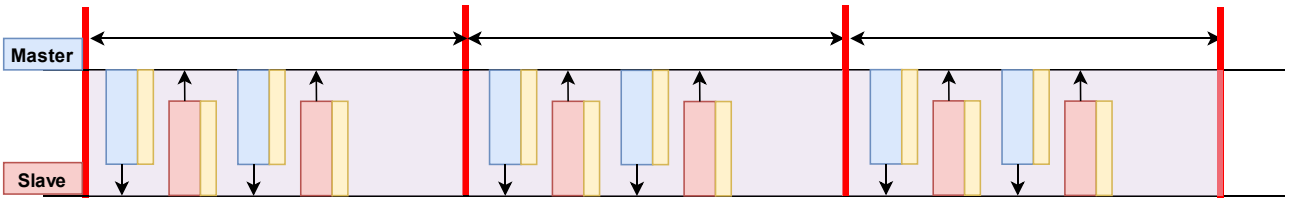

図 3.10 暗号化通信時の Link Layer の模式図

通信の暗号化は、gap auth start コマンドを実行してください。R_BLE_API から暗号化を行う場合は、R_BLE_GAP_StartEnc や R_BLE_GAP_StartPairing を使用します。app_lib の Abstraction API の R_BLE_ABS_StartAuth を使用することもできます。これらの API についての詳細は「RX23W グループ BLE Module Firmware Integration Technology アプリケーションノート(R01AN4860)」に同梱されている「R_BLE API ドキュメント(r_ble_api_spec.chm)」を確認してください。

3.2　Generic Attribute Profile (GATT)

Bluetooth Low Energy では、通信デバイスの検出と接続(アドバタイズとスキャニング、イニシエ―ティング)、接続後の継続的な通信を行うための情報は、GAP により管理されます。

一方で、アプリケーションデータ(センサーデータ等)の通信手順は Generic Attribute Profile (GATT)によって決められています。GATT は、GAP により確立された通信経路上でクライアント-サーバー方式のアーキテクチャを実現します。クライアントは、決められた手順を用いてサーバーが持つ GATT データベースに対してデータの読み書きします。この時、サーバーから、クライアントに対して応答が返されます。一方で、サーバーからクライアントに対して通知を行うことも可能です(図 3.11)。Bluetooth Low Energy のデータ通信を行うアプリケーションは全て GATT に従ってデータ通信を行います。

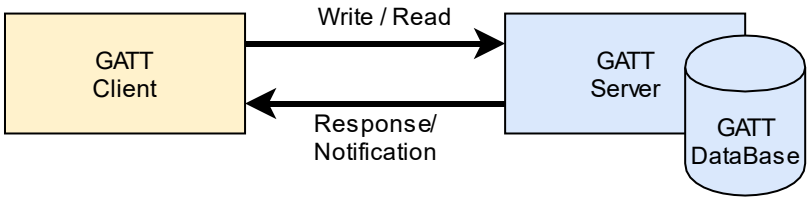

図 3.11 GATT 通信の基礎

GATT 通信では、アプリケーションの機能に注目して通信します。アプリケーションの機能のことを「サービス」、その機能に必要なデータのことを「キャラクタリスティック」と呼びます。アプリケーションを実現するために必要な機能(サービス)をまとめ、アプリケーションの通信仕様を定めたものが「プロファイル」です。

GATT 通信をする際は、アプリケーションとしての機能(サービス)と、その機能を実現するために必要なデータ(キャラクタリスティック)の情報を予め共有しておく必要があります。サーバーは自分が持つ「サービス」に関する情報と、そのサービスのキャラクタリスティックをデータベース上に保持します。

図 3.12 に体温計を例にした時のプロファイルとサービス、キャラクタリスティックの関係を示します。体温計アプリケーションの機能は体温測定とデバイス情報です。機能とデータは、サービスとキャラクタリスティックとして GATT データベース上に保持されます。

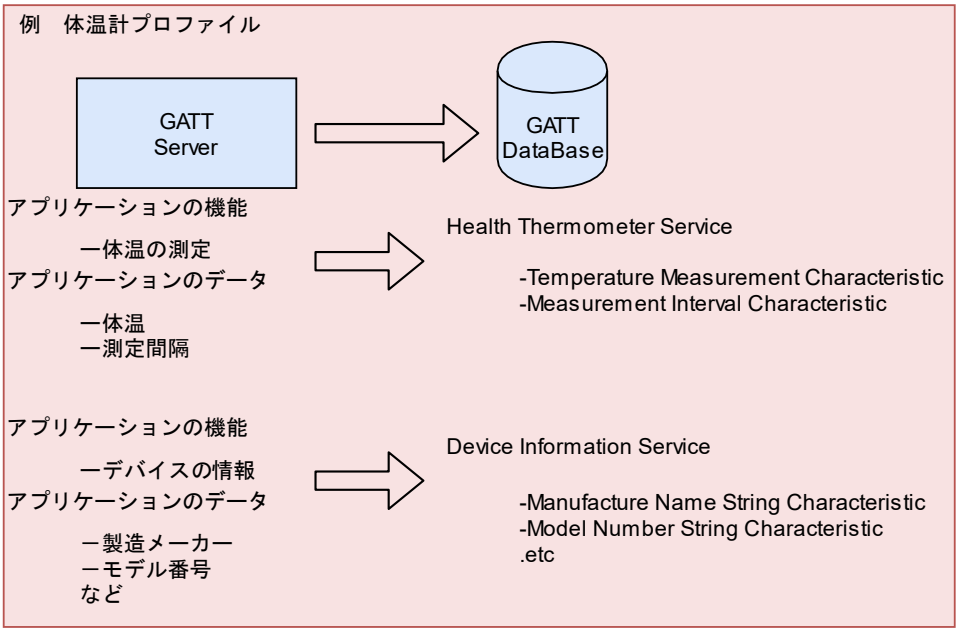

図 3.12 アプリケーションデータと GATT データベース

接続を確立しただけでは、クライアントは、サーバーのサービス情報をもちません。クライアントは、サービスディスカバリという手順を用いてサーバーに特定のサービスがあるか問い合わせます (図 3.13)。この手順によって、サーバーにクライアントが望むサービスの有無と、データベース上のデータのハンドル情報を得ます。クライアントは、このハンドル情報を利用してデータベースへの読み書きします。

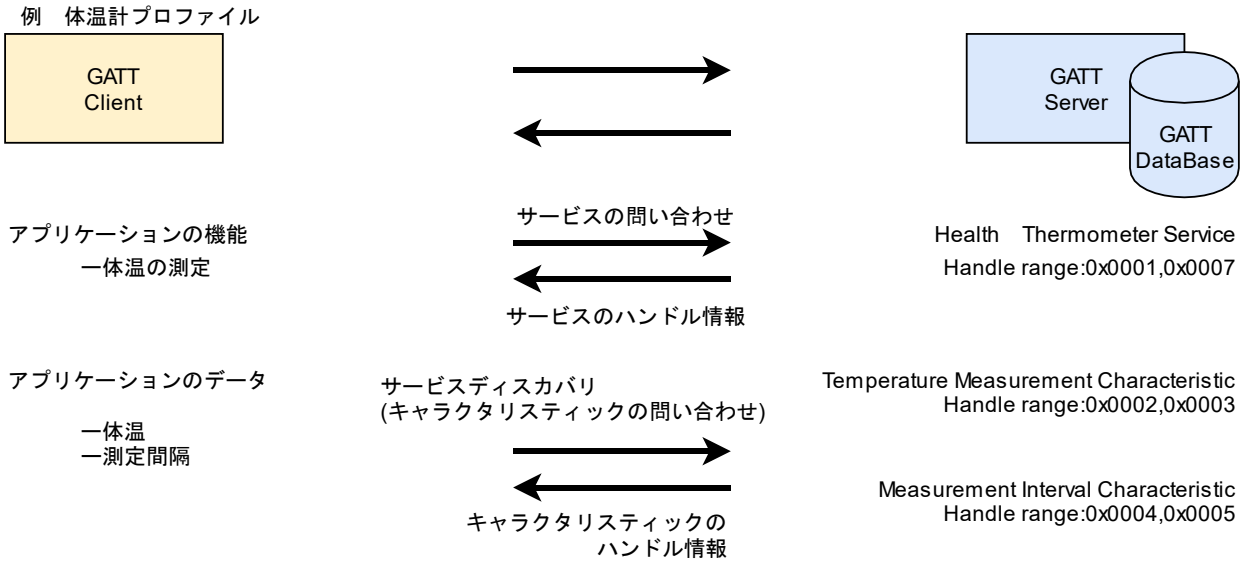

図 3.13 サービスディスカバリ動作

キャラクタリスティックは、データとその構造、及びクライアント-サーバー間でのデータのやり取りの手順が決められています。キャラクタリスティックで決められるデータと通信手順に対して、追加のオプションがある場合には、キャラクタリスティックディスクリプターに記述されます。

キャラクタリスティックデータの送受信は、キャラクタリスティックで決められた手順に従い実行されます。表 3.1 に代表的な手順をまとめます。これらの動作は、データ送信の向きと、相手からの応答を待つかどうかで特徴付けられます。応答が必要な手順では、相手から応答を受ける前に同じ手順を実行することはできません。クライアントがサーバーのデータを読み出す Read 動作の模式図を図 3.14 に示します。GATT によるデータ通信は、ハンドル情報に基づいて行われます。

表 3.1 GATT 通信の代表的な通信手順

手順名	動作	送信方向	応答
Read	読み取り	クライアント→サーバー	あり
Write	書き込み	クライアント→サーバー	あり
Write Without Response	書き込み	クライアント→サーバー	なし
Indication	通知	サーバー→クライアント	あり
Notification	通知	サーバー→クライアント	なし

例　体温計プロファイル　体温データのRead動作

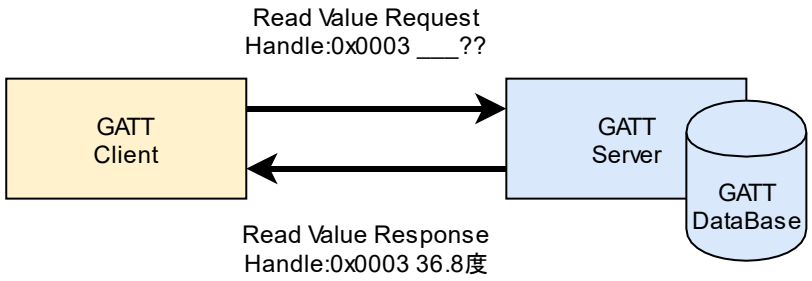

図 3.14 Read 動作

3.2.1　応答なし動作(Notification / Write Without Response)

Notification や Write Without Response 動作では、相手からの応答を待たずに次のパケットを送信することができます。そのため連続して送信要求を出すことで、More Data 通信を行うことができます。図 3.15 に、この時の Link Layer での Notification 動作の模式図を示します。

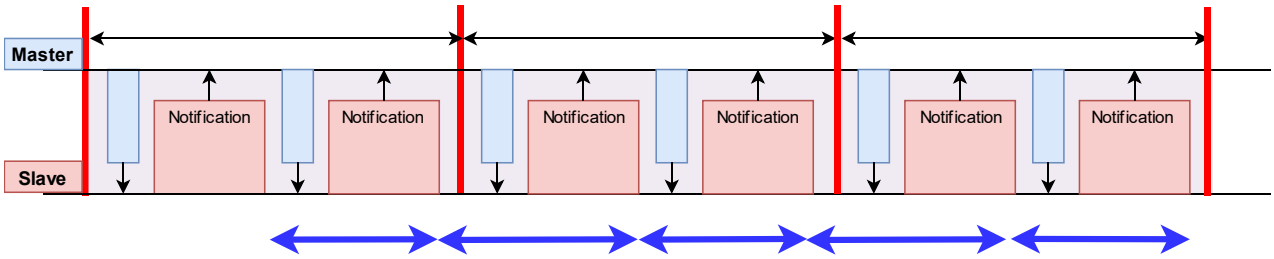

図 3.15 Notification 動作の Link Layer の模式図

上図では、スレーブがサーバーとして動作し Notification を行っています。スレーブやマスターは Link Layer でのロールであり、GATT ロールとの関係はありません。

3.2.2　応答あり動作(Indication / Write)

Indication と Write 動作では、送信後、次のデータを送信するためには相手からの応答を待つ必要があります。そのため、一つのコネクションインベントで次のデータの送信要求を出すことができず、More Data 通信を行うことができません。Indication の場合の Link Layer 動作の模式図を図 3.16 に示します。一つのデータパケットを送信するのにコネクションインターバルの 2 倍の時間がかかります。

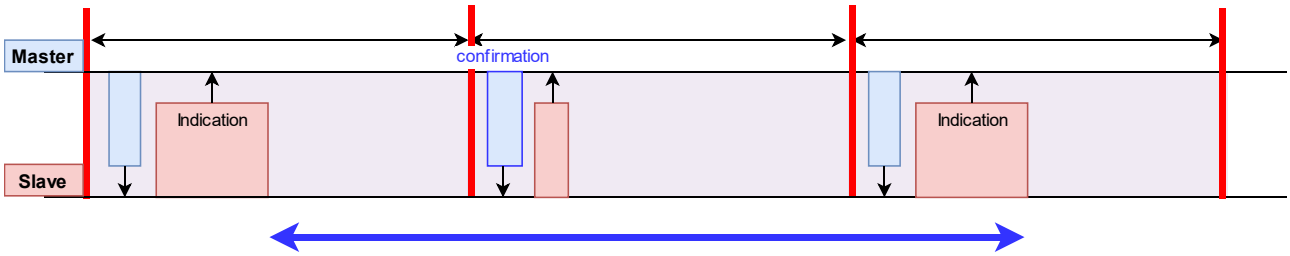

図 3.16 Indication 動作時の Link Layer の模式図

上図では、スレーブがサーバーとして動作し Indication を行っています。スレーブやマスターは Link Layer でのロールであり、GATT ロールとの関係はありません。

4. BLE ソリューションツール

本サンプルプログラムは、BLE FIT Module と QE for BLE を利用して作成されています。一般に、Bluetooth Low Energy アプリケーションを開発する際には、アプリケーションの開発に加えて、アプリケーションデータをやり取りするためのプロファイルを作成する必要があります。一部のアプリケーションでは既に Bluetooth SIG によってそのプロファイルが定義されています。

図 4.1 に BLE ソリューションツールと Bluetooth Low Energy アプリケーションの関係を示します。

BLE FIT Module は RX23W の Bluetooth Low Energy 機能を利用するための FIT モジュールです。BLE FIT モジュールは、Bluetooth Low Energy 通信を行うための R_BLE_API と、この API を使用してアプリケーション作成を補助する app_lib からなります。

QE for BLE は、GUI で GATT プロファイルの設計を行うツールです。コード生成機能によって設計したプロファイルを実現するためのアプリケーションとプロファイルのフレームワークを生成します。

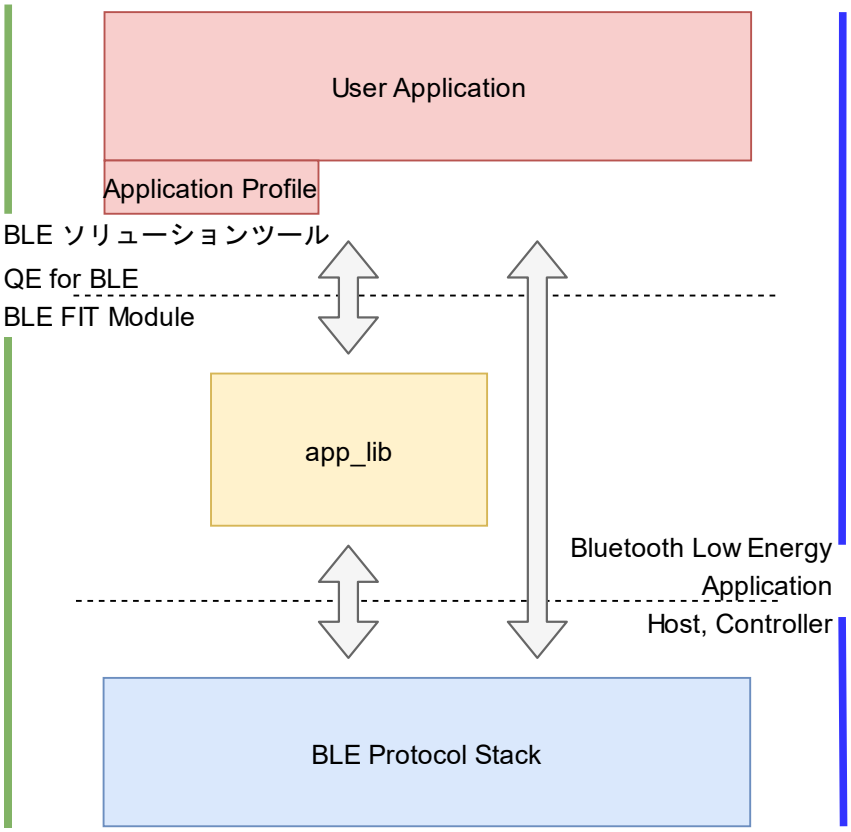

図 4.1 BLE ソリューションツールと Bluetooth Low Energy アプリケーションの構成

QE for BLE が生成するアプリケーションフレームワークと app_lib, R_BLE_API との関係を図 4.2 に示します。アプリケーションフレームワークは、設計した GATT プロファイルに関するプログラムと、Bluetooth Low Energy アプリケーションのための GAP の基本動作を行う部分を持ちます。

GAP に関する基本動作部分は、app_lib の Abstraction API を利用して実現されています。GATT プロファイルに関するプログラムは、生成されるサービス API と GATT データベースから、app_lib の Profile Common Library を通して通信を行います。

サービス API は、アプリケーションと GATT を繋ぐインタフェースです。Profile Common Library の GATT Discovery Library と Profile Common Server/Client Library は、アトリビュートハンドルやキャラクタリスティックのデータ構造などの GATT データベースの情報を抽象化します。アプリケーションは、サービス API を利用することで、GATT データベースを意識することなくアプリケーションデータの通信を行うことができます。

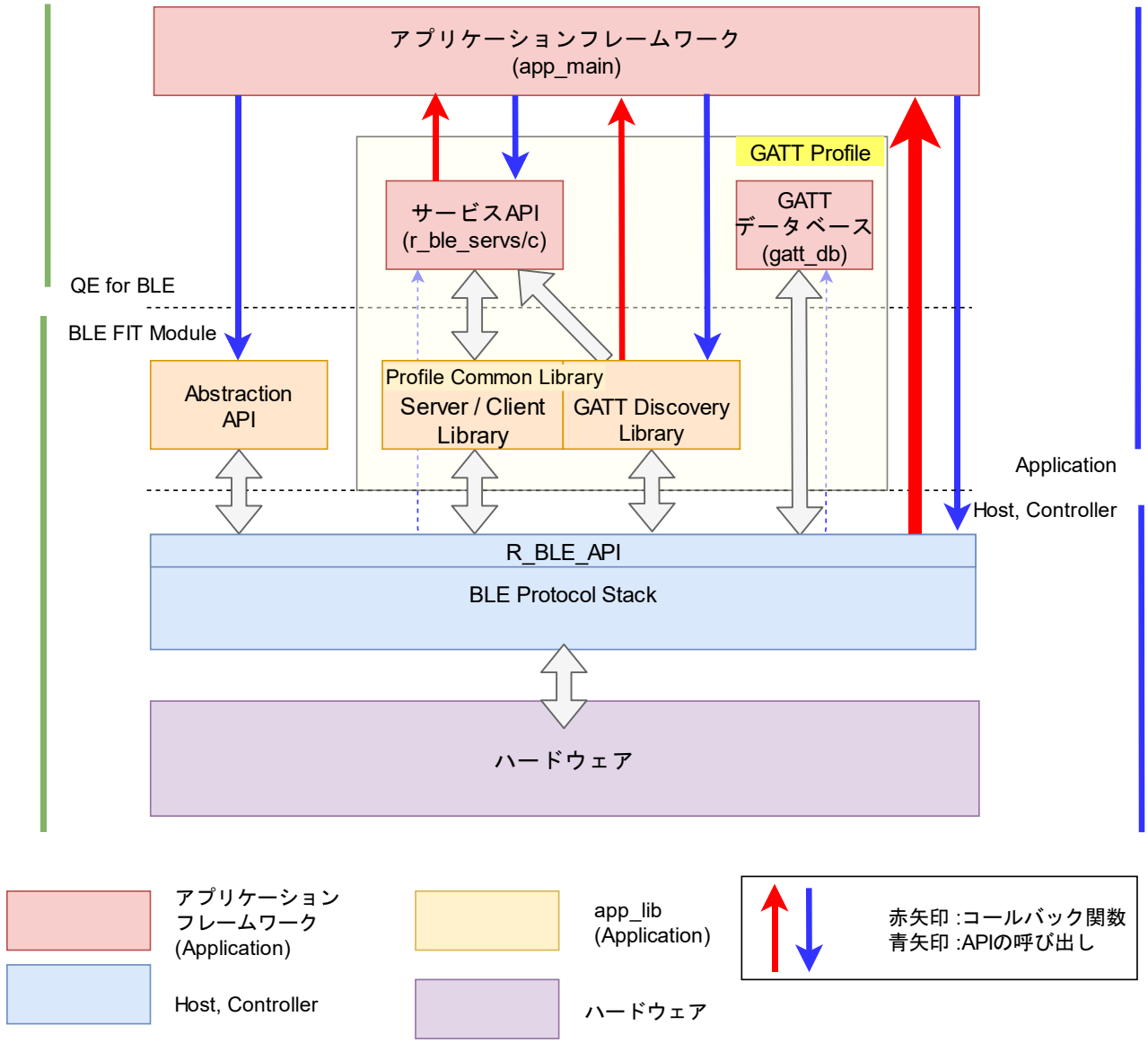

図 4.2 QE for BLE が生成するアプリケーションフレームワーク

app_lib 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、QE for BLE については「RX23W グループ Bluetooth Low Energy プロファイル開発者ガイド アプリケーションノート(R01AN4553)」をご覧ください。

4.1　QE for BLE によるプロファイル設計

サンプルプログラムで使用している独自(スループット)サービスを例に、QE for BLE を利用したプロファイル設計の方法を紹介します。スループットサービスは、Write/ Write Without Response を行うための Throughput Data 1 Characteristic と Notification/ Indication 動作のための Throughput Data2 Characteristic を持ちます(図 4.3 左側のタブ)。これらのキャラクタリスティックは、スループット測定のために送受信されるデータを表します。

この画面は、プロジェクトエクスプローラーから{project_name}.scfg ファイルを開きコンポーネントタブから[Config_BLE_Profile]を選択して開きます。

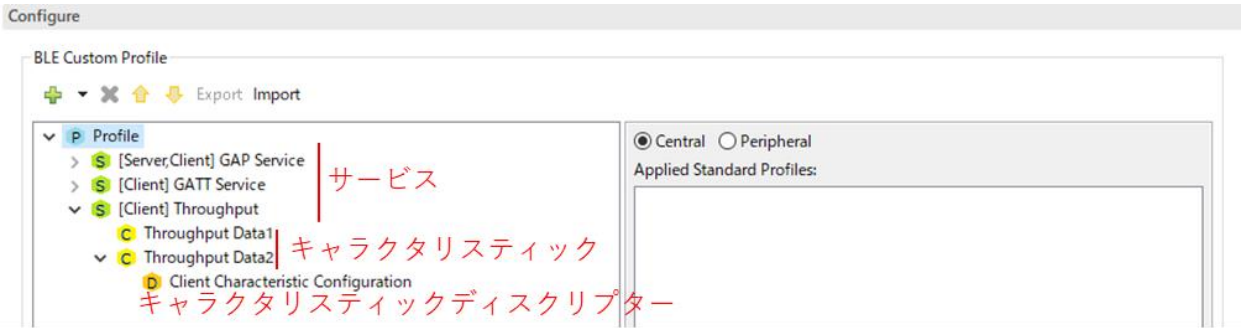

図 4.3 QE for BLE のプロファイル画面

図 4.4 に、スループットサービスの Throughput Data 1 Characteristic のクライアントの設定画面を示します。Name, UUID, Abbreviation を設定し、Properties の Write と Write Without Response にチェックを入れます。

Aux Properties は、GATT データベースでのデータ管理の設定を行います。スループットサービスでは設定する項目はありません。

DBsize と Value を設定します。DBsize は、GATT データベース上にデータを保存するために確保されるバイト数を設定します。クライアントは GATT データベースを持ちませんが、送受信データをホストスタックとやり取りする際に Profile Common Library によって確保されるバイト数を示します。Value は、キャラクタリスティックの初期値を表します。

最後に Fields を設定します。Field はアプリケーションで扱うキャラクタリスティックの構造体を定義します。このキャラクタリスティック構造体はサービス API に設定される encode/decode 関数によって、パケットデータ若しくは GATT データベース上のデータと相互に変換されます。

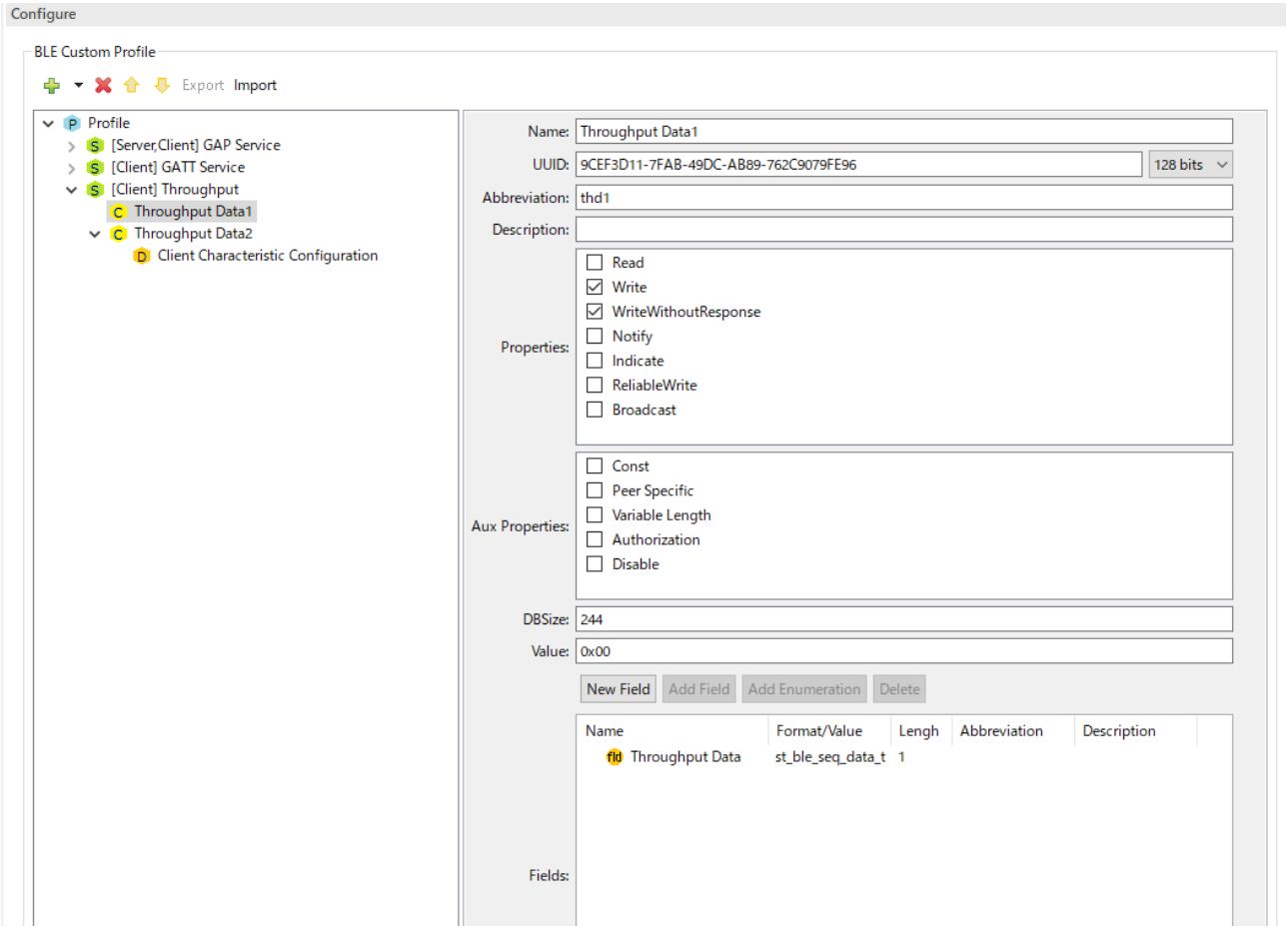

図 4.4 Throughput Data 1 Characteristic のキャラクタリスティック画面

サンプルプログラムのスループットサービスが持つキャラクタリスティックは、Profile Common Library が定義する st_ble_seq_data_t 構造体を使用します。この構造体は、配列データの先頭アドレスとデータ長を示すメンバを持っています。この構造体の encode/decode 関数は Profile Common Library に実装されています。そのため encode/decode 関数を新たに実装することなく配列データの送受信できます。

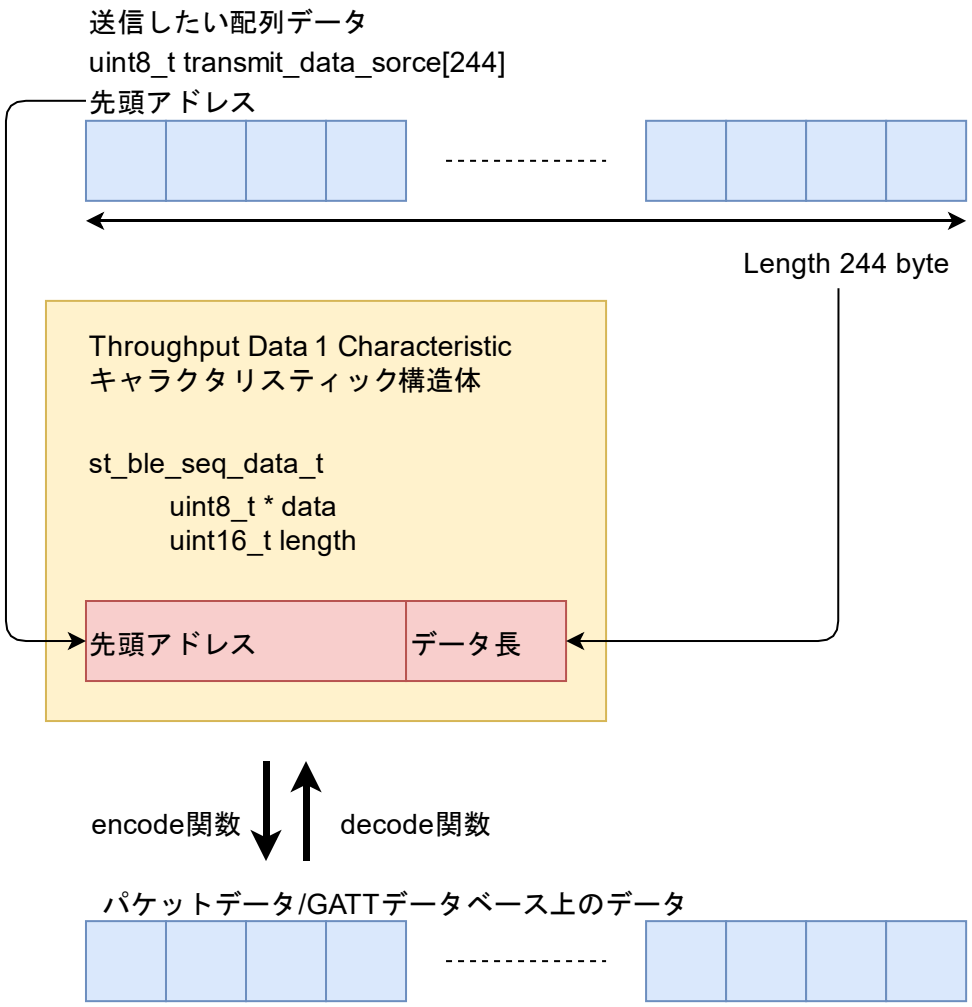

図 4.5 st_ble_seq_data_t 構造体を利用した配列データの送信

QE for BLE を使用したプロファイルの設計方法についての詳細は「RX23W グループ Bluetooth Low Energy プロファイル開発者ガイド アプリケーションノート(R01AN4553)」をご覧ください。

5.　付録

5.1　ターゲットボードへのプログラムファイルの書き込み方法

ターゲットボードにはエミュレータ回路が搭載されているため、USB ケーブルを接続してプログラムを書き込むことができます。

以下に Renesas Flash Programmer (RFP)を使用してターゲットボードにプログラムファイルの書き込み方法を示します。

1. ボード左上のエミュレータ端子に USB を接続し、エミュレータリセットスイッチの 2 を ON に変更し USB ケーブルで PC と接続します。

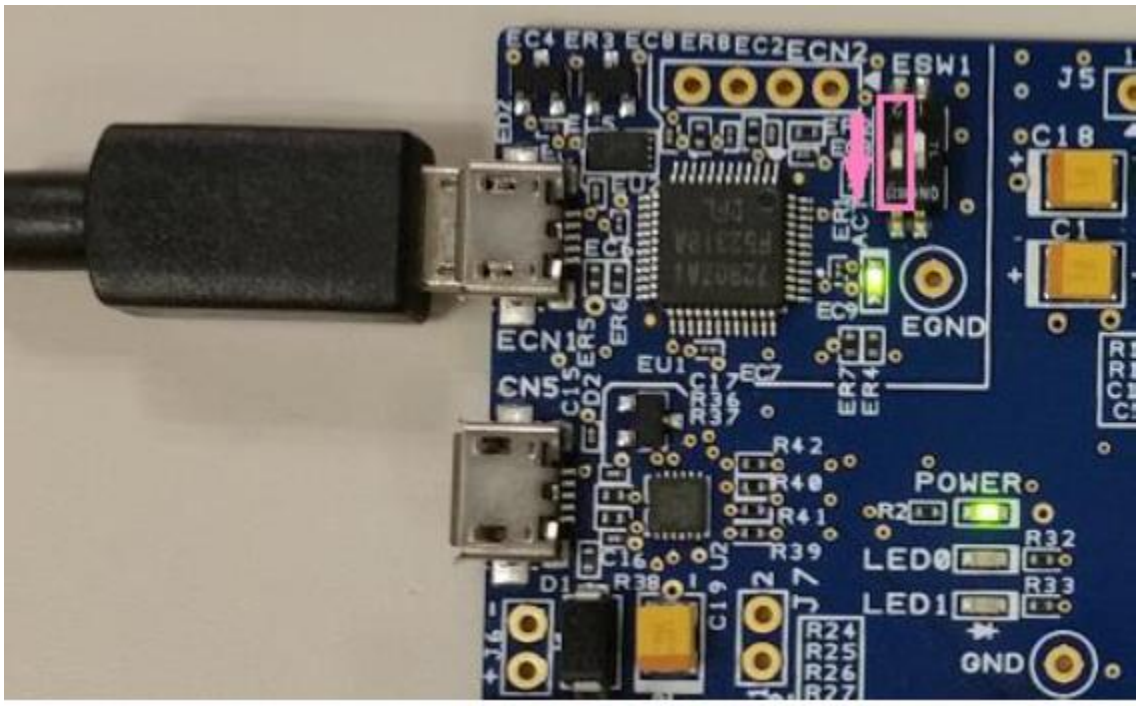

図 5.1 プログラムの書き込み

2. RFP を起動し、[ファイル]→[新しいプロジェクトを作成]を選択します。

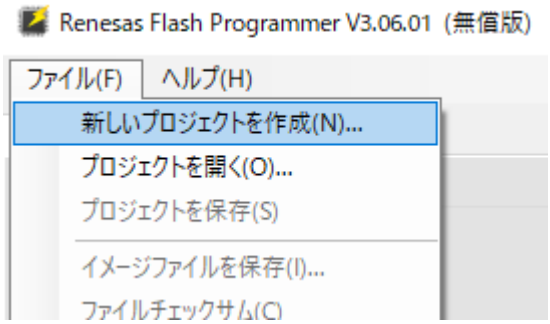

図 5.2 新規プロジェクトの作成

3. [新しいプロジェクトの作成]ウインドウで、以下の設定を行い[接続]ボタンをクリックします。
   - マイクロコントローラー：RX200
   - プロジェクト名：任意のプロジェクト名
   - 作成場所：任意のフォルダを選択
   - 通信　ツール：”E2 emulator Lite”を選択、インタフェース：”FINE”
   - 電源：”供給しない” (デフォルト)

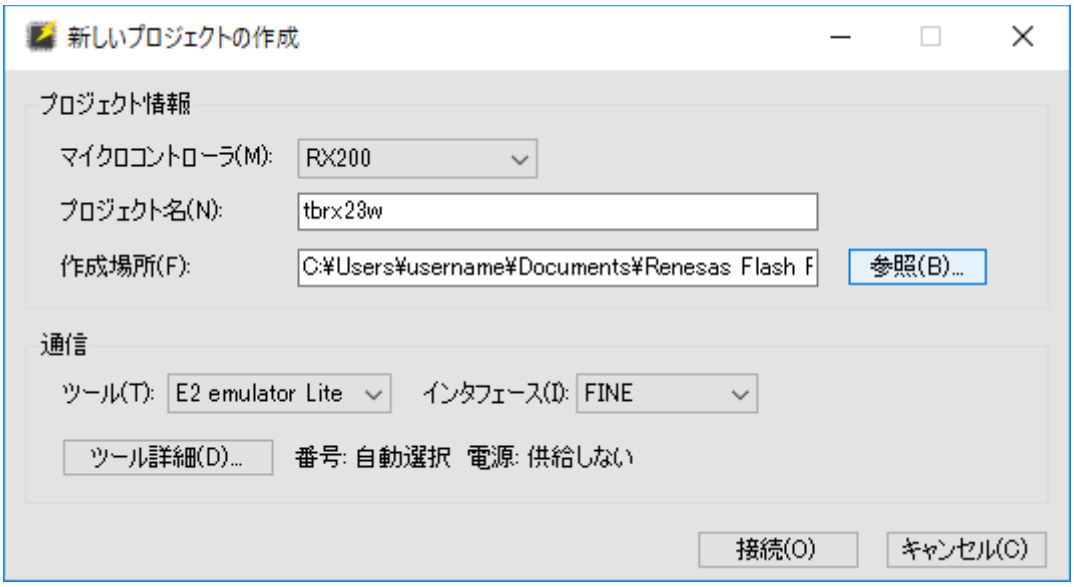

図 5.3 プロジェクトの設定

4. [ID コードの設定]の入力を求められます。出荷時には、”45FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF” が設定されています。入力し[OK]ボタンを押します。

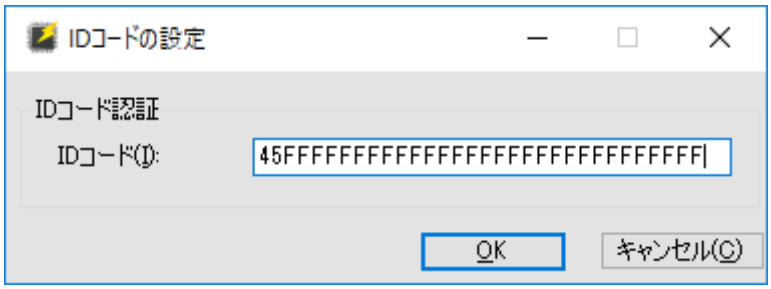

図 5.4 ID コード設定

5. 接続に成功すると、「**操作が成功しました。**」と表示されます。

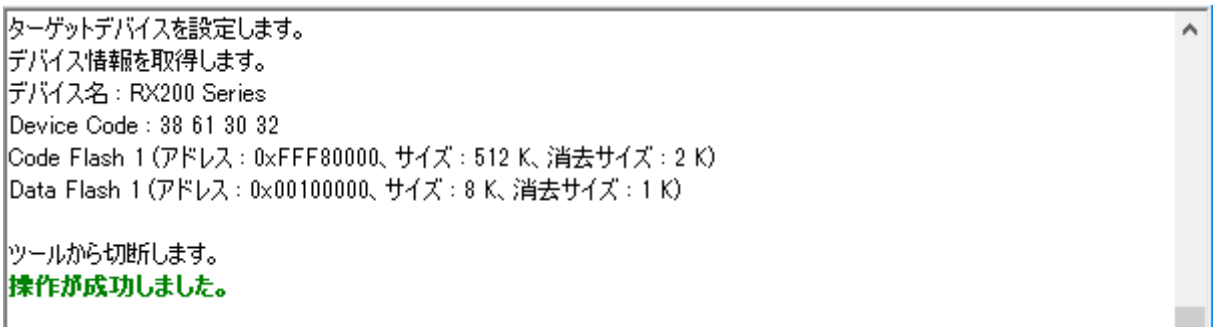

図 5.5 接続成功画面

6. [参照]ボタンをクリックし、アプリケーションノートのフォルダ内の mot フォルダの書き込む mot ファイルを選択し、[開く]ボタンをクリックします。書き込むプログラムを選択後、 [操作]タブの[スタート]ボタンをクリックし、プログラムファイルの書き込みを開始します。

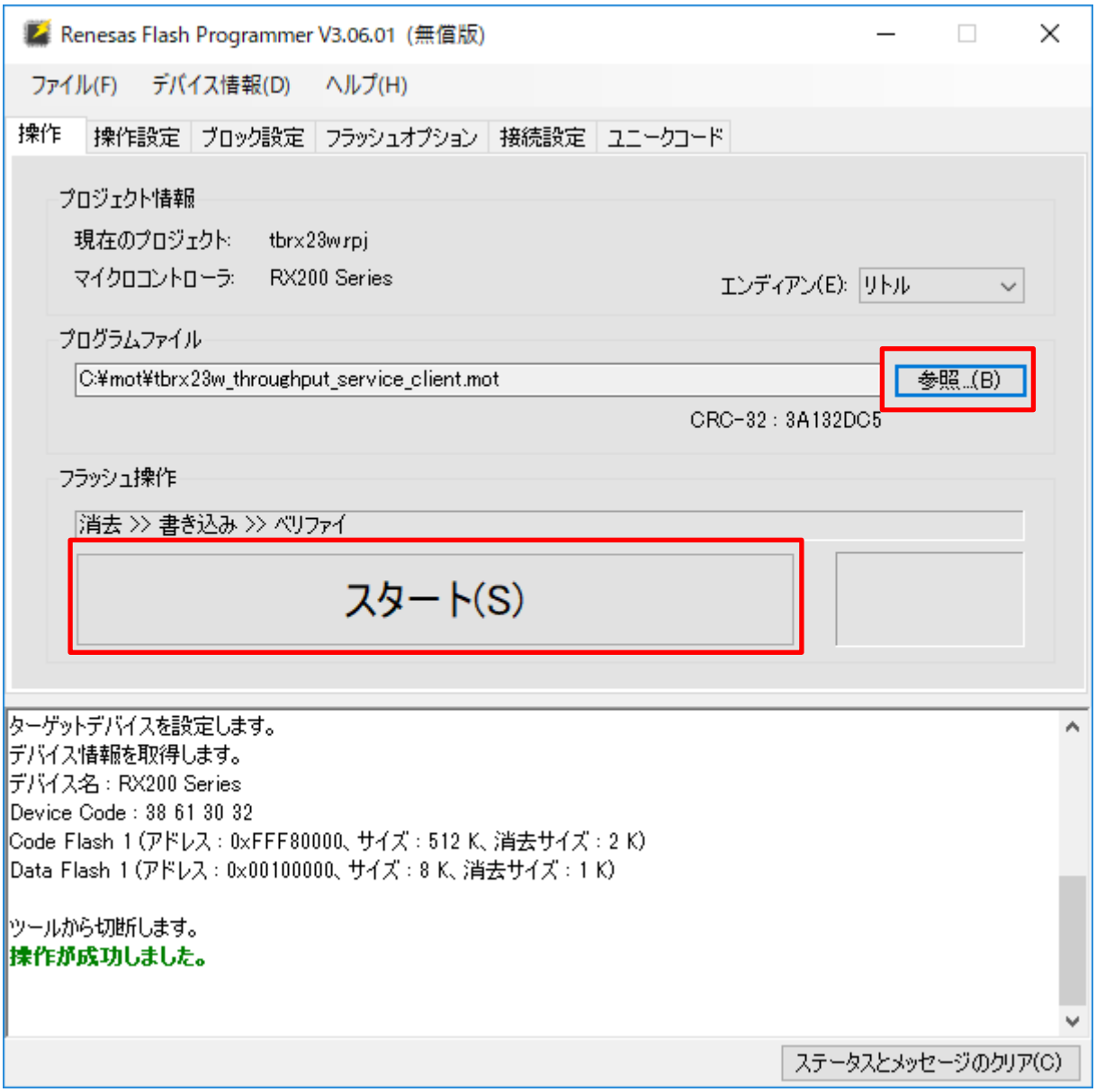

図 5.6 mot ファイルの選択とスタート

7. 書き込みが正常に終了すると、「操作が成功しました。」および「正常終了」と表示されます。

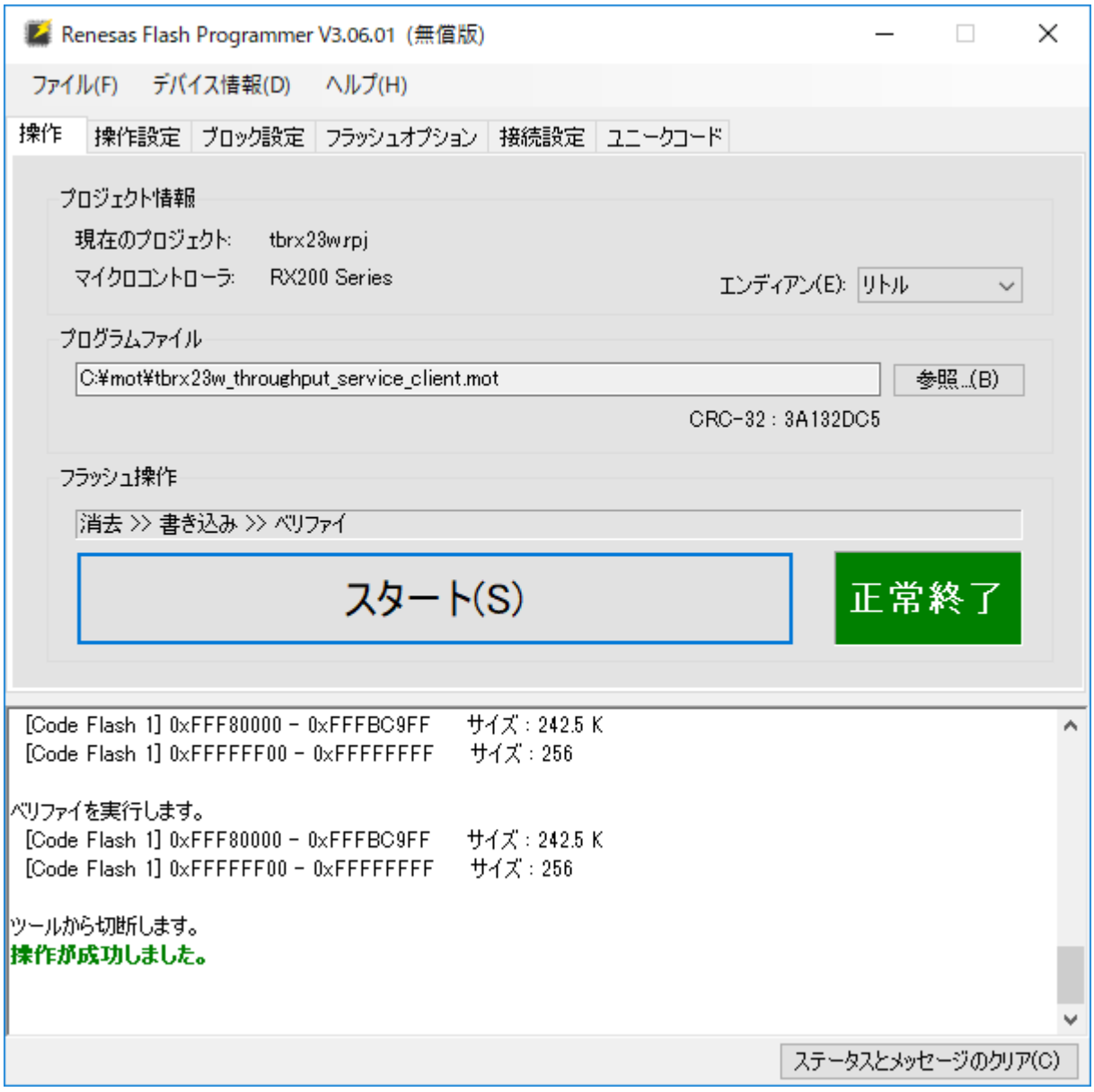

図 5.7 プログラムの書き込み終了

5.2　e²studio のワークスペースへのプロジェクト追加方法

サンプルプログラムのプロジェクトを e²studio のワークスペースへ追加する方法を説明します。

1.　[ファイル]メニューから[インポート]を選択します。

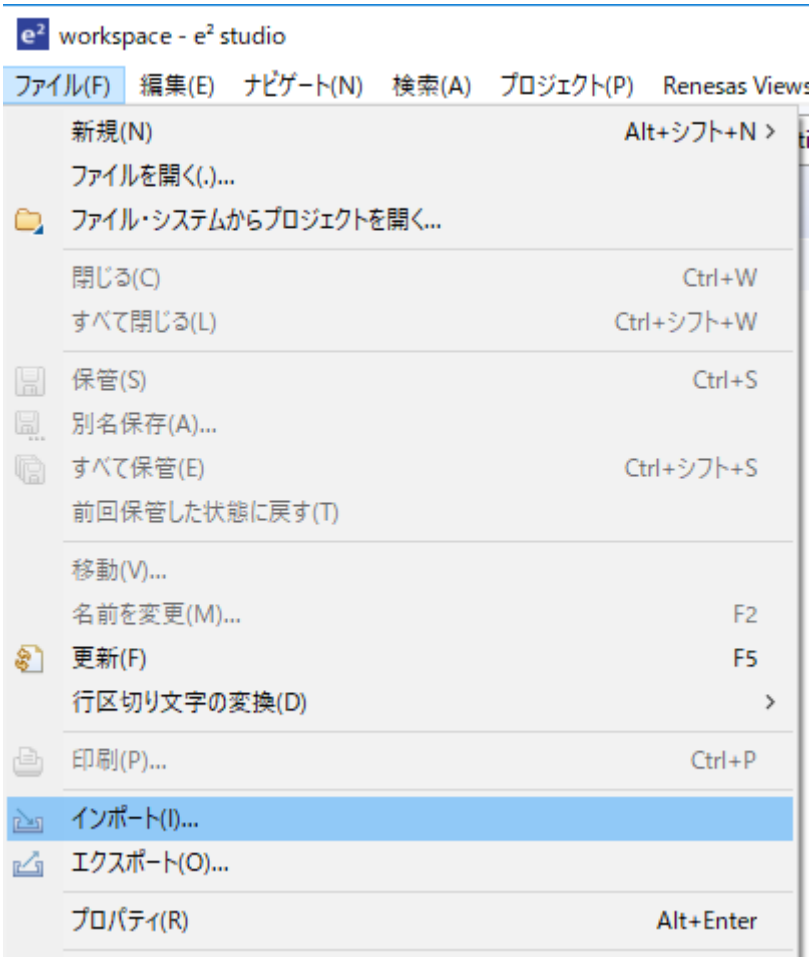

図 5.8 ファイルメニュー

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

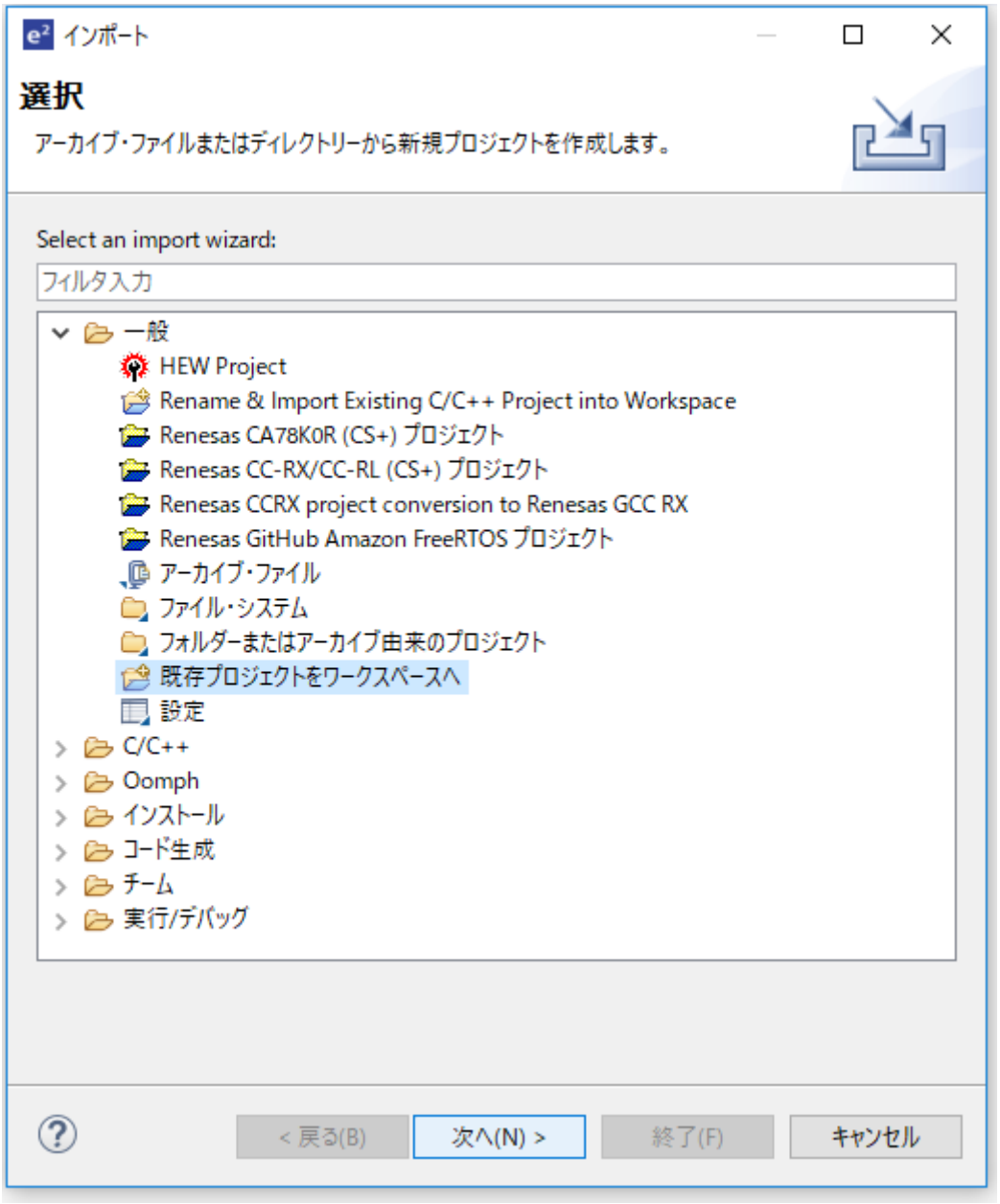

図 5.9 インポートの選択画面

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、追加するプロジェクトの zip ファイルを選択します。[終了]ボタンをクリックすると、選択したプロジェクトがインポートされます。

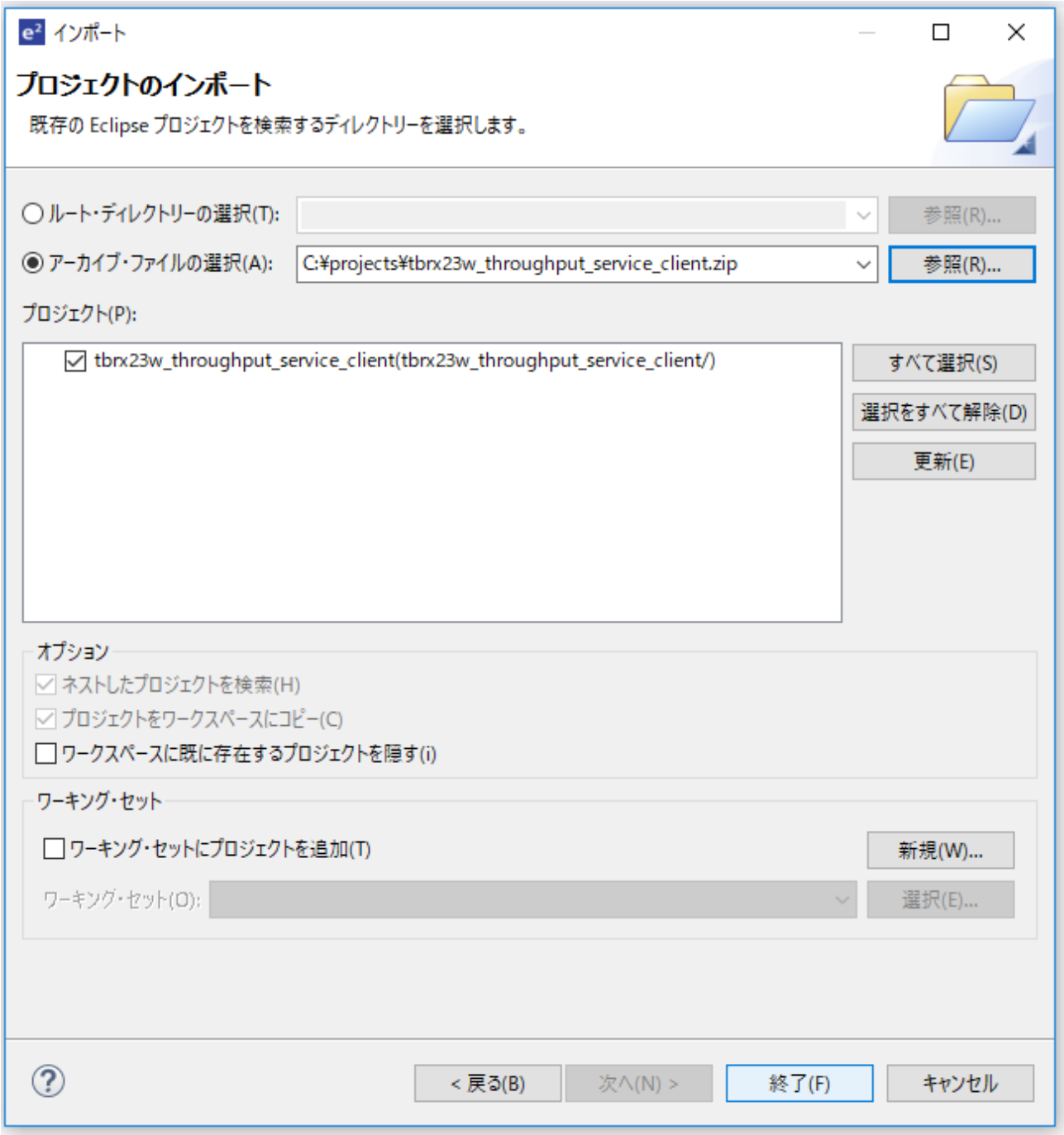

図 5.10 インポートするプロジェクトの選択画面

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2020/5/14		初版発行

製品ご使用上の注意事項

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

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

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。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みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、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）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
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といたします。
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。