

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

R01AN1375JJ0120
Rev.1.20
2017.7.31

要旨

このマニュアルは、Bluetooth Low Energy ソフトウェア（BLE ソフトウェア）に含まれるサンプルプログラムのインストール、構成、使用方法について記載しています。

BLE ソフトウェアは、Bluetooth Low Energy 仕様（Bluetooth 仕様 v4.2）に準拠した Bluetooth Low Energy プロトコルスタック（BLE プロトコルスタック）を含むソフトウェア一式です。BLE プロトコルスタックは、Bluetooth Low Energy マイコン RL78/G1D 上で動作するように設計されています。

対象デバイス

RL78/G1D

目次

1. 概要	4
2. 適用	4
3. インストール	4
3.1 内容物	4
3.2 インストール手順	4
4. サンプルプログラムの構成	5
4.1 動作環境と開発環境	5
4.2 構成	6
5. コンソール入出力サンプルプログラムの使用方法	8
5.1 パラメータ変更方法	8
5.2 Modem 構成で使用する際の起動方法	9
5.3 Embedded 構成で使用する際の起動方法	9
5.4 コンソール入出力サンプルプログラムの使用方法	10
5.5 Generic Access Profile (GAP)	12
5.6 Security Manager (SM)	13
5.7 Generic Attribute Profile (GATT)	16
5.8 Find Me Profile (FMP)	18
5.9 Proximity Profile (PXP)	19
5.10 Health Thermometer Profile (HTP)	22
5.11 Blood Pressure Profile (BLP)	24
5.12 HID over GATT Profile (HOGP)	26
5.13 Scan Parameters Profile (ScPP)	29
5.14 Heart Rate Profile (HRP)	31
5.15 Cycling Speed and Cadence Profile (CSCP)	35

5.16 Cycling Power Profile (CPP)	39
5.17 Alert Notification Profile (ANP)	43
5.18 Location and Navigation Profile (LNP)	46
5.19 Vendor Specific (VS)	50
 6. 簡易サンプルプログラムの使用方法	52
6.1 構成	52
6.2 HEX ファイルの入手	52
6.3 動作概要	52
6.4 Android デバイスでの確認方法	53
6.5 iOS デバイスでの確認方法	55
 7. 付録	57
7.1 Windows 向けサンプルプログラムによる BLE-MCU への送受信動作	57
7.2 APP MCU のシリアル通信ドライバの要件と実装フローチャート	60
7.2.1 UART 2 線接続方式の送信実装例	62
7.2.2 UART 2 線接続方式の受信実装例	62
7.2.3 UART 3 線接続方式の送信実装例	63
7.2.4 UART 2 線分岐接続方式の送信実装例	64
7.2.5 UART 3 線接続方式、2 線分岐接続方式の受信実装例	65
7.2.6 CSI 4 線接続方式の送信実装例	66
7.2.7 CSI 5 線接続方式の送信実装例	67
7.2.8 CSI の受信実装例	68
7.2.9 IIC 3 線接続方式の送信実装例	69
7.2.10 IIC 3 線接続方式の受信実装例	69
7.3 APP MCU 向け組み込みサンプルプログラム	71
7.4 Direct Test Mode の使用方法	73
7.4.1 Direct Test Mode (Receiver)	74
7.4.2 Direct Test Mode (Transmitter)	75
7.4.3 Direct Test Mode (Parameter Set)	76
7.5 Sample Custom Profile	78
7.5.1 Sample Custom Profile 仕様	78
7.5.2 Sample Custom Profile ファイル構成	79
7.5.3 Sample Custom Profile IF 関数仕様	80
7.5.4 Sample Custom Profile EVENT 仕様	84
7.5.5 Sample Custom Profile サンプルプログラム制御方法	86
7.6 簡易サンプルプロファイル	89
7.6.1 Characteristic 仕様	89
7.6.2 簡易サンプルプロファイルのファイル構成	89
7.6.3 Simple Sample Profile の詳細	89
7.7 RF テスターによる Direct Test Mode サンプルプログラム	90
7.8 Embedded 構成の printf プログラム	93
7.9 FW アップデートサンプルプログラム	94
7.9.1 FW アップデートプロファイル仕様	94
7.9.2 FW アップデートサンプルプログラムファイル構成	95
7.9.3 FW アップデートプロファイル IF 関数仕様	97
7.9.4 FW アップデートプロファイル EVENT 仕様	101

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

7.9.5 FW アップデートサンプルプログラム制御方法	102
7.10 FW アップデートサンプルプログラムを使用するためのプロジェクト設定方法	105
7.10.1 Receiver デバイス	105
7.10.2 Sender デバイス	109
7.10.3 FW アップデート環境を作成する際の注意事項	110
7.11 参考文献	111
7.12 用語説明	112

1. 概要

このマニュアルは、Bluetooth Low Energy ソフトウェア（BLE ソフトウェア）に含まれるサンプルプログラムのインストール、構成、使用方法について記載しています。

BLE ソフトウェアは、Bluetooth Low Energy 仕様（Bluetooth 仕様 v4.2）に準拠した Bluetooth Low Energy プロトコルスタック（BLE プロトコルスタック）を含むソフトウェア一式です。BLE プロトコルスタックは、Bluetooth Low Energy マイコン RL78/G1D 上で動作するように設計されています。

BLE プロトコルスタックの API の詳細につきましては、Bluetooth Low Energy プロトコルスタック API リファレンスマニュアルを参照してください。

2. 適用

このマニュアルの記載内容は、BLE プロトコルスタック Version1.20 以降に適用します。

3. インストール

サンプルプログラムは BLE プロトコルスタックの zip 圧縮パッケージに同梱されています。

3.1 内容物

BLE プロトコルスタックパッケージには、以下に示すものが含まれています。

- ドキュメント
 - Bluetooth Low Energy プロトコルスタックユーザーズマニュアル
 - Bluetooth Low Energy プロトコルスタック API リファレンスマニュアル
 - Bluetooth Low Energy プロトコルスタックサンプルプログラムアプリケーションノート（本書）
 - rBLE コマンド仕様書
- 実行ファイル作成用ファイル一式
 - 実行ファイル
 - BLE ソフトウェアライブラリ
 - サンプルプログラムのソースファイル
 - 各種パラメータ設定用ソースファイル
 - CS+ for CA, CX 用プロジェクトファイル
 - CS+ for CC 用プロジェクトファイル
 - IAR Embedded Workbench 用ワークスペースファイル
 - e² studio 用プロジェクトファイル
- PC 用サンプルプログラム一式
 - 実行ファイル
 - サンプルプログラムのソースファイル
 - Microsoft Visual Studio 2015 Express for Desktop 用プロジェクトファイル
- HCI パケットモニタ PC 用アプリケーション一式
 - 実行ファイル
 - INI ファイル

3.2 インストール手順

パッケージを解凍して任意のフォルダにコピーしてください。

【注】e² studio をご使用になる場合はフォルダパスにマルチバイト文字(全角文字)およびブランクを含まない場所にコピーしてください。

4. サンプルプログラムの構成

サンプルプログラムは、BLE ソフトウェアの使用方法を示すサンプルプログラムです。BLE ソフトウェアは、以下の 2 つのサンプルプログラムを含みます。

- コンソール入出力サンプルプログラム 5章に記載
- 簡易サンプルプログラム 6章に記載

本章では、サンプルプログラムに共通する内容に関して記載します。各サンプルプログラムの詳細は、それぞれの章を参照してください。

【注】 本アプリケーションノートに掲載されている各サンプルプログラムはサンプル提供となります。量産でお使いになる場合、お客様の責任で動作確認の上、ご使用ください。

4.1 動作環境と開発環境

BLE ソフトウェアは、二種類の異なるシステム構成、Modem 構成と Embedded 構成、に対応しています。ここでは、それぞれのシステム構成におけるサンプルプログラムの開発環境、動作環境について説明します。

● Modem 構成

Modem 構成では、コントローラスタック、ホストスタック、プロファイルが BLE MCU (RL78/G1D) 上に実装され、アプリケーションは、別の APP MCU 上に実装されます。

BLE ソフトウェアは、パソコンを APP MCU として動作するサンプルプログラムを提供します。パソコンを用いて BLE ソフトウェアを簡単に評価することができます。

Modem 構成のサンプルプログラムは、次の環境で動作します。

— ハードウェア

- PC/AT™ 互換機
- プロセッサ : 1.6GHz 以上
- メモリ : 1.0GB 以上
- ディスプレイ : 1024×768(XGA)以上の解像度, 65536 色以上
- インタフェース : USB2.0(E1 および USB-シリアル変換ケーブル)

— ソフトウェア

- Windows 7 以降
- Microsoft Visual Studio Express 2015for Desktop
- Microsoft .NET Framework 4+言語パック

● Embedded 構成時の実行環境

Embedded 構成では、コントローラスタック、ホストスタック、プロファイル、アプリケーションが BLE MCU (RL78/G1D) 上に実装されます。

BLE ソフトウェアは、BLE MCU 上で動作するサンプルプログラムも提供します。

Embedded 構成は、開発環境に CS+ for CA, CX、CS+ for CC、IAR Embedded Workbench、e² studio のいずれかを使用し、以下の実行環境で動作します。

— ハードウェア環境

- RL78/G1D 用評価ボード

— 使用ツール

- Renesas オンチップデバッグエミュレータ E1
- Windows マシン用ターミナルソフト

— ソフトウェア環境

- Renesas 製統合開発環境 CS+ for CA, CX、CS+ for CC、e² studio
もしくは IAR Systems 製 IAR Embedded Workbench
- Renesas Flash Programmer V3 無償版
(<https://www.renesas.com/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html>)より入手可能)

4.2 構成

図 4-1に BLE ソフトウェアの構成図を示します。

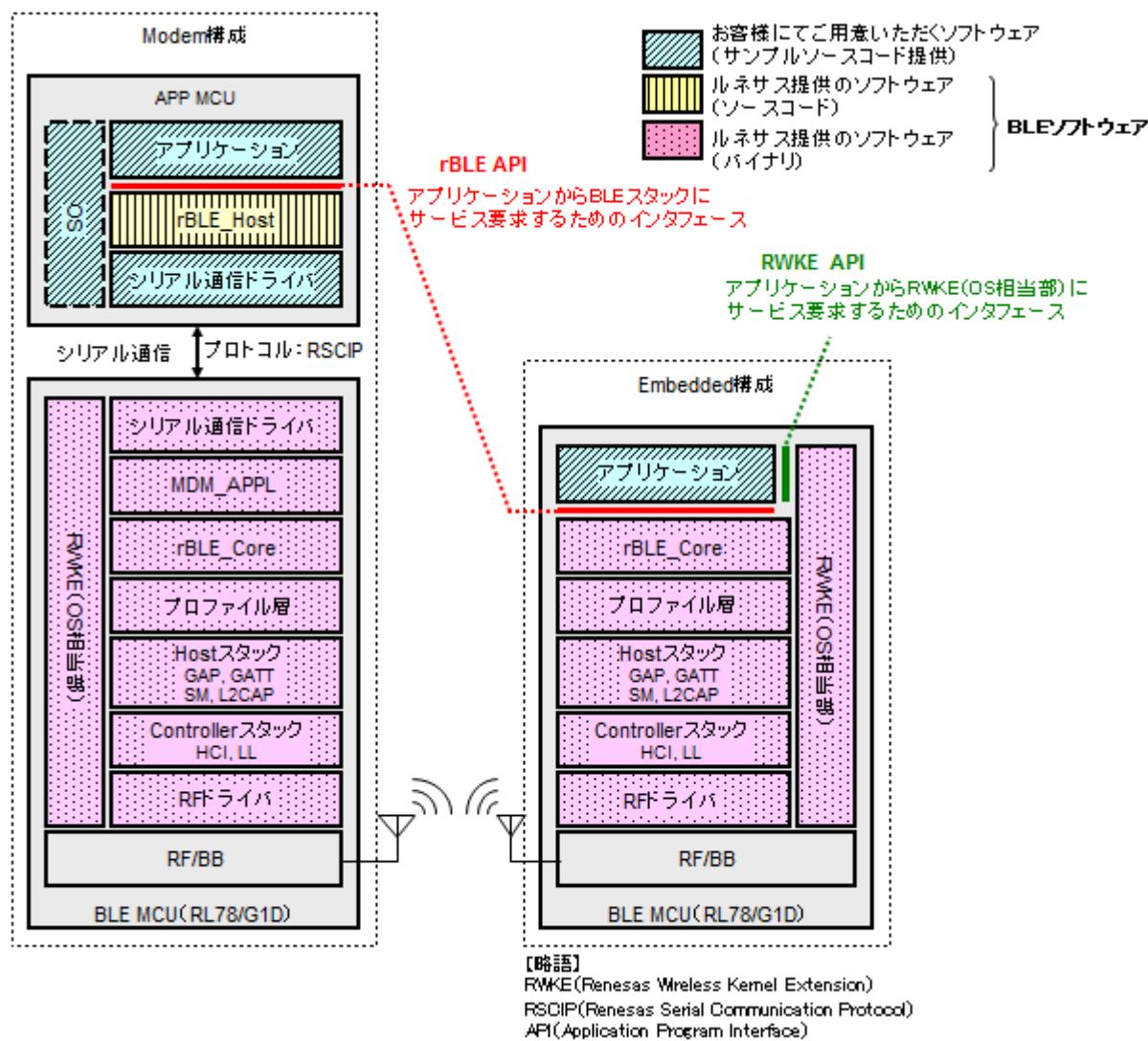


図 4-1 BLE ソフトウェアの構成図

Modem 構成時の BLE ソフトウェアは、APP MCU と BLE MCU(RL78/G1D)の 2 つのチップで動作し、APP MCU で動作する「rBLE_Host」部（図の ブロック）と BLE MCU で動作するソフトウェア（図の ブロック）で構成されます。

また、お客様にてご用意いただくソフトウェア（図の ブロック）は、APP MCU の「アプリケーション」部と「シリアル通信ドライバ」部、および「OS」部になります。ただし、「rBLE_Host」部は OS のリソ-

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

スを使用していないため、APP MCU に OS が搭載されていない場合には、「OS」部のソフトウェアを用意する必要はありません。

一方、Embedded 構成時の BLE ソフトウェアは、BLE MCU(RL78/G1D)のみの 1 チップで動作します。お客様にてご用意いただくソフトウェアは、「アプリケーション」部のみとなり、BLE MCU 上に実装されます。

5. コンソール入出力サンプルプログラムの使用方法

5.1 パラメータ変更方法

コンソール入出力サンプルプログラムには、rBLE API呼び出し時のパラメータを変更する機能があり、予め用意しておいたパラメータを選択して実行することができます。

パラメータの選択は下記のように入力します。

メニュー番号[スペース]パラメータ番号

メニュー実行時に呼び出される関数において、入力されたパラメータ番号によって rBLE API 呼び出し時のパラメータを変更しています。

5.2 Modem 構成で使用する際の起動方法

Modem 構成のコンソール入出力サンプルプログラムは、パッケージインストール後のフォルダ /Renesas/BLE_Software_Ver_X_XX/BLE_Sample/project/windows/Exe に格納されている EXE ファイル「rBLE_Sample.exe」を実行することで起動します。

ただし、本 EXE ファイル「rBLE_Sample.exe」には、起動時に設定する引数が必要となりますので、起動の引数は、EXE ファイルと同じフォルダに格納されているバッチファイル「run.bat」の内容を編集して実行してください。以下に、起動時に必要となる引数について説明します。

表 5-1 起動時に必要となる引数設定

引数名称	詳細説明													
シリアル COM ポート番号	Windows PC で使用する COM ポート番号を設定してください。													
シリアル通信の設定	BLE ソフトウェアの設定に合うように 4800～250,000 の間で指定してください。	<table border="1"> <thead> <tr> <th>COM ポート設定</th><th>設定値</th></tr> </thead> <tbody> <tr> <td>ボーラート</td><td>4800bps～250,000bps</td></tr> <tr> <td>データ長</td><td>8bit</td></tr> <tr> <td>パリティ</td><td>なし</td></tr> <tr> <td>ストップビット</td><td>1bit</td></tr> <tr> <td>フロー制御</td><td>なし</td></tr> </tbody> </table>	COM ポート設定	設定値	ボーラート	4800bps～250,000bps	データ長	8bit	パリティ	なし	ストップビット	1bit	フロー制御	なし
COM ポート設定	設定値													
ボーラート	4800bps～250,000bps													
データ長	8bit													
パリティ	なし													
ストップビット	1bit													
フロー制御	なし													
対向機の BD アドレス (パブリックアドレス)	接続対象となる対向機の BD アドレスを設定してください。この設定により、デバイス検索による BD アドレス取得をすることなく、接続手続きを行うことが出来ます。ただし、指定できるアドレスタイプはパブリックタイプのみです。ご注意下さい。													
UART2 線分岐接続	2 線分岐接続を選択 : -div2wire 2 線接続(通常の UART) : なし													

尚、BLE-MCU に書き込む HEX ファイルは、パッケージインストール後のフォルダ /Renesas/BLE_Software_Ver_X_XX/RL78_G1D/ROM_File に格納されている「RL78_G1D_CM(*).hex」、「RL78_G1D_IM(*).hex」、「RL78_G1D_CCM(*).hex」のいずれかが使用できます。これらの HEX ファイルを使用する場合は、シリアル通信のボーラートは 4800bps になります。

5.3 Embedded 構成で使用する際の起動方法

Embedded 構成のサンプルプログラムは、パッケージインストール後のフォルダ /Renesas/BLE_Software_Ver_X_XX/RL78_G1D/ROM_File に格納されている HEX ファイル「RL78_G1D_CE(*).hex」、「RL78_G1D_IE(*).hex」、「RL78_G1D_CCE(*).hex」のいずれかを RL78/G1D 用評価ボードに書き込んで、評価ボードをリセットすることで起動します。

ただし、本 HEX ファイルを使用するには、RL78/G1D 用評価ボードと Windows PC を USB ケーブルで接続して、PC 上のターミナルソフトからコマンドで操作する必要があります。その際、シリアルポートの設定を以下のように行ってください。

表 5-2 シリアルポート設定

ポート設定	設定値
ボーラート	250,000bps
データ長	8bit
パリティ	なし
ストップビット	1bit
フロー制御	なし

また、ターミナルソフトの受信改行コード設定は LF を指定してください。

図 5-1に、ターミナルソフトとしてフリーソフトの Tera Term を使用した場合の設定画面を示します。

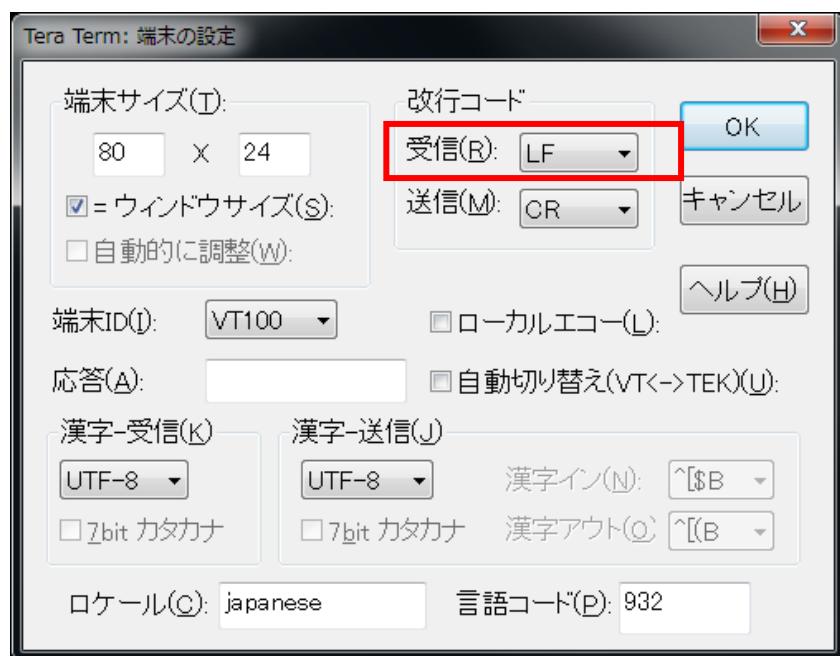


図 5-1 ターミナルソフトの受信改行コード設定例

5.4 コンソール入出力サンプルプログラムの使用方法

コンソール入出力サンプルプログラムを起動すると図 5-2のようなコマンドプロンプト画面が表示されます。

以降では EXE ファイルを実行した際のスクリーンショットで説明します。

【注】以降のスクリーンショット内のコマンド番号は、プロファイル実装数などにより変化することがあります。

図 5-2 起動画面

図 5-2の画面にて、「rBLE Mode (ACTIVE)」という表示が出ていることを確認してください。もし、表示されていなければ何らかの問題により正常起動出来ていない状態です。接続や設定などを再確認してください。

コンソール入出力サンプルプログラムは、基本的に番号指定によるメニュー選択で動作します。起動時に、下記メニューが表示されます。

```
-- BLE Sample Program Menu --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
ESC Key: Menu exit
```

図 5-3 起動時のメニュー画面

この表示では、1～4 のメニュー選択が表示されています。数字キーを入力の上、エンターキーによりメニューを選択する操作方法となります。

また、現状の選択メニューから一つ前のメニューに戻る操作方法は、ESC キーで実施されます。

もう一度、メニューリストを確認したい場合は、番号キーを入力せず、エンターキーを入力してください。

サンプルプログラムを終了する方法は、起動時のメニュー選択まで ESC キーで戻り、再度 ESC キーを入力します。

また、ログ表示は色分け表示されており、水色表記はコマンド実行を意味します。黄色表記はイベント通知を意味します。下図のように、コマンド実行とイベント通知により色分け表記されます。

The screenshot shows a Windows command prompt window titled 'cmd C:\\$WINDOWS\\$system32\cmd.exe'. The window displays a list of BLE command codes from 27 to 49, each followed by its corresponding command name. Below this list, the text 'ESC Key: Menu exit' is visible. At the bottom of the list, there is a blue text entry 'CMD -> GAP Reset Status(RBLE_OK)'. Following this, there are two yellow text entries: 'rBLE_GAP EVENT (RESET_RESULT) Status(RBLE_OK)' and 'rBLE Version = Major(02),Minor(00)'. The entire window has a standard Windows XP-style interface with a blue title bar and a scroll bar on the right side.

```
27.GAP_Channel_Map_Req
28.GAP_Read_RSSI
29.SM_Set_Key
30.SM_Start_Enc
31.SM_Tk_Req_Resp
32.SM_Ltk_Req_Resp
33.SM_Irk_Req_Resp
34.SM_Csrk_Req_Resp
35.SM_Chk_Bd_Addr_Req_Resp
36.GATT_Enable
37.GATT_Discovery_Char_Request
38.GATT_Discovery_Service_Request
39.GATT_Discovery_Char_Descriptor_Request
40.GATT_Read_Char_Request
41.GATT_Write_Char_Request
42.GATT_Write_Reliable_Request
43.GATT_Execute_Write_Char_Request
44.GATT_Notify_Request
45.GATT_Indicate_Request
46.GATT_Write_Response
47.GATT_Set_Permission
48.GATT_Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
Status(RBLE_OK)
>>
rBLE_GAP EVENT (RESET_RESULT) Status(RBLE_OK)
rBLE Version = Major(02),Minor(00)
>>
```

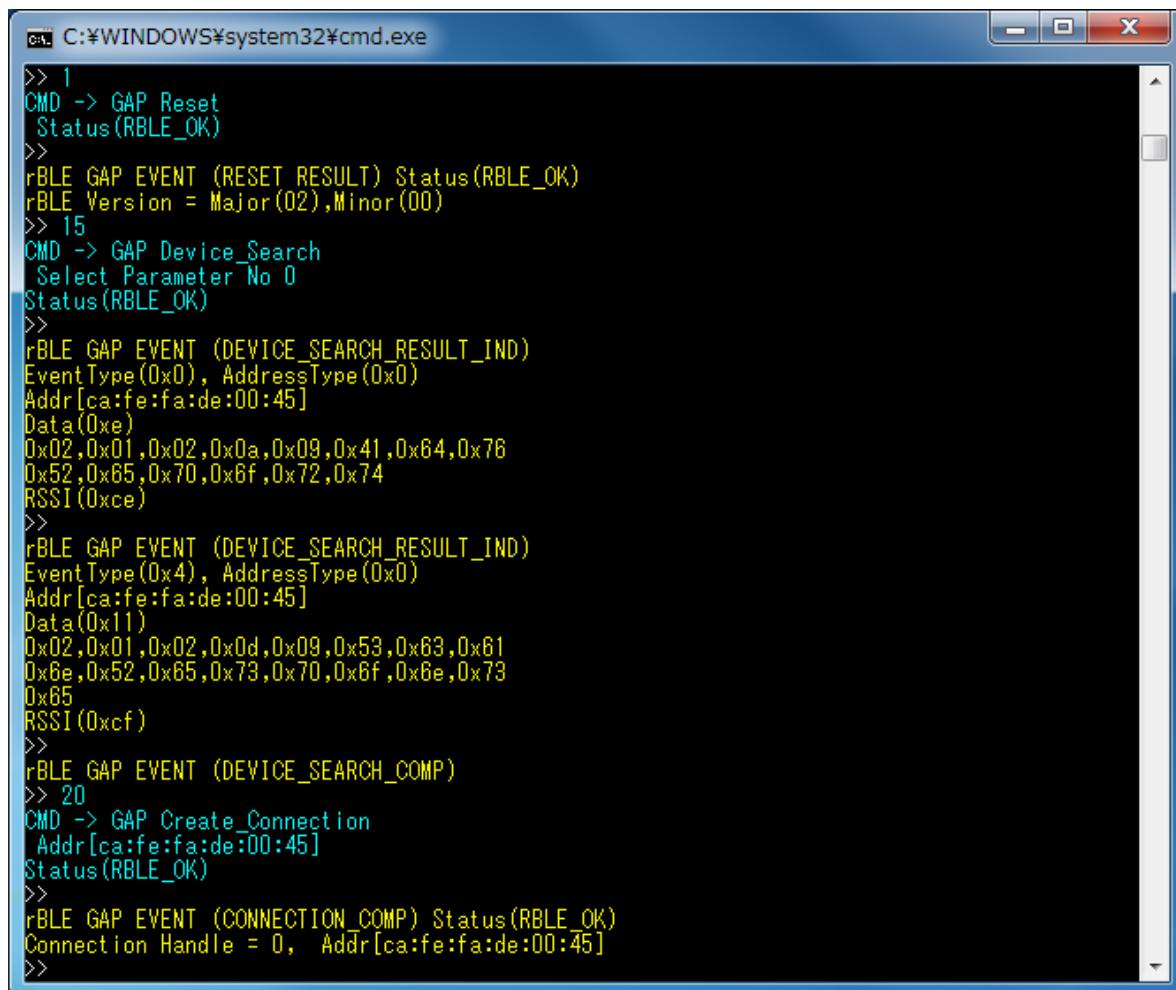
図 5-4 RBLE_GAP_Reset 関数実行例

次節より、各階層における基本的な使用方法を説明します。

5.5 Generic Access Profile (GAP)

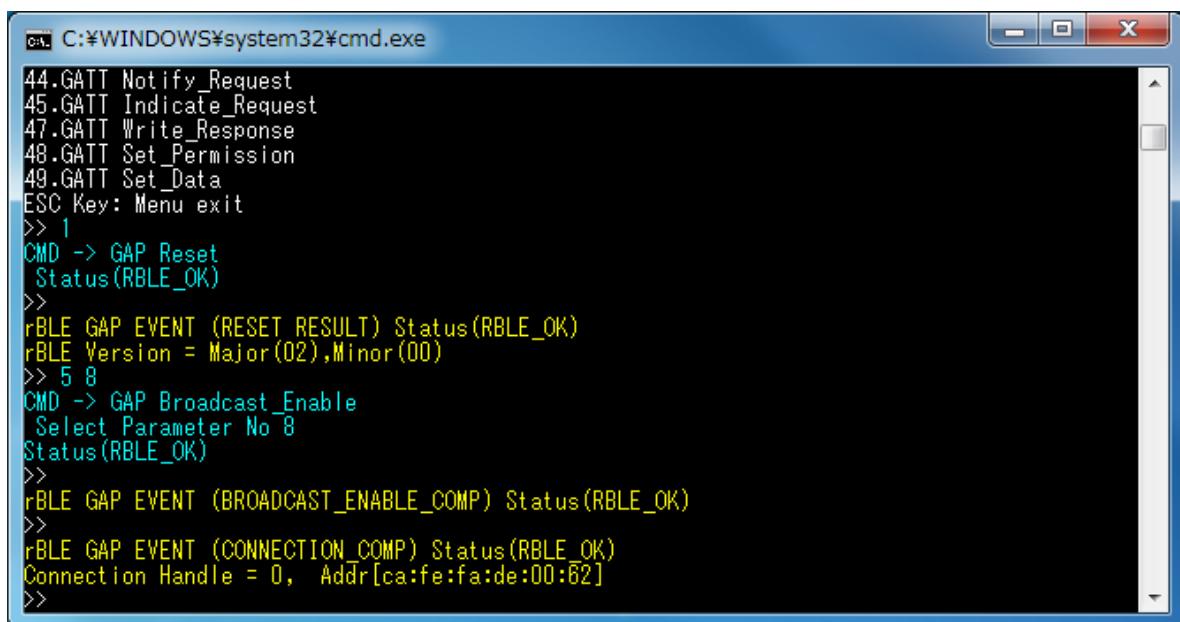
GAP の基本的な動作として、セキュリティなしで接続する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Master デバイス側のログを図 5-5に、Slave デバイス側のログを図 5-6に示します。

内容	Master 側(Command&Event)	Slave 側(Command&Event)
初期化	GAP Reset RESET_RESULT	GAP Reset RESET_RESULT
Advertising を送信		GAP Broadcast_Enable BROADCAST_ENABLE_COMP
Device 検索 (省略可能)	GAP Device_Search DEVICE_SEARCH_RESULT_IND DEVICE_SEARCH_COMP	
接続	GAP Create_Connection CONNECTION_COMP	CONNECTION_COMP



```
C:\> C:\WINDOWS\system32\cmd.exe
>> 1
CMD -> GAP Reset
Status(RBLE_OK)
>>
rBLE GAP EVENT (RESET_RESULT) Status(RBLE_OK)
rBLE Version = Major(02),Minor(00)
>> 15
CMD -> GAP Device_Search
Select Parameter No 0
Status(RBLE_OK)
>>
rBLE GAP EVENT (DEVICE_SEARCH_RESULT_IND)
EventType(0x0), AddressType(0x0)
Addr[ca:fe:fa:de:00:45]
Data(0xe)
0x02,0x01,0x02,0x0a,0x09,0x41,0x64,0x78
0x52,0x65,0x70,0x8f,0x72,0x74
RSSI(0xce)
>>
rBLE GAP EVENT (DEVICE_SEARCH_RESULT_IND)
EventType(0x4), AddressType(0x0)
Addr[ca:fe:fa:de:00:45]
Data(0x11)
0x02,0x01,0x02,0x0d,0x09,0x53,0x63,0x81
0x6e,0x52,0x65,0x73,0x70,0x6f,0x6e,0x73
0x65
RSSI(0xcf)
>>
rBLE GAP EVENT (DEVICE_SEARCH_COMP)
>> 20
CMD -> GAP Create_Connection
Addr[ca:fe:fa:de:00:45]
Status(RBLE_OK)
>>
rBLE GAP EVENT (CONNECTION_COMP) Status(RBLE_OK)
Connection Handle = 0, Addr[ca:fe:fa:de:00:45]
>>
```

図 5-5 セキュリティなしで接続 Master ログ



```
C:\WINDOWS\system32\cmd.exe
44.GATT Notify_Request
45.GATT Indicate_Request
47.GATT Write_Response
48.GATT Set_Permission
49.GATT Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
Status(RBLED_OK)
>>
rBLE GAP EVENT (RESET_RESULT) Status(RBLED_OK)
rBLE Version = Major(02),Minor(00)
>> 5 8
CMD -> GAP Broadcast_Enable
Select Parameter No 8
Status(RBLED_OK)
>>
rBLE GAP EVENT (BROADCAST_ENABLE_COMP) Status(RBLED_OK)
>>
rBLE GAP EVENT (CONNECTION_COMP) Status(RBLED_OK)
Connection Handle = 0, Addr[ca:fe:fa:de:00:62]
>>
```

図 5-6 セキュリティなしで接続 Slave ログ

5.6 Security Manager (SM)

SM の基本的な動作として、セキュリティありで接続する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Master デバイス側のログを図 5-7 と図 5-8 に、Slave デバイス側のログを図 5-9 と図 5-10 に示します。ログでは、Device 検索を省略しています。

内容	Master 側(Command&Event)	Slave 側(Command&Event)
初期化	GAP Reset RESET_RESULT	GAP Reset RESET_RESULT
セキュリティ設定	GAP Set_Security_Request SET_SECURITY_REQUEST_COMP GAP_Set_Bonding_Mode SET_BONDING_MODE_COMP	GAP Set_Security_Request SET_SECURITY_REQUEST_COMP GAP_Set_Bonding_Mode SET_BONDING_MODE_COMP
Advertising を送信		GAP Broadcast_Enable BROADCAST_ENABLE_COMP
Device 検索 (省略可能)	GAP Device_Search DEVICE_SEARCH_RESULT_IND DEVICE_SEARCH_COMP	
接続	GAP Create_Connection CONNECTION_COMP	CONNECTION_COMP
Device 確認	BD_ADDR_REQ_IND SM Chk_Bd_Addr_Req_Resp	BD_ADDR_REQ_IND SM Chk_Bd_Addr_Req_Resp
Bonding 開始	GAP Start_Bonding	
Bonding 要求 および応答		BONDING_REQ_IND GAP Bonding_Response
TK 要求	TK_REQ_IND	

および応答	SM Tk_Req_Resp	TK_REQ_IND SM Tk_Req_Resp
LTK の配布		LTK_REQ_IND SM Ltk_Req_Resp
Key 通知	KEY_IND	KEY_IND
Bonding 完了	BONDING_COMP	BONDING_COMP

```
C:\> C:\WINDOWS\system32\cmd.exe
44.GATT Notify_Request
45.GATT Indicate_Request
47.GATT Write_Response
48.GATT Set_Permission
49.GATT Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
Status(RBle_OK)
>>
rBLE GAP EVENT (RESET_RESULT) Status(RBle_OK)
rBLE Version = Major(02),Minor(00)
>> 7
CMD -> GAP_Set_Bonding_Mode
Status(RBle_OK)
>>
rBLE GAP EVENT (SET_BONDING_MODE_COMP) Status(RBle_OK)
>> 8
CMD -> GAP_Set_Security_Request
Status(RBle_OK)
>>
rBLE GAP EVENT (SET_SECURITY_REQUEST_COMP) Status(RBle_OK), SEC(1)
>> 20
CMD -> GAP_Create_Connection
Addr[ca:fe:fa:de:00:45]
Status(RBle_OK)
>>
rBLE GAP EVENT (CONNECTION_COMP) Status(RBle_OK)
Connection Handle = 0, Addr[ca:fe:fa:de:00:45]
>>
rBLE SM EVENT(BD_ADDR_REQ_IND)
idx = 0, type = 0, Addr[ca:fe:fa:de:00:45]
>> 35
CMD -> SM_Clk_Bd_Addr_Req_Resp
Status(RBle_OK)
>> 23
CMD -> GAP_Start_Bonding
Select Parameter No 0
Status(RBle_OK)
>>
rBLE SM EVENT(TK_REQ_IND)
idx = 0, oob_en = 0, disp_en = 0
>> 31
CMD -> SM_Tk_Req_Resp
Status(RBle_OK)
>>
rBLE SM EVENT(KEY_IND)
idx = 0, ediv = 4660, key_code = Encryption key
RandomData:29,23,be,84,e1,6c,d6,ae
```

図 5-7 セキュリティありで接続 Master ログ

```
C:\WINDOWS\system32\cmd.exe
>>
rBLE SM EVENT(KEY_IND)
idx = 0, ediv = 4660, key_code = Encryption key
RandomData:29,23,be,84,e1,6c,d6,ae

KeyData:52,90,49,f1,f1,bb,e9,eb,b3,a6,db,3c,87,0c,3e,99
>>
rBLE SM EVENT(LTK_REQ_IND)
+idx = 0
>> 32
CMD -> SM Ltk_Req_Resp
RandomData:29,23,be,84,e1,6c,d6,ae

KeyData:52,90,49,f1,f1,bb,e9,eb,b3,a6,db,3c,87,0c,3e,99
Status(RBLE_OK)
>>
rBLE GAP EVENT (BONDING_COMP) Status(RBLE_OK)
>>
```

図 5-8 セキュリティありで接続 Master ログ(続き)

```
C:\WINDOWS\system32\cmd.exe
48.GATT Set_Permission
49.GATT Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
Status(RBLE_OK)
>>
rBLE GAP EVENT (RESET_RESULT) Status(RBLE_OK)
rBLE Version = Major(02),Minor(00)
>> 7
CMD -> GAP_Set_Bonding_Mode
Status(RBLE_OK)
>>
rBLE GAP EVENT (SET_BONDING_MODE_COMP) Status(RBLE_OK)
>> 8
CMD -> GAP_Set_Security_Request
Status(RBLE_OK)
>>
rBLE GAP EVENT (SET_SECURITY_REQUEST_COMP) Status(RBLE_OK), SEC(1)
>> 5 8
CMD -> GAP_Broadcast_Enable
Select Parameter No 8
Status(RBLE_OK)
>>
rBLE GAP EVENT (BROADCAST_ENABLE_COMP) Status(RBLE_OK)
>>
rBLE GAP EVENT (CONNECTION_COMP) Status(RBLE_OK)
Connection Handle = 0, Addr[ca:fe:fa:de:00:62]
>>
rBLE SM EVENT(BD_ADDR_REQ_IND)
idx = 0, type = 0, Addr[ca:fe:fa:de:00:62]
>> 35
CMD -> SM Chk_Bd_Addr_Req_Resp
Status(RBLE_OK)
>>
rBLE GAP EVENT (BONDING_REQ_IND)
```

図 5-9 セキュリティありで接続 Slave ログ

```

C:\> rBLE_GAP_EVENT (BONDING_REQ_IND)
Addr[ca:fe:fa:de:00:62]
>> 25
CMD -> GAP Bonding_Response
Select Parameter No 0
Status(RBLE_OK)
>>
rBLE_SM_EVENT(TK_REQ_IND)
idx = 0, oob_en = 0, disp_en = 1
>> 31
CMD -> SM Tk_Req_Resp
Status(RBLE_OK)
>>
rBLE_SM_EVENT(LTK_REQ_IND)
idx = 0
>> 32
CMD -> SM Ltk_Req_Resp
RandomData:29,23,be,84,e1,6c,d6,ae
KeyData:52,90,49,f1,f1,bb,e9,eb,b3,a6,db,3c,87,0c,3e,99
Status(RBLE_OK)
>>
rBLE_SM_EVENT(KEY_IND)
idx = 0, ediv = 4660, key_code = Encryption key
RandomData:29,23,be,84,e1,6c,d6,ae
KeyData:52,90,49,f1,f1,bb,e9,eb,b3,a6,db,3c,87,0c,3e,99
>>
rBLE_GAP_EVENT (BONDING_COMP) Status(RBLE_OK)
>>

```

図 5-10 セキュリティありで接続 Slave ログ(続き)

5.7 Generic Attribute Profile (GATT)

GATT の基本的な動作として、対向機のサービスでグループ化された特性(Characteristic)と呼ばれるデータのハンドルを取得する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Master デバイス側のログを図 5-11に、Slave デバイス側のログを図 5-12に示します。

内容	Master 側(Command&Event)	Slave 側(Command&Event)
対向機 と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
GATT を有効		GATT Enable
特性情報を 取得		GATT Discovery_Char_Request DISC_CHAR_BY_UUID_CMP DISC_CHAR_BY_UUID_CMP COMPLETE

```
C:\WINDOWS\system32\cmd.exe
48.GATT Set_Permission
49.GATT Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
Status(RBLE_OK)
>>
rBLE GAP EVENT (RESET_RESULT) Status(RBLE_OK)
rBLE Version = Major(02),Minor(00)
>> 20
CMD -> GAP Create_Connection
Addr[ca:fe:fa:de:00:45]
Status(RBLE_OK)
>>
rBLE GAP EVENT (CONNECTION_COMP) Status(RBLE_OK)
Connection Handle = 0, Addr[ca:fe:fa:de:00:45]
>> 36
CMD -> GATT_Enable
Status(RBLE_OK)
>> 37
CMD -> GATT_Discovery_Char_Request
Status(RBLE_OK)
>>
rBLE GATT EVENT (DISC_CHAR_BY_UUID_CMP) Att_Code(0), NbEntry(1)
attribute_handle:(0002), prop:(02), pointer_hdl:(0003), uuid:(2A00)
>>
rBLE GATT EVENT (COMPLETE) Att_Code(ATTRIBUTE_NOT_FOUND)
>>
```

図 5-11 GATT で特性(Characteristic)情報を取得 Master ログ

```
C:\WINDOWS\system32\cmd.exe
48.GATT Set_Permission
49.GATT Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
Status(RBLE_OK)
>>
rBLE GAP EVENT (RESET_RESULT) Status(RBLE_OK)
rBLE Version = Major(02),Minor(00)
>> 5 8
CMD -> GAP Broadcast_Enable
Select Parameter No 8
Status(RBLE_OK)
>>
rBLE GAP EVENT (BROADCAST_ENABLE_COMP) Status(RBLE_OK)
>>
rBLE GAP EVENT (CONNECTION_COMP) Status(RBLE_OK)
Connection Handle = 0, Addr[ca:fe:fa:de:00:62]
>> 36
CMD -> GATT_Enable
Status(RBLE_OK)
>> 37
CMD -> GATT_Discovery_Char_Request
Status(RBLE_OK)
>>
rBLE GATT EVENT (DISC_CHAR_BY_UUID_CMP) Att_Code(0), NbEntry(1)
attribute_handle:(0002), prop:(02), pointer_hdl:(0003), uuid:(2A00)
>>
rBLE GATT EVENT (COMPLETE) Att_Code(ATTRIBUTE_NOT_FOUND)
>>
```

図 5-12 GATT で特性(Characteristic)情報を取得 Slave ログ

5.8 Find Me Profile (FMP)

FMP の基本的な動作として、Alert データを設定する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Locator デバイス側のログを図 5-13に、Target デバイス側のログを図 5-14に示します。

内容	Locator 側(Command&Event)	Target 側(Command&Event)
対向機 と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
Target を有効		FMP Target_Enable TARGET_ENABLE_COMP
Locator を有効	FMP Locator_Enable LOCATOR_ENABLE_COMP	
Alert 設定	FMP Locator_Set_Alert	TARGET_ALERT_IND

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profileおよび5.6Security Managerを参照ください。

```

C:\WINDOWS\system32\cmd.exe
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 1
-- BLE Sample Program Find Me Profile Test Menu --
1.FMP Target_Enable
2.FMP Target_Disable
3.FMP Locator_Enable
4.FMP Locator_Disable
5.FMP Locator_Set_Alert
ESC Key: Menu exit
>> 3
CMD -> FMP Locator_Enable
Status(RBLE_OK)
>>
rBLE FMP EVENT (LOCATOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Immediate Alert service
  Start Handle    = 0x0015
  End Handle     = 0x0017
  alert_char_hdl  = 0x0016
  alert_val_hdl   = 0x0017
  alert_char_prop = 0x04

>> 5
CMD -> FMP Locator_Set_Alert
Select Parameter No 0
Status(RBLE_OK)
>>

```

図 5-13 FMP Locator 側ログ

```

C:\WINDOWS\system32\cmd.exe
>>
rBLE GAP EVENT (CONNECTION_COMP) Status(RBLE_OK)
Connection Handle = 0, Addr[ca:fe:fa:de:00:b2]
>> <
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 1
-- BLE Sample Program Find Me Profile Test Menu --
1.FMP Target_Enable
2.FMP Target_Disable
3.FMP Locator_Enable
4.FMP Locator_Disable
5.FMP Locator_Set_Alert
ESC Key: Menu exit
>> 1
CMD -> FMP Target_Enable
Status(RBLE_OK)
>>
rBLE FMP EVENT (TARGET_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE FMP EVENT (TARGET_ALERT_IND)
Connection Handle = 0
alert_lvl = 2
>>

```

図 5-14 FMP Target 側ログ

5.9 Proximity Profile (PXP)

PXP の基本的な動作として、Alert Level を読み出し、設定する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Monitor デバイス側のログを図 5-15 と図 5-16 に、Reporter デバイス側のログを図 5-17 に示します。

内容	Monitor 側(Command&Event)	Reporter 側(Command&Event)
対向機 と接続	5.5Generic Access Profile (GAP) および 5.6Security Manager (SM) を参照	
Reporter を有効		PXP Reporter_Enable REPORTER_ENABLE_COMP
Monitor を有効	PXP Monitor_Enable MONITOR_ENABLE_COMP	
Alert Level 読み出し	PXP Monitor_Get_Alert_Level MONITOR_READ_CHAR_RESPONSE	
Alert Level 設定	PXP Monitor_Set_Alert_Level MONITOR_WRITE_CHAR_RESPONSE	

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profileおよび5.6Security Managerを参照ください。

```
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 3
-- BLE Sample Program Proximity Profile Test Menu --
1.PXP Reporter_Enable
2.PXP Reporter_Disable
3.PXP Monitor_Enable
4.PXP Monitor_Disable
5.PXP Monitor_Get_Alert_Level
6.PXP Monitor_Set_Alert_Level
7.PXP Monitor_Get_Tx_Power
ESC Key: Menu exit
>> 3
CMD -> PXP Monitor_Enable
Status(RBLE_OK)
>>
rBLE PXP EVENT (MONITOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Link Loss Service
  Start Handle    = 0x000F
  End Handle     = 0x0011

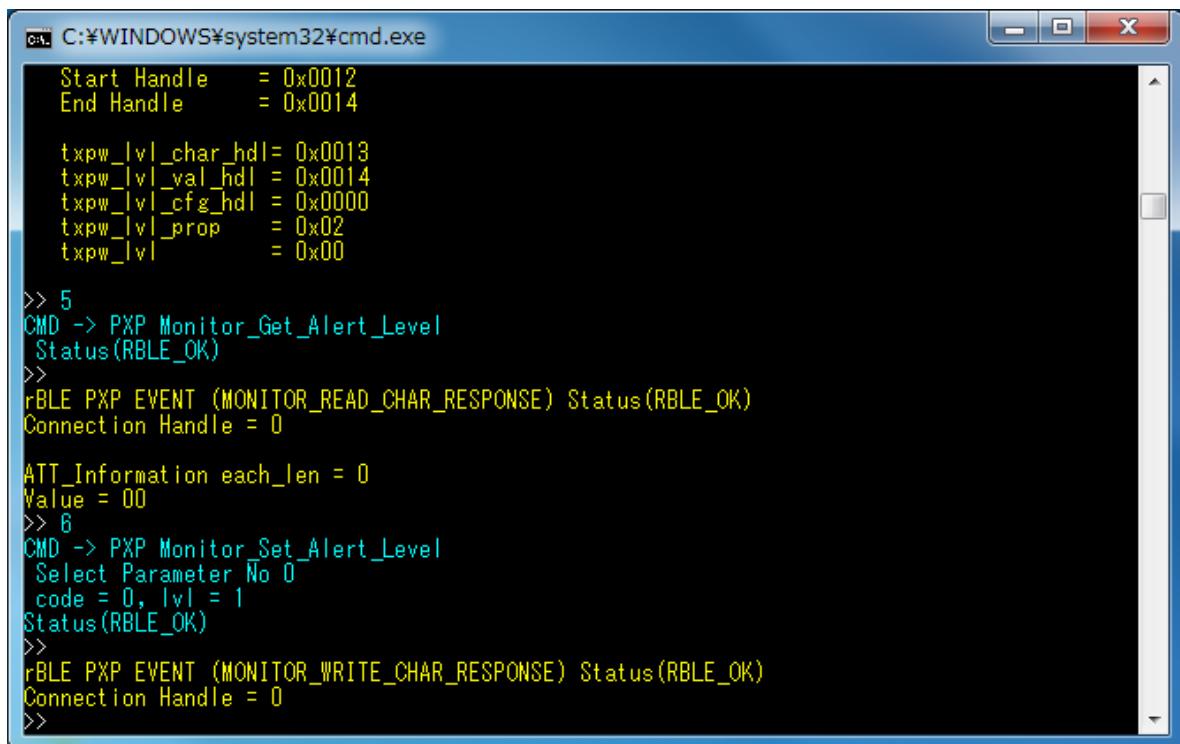
  Alert level char handle = 0x0010
  alert Level value handle= 0x0011
  Alert level properties = 0x0A
  Alert value       = 0x00

* Immediate Alert service
  Start Handle    = 0x0015
  End Handle     = 0x0017

  Alert level char handle = 0x0016
  alert Level value handle= 0x0017
  Alert level properties = 0x04
  Alert value       = 0x00

* Tx Power Service
  Start Handle    = 0x0012
  End Handle     = 0x0014
```

図 5-15 PXP Monitor 側ログ



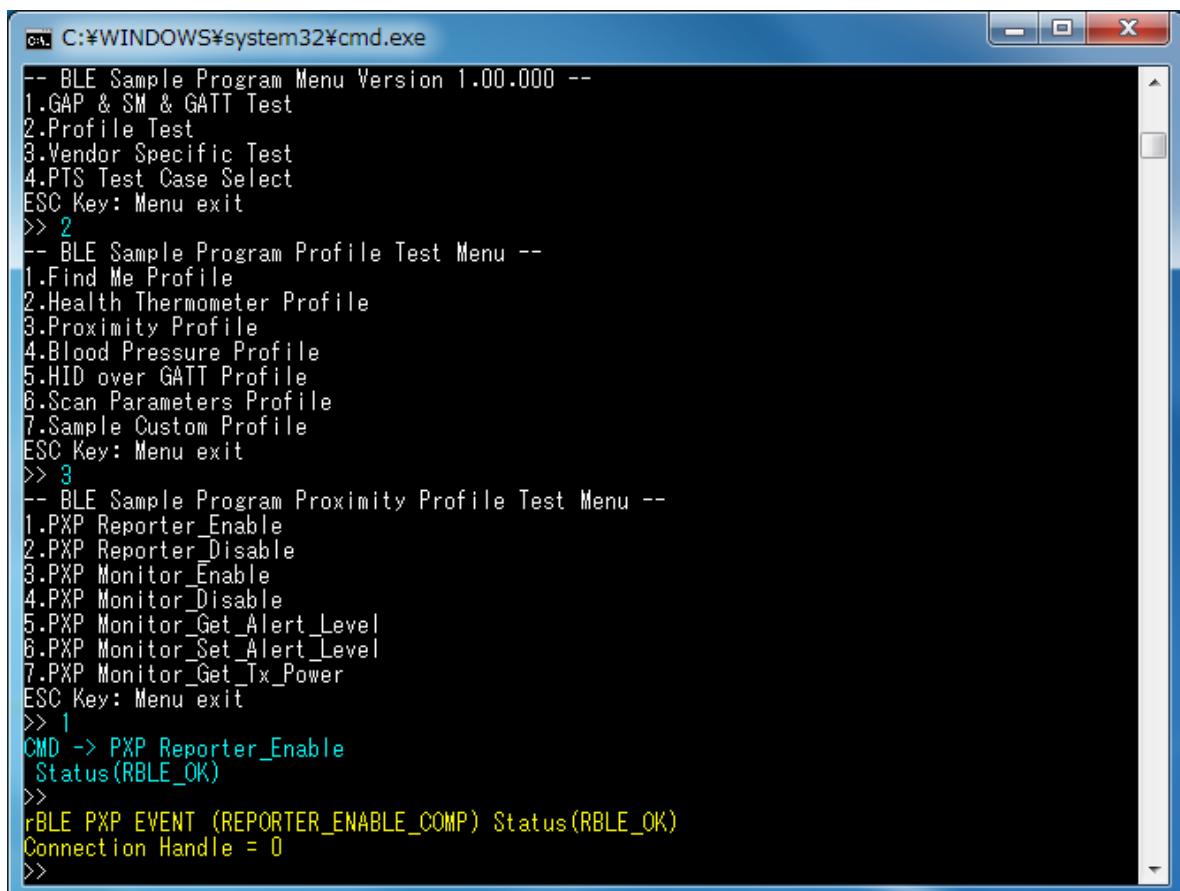
```
C:\WINDOWS\system32\cmd.exe
Start Handle      = 0x0012
End Handle       = 0x0014

txpw_lv1_char_hdl = 0x0013
txpw_lv1_val_hdl = 0x0014
txpw_lv1_cfg_hdl = 0x0000
txpw_lv1_prop     = 0x02
txpw_lv1          = 0x00

>> 5
CMD -> PXP Monitor_Get_Alert_Level
Status(RBLE_OK)
>>
rBLE PXP EVENT (MONITOR_READ_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0

ATT_Information each_len = 0
Value = 00
>> 6
CMD -> PXP Monitor_Set_Alert_Level
Select Parameter No 0
code = 0, lvl = 1
Status(RBLE_OK)
>>
rBLE PXP EVENT (MONITOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
```

図 5-16 PXP Monitor 側ログ(続き)



```
C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 3
-- BLE Sample Program Proximity Profile Test Menu --
1.PXP Reporter_Enable
2.PXP Reporter_Disable
3.PXP Monitor_Enable
4.PXP Monitor_Disable
5.PXP Monitor_Get_Alert_Level
6.PXP Monitor_Set_Alert_Level
7.PXP Monitor_Get_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> PXP Reporter_Enable
Status(RBLE_OK)
>>
rBLE PXP EVENT (REPORTER_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
```

図 5-17 PXP Reporter 側ログ

5.10 Health Thermometer Profile (HTP)

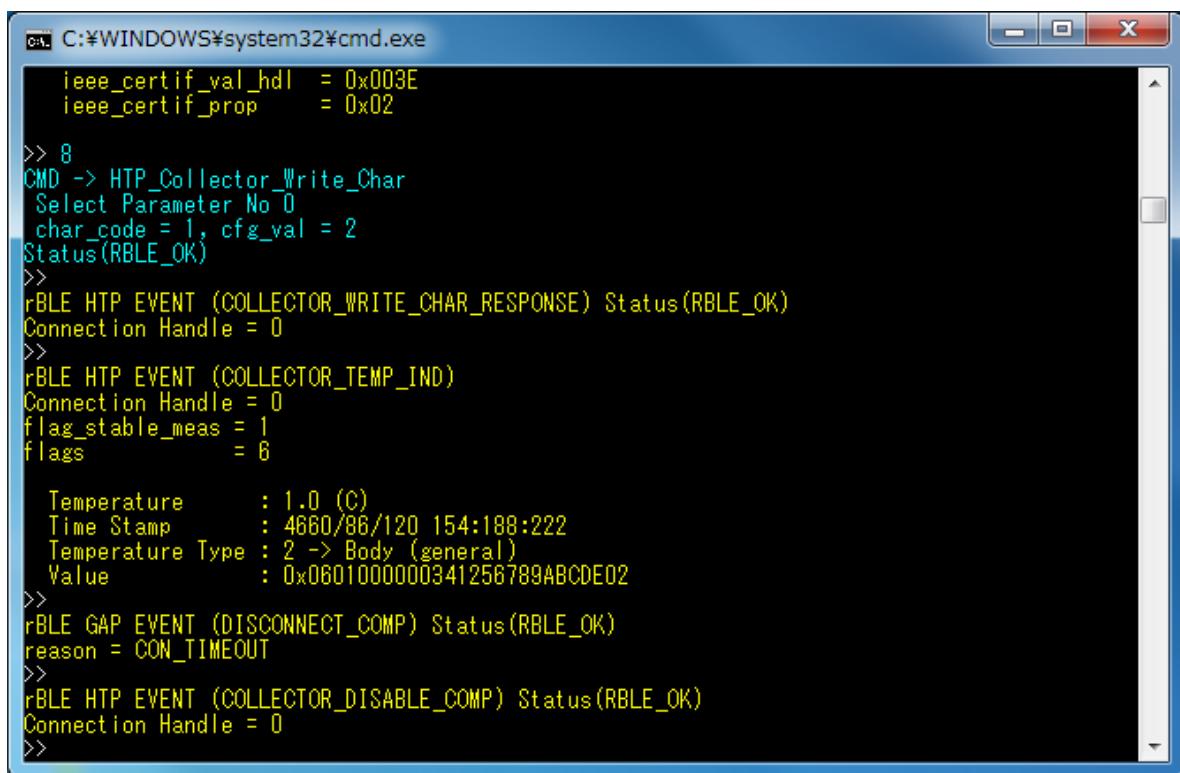
HTP の基本的な動作として、Thermometer データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Collector デバイス側のログを図 5-18 と図 5-19 に、Thermometer デバイス側のログを図 5-20 と図 5-21 に示します。

内容	Collector 側(Command&Event)	Thermometer 側(Command&Event)
対向機と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
Thermometerを有効		HTP Thermometer_Enable THERMOMETER_ENABLE_COMP
Collectorを有効	HTP Collector_Enable COLLECTOR_ENABLE_COMP	
Indicationを有効	HTP Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	THERMOMETER_CFG_INDNTF_IND
Thermometerを送受信	COLLECTOR_TEMP_IND	HTP Thermometer_Send_Temp THERMOMETER_SEND_TEMP_COMP

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profileおよび5.6Security Managerを参照ください。

```
C:\> C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 2
-- BLE Sample Program Health Thermometer Profile Test Menu --
1.HTP Thermometer_Enable
2.HTP Thermometer_Disable
3.HTP Thermometer_Send_Temp
4.HTP Thermometer_Req_Measurement_Period_Ind
5.HTP Collector_Enable
6.HTP Collector_Disable
7.HTP_Collector_Read_Char
8.HTP_Collector_Write_Char
9.HTP_Collector_Set_Measurement_Period
ESC Key: Menu exit
>> 5
CMD -> HTP_Collector_Enable
Status(RBLE_OK)
>>
rBLE HTP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
```

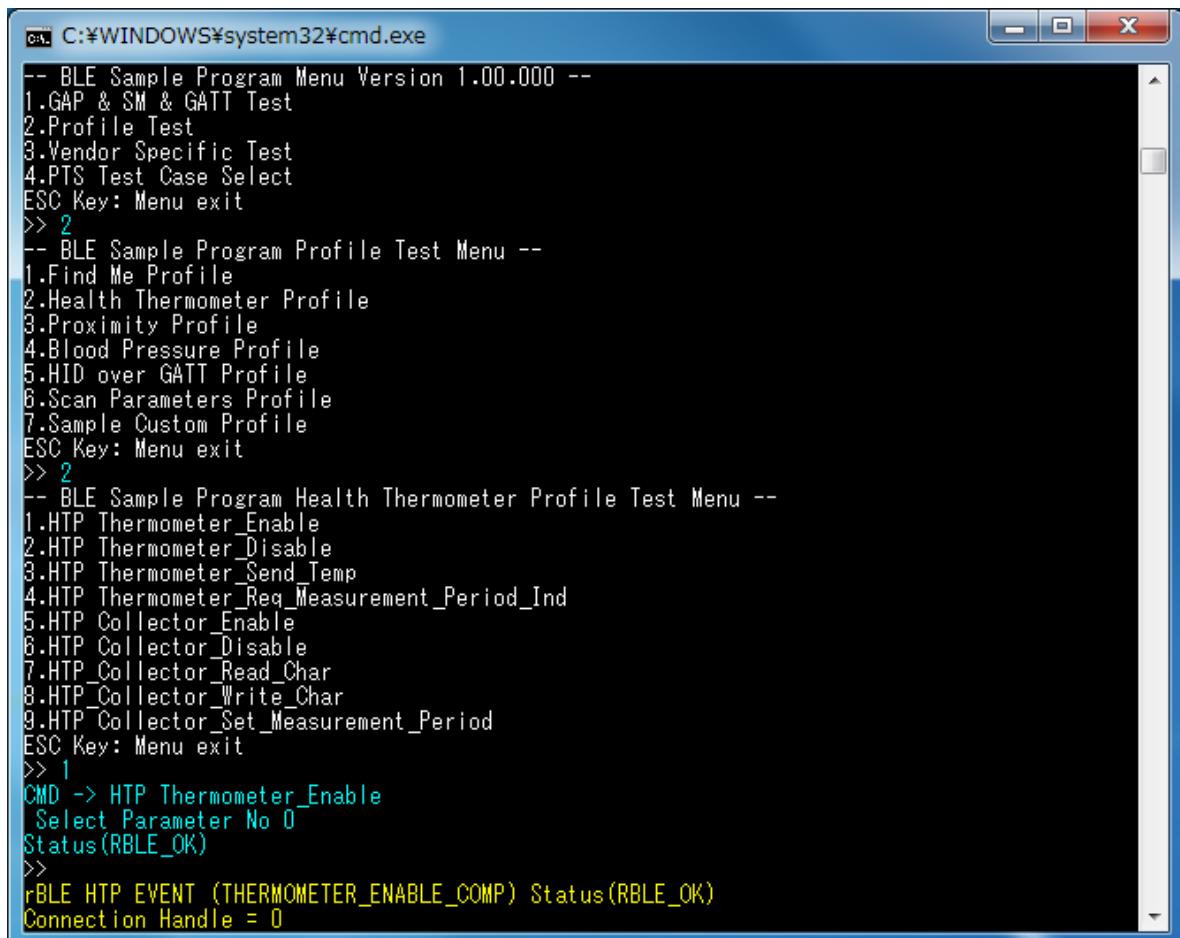
図 5-18 HTP Collector 側ログ



```
C:\WINDOWS\system32\cmd.exe
ieee_certif_val_hdl = 0x003E
ieee_certif_prop = 0x02

>> 8
CMD -> HTP_Collector_Write_Char
Select Parameter No 0
char_code = 1, cfg_val = 2
Status(RBLE_OK)
>>
rBLE HTP EVENT (COLLECTOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE HTP EVENT (COLLECTOR_TEMP_IND)
Connection Handle = 0
flag_stable_meas = 1
flags = 6
Temperature : 1.0 (C)
Time Stamp : 4660/06/12 154:188:222
Temperature Type : 2 -> Body (general)
Value : 0x0601000000341256789ABCDE02
>>
rBLE GAP EVENT (DISCONNECT_COMP) Status(RBLE_OK)
reason = CON_TIMEOUT
>>
rBLE HTP EVENT (COLLECTOR_DISABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
```

図 5-19 HTP Collector 側ログ(続き)



```
C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 2
-- BLE Sample Program Health Thermometer Profile Test Menu --
1.HTP_Thermometer_Enable
2.HTP_Thermometer_Disable
3.HTP_Thermometer_Send_Temp
4.HTP_Thermometer_Req_Measurement_Period_Ind
5.HTP_Collector_Enable
6.HTP_Collector_Disable
7.HTP_Collector_Read_Char
8.HTP_Collector_Write_Char
9.HTP_Collector_Set_Measurement_Period
ESC Key: Menu exit
>> 1
CMD -> HTP_Thermometer_Enable
Select Parameter No 0
Status(RBLE_OK)
>>
rBLE HTP EVENT (THERMOMETER_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
```

図 5-20 HTP Thermometer 側ログ

```

C:\> C:\WINDOWS\system32\cmd.exe
rBLE HTP EVENT (THERMOMETER_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE HTP EVENT (THERMOMETER_CFG_INDNTF_IND)
Connection Handle = 0
Char code = 1
Cfg val = 2
>> 3
CMD -> HTP Thermometer_Send_Temp
Select Parameter No 0
Value = 1
Status(RBLE_OK)
>>
rBLE GAP EVENT (DISCONNECT_COMP) Status(RBLE_OK)
reason = CON_TIMEOUT
>>
rBLE HTP EVENT (THERMOMETER_DISABLE_COMP)
Connection Handle = 0
>>

```

図 5-21 HTP Thermometer 側ログ(続き)

5.11 Blood Pressure Profile (BLP)

BLP の基本的な動作として、測定データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Collector デバイス側のログを図 5-22 と図 5-23 に、Sensor デバイス側のログを図 5-24 に示します。

内容	Collector 側(Command&Event)	Sensor 側(Command&Event)
対向機 と接続	5.5Generic Access Profile (GAP) および 5.6Security Manager (SM) を参照	
Sensor を有効		BLP_Sensor_Enable SENSOR_ENABLE_COMP
Collector を有効	BLP_Collector_Enable COLLECTOR_ENABLE_COMP	
Indication を 有効	BLP_Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_INDNTF_IND
測定データ を送受信	E COLLECTOR_MEASUREMENTS_IND	BLP_Sensor_Send_Measurements SENSOR_SEND_MEASUREMENTS_COMP

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profile および 5.6Security Manager を参照ください。

```

C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 4
-- BLE Sample Program Blood Pressure Profile Test Menu --
1.BLP_Sensor_Enable
2.BLP_Sensor_Disable
3.BLP_Sensor_Send_Measurements
4.BLP_Collector_Enable
5.BLP_Collector_Disable
6.BLP_Collector_Read_Char
7.BLP_Collector_Write_Char
ESC Key: Menu exit
>> 4
CMD -> BLP_Collector_Enable
Status(RBLE_OK)
>>
rBLE BLP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Blood Pressure Service
  Start Handle = 0x0025
  End Handle   = 0x002D

```

図 5-22 BLP Collector 側ログ

```

C:\WINDOWS\system32\cmd.exe
manuf_name_val_hdl  = 0x003C
manuf_name_prop     = 0x02

ieee_certif_char_hdl = 0x003D
ieee_certif_val_hdl = 0x003E
ieee_certif_prop     = 0x02

>> 7
CMD -> BLP_Collector_Write_Char
Select Parameter No 0
Status(RBLE_OK)
>>
rBLE BLP EVENT (COLLECTOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE BLP EVENT (COLLECTOR_MEASUREMENTS_IND)
Connection Handle = 0
S:123.0, D:85.0, M:103.0 (mmHg)
TS: 2012/09/01 12:34:56
Rate:81.7
UserID:1
M.Sts:0008
>>

```

図 5-23 BLP Collector 側ログ(続き)

```

C:\WINDOWS\system32\cmd.exe
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 4
-- BLE Sample Program Blood Pressure Profile Test Menu --
1.BLP_Sensor_Enable
2.BLP_Sensor_Disable
3.BLP_Sensor_Send_Measurements
4.BLP_Collector_Enable
5.BLP_Collector_Disable
6.BLP_Collector_Read_Char
7.BLP_Collector_Write_Char
ESC Key: Menu exit
>> 1
CMD -> BLP_Sensor_Enable
Status(RBLE_OK)
>>
rBLE BLP EVENT (SENSOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE BLP EVENT (SENSOR_CFG_INDNTF_IND)
    Char_Code = BLDPRS_MEAS
    Cfg_Value = START_NTF_IND
>> 3 1
CMD -> BLP_Sensor_Send_Measurements
Select Parameter-> Stable
Status(RBLE_OK)
>>
rBLE BLP EVENT (SENSOR_SEND_MEASUREMENTS_COMP) Status(RBLE_OK)
Connection Handle = 0
>>

```

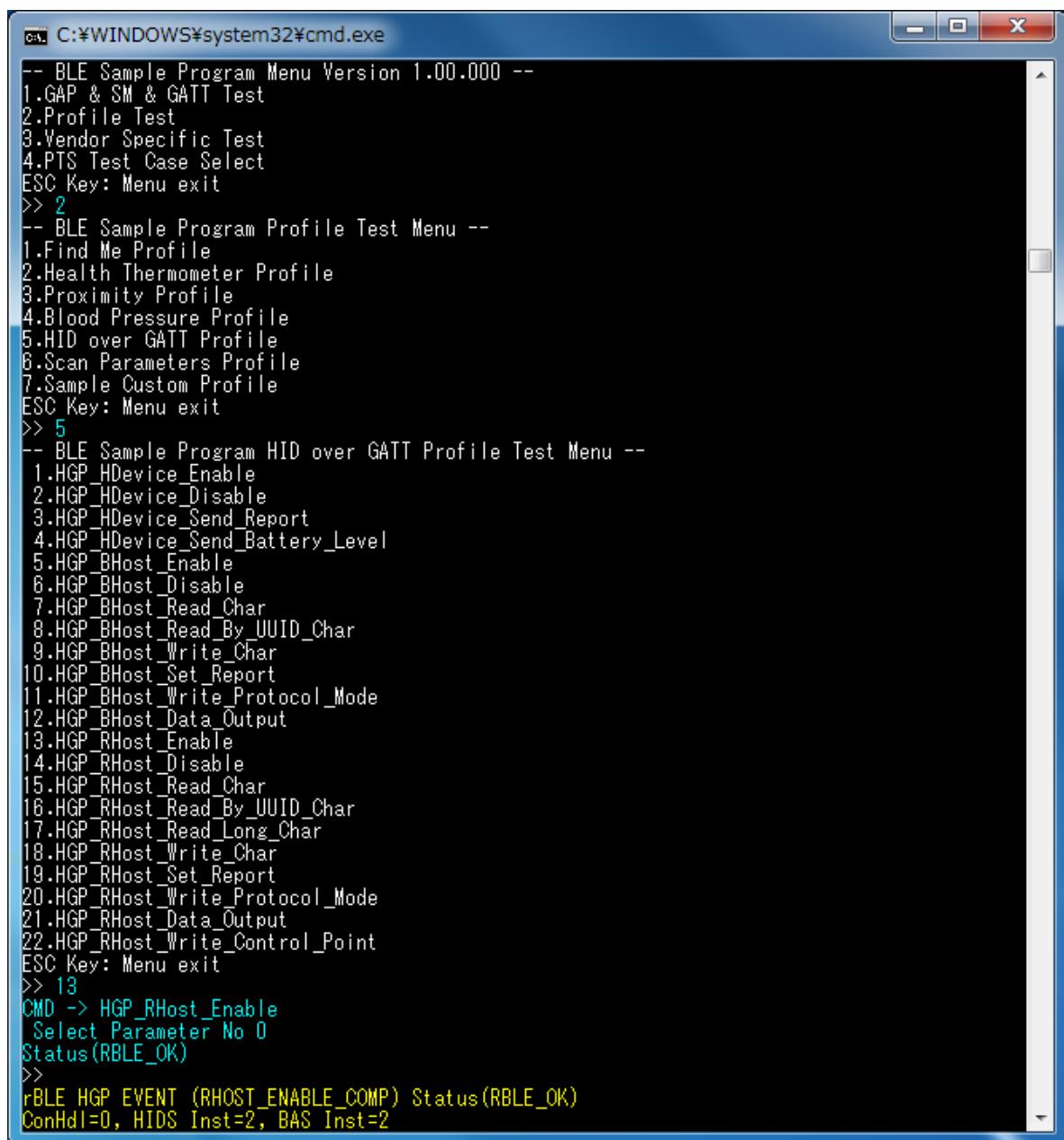
図 5-24 BLP Sensor 側ログ

5.12 HID over GATT Profile (HOGP)

HOGP の基本的な動作として、Input Report データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Report Host デバイス側のログを図 5-25 と図 5-26 に、HID Device デバイス側のログを図 5-27 と図 5-28 に示します。

内容	ReportHost 側(Command&Event)	HID Device 側(Command&Event)
対向機と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
HID デバイスを有効		HGP_HDevice_Enable HDEVICE_ENABLE_COMP
Report Host を有効	HGP_RHost_Enable RHOST_ENABLE_COMP	
Input Report データを送受信	HGP_RHost_Set_Report RHOST_WRITE_CHAR_RESPONSE	HDEVICE_REPORT_IND

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profileおよび5.6Security Managerを参照ください。



```

C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 5
-- BLE Sample Program HID over GATT Profile Test Menu --
1.HGP_HDevice_Enable
2.HGP_HDevice_Disable
3.HGP_HDevice_Send_Report
4.HGP_HDevice_Send_Battery_Level
5.HGP_BHost_Enable
6.HGP_BHost_Disable
7.HGP_BHost_Read_Char
8.HGP_BHost_Read_By_UUID_Char
9.HGP_BHost_Write_Char
10.HGP_BHost_Set_Report
11.HGP_BHost_Write_Protocol_Mode
12.HGP_BHost_Data_Output
13.HGP_RHost_Enable
14.HGP_RHost_Disable
15.HGP_RHost_Read_Char
16.HGP_RHost_Read_By_UUID_Char
17.HGP_RHost_Read_Long_Char
18.HGP_RHost_Write_Char
19.HGP_RHost_Set_Report
20.HGP_RHost_Write_Protocol_Mode
21.HGP_RHost_Data_Output
22.HGP_RHost_Write_Control_Point
ESC Key: Menu exit
>> 13
CMD -> HGP_RHost_Enable
Select Parameter No 0
Status(RBLE_OK)
>>
rBLE_HGP_EVENT (RHOST_ENABLE_COMP) Status(RBLE_OK)
ConHdl=0, HIDS Inst=2, BAS Inst=2

```

図 5-25 Report Host 側ログ

```
C:\WINDOWS\system32\cmd.exe
Batt.Lvl AttHdl=7A, Prop=12, Hdl=7B, Cfg=7C, Ref=7E
>> 19
CMD -> HGP_RHost_Set_Report
Select Parameter No 0
Select inst_idx No 0
Device type = 1, Report type = 1
Status(RBLE_OK)
>>
rBLE HGP EVENT (RHOST_WRITE_CHAR_RESPONSE)
Connection Handle = 0
att_code = 0
>>
```

図 5-26 Report Host 側ログ(続き)

```
C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 5
-- BLE Sample Program HID over GATT Profile Test Menu --
1.HGP_HDevice_Enable
2.HGP_HDevice_Disable
3.HGP_HDevice_Send_Report
4.HGP_HDevice_Send_Battery_Level
5.HGP_BHost_Enable
6.HGP_BHost_Disable
7.HGP_BHost_Read_Char
8.HGP_BHost_Read_By_UUID_Char
9.HGP_BHost_Write_Char
10.HGP_BHost_Set_Report
11.HGP_BHost_Write_Protocol_Mode
12.HGP_BHost_Data_Output
13.HGP_RHost_Enable
14.HGP_RHost_Disable
15.HGP_RHost_Read_Char
16.HGP_RHost_Read_By_UUID_Char
17.HGP_RHost_Read_Long_Char
18.HGP_RHost_Write_Char
19.HGP_RHost_Set_Report
20.HGP_RHost_Write_Protocol_Mode
21.HGP_RHost_Data_Output
22.HGP_RHost_Write_Control_Point
ESC Key: Menu exit
>> 1
CMD -> HGP_HDevice_Enable
Select Parameter No 0
Status(RBLE_OK)
>>
rBLE HGP EVENT (HDEVICE_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
```

図 5-27 HID Device 側ログ

```
C:\WINDOWS\system32\cmd.exe
rBLE HGP EVENT (HDEVICE_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE HGP EVENT (HDEVICE_REPORT_IND)
Connection Handle = 0
inst_idx = 0
device_type = 1
report_type = 1
value_size = 16
value[16] =
0x0F 0x0E 0x0D 0x0C 0x0B 0x0A 0x09 0x08 0x07 0x06 0x05 0x04 0x03 0x02 0x01 0x00
```

図 5-28 HID Device 側ログ(続き)

5.13 Scan Parameters Profile (ScPP)

ScPP の基本的な動作として、Scan Interval Window データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Client デバイス側のログを図 5-29に、Server デバイス側のログを図 5-30に示します。

内容	Scan Client 側(Command&Event)	Scan Server 側(Command&Event)
対向機と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
Server を有効		SPP_Server_Enable SPPS_ENABLE_COMP
Client を有効	SPP_Client_Enable SPPC_ENABLE_COMP	
Scan Interval Window データを送受信	SPP_Client_Write_Interval	SPPS_INTERVAL_WINDOW_CHG_EVT

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profileおよび5.6Security Managerを参照ください。

```
C:\> C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 6
-- BLE Sample Program Scan Parameters Profile Test Menu --
1.SPP_Server_Enable
2.SPP_Server_Disable
3.SPP_Server_Send_Refresh
4.SPP_Client_Enable
5.SPP_Client_Disable
6.SPP_Client_Write_Char
7.SPP_Client_Write_Interval
ESC Key: Menu exit
>> 4
CMD -> SPP_Client_Enable
Status(RBLE_OK)
>>
rBLE SPP EVENT (CLIENT_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Scan Parameters Service
  Start Handle      = 0x007F
  End Handle       = 0x0084

  intv_window_char_hdl = 0x0080
  intv_window_val_hdl = 0x0081
  intv_window_prop    = 0x04

  refresh_char_hdl   = 0x0082
  refresh_val_hdl    = 0x0083
  refresh_cfg_hdl    = 0x0084
  refresh_prop        = 0x10

>> 7
CMD -> SPP_Client_Write_Interval
Select Parameter No 0
  interval value = 0000, window value = 0000
Status(RBLE_OK)
>>
```

図 5-29 Scan Client 側ログ

```

C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 6
-- BLE Sample Program Scan Parameters Profile Test Menu --
1.SPP_Server_Enable
2.SPP_Server_Disable
3.SPP_Server_Send_Refresh
4.SPP_Client_Enable
5.SPP_Client_Disable
6.SPP_Client_Write_Char
7.SPP_Client_Write_Interval
ESC Key: Menu exit
>> 1
CMD -> SPP_Server_Enable
Status(RBLE_OK)
>>
rBLE SPP EVENT (SERVER_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE SPP EVENT (SERVER_INTERVAL_CHG_EVT)
Connection Handle = 0
    interval value = 0000
    window value    = 0000
>>

```

図 5-30 Scan Server 側ログ

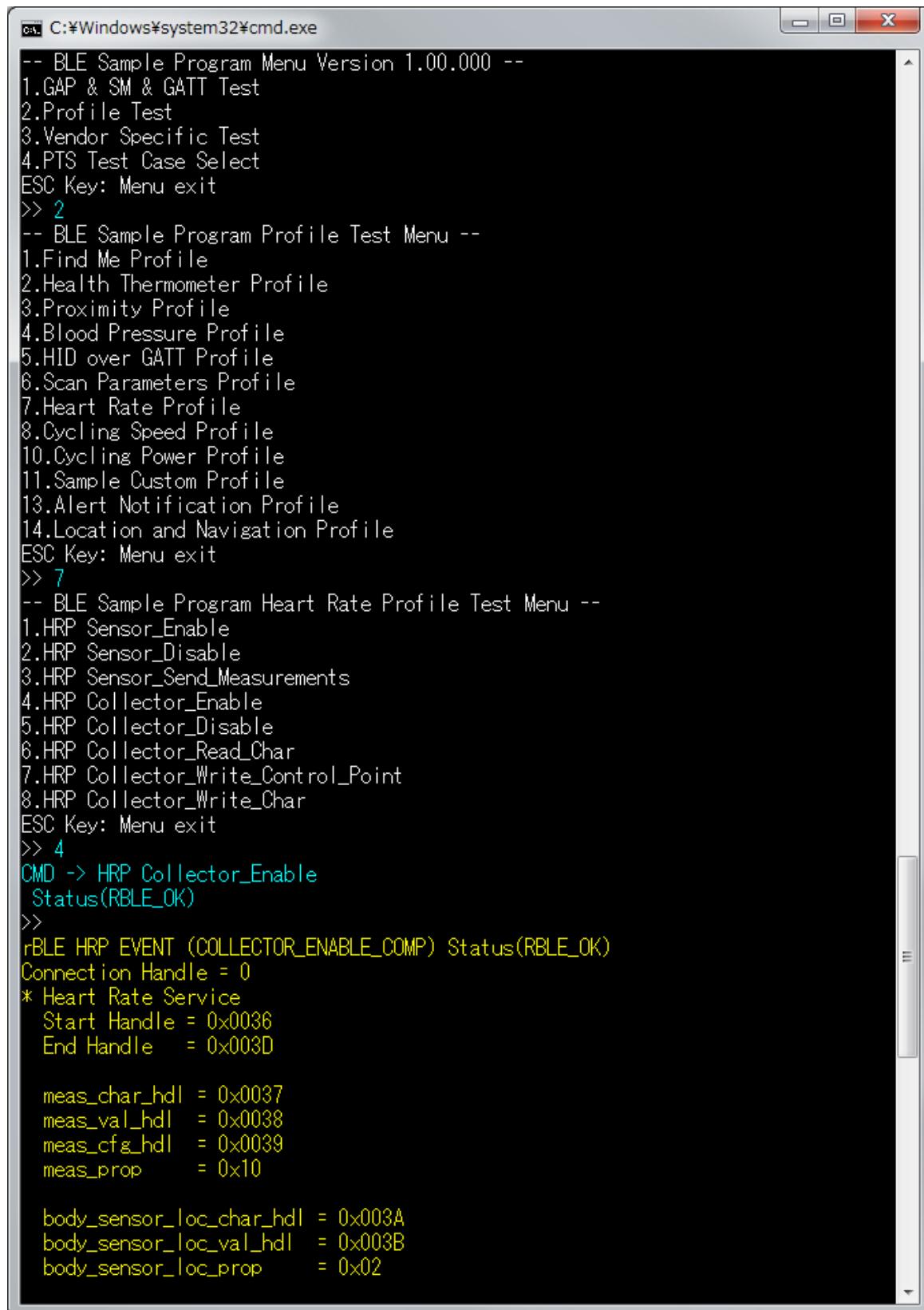
5.14 Heart Rate Profile (HRP)

HRP の基本的な動作として、Measurement データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Collector デバイス側のログを図 5-31 と図 5-32 に、Sensor デバイス側のログを図 5-33 に示します。

内容	Heart Rate Collector 側(Command&Event)	Heart Rate Sensor 側(Command&Event)
対向機と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
Sensorを有効		HRP_Sensor_Enable SENSOR_ENABLE_COMP
Collectorを有効	HRP_Collector_Enable COLLECTOR_ENABLE_COMP	
Notificationを有効	HRP_Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_NTF_IND
Measurementデータを受信	E COLLECTOR_MEASUREMENTS_NTF	HRP_Sensor_Send_Measurements SENSOR_SEND_MEASUREMENTS_COMMAND

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profileおよび5.6Security Managerを参照ください。



```
C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 7
-- BLE Sample Program Heart Rate Profile Test Menu --
1.HRP_Sensor_Enable
2.HRP_Sensor_Disable
3.HRP_Sensor_Send_Measurements
4.HRP_Collector_Enable
5.HRP_Collector_Disable
6.HRP_Collector_Read_Char
7.HRP_Collector_Write_Control_Point
8.HRP_Collector_Write_Char
ESC Key: Menu exit
>> 4
CMD -> HRP_Collector_Enable
| Status(RBLE_OK)
>>
rBLE HRP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Heart Rate Service
Start Handle = 0x0036
End Handle = 0x003D

meas_char_hdl = 0x0037
meas_val_hdl = 0x0038
meas_cfg_hdl = 0x0039
meas_prop = 0x10

body_sensor_loc_char_hdl = 0x003A
body_sensor_loc_val_hdl = 0x003B
body_sensor_loc_prop = 0x02
```

図 5-31 HRP Collector 側ログ

```

C:\Windows\system32\cmd.exe
control_point_char_hdl = 0x003C
control_point_val_hdl = 0x003D
control_point_prop = 0x08

* Device Information Service
Start Handle = 0x0025
End Handle = 0x0035

sys_id_char_hdl = 0x0026
sys_id_val_hdl = 0x0027
sys_id_prop = 0x02

model_nb_char_hdl = 0x0028
model_nb_val_hdl = 0x0029
model_nb_prop = 0x02

serial_nb_char_hdl = 0x002A
serial_nb_val_hdl = 0x002B
serial_nb_prop = 0x02

fw_rev_char_hdl = 0x002C
fw_rev_val_hdl = 0x002D
fw_rev_prop = 0x02

hw_rev_char_hdl = 0x002E
hw_rev_val_hdl = 0x002F
hw_rev_prop = 0x02

sw_rev_char_hdl = 0x0030
sw_rev_val_hdl = 0x0031
sw_rev_prop = 0x02

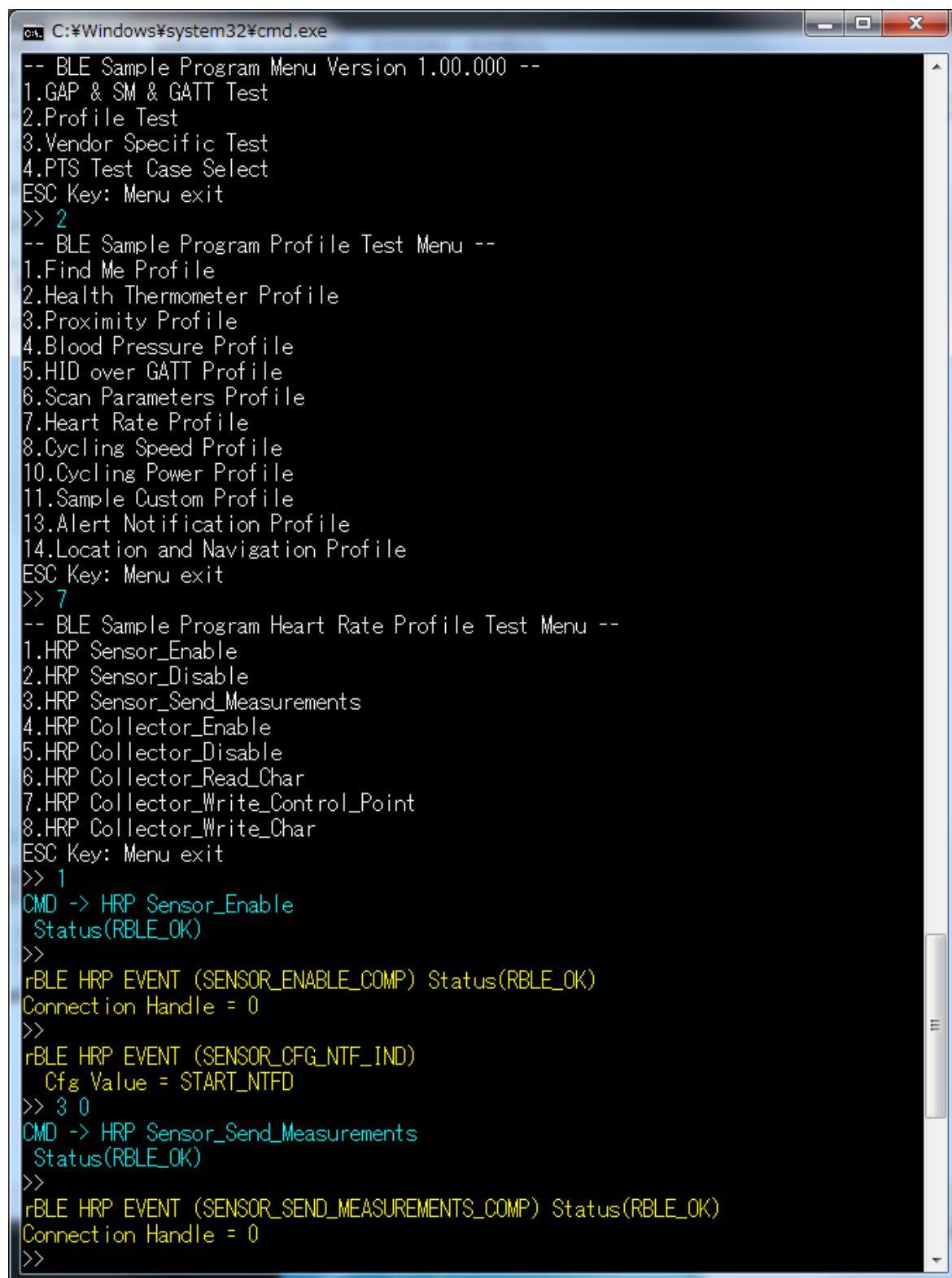
manuf_name_char_hdl = 0x0032
manuf_name_val_hdl = 0x0033
manuf_name_prop = 0x02

ieee_certif_char_hdl = 0x0034
ieee_certif_val_hdl = 0x0035
ieee_certif_prop = 0x02

>> 8 1
CMD -> HRP_Collector_Write_Char
Start Ntf(1)
Status(RBLE_OK)
>>
rBLE HRP EVENT (COLLECTOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE HRP EVENT (COLLECTOR_MEASUREMENTS_NTF)
Measure:ff(255)
Energy:0010(16)
RR Interval00:000a(10)
>>

```

図 5-32 HRP Collector 側ログ(続き)



C:\Windows\system32\cmd.exe

```
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 7
-- BLE Sample Program Heart Rate Profile Test Menu --
1.HRP_Sensor_Enable
2.HRP_Sensor_Disable
3.HRP_Sensor_Send_Measurements
4.HRP_Collector_Enable
5.HRP_Collector_Disable
6.HRP_Collector_Read_Char
7.HRP_Collector_Write_Control_Point
8.HRP_Collector_Write_Char
ESC Key: Menu exit
>> 1
CMD -> HRP_Sensor_Enable
Status(RBLE_OK)
>>
rBLE HRP EVENT (SENSOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE HRP EVENT (SENSOR_CFG_NTF_IND)
  Cfg Value = START_NTFD
>> 3 0
CMD -> HRP_Sensor_Send_Measurements
Status(RBLE_OK)
>>
rBLE HRP EVENT (SENSOR_SEND_MEASUREMENTS_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
```

図 5-33 HRP Sensor 側ログ

5.15 Cycling Speed and Cadence Profile (CSCP)

CSCP の基本的な動作として、CSC Measurement データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Collector デバイス側のログを図 5-34 と図 5-35、図 5-36 に、Sensor デバイス側のログを図 5-37 に示します。

内容	Cycling Speed and Cadence Collector 側 (Command&Event)	Cycling Speed and Cadence Sensor 側 (Command&Event)
対向機と接続	5.5 Generic Access Profile (GAP) および 5.6 Security Manager (SM) を参照	
Sensor を有効		CSCP_Sensor_Enable SENSOR_ENABLE_COMP
Collector を有効	CSCP_Collector_Enable COLLECTOR_ENABLE_COMP	
Notification を有効	CSCP_Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_INDNTF_IND
CSC Measurement データを送受信	COLLECTOR_MEASUREMENTS_NTF	CSCP_Sensor_Send_Measurements SENSOR_SEND_MEASUREMENTS_COMMAND

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5 Generic Access Profile および 5.6 Security Manager を参照ください。

```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 8

```

図 5-34 CSCP Collector 側ログ

C:\>Windows\system32\cmd.exe

```
-- BLE Sample Program Cycling Speed and Cadence Profile Test Menu --
1.CSCP_Sensor_Enable
2.CSCP_Sensor_Disable
3.CSCP_Sensor_Send_Measurements
4.CSCP_Sensor_Send_Sc_Control_Point
5.CSCP_Collector_Enable
6.CSCP_Collector_Disable
7.CSCP_Collector_Read_Char
8.CSCP_Collector_Write_Sc_Control_Point
9.CSCP_Collector_Write_Char
ESC Key: Menu exit
>> 5
CMD -> CSCP_Collector_Enable
Status(RBLE_OK)
>>
rBLE CSCP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Cycling Speed and Cadence Service
  Start Handle = 0x0025
  End Handle   = 0x002F

meas_char_hdl      = 0x0026
meas_val_hdl       = 0x0027
meas_cfg_hdl       = 0x0028
meas_prop          = 0x10

feature_char_hdl   = 0x0029
feature_val_hdl    = 0x002A
feature_prop        = 0x0002

sensor_loc_char_hdl = 0x002B
sensor_loc_val_hdl = 0x002C
sensor_loc_prop     = 0x0002

sc_control_point_char_hdl = 0x002D
sc_control_point_val_hdl = 0x002E
sc_control_point_cfg_hdl = 0x002F
sc_control_point_prop   = 0x28

* Device Information Service
  Start Handle   = 0x000F
  End Handle     = 0x001F

sys_id_char_hdl    = 0x0010
sys_id_val_hdl     = 0x0011
sys_id_prop         = 0x02

model_nb_char_hdl   = 0x0012
model_nb_val_hdl    = 0x0013
model_nb_prop        = 0x02
```

図 5-35 CSCP Collector 側ログ(続き 1)

```
C:\> C:\Windows\system32\cmd.exe
serial_nb_char_hdl    = 0x0014
serial_nb_val_hdl     = 0x0015
serial_nb_prop         = 0x02

fw_rev_char_hdl       = 0x0016
fw_rev_val_hdl        = 0x0017
fw_rev_prop            = 0x02

hw_rev_char_hdl       = 0x0018
hw_rev_val_hdl        = 0x0019
hw_rev_prop            = 0x02

sw_rev_char_hdl       = 0x001A
sw_rev_val_hdl        = 0x001B
sw_rev_prop            = 0x02

manuf_name_char_hdl   = 0x001C
manuf_name_val_hdl    = 0x001D
manuf_name_prop        = 0x02

ieee_certif_char_hdl  = 0x001E
ieee_certif_val_hdl   = 0x001F
ieee_certif_prop        = 0x02

>> 9 1 1
CMD -> CSCP_Collector_Write_Char
Start Ntf
Status(RBLE_OK)
>>
rBLE CSCP EVENT (COLLECTOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE CSCP EVENT (COLLECTOR_MEASUREMENTS_NTF)
Flag    :03
Wheel Rev   :0x00ff00ff(16711935)
Wheel Ev Time:0x0010(16)
Speed:-- (first event)
Crank Rev   :0x0200(512)
Crank Ev Time:0x0030(48)
Cadence:-- (first event)
>>
```

図 5-36 CSCP Collector 側ログ(続き 2)

```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 8
-- BLE Sample Program Cycling Speed and Cadence Profile Test Menu --
1.CSCP_Sensor_Enable
2.CSCP_Sensor_Disable
3.CSCP_Sensor_Send_Measurements
4.CSCP_Sensor_Send_Sc_Control_Point
5.CSCP_Collector_Enable
6.CSCP_Collector_Disable
7.CSCP_Collector_Read_Char
8.CSCP_Collector_Write_Sc_Control_Point
9.CSCP_Collector_Write_Char
ESC Key: Menu exit
>> 1
CMD -> CSCP_Sensor_Enable
Status(RBLE_OK)
>>
rBLE CSCP EVENT (SENSOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE CSCP EVENT (SENSOR_CFG_INDNTF_IND)
char_code = MEAS Cfg Value = START_NTF/IND
>> 3 0
CMD -> CSCP_Sensor_Send_Measurements
Status(RBLE_OK)
>>
rBLE CSCP EVENT (SENSOR_SEND_MEASUREMENTS_COMP) Status(RBLE_OK)
Connection Handle = 0
>>

```

図 5-37 CSCP Sensor 側ログ

5.16 Cycling Power Profile (CPP)

CPP の基本的な動作として、Cycling Power Measurement データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Collector デバイス側のログを図 5-38 と図 5-39、図 5-40 に、Sensor デバイス側のログを図 5-41 に示します。

内容	Cycling Power Collector 側 (Command&Event)	Cycling Power Sensor 側 (Command&Event)
対向機と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
Sensor を有効		CPP_Sensor_Enable SENSOR_ENABLE_COMP
Collector を有効	CPP_Collector_Enable COLLECTOR_ENABLE_COMP	
Notification を有効	CPP_Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_INDNTFBRD_IND
Cycling Power Measurement データを送受信	COLLECTOR_MEASUREMENTS_NTF	CPP_Sensor_Send_Measurements SENSOR_SEND_MEASUREMENTS_COMP

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profile および 5.6Security Manager を参照ください。

```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 10

```

図 5-38 CPP Collector 側ログ

```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Cycling Power Profile Test Menu --
1.CPP_Sensor_Enable
2.CPP_Sensor_Disable
3.CPP_Sensor_Send_Measurements
4.CPP_Sensor_Broadcast_Measurements
5.CPP_Sensor_Send_Vector
6.CPP_Sensor_Send_CP_Control_Point
7.CPP_Sensor_Send_Battery_Level
8.CPP_Sensor_Send_Write_Response
9.CPP_Collector_Enable
10.CPP_Collector_Disable
11.CPP_Collector_Read_Char
12.CPP_Collector_Write_CP_Control_Point
13.CPP_Collector_Write_Char
ESC Key: Menu exit
>> 9
CMD -> CPP_Collector_Enable
Status(RBLE_OK)
>>
rBLE CPP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Cycling Power Service
  Start Handle = 0x0030
  End Handle   = 0x003E

  meas_char_hdl      = 0x0031
  meas_val_hdl      = 0x0032
  meas_cfg_hdl      = 0x0033
  meas_brd_cfg_hdl = 0x0034
  meas_prop          = 0x11

  feature_char_hdl  = 0x0035
  feature_val_hdl   = 0x0036
  feature_prop       = 0x02

  sensor_loc_char_hdl = 0x0037
  sensor_loc_val_hdl = 0x0038
  sensor_loc_prop    = 0x02

  vector_char_hdl   = 0x0039
  vector_val_hdl    = 0x003A
  vector_cfg_hdl    = 0x003B
  vector_prop        = 0x10

  cp_cp_char_hdl   = 0x003C
  cp_cp_val_hdl    = 0x003D
  cp_cp_cfg_hdl    = 0x003E
  cp_cp_prop        = 0x28

* Device Information Service
  Start Handle     = 0x000F
  End Handle       = 0x001F

```

図 5-39 CPP Collector 側ログ(続き 1)

```
C:\> C:\Windows\system32\cmd.exe
sys_id_char_hdl      = 0x0010
sys_id_val_hdl       = 0x0011
sys_id_prop          = 0x02

model_nb_char_hdl    = 0x0012
model_nb_val_hdl     = 0x0013
model_nb_prop         = 0x02

serial_nb_char_hdl   = 0x0014
serial_nb_val_hdl    = 0x0015
serial_nb_prop        = 0x02

fw_rev_char_hdl      = 0x0016
fw_rev_val_hdl       = 0x0017
fw_rev_prop           = 0x02

hw_rev_char_hdl      = 0x0018
hw_rev_val_hdl       = 0x0019
hw_rev_prop           = 0x02

sw_rev_char_hdl      = 0x001A
sw_rev_val_hdl       = 0x001B
sw_rev_prop           = 0x02

manuf_name_char_hdl  = 0x001C
manuf_name_val_hdl   = 0x001D
manuf_name_prop       = 0x02

ieee_certif_char_hdl = 0x001E
ieee_certif_val_hdl  = 0x001F
ieee_certif_prop      = 0x02

* Battery Service
Start Handle = 0x0020
End Handle   = 0x0024

battery_lvl_char_hdl = 0x0021
battery_lvl_val_hdl  = 0x0022
battery_lvl_cfg_hdl  = 0x0023
battery_lvl_prop      = 0x12
>> 13 0 1
CMD -> CPP Collector_Write_Char
Select char:1, cfg:1
Status(RBLE_OK)
>>
rBLE CPP EVENT (COLLECTOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE CPP EVENT (COLLECTOR_MEASUREMENTS_NTF)
Connection Handle = 0
flags :0003
Instant Power      :100(0x0064)
Pedal Power Balance:170(0xaa)
>>
```

図 5-40 CPP Collector 側ログ(続き 2)

```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 10
-- BLE Sample Program Cycling Power Profile Test Menu --
1.CPP_Sensor_Enable
2.CPP_Sensor_Disable
3.CPP_Sensor_Send_Measurements
4.CPP_Sensor_Broadcast_Measurements
5.CPP_Sensor_Send_Vector
6.CPP_Sensor_Send_CPP_Control_Point
7.CPP_Sensor_Send_Battery_Level
8.CPP_Sensor_Send_Write_Response
9.CPP_Collector_Enable
10.CPP_Collector_Disable
11.CPP_Collector_Read_Char
12.CPP_Collector_Write_CPP_Control_Point
13.CPP_Collector_Write_Char
ESC Key: Menu exit
>> 1
CMD -> CPP_Sensor_Enable
Status(RBLE_OK)
>>
rBLE CPP EVENT (SENSOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE CPP EVENT (SENSOR_CFG_INDNTFBRD_IND)
Char Code = 1
Cfg Value = START
>> 3 0x3
CMD -> CPP_Sensor_Send_Measurements
Status(RBLE_OK)
>>
rBLE CPP EVENT (SENSOR_SEND_MEASUREMENTS_COMP) Status(RBLE_OK)
Connection Handle = 0
>>

```

図 5-41 CPP Sensor 側ログ

5.17 Alert Notification Profile (ANP)

ANP の基本的な動作として、New Alert データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Client デバイス側のログを図 5-42 と図 5-43 に、Server デバイス側のログを図 5-44 に示します。

内容	Alert Notification Client 側 (Command&Event)	Alert Notification Server 側 (Command&Event)
対向機と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
Sensor を有効		ANP Server_Enable SERVER_ENABLE_COMP
Collector を有効	ANP Client_Enable CLIENT_ENABLE_COMP	
Notification を有効	ANP Client_Write_Char CLIENT_WRITE_CHAR_RESPONSE	SERVER_CFG_NTF_IND
New Alert データを送受信	CLIENT_NEW_ALERT_NTF	ANP Sensor_Send_New_Alert SERVER_SEND_NEW_ALERT_COMP

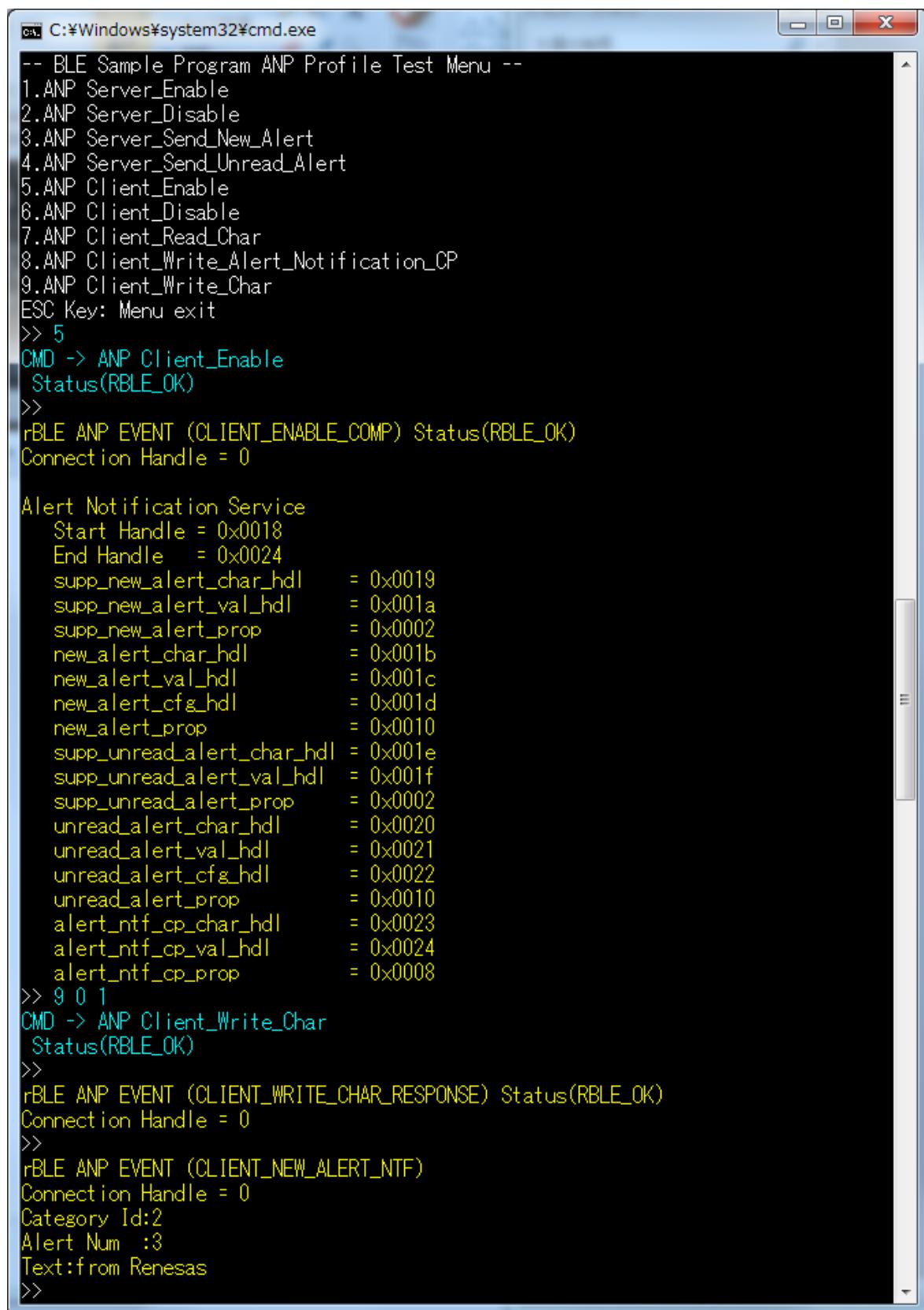
【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profile および 5.6Security Manager を参照ください。

```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 13

```

図 5-42 ANP Client 側ログ



C:\Windows\system32\cmd.exe

```
-- BLE Sample Program ANP Profile Test Menu --
1.ANP Server_Enable
2.ANP Server_Disable
3.ANP Server_Send_New_Alert
4.ANP Server_Send_Unread_Alert
5.ANP Client_Enable
6.ANP Client_Disable
7.ANP Client_Read_Char
8.ANP Client_Write_Alert_Notification_Cp
9.ANP Client_Write_Char
ESC Key: Menu exit
>> 5
CMD -> ANP Client_Enable
Status(RBLE_OK)
>>
rBLE ANP EVENT (CLIENT_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0

Alert Notification Service
Start Handle = 0x0018
End Handle   = 0x0024
supp_new_alert_char_hdl    = 0x0019
supp_new_alert_val_hdl    = 0x001a
supp_new_alert_prop        = 0x0002
new_alert_char_hdl         = 0x001b
new_alert_val_hdl          = 0x001c
new_alert_cfg_hdl          = 0x001d
new_alert_prop              = 0x0010
supp_unread_alert_char_hdl = 0x001e
supp_unread_alert_val_hdl = 0x001f
supp_unread_alert_prop     = 0x0002
unread_alert_char_hdl      = 0x0020
unread_alert_val_hdl       = 0x0021
unread_alert_cfg_hdl       = 0x0022
unread_alert_prop           = 0x0010
alert_ntf_cp_char_hdl     = 0x0023
alert_ntf_cp_val_hdl       = 0x0024
alert_ntf_cp_prop           = 0x0008
>> 9 0 1
CMD -> ANP Client_Write_Char
Status(RBLE_OK)
>>
rBLE ANP EVENT (CLIENT_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE ANP EVENT (CLIENT_NEW_ALERT_NTF)
Connection Handle = 0
Category Id:2
Alert Num :3
Text:from Renesas
>>
```

図 5-43 ANP Client 側ログ(続き)



```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 13
-- BLE Sample Program ANP Profile Test Menu --
1.ANP Server_Enable
2.ANP Server_Disable
3.ANP Server_Send_New_Alert
4.ANP Server_Send_Unread_Alert
5.ANP Client_Enable
6.ANP Client_Disable
7.ANP Client_Read_Char
8.ANP Client_Write_Alert_Notification_Cp
9.ANP Client_Write_Char
ESC Key: Menu exit
>> 1
CMD -> ANP Server_Enable
Status(RBLE_OK)
>>
rBLE ANP EVENT (SERVER_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE ANP EVENT (SERVER_CFG_NTF_IND)
Connection Handle = 0
char:0 cfg:1
>> 3 2 3
CMD -> ANP Server_Send_New_Alert
Status(RBLE_OK)
>>
rBLE ANP EVENT (SERVER_SEND_NEW_ALERT_COMP) Status(RBLE_OK)
Connection Handle = 0
>>

```

図 5-44 ANP Serve 側ログ

5.18 Location and Navigation Profile (LNP)

LNP の基本的な動作として、Location Speed データを送信する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の Collector デバイス側のログを図 5-45 と図 5-46、図 5-47 に、Sensor デバイス側のログを図 5-48 に示します。

内容	Location and Navigation Collector 側 (Command&Event)	Location and Navigation Sensor 側 (Command&Event)
対向機と接続	5.5Generic Access Profile (GAP)および5.6Security Manager (SM)を参照	
Sensor を有効		LNP_Sensor_Enable SENSOR_ENABLE_COMP
Collector を有効	LNP_Collector_Enable COLLECTOR_ENABLE_COMP	
Notification を有効	LNP_Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_INDNTF_IND
Measurement データを送受信	COLLECTOR_LOCATION_SPEED_NTF	LNP_Sensor_Send_Location_Speed SENSOR_SEND_LOCATION_SPEED_COMMAND

【注】 全ての Profile 層は GAP&SM コマンドにて対向機と接続し、接続時に通知されたハンドルを使用します。Profile 層のコマンドとイベントは、接続した後からのコマンドとイベントを記載しています。
対向機との接続については、5.5Generic Access Profile および 5.6Security Manager を参照ください。

```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 14

```

図 5-45 LNP Collector 側ログ

```
C:\Windows\system32\cmd.exe
-- BLE Sample Program LNP Profile Test Menu --
1.LNP_Sensor_Enable
2.LNP_Sensor_Disable
3.LNP_Sensor_Send_Location_Speed
4.LNP_Sensor_Set_Position_Quality
5.LNP_Sensor_Send_LN_Control_Point
6.LNP_Sensor_Send_Navigation
7.LNP_Sensor_Send_Battery_Level
8.LNP_Collector_Enable
9.LNP_Collector_Disable
10.LNP_Collector_Read_Char
11.LNP_Collector_Write_LN_Control_Point
12.LNP_Collector_Write_Char
ESC Key: Menu exit
>> 8
CMD -> LNP_Collector_Enable
Status(RBLE_OK)
>>
rBLE LNP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0

Location and Navigation Service
Start Handle = 0x003f
End Handle = 0x004c
ln_feature_char_hdl = 0x0040
ln_feature_val_hdl = 0x0041
ln_feature_prop = 0x0002
location_speed_char_hdl = 0x0042
location_speed_val_hdl = 0x0043
location_speed_cfg_hdl = 0x0044
location_speed_prop = 0x0010
position_quality_char_hdl = 0x0045
position_quality_val_hdl = 0x0046
position_quality_prop = 0x0002
ln_cp_char_hdl = 0x0047
ln_cp_val_hdl = 0x0048
ln_cp_cfg_hdl = 0x0049
ln_cp_prop = 0x0028
navigation_char_hdl = 0x004a
navigation_val_hdl = 0x004b
navigation_cfg_hdl = 0x004c
navigation_prop = 0x0010
* Device Information Service
Start Handle = 0x000F
End Handle = 0x001F
```

図 5-46 LNP Collector 側ログ(続き 1)

```
C:\Windows\system32\cmd.exe
sys_id_char_hdl      = 0x0010
sys_id_val_hdl       = 0x0011
sys_id_prop          = 0x02

model_nb_char_hdl    = 0x0012
model_nb_val_hdl     = 0x0013
model_nb_prop         = 0x02

serial_nb_char_hdl   = 0x0014
serial_nb_val_hdl    = 0x0015
serial_nb_prop        = 0x02

fw_rev_char_hdl      = 0x0016
fw_rev_val_hdl       = 0x0017
fw_rev_prop           = 0x02

hw_rev_char_hdl      = 0x0018
hw_rev_val_hdl       = 0x0019
hw_rev_prop           = 0x02

sw_rev_char_hdl      = 0x001A
sw_rev_val_hdl       = 0x001B
sw_rev_prop           = 0x02

manuf_name_char_hdl  = 0x001C
manuf_name_val_hdl   = 0x001D
manuf_name_prop       = 0x02

ieee_certif_char_hdl = 0x001E
ieee_certif_val_hdl  = 0x001F
ieee_certif_prop      = 0x02

* Battery Service
Start Handle = 0x0020
End Handle   = 0x0024

battery_lvl_char_hdl = 0x0021
battery_lvl_val_hdl  = 0x0022
battery_lvl_cf_g_hdl = 0x0023
battery_lvl_prop      = 0x12
>> 12 0 1
CMD -> LNP_Collector_Write_Char
Status(RBLE_OK)
>>
rBLE LNP EVENT (COLLECTOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE LNP EVENT (COLLECTOR_LOCATION_SPEED_NTF)
Connection Handle = 0
Flags:0x000e
total_distance:200000(0x00030d40)
latitude       :10500000000(0x3e95ba80)
longitude      :14300000000(0x553c1180)
elevation      :-59(0xfffffff5)
>>
```

図 5-47 LNP Collector 側ログ(続き 2)

```

C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 14
-- BLE Sample Program LNP Profile Test Menu --
1.LNP_Sensor_Enable
2.LNP_Sensor_Disable
3.LNP_Sensor_Send_Location_Speed
4.LNP_Sensor_Set_Position_Quality
5.LNP_Sensor_Send_LN_Control_Point
6.LNP_Sensor_Send_Navigation
7.LNP_Sensor_Send_Battery_Level
8.LNP_Collector_Enable
9.LNP_Collector_Disable
10.LNP_Collector_Read_Char
11.LNP_Collector_Write_LN_Control_Point
12.LNP_Collector_Write_Char
ESC Key: Menu exit
>> 1
CMD -> LNP_Sensor_Enable
Status(RBLE_OK)
>>
rBLE LNP EVENT (SENSOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE LNP EVENT (SENSOR_CFG_INDNTF_IND)
Connection Handle = 0
char:0 cfg:1
>> 3 0xe
CMD -> LNP_Sensor_Send_Location_Speed
Status(RBLE_OK)
>>
rBLE LNP EVENT (SENSOR_SEND_LOCATION_SPEED_COMP) Status(RBLE_OK)
Connection Handle = 0
>>

```

図 5-48 LNP Sensor 側ログ

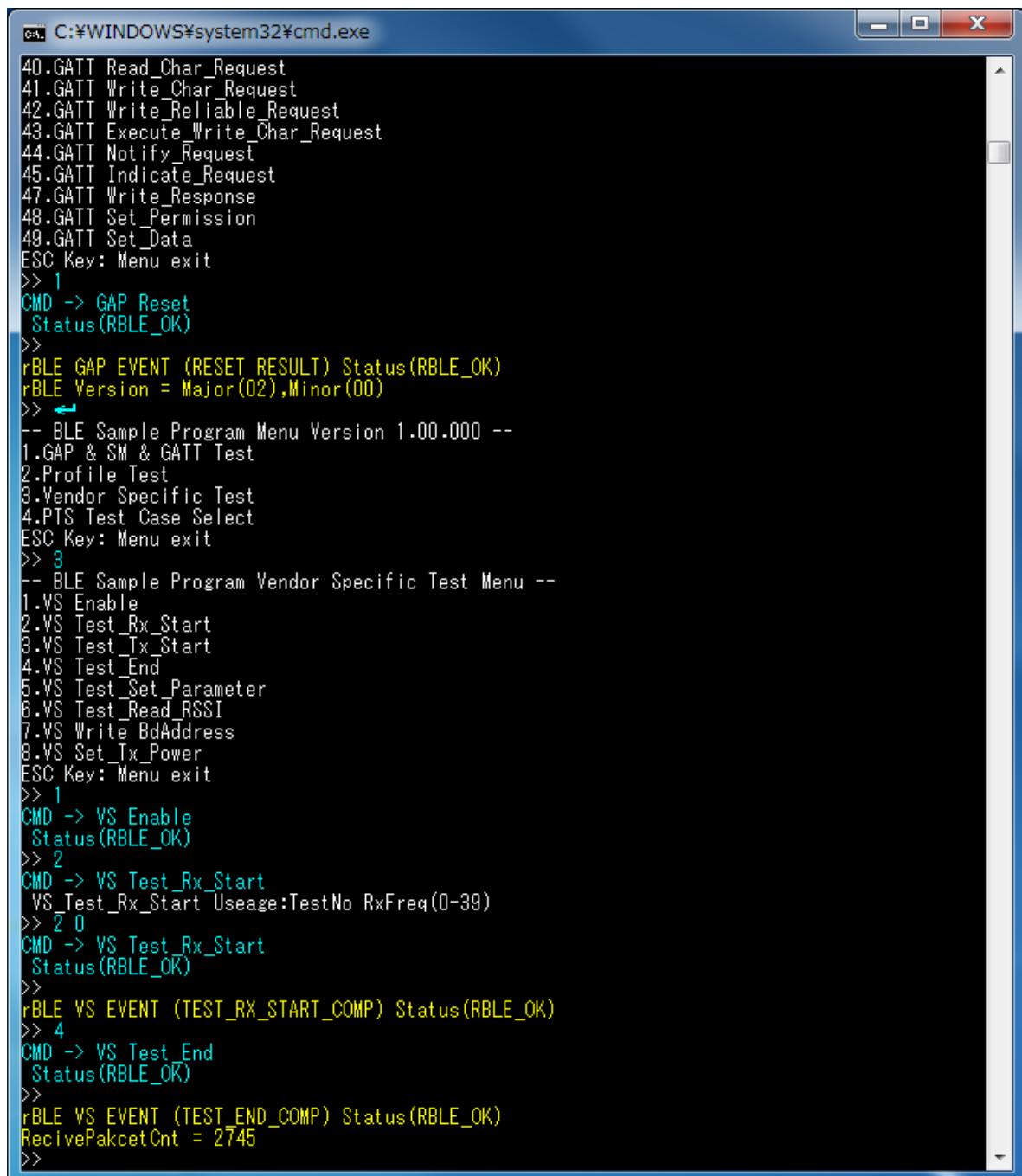
5.19 Vendor Specific (VS)

VS の基本的な動作として、Direct Test モードを使用する場合のコマンドとイベントを以下の表に示します。また、下表の処理を実行した際の送信デバイス側のログを図 5-49に、受信デバイス側のログを図 5-50に示します。

内容	送信側(Command&Event)	受信側(Command&Event)
VS を有効	VS Enable	VS Enable
テスト開始	VS Test_Tx_Start TEST_TX_START_COMP	VS Test_Rx_Start TEST_RX_START_COMP
テスト終了	VS Test_End TEST_END_COMP	VS Test_End TEST_END_COMP

```
C:\> C:\WINDOWS\system32\cmd.exe
48.GATT Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
Status(RBLE_OK)
>>
rBLE GAP EVENT (RESET_RESULT) Status(RBLE_OK)
rBLE Version = Major(02),Minor(00)
>> ←
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 3
-- BLE Sample Program Vendor Specific Test Menu --
1.VS Enable
2.VS Test_Rx_Start
3.VS Test_Tx_Start
4.VS Test_End
5.VS Test_Set_Parameter
6.VS Test_Read_RSSI
7.VS Write BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
Status(RBLE_OK)
>> 3
CMD -> VS Test_Tx_Start
VS_Test_Tx_Start Useage:TestNo TxFreq(0-39) DataLen(1-37) PayloadType(0-7)
[PayloadType]
0: Pseudo-Random bit sequence 9,
1: Pattern of alternating bits '11110000',
2: Pattern of alternating bits '10101010',
3: Pseudo-Random bit sequence 15
4: Pattern of All '1' bits
5: Pattern of All '0' bits
6: Pattern of alternating bits '00001111'
7: Pattern of alternating bits '0101'
>> 3 0 27 0
CMD -> VS Test_Tx_Start
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_TX_START_COMP) Status(RBLE_OK)
>> 4
CMD -> VS Test_End
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
ReceivePakcetCnt = 0
>>
```

図 5-49 Direct Test Mode 送信側ログ



C:\> C:\WINDOWS\system32\cmd.exe

```
40.GATT Read_Char_Request
41.GATT Write_Char_Request
42.GATT Write_Reliable_Request
43.GATT Execute_Write_Char_Request
44.GATT Notify_Request
45.GATT Indicate_Request
46.GATT Write_Response
48.GATT Set_Permission
49.GATT Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
Status(RBLE_OK)
>>
rBLE GAP EVENT (RESET_RESULT) Status(RBLE_OK)
rBLE Version = Major(02),Minor(00)
>> ↲
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 3
-- BLE Sample Program Vendor Specific Test Menu --
1.VS Enable
2.VS Test_Rx_Start
3.VS Test_Tx_Start
4.VS Test_End
5.VS Test_Set_Parameter
6.VS Test_Read_RSSI
7.VS Write_BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
Status(RBLE_OK)
>> 2
CMD -> VS Test_Rx_Start
VS_Test_Rx_Start Useage:TestNo RxFreq(0-39)
>> 2 0
CMD -> VS Test_Rx_Start
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLE_OK)
>> 4
CMD -> VS Test_End
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
ReceivePakcetCnt = 2745
>>
```

図 5-50 Direct Test Mode 受信側ログ

6. 簡易サンプルプログラムの使用方法

簡易サンプルプログラムは、BLE ソフトウェアの使用方法を示すサンプルプログラムです。前章までのサンプルプログラムとは異なり、少數の簡易な機能のみを含むため、動作や実装を容易に理解することができます。

簡易サンプルプログラムは、「Embedded 構成サンプルアプリケーション (r01an3319)」の Peripheral 向けサンプルアプリケーションと同一です。詳細に関しては、「Embedded 構成サンプルアプリケーション (r01an3319)」のアプリケーションノートを参照してください。

6.1 構成

簡易サンプルプログラムは、Embedded 構成としてのみ動作します。Modem 構成では動作しません。

6.2 HEX ファイルの入手

HEX ファイルの入手は、事前にビルド済みの HEX ファイルを使用する方法と、ソースコードをビルドする方法があります。

事前にビルド済みの HEX ファイルは、/Renesas/BLE_Software_Ver_X_XX/RL78_G1D/ROM_File に格納されています。BLE ソフトウェアがサポートする各コンパイラ (CC-RL, IAR, CA78K0R) を使用してビルドした HEX ファイルを用意しています。

ソースコードをビルドする場合は、/Renesas/BLE_Software_Ver_X_XX/RL78_G1D/renesas/tools/simple_sample 配下に格納されているプロジェクトファイルを使用してください。BLE ソフトウェアがサポートしている各開発環境 (e² studio, CS+, IAR Embedded Workbench) 向けのプロジェクトファイルを用意しています。

6.3 動作概要

6.2章で入手した HEX ファイルを RL78/G1D 評価ボードに書き込み、評価ボードをリセットします。リセット後、RL78/G1D 評価ボードの LED1/LED2 が交互に点滅することを確認してください。

簡易サンプルプログラムは、動作開始すると自動的に Advertise を開始します。ユーザは、RL78/G1D 評価ボードと対向デバイス間の接続を確立した後、以下の操作を行えます。

- ・ 対向デバイスから RL78/G1D 評価ボードの LED4 の点灯・消灯を制御する
- ・ 対向デバイスで RL78/G1D 評価ボードの SW4 の押下・開放状態を受け取る

簡易サンプルプログラムの動作確認には、対向デバイスが必要です。本書では、対向デバイスとして、スマートフォン (iOS デバイスおよび Android デバイス) を使用する際の手順について記載します。

6.4 Android デバイスでの確認方法

Android デバイスを使用する場合について記載します。Android アプリケーションとして、「BLE Scanner Version 3.6」を使用します。BLE Scanner に関しては、以下の URL を参照してください。

<https://play.google.com/store/apps/details?id=com.macdom.ble.blescanner&hl=en>

- 1) BLE Scanner を起動し、周辺デバイスの Scan を行います。Scan により発見したデバイスのリストからデバイス名が「REL-BLE」であるデバイスを選択します。これにより、RL78/G1D 評価ボードとの接続が確立されます（図 a）。
- 2) Service のリストから CUSTOM SERVICE (UUID: 5BC1B9F7-A1F1-40AF-9043-C43692C18D7A) を選択します（図 b）。
- 3) RL78/G1D 評価ボード上の SW4 状態を受信する方法について記載します。
SW4 状態の受信は、CUSTOM CHARACTERISTIC (UUID: 5BC18D80-A1F1-40AF-9043-C43692C18D7A) を通じて行います。(N)マークを押下すると（図 c 上側矢印）、RL78/G1D 評価ボードは SW 状態の送信を開始します。以降、RL78/G1D 評価ボード上の SW4 状態にしたがって「Hex」の値が変化します（図 c 下側矢印）。RL78/G1D 評価ボードの SW4 を押下すると 0x01 が、開放すると 0x00 が表示されます。RL78/G1D 評価ボードからの SW4 状態の送信を停止したい場合は、再度(N)マークを押下します。
- 4) RL78/G1D 評価ボード上の LED4 を制御する方法について記載します。
LED4 の制御は、CUSTOM CHARACTERISTIC (UUID: 5BC143EE-A1F1-40AF-9043-C43692C18D7A) を通じて行います。(W)マークを押下すると（図 d）、「Write Value」ダイアログが表示されるので、「Byte Array」を選択した後、「01」を入力し、「OK」を押下します（図 e）。これにより、RL78/G1D 評価ボード上の LED4 が点灯します。消灯する場合は、Write する値を「00」にします。

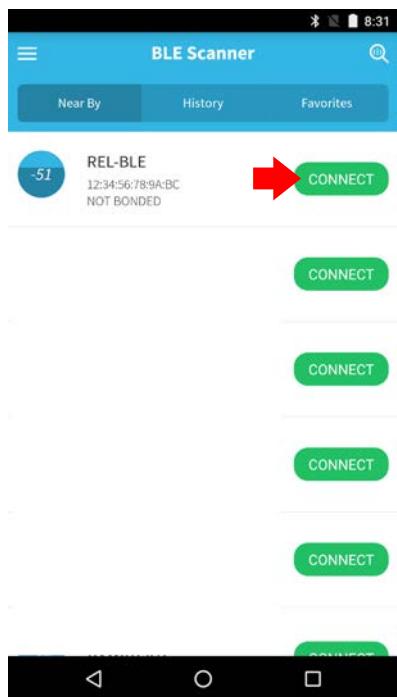


図 a

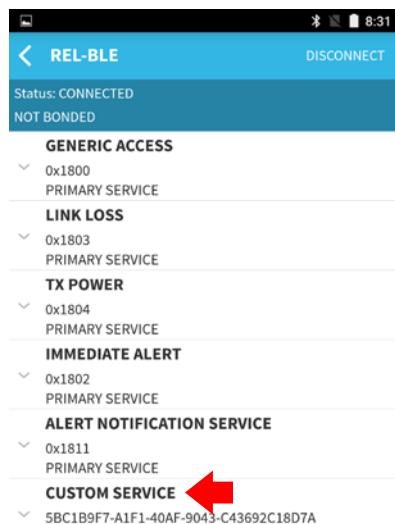


図 b

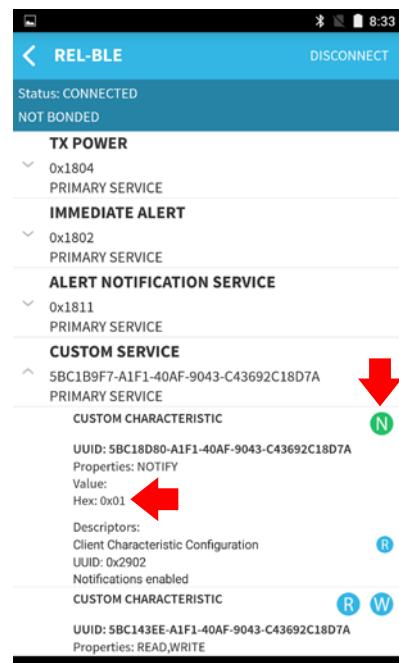


図 c

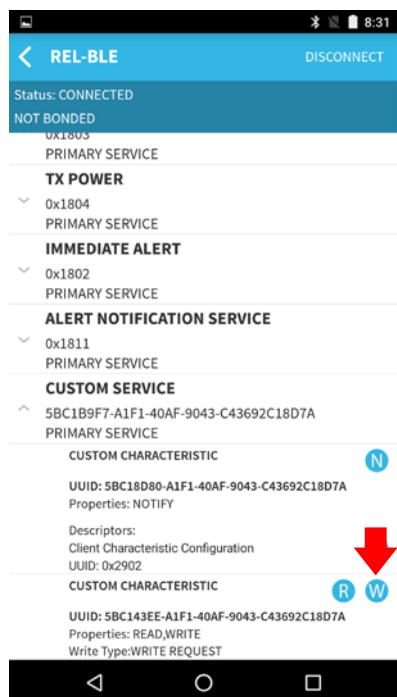


図 d

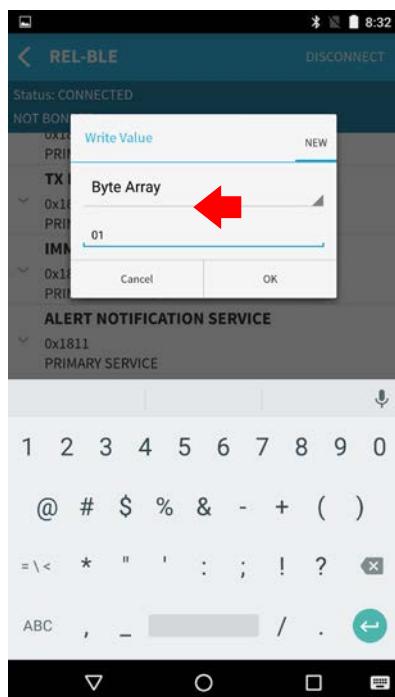


図 e

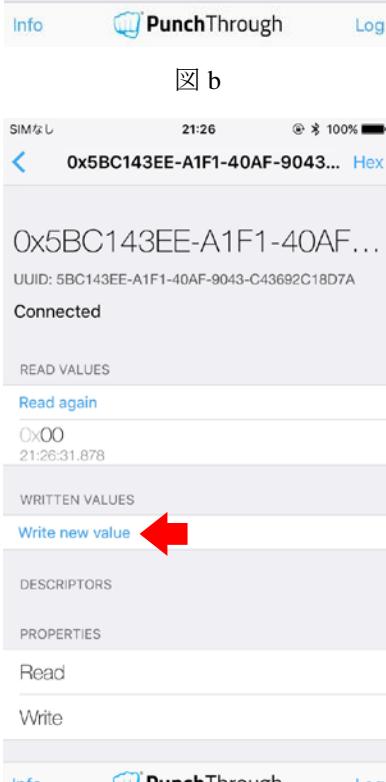
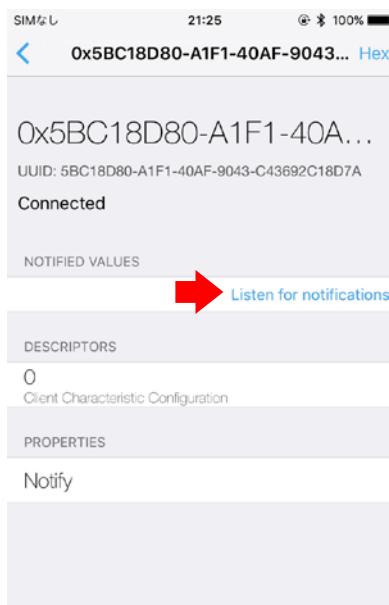
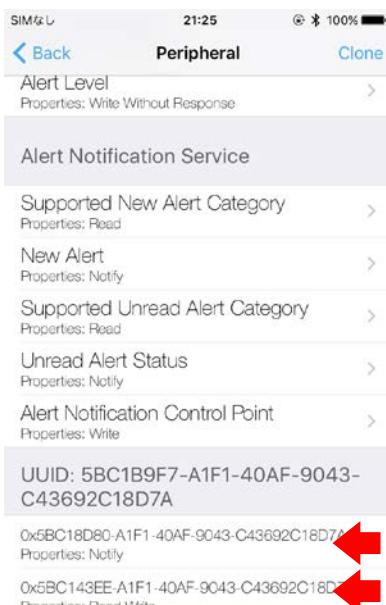
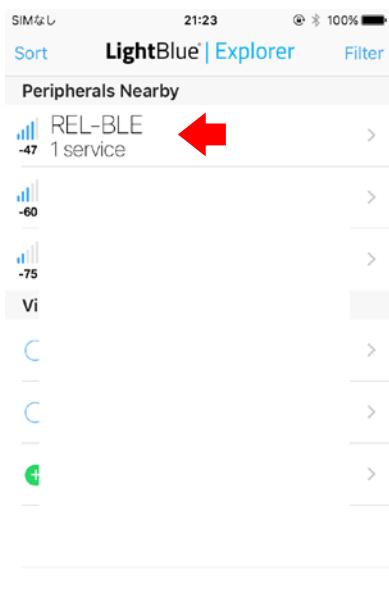
6.5 iOS デバイスでの確認方法

iOS デバイスを使用する場合について記載します。iOS アプリケーションとして、「LightBlue Version 2.4.0」を使用します。LightBlue については、以下の URL を参照してください。

<https://itunes.apple.com/en/app/lightblue-explorer-bluetooth/id557428110?mt=8>

- 1) LightBlue を起動し、周辺デバイスの Scan を行います。Scan により発見したデバイスのリストからデバイス名が「REL-BLE」のデバイスを選択します。これにより RL78/G1D 評価ボードとの接続が確立されます（図 a）。ただし、iOS はデバイス名をキャッシュするため、「REL-BLE」とは異なるデバイス名が表示される場合があります。
- 2) RL78/G1D 評価ボード上の SW4 状態を受信する方法について記載します。
SW4 状態を受信する場合、Characteristic (UUID:5BC18D80-A1F1-40AF-9043-C43692C18D7A) を選択します（図 b 上側矢印）。「Listen for notifications」を押下すると（図 c）、RL78/G1D 評価ボードは SW4 の状態の送信を開始します。以降、RL78/G1D 評価ボード上の SW4 状態にしたがって、「NOTIFIED VALUES」の値が変化します（図 d 下側矢印）。RL78/G1D 評価ボードの SW4 を押下すると 0x01 が、開放すると 0x00 が表示されます。RL78/G1D 評価ボードからの SW4 状態の送信を停止したい場合は、「Stop Listening」を押下します（図 d 上側矢印）。
- 3) RL78/G1D 評価ボード上の LED4 を制御する方法について記載します。
Service のリストから Characteristic (UUID:5BC143EE-A1F1-40AF-9043-C43692C18D7A) を選択します（図 b 下側矢印）。「write new value」を選択すると（図 e）、「Edit Value」ダイアログが表示されるので、「01」を入力し、「Done」を押下します（図 f）。これにより、RL78/G1D 評価ボード上の LED4 が点灯します。消灯する場合は、Write する値を「00」にします。

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート



D	E	F
A	B	C
7	8	9
4	5	6
1	2	3
<input type="button" value="X"/>	0	Done

図 d

図 e

図 f

7. 付録

7.1 Windows 向けサンプルプログラムによる BLE-MCU への送受信動作

APP MCU で動作するアプリケーションは、rBLE_Host を介して BLE MCU と BLE サービスのやり取りが行われます。APP MCU と BLE MCU は物理的に UART、CSI、IIC いずれかで接続され、rBLE_Host の制御により RSCIP (Renesas Serial Communication Interface Protocol)を使用した通信が行われます。

図 7-1に Windows 向けサンプルプログラムの内部構成を示します。Windows 向けサンプルプログラムは、図に示すようにメイン処理からコマンド入出力関数および rBLE ソフトを呼び出して動作します。

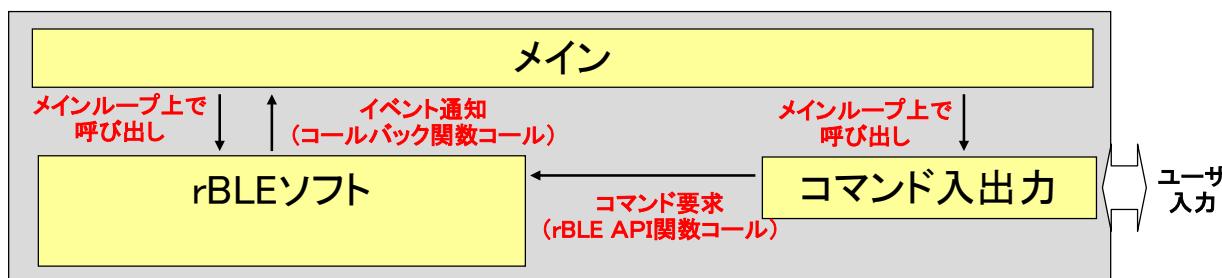


図 7-1 サンプルプログラムの内部構成

rBLE の送受信処理は、メイン処理から rBLE_Run 関数を呼び出すことで処理されます。

rBLE_Run 関数は、BLE-MCU への送信用バッファをチェックし、送信データが存在した場合、RSCIP ドライバの送信関数を呼び出し、BLE-MCU からの受信用バッファに受信データが存在した場合は、データを解析し、イベント情報に基づき登録されたアプリケーション関数を呼び出します。

また、イベント通知が存在した場合は、該当するイベントに応じた RSCIP 関数を呼び出します。図 7-2に内部処理のシーケンスを示します。

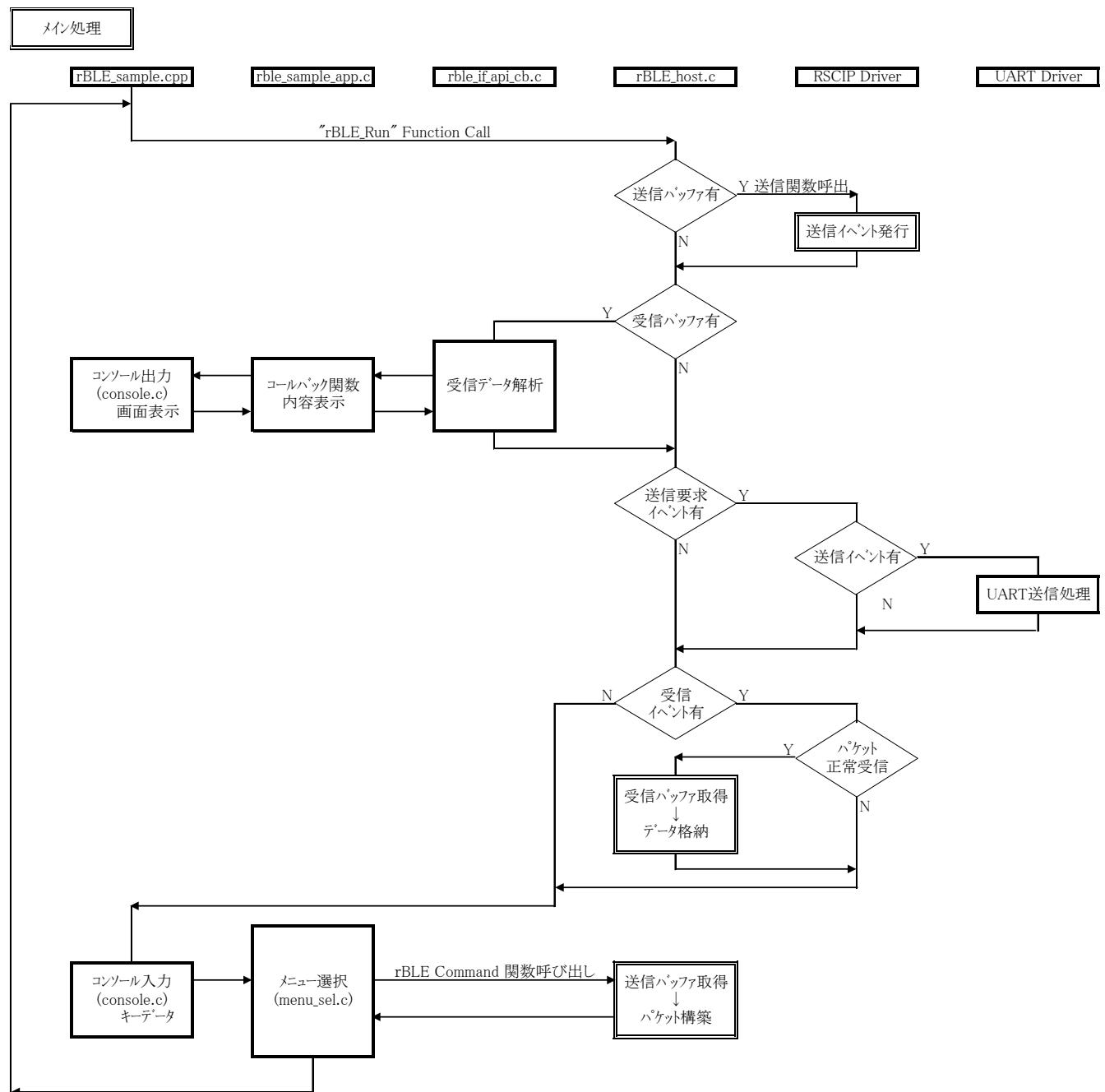


図 7-2 サンプルプログラムの内部処理(メイン処理)

図 7-3にRSCIPからの送信イベント発行時のシーケンスを示します。RSCIPは再送処理からの送信要求とユーザからの送信要求を一箇所で処理するため、双方から送信要求が発生した場合、送信イベント要求をrBLEに発行します。rBLEは図 7-2に記載していますように、送信イベント要求があった場合、RSCIPの送信関数を呼び出します。

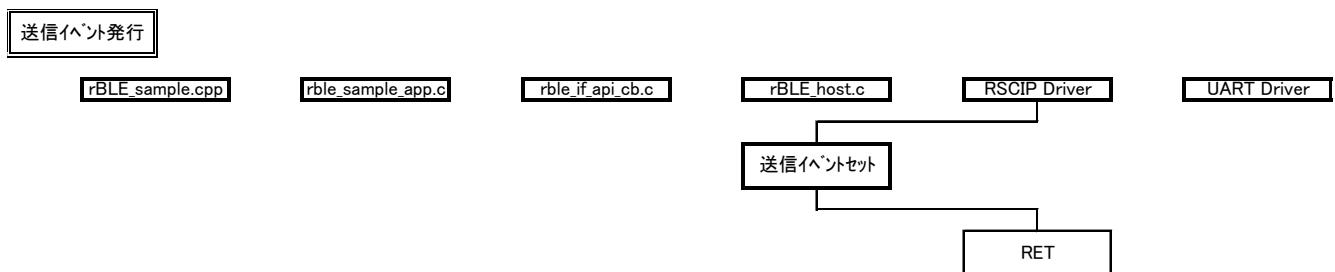


図 7-3 サンプルプログラムの内部処理(送信イベント発行)

図 7-4にRSCIPからの受信イベント発行時のシーケンスを示します。RSCIPはSerial_Driverからのデータ受信通知が割り込みから呼び出されることを考慮して、パケット受信が完了した場合、受信イベント要求をrBLEに発行します。rBLEは図 7-2に記載していますように、受信イベント要求があった場合、RSCIPのパケット受信処理関数を呼び出します。

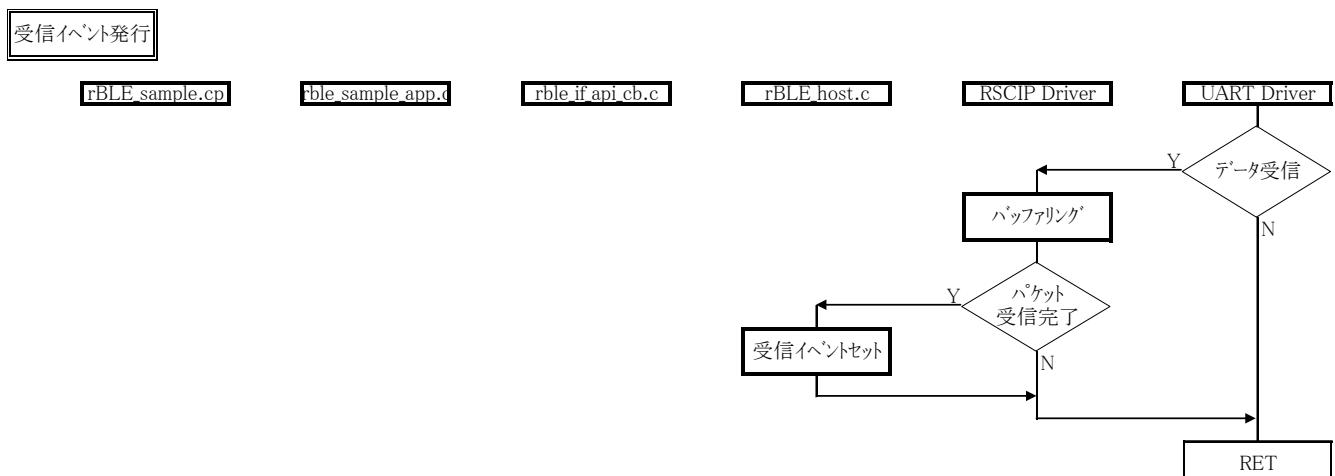


図 7-4 サンプルプログラムの内部処理(受信イベント発行)

7.2 APP MCU のシリアル通信ドライバの要件と実装フローチャート

Modem 構成時のアプリケーション開発に必要となる APP MCU の要件を以下に纏めます。

- H/W リソース

BLE MCU との通信用にシリアル通信のための UART、CSI(クロック同期式シリアル通信)、IIC いずれか 1ch 必要となります。

- タイマー

RSCIP ドライバ(rBLE_Host の中の RSCIP 実装部)にてタイムアウト機能が必要となります。

- シリアル通信ドライバ

UART、CSI、IIC にて通信するためのシリアル通信ドライバはお客様にご用意いただく必要があります。

また、ご用意いただくシリアル通信ドライバには RSCIP ドライバとのインターフェースとして以下に示す関数が必要となります。

関数名	BOOL serial_init (void)	
概要	シリアル通信ドライバ初期化用関数	
説明	シリアル通信の初期化を行う関数です。 Bluetooth Low Energy プロトコルスタック ユーザーズマニュアルに記載されている設定値にてシリアル通信を初期化してください。	
引数	なし	
戻り値	TRUE	シリアル通信初期化処理正常終了
	FALSE	シリアル通信初期化処理エラー

関数名	BOOL serial_write (uint8_t *bufptr, uint16_t size)	
概要	シリアル送信用関数	
説明	指定データのシリアル送信を行うノンブロッキング関数です。 引数で指定された*bufptr の指示する領域のデータを size 分送信してください。 送信完了時には下記 RSCIP ドライバへの送信完了通知関数を呼び出してください。 void RSCIP_Uart_Tx_Done(void); UART 2 線接続方式以外の方式を使用する場合は、Bluetooth Low Energy プロトコルスタック ユーザーズマニュアルに記載された送信動作時のハンドシェイク手順を行ってください。	
引数	uint8_t *bufptr	送信データバッファへのポインタ
	uint16_t size	送信データサイズ
戻り値	TRUE	シリアル送信処理正常終了
	FALSE	シリアル送信処理エラー
補足	本関数は割り込みから呼び出される場合があります。 サンプルプログラムでは最低限必要となる処理以外は、メインループから呼び出される関数 rBLE_Run()にて送信処理を行っています。	

関数名	BOOL serial_read (uint8_t *bufptr, uint16_t size)	
概要	シリアル受信用関数	

説明	<p>指定サイズのシリアル受信を行うノンブロッキング関数です。</p> <p>引数で指定された size 分のデータを受信し、*bufptr の指し示す領域に受信データを格納してください。</p> <p>受信完了時には下記 RSCIP ドライバへの受信完了通知関数を呼び出してください。</p> <pre>void RSCIP_Uart_Rx_Done (void); UART 2 線接続方式以外の方式を使用する場合は、Bluetooth Low Energy プロトコルスタック ユーザーズマニュアルに記載された受信動作時のハンドシェイク手順を行ってください。また、RSCIP への受信完了通知関数の呼び出し後、連続してデータを取得するかを判断するため、下記 RSCIP ドライバの受信状態取得関数を呼び出し、返却値を確認してください。</pre> <pre>BOOL RSCIP_Uart_Rx_Idle(void); FALSE : パケット受信が継続状態であることを示します。 TRUE : パケット受信が完了状態であり、次のパケットの先頭待ちであることを示します。</pre>				
引数	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">uint8_t *bufptr</td> <td style="padding: 2px;">受信データバッファへのポインタ</td> </tr> <tr> <td style="padding: 2px;">uint16_t size</td> <td style="padding: 2px;">受信要求データサイズ</td> </tr> </table>	uint8_t *bufptr	受信データバッファへのポインタ	uint16_t size	受信要求データサイズ
uint8_t *bufptr	受信データバッファへのポインタ				
uint16_t size	受信要求データサイズ				
戻り値	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">TRUE</td> <td style="padding: 2px;">シリアル受信処理正常終了</td> </tr> <tr> <td style="padding: 2px;">FALSE</td> <td style="padding: 2px;">シリアル受信処理エラー</td> </tr> </table>	TRUE	シリアル受信処理正常終了	FALSE	シリアル受信処理エラー
TRUE	シリアル受信処理正常終了				
FALSE	シリアル受信処理エラー				
補足	<p>本関数は割り込みから呼び出される場合があります。</p> <p>サンプルプログラムでは最低限必要となる処理以外は、メインループから呼び出される関数 rBLE_Run()にて送信処理を行っています。</p>				

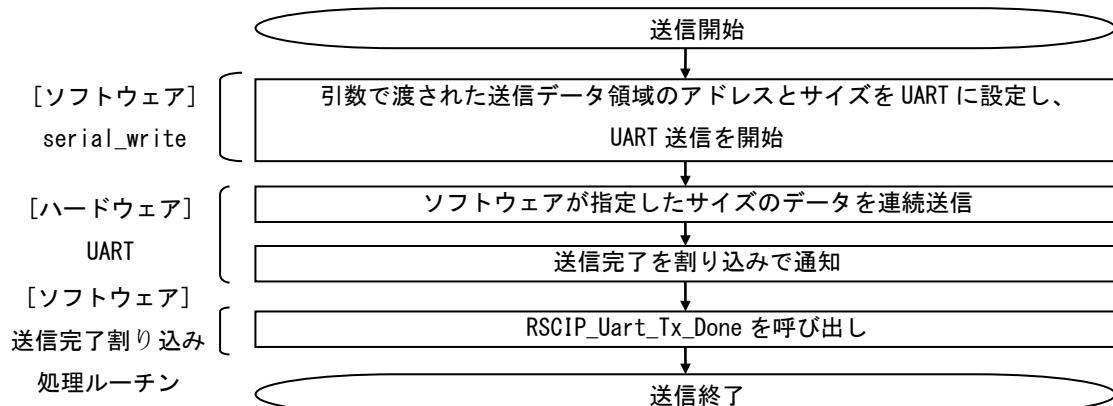
関数名	void serial_exit (void)
概要	シリアル通信ドライバ終了関数
説明	シリアル通信ドライバの終了処理を行う関数です。 シリアル通信ドライバの終了処理を実行ください。
引数	なし
戻り値	なし

Modem 構成時のシリアル通信では、下記の接続方式があります。シリアル通信の接続方式の詳細につきましては Bluetooth Low Energy プロトコルスタック ユーザーズマニュアルをご参照ください。後述するシリアル通信ドライバの実装フローチャート例をご参考に、ご使用のリソースに合った実装をしてください。

シリアル	接続方式	送信実装例	受信実装例
UART	2 線接続方式	後述の「7.2.1 UART 2 線接続方式の送信実装例」をご参照ください。	後述の「7.2.2 UART 2 線接続方式の受信実装例」をご参照ください。
	3 線接続方式	後述の「7.2.3 UART 3 線接続方式の送信実装例」をご参照ください。	後述の「7.2.5 UART 3 線接続方式、2 線分岐接続方式の受信実装例」をご参照ください。
	2 線分岐接続方式	後述の「7.2.4 UART 2 線分岐接続方式の送信実装例」をご参照ください。	
CSI	4 線接続方式	後述の「7.2.6 CSI 4 線接続方式の送信実装例」をご参照ください。	後述の「7.2.8 CSI の受信実装例」をご参照ください。
	5 線接続方式	後述の「7.2.7 CSI 5 線接続方式の送信実装例」をご参照ください。	
IIC	3 線接続方式	後述の「7.2.9 IIC 3 線接続方式の送信実装例」をご参照ください。	後述の「7.2.10 IIC 3 線接続方式の受信実装例」をご参照ください。

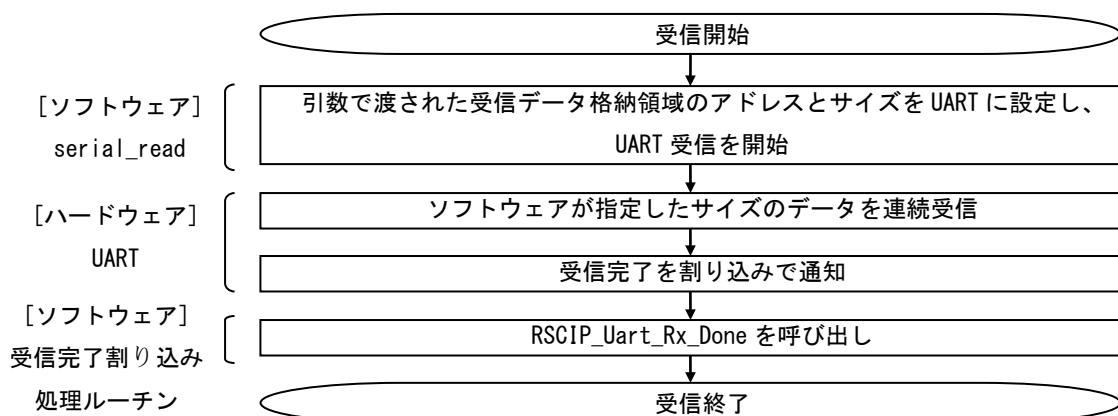
7.2.1 UART 2線接続方式の送信実装例

UART における 2 線接続方式を実現するための送信実装例を以下に示します。前提条件として、使用する UART ハードウェアは、送信データ領域のアドレスとサイズをレジスタに設定し送信開始すると、指定した送信データを送信し、送信完了後に割り込みが発生することとします。



7.2.2 UART 2 線接続方式の受信実装例

UART における 2 線接続方式、3 線接続方式、2 線分岐接続方式を実現するための受信実装例を以下に示します。前提条件として、使用する UART ハードウェアは、受信データ領域のアドレスとサイズをレジスタに設定し受信開始すると、指定したサイズのデータを受信し、受信完了後に割り込みが発生することとします。

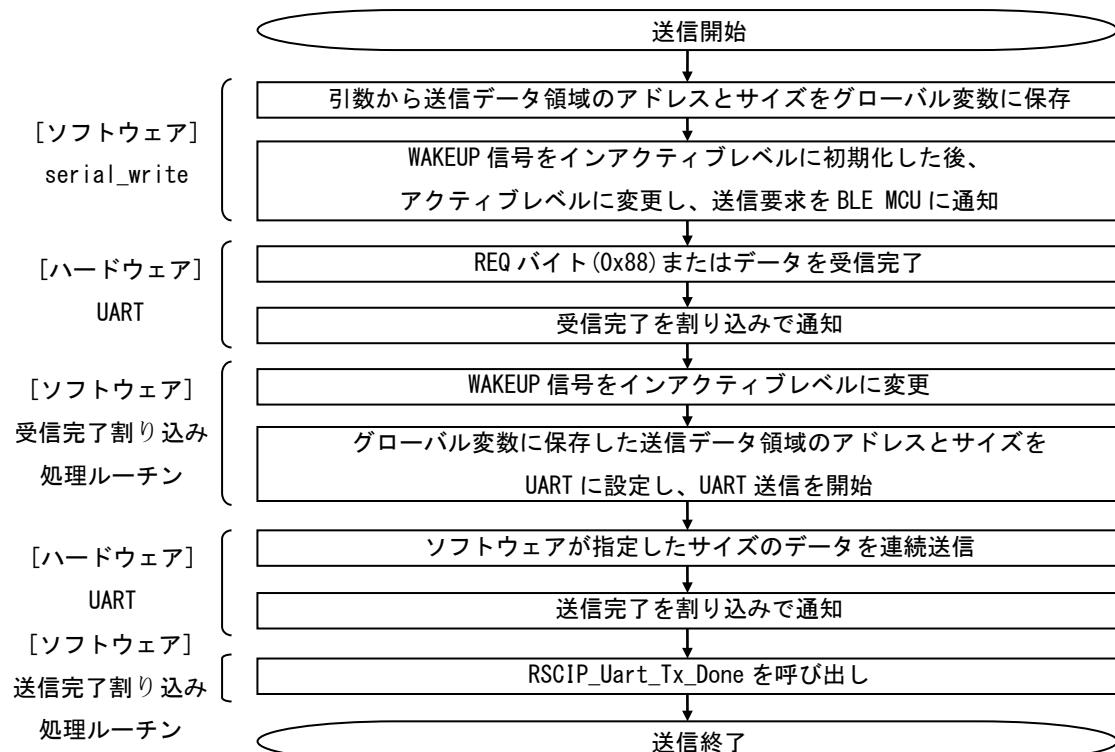


※RSCIP_Uart_Rx_Done 関数は serial_read 関数を呼び出し、次の受信動作を開始します。

7.2.3 UART 3 線接続方式の送信実装例

UART における 3 線接続方式を実現するための送信実装例を以下に示します。前提条件として、使用する UART ハードウェアは、送信データ領域のアドレスとサイズをレジスタに設定し送信開始すると、指定した送信データを送信し、送信完了後に割り込みが発生することとします。また、前述の受信実装を実現していることとします。

なお、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行する処理を追加してください。



※受信完了割り込み処理ルーチンは、受信動作においても使用します。

7.2.4 UART 2 線分岐接続方式の送信実装例

UART における 2 線分岐接続方式を実現するための送信実装例を以下に示します。前提条件として、使用する UART ハードウェアは、送信データ領域のアドレスとサイズをレジスタに設定し送信開始すると、指定した送信データを送信し、送信完了後に割り込みが発生することとします。また、前述の受信実装を実現していることとします。

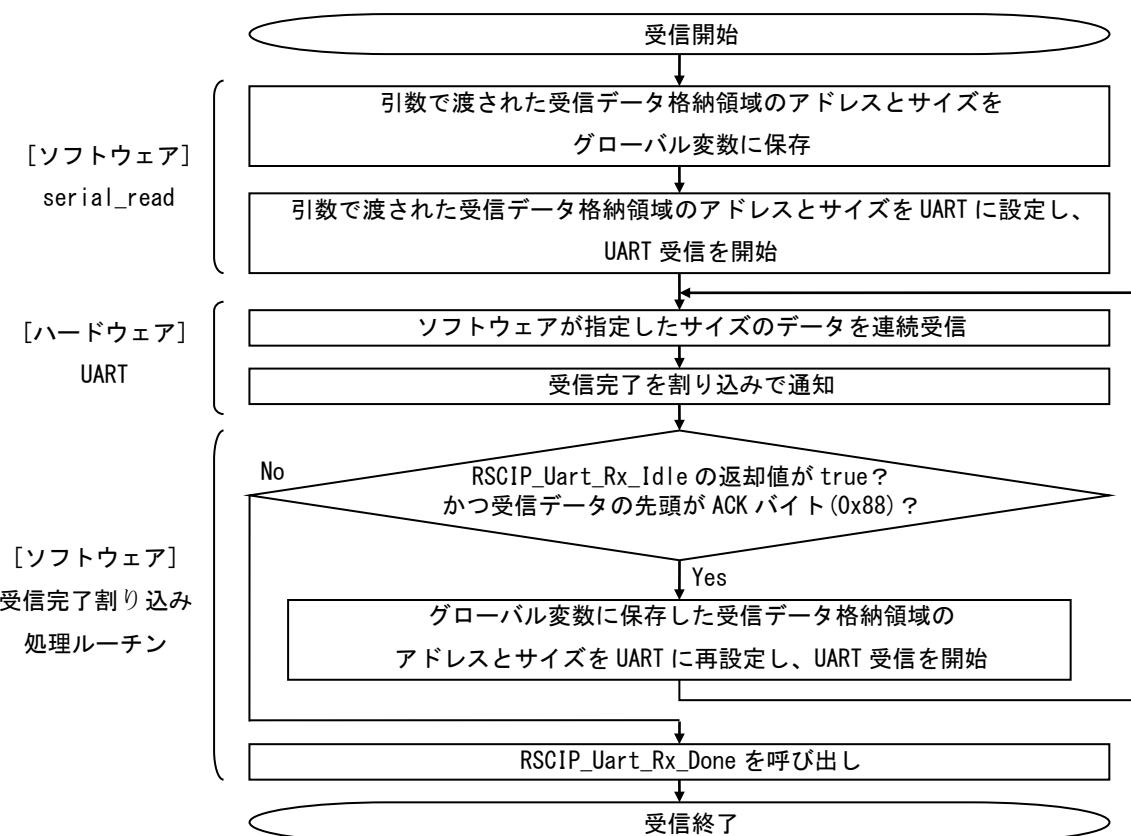
なお、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行する処理を追加してください。



※受信完了割り込み処理ルーチンは、受信動作においても使用します。

7.2.5 UART 3 線接続方式、2 線分岐接続方式の受信実装例

UART における 3 線接続方式、2 線分岐接続方式を実現するための受信実装例を以下に示します。前提条件として、使用する UART ハードウェアは、受信データ領域のアドレスとサイズをレジスタに設定し受信開始すると、指定したサイズのデータを受信し、受信完了後に割り込みが発生することとします。

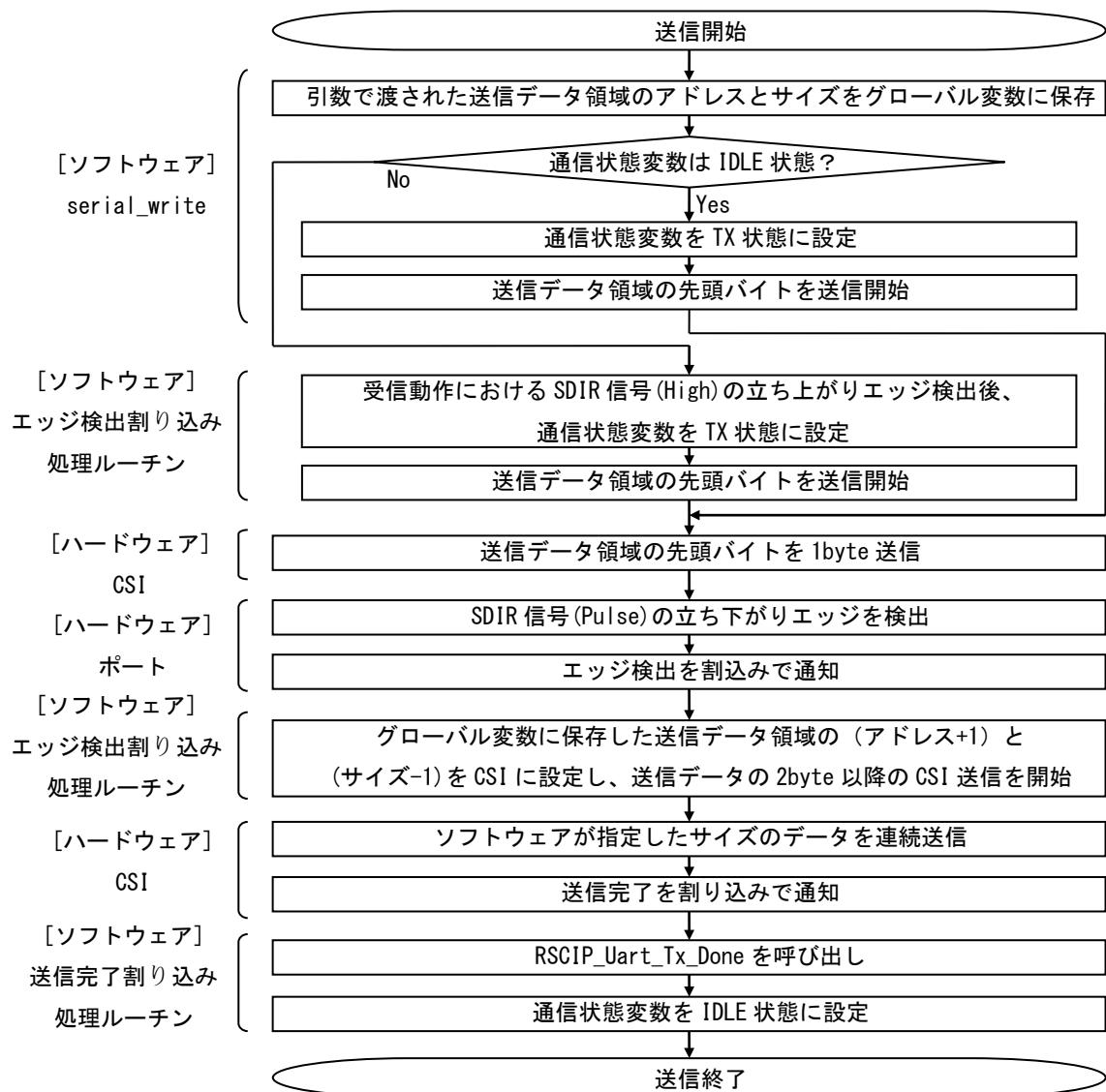


※RSCIP_Uart_Rx_Done 関数は serial_read 関数を呼び出し、次の受信動作を開始します。

7.2.6 CSI 4 線接続方式の送信実装例

CSIにおける4線接続方式を実現するための送信実装例を以下に示します。前提条件として、使用するCSIハードウェアは、送信データ領域のアドレスとサイズをレジスタに設定し送信開始すると、指定したサイズのデータを送信し、送信完了後に割り込みが発生することとします。またBLE MCUのSDIR信号端子と接続しているAPP MCUのポートは、SDIR信号の立ち下がりエッジ、立ち上がりエッジを検出しエッジ検出割込みが発生することとします。

なお、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行する処理を追加してください。



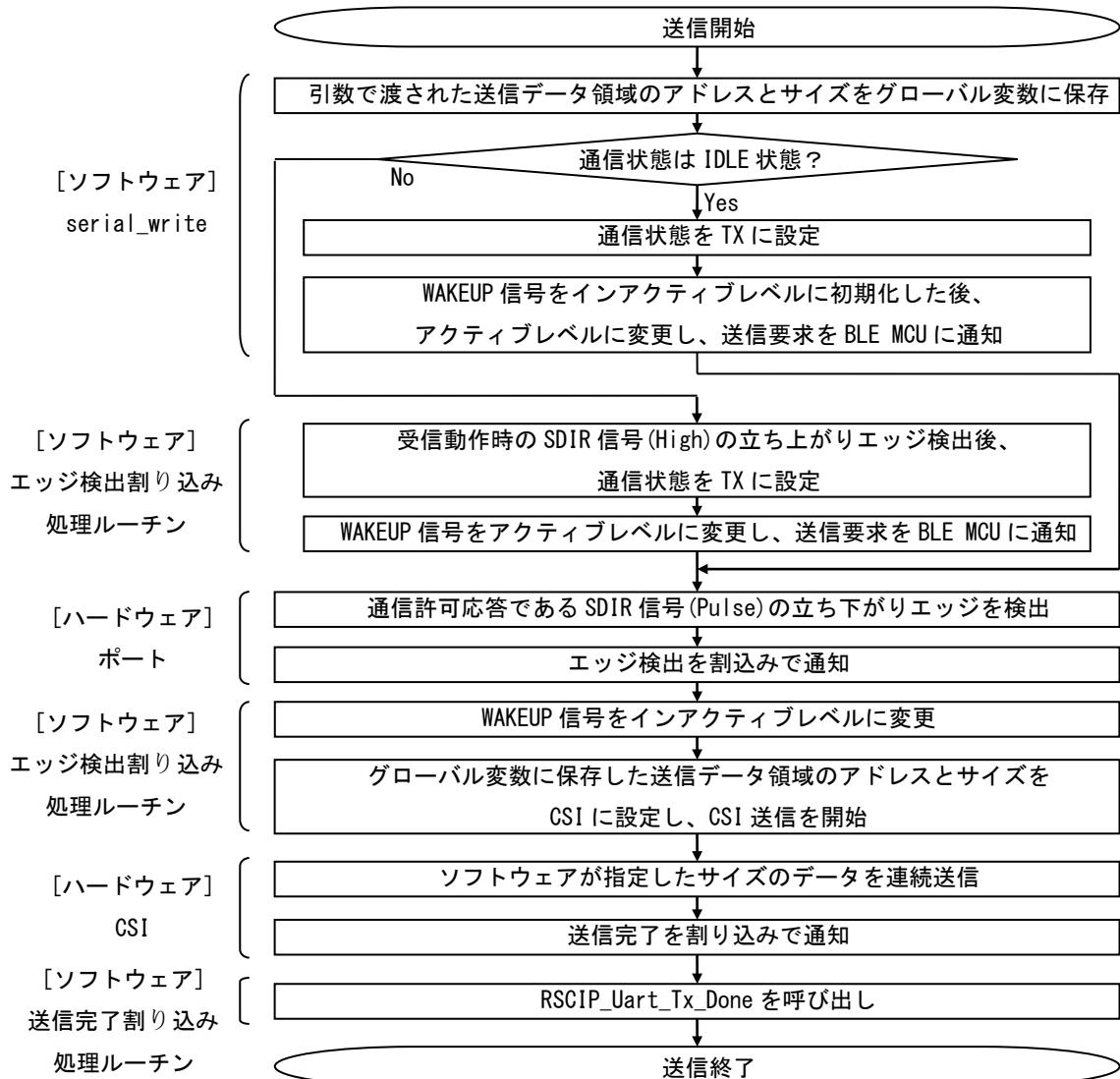
※エッジ検出割り込み処理ルーチンは、送信動作において複数回使用します。

※エッジ検出割り込み処理ルーチンは、受信動作においても使用します。

7.2.7 CSI 5 線接続方式の送信実装例

CSIにおける5線接続方式を実現するための送信実装例を以下に示します。前提条件として、使用するCSIハードウェアは、送信データ領域のアドレスとサイズをレジスタに設定し送信開始すると、指定したサイズのデータを送信し、送信完了後に割り込みが発生することとします。またBLE MCUのSDIR信号端子と接続しているAPP MCUのポートは、SDIR信号の立ち下がりエッジ、立ち上がりエッジを検出しエッジ検出割込みが発生することとします。

なお、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行する処理を追加してください。



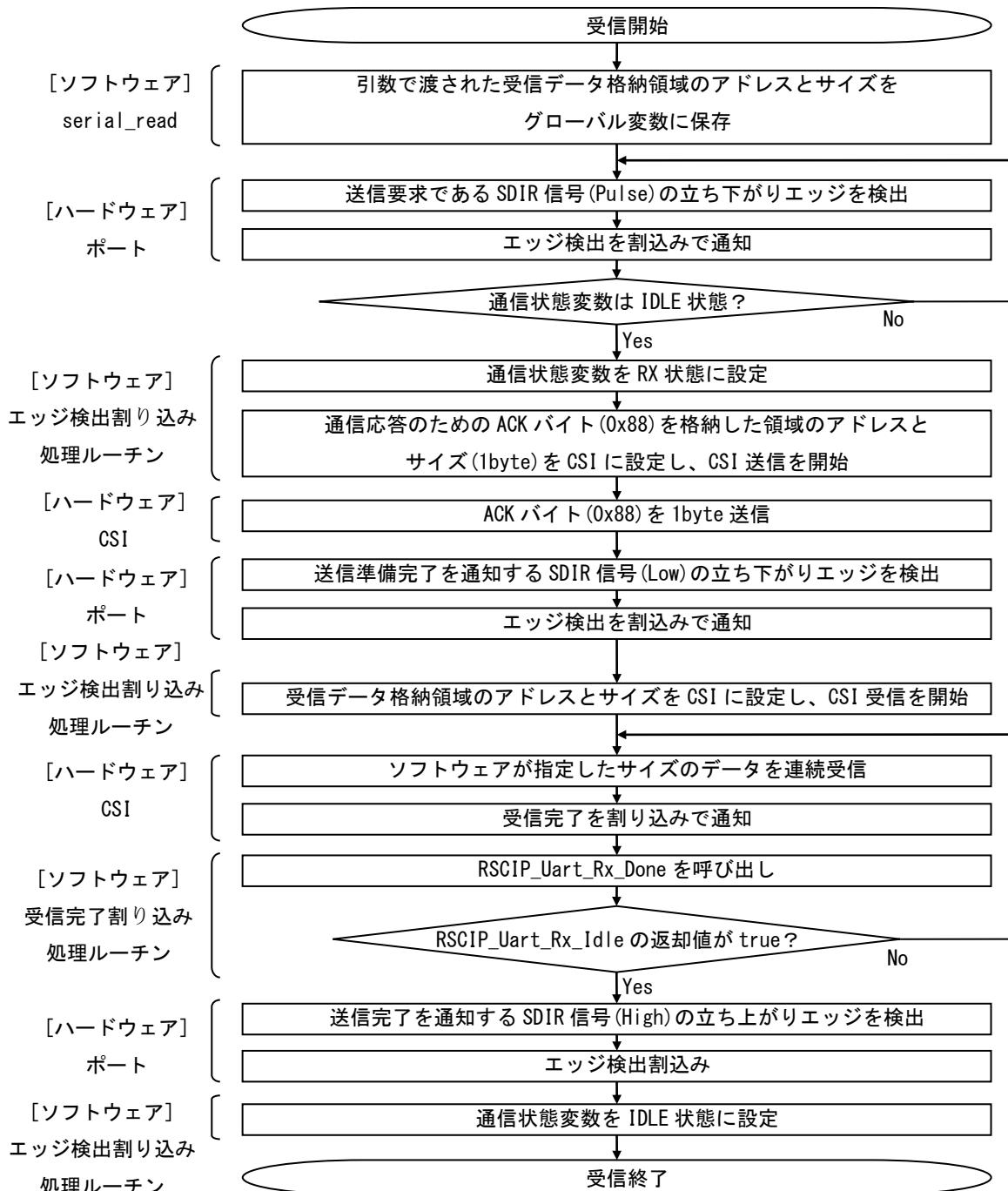
※RSCIP_Uart_Rx_Done 関数は serial_read 関数を呼び出し、次の受信動作を開始します。

※エッジ検出割り込み処理ルーチンは、送信動作において複数回使用します。

※エッジ検出割り込み処理ルーチンは、受信動作においても使用します。

7.2.8 CSI の受信実装例

CSIにおける4線接続方式、5線接続方式を実現するための受信実装例を以下に示します。前提条件として、使用するCSIハードウェアは、受信データ領域のアドレスとサイズをレジスタに設定し受信開始すると、指定したサイズのデータを受信し、受信完了後に割り込みが発生することとします。またBLE MCUのSDIR信号端子と接続しているAPP MCUのポートは、SDIR信号の立ち下がりエッジ、立ち上がりエッジを検出しエッジ検出割込みが発生することとします。



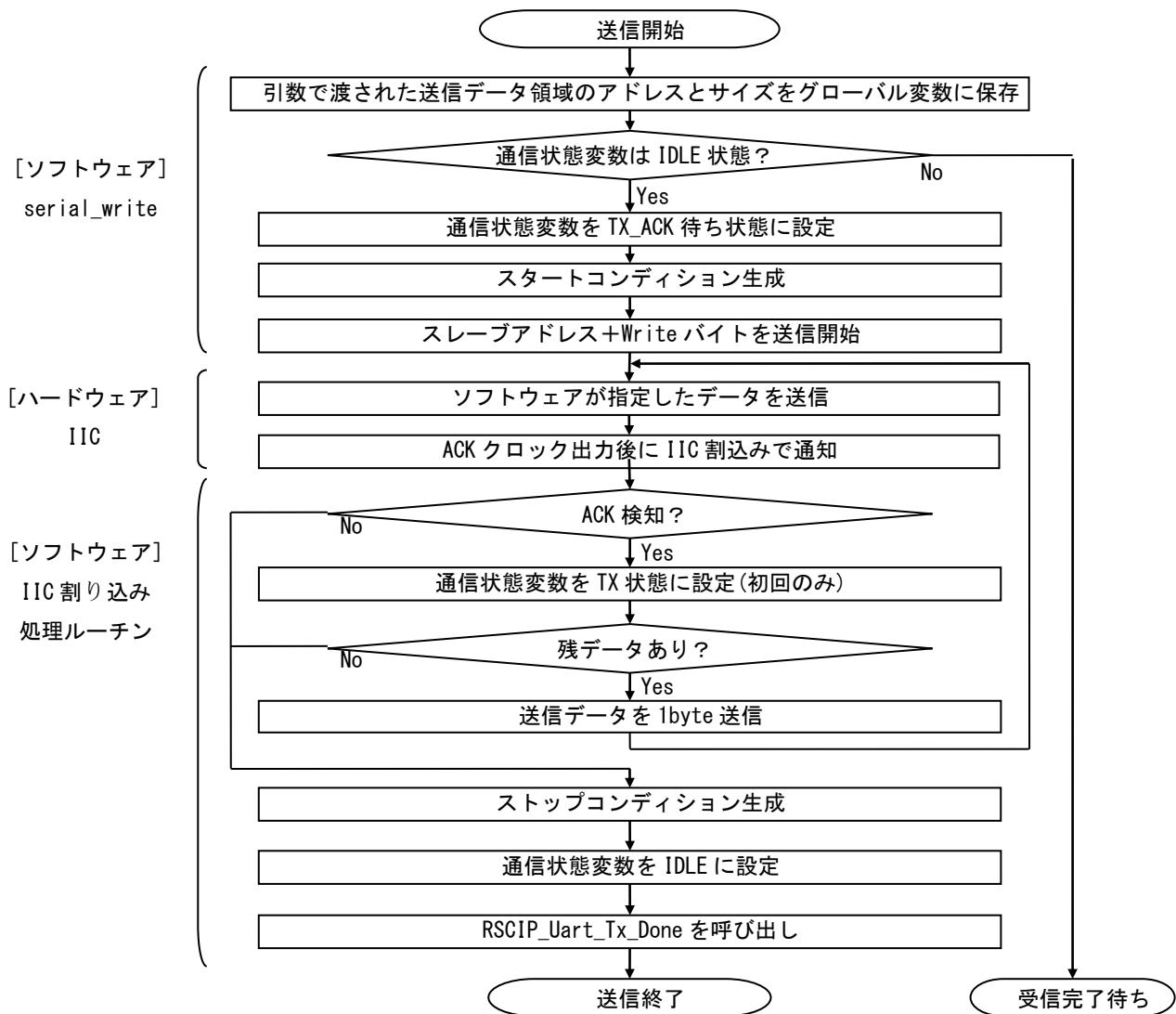
※エッジ検出割り込み処理ルーチンは、送信動作において複数回使用します。

※エッジ検出割り込み処理ルーチンは、受信動作においても使用します。

7.2.9 IIC 3 線接続方式の送信実装例

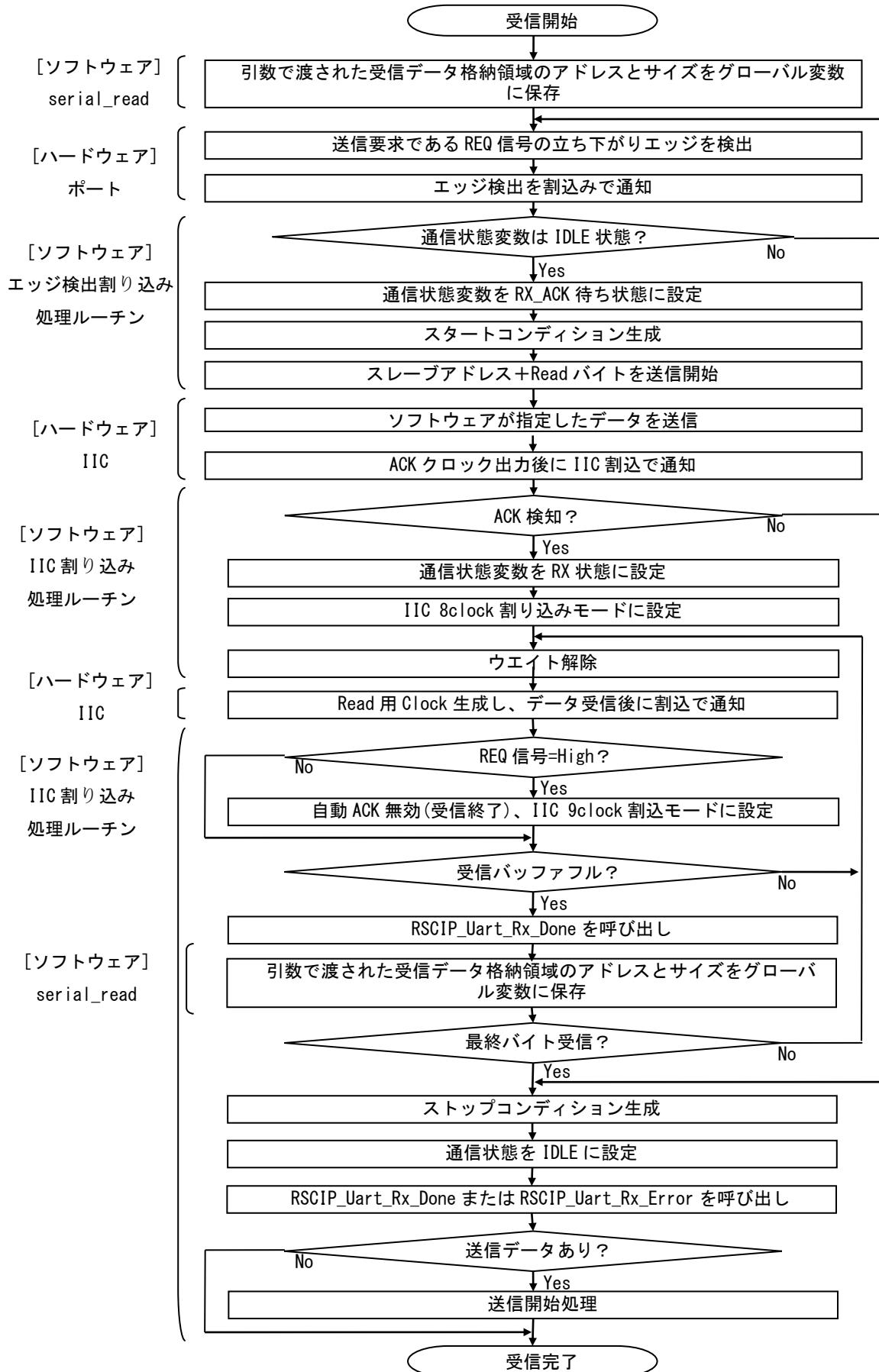
IIC における 3 線接続方式を実現するための送信実装例を以下に示します。前提条件として、IIC ハードウェアは 1 バイト送信毎に割り込みが発生することとします。

なお、確実な通信を行うため、ハンドシェイク時にはタイムアウトによる監視を行い、タイムアウト発生時にはハンドシェイクを再実行する処理を追加してください。



7.2.10 IIC 3 線接続方式の受信実装例

IIC における 3 線接続方式を実現するための受信実装例を以下に示します。前提条件として、使用する IIC ハードウェアは 1 バイト受信毎に割り込みが発生することとします。また BLE MCU の REQ 信号端子と接続している APP MCU のポートは、REQ 信号の立ち下がりエッジを検出しエッジ検出割込みが発生することとします。



7.3 APP MCU 向け組み込みサンプルプログラム

APP MCU にサンプルプログラムを組み込むための APP MCU 依存のソースファイルは以下に格納されています。

/Renesas/BLE_Software_Ver_X_XX/BLE_Sample/src/Platform/G1D_cs_iar/

本ソースファイルは、APP MCU に RL78/G1D を使用した場合のサンプルプログラムとなりますので、RL78/G1D 評価ボードを 2 台使用して動作確認することが可能です。

APP MCU に別の MCU を使用する場合には、これらのソースファイルをお客様のシステムに合うように修正してください。

サンプルプログラムを APP MCU に移植する場合には、サンプルプログラムを参考にお客様で新規開発していただくものと、サンプルプログラムをそのまま利用していただけるものがありますので、それらの分類を表 7-1 に示します。

表 7-1 サンプルプログラムの移植方法

フォルダ名	移植方法	移植内容
BLE_Sample/src/Platform/G1D_cs_iar	新規開発	サンプルプログラムを参考に、APP MCU のリソースに合うよう、お客様で新規開発してください
BLE_Sample/src/rBLE/src/host	再利用	そのまま再利用が可能です
BLE_Sample/src/rBLE/src/include	再利用	そのまま再利用が可能です
BLE_Sample/src/rBLE/src/rscip	再利用	そのまま再利用が可能です
BLE_Sample/src/rBLE/src/sample_app	新規開発	サンプルプログラムの API の使用方法を参考に、お客様のアプリケーションを新規開発してください

尚、再利用可能なサンプルプログラムのサイズの参考値を表 7-2 に示します。これらの値は、RL78/G1D でコンパイルした結果です。

ビルド環境 : CS+ for CC V4.00.00 / RL78 コンパイラー CC-RL V1.03.00

表 7-2 ROM サイズ・RAM サイズ

コンポーネント	ROM サイズ	RAM サイズ
rBLE (BLE_Sample/src/rBLE/src/host)	52,519 byte	2,898 byte
RSCIP (BLE_Sample/src/rBLE/src/rscip)	4,279 byte	1,268 byte

上記の参考値から以下の対策を実施することで、使用 RAM サイズを約 2KB 削減することができます。

1) BLE_Sample\src\rBLE\src\host\rble_host.c

変更前 : #define MAX_BUFF_NUM	8
↓	
変更後 : #define MAX_BUFF_NUM	2

※MAX_BUFF_NUM を超えるコマンドを連続して実行することができなくなります。

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

2) BLE_Sample¥src¥rBLE¥src¥host¥rble_if_api_cb.c

変更前 : static uint8_t rBLE_Over_Packet_Temp[0x256];

↓

変更後 : static uint8_t rBLE_Over_Packet_Temp[1];

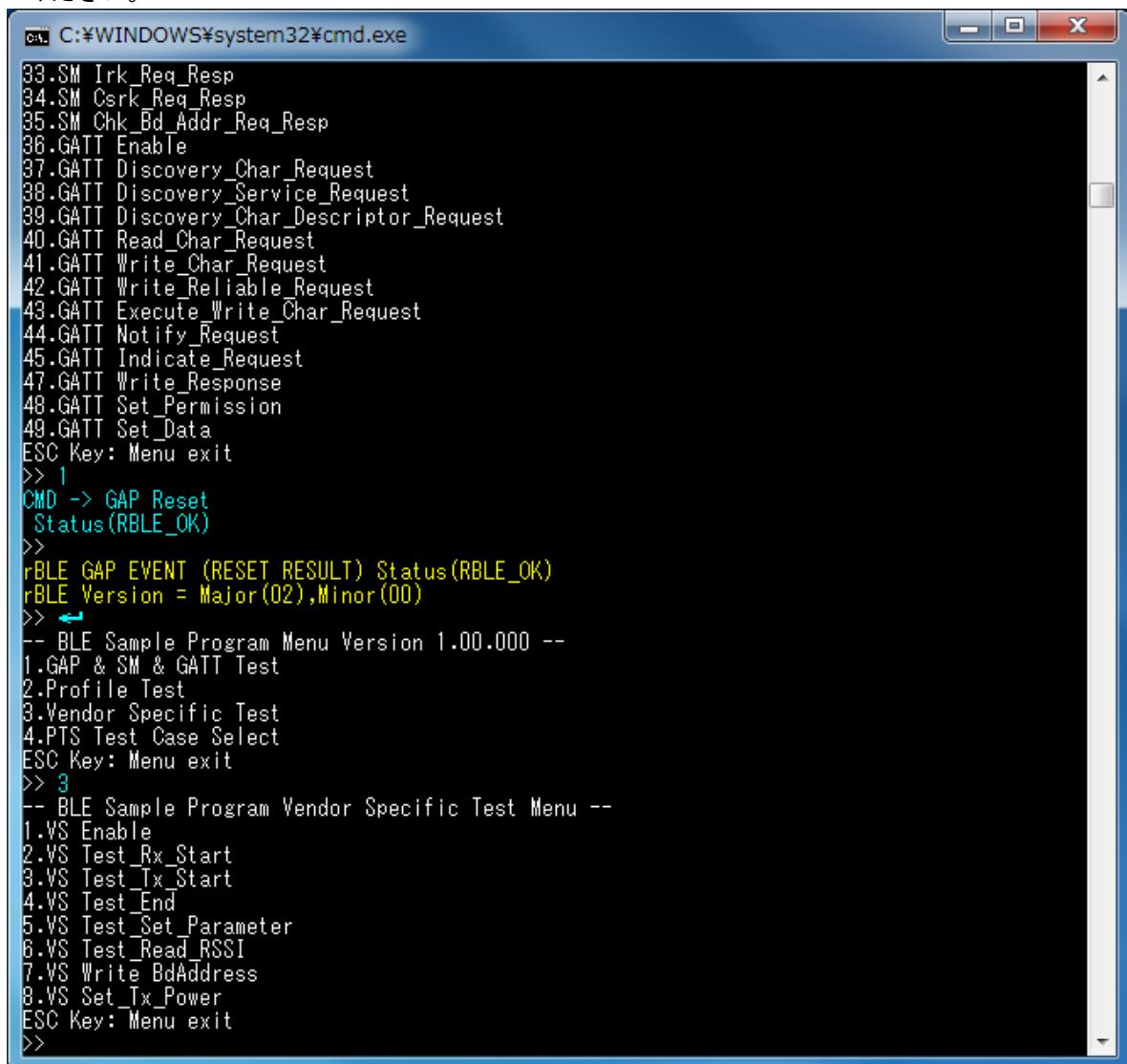
※RBLE_VS_Flash_Access()で 120Byte 以上のデータが扱えなくなります。

(実行した場合、不正メモリアクセスが発生します。)

7.4 Direct Test Mode の使用方法

Direct Test Mode は、Vendor Specific (VS)コマンドにより実行されます。図 7-5に Vendor Specific (VS)コマンドメニューを示します。メニュー2~5が Direct Test Mode 関連の項目になります。以降に、Direct Test Mode 関連コマンドについて記載します。

【注】 Direct Test Mode の詳細については、API リファレンスマニュアル 基本編 8 章 Vendor Specific を参照ください。



```

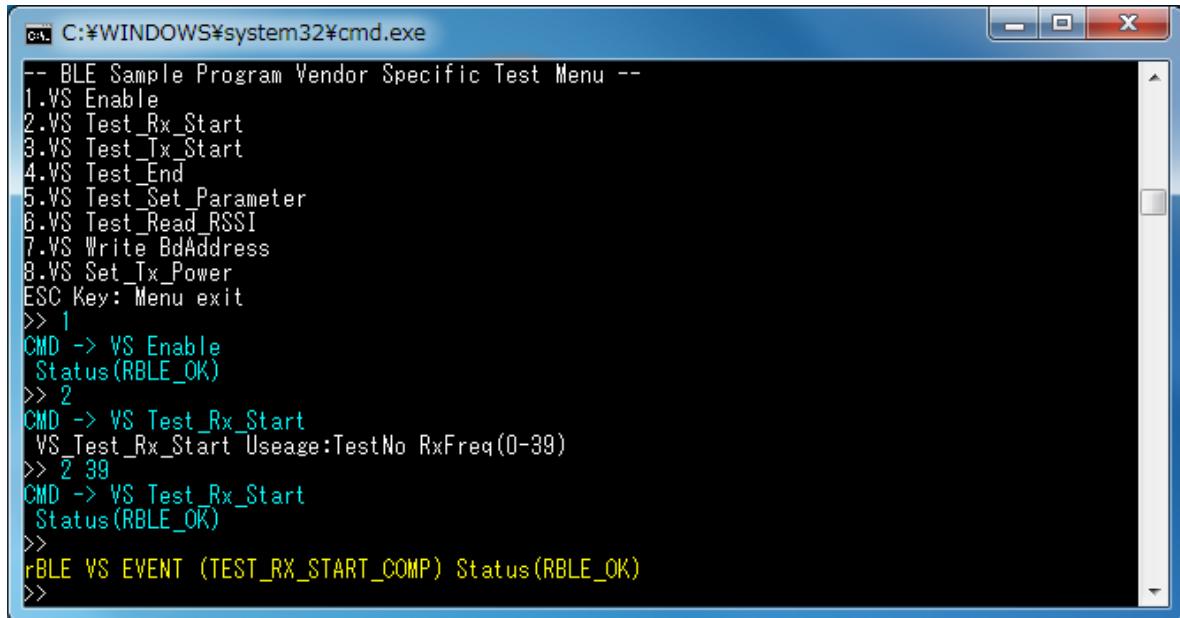
C:\WINDOWS\system32\cmd.exe
33.SM_Irk_Req_Resp
34.SM_Csrk_Req_Resp
35.SM_Clk_Bd_Addr_Req_Resp
36.GATT_Enable
37.GATT_Discovery_Char_Request
38.GATT_Discovery_Service_Request
39.GATT_Discovery_Char_Descriptor_Request
40.GATT_Read_Char_Request
41.GATT_Write_Char_Request
42.GATT_Write_Reliable_Request
43.GATT_Execute_Write_Char_Request
44.GATT_Notify_Request
45.GATT_Indicate_Request
47.GATT_Write_Response
48.GATT_Set_Permission
49.GATT_Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP_Reset
Status(RBLE_OK)
>>
rBLE_GAP_EVENT (RESET_RESULT) Status(RBLE_OK)
rBLE_Version = Major(02),Minor(00)
>> ←
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 3
-- BLE Sample Program Vendor Specific Test Menu --
1.VS_Enable
2.VS_Test_Rx_Start
3.VS_Test_Tx_Start
4.VS_Test_End
5.VS_Test_Set_Parameter
6.VS_Test_Read_RSSI
7.VS_Write_BdAddress
8.VS_Set_Tx_Power
ESC Key: Menu exit
>>

```

図 7-5 Vendor Specific (VS)メニュー

7.4.1 Direct Test Mode (Receiver)

VS メニュー番号 2 “VS Test_Rx_Start”により Direct Test Mode (Receiver)を開始できます。

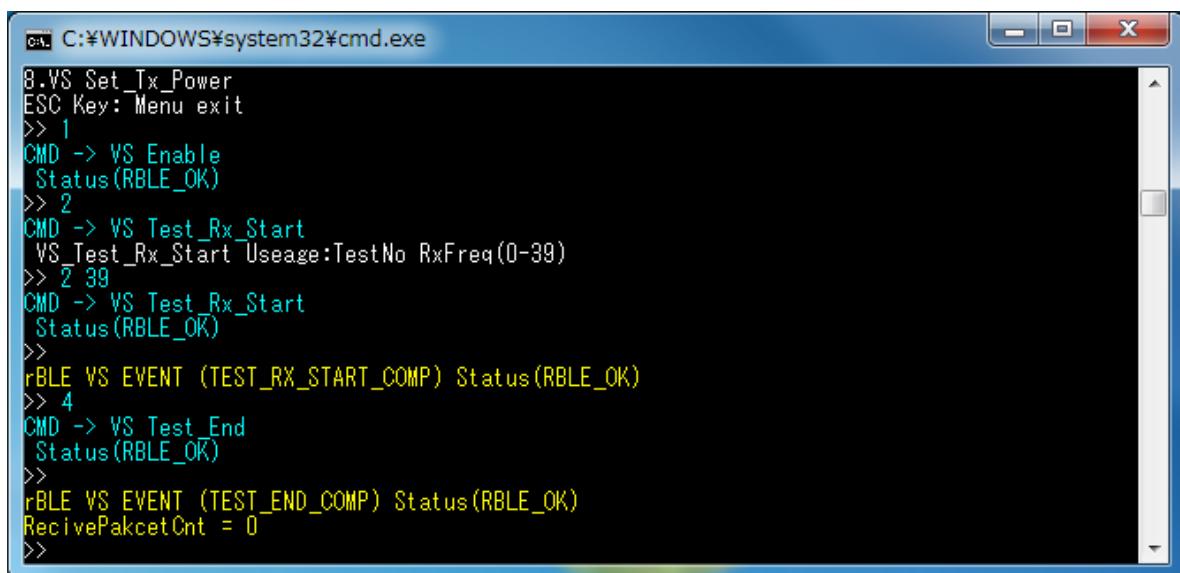


```
-- BLE Sample Program Vendor Specific Test Menu --
1.VS Enable
2.VS Test_Rx_Start
3.VS Test_Tx_Start
4.VS Test_End
5.VS Test_Set_Parameter
6.VS Test_Read_RSSI
7.VS Write_BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
Status(RBLE_OK)
>> 2
CMD -> VS Test_Rx_Start
VS_Test_Rx_Start Useage:TestNo RxFreq(0-39)
>> 2 39
CMD -> VS Test_Rx_Start
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLE_OK)
>>
```

図 7-6 Direct Test Mode (Receiver) Start ログ

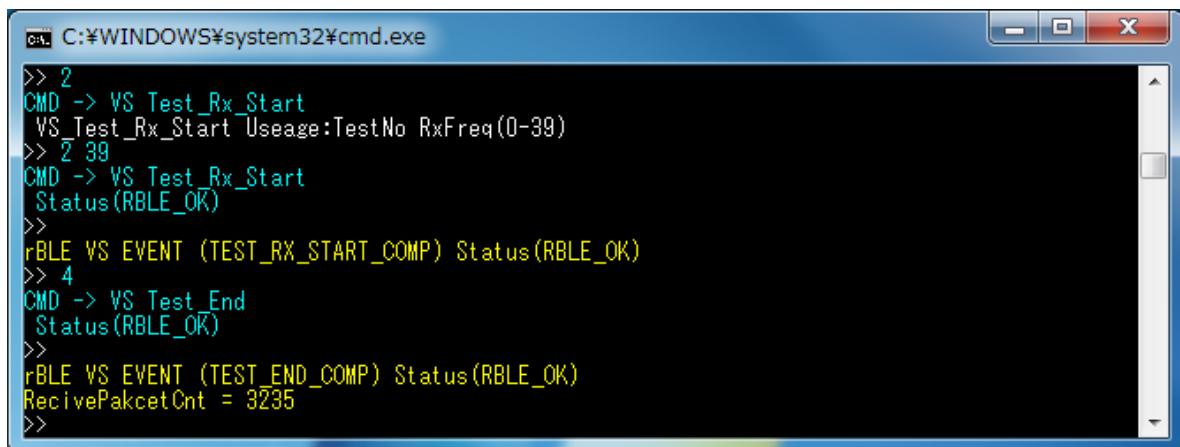
VS メニュー番号 2 “VS Test_Rx_Start”は、引数に受信周波数 (ch 番号) を設定します。引数を入力しなかつた場合は、このコマンドの使用方法を表示します。図 7-6では、受信周波数に 39ch(2480MHz)を設定したログになります。

Direct Test Mode(Receiver)の終了は VS メニュー番号 4 “VS Test_End”で終了します。図 7-7に終了時のログを示します。終了すると受信回数を表示します。図 7-7では 0 回、図 7-8では 3235 回データを受信したことを見示します。



```
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
Status(RBLE_OK)
>> 2
CMD -> VS Test_Rx_Start
VS_Test_Rx_Start Useage:TestNo RxFreq(0-39)
>> 2 39
CMD -> VS Test_Rx_Start
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLE_OK)
>> 4
CMD -> VS Test_End
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
ReceivePakcetCnt = 0
>>
```

図 7-7 Direct Test Mode (Receiver) End ログ 1

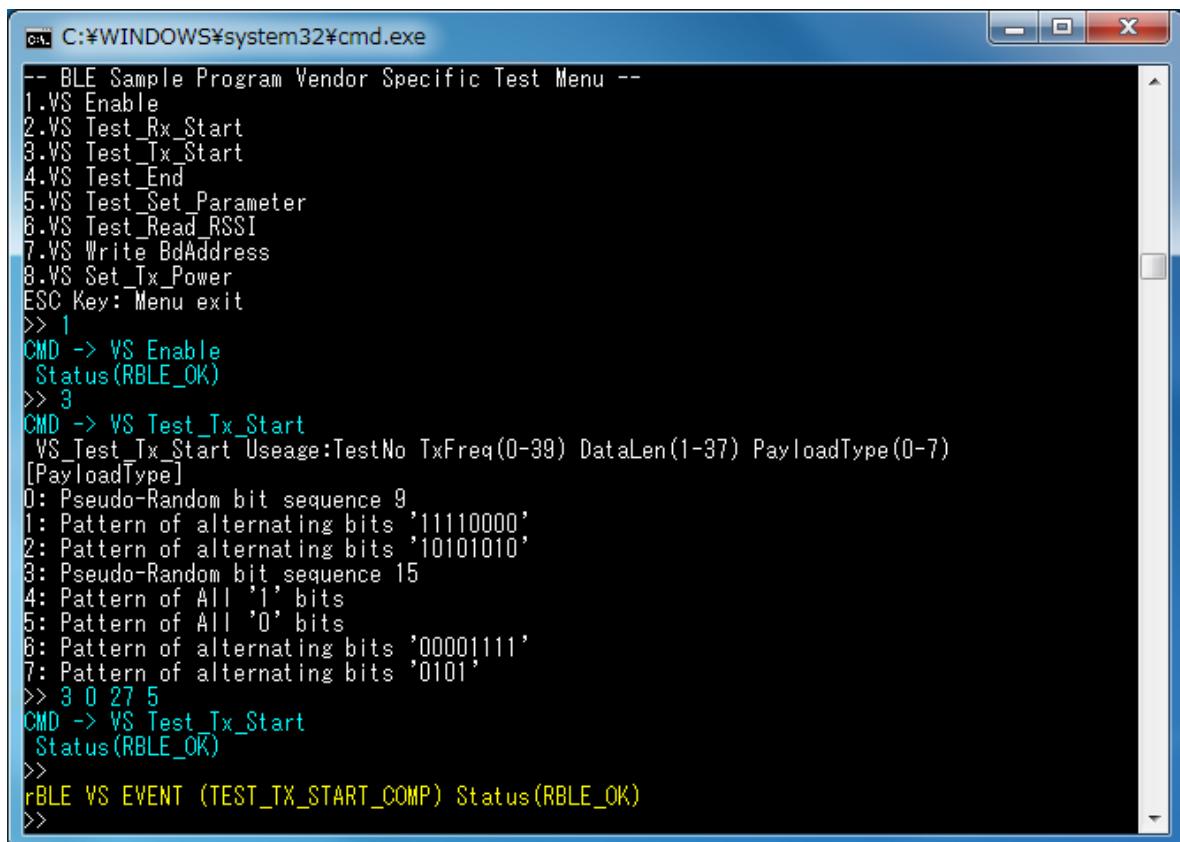


```
C:\> 2
CMD -> VS Test_Rx_Start
VS_Test_Rx_Start Useage:TestNo RxFreq(0-39)
>> 2 39
CMD -> VS Test_Rx_Start
Status(RBLED_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLED_OK)
>> 4
CMD -> VS Test_End
Status(RBLED_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLED_OK)
ReceivePakcetCnt = 3235
>>
```

図 7-8 Direct Test Mode (Receiver) End ログ 2

7.4.2 Direct Test Mode (Transmitter)

VS メニュー番号 3 “VS Test_Tx_Start”により Direct Test Mode (Transmitter)を開始できます。

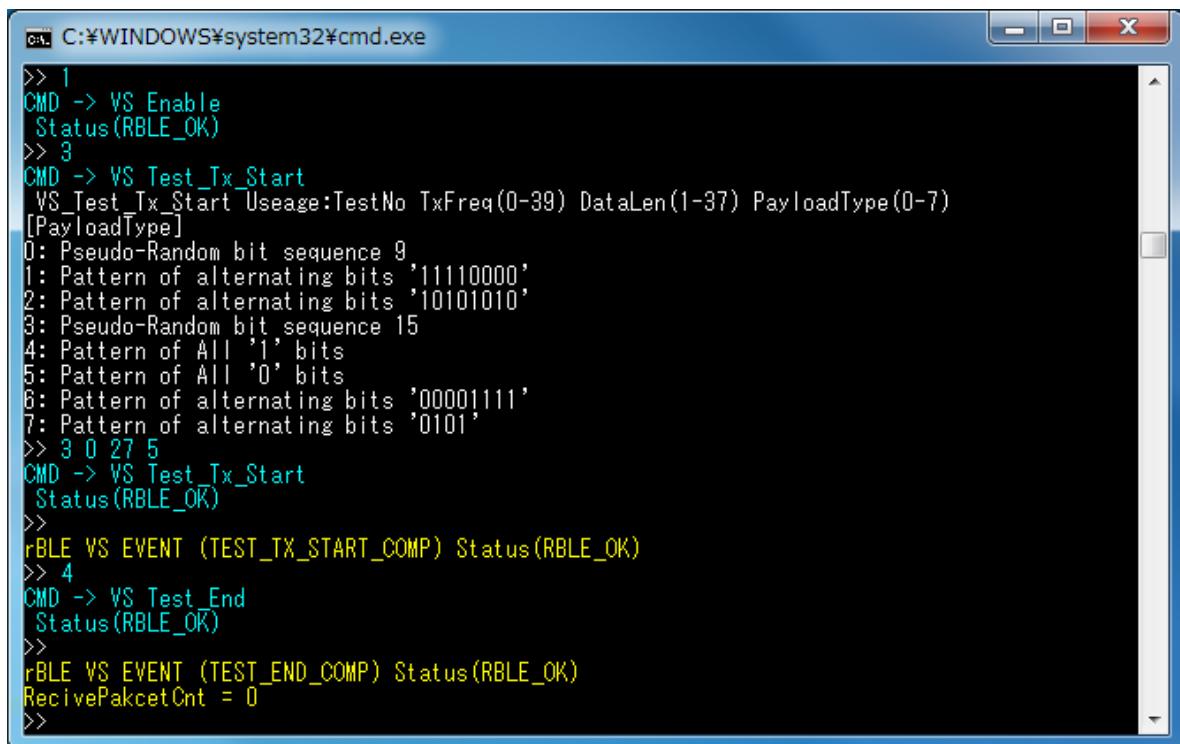


```
-- BLE Sample Program Vendor Specific Test Menu --
1.VS Enable
2.VS Test_Rx_Start
3.VS Test_Tx_Start
4.VS Test_End
5.VS Test_Set_Parameter
6.VS Test_Read_RSSI
7.VS Write BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
Status(RBLED_OK)
>> 3
CMD -> VS Test_Tx_Start
VS_Test_Tx_Start Useage:TestNo TxFreq(0-39) DataLen(1-37) PayloadType(0-7)
[PayloadType]
0: Pseudo-Random bit sequence 9
1: Pattern of alternating bits '11110000'
2: Pattern of alternating bits '10101010'
3: Pseudo-Random bit sequence 15
4: Pattern of All '1' bits
5: Pattern of All '0' bits
6: Pattern of alternating bits '00001111'
7: Pattern of alternating bits '0101'
>> 3 0 27 5
CMD -> VS Test_Tx_Start
Status(RBLED_OK)
>>
rBLE VS EVENT (TEST_TX_START_COMP) Status(RBLED_OK)
>>
```

図 7-9 Direct Test Mode (Transmitter) Start ログ

VS メニュー番号 3 “VS Test_Tx_Start”は、引数に送信周波数 (ch 番号) 、データサイズ、データタイプを設定します。引数を入力しなかった場合は、このコマンドの使用方法を表示します。図 7-9では、送信周波数に 0ch(2402MHz)を、データサイズに 27 バイトを、データタイプに ALL0 を設定したログになります。

Direct Test Mode(Transmitter)の終了は VS メニュー番号 4 “VS Test_End”で終了します。図 7-10に終了時のログを示します。送信テスト終了でも受信回数を表示しますが、回数は 0 になります。

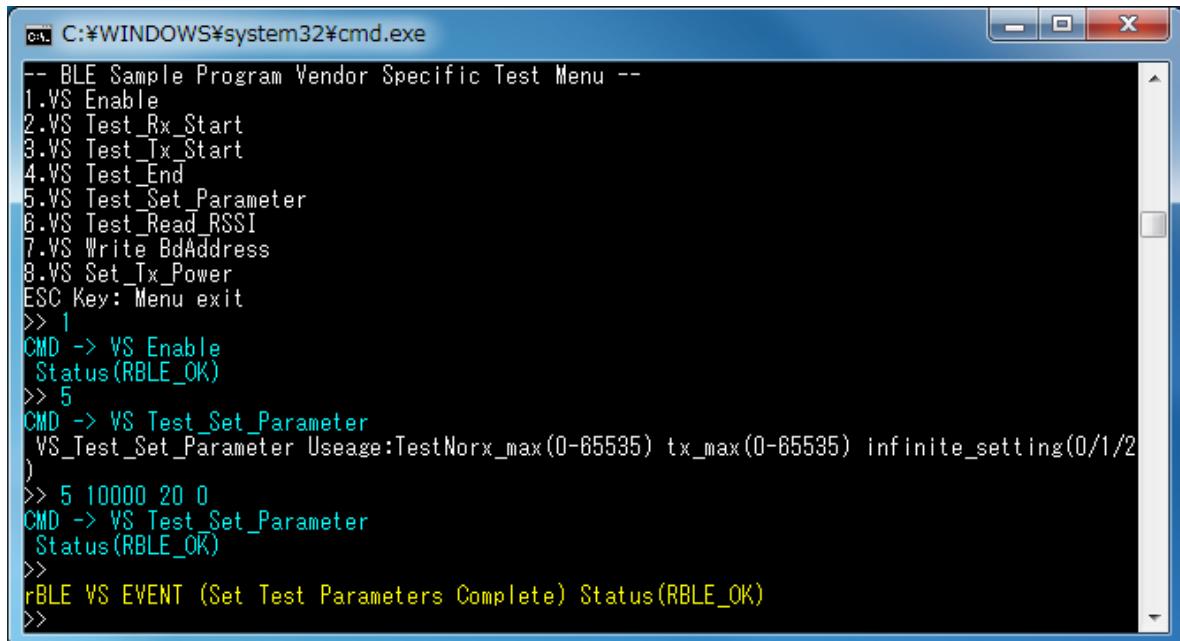


```
C:\> C:\WINDOWS\system32\cmd.exe
>> 1
CMD -> VS_Enable
Status(RBLE_OK)
>> 3
CMD -> VS_Test_Tx_Start
VS_Test_Tx_Start Useage:TestNo TxFreq(0-39) DataLen(1-37) PayloadType(0-7)
[PayloadType]
0: Pseudo-Random bit sequence 9,
1: Pattern of alternating bits '11110000'
2: Pattern of alternating bits '10101010'
3: Pseudo-Random bit sequence 15
4: Pattern of All '1' bits
5: Pattern of All '0' bits
6: Pattern of alternating bits '00001111'
7: Pattern of alternating bits '0101'
>> 3 0 27 5
CMD -> VS_Test_Tx_Start
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_TX_START_COMP) Status(RBLE_OK)
>> 4
CMD -> VS_Test_End
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
ReceivePakcetCnt = 0
>>
```

図 7-10 Direct Test Mode (Transmitter) End ログ

7.4.3 Direct Test Mode (Parameter Set)

VS メニュー番号 5 “VS Test_Set_Parameter”により Direct Test Mode (Receiver)および Direct Test Mode (Transmitter)に関するパラメータを設定できます。



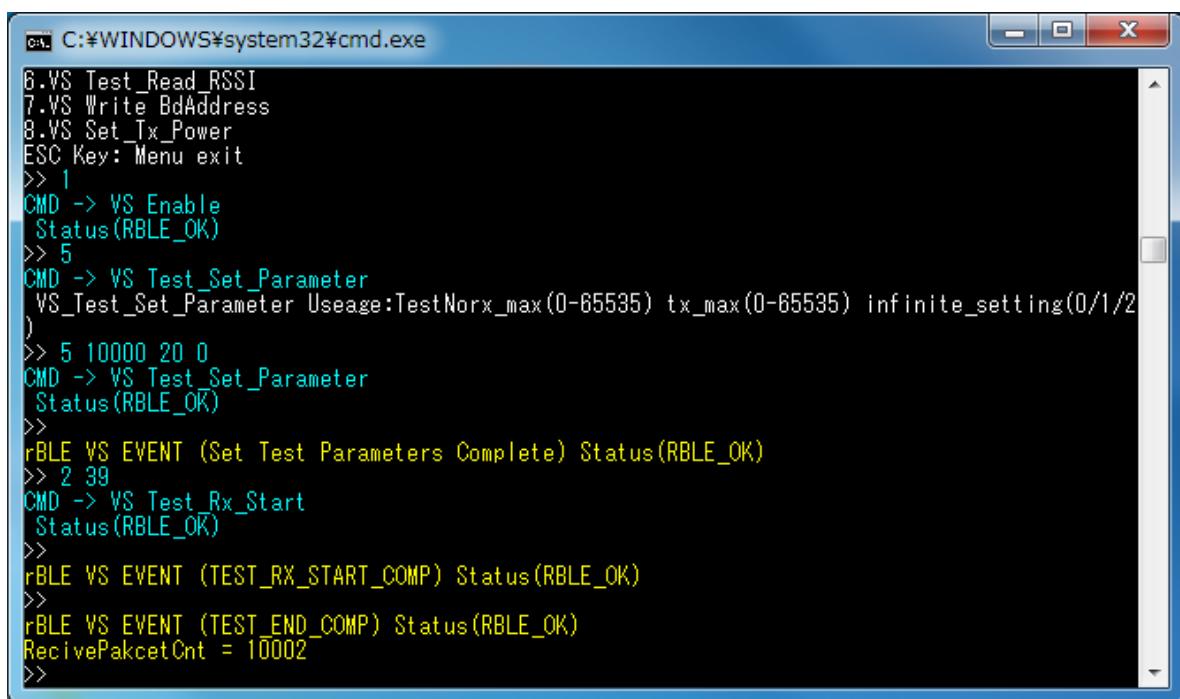
```
C:\> C:\WINDOWS\system32\cmd.exe
-- BLE Sample Program Vendor Specific Test Menu --
1.VS_Enable
2.VS_Test_Rx_Start
3.VS_Test_Tx_Start
4.VS_Test_End
5.VS_Test_Set_Parameter
6.VS_Test_Read_RSSI
7.VS_Write_BdAddress
8.VS_Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS_Enable
Status(RBLE_OK)
>> 5
CMD -> VS_Test_Set_Parameter
VS_Test_Set_Parameter Useage:TestNo rx_max(0-65535) tx_max(0-65535) infinite_setting(0/1/2)
)
>> 5 10000 20 0
CMD -> VS_Test_Set_Parameter
Status(RBLE_OK)
>>
rBLE VS EVENT (Set Test Parameters Complete) Status(RBLE_OK)
>>
```

図 7-11 Direct Test Mode Parameter Set ログ

VS メニュー番号 5 “VS Test_Set_Parameter”は、引数に受信回数、送信回数、バースト転送有効／無効を設定します。引数を入力しなかった場合は、このコマンドの使用方法を表示します。図 7-11では、受信回数に 10000 回を、送信回数に 20 回を、バースト転送を無効に設定したログになります。

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

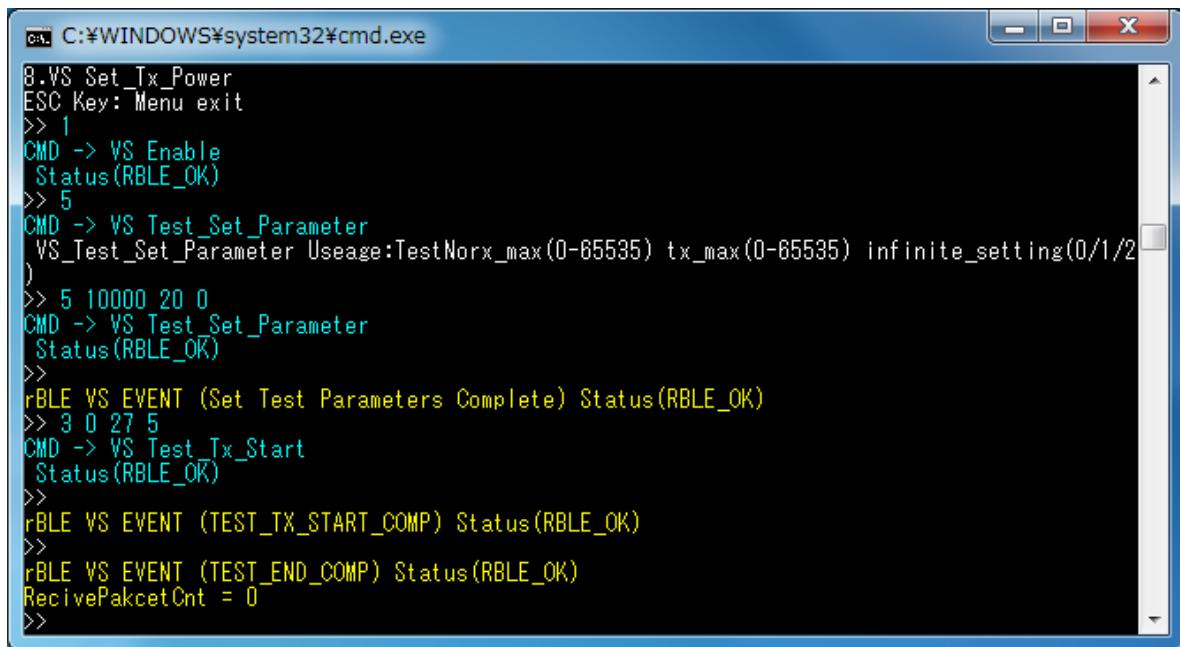
図 7-12は、上記パラメータを設定した後に Direct Test Mode (Receiver)を実行した際のログになります。10000 回以上のパケットを受信したことで、自動的に Direct Test Mode を終了しています。



```
C:\> C:\WINDOWS\system32\cmd.exe
6.VS Test_Read_RSSI
7.VS Write_BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
Status(RBLE_OK)
>> 5
CMD -> VS Test_Set_Parameter
VS_Test_Set_Parameter Usage:TestNorx_max(0-65535) tx_max(0-65535) infinite_setting(0/1/2)
)
>> 5 10000 20 0
CMD -> VS Test_Set_Parameter
Status(RBLE_OK)
>>
rBLE VS EVENT (Set Test Parameters Complete) Status(RBLE_OK)
>> 2 39
CMD -> VS Test_Rx_Start
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
ReceivePakcetCnt = 10002
>>
```

図 7-12 Direct Test Mode Parameter Set 後 Direct Test Mode (Receiver)実行後ログ

図 7-13は、上記パラメータを設定した後に Direct Test Mode (Transmitter)を実行した際のログになります。20 パケットを送信したことで、自動的に Direct Test Mode を終了しています。



```
C:\> C:\WINDOWS\system32\cmd.exe
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
Status(RBLE_OK)
>> 5
CMD -> VS Test_Set_Parameter
VS_Test_Set_Parameter Usage:TestNorx_max(0-65535) tx_max(0-65535) infinite_setting(0/1/2)
)
>> 5 10000 20 0
CMD -> VS Test_Set_Parameter
Status(RBLE_OK)
>>
rBLE VS EVENT (Set Test Parameters Complete) Status(RBLE_OK)
>> 3 0 27 5
CMD -> VS Test_Tx_Start
Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_TX_START_COMP) Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
ReceivePakcetCnt = 0
>>
```

図 7-13 Direct Test Mode Parameter Set 後 Direct Test Mode (Transmitter)実行後ログ

7.5 Sample Custom Profile

GATT API を使用して作成した Sample Custom Profile(SCP)についての説明を以降に記述します。

なお、本サンプルプログラムを使用する場合は、プロジェクトのコンパイルオプションに「USE_SAMPLE_PROFILE」のマクロ定義を追加してください。

7.5.1 Sample Custom Profile 仕様

Sample Custom Profile(SCP)は Client Role と Server Role の 2 つの Role を保持しています。

SCP が保持している Characteristic、Descriptor を表 7-3に記述します。

表 7-3 Sample Custom Profile Characteristic/Descriptor 機能一覧

Characteristic 名	Properties	format	説明
Notify Characteristic	Notify	uint8_t[]	0~20byteまでの任意のデータを Notify で送信 送信データサイズは Notify Length Characteristic で指定可能
-Client Characteristic Configuration	Read/Write	uint16_t	Notify の ON/OFF を指定
Indicate Characteristic	Indicate	uint8_t[]	0~20byteまでの任意のデータを Indicate で送信 送信データサイズは Indicate Length Characteristic で指定可能
-Client Characteristic Configuration	Read/Write	uint16_t	Indicate の ON/OFF を指定
Interval Characteristic	Read/Write	uint16_t	Indicate/Notify の送信間隔を 10(ms)単位で指定
Notify Length Characteristic	Read/Write	uint8_t	Notify の送信データサイズを指定
Indicate Length Characteristic	Read/Write	uint8_t	Indicate の送信データサイズを指定

7.5.2 Sample Custom Profile ファイル構成

Sample Custom Profile に関するファイル構成を以下に記述します。

```

Renesas
└ BLE_Software_Ver_X_XX
    └ BLE_Sample
        └ src
            └ rBLE
                └ src
                    └ include
                    └ rble_api_custom.h
                    └ rble_app.h
                    └ sample_profile
                    └ db_handle.h
                    └ scp
                        └ scpc.c
                        └ scps.c
                    └ sample_app
                        └ rble_sample_app.c
                        └ rble_sample_app_custom.c
                └ PC_用サンプルプログラム格納フォルダ
                └ BLEサンプルプログラム格納フォルダ
                └ Custom Profile 追加 API ヘッダファイル
                └ サンプルプログラムヘッダファイル
                └ Sample Profile フォルダ
                └ Attribute database handles ヘッダファイル
                └ Sample Custom Profile 格納フォルダ
                └ Sample Custom Profile Client ファイル
                └ Sample Custom Profile Server ファイル
                └ サンプルプログラムファイル
                └ サンプルプログラムファイル(Sample Custom Profile)
                └ BLE MCU 向け BLE ソフトウェア格納フォルダ

└ RL78_G1D
    └ Project_Source
        └ rBLE
            └ src
                └ include
                └ rble_api_custom.h
                └ rble_app.h
                └ sample_profile
                └ scp
                    └ scpc.c
                    └ scps.c
                └ sample_app
                    └ rble_sample_app.c
                    └ rble_sample_app_custom.c
                └ rBLE 格納フォルダ
                └ Custom Profile 追加 API ヘッダファイル
                └ サンプルプログラムヘッダファイル
                └ Sample Profile フォルダ
                └ Sample Custom Profile 格納フォルダ
                └ Sample Custom Profile Client ファイル
                └ Sample Custom Profile Server ファイル
                └ サンプルプログラムファイル
                └ サンプルプログラムファイル(Sample Custom Profile)

    └ renesas
        └ src
            └ arch
                └ r178
                    └ prf_config.c
                    └ prf_config.h
                    └ prf_sel.h
                    └ db_handle.h
                └ プロファイル向けパラメータ設定ファイル
                └ プロファイル向けパラメータ設定ヘッダファイル
                └ プロファイル選択設定ヘッダファイル
                └ Attribute database handles ヘッダファイル

```

7.5.3 Sample Custom Profile IF 関数仕様

Sample Custom Profile(SCP)の IF 関数仕様を以降に記述します。

(1) RBLE_SCP_Clinet_Enable

```
RBLE_STATUS RBLE_SCP_Client_Enable( uint16_t conhdl,
                                      uint8_t con_type,
                                      RBLE_SCS_CONTENT *scs,
                                      RBLE_SCPC_EVENT_HANDLER call_back )
```

Client Role を有効化します。

初回に接続する場合は con_type に RBLE_SCP_CON_CFG を指定し、Server のサービス発見を行う必要があります。

初回に取得したサービスの情報を保持しておき、2 回目以降の有効化時に引数 scs に情報を指定し、con_type に RBLE_SCP_CON_NORMAL を指定することで、サービス発見の再実行を行わないため、高速な Role の有効化が行えます。

結果は Client 有効完了イベント(RBLE_SCP_EVENT_CLIENT_ENABLE_COMP)で通知されます。

Parameters:

conhdl	コネクションハンドル
con_type	接続方法指定
scs	SCP のハンドル情報(con_type に RBLE_SCP_CON_NORMAL を指定時のみ有効)
call_back	イベント通知を行う Callback 用関数の指定

Return:

RBLE_OK	正常終了
RBLE_PARAM_ERR	パラメータ異常
RBLE_STATUS_ERROR	SCP Client が無効状態以外のため実行不可

(2) RBLE_SCP_Clinet_Disable

```
RBLE_STATUS RBLE_SCP_Client_Disable( uint16_t conhdl, )
```

Client Role を無効化します。

結果は Client 無効化完了イベント(RBLE_SCP_EVENT_CLIENT_DISABLE_COMP)で通知されます。

Parameters:

conhdl	コネクションハンドル
--------	------------

Return:

RBLE_OK	正常終了
RBLE_PARAM_ERR	パラメータ異常
RBLE_STATUS_ERROR	SCP Client が有効状態以外のため実行不可

(3) RBLE SCP Clinet Read Char

```
RBLE_STATUS RBLE_SCP_Client_Read_Char ( uint16_t conhdl,
                                         uint8_t char_code,)
```

char_code に指定した characteristicdescriptor の値を取得します。

結果は特性値取得要求応答イベント(RBLE_SCP_EVENT_CLIENT_READ_CHAR_RESPONSE)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
<i>char_code</i>	読みだす characteristicdescriptor を指定。 RBLE_SCP_SCS_NTF_CFG Notify の ClientConfiguration を取得 RBLE_SCP_SCS_IND_CFG Indicate の ClientConfiguration を取得 RBLE_SCP_SCS_INTERVAL Interval Characteristic の値を取得 RBLE_SCP_SCS_NTF_LEN Notify Length Characteristic の値を取得 RBLE_SCP_SCS_IND_LEN Indicate Length Characteristic の値を取得

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	SCP Client が有効状態以外のため実行不可

(4) RBLE SCP Clinet Write Char

```
RBLE_STATUS RBLE_SCP_Client_Write_Char ( uint16_t conhdl,
                                         uint8_t char_code,
                                         uint8_t *write_value)
```

char_code に指定した characteristicdescriptor へ設定を行います。

結果は特性値設定要求応答イベント(RBLE_SCP_EVENT_CLIENT_WRITE_CHAR_RESPONSE)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
<i>char_code</i>	設定先の characteristicdescriptor を指定。 RBLE_SCP_SCS_NTF_CFG Notify の ClientConfiguration への設定 RBLE_SCP_SCS_IND_CFG Indicate の ClientConfiguration への設定 RBLE_SCP_SCS_INTERVAL Interval Characteristic への設定 RBLE_SCP_SCS_NTF_LEN Notify Length Characteristic への設定 RBLE_SCP_SCS_IND_LEN Indicate Length Characteristic への設定

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	SCP Client が有効状態以外のため実行不可

(5) RBLE_SCP_Server_Enable

```
RBLE_STATUS RBLE_SCP_Server_Enable ( uint16_t conhdl,
                                      uint8_t con_type,
                                      RBLE_SCP_SERVER_PARAM *param,
                                      RBLE_SCPS_EVENT_HANDLER call_back)
```

SCP 機能の Server Role を有効にします。

測定結果の通知を Client から設定される場合は con_type に RBLE_SCP_CON_CFG を、Server で設定する場合は con_type に RBLE_SCP_CON_NORMAL を指定し、param に設定を行って下さい。

結果は Server Role 有効化完了イベント(RBLE_SCP_EVENT_SERVER_ENABLE_COMP)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
<i>con_type</i>	接続方法指定
<i>param</i>	Server の初期設定
	<i>data_ntf_en</i> Notify の ClientConfiguration の初期値を指定
	<i>data_ind_en</i> Indicate の ClientConfiguration の初期値を指定
<i>call_back</i>	イベント通知を行う Callback 用関数の指定

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	SCP Server が無効状態以外のため実行不可

(6) RBLE_SCP_Server_Disable

```
RBLE_STATUS RBLE_SCP_Server_Disable( uint16_t conhdl, )
```

ServerRole を無効化します。

結果は Server 無効化完了イベント(RBLE_SCP_EVENT_SERVER_DISABLE_COMP)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
---------------	------------

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	SCP Server が有効状態以外のため実行不可

(7) RBLE_SCP_Server_Send_Notify

```
RBLE_STATUS RBLE_SCP_Server_Send_Notify ( uint16_t conhdl,
                                         RBLE_SCP_NOTIFY_INFO *notify_info)
```

指定されたデータを送信します。

結果は Server Role Notify 送信完了イベント(RBLE_SCP_EVENT_SERVER_SEND_NOTIFY_COMP)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
<i>notify_info</i>	送信する notify データ情報を指定
<i>data_len</i>	送信データサイズ
<i>data[]</i>	送信データ

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	SCP Server が有効状態以外のため実行不可

(8) RBLE_SCP_Server_Send_Indicate

```
RBLE_STATUS RBLE_SCP_Server_Send_Indicate ( uint16_t conhdl,
                                             RBLE_SCP_IND_INFO *ind_info)
```

指定されたデータを送信します。

結果は Server Role Indicate 送信完了イベント(RBLE_SCP_EVENT_SERVER_SEND_IND_COMP)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
<i>ind_info</i>	送信する indicate データ情報を指定
<i>data_len</i>	送信データサイズ
<i>data[]</i>	送信データ

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	SCP Server が有効状態以外のため実行不可

7.5.4 Sample Custom Profile EVENT 仕様

Sample Custom Profile(SCP)から通知されるイベントの説明を表 7-4に記述します。

表 7-4 Sample Custom Profile 通知 Event 一览

Role	イベント名	説明	parameter 構造体
Server	RBLE SCP EVENT SERVER _ENABLE_COMP	Enable 完了通知	struct RBLE_SCP_Server_Enable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }server_enable;
	RBLE SCP EVENT SERVER _DISABLE_COMP	Disable 完了通知	struct RBLE_SCP_Server_Disable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; RBLE_SCP_SERVER_PARAM server_info; }server_disable;
	RBLE SCP EVENT SERVER _ERROR_IND	エラー通知 ※未使用	struct RBLE_SCP_Server_Error_Ind_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }error_ind;
	RBLE SCP EVENT SERVER _SEND_NOTIFY_COMP	Notify 送信完了通知	struct RBLE_SCP_Server_Send_Notify_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }send_notify;
	RBLE SCP EVENT SERVER _SEND_IND_COMP	Indicate 送信完了通知	struct RBLE_SCP_Server_Send_Indicate_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }send_ind;
	RBLE SCP EVENT SERVER _CHG_INDNTF_IND	Client Configuration 变化通知	struct RBLE_SCP_Server_Cfg_Indntf_Ind_t{ uint16_t conhdl; uint8_t char_code; uint8_t reserved; uint16_t cfg_val; }cfg_indntf;
	RBLE SCP EVENT SERVER _CHG_CHAR_IND	Characteristic の 变化通知	struct RBLE_SCP_Server_Write_Chara_Ind_t{ uint16_t conhdl; uint8_t char_code; uint8_t reserved; uint8_t value[RBLE_SCPC_WRITE_CHAR_MAX]; }write_char;
	RBLE SCP EVENT SERVER _COMMAND_DISALLOWED_IND	コマンド拒否通知 イベント ※未使用	struct RBLE_SCP_Server_Command_Disallowed_Ind_t{ RBLE_STATUS status; uint8_t reserved; uint16_t opcode; }cmd_disallowed_ind;
Client	RBLE SCP EVENT CLIENT	Enable 完了	struct RBLE_SCP_Client_Enable_t{

	_ENABLE_COMP	通知	<pre>uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; RBLE_SCS_CONTENT scs; }client_enable;</pre>
	RBLE SCP EVENT_CLIENT _DISABLE_COMP	Disable 完了 通知	<pre>struct RBLE_SCP_Client_Disable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }client_disable;</pre>
	RBLE SCP EVENT_CLIENT _ERROR_IND	エラー通知 ※未使用	<pre>struct RBLE_SCP_Client_Error_Ind_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }error_ind;</pre>
	RBLE SCP EVENT_CLIENT _NOTIFY	Notify 受信通知	<pre>struct RBLE_SCP_Client_Notify_Ind_t{ uint16_t conhdl; uint8_t data_len; uint8_t data[]; }notify;</pre>
	RBLE SCP EVENT_CLIENT _INDICATE	Indicate 受信通知	<pre>struct RBLE_SCP_Client_Indicate_Ind_t{ uint16_t conhdl; uint8_t data_len; uint8_t data[]; }ind;</pre>
	RBLE SCP EVENT_CLIENT _READ_CHAR_RESPONSE	特性取得要求 応答イベント	<pre>struct RBLE_SCP_Client_Read_Char_Response_t{ uint16_t conhdl; uint8_t att_code; RBLE_ATT_INFO_DATA data; }rd_char_resp;</pre>
	RBLE SCP EVENT_CLIENT _WRITE_CHAR_RESPONSE	特性設定要求 応答イベント	<pre>struct RBLE_SCP_Client_Write_Char_Response_t{ uint16_t conhdl; uint8_t att_code; }wr_char_resp;</pre>
	RBLE SCP EVENT_CLIENT _COMMAND_DISALLOWED_IND	コマンド拒否 通知イベント ※未使用	<pre>struct RBLE_SCP_Client_Command_Disallowed_Ind_t{ RBLE_STATUS status; uint8_t reserved; uint16_t opcode; }cmd_disallowed_ind;</pre>

7.5.5 Sample Custom Profile サンプルプログラム制御方法

Sample Custom Profile(SCP)のサンプルプログラムの制御方法を以降に記述します。

本サンプルプログラムは、Server Role を Embedded 構成、Client Role を Modem 構成で動作させることを想定した初期設定になっています。

そのため、Embedded 構成の Server Role では外部からコマンド制御を行わずに動作するようになっています。動作の詳細は Server 側制御の説明を参照してください。

また、Client 側を Embedded 構成で動作させたい場合は、sample_app フォルダ以下にある、prf_sel.h の「USE_CUSTOM_DEMO」のマクロを無効にしてください。

(1) Client 側制御説明

Client 側のサンプルプログラムの制御方法を記述します。

GAP のコマンドを使用し、Server 側との接続を完了後に以下の手順でコマンドを発行することで、SCP 用のコマンドの発行が可能になります。

- ① Profile Test を選択(図の場合は 2 を発行)



図 7-14 Client 側初期 Menu 画面

- ② Sample Custom Profile を選択(図の場合は 7 を発行)

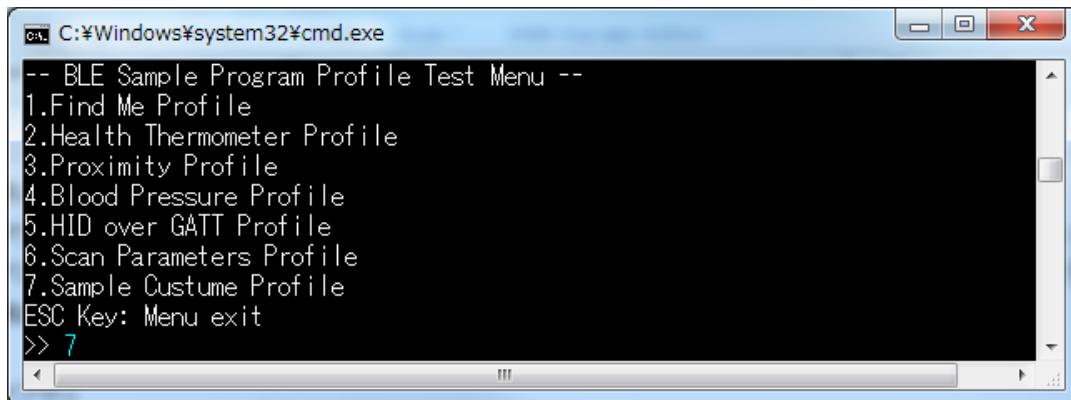


図 7-15 Client 側 Profile Test Menu 画面(Profile Test 選択後の画面)

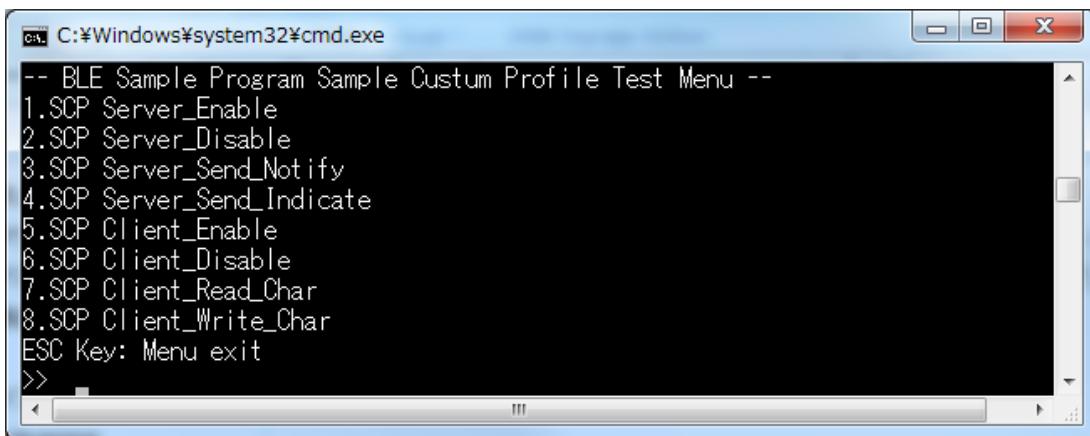


図 7-16 Client 側 Sample Custom Test Menu 画面(Sample Custom Profile 選択後の画面)

以降に Client 側のサンプルプログラムでコンソールより実行可能なコマンドとその説明を記述します。

コマンド番号	実行動作	引数	説明	発行例
1	Server Enable	-	Server 制御用コマンド	-
2	Server Disable	-	Server 制御用コマンド	-
3	Server Send Notify	-	Server 制御用コマンド	-
4	Server Send Indicate	-	Server 制御用コマンド	-
5	Client Enable	-	RBLE_SCP_Client_Enable 関数を実行し、Client Role を Enable 状態にします。 ※必ず RBLE_SCP_CON_CFG で動作します。	5
6	Client Disable	-	RBLE_SCP_Client_Disable 関数を実行し、Client Role を Disable 状態にします。	6
7	Client Read Char	char_code	RBLE_SCP_Client_Read_Char 関数を実行し、特性値の取得を行います。 コマンドに続けて値を指定することで、指定した characteristic を取得できます。 0:Notify の Client Characteristic Configuration (RBLE_SCP_SCS_NTF_CFG)を取得 1:Indicate の Client Characteristic Configuration (RBLE_SCP_SCS_IND_CFG)を取得 2:Interval の Characteristic (RBLE_SCP_SCS_INTERVAL)を取得 3:Notify Length の Characteristic (RBLE_SCP_SCS_NTF_LEN)を取得 4:Indicate Length の Characteristic (RBLE_SCP_SCS_IND_LEN)を取得	7 2
8	Client Write Char	char_code data	RBLE_SCP_Client_Write_Char 関数を実行し、特性値の取得を行います。 コマンドに続けて値を指定することで、指定した characteristic を取得できます。 また、書き込む値を続けて指定できます。 0:Notify の Client Characteristic Configuration (RBLE_SCP_SCS_NTF_CFG)を設定 1:Indicate の Client Characteristic Configuration	8 1 2

		(RBLE_SCP_SCS_IND_CFG)を設定 2:Interval の Characteristic (RBLE_SCP_SCS_INTERVAL)を設定 3:Notify Length の Characteristic (RBLE_SCP_SCS_NTF_LEN)を設定 4:Indicate Length の Characteristic (RBLE_SCP_SCS_IND_LEN)を設定	
--	--	--	--

(2) Server 側制御説明

電源 ON すると自動で接続待ち状態になります。

Client 側から接続を行い、接続が完了すると自動で Server の Enable を実行し、制御可能な状態になります。

Client 側から、Notify、Indicate の設定を行った後に RL78/G1D 評価ボードの SW2 を押すことで、Notify、Indicate が開始され、もう一度 SW2 を押すことで Notify、Indicate の通知が停止します。

Notify と Indicate は設定に応じて送信され、設定された間隔(Interval)で、設定されたデータ数を送信します。

※Interval は指定した値×10(ms) 間隔となります。

7.6 簡易サンプルプロファイル

GATT API を使用して作成した簡易サンプルプロファイルについての説明を以降に記述します。なお、本サンプルプログラムを使用する場合は、プロジェクトのコンパイルオプションに「USE_SIMPLE_SAMPLE_PROFILE」のマクロ定義を追加してください。

7.6.1 Characteristic 仕様

簡易サンプルプロファイルが保持している Characteristic を表 7-5 に記述します。

表 7-5 Simple Sample Custom Profile Characteristic 一覧

Characteristic 名	Properties	Format	説明
Switch State Characteristic UUID: 5BC18D80-A1F1-40AF-9043-C43692C18D7A	Notify	uint8_t	SW4 の押下・開放状態の通知に使用。0x00 の場合は開放、0x01 の場合は押下。
- Client Characteristic Configuration	Read/Write	uint16_t	Notify の ON/OFF を指定。
LED Control Characteristic UUID: 5BC143EE-A1F1-40AF-9043-C43692C18D7A	Read/Write	uint8_t	LED4 の点灯・消灯状態の設定・取得に使用。0x00 の場合は消灯、0x01 の場合は点灯。

7.6.2 簡易サンプルプロファイルのファイル構成

簡易サンプルプロファイルに関連するファイル構成を以下に記述します。

```

Renesas
└ BLE_Software_Ver_X_XX
  └ RL78_G1D
    └ Project_Source
      └ rBLE
        └ src
          └ sample_simple
            └ sam
              └ sams.c
              └ sams.h
            └ console.c
            └ console.h
            └ rble_sample_app_peripheral.c
            └ rble_sample_app_peripheral.h
      └ renesas
        └ src
          └ arch
            └ rl78
              └ prf_config.c
              └ prf_config.h
              └ ke_conf_simple.c
              └ db_handle.h

```

簡易サンプルプロファイル・簡易サンプルプログラム格納フォルダ
簡易サンプルプロファイル格納フォルダ
簡易サンプルプロファイルソースファイル
簡易サンプルプロファイルヘッダファイル
コンソールドライバソースファイル
コンソールドライバヘッダファイル
簡易サンプルプログラムソースファイル
簡易サンプルプログラムヘッダファイル

プロファイル向けパラメータ設定ソースファイル
プロファイル向けパラメータ設定ヘッダファイル
RWKE タスク定義ファイル
Attribute database handles ヘッダファイル

7.6.3 Simple Sample Profile の詳細

簡易サンプルプロファイルは、「Embedded 構成サンプルアプリケーション (r01an3319)」の Peripheral 向けサンプルアプリケーションと同一のものです。詳細は、「Embedded 構成サンプルアプリケーション (r01an3319)」のアプリケーションノートを参照してください。

7.7 RF テスタによる Direct Test Mode サンプルプログラム

RF 認証時に使用する RF テスタとの 2-Wire UART 接続による Direct Test Mode (DTM) に対応したサンプルプログラムを格納しています。

本サンプルプログラムは、以下の定義を有効 (= 1) にすることで、使用することができます。

```
#define __DTM2WIRE_UART_USE__
```

本サンプルプログラムに関連するファイル構成を以下に記述します。

Renesas	
└ BLE_Software_Ver_X_XX	
└ RL78_G1D	BLE MCU 向け BLE ソフトウェア格納フォルダ
└ Project_Source	
├ bleip	BLE スタック格納フォルダ
└ src	
└ rwble	
└ rwble_config.h	BLE ソフトウェアコンフィギュレーションヘッダファイル
└ renesas	
└ src	
├ arch	
└ r178	
└ arch_main.c	BLE ソフトウェアメインファイル
└ ke_conf.c	RWKE タスク管理ファイル
└ driver	
├ DTM2Wire	
└ DTM2Wire.c	2-Wire UART Direct Test Mode ドライバファイル
└ DTM2Wire.h	2-Wire UART Direct Test Mode ドライバヘッダファイル
└ uart	
└ uart.c	UART ドライバファイル
└ uart.h	UART ドライバヘッダファイル

本サンプルプログラムを有効にした場合、リセット起動直後に通常動作で起動するか、2-Wire UartによるDirect Test Modeで起動するかの切り替えを行います。Direct Test Modeで起動した場合、ポートレートは9600bpsに設定されます。

リセット起動時の起動モード判定は、Modem 構成／Embedded 構成の違いにより異なります。

(1) Modem 構成時

リセット起動後、シリアル受信エラーが発生しない場合、DTM モードで動作します。

(2) Embedded 構成時

RL78/G1D 評価ボードの SW2 を押したままリセットした場合、DTM モードで動作します。

各システム構成の違いによる起動シーケンスを図 7-17、図 7-18及び図 7-19に記述します。

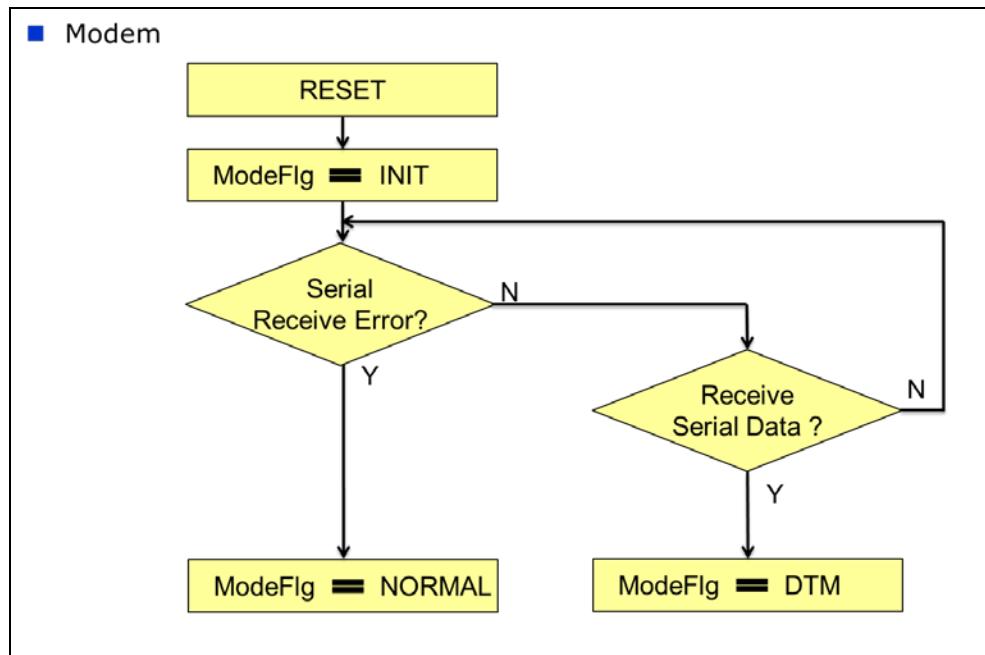


図 7-17 リセット起動時の起動モード判定（Modem 構成）

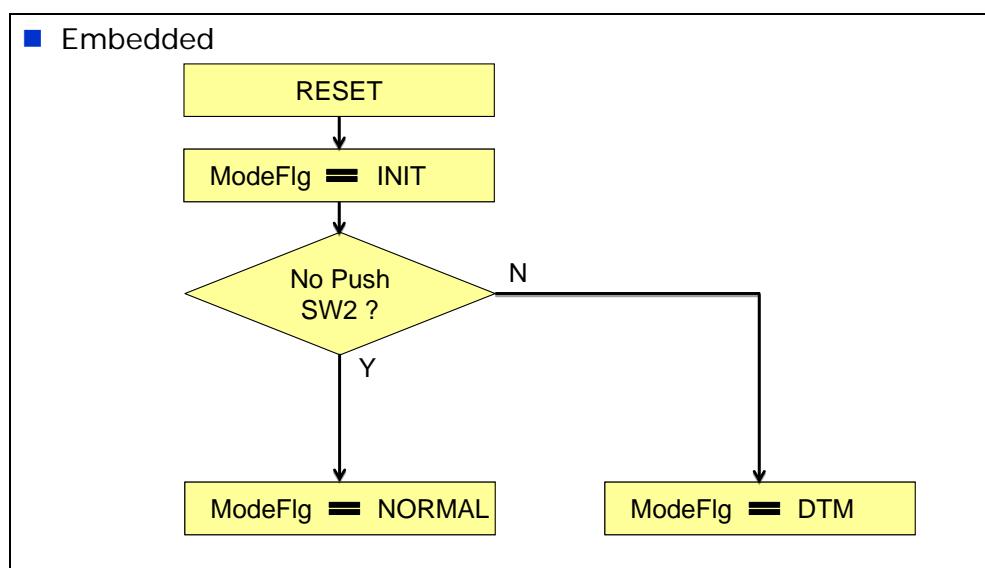


図 7-18 リセット起動時の起動モード判定（Embedded 構成）

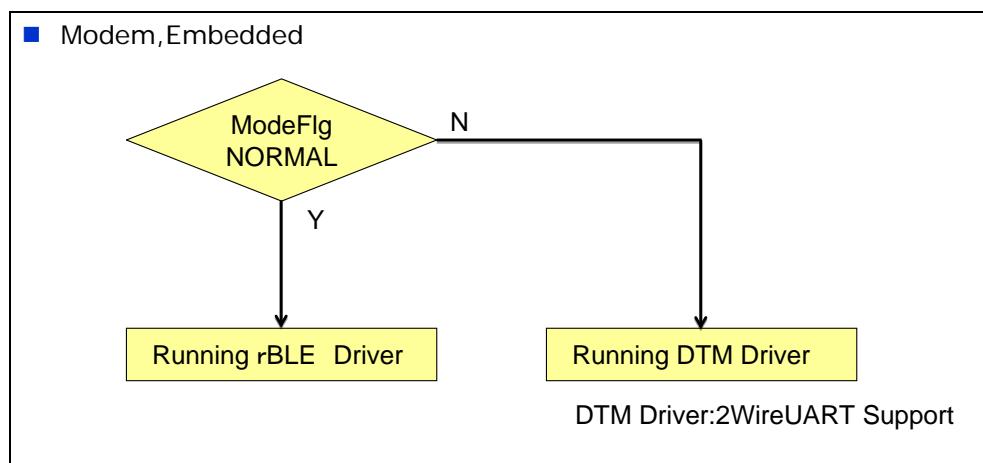


図 7-19 起動モード判定後の動作

7.8 Embedded 構成の printf プログラム

Embedded 構成のサンプルプログラムでは、標準入出力へのアクセスを `console.c` のプログラムで実現しています。

`printf`においては、標準ライブラリの `printf` 関数を使用せず、`console.h` に定義された以下の定義マクロによって、`console.c` に定義された `Printf` 関数を呼び出します。

```
#define printf Printf
```

`Printf` 関数では、可変長引数リストのデータを書式文字列に従ってバッファへ書き込み、そのバッファをシリアルに出力します。このバッファサイズは以下の定義マクロにより、デフォルトで 80Byte に設定しています。

```
#define STREAM_MEMORY_MAX_LINE_SIZE 80
```

このため、書式変換後のサイズが 80Byte を超える文字列を出力する必要がある場合は、バッファサイズの調整を行ってください。

7.9 FW アップデートサンプルプログラム

FW アップデートのサンプルプログラムについての説明を以降に記載します。

FW アップデートでは FW アップデート用データを送信するデバイス(Sender デバイス)と FW アップデート用データを受信し、FW アップデートされるデバイス(Receiver デバイス)が必要となります。

以下に FW アップデートの動作イメージを記載します。

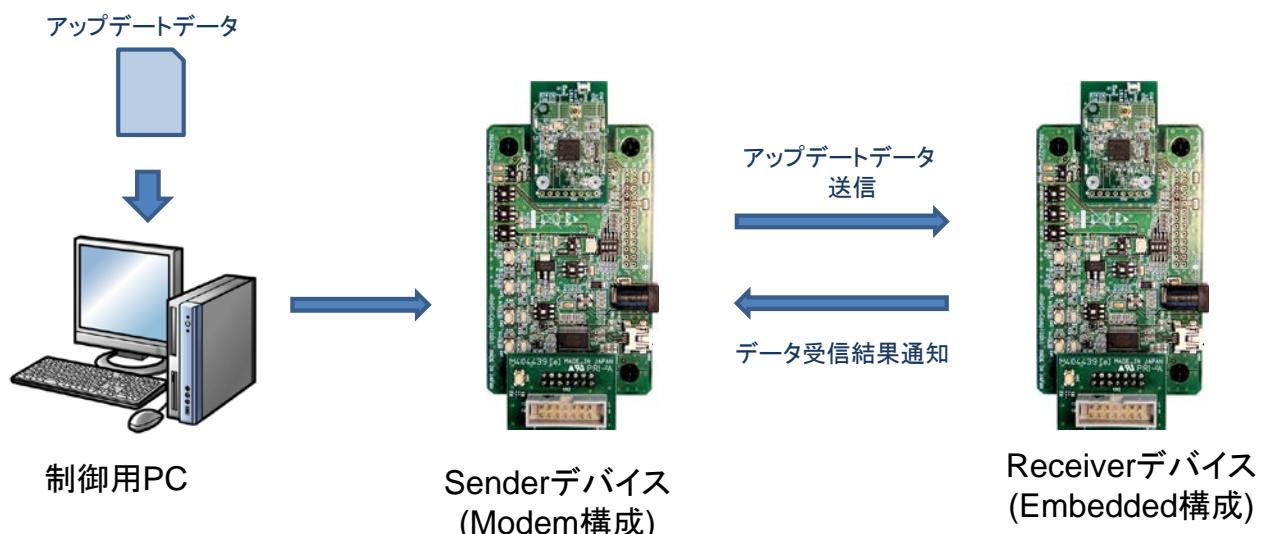


図 7-20 FW アップデート動作イメージ図

本サンプルでは Sender デバイス側は Modem 構成、Receiver デバイス側は Embedded 構成で動作させます。

7.9.1 FW アップデートプロファイル仕様

FW アップデートプロファイルは Sender Role と Receiver Role の 2 つの Role を保持しています。

FW アップデートプロファイルが保持している Characteristic を表 7-6に記述します。

表 7-6 FW アップデートプロファイル Characteristic 機能一覧

Characteristic 名	Properties	format	説明
Data Control Characteristic	Write	uint8_t[]	データ送信の制御情報を Write Request で書き込む
Data Characteristic	Write without Response	uint8_t[]	1~20byte のアップデートデータを Write Command で書き込む

7.9.2 FW アップデートサンプルプログラムファイル構成

FW アップデート用サンプルプログラムに関するファイル構成を以下に示します。

BLE_Software_Ver_X_XX	
└ BLE_Sample	PC 用サンプルプログラム格納フォルダ
└ src	
└ rBLE	BLE サンプルプログラム格納フォルダ
└ src	
└ include	FW Update profile ヘッダファイル
└ rble_api_fwup.h	Sample Profile フォルダ
└ sample_profile	FW Update Profile 格納フォルダ
└ fwup	FW Update Profile Sender ファイル
└ fwups.c	
└ sample_app	サンプルプログラムファイル
└ rble_sample_app.c	FW アップデート用サンプルプログラムファイル(Sender)
└ rble_fw_up_sender_app.c	FW アップデート用サンプル格納フォルダ
└ Fwup	バイナリデータ格納フォルダ
└ bin	CA78K0R でビルドした ROM ファイルを変換したバイナリ格納フォルダ
└ ca78k0r	Embedded 構成用バイナリファイル(PXP/FMP/ANP 版)
└ RL78_G1D_CE(PXP,FMP,ANP).bin	Embedded 構成用バイナリファイル(HTP,BLP,HRP 版)
└ RL78_G1D_CE(HTP,BLP,HRP).bin	CC-RL でビルドした ROM ファイルを変換したバイナリ格納フォルダ
└ ccrl	Embedded 構成用バイナリファイル(PXP/FMP/ANP 版)
└ RL78_G1D_CCE(PXP,FMP,ANP).bin	Embedded 構成用バイナリファイル(HTP,BLP,HRP 版)
└ RL78_G1D_CCE(HTP,BLP,HRP).bin	IAR v2 でビルドした ROM ファイルを変換したバイナリ格納フォルダ
└ iar_v2	Embedded 構成用バイナリファイル(PXP/FMP/ANP 版)
└ RL78_G1D_IE(PXP,FMP,ANP).bin	Embedded 構成用バイナリファイル(HTP,BLP,HRP 版)
└ RL78_G1D_IE(HTP,BLP,HRP).bin	hex データ格納フォルダ
└ hex	Sender デバイス用 ROM ファイルの格納フォルダ
└ Sender	CA78K0R でビルドした ROM ファイル(FWUP Sender 版)
└ RL78_G1D_CM(Sender).hex	CC-RL でビルドした ROM ファイル(FWUP Sender 版)
└ RL78_G1D_CCM(Sender).hex	IAR v2 でビルドした ROM ファイル(FWUP Sender 版)
└ RL78_G1D_IM_V2(Sender).hex	Receiver デバイス用 ROM ファイルの格納フォルダ
└ Receiver	CA78K0R でビルドした ROM ファイルの格納フォルダ
└ ca78k0r	Embedded 構成用 ROM ファイル格納フォルダ
└ Embedded	Embedded 構成用 ROM ファイル(PXP/FMP/ANP 版)
└ RL78_G1D_CE(PXP,FMP,ANP).hex	Embedded 構成用 ROM ファイル(HTP,BLP,HRP 版)
└ RL78_G1D_CE(HTP,BLP,HRP).hex	CC-RL でビルドした ROM ファイルの格納フォルダ
└ ccrl	Embedded 構成用 ROM ファイル格納フォルダ
└ Embedded	Embedded 構成用 ROM ファイル(PXP/FMP/ANP 版)
└ RL78_G1D_CCE(PXP,FMP,ANP).hex	Embedded 構成用 ROM ファイル(HTP,BLP,HRP 版)
└ RL78_G1D_CCE(HTP,BLP,HRP).hex	IAR v2 でビルドした ROM ファイルの格納フォルダ
└ iar_v2	Embedded 構成用 ROM ファイル格納フォルダ
└ Embedded	Embedded 構成用 ROM ファイル(PXP/FMP/ANP 版)
└ RL78_G1D_IE(PXP,FMP,ANP).hex	Embedded 構成用 ROM ファイル(HTP,BLP,HRP 版)
└ RL78_G1D_IE(HTP,BLP,HRP).hex	rBLE フォルダ
└ RL78_G1D	
└ Project_Source	
└ rBLE	
└ src	
└ include	FW Update profile ヘッダファイル
└ rble_api_fwup.h	Sample Profile フォルダ
└ sample_profile	FW Update Profile 格納フォルダ
└ fwup	FW Update Profile Receiver ファイル
└ fwupr.c	Sample Program ファイル
└ sample_app	FW アップデート用サンプルプログラムファイル(Receiver)
└ rble_fw_up_receiver_app.c	

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

以下の手順に従って、サンプルプログラムを動作させるための準備を行います。

- (1) Sender デバイスとして動作させる RL78/G1D 評価ボードに、以下のいずれかの Hex ファイルを書き込みます。

格納先フォルダ : BLE_Software_Ver_X_XX\BLE_Sample\Fwup\hex\Sender

ファイル名 :

- RL78_G1D_CM(Sender).hex
- RL78_G1D_CCM(Sender).hex
- RL78_G1D_IM_V2(Sender).hex

- (2) Receiver デバイスとして動作させる RL78/G1D 評価ボードに、下記フォルダに格納された Hex ファイルを書き込みます。

格納先フォルダ名 : BLE_Software_Ver_X_XX\BLE_Sample\Fwup\hex\Receiver\<環境名>

CA78K0R 環境 : ca78k0r

CC-RL 環境 : ccrl

IAR V2 環境 : iar_v2

【注】 FW アップデート機能を確認する環境に合わせて書き込む Hex ファイルを選択してください。

- (3) 下記フォルダに FW アップデート用データを格納してください。FW アップデート用データは Hex ファイルをバイナリ形式に変換したものを使用します。

格納先フォルダ名 : BLE_Software_Ver_X_XX\BLE_Sample\project\windows\Exe

【注】 Receiver デバイスに書き込んだ Hex ファイルと同一環境のバイナリファイルを格納してください。たとえば Receiver デバイスに RL78_G1D_CE(PXP,FMP,ANP).hex を書き込んだ場合、RL78_G1D_CE(HTP,BLP,HRP).hex をバイナリ形式に変換したもの上記フォルダに格納します。

Hex ファイルをバイナリ形式に変換したものをサンプルとして、下記のフォルダに格納しています。

格納先フォルダ名 : BLE_Software_Ver_X_XX\BLE_Sample\Fwup\bin

- (4) Sender デバイスの接続されたシリアルポート番号を指定し、rBLE_Sample.exe を起動します。このとき、ポートレートは Sender デバイスに書き込んだ Hex ファイルに合わせて 76800bps を指定します。

Modem 構成におけるサンプルプログラムの起動方法は 5.1 を参照してください。

- (5) Receiver デバイスについては、5.3 を参考にサンプルプログラムを起動します。

7.9.3 FW アップデートプロファイル IF 関数仕様

FW アップデートプロファイル(FWUP)の IF 関数仕様を以降に記述します。

(1) RBLE_FWUP_Sender_Enable

```
RBLE_STATUS RBLE_FWUP_Sender_Enable ( uint16_t conhdl,
                                         uint8_t con_type,
                                         RBLE_FWUS_CONTENT *fwus,
                                         RBLE_FWUPS_EVENT_HANDLER call_back )
```

Sender Role を有効化します。

初回に接続する場合は con_type に RBLE_FWUP_CON_CFG を指定し、Receiver のサービス発見を行う必要があります。

初回に取得したサービスの情報を保持しておき、2 回目以降の有効化時に引数 fwus に情報を指定し、con_type に RBLE_FWUP_CON_NORMAL を指定することで、サービス発見の再実行を行わないため、高速な Role の有効化が行えます。

結果は Sender 有効完了イベント(RBLE_FWUP_EVENT_SENDER_ENABLE_COMP)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
<i>con_type</i>	接続方法指定
<i>fwus</i>	FWUP のハンドル情報(con_type に RBLE_FWUP_CON_NORMAL を指定時のみ有効)
<i>call_back</i>	イベント通知を行う Callback 用関数の指定

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	FWUP Sender が無効状態以外のため実行不可

(2) RBLE_FWUP_Sender_Disable

<code>RBLE_STATUS RBLE_FWUP_Sender_Disable (uint16_t conhdl)</code>

Sender Role を無効化します。

結果は Sender 無効化完了イベント(RBLE_FWUP_EVENT_SENDER_DISABLE_COMP)で通知されます。

Parameters:

<code>conhdl</code>	コネクションハンドル
---------------------	------------

Return:

<code>RBLE_OK</code>	正常終了
<code>RBLE_PARAM_ERR</code>	パラメータ異常
<code>RBLE_STATUS_ERROR</code>	FWUP Sender が有効状態以外のため実行不可

(3) RBLE_FWUP_Sender_Write_Data_Cntl

<code>RBLE_STATUS RBLE_FWUP_Sender_Write_Cntl (uint16_t conhdl,</code>

`uint8_t type,`

`uint8_t block_num,`

`uint16_t data_size)`

Data Control Characteristic へ設定を行います。

`type` に RBLE_FWUP_DATA_SEND_START を指定した場合は、`block_num`、`data_size` の値が有効です。それ以外の場合は `block_num`、`data_size` に指定した値は無効です。

結果は特性値設定要求応答イベント(RBLE_FWUP_EVENT_SENDER_WRITE_CHAR_RES)で通知されます。

Parameters:

<code>conhdl</code>	コネクションハンドル
<code>type</code>	制御コマンドタイプを指定。 RBLE_FWUP_DATA_SEND_START データ送信を開始 RBLE_FWUP_DATA_SEND_COMP データ送信完了(指定サイズの送信が完了) RBLE_FWUP_DATA_CHECK_WRITE データ書き込み確認 RBLE_FWUP_DATA_SEND_FINISH 全データの送信が完了 RBLE_FWUP_DATA_CHECK_UPDATE FW アップデート完了確認
<code>block_num</code>	コードフラッシュの書き込み先ブロック番号を指定(0~255) <code>type</code> に RBLE_FWUP_DATA_SEND_START を指定した場合のみ有効
<code>data_size</code>	コードフラッシュへの書き込みデータサイズを指定(4~1024 4byte 単位) <code>type</code> に RBLE_FWUP_DATA_SEND_START を指定した場合のみ有効

Return:

<code>RBLE_OK</code>	正常終了
<code>RBLE_PARAM_ERR</code>	パラメータ異常
<code>RBLE_STATUS_ERROR</code>	FWUP Sender が有効状態以外のため実行不可

(4) RBLE_FWUP_Sender_Write_Data

```
RBLE_STATUS RBLE_FWUP_Sender_Write_Data ( uint16_t conhdl,
                                            uint8_t *data,
                                            uint8_t data_size )
```

Data Characteristic へ設定を行います。

Parameters:

<i>conhdl</i>	コネクションハンドル
<i>*data</i>	Receiver に書き込むデータの先頭アドレスを指定
<i>data_size</i>	設定するデータサイズを指定(1~20byte)

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	FWUP Sender が有効状態以外のため実行不可

(5) RBLE_FWUP_Receiver_Enable

```
RBLE_STATUS RBLE_FWUP_Receiver_Enable ( uint16_t conhdl,
                                         RBLE_FWUPR_EVENT_HANDLER call_back )
```

FWUP 機能の Receiver Role を有効にします。

結果は Receiver Role 有効化完了イベント(RBLE_FWUP_EVENT_RECEIVER_ENABLE_COMP)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
<i>call_back</i>	イベント通知を行う Callback 用関数の指定

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	FWUP Receiver が無効状態以外のため実行不可

(6) RBLE_FWUP_Receiver_Disable

```
RBLE_STATUS RBLE_FWUP_Receiver_Disable ( uint16_t conhdl )
```

Receiver Role を無効化します。

結果は Receiver 無効化完了イベント(RBLE_FWUP_EVENT_RECEIVER_DISABLE_COMP)で通知されます。

Parameters:

<i>conhdl</i>	コネクションハンドル
---------------	------------

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	FWUP Receiver が有効状態以外のため実行不可

(7) RBLE_FWUP_Receiver_Send_Data_Cntl_Res

```
RBLE_STATUS RBLE_FWUP_Receiver_Send_Data_Cntl_Res ( uint16_t conhdl,
                                                    RBLE_STATUS status )
```

Data Control Characteristic への Write Request に対する Response を返します。

status には Data Control Characteristic に設定された制御コマンドに応じた結果を設定します。

コマンドが RBLE_FWUP_DATA_SEND_START の場合、ブロック番号とサイズが正常ならば、RBLE_OK を、不正ならば RBLE_ERR を設定します。

コマンドが RBLE_FWUP_DATA_SEND_COMP もしくは RBLE_FWUP_DATA_SEND_FINISH の場合、指定されたサイズ分だけデータが受信できていれば RBLE_OK を、指定されたサイズと異なるなど、受信したデータに問題があった場合は RBLE_ERR を設定します。

コマンドが RBLE_FWUP_DATA_CHECK_WRITE の場合、フラッシュへの書き込みが正常に完了していれば RBLE_OK を、失敗した場合は RBLE_ERR を設定します。

コマンドが RBLE_FWUP_DATA_CHECK_UPDATE の場合は、FW アップデートが完了していれば RBLE_OK を、失敗した場合は RBLE_ERR を設定します。

Parameters:

<i>conhdl</i>	コネクションハンドル	
		Write コマンドに応じた結果を設定
<i>status</i>	RBLE_OK	成功
	RBLE_ERR	失敗

Return:

<i>RBLE_OK</i>	正常終了
<i>RBLE_PARAM_ERR</i>	パラメータ異常
<i>RBLE_STATUS_ERROR</i>	FWUP Receiver が有効状態以外のため実行不可

7.9.4 FW アップデートプロファイル EVENT 仕様

FW アップデートプロファイル(FWUP)から通知されるイベントの説明を表 7-7に記述します。

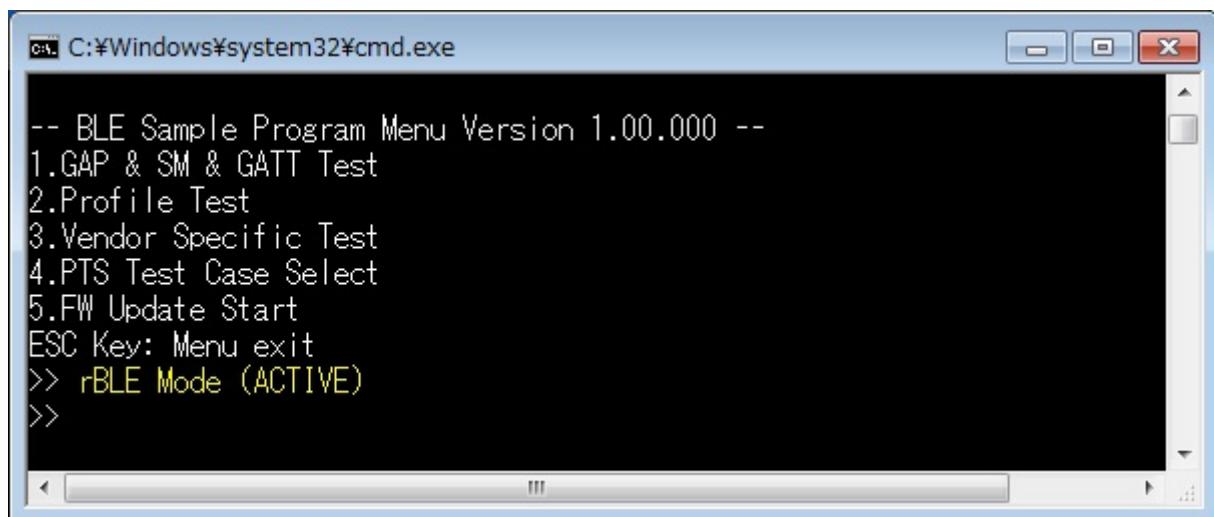
表 7-7 FW アップデートプロファイル通知 Event 一覧

Role	イベント名	説明	parameter 構造体
Receiver	RBLE_FWUP_EVENT_RECEIVER_ENABLE_COMP	Enable 完了通知	struct RBLE_FWUP_Receiver_Enable_t{ uint16_t conhdl; RBLE_STATUS status; }receiver_enable;
	RBLE_FWUP_EVENT_RECEIVER_DISABLE_COMP	Disable 完了通知	struct RBLE_FWUP_Receiver_Disable_t{ uint16_t conhdl; RBLE_STATUS status; }receiver_disable;
	RBLE_FWUP_EVENT_RECEIVER_CHG_DATA_CNTL_IND	Data Control 設定変化通知	struct RBLE_FWUP_Receiver_Chg_Data_Cntl_Ind_t{ uint16_t conhdl; uint8_t type; uint8_t block_num; uint16_t data_size; }data_cntl_ind;
	RBLE_FWUP_EVENT_RECEIVER_CHG_DATA_IND	Data 設定変化通知	struct RBLE_FWUP_Receiver_Chg_Data_Ind_t{ uint16_t conhdl; uint8_t data_size; uint8_t data[RBLE_FWUP_DATA_MAX]; }data_ind;
Sender	RBLE_FWUP_EVENT_SENDER_ENABLE_COMP	Enable 完了通知	struct RBLE_FWUP_Sender_Enable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; RBLE_FWUS_CONTENT fwus; }sender_enable;
	RBLE_FWUP_EVENT_SENDER_DISABLE_COMP	Disable 完了通知	struct RBLE_FWUP_Sender_Disable_t{ uint16_t conhdl; RBLE_STATUS status; }sender_disable;
	RBLE_FWUP_EVENT_SENDER_WRITE_CHAR_RES	特性設定要求応答イベント	struct RBLE_FWUP_Sender_Write_Char_Res_t{ uint16_t conhdl; uint8_t att_code; }wr_char_resp;

7.9.5 FW アップデートサンプルプログラム制御方法

7.9.2に従って、Sender デバイス、Receiver デバイスのサンプルを起動させた場合、コンソールに以下のよ
うな内容が表示されています。

※Receiver デバイス側のコンソールには「5.FW Update Start」コマンドは表示されません。



```
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
5.FW Update Start
ESC Key: Menu exit
>> rBLE Mode (ACTIVE)
>>
```

図 7-21 サンプル起動直後のコンソール画面

この状態から FW アップデートを行うための制御手順について以降に記載します。

(1) 5.5の手順を使用して、Sender デバイス (Master) に Receiver デバイス(Slave)の BD アドレスを取得さ
せます。

(2) Receiver デバイスを FW アップデートモードに遷移させるために SW2(図 7-22の赤枠)を押します。

※このスイッチを押した後、Receiver デバイスは FW アップデートが完了するまで、コンソールからの
コマンド入力を受け付けません。

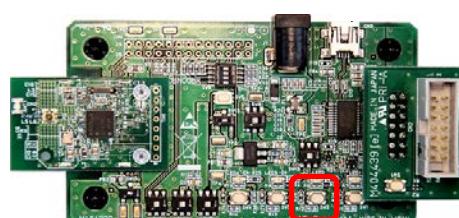


図 7-22 FW アップデートモードへの切り替えスイッチ

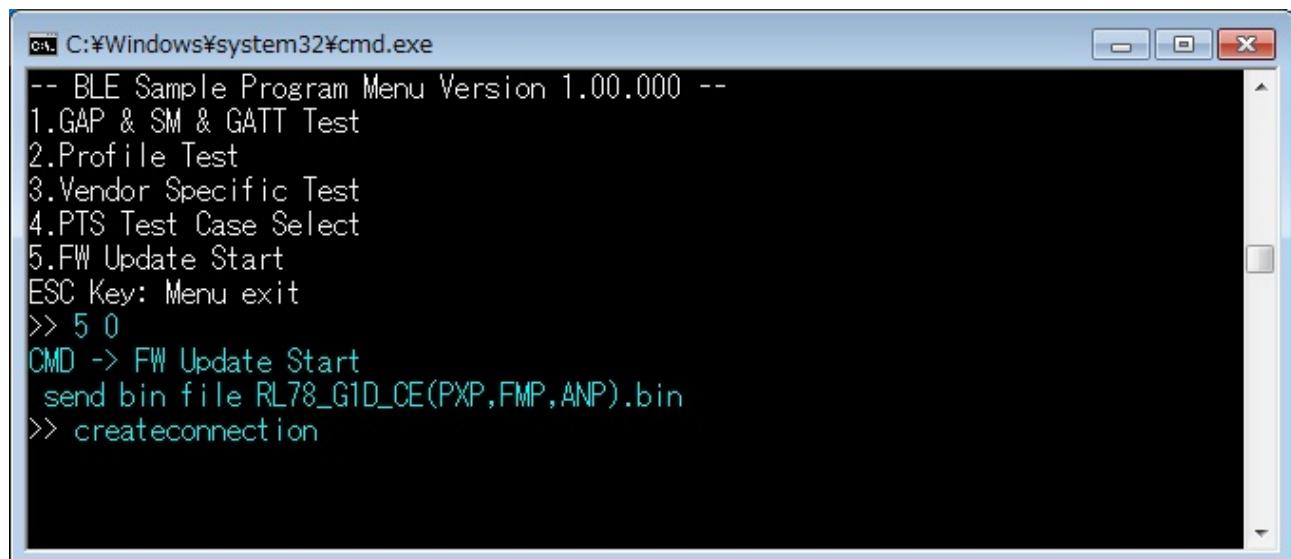
(3) Sender デバイスの「5.FW Update Start」コマンドを実行します。

この際に、送信したい FW アップデート用データを下の表に記載されている番号で指定します。

以下に FW アップデート用データとコマンドの対応を記載します。

番号	ファイル名
0	RL78_G1D_CE(PXP,FMP,ANP).bin
1	RL78_G1D_CE(HTP,BLP,HRP).bin
2	RL78_G1D_IE(PXP,FMP,ANP).bin
3	RL78_G1D_IE(HTP,BLP,HRP).bin
4	RL78_G1D_CCE(PXP,FMP,ANP).bin
5	RL78_G1D_CCE(HTP,BLP,HRP).bin

また、以下に RL78_G1D_CE(PXP,FMP,ANP).bin を送信する場合のコマンド発行例を記載します。



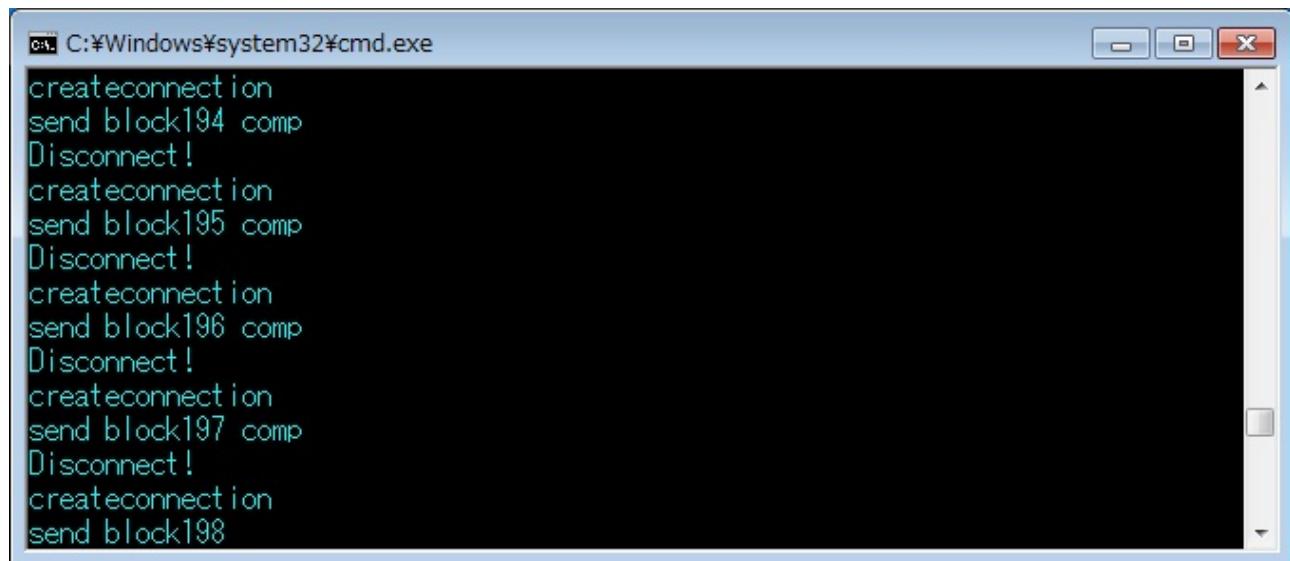
```
C:\Windows\system32\cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
5.FW Update Start
ESC Key: Menu exit
>> 5 0
CMD -> FW Update Start
  send bin file RL78_G1D_CE(PXP,FMP,ANP).bin
>> createconnection
```

図 7-23 FW アップデート開始時のコンソール画面(Sender デバイス側)

(4) 「5.FW Update Start」 コマンドを実行すると、以降は FW アップデートが完了するまで自動でアプリケーションが動作します。

以下に FW アップデートデータ送信中の Sender デバイス側のコンソール画面を記載します。

※送信中は接続→データ送信→切断を Block 単位で繰り返し行います。



```
C:\Windows\system32\cmd.exe
createconnection
send block194 comp
Disconnect!
createconnection
send block195 comp
Disconnect!
createconnection
send block196 comp
Disconnect!
createconnection
send block197 comp
Disconnect!
createconnection
send block198
```

図 7-24 FW アップデート中のコンソール画面(Sender デバイス側)

(5) FW アップデートが完了すると、Sender デバイス側のコンソールに「fw update finish」と表示されます。

また、Receiver デバイスはリセットされ、サンプル起動直後のコンソール画面に戻り、コンソールからのコマンド発行が可能になります。

7.10 FW アップデートサンプルプログラムを使用するためのプロジェクト設定方法

FW アップデートサンプルプログラムを使用するためのプロジェクトの設定方法について、以降に記載します。

7.10.1 Receiver デバイス

(1) CS+ for CA,CX のプロジェクト設定

CS+ for CA,CX の場合は、以下の手順でプロジェクトの設定を行います。

- (1) Modem 構成または Embedded 構成のプロジェクトを起動します。
- (2) プロジェクトツリーの中から BLE_Emb(サブプロジェクト)内のビルド・ツールを選択します。
- (3) 共通オプションタブの定義マクロを選択し、定義を以下のように変更します。

`noUSE_FW_UPDATE_PROFILE → USE_FW_UPDATE_PROFILE`

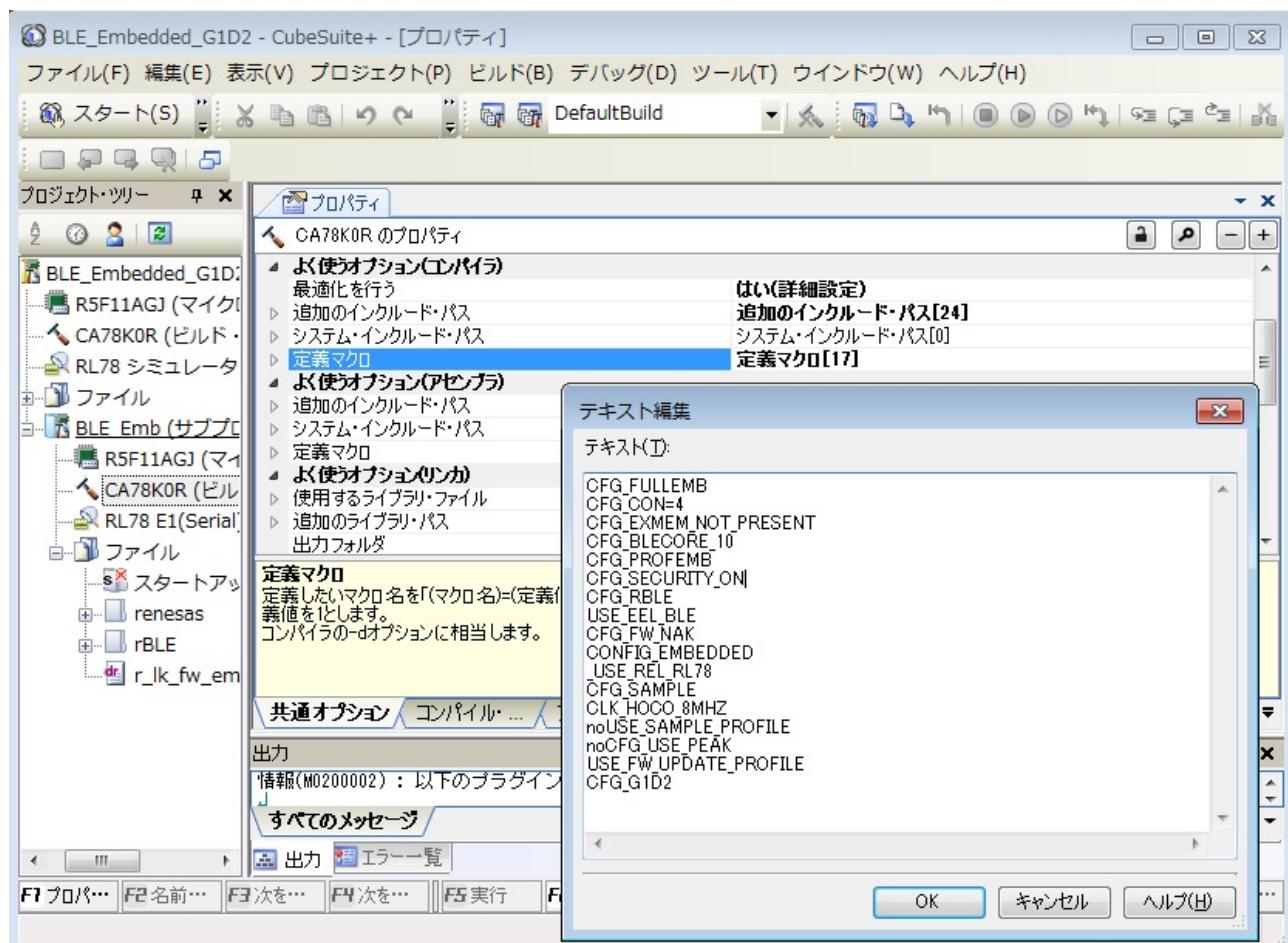


図 7-25 定義マクロ変更(CS+ for CA,CX)

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

- (4) FW アップデート用の dr ファイルに変更するために、以下のファイルを選択し、ビルド設定タブの「ビルド対象とする」を「はい」に変更します。

Embedded : r_lk_fw_emb.dr

Modem : r_lk_fw_mdm.dr

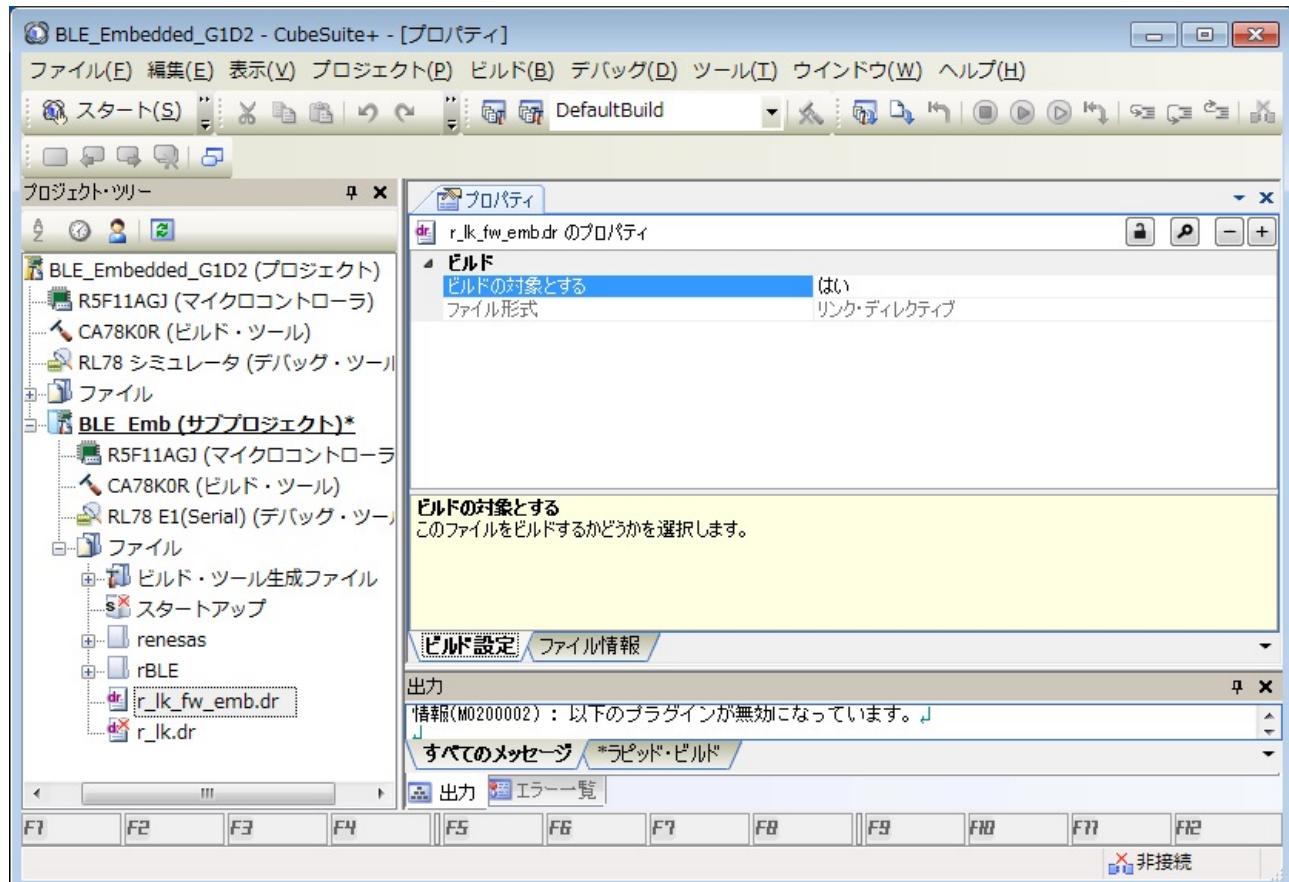


図 7-26 dr ファイル変更(CS+ for CA,CX)

- (5) ビルドを実行します。

(2) e² studio のプロジェクト設定

e² studio の場合は、以下の手順でプロジェクトの設定を行います。

- (1) e² studio を起動し、ワークスペースを開きます。
- (2) プロジェクト・エクスプローラーから、rBLE_Emb または rBLE_Mdm のプロジェクトを右クリックし、コンテキストメニューの Renesas Tool Settings を選択します。
- (3) ツール設定タブの左側ツリーより、[Compiler]→[ソース]と辿り、右側の定義マクロの中から以下の定義を変更します。

noUSE_FW_UPDATE_PROFILE → USE_FW_UPDATE_PROFILE

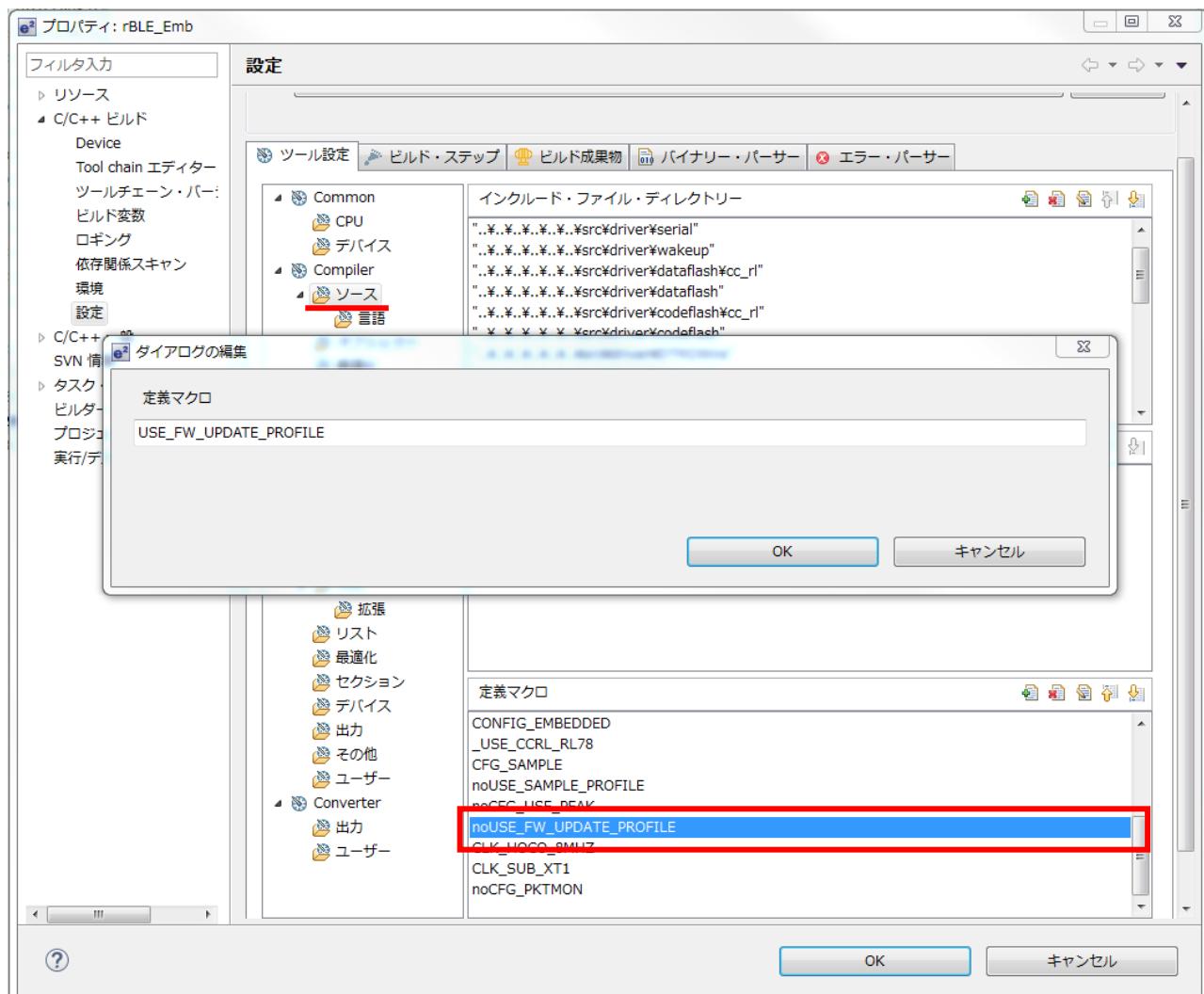


図 7-27 定義マクロ変更(e2stUDIO)

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

- (4) ツール設定タブの左側ツリーより、[Linker]→[セクション]と辿り、右側のインポートボタンを押下し、以下の FW アップデート用のセクション情報ファイル(esi)を選択します。

Embedded : renesas¥tools¥project¥e2studio¥BLE_EMBEDDED¥rBLE_Emb¥sect_emb_fwup.esi
Modem : renesas¥tools¥project¥e2studio¥BLE_Modem¥rBLE_Mdm¥sect_mdm_fwup.esi

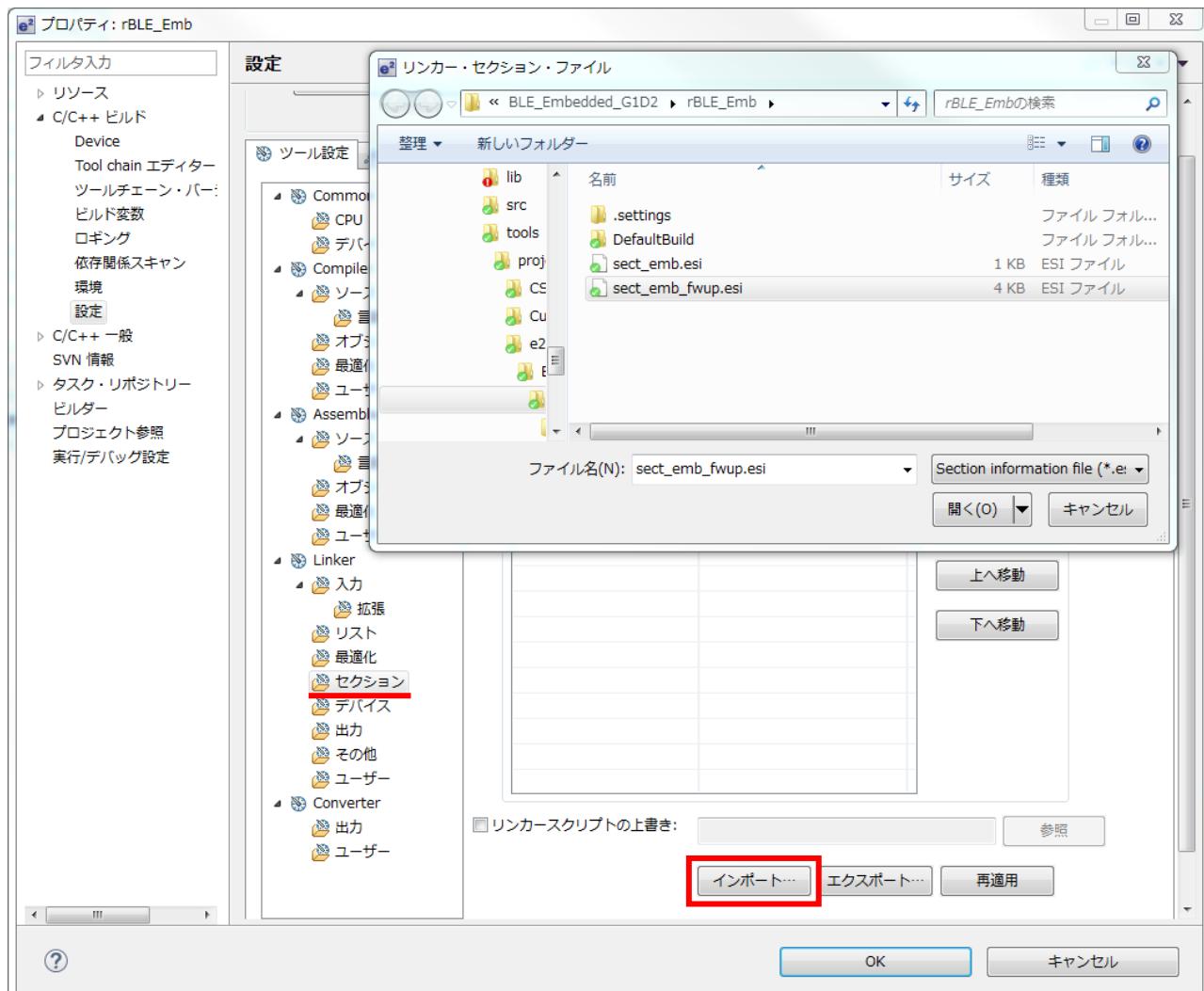


図 7-28 セクション情報ファイル(esi)のインポート (e² studio)

- (5) OK を押下し、変更を適用します。

- (6) ビルドを実行します。

7.10.2 Sender デバイス

Sender デバイス側の環境の作成方法を記載します。

Sender デバイス側では FW アップデート用データを Windows のサンプルプログラムから受け取るため、動作周波数を 32MHz に、UART 通信の速度を 76800bps に変更します。

※低クロックでも動作は可能ですが、FW アップデートにかかる時間が長くなってしまいます。

(1) UART 通信速度変更

UART ドライバのソースコードを変更し、動作周波数が 32MHz の場合に、UART 通信の速度が 76800bps になるように変更します。

Renesas/RL78_G1D/Project_Source/renesas/src/driver/uart/uart.c の serial_init() 関数内の処理を以下のように変更します。

※赤字が変更箇所

```
#if (1)
#ifndef CONFIG_EMBEDDED
/* MCK = fclk/n = 1MHz */
write_sfr(SPS0L, (uint8_t)((read_sfr(SPS0L) | UART_VAL_SPS_2MHZ)));

/* baudrate 4800bps(when MCK = 1MHz) */
write_sfwp(UART_TXD_SDR, (uint16_t)0x1800U);
write_sfwp(UART_RXD_SDR, (uint16_t)0x1800U);
#else /*CONFIG_EMBEDDED*/
...
#endif SERIAL_U_2WIRE
#if (1)
#ifndef CONFIG_EMBEDDED
/* if baudrate is 4800bps, set enable */
stop_flg = false;
#else /*CONFIG_EMBEDDED*/
...

```

(2) 動作周波数変更

動作周波数の変更については Bluetooth Low Energy プロトコルスタックユーザーズマニュアルを参照し、32MHz に設定してください。

7.10.3 FW アップデート環境を作成する際の注意事項

- ・関数の強制リンクについて

FW アップデートではコード領域のうちアプリケーション、プロファイルのコード領域のみを更新します。

そのため、FW アップデートの前後でリンクするランタイムライブラリと標準ライブラリを変更することはできません。

※アップデート対象外のコード領域から、ランタイムライブラリと標準ライブラリがアクセスできなくなるためです。

このため、予めランタイムライブラリと標準ライブラリを強制的にリンクさせておき、FW アップデート前後で、リンクされるランタイムライブラリと標準ライブラリが変更されないようにする必要があります。

CS+ for CA,CX 環境の場合のリンクの例については

Renesas/RL78_G1D/Project_Source/renesas/src/arch/r78/main.c の runtime_dummy_func() 関数を参照してください。

7.11 参考文献

1. Bluetooth Core Specification v4.2, Bluetooth SIG
2. Find Me Profile Specification v1.0, Bluetooth SIG
3. Immediate Alert Service Specification v1.0, Bluetooth SIG
4. Proximity Profile Specification v1.0, Bluetooth SIG
5. Link Loss Service Specification v1.0, Bluetooth SIG
6. Tx Power Service Specification v1.0, Bluetooth SIG
7. Health Thermometer Profile Specification v1.0, Bluetooth SIG
8. Health Thermometer Service Specification v1.0, Bluetooth SIG
9. Device Information Service Specification v1.1, Bluetooth SIG
10. Blood Pressure Profile Specification v1.0, Bluetooth SIG
11. Blood Pressure Service Specification v1.0, Bluetooth SIG
12. HID over GATT Profile Specification v1.0, Bluetooth SIG
13. HID Service Specification v1.0, Bluetooth SIG
14. Battery Service Specification v1.0, Bluetooth SIG
15. Scan Parameters Profile Specification v1.0, Bluetooth SIG
16. Scan Parameters Service Specification v1.0, Bluetooth SIG
17. Heart Rate Profile Specification v1.0, Bluetooth SIG
18. Heart Rate Service Specification v1.0, Bluetooth SIG
19. Cycling Speed and Cadence Profile Specification v1.0, Bluetooth SIG
20. Cycling Speed and Cadence Service Specification v1.0, Bluetooth SIG
21. Cycling Power Profile Specification v1.0, Bluetooth SIG
22. Cycling Power Service Specification v1.0, Bluetooth SIG
23. Glucose Profile Specification v1.0, Bluetooth SIG
24. Glucose Service Specification v1.0, Bluetooth SIG
25. Time Profile Specification v1.0, Bluetooth SIG
26. Current Time Service Specification v1.0, Bluetooth SIG
27. Next DST Change Service Specification v1.0, Bluetooth SIG
28. Reference Time Update State Service Specification v1.0, Bluetooth SIG
29. Alert Notification Service Specification v1.0, Bluetooth SIG
30. Alert Notification Profile Specification v1.0, Bluetooth SIG
31. Location and Navigation Service Specification v1.0, Bluetooth SIG
32. Location and Navigation Profile Specification v1.0, Bluetooth SIG
33. Phone Alert Status Service Specification v1.0, Bluetooth SIG
34. Phone Alert Status Profile Specification v1.0, Bluetooth SIG
35. Bluetooth SIG Assigned Numbers <https://www.bluetooth.com/specifications/assigned-numbers>
36. Services & Characteristics UUID <https://www.bluetooth.com/specifications/gatt>
37. Personal Health Devices Transcoding White Paper v1.2, Bluetooth SIG

7.12 用語説明

用語	英語	説明
サービス	Service	サービスは GATT サーバから GATT クライアントへ提供され、GATT サーバはインターフェースとしていくらかの特性を公開します。サービスは公開された特性へのアクセス手順について規定します。
プロファイル	Profile	1つ以上のサービスを使用してユースケースの実現を可能にします。使用するサービスは各プロファイルの仕様にて規定されます。
特性	Characteristic	特性はサービスを識別する値で、各サービスにて公開する特性やそのフォーマットが定義されます。
ロール	Role	役割。それぞれのデバイスが、プロファイルやサービスで規定される役割を果たすことで、ユースケースの実現が可能になります。
クライアント特性コンフィギュレーション記述子	Client Characteristic Configuration Descriptor	クライアント特性コンフィギュレーション記述子を持つ特性値の GATT サーバからの送信(Notification / Indication)を制御するために使用します。
コネクションハンドル	Connection Handle	リモートデバイスとの接続を識別するための Controller スタックによって決定されるハンドルです。ハンドルの有効範囲は 0x0000～0x0EFF です。
UUID	Universally Unique Identifier	一意に識別するための識別子です。BLE 規格ではサービスや特性等を識別するために 16bit の UUID が定義されています。
BD アドレス	Bluetooth Device Address	Bluetooth デバイスを識別するための 48bit のアドレスです。BLE 規格ではパブリックアドレスとランダムアドレスが規定されており、少なくともどちらか一方をサポートする必要があります。
パブリックアドレス	Public Address	IEEE に登録し割り当てられた 24bit の OUI(Organizational Unique Identifier)を含むアドレスです。
ランダムアドレス	Random Address	乱数を含むアドレスで、以下の 3 つに分類されます。 スタティックアドレス Non-resolvable private アドレス Resolvable private アドレス
スタティックアドレス	Static Address	上位 2bit は共に 1 で、残 46bit は全てが 1 または 0 ではない乱数からなるアドレスです。電源断まではそのスタティックアドレスを変更できません。

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

Non-resolvable private アドレス	Non-resolvable private Address	上位 2bit は共に 0 で、残 46bit は全てが 1 または 0 ではない乱数からなるアドレスです。スタティックおよびパブリックアドレスと等しくてはなりません。 短い期間でアドレスを変更することで攻撃者からの追跡を困難にする目的で使用されます。
Resolvable private アドレス	Resolvable private Address	IRK と 24bit の乱数から生成されるアドレスです。上位 2bit は 0 と 1、上位の残 22bit は全てが 1 または 0 ではない乱数で、下位 24bit は IRK と上位の乱数を元に計算されます。 短い期間でアドレスを変更することで攻撃者からの追跡を困難にする目的で使用されます。IRK を対向機に配布することで、対向機はその IRK を使用してデバイスを特定することができます。
Broadcaster	Broadcaster	GAP のロールのひとつで、Advertising データを送信します。
Observer	Observer	GAP のロールのひとつで、Advertising データを受信します。
Central	Central	GAP のロールのひとつで、物理リンクの確立を行います。Link Layer では Master と呼ばれます。
Peripheral	Peripheral	GAP のロールのひとつで、物理リンクの確立を受け入れます。Link Layer では Slave と呼ばれます。
Advertising	Advertising	接続確立や、データ送信の目的の為に特定チャネル上でデータを送信します。
Scan	Scan	Advertising データを受信します。Scan には、ただ受信するのみの Passive Scan と、SCAN_REQ を送信することで追加情報を要求する Active Scan があります。
White List	White List	接続済みやボンディング済みなどの既知デバイスを White List に登録しておくことで、Advertising データや接続要求を受け取ることを許可するデバイスをフィルタリングすることができます。
デバイス名	Device Name	Bluetooth デバイスに任意につけられたデバイスを識別するためのユーザフレンドリーな名前です。 BLE 規格では、GAP の特性として GATT サーバによって対向機に公開されます。
Reconnection Address	Reconnection Address	Non-resolvable private アドレスを使用して、短い期間でアドレスを変更する場合、攻撃者だけでなく対向機もデバイスの特定が困難になります。そのため対向機の公開する Reconnection Address 特性に新しい Reconnection Address を設定することで再接続時のアドレスを通知します。
スキャンインターバル	Scan Interval	Advertising データの受信を行う間隔です。

Bluetooth® Low Energy プロトコルスタック サンプルプログラムアプリケーションノート

スキャンウインドウ	Scan Window	スキャンインターバルごとに Advertising データの受信をおこなう期間です。
コネクションインターバル	Connection Interval	接続確立後に定期的にデータの送受信を行う間隔です。
コネクションイベント	Connection Event	コネクションインターバルごとにデータの送受信を行う期間です。
スレーブレイテンシー	Slave Latency	スレーブレイテンシーの回数分のコネクションインターバルにおいて Slave デバイスは受信をする必要がありません。
スーパービジョンタイムアウト	Supervision Timeout	対向機からの応答がなく、リンクが切断されたとみなすタイムアウト時間です。
Passkey Entry	Passkey Entry	ペアリング方式の一つで、互いのデバイスで 6 行の数値入力または、一方で 6 行の数値表示、もう一方でその数値入力を行います。
Just Works	Just Works	ペアリング方式の一つで、ユーザアクションを必要としません。
OOB	OOB	ペアリング方式の一つで、Bluetooth 以外の通信方式で取得したデータを使用してペアリングを行います。
IRK	Identity Resolving Key	Resolvable private アドレスの生成や解決に用いる 128bit のキーです。
CSRK	Connection Signature Resolving Key	データ署名の作成および、受信データの署名の確認に使用される 128bit のキーです。
LTK	Long Term Key	暗号化に使用される 128bit のキーです。使用的キーサイズはペアリング時に同意されたサイズになります。
STK	Short Term Key	キー交換時に暗号化するために使用される 128bit のキーです。TK を用いて生成されます。
TK	Temporary Key	STK 生成に必要となる 128bit のキーです。Just Works の場合は 0、Passkey Entry は入力された 6 行の数値、OOB は OOB データが TK の値となります。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

<http://japan.renesas.com/>

お問合せ先

<http://japan.renesas.com/contact/>

Bluetooth は、Bluetooth SIG, Inc., U.S.A.の登録商標です。
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。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
0.80	2012.09.14	—	初版発行
0.81	2012.09.28	—	Direct Test Mode およびドキュメント番号を追加
0.82	2012.10.05	—	アプリケーション開発上の注意に追記 PTS 使用時のサンプルプログラム実行方法を追加 その他、誤記修正
0.83	2012.10.16	—	使用方法の COM ポート設定値を更新
1.00	2012.11.19	—	Embedded 構成に関する記述を追加し、BLE ソフトウェア V1.00 に適用
1.01	2012.12.26	—	英文マニュアルと同期
1.10	2013.03.27	—	シリアル通信、Custom Profile、2-Wire Uart DTM に関する記述を追加し、BLE ソフトウェア V2.00 に適用
1.11	2013.04.12	—	コマンドプロンプト画面のキャプチャ画像を差し替え
1.12	2013.06.28	2	フォルダ構成を更新
1.13	2013.11.29	— 2 10 31 93 111	BLE ソフトウェア V2.30 に適用 フォルダ構成を更新(HRP/CSCP/CPP/ANP/LNP を追加) サンプルプログラム使用方法に注釈を追加 HRP/CSCP/CPP/ANP/LNP の使用方法を追加 Embedded 構成の printf プログラムを追加 参考文献に追加プロファイルの仕様書を追記
1.14	2014.09.19	— — 4 2 5 8 94 105 —	Bluetooth v4.1 に準拠 BLE ソフトウェア V0.50 に適用 VC++ のバージョンを更新 フォルダ構成を削除 動作環境と開発環境を更新 EXE ファイルのフォルダパスを更新 FW アップデートサンプルプログラムの使用方法を追加 FW アップデート環境作成方法を追加 その他、誤記修正
1.15	2014.01.30	— 109	BLE ソフトウェア V0.90 に適用 FW アップデートの UART ボー・レートを変更
1.16	2015.04.17	— — 94	BLE ソフトウェア V1.00 に適用 シリアル通信に IIC を追加 FW アップデートプロファイル仕様を追加
1.17	2015.07.10	— 89	BLE ソフトウェア V1.01 に適用 Direct Test Mode の Modem 構成時モード切り替え仕様変更
1.18	2015.10.30	— 5, 107 89	BLE ソフトウェア V1.10 に適用 CS+ fo CC/ e ² studio(CC-RL)に関連する記述を追加 Direct Test Mode 動作時のボー・レート情報を追記
1.19	2016.08.31	— — 5 8 9 60 71 79 95	BLE ソフトウェア V.1.20 に適用 CD の記述をパッケージに変更 Renesas Flash Programmer のバージョンを更新 5.1 パラメータ変更方法の記載場所を付録から移動 表 5-1に"UART2 線分岐接続"を追記 7.2 APP MCU のシリアル通信ドライバの要件と実装フロー チャートの章構成を変更 ROM/RAM サイズの参考値を追記 ファイル構成を更新 FW アップデートサンプルプログラムのファイル構成を更新
1.20	2017.07.31	-	Microsoft Visual Studio のバージョンを更新。 IAR V1 関連の記載を削除。

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. 未使用端子の処理

【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

【注意】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. リザーブアドレス（予約領域）のアクセス禁止

【注意】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、将来の機能拡張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。

4. クロックについて

【注意】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。

5. 製品間の相違について

【注意】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、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品デ-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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製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）、交通用信号機器、
防災・防犯装置、各種安全装置等
- 当社製品は、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（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
- 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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。
- 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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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ペンジング処理等、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。
- 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。お客様が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。
- 本資料に記載されている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。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外國為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
- お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
- 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。

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その総株主の議決権の過半数を直接または間接に保有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。



ルネサス エレクトロニクス株式会社

営業お問合せ窓口

<http://www.renesas.com>

営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：<https://www.renesas.com/contact/>