Bio Sensing Software Platform

BLE 高速転送カスタムプロファイルを使ったデータ転送応用例

要旨

本アプリケーションノートは、RL78/G1D 用 Bluetooth® low energy プロトコルスタック（以下 BLE ソフトウェアと略します）付属のカスタムプロファイルを基に開発した高速転送プロファイル（Renesas Rapid Transfer Profile、以下 RRTP と略します）を説明します。

サンプルプログラムには、Central 機器、Peripheral 機器それぞれで高速転送プロファイルの動作を確認するためのデモソフトウェアが含まれます。

また、この RRTP を使った高速データ転送動作デモを紹介します。

動作デモでは、Peripheral 機器（サーバ）として血圧測定評価キットを利用し、RL78/H1D 上のΔΣAD コンバータで取得した測定値データを血圧測定評価キット上の RL78/G1D 経由で RRTP により送信します。対向機の Central 機器（クライアント）として RL78/G1D 評価ボードを利用し、受信したデータをテキストデータに変換し、PC に転送します。受信した測定データをグラフ化して波形を確認することができます。

動作確認デバイス

RL78/G1D

動作確認ボード

Bluetooth Central 機器および Peripheral 機器：

- RL78/G1D 評価ボード（RTK0EN0001D01001BZ）
- https://www.renesas.com/jp/ja/solutions/key-technology/connectivity/bluetooth-smart/evaluation-board.html

Bluetooth Peripheral 機器（高速データ転送動作デモ用）：

- RL78/H1D 用血圧測定評価キット（RTK0EH0003S02001BR）
- https://www.renesas.com/jp/ja/products/software-tools/boards-and-kits/renesas-solution-starter-kits/bpm-evaluation-kits-rl78h1d.html

関連ドキュメント

- Bluetooth low energy プロトコルスタック　アプリケーション作成ガイド　(R01AN2768JJ)
- Bluetooth low energy プロトコルスタック　センサアプリケーション　(R01AN4159JJ)
- Bluetooth low energy プロトコルスタック　ユーザーズマニュアル　(R01UW0095JJ)
- Renesas Flash Programmer V3.05　フラッシュ書き込みソフトウェア　ユーザーズマニュアル (R20UT4307JJ0130)
- Renesas Solution Starter Kit　BPMEK 用 GUI Tool 操作マニュアル(R01AN4396JJ)
- Renesas Solution Starter Kit　Blood Pressure Monitoring Evaluation Kit for RL78/H1D　ユーザーズマニュアル(R20UT4128JJ)

目次

1.　概要

動作対象の BLE 機器には RL78/G1D 評価ボードを使用します。

Central 機器用と Peripheral 機器用の 2 台を用意し、「図 1-1 サンプルプログラムの動作環境概要」に示す構成でサンプルプログラムを実行します。

2 つの RL78/G1D 評価ボードを使用し、一方を Peripheral 機器としてダミーデータの作成と高速転送プロファイルを用いたデータ送信し、他方を Central 機器として高速転送プロファイルによるデータ受信を行います。

Peripheral 機器のサンプルプログラムは起動すると自動的に Advertising 状態になり、Central 機器との接続が確認されるとデータの送信を開始します。Central 機器のサンプルプログラムは起動すると登録された BD Address を Advertising 状態の Peripheral 機器から検索し、BD Address が一致した Peripheral 機器と接続します。接続が確認されるとデータの受信を開始します。

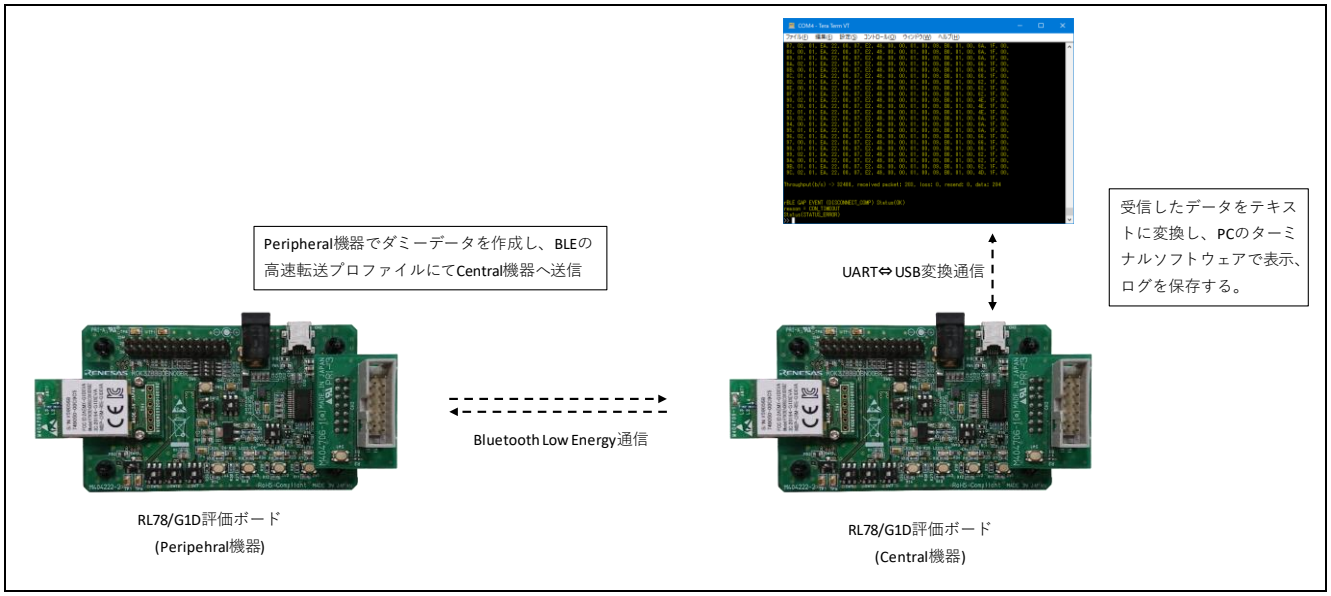


図 1-1 サンプルプログラムの動作環境概要

1.1　サンプルプログラムの機能

- Peripheral 機器：
  アドバタイズの自動制御

  1 パケット 20 バイトのダミーデータ作成と送信処理

  Connection Interval 毎に最大 4 個の Notification パケットを送信

- Central 機器：
  接続対象の Peripheral 機器の BD Address（データフラッシュに保存）の検索と接続の自動化

  Connection Interval 値設定（データフラッシュに保存）

  ターミナル・ソフトウェアを接続すると、受信データを 16 進数のテキストデータとして表示

1.2　用語／略語

表 1-1　用語／略語一覧表

用語／略語	内容
BLE	Bluetooth low energy の略です。
BLE 機器	BLE Central 機器および BLE Peripheral 機器を示します。
Central 機器	BLE Central 機器を示します。
Notification	Characteristic Value Notification（特性値通知）の略で、サーバのタイミングでクライアントにデータを送信（通知）する手順を示します。
Peripheral 機器	RL78/G1D を Peripheral 設定で利用する機器を示します。
RFP	Renesas Flash Programmer の略です。
Renesas Rapid Transfer Profile	本アプリケーションノートに示す高速転送プロファイルを示します。
RRTP	Renesas Rapid Transfer Profile の略です。
RRTS	Renesas Rapid Transfer Service の略です。
対向機	データ受信側の BLE Central 機器を示します。
血圧測定評価キット	Blood Pressure Monitoring Evaluation Kit for RL78/H1D（RTK0EH0003S02001BR）を示します。

1.3　使用上の注意事項／制限事項

(1)　Peripheral 機器－Central 機器間の電波等の状況により、受信データがパケット単位（1 パケット＝ユーザデータ 20 バイト）で欠損する場合があります。また、BLE 機器間の再送処理でパケットの受信順番が入れ替わる場合があります。

(2)　Peripheral 機器－Central 機器間の接続を行う際、接続が失敗する場合があります。

接続失敗時に、Peripheral 機器が Advertising 状態を維持している場合、Central 機器側で再度接続処理を行ってください。
接続失敗時に、Peripheral 機器が Advertising 以降の状態に進んでいる場合、Central 機器と Peripheral 機器共に再起動してください。

(3)　Peripheral（サーバ）機器 ⇒ Central（クライアント）機器でのユーザデータのスループットの最大値は 85.3 kbps です。

最大スループットが得られる条件は、以下の場合です。
— Connection Interval：7.5 ミリ秒
— 1 Connection Interval 期間中に 4 個の Notification パケットを送信

最大スループット設定の場合、パケット欠損や受信データ入れ替わりが発生しやすくなります。
1 Connection Interval 期間中の送信パケット数を 3 パケット以下（スループット理論値 64kbps 以下）に設定するとパケット欠損の発生頻度は低下します。

(4)　メイン・システム・クロック周波数を 32MHz にすることを推奨します。

1.4　開発環境

開発環境を表 1-2 に示します。

表 1-2　開発環境

項目	内容
統合開発環境	ルネサス エレクトロニクス製 CS+ for CC V7.00.00
C コンパイラ	ルネサス エレクトロニクス製 CC-RL V1.07.00

1.5　ファイル構造

ファイル構造を表 1-3 に示します。

ROM_Files フォルダ内に同一ファイル名（rBLE_Emb_CCRL.mot ファイル）が複数あるため、これらのファイルに限り、識別目的でファイル更新日時を記載しています。

表 1-3　ファイル構造

r01an4569jj0101-bsspf-communication	File creation date
├──r01an4569jj0101-bsspf-communication.pdf	
├──ROM_Files	
│　├──BPMEK_RL78_G1D	
│　│　└──rBLE_Emb_CCRL.mot	: 2019/02/27 11:14
│　├──BPMEK_RL78_H1D	
│　│　└──BPM_solution_H1D.mot	
│　├──RL78_G1D_central	
│　│　└──rBLE_Emb_CCRL.mot	: 2019/02/22 13:54
│　└──RL78_G1D_peripheral	
│　　└──rBLE_Emb_CCRL.mot	: 2019/02/26 16:58
├──tools	
│　├──IIR_filter_coefficient_OSR1024_reference.csv	
│　├──RRTP sample.log	
│　└──RRTP_graph_sample_rev101.xlsm	
├──unique_code	
│　└──sample_central.ruc	
├──workspace_RL78_G1D_central	
│　└──Project_Source	
│　　├──bleip	
│　　│　└──src	
│　　│　　├──common	
│　　│　　│　└──co_bt.h	
│　　│　　└──rwble	
│　　│　　　├──rwble.h	
│　　│　　　└──rwble_config.h	
│　　├──rBLE	
│　　│　└──src	
│　　│　　├──include	
│　　│　　│　├──rble.h	
│　　│　　│　├──rble_api.h	
│　　│　　│　├──rble_api_custom.h	
│　　│　　│　├──rble_api_fwup.h	
│　　│　　│　├──rble_api_rrtp.h	
│　　│　　│　├──rble_app.h	
│　　│　　│　├──rble_rwke.h	
│　　│　　│　├──rble_trans.h	
│　　│　　│　└──rscip_api.h	
│　　│　　├──sample_app	
│　　│　　│　├──beacon_app.c	
│　　│　　│　├──beacon_app.h	
│　　│　　│　├──Console.c	
│　　│　　│　├──console.h	
│　　│　　│　├──menu_sel.c	
│　　│　　│　├──menu_sel.h	
│　　│　　│　├──rble_app_rrtpc.c	
│　　│　　│　├──rble_app_rrtpc.h	
│　　│　　│　├──rble_fw_up_receiver_app.c	
│　　│　　│　├──rble_sample_app.c	
│　　│　　│　├──rble_sample_app_anp.c	
│　　│　　│　├──rble_sample_app_blp.c	
│　　│　　│　├──rble_sample_app_cpp.c	
│　　│　　│　├──rble_sample_app_cscp.c	

```
│   │   │   ├──rble_sample_app_custom.c
│   │   │   ├──rble_sample_app_fmp.c
│   │   │   ├──rble_sample_app_fwup.c
│   │   │   ├──rble_sample_app_gap_sm_gatt.c
│   │   │   ├──rble_sample_app_glp.c
│   │   │   ├──rble_sample_app_hogp.c
│   │   │   ├──rble_sample_app_hrp.c
│   │   │   ├──rble_sample_app_htp.c
│   │   │   ├──rble_sample_app_lnp.c
│   │   │   ├──rble_sample_app_pasp.c
│   │   │   ├──rble_sample_app_pxp.c
│   │   │   ├──rble_sample_app_rscp.c
│   │   │   ├──rble_sample_app_scpp.c
│   │   │   ├──rble_sample_app_tip.c
│   │   │   └──rble_sample_app_vendor.c
│   │   ├──sample_profile
│   │   │   ├──fwup
│   │   │   │   └──fwupr.c
│   │   │   ├──rrtp
│   │   │   │   ├──rtpc.c
│   │   │   │   └──rtps.c
│   │   │   └──scp
│   │   │       ├──scpc.c
│   │   │       └──scps.c
│   │   └──sample_simple
│   │       ├──console.c
│   │       ├──console.h
│   │       ├──rble_sample_app_peripheral.c
│   │       ├──rble_sample_app_peripheral.h
│   │       └──sam
│   │           ├──sam.h
│   │           ├──sams.c
│   │           └──sams.h
│   └──renesas
│       ├──config
│       │   └──split
│       │       └──emb
│       │           ├──r_lk.dr
│       │           ├──r_lk_fw_emb.dr
│       │           ├──r_lk_fw_modem.dr
│       │           ├──r_lk_modem.dr
│       │           ├──r_lk_modem_R5F11AGH.dr
│       │           └──r_lk_R5F11AGH.dr
│       ├──lib
│       │   ├──BLE_CONTROLLER_lib.a
│       │   ├──BLE_CONTROLLER_LIB.lib
│       │   ├──BLE_CONTROLLER_LIB_CCRL.lib
│       │   ├──BLE_HOST_lib.a
│       │   ├──BLE_HOST_lib.lib
│       │   ├──BLE_HOST_lib_CCRL.lib
│       │   ├──BLE_PROFILE_ANP_lib.a
│       │   ├──BLE_PROFILE_ANP_LIB.lib
│       │   ├──BLE_PROFILE_ANP_LIB_CCRL.lib
│       │   ├──BLE_PROFILE_BLP_lib.a
│       │   ├──BLE_PROFILE_BLP_LIB.lib
│       │   ├──BLE_PROFILE_BLP_LIB_CCRL.lib
│       │   ├──BLE_PROFILE_CPP_lib.a
│       │   ├──BLE_PROFILE_CPP_LIB.lib
│       │   ├──BLE_PROFILE_CPP_LIB_CCRL.lib
│       │   ├──BLE_PROFILE_CSP_lib.a
│       │   ├──BLE_PROFILE_CSP_LIB.lib
│       │   ├──BLE_PROFILE_CSP_LIB_CCRL.lib
│       │   ├──BLE_PROFILE_FMP_lib.a
```

```
│   │   ├──BLE_PROFILE_FMP_LIB.lib
│   │   ├──BLE_PROFILE_FMP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_GLP_lib.a
│   │   ├──BLE_PROFILE_GLP_LIB.lib
│   │   ├──BLE_PROFILE_GLP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_HGP_lib.a
│   │   ├──BLE_PROFILE_HGP_LIB.lib
│   │   ├──BLE_PROFILE_HGP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_HRP_lib.a
│   │   ├──BLE_PROFILE_HRP_LIB.lib
│   │   ├──BLE_PROFILE_HRP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_HTP_lib.a
│   │   ├──BLE_PROFILE_HTP_LIB.lib
│   │   ├──BLE_PROFILE_HTP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_LNP_lib.a
│   │   ├──BLE_PROFILE_LNP_LIB.lib
│   │   ├──BLE_PROFILE_LNP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_PAP_lib.a
│   │   ├──BLE_PROFILE_PAP_LIB.lib
│   │   ├──BLE_PROFILE_PAP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_PXP_lib.a
│   │   ├──BLE_PROFILE_PXP_LIB.lib
│   │   ├──BLE_PROFILE_PXP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_RSP_lib.a
│   │   ├──BLE_PROFILE_RSP_LIB.lib
│   │   ├──BLE_PROFILE_RSP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_SCP_lib.a
│   │   ├──BLE_PROFILE_SCP_LIB.lib
│   │   ├──BLE_PROFILE_SCP_LIB_CCRL.lib
│   │   ├──BLE_PROFILE_TIP_lib.a
│   │   ├──BLE_PROFILE_TIP_LIB.lib
│   │   ├──BLE_PROFILE_TIP_LIB_CCRL.lib
│   │   ├──BLE_PROFILES_COMMON_lib.a
│   │   ├──BLE_PROFILES_COMMON_LIB.lib
│   │   ├──BLE_PROFILES_COMMON_LIB_CCRL.lib
│   │   ├──BLE_rBLE_lib.a
│   │   ├──BLE_rBLE_lib.lib
│   │   └──BLE_rBLE_lib_CCRL.lib
│   ├──src
│   │   ├──arch
│   │   │   └──rl78
│   │   │       ├──arch.h
│   │   │       ├──arch_main.c
│   │   │       ├──config.h
│   │   │       ├──db_handle.h
│   │   │       ├──fw_update_count0.c
│   │   │       ├──fw_update_count1.c
│   │   │       ├──hw_config.h
│   │   │       ├──ke_conf.c
│   │   │       ├──ke_conf_simple.c
│   │   │       ├──ll
│   │   │       │   └──ll.h
│   │   │       ├──main.c
│   │   │       ├──prf_config.c
│   │   │       ├──prf_config.h
│   │   │       ├──prf_config_host.c
│   │   │       ├──prf_sel.h
│   │   │       ├──rble_core_config.c
│   │   │       ├──rble_core_config.h
│   │   │       ├──rble_modem_config.c
│   │   │       ├──rble_modem_config.h
│   │   │       ├──rwble_mem.c
│   │   │       ├──rwble_mem.h
```

```
│   │   │   │   └──rwke_api.h
│   │   │   ├──compiler
│   │   │   │   ├──ccrl
│   │   │   │   │   ├──cstart.asm
│   │   │   │   │   ├──hdwinit.asm
│   │   │   │   │   └──stkinit.asm
│   │   │   │   ├──compiler.h
│   │   │   │   └──iodefine.h
│   │   │   ├──driver
│   │   │   │   ├──codeflash
│   │   │   │   │   ├──cc_rl
│   │   │   │   │   │   ├──fsl.h
│   │   │   │   │   │   ├──fsl.lib
│   │   │   │   │   │   └──fsl_types.h
│   │   │   │   │   ├──codeflash.c
│   │   │   │   │   ├──codeflash.h
│   │   │   │   │   ├──cs
│   │   │   │   │   │   ├──fsl.h
│   │   │   │   │   │   ├──fsl.lib
│   │   │   │   │   │   └──fsl_types.h
│   │   │   │   │   └──iar_v2
│   │   │   │   │       ├──fsl.a
│   │   │   │   │       ├──fsl.h
│   │   │   │   │       └──fsl_types.h
│   │   │   │   ├──csi
│   │   │   │   │   ├──csi.c
│   │   │   │   │   └──csi.h
│   │   │   │   ├──dataflash
│   │   │   │   │   ├──cc_rl
│   │   │   │   │   │   ├──eel.h
│   │   │   │   │   │   ├──eel.lib
│   │   │   │   │   │   ├──eel_types.h
│   │   │   │   │   │   ├──fdl.h
│   │   │   │   │   │   ├──fdl.lib
│   │   │   │   │   │   └──fdl_types.h
│   │   │   │   │   ├──cs
│   │   │   │   │   │   ├──eel.h
│   │   │   │   │   │   ├──eel.lib
│   │   │   │   │   │   ├──eel_types.h
│   │   │   │   │   │   ├──fdl.h
│   │   │   │   │   │   ├──fdl.lib
│   │   │   │   │   │   └──fdl_types.h
│   │   │   │   │   ├──dataflash.c
│   │   │   │   │   ├──dataflash.h
│   │   │   │   │   ├──eel_descriptor_t02.c
│   │   │   │   │   ├──eel_descriptor_t02.h
│   │   │   │   │   ├──fdl_descriptor_t02.c
│   │   │   │   │   ├──fdl_descriptor_t02.h
│   │   │   │   │   └──iar_v2
│   │   │   │   │       ├──eel.a
│   │   │   │   │       ├──eel.h
│   │   │   │   │       ├──eel_types.h
│   │   │   │   │       ├──fdl.a
│   │   │   │   │       ├──fdl.h
│   │   │   │   │       └──fdl_types.h
│   │   │   │   ├──DTM2Wire
│   │   │   │   │   ├──DTM2Wire.c
│   │   │   │   │   └──DTM2Wire.h
│   │   │   │   ├──iic
│   │   │   │   │   ├──iic_slave.c
│   │   │   │   │   └──iic_slave.h
│   │   │   │   ├──led
│   │   │   │   │   ├──led.c
```

```
│   │   │   │   └──led.h
│   │   │   ├──led_onoff
│   │   │   │   ├──led_onoff.c
│   │   │   │   └──led_onoff.h
│   │   │   ├──peak
│   │   │   │   ├──peak.h
│   │   │   │   └──peak_isr.c
│   │   │   ├──pktmon
│   │   │   │   └──pktmon.h
│   │   │   ├──plf
│   │   │   │   ├──plf.c
│   │   │   │   └──plf.h
│   │   │   ├──port
│   │   │   │   └──port.h
│   │   │   ├──push_state
│   │   │   │   ├──push_state.c
│   │   │   │   └──push_state.h
│   │   │   ├──push_sw
│   │   │   │   ├──push_sw.c
│   │   │   │   └──push_sw.h
│   │   │   ├──rf
│   │   │   │   └──rf.h
│   │   │   ├──serial
│   │   │   │   └──serial.h
│   │   │   ├──uart
│   │   │   │   ├──uart.c
│   │   │   │   └──uart.h
│   │   │   └──wakeup
│   │   │       ├──wakeup.c
│   │   │       └──wakeup.h
│   │   └──types.h
│   └──tools
│       ├──project
│       │   ├──CS_CCRL
│       │   │   ├──BLE_Embedded
│       │   │   │   ├──BLE_Embedded.mtpj
│       │   │   │   ├──BLE_Embedded.rcpe
│       │   │   │   ├──DefaultBuild
│       │   │   │   └──rBLE_Emb
│       │   │   │       ├──DefaultBuild
│       │   │   │       │   ├──rBLE_Emb_CCRL.hex
│       │   │   │       │   ├──rBLE_Emb_CCRL.lmf
│       │   │   │       │   ├──rBLE_Emb_CCRL.map
│       │   │   │       │   └──rBLE_Emb_CCRL.sni
│       │   │   │       ├──rBLE_Emb.mtsp
│       │   │   │       ├──sect_emb.hsi
│       │   │   │       └──sect_emb_fwup.hsi
│       │   │   └──BLE_Modem
│       │   │       ├──BLE_Modem.mtpj
│       │   │       └──rBLE_Mdm
│       │   │           ├──rBLE_Mdm.mtsp
│       │   │           ├──sect_mdm.hsi
│       │   │           └──sect_mdm_fwup.hsi
│       │   ├──CubeSuite
│       │   │   ├──BLE_Embedded
│       │   │   │   ├──BLE_Emb
│       │   │   │   │   └──BLE_Emb.mtsp
│       │   │   │   └──BLE_Embedded.mtpj
│       │   │   └──BLE_Modem
│       │   │       ├──BLE_Modem.mtpj
│       │   │       └──rBLE_emb
│       │   │           └──rBLE_emb.mtsp
│       │   ├──e2studio
```

```
│   │   │   ├──BLE_Embedded
│   │   │   │   └──rBLE_Emb
│   │   │   │       ├──.cproject
│   │   │   │       ├──.DefaultBuildlinker
│   │   │   │       ├──.info
│   │   │   │       ├──.project
│   │   │   │       ├──rBLE_Emb_CCRL.x.launch
│   │   │   │       ├──sect_emb.esi
│   │   │   │       └──sect_emb_fwup.esi
│   │   │   └──BLE_Modem
│   │   │       └──rBLE_Mdm
│   │   │           ├──.cproject
│   │   │           ├──.DefaultBuildlinker
│   │   │           ├──.info
│   │   │           ├──.project
│   │   │           ├──rBLE_Mdm_CCRL.x.launch
│   │   │           ├──sect_mdm.esi
│   │   │           └──sect_mdm_fwup.esi
│   │   └──iar_v2
│   │       ├──BLE_Embedded
│   │       │   ├──BLE_Emb
│   │       │   │   └──BLE_Emb.ewp
│   │       │   └──BLE_Embedded.eww
│   │       ├──BLE_Modem
│   │       │   ├──BLE_Emb
│   │       │   │   └──BLE_Emb.ewp
│   │       │   └──BLE_Modem.eww
│   │       └──config
│   │           ├──lnkr5f11agj.icf
│   │           ├──lnkr5f11agj_fw.icf
│   │           └──lnkr5f11agj_fw_mdm.icf
│   ├──project_devices
│   │   ├──CS_CCRL
│   │   │   ├──BLE_Embedded
│   │   │   │   ├──BLE_Embedded_R5F11AGG.mtpj
│   │   │   │   ├──BLE_Embedded_R5F11AGH.mtpj
│   │   │   │   ├──rBLE_Emb_R5F11AGG
│   │   │   │   │   └──rBLE_Emb_R5F11AGG.mtsp
│   │   │   │   └──rBLE_Emb_R5F11AGH
│   │   │   │       └──rBLE_Emb_R5F11AGH.mtsp
│   │   │   └──BLE_Modem
│   │   │       ├──BLE_Modem_R5F11AGG.mtpj
│   │   │       ├──BLE_Modem_R5F11AGH.mtpj
│   │   │       ├──rBLE_Mdm_R5F11AGG
│   │   │       │   └──rBLE_Mdm_R5F11AGG.mtsp
│   │   │       └──rBLE_Mdm_R5F11AGH
│   │   │           └──rBLE_Mdm_R5F11AGH.mtsp
│   │   ├──CubeSuite
│   │   │   ├──BLE_Embedded
│   │   │   │   ├──BLE_Emb_R5F11AGH
│   │   │   │   │   └──BLE_Emb_R5F11AGH.mtsp
│   │   │   │   └──BLE_Embedded_R5F11AGH.mtpj
│   │   │   └──BLE_Modem
│   │   │       ├──BLE_Modem_R5F11AGH.mtpj
│   │   │       └──rBLE_emb_R5F11AGH
│   │   │           └──rBLE_emb_R5F11AGH.mtsp
│   │   ├──e2studio
│   │   │   ├──BLE_Embedded
│   │   │   │   ├──rBLE_Emb_R5F11AGG
│   │   │   │   │   ├──.cproject
│   │   │   │   │   ├──.DefaultBuildlinker
│   │   │   │   │   ├──.info
│   │   │   │   │   ├──.project
```

```
│   │   │   │   ├──rBLE_Emb_R5F11AGG_CCRL.x.launch
│   │   │   │   └──sect_emb.esi
│   │   │   └──rBLE_Emb_R5F11AGH
│   │   │       ├──.cproject
│   │   │       ├──.DefaultBuildlinker
│   │   │       ├──.info
│   │   │       ├──.project
│   │   │       ├──rBLE_Emb_R5F11AGH_CCRL.x.launch
│   │   │       └──sect_emb.esi
│   │   └──BLE_Modem
│   │       ├──rBLE_Mdm_R5F11AGG
│   │       │   ├──.cproject
│   │       │   ├──.DefaultBuildlinker
│   │       │   ├──.info
│   │       │   ├──.project
│   │       │   ├──rBLE_Mdm_R5F11AGG_CCRL.x.launch
│   │       │   └──sect_mdm.esi
│   │       └──rBLE_Mdm_R5F11AGH
│   │           ├──.cproject
│   │           ├──.DefaultBuildlinker
│   │           ├──.info
│   │           ├──.project
│   │           ├──rBLE_Mdm_R5F11AGH_CCRL.x.launch
│   │           └──sect_mdm.esi
│   └──iar_v2
│       ├──BLE_Embedded
│       │   ├──BLE_Emb_R5F11AGG
│       │   │   └──BLE_Emb_R5F11AGG.ewp
│       │   ├──BLE_Emb_R5F11AGH
│       │   │   └──BLE_Emb_R5F11AGH.ewp
│       │   ├──BLE_Embedded_R5F11AGG.eww
│       │   └──BLE_Embedded_R5F11AGH.eww
│       ├──BLE_Modem
│       │   ├──BLE_Emb_R5F11AGG
│       │   │   └──BLE_Emb_R5F11AGG.ewp
│       │   ├──BLE_Emb_R5F11AGH
│       │   │   └──BLE_Emb_R5F11AGH.ewp
│       │   ├──BLE_Modem_R5F11AGG.eww
│       │   └──BLE_Modem_R5F11AGH.eww
│       └──config
│           ├──lnkr5f11agg.icf
│           └──lnkr5f11agh.icf
└──project_simple
    ├──CS_CCRL
    │   └──BLE_Embedded
    │       ├──BLE_Embedded.mtpj
    │       └──rBLE_Emb
    │           ├──rBLE_Emb.mtsp
    │           └──sect_emb.hsi
    ├──CubeSuite
    │   └──BLE_Embedded
    │       ├──BLE_Emb
    │       │   └──BLE_Emb.mtsp
    │       └──BLE_Embedded.mtpj
    ├──e2studio
    │   └──BLE_Embedded
    │       └──rBLE_Emb
    │           ├──.cproject
    │           ├──.DefaultBuildlinker
    │           ├──.info
    │           ├──.project
    │           ├──rBLE_Emb_CCRL.x.launch
    │           └──sect_emb.esi
```

```
│                   └──iar_v2
│                       ├──BLE_Embedded
│                       │   ├──BLE_Emb
│                       │   │   └──BLE_Emb.ewp
│                       │   └──BLE_Embedded.eww
│                       └──config
│                           └──lnkr5f11agj.icf
└──workspace_RL78_G1D_peripheral
    └──Project_Source
        ├──bleip
        │   └──src
        │       ├──common
        │       │   └──co_bt.h
        │       └──rwble
        │           ├──rwble.h
        │           └──rwble_config.h
        ├──rBLE
        │   └──src
        │       ├──include
        │       │   ├──rble.h
        │       │   ├──rble_api.h
        │       │   ├──rble_api_custom.h
        │       │   ├──rble_api_fwup.h
        │       │   ├──rble_api_rrtp.h
        │       │   ├──rble_app.h
        │       │   ├──rble_external.h
        │       │   ├──rble_rwke.h
        │       │   ├──rble_trans.h
        │       │   └──rscip_api.h
        │       ├──sample_app
        │       │   ├──beacon_app.c
        │       │   ├──beacon_app.h
        │       │   ├──Console.c
        │       │   ├──console.h
        │       │   ├──menu_sel.c
        │       │   ├──menu_sel.h
        │       │   ├──r_rrtp.c
        │       │   ├──r_rrtp.h
        │       │   ├──rble_app_common.c
        │       │   ├──rble_app_common.h
        │       │   ├──rble_app_rrtp.c
        │       │   ├──rble_app_rrtp.h
        │       │   ├──rble_fw_up_receiver_app.c
        │       │   ├──rble_sample_app.c
        │       │   ├──rble_sample_app_anp.c
        │       │   ├──rble_sample_app_blp.c
        │       │   ├──rble_sample_app_cpp.c
        │       │   ├──rble_sample_app_cscp.c
        │       │   ├──rble_sample_app_custom.c
        │       │   ├──rble_sample_app_fmp.c
        │       │   ├──rble_sample_app_fwup.c
        │       │   ├──rble_sample_app_gap_sm_gatt.c
        │       │   ├──rble_sample_app_glp.c
        │       │   ├──rble_sample_app_hogp.c
        │       │   ├──rble_sample_app_hrp.c
        │       │   ├──rble_sample_app_htp.c
        │       │   ├──rble_sample_app_lnp.c
        │       │   ├──rble_sample_app_pasp.c
        │       │   ├──rble_sample_app_pxp.c
        │       │   ├──rble_sample_app_rscp.c
        │       │   ├──rble_sample_app_scpp.c
        │       │   ├──rble_sample_app_tip.c
        │       │   └──rble_sample_app_vendor.c
```

```
│       ├──sample_profile
│       │   ├──fwup
│       │   │   └──fwupr.c
│       │   ├──rrtp
│       │   │   ├──rtpc.c
│       │   │   └──rtps.c
│       │   └──scp
│       │       ├──scpc.c
│       │       └──scps.c
│       └──sample_simple
│           ├──console.c
│           ├──console.h
│           ├──rble_sample_app_peripheral.c
│           ├──rble_sample_app_peripheral.h
│           └──sam
│               ├──sam.h
│               ├──sams.c
│               └──sams.h
└──renesas
    ├──config
    │   └──split
    │       └──emb
    │           ├──r_lk.dr
    │           ├──r_lk_fw_emb.dr
    │           ├──r_lk_fw_modem.dr
    │           ├──r_lk_modem.dr
    │           ├──r_lk_modem_R5F11AGH.dr
    │           └──r_lk_R5F11AGH.dr
    ├──lib
    │   ├──BLE_CONTROLLER_lib.a
    │   ├──BLE_CONTROLLER_LIB.lib
    │   ├──BLE_CONTROLLER_LIB_CCRL.lib
    │   ├──BLE_HOST_lib.a
    │   ├──BLE_HOST_lib.lib
    │   ├──BLE_HOST_lib_CCRL.lib
    │   ├──BLE_PROFILE_ANP_lib.a
    │   ├──BLE_PROFILE_ANP_LIB.lib
    │   ├──BLE_PROFILE_ANP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_BLP_lib.a
    │   ├──BLE_PROFILE_BLP_LIB.lib
    │   ├──BLE_PROFILE_BLP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_CPP_lib.a
    │   ├──BLE_PROFILE_CPP_LIB.lib
    │   ├──BLE_PROFILE_CPP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_CSP_lib.a
    │   ├──BLE_PROFILE_CSP_LIB.lib
    │   ├──BLE_PROFILE_CSP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_FMP_lib.a
    │   ├──BLE_PROFILE_FMP_LIB.lib
    │   ├──BLE_PROFILE_FMP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_GLP_lib.a
    │   ├──BLE_PROFILE_GLP_LIB.lib
    │   ├──BLE_PROFILE_GLP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_HGP_lib.a
    │   ├──BLE_PROFILE_HGP_LIB.lib
    │   ├──BLE_PROFILE_HGP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_HRP_lib.a
    │   ├──BLE_PROFILE_HRP_LIB.lib
    │   ├──BLE_PROFILE_HRP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_HTP_lib.a
    │   ├──BLE_PROFILE_HTP_LIB.lib
    │   ├──BLE_PROFILE_HTP_LIB_CCRL.lib
    │   ├──BLE_PROFILE_LNP_lib.a
```

```
│   ├──BLE_PROFILE_LNP_LIB.lib
│   ├──BLE_PROFILE_LNP_LIB_CCRL.lib
│   ├──BLE_PROFILE_PAP_lib.a
│   ├──BLE_PROFILE_PAP_LIB.lib
│   ├──BLE_PROFILE_PAP_LIB_CCRL.lib
│   ├──BLE_PROFILE_PXP_lib.a
│   ├──BLE_PROFILE_PXP_LIB.lib
│   ├──BLE_PROFILE_PXP_LIB_CCRL.lib
│   ├──BLE_PROFILE_RSP_lib.a
│   ├──BLE_PROFILE_RSP_LIB.lib
│   ├──BLE_PROFILE_RSP_LIB_CCRL.lib
│   ├──BLE_PROFILE_SCP_lib.a
│   ├──BLE_PROFILE_SCP_LIB.lib
│   ├──BLE_PROFILE_SCP_LIB_CCRL.lib
│   ├──BLE_PROFILE_TIP_lib.a
│   ├──BLE_PROFILE_TIP_LIB.lib
│   ├──BLE_PROFILE_TIP_LIB_CCRL.lib
│   ├──BLE_PROFILES_COMMON_lib.a
│   ├──BLE_PROFILES_COMMON_LIB.lib
│   ├──BLE_PROFILES_COMMON_LIB_CCRL.lib
│   ├──BLE_rBLE_lib.a
│   ├──BLE_rBLE_lib.lib
│   └──BLE_rBLE_lib_CCRL.lib
├──src
│   ├──arch
│   │   └──rl78
│   │       ├──arch.h
│   │       ├──arch_main.c
│   │       ├──config.h
│   │       ├──db_handle.h
│   │       ├──fw_update_count0.c
│   │       ├──fw_update_count1.c
│   │       ├──hw_config.h
│   │       ├──ke_conf.c
│   │       ├──ke_conf_simple.c
│   │       ├──ll
│   │       │   └──ll.h
│   │       ├──main.c
│   │       ├──prf_config.c
│   │       ├──prf_config.h
│   │       ├──prf_config_host.c
│   │       ├──prf_sel.h
│   │       ├──rble_core_config.c
│   │       ├──rble_core_config.h
│   │       ├──rble_modem_config.c
│   │       ├──rble_modem_config.h
│   │       ├──rwble_mem.c
│   │       ├──rwble_mem.h
│   │       └──rwke_api.h
│   ├──compiler
│   │   ├──ccrl
│   │   │   ├──cstart.asm
│   │   │   ├──hdwinit.asm
│   │   │   └──stkinit.asm
│   │   ├──compiler.h
│   │   └──iodefine.h
│   ├──driver
│   │   ├──codeflash
│   │   │   ├──cc_rl
│   │   │   │   ├──fsl.h
│   │   │   │   ├──fsl.lib
│   │   │   │   └──fsl_types.h
│   │   │   ├──codeflash.c
```

```
│   │   │   ├──codeflash.h
│   │   │   ├──cs
│   │   │   │   ├──fsl.h
│   │   │   │   ├──fsl.lib
│   │   │   │   └──fsl_types.h
│   │   │   └──iar_v2
│   │   │       ├──fsl.a
│   │   │       ├──fsl.h
│   │   │       └──fsl_types.h
│   │   ├──csi
│   │   │   ├──csi.c
│   │   │   └──csi.h
│   │   ├──dataflash
│   │   │   ├──cc_rl
│   │   │   │   ├──eel.h
│   │   │   │   ├──eel.lib
│   │   │   │   ├──eel_types.h
│   │   │   │   ├──fdl.h
│   │   │   │   ├──fdl.lib
│   │   │   │   └──fdl_types.h
│   │   │   ├──cs
│   │   │   │   ├──eel.h
│   │   │   │   ├──eel.lib
│   │   │   │   ├──eel_types.h
│   │   │   │   ├──fdl.h
│   │   │   │   ├──fdl.lib
│   │   │   │   └──fdl_types.h
│   │   │   ├──dataflash.c
│   │   │   ├──dataflash.h
│   │   │   ├──eel_descriptor_t02.c
│   │   │   ├──eel_descriptor_t02.h
│   │   │   ├──fdl_descriptor_t02.c
│   │   │   ├──fdl_descriptor_t02.h
│   │   │   └──iar_v2
│   │   │       ├──eel.a
│   │   │       ├──eel.h
│   │   │       ├──eel_types.h
│   │   │       ├──fdl.a
│   │   │       ├──fdl.h
│   │   │       └──fdl_types.h
│   │   ├──DTM2Wire
│   │   │   ├──DTM2Wire.c
│   │   │   └──DTM2Wire.h
│   │   ├──iic
│   │   │   ├──iic_slave.c
│   │   │   └──iic_slave.h
│   │   ├──led
│   │   │   ├──led.c
│   │   │   └──led.h
│   │   ├──led_onoff
│   │   │   ├──led_onoff.c
│   │   │   └──led_onoff.h
│   │   ├──peak
│   │   │   ├──peak.h
│   │   │   └──peak_isr.c
│   │   ├──pktmon
│   │   │   └──pktmon.h
│   │   ├──plf
│   │   │   ├──plf.c
│   │   │   └──plf.h
│   │   ├──port
│   │   │   └──port.h
│   │   ├──push_state
```

```
│   │   │   ├──push_state.c
│   │   │   └──push_state.h
│   │   ├──push_sw
│   │   │   ├──push_sw.c
│   │   │   └──push_sw.h
│   │   ├──rf
│   │   │   └──rf.h
│   │   ├──serial
│   │   │   └──serial.h
│   │   ├──uart
│   │   │   ├──uart.c
│   │   │   └──uart.h
│   │   └──wakeup
│   │       ├──wakeup.c
│   │       └──wakeup.h
│   └──types.h
└──tools
    ├──project
    │   ├──CS_CCRL
    │   │   ├──BLE_Embedded
    │   │   │   ├──BLE_Embedded.mtpj
    │   │   │   ├──BLE_Embedded.rcpe
    │   │   │   └──rBLE_Emb
    │   │   │       ├──rBLE_Emb.mtsp
    │   │   │       ├──sect_emb.hsi
    │   │   │       └──sect_emb_fwup.hsi
    │   │   └──BLE_Modem
    │   │       ├──BLE_Modem.mtpj
    │   │       └──rBLE_Mdm
    │   │           ├──rBLE_Mdm.mtsp
    │   │           ├──sect_mdm.hsi
    │   │           └──sect_mdm_fwup.hsi
    │   ├──CubeSuite
    │   │   ├──BLE_Embedded
    │   │   │   ├──BLE_Emb
    │   │   │   │   └──BLE_Emb.mtsp
    │   │   │   └──BLE_Embedded.mtpj
    │   │   └──BLE_Modem
    │   │       ├──BLE_Modem.mtpj
    │   │       └──rBLE_emb
    │   │           └──rBLE_emb.mtsp
    │   ├──e2studio
    │   │   ├──BLE_Embedded
    │   │   │   └──rBLE_Emb
    │   │   │       ├──.cproject
    │   │   │       ├──.DefaultBuildlinker
    │   │   │       ├──.info
    │   │   │       ├──.project
    │   │   │       ├──rBLE_Emb_CCRL.x.launch
    │   │   │       ├──sect_emb.esi
    │   │   │       └──sect_emb_fwup.esi
    │   │   └──BLE_Modem
    │   │       └──rBLE_Mdm
    │   │           ├──.cproject
    │   │           ├──.DefaultBuildlinker
    │   │           ├──.info
    │   │           ├──.project
    │   │           ├──rBLE_Mdm_CCRL.x.launch
    │   │           ├──sect_mdm.esi
    │   │           └──sect_mdm_fwup.esi
    │   └──iar_v2
    │       ├──BLE_Embedded
    │       │   ├──BLE_Emb
```

```
│   │   │   └──BLE_Emb.ewp
│   │   └──BLE_Embedded.eww
│   ├──BLE_Modem
│   │   ├──BLE_Emb
│   │   │   └──BLE_Emb.ewp
│   │   └──BLE_Modem.eww
│   └──config
│       ├──lnkr5f11agj.icf
│       ├──lnkr5f11agj_fw.icf
│       └──lnkr5f11agj_fw_mdm.icf
├──project_devices
│   ├──CS_CCRL
│   │   ├──BLE_Embedded
│   │   │   ├──BLE_Embedded_R5F11AGG.mtpj
│   │   │   ├──BLE_Embedded_R5F11AGH.mtpj
│   │   │   ├──rBLE_Emb_R5F11AGG
│   │   │   │   └──rBLE_Emb_R5F11AGG.mtsp
│   │   │   └──rBLE_Emb_R5F11AGH
│   │   │       └──rBLE_Emb_R5F11AGH.mtsp
│   │   └──BLE_Modem
│   │       ├──BLE_Modem_R5F11AGG.mtpj
│   │       ├──BLE_Modem_R5F11AGH.mtpj
│   │       ├──rBLE_Mdm_R5F11AGG
│   │       │   └──rBLE_Mdm_R5F11AGG.mtsp
│   │       └──rBLE_Mdm_R5F11AGH
│   │           └──rBLE_Mdm_R5F11AGH.mtsp
│   ├──CubeSuite
│   │   ├──BLE_Embedded
│   │   │   ├──BLE_Emb_R5F11AGH
│   │   │   │   └──BLE_Emb_R5F11AGH.mtsp
│   │   │   └──BLE_Embedded_R5F11AGH.mtpj
│   │   └──BLE_Modem
│   │       ├──BLE_Modem_R5F11AGH.mtpj
│   │       └──rBLE_emb_R5F11AGH
│   │           └──rBLE_emb_R5F11AGH.mtsp
│   ├──e2studio
│   │   ├──BLE_Embedded
│   │   │   ├──rBLE_Emb_R5F11AGG
│   │   │   │   ├──.cproject
│   │   │   │   ├──.DefaultBuildlinker
│   │   │   │   ├──.info
│   │   │   │   ├──.project
│   │   │   │   ├──rBLE_Emb_R5F11AGG_CCRL.x.launch
│   │   │   │   └──sect_emb.esi
│   │   │   └──rBLE_Emb_R5F11AGH
│   │   │       ├──.cproject
│   │   │       ├──.DefaultBuildlinker
│   │   │       ├──.info
│   │   │       ├──.project
│   │   │       ├──rBLE_Emb_R5F11AGH_CCRL.x.launch
│   │   │       └──sect_emb.esi
│   │   └──BLE_Modem
│   │       ├──rBLE_Mdm_R5F11AGG
│   │       │   ├──.cproject
│   │       │   ├──.DefaultBuildlinker
│   │       │   ├──.info
│   │       │   ├──.project
│   │       │   ├──rBLE_Mdm_R5F11AGG_CCRL.x.launch
│   │       │   └──sect_mdm.esi
│   │       └──rBLE_Mdm_R5F11AGH
│   │           ├──.cproject
│   │           ├──.DefaultBuildlinker
│   │           ├──.info
```

```
│   │       ├──.project
│   │       ├──rBLE_Mdm_R5F11AGH_CCRL.x.launch
│   │       └──sect_mdm.esi
│   └──iar_v2
│       ├──BLE_Embedded
│       │   ├──BLE_Emb_R5F11AGG
│       │   │   └──BLE_Emb_R5F11AGG.ewp
│       │   ├──BLE_Emb_R5F11AGH
│       │   │   └──BLE_Emb_R5F11AGH.ewp
│       │   ├──BLE_Embedded_R5F11AGG.eww
│       │   └──BLE_Embedded_R5F11AGH.eww
│       ├──BLE_Modem
│       │   ├──BLE_Emb_R5F11AGG
│       │   │   └──BLE_Emb_R5F11AGG.ewp
│       │   ├──BLE_Emb_R5F11AGH
│       │   │   └──BLE_Emb_R5F11AGH.ewp
│       │   ├──BLE_Modem_R5F11AGG.eww
│       │   └──BLE_Modem_R5F11AGH.eww
│       └──config
│           ├──lnkr5f11agg.icf
│           └──lnkr5f11agh.icf
└──project_simple
    ├──CS_CCRL
    │   └──BLE_Embedded
    │       ├──BLE_Embedded.mtpj
    │       └──rBLE_Emb
    │           ├──rBLE_Emb.mtsp
    │           └──sect_emb.hsi
    ├──CubeSuite
    │   └──BLE_Embedded
    │       ├──BLE_Emb
    │       │   └──BLE_Emb.mtsp
    │       └──BLE_Embedded.mtpj
    ├──e2studio
    │   └──BLE_Embedded
    │       └──rBLE_Emb
    │           ├──.cproject
    │           ├──.DefaultBuildlinker
    │           ├──.info
    │           ├──.project
    │           ├──rBLE_Emb_CCRL.x.launch
    │           └──sect_emb.esi
    └──iar_v2
        ├──BLE_Embedded
        │   ├──BLE_Emb
        │   │   └──BLE_Emb.ewp
        │   └──BLE_Embedded.eww
        └──config
            └──lnkr5f11agj.icf
```

2.　RRTP（ルネサス高速転送プロファイル）

2.1　RRTP の概要

RRTP は RL78/G1D 用 BLE ソフトウェア付属のカスタムプロファイルを基に開発した高速転送プロファイル(Renesas Rapid Transfer Profile)です。

この RRTP では、以下の方法で単位時間当たりのデータ量を増やします。

— Central 機器が Connection Interval 値を小さい値（規格上の最小値：7.5 ミリ秒）で接続
— 通信中に Peripheral（サーバ）機器がデータチャンネルのヘッダに含まれる MD (More Data)ビット制御により Connection Interval 毎に最大 4 個の Notification [注]パケットを送信

注：Characterstic Value Notification の手順を示します。

上記の制御により、ユーザデータ 20 バイトの場合、以下のスループットを達成できます。

表 2-1　RRTP によるユーザデータのスループット（設計値）

1Connection Interval 中のパケット数	ユーザデータのスループット（設計値）
4	（20 バイト×4 パケット）／7.5 ミリ秒 = 10,666 バイト／秒 = 85.33 kbps
3	（20 バイト×3 パケット）／7.5 ミリ秒 = 8,000 バイト／秒 = 64 kbps
2	（20 バイト×2 パケット）／7.5 ミリ秒 = 5,333 バイト／秒 = 42.67 kbps
1	（20 バイト×1 パケット）／7.5 ミリ秒 = 2,666 バイト／秒 = 21.33 kbps

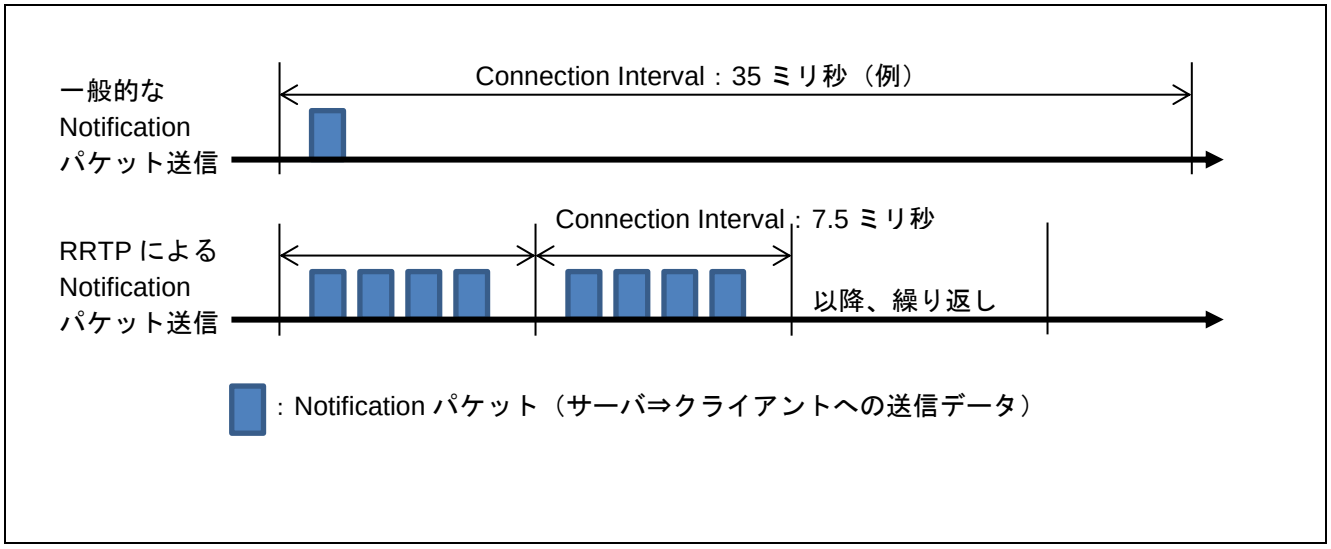


図 2-1　RRTP の Notifiction パケットによる通知（サーバからのデータ送信）

2.1.1　Connection Interval 設定

接続が完了し通信中状態になると、データチャネルでは、Central 機器のタイミングにより一定間隔でデータを送受信します。このタイミングを Connection Event といいます。また、間隔のことを Connection Interval といいます。Bluetooth low energy の規格上の最小値は 7.5 ミリ秒です。

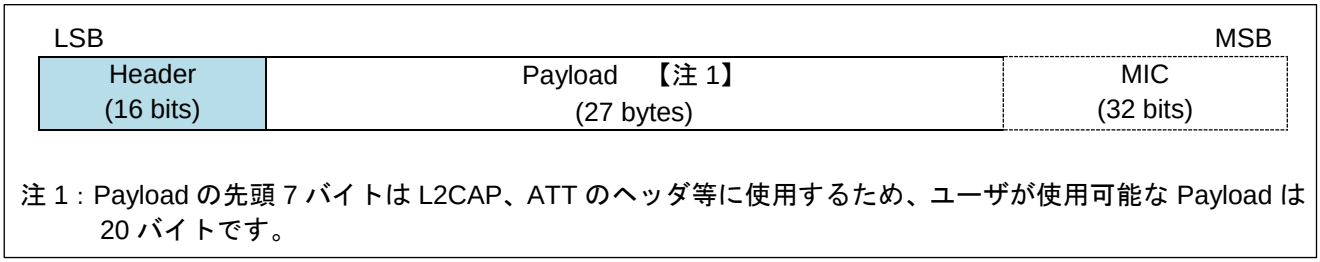
RRTP では、スループット向上のため、Connection Interval を短く設定しています。

2.1.2　MD ビット制御による複数 Characterstic Value Notification（特性値通知）パケット送信

Data Channel PDU (Protocol Data Unit)の Header の中に、MD ビットがあります。

MD は接続先に送信すべきデータがあることを示すビットで、MD=0 の場合、続きのデータが無いこと、MD=1 の場合、続きのデータが有ることを示します。そのため、MD=1 であれば、その Connection Event が延長されることによって通信が継続します。

1 Connection Interval 中に複数のユーザデータを BLE ソフトウェアに引き渡すと、BLE ソフトウェアが MD ビットを制御し、1 Connection Interval 中に複数の Data Channel PDU を送信します。



注 1：Payload の先頭 7 バイトは L2CAP、ATT のヘッダ等に使用するため、ユーザが使用可能な Payload は 20 バイトです。

図 2-2 Data Channel PDU の構造（Bluetooth Ver.4.1 の場合）

Header						
LLID (2 bits)	NESN (1 bit)	SN (1 bit)	**MD** (1 bit)	RFU (3 bits)	Length (5 bits)	RFU (3 bits)

図 2-3 PDU header の構造（Bluetooth Ver.4.1 の場合）

また、サーバからクライアントにデータを送信する手順(procedure)として、以下があります。

- Characterstic Value Read
  クライアントが定期的にサーバに読み出しを要求し、その応答としてサーバからデータを送信します。

- Characterstic Value Notification（以下 Notification と略す）
  測定値等が変更された場合に、クライアントがその通知を受け取る BLE 特有の仕組みで、サーバのタイミングでクライアントにデータを送信（データ変更を通知）します。

- Characterstic Value Indication
  サーバからクライアントにデータを送信します。ただし、クライアントからサーバへの返答が必要です。

RRTP では、通信中の Notification により、サーバ⇔クライアント間の通信を最小限にしています。そのため、Peripheral（サーバ）機器では、RF 受信処理が不要になり、RF 送信処理のみを行います。

2.2　RRTP の仕様

高速転送制御するための独自の GATT ベースプロファイルを定義しています。

2.2.1　プロファイル仕様

サンプルプログラムの高速転送プロファイルの仕様を示します。

(1)　ロール

- センサが接続された Peripheral 機器の RL78/G1D を RRTP のサーバとします
  サーバは、Renesas Rapid Transfer Service (RRTS)を保持します。

- RRTP サーバと接続してセンサを制御する Central 機器を RRTP のクライアントとします
  クライアントは、サーバの RRTS にアクセスします。

  本アプリケーションノートでは RL78/G1D 評価ボードがクライアントです。

(2)　サービスと Characteristic（特性）

- RRTS は、データ送信を制御するための Characteristic で構成されています
- サーバは Notification によって Characteristic Value（特性値）をクライアントに通知します

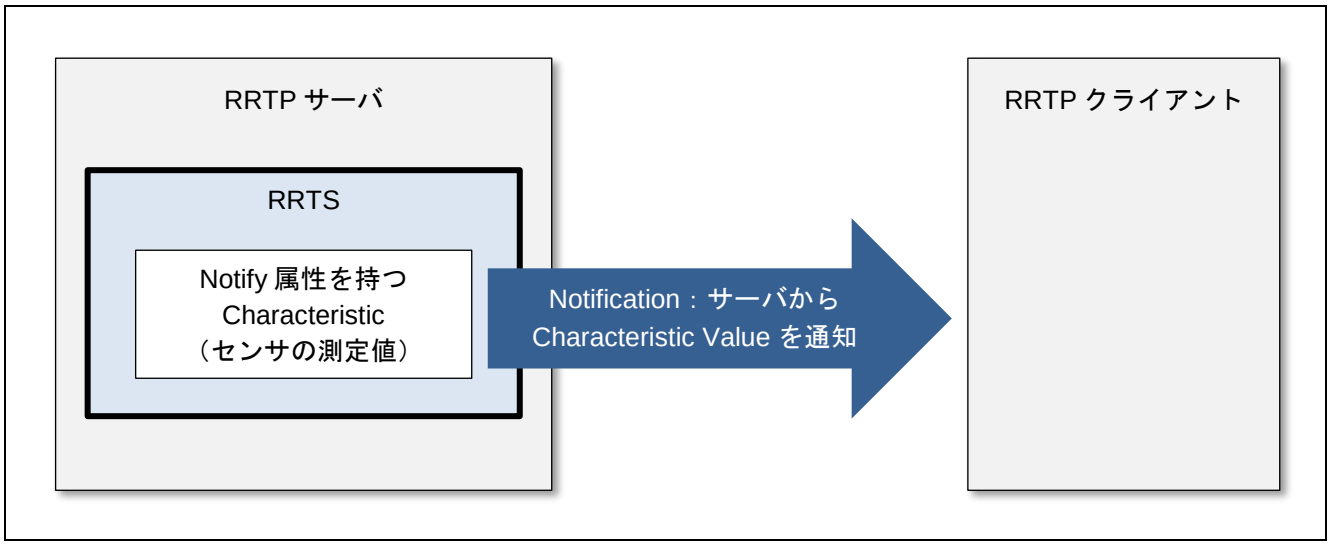


図 2-4 RRTP

2.2.2　サービス仕様

表 2-2 にサンプルプログラムの高速転送プロファイルのサービス仕様を示します。

表 2-2　高速転送プロファイルサービス仕様

Attribute Handle	Attribute Type	Attribute Value
Renesas Rapid Transfer Service		
0x000C	Primary Service Declaration (0x2800)	UUID: E01B0001-BA28-466A-88FA-337721DC020B
Rapid Transfer Data Value Characteristic (Notify Property)		
0x000D	Characteristic Declaration (0x2803)	Properties: Notify (0x10) Value Handle: 0x000E UUID: E01B0002-BA28-466A-88FA-337721DC020B
0x000E	Characteristic Value (0xFFFF)	Transfer data (20-byte)　【注 1】
0x000F	Characteristic Declaration (0x2803)	Properties: Notify (0x10) Value Handle: 0x0010 UUID: E01B0002-BA28-466A-88FA-337721DC020B
0x0010	Characteristic Value (0xFFFF)	Transfer data (20-byte)　【注 1】
0x0011	Characteristic Declaration (0x2803)	Properties: Notify (0x10) Value Handle: 0x0012 UUID: E01B0002-BA28-466A-88FA-337721DC020B
0x0012	Characteristic Value (0xFFFF)	Transfer data (20-byte)　【注 1】
0x0013	Characteristic Declaration (0x2803)	Properties: Notify (0x10) Value Handle: 0x0014 UUID: E01B0002-BA28-466A-88FA-337721DC020B
0x0014	Characteristic Value (0xFFFF)	Transfer data (20-byte)　【注 1】
0x0015	Client Characteristic Configuration (0x2902)	Properties: Read, Write (0x0A) Notification Configuration (2-byte)　【注 2】

注 1：RRTP を使って転送するユーザデータを示します。

注 2：0x0001 (= RBLE_RRTP_START_NTF)であり、Notification 通知開始を示します。

2.3　ベースのソフトウェア

Central 機器、Peripheral 機器共に同じ BLE ソフトウェアをベースとして使用します。

BLE ソフトウェアを以下から、ダウンロードしてください。

- Bluetooth low energy Protocol Stack (Ver.1.21)　RTM5F11A00NBLE0F10RZ-V0121.zip
- https://www.renesas.com/jp/ja/products/software-tools/software-os-middleware-driver/protocol-stack/ble-protocol-stack.html#downloads

2.3.1　プロジェクト階層

使用するプロジェクトを図 2-5 に示します。

```
RTM5F11A00NBLE0F10RZ-V0121　（ZIP ファイル）
    └SoftwarePackage
        └BLE_Software_Ver_1_21　（ZIP ファイル）
            └Renesas
                └BLE_Software_Ver_1_21
                    └RL78_G1D
                        └Project_Source　（WORKSPACE Folder）
```

図 2-5　実行環境プロジェクト階層（v1.21）

2.4　Peripheral 機器

Peripheral 機器は、電源投入後、自動的にアドバタイズを開始します。また、Central 機器との接続を確認すると、1 パケット 20 バイトのデータの送信を開始します。

2.4.1　変更差分対象ファイル一覧

Peripheral 機器の BLE ソフトウェアについて、ベースソフトウェアからの変更差分を表 2-3 に示します。

表 2-3 変更差分ファイル一覧

階層	ファイル名	処置	備考
rBLE/src/include	rble_api_rrtp.h	追加	RRTP で使用する定数や構造体の定義
rBLE/src/include	rble_external.h	追加	初期化時に使用する構造体の定義
rBLE/src/sample_app	r_rrtp.c	追加	送信キューに関する関数
rBLE/src/sample_app	r_rrtp.h	追加	r_rrtp.c に関連するヘッダファイル
rBLE/src/sample_app	rble_app_common.c	追加	rble 初期化等の共通関数
rBLE/src/sample_app	rble_app_common.h	追加	rble_app_common.c のヘッダファイル
rBLE/src/sample_app	rble_app_rrtp.c	追加	RRTP の API とコールバック関数
rBLE/src/sample_app	rble_app_rrtp.h	追加	rble_app_rrtp.c のヘッダファイル
rBLE/src/sample_profile/rrtp	rtps.c	追加	RRTP クライアントのサービス処理関数群
renesas/src/arch/rl78	arch_main.c	変更	
renesas/src/arch/rl78	db_handle.h	変更	
renesas/src/arch/rl78	main.c	変更	
renesas/src/arch/rl78	prf_config.c	変更	
renesas/src/arch/rl78	prf_config.h	変更	
renesas/src/arch/rl78	prf_sel.h	変更	

2.4.2　変更差分詳細

BLE ソフトウェアからの差分情報について、以下に示します。変更指定行は変更後の行位置を主に示します。行頭の「－」は行の削除、「＋」は行の追加を表します。

(1)　arch_main.c

(a)　59 行目に追加（サンプルコードの 59～62 行目）

組み込むヘッダファイルを追加します。

```
+#include "rble_external.h"
+#include "rble_app_common.h"
+#include "rble_app_rrtp.h"
+
```

(b)　68 行目を変更（サンプルコードの 72～88 行目）

自動接続を行うための定義の追加と変数の初期化を行います。

```
-
+#include "r_rrtp.h"
+
+typedef enum
+{
+       E_RF_STATE_UNINITIALIZED = 0,
```

```
+       E_RF_STATE_REQ_INITIALIZE,
+       E_RF_STATE_WAIT_INITIALIZE,
+       E_RF_STATE_REQ_RF_ENABLE,
+       E_RF_STATE_WAIT_RF_ENABLE,
+       E_RF_STATE_REQ_ADVERTISE,
+       E_RF_STATE_WAIT_ADVERTISE,
+       E_RF_STATE_WAIT_CONNECTION,
+       E_RF_STATE_WAIT_SERVER_ENABLE,
+       E_RF_STATE_CONNECTED,
+} e_rf_state_t;
+
+e_rf_state_t g_rf_state = E_RF_STATE_UNINITIALIZED;
```

(c)　77 行目に追加（サンプルコードの 97～99 行目）

自動接続およびデータ転送用関数のプロトタイプ宣言を追加します。

```
+void rble_rf_control(void);
+void rrtp_data_send(void);
+
```

(d)　317 行目に追加（サンプルコードの 340,341 行目）

自動接続用変数の初期化および BLE ソフトウェアの初期化を行います。

```
+    g_rble_rf_status = RBLE_RF_UNINITIALIZED;
+    RBLE_App_Init();
```

(e)　388 行目に追加（サンプルコードの 413～420 行目）

自動接続シーケンス処理の呼び出しとデータ転送用処理をメインルーチンに追加します。

```
+       rble_rf_control();
+
+       if (E_RF_STATE_CONNECTED == g_rf_state)
+       {
+           rrtp_data_send ();
+           RBLE_App_RapidTransferHandler();
+       }
+
```

(f)　451 行目を変更（サンプルコードの 484 行目）

スリープを無効化するため判定関数の戻り値を変更します。

```
-    return true;
+    return false;
```

(g)　483 行目に追加（サンプルコードの 516～665 行目）

自動接続およびデータ転送用関数の実体を追加します。

```
+void rrtp_data_send(void)
+{
+    uint8_t data_array[D_RAPID_TRANSFER_BUFFER_SIZE];
+    uint8_t data_len = D_RAPID_TRANSFER_BUFFER_SIZE;
+
+    static uint8_t serial_num = 0;
+
+    data_array[0] = serial_num;
+
+    if (TRUE == R_RRTP_DataEnqueue(data_array, data_len))
+    {
+        serial_num++;
+    }
+    else
+    {
+        /* Do nothing */
```

```
+        __nop();
+    }
+}
+
+void rble_rf_control(void)
+{
+    uint8_t           temp;
+    st_init_api_info_t init_api_info;
+    const uint8_t     gs_device_name[] = "RSSK-RRTP-TEST";
+
+       switch(g_rf_state)
+       {
+           case E_RF_STATE_UNINITIALIZED:
+               {
+                   g_rf_state++;
+               }
+               break;
+           case E_RF_STATE_REQ_INITIALIZE:
+               {
+                   for (temp = 0; temp < RF_DEVICE_NAME_SZ; temp++)
+                   {
+                       if (temp < sizeof(gs_device_name))
+                       {
+                           init_api_info.device_name[temp] =
gs_device_name[temp];
+                       }
+                       else
+                       {
+                           init_api_info.device_name[temp] = 0x00;
+                       }
+                   }
+
+                   init_api_info.service_list.list_num    = 0x05;
+                   init_api_info.service_list.service_list = 0x0010;
+                   init_api_info.iocap                    =
RBLE_IO_CAP_NO_INPUT_NO_OUTPUT;
+                   init_api_info.auth                     =
RBLE_AUTH_REQ_NO_MITM_BOND;
+                   init_api_info.op.context               = 0x01;
+
+                   RBLE_App_InitializeAPIRequested(&init_api_info);
+
+                   g_rf_state++;
+               }
+               break;
+           case E_RF_STATE_WAIT_INITIALIZE:
+               {
+                   if (RBLE_RF_INITIALIZED == g_rble_rf_status)
+                   {
+                       g_rf_state++;
+                   }
+                   else
+                   {
+                       /* Do nothing */
+                       __nop();
+                   }
+               }
+               break;
+           case E_RF_STATE_REQ_RF_ENABLE:
```

```
+                {
+                    RBLE_App_EnableRfRequested(FALSE);
+                    g_rf_state++;
+                }
+                break;
+            case E_RF_STATE_WAIT_RF_ENABLE:
+                {
+                    if (RBLE_RF_ENABLED == g_rble_rf_status)
+                    {
+                        g_rf_state++;
+                    }
+                    else
+                    {
+                        /* Do nothing */
+                        __nop();
+                    }
+                }
+                break;
+            case E_RF_STATE_REQ_ADVERTISE:
+                {
+                    RBLE_App_StartAdvertiseRequested();
+                    g_rf_state++;
+                }
+                break;
+            case E_RF_STATE_WAIT_ADVERTISE:
+                {
+                    if (RBLE_RF_ADVERTISING == g_rble_rf_status)
+                    {
+                        g_rf_state++;
+                    }
+                    else
+                    {
+                        /* Do nothing */
+                        __nop();
+                    }
+                }
+                break;
+            case E_RF_STATE_WAIT_CONNECTION:
+                {
+                    if (RBLE_RF_CONNECTED == g_rble_rf_status)
+                    {
+                        g_rf_state++;
+                    }
+                    else
+                    {
+                        /* Do nothing */
+                        __nop();
+                    }
+                }
+                break;
+            case E_RF_STATE_WAIT_SERVER_ENABLE:
+                {
+                    if (RBLE_RF_SERVER_ENABLED == g_rble_rf_status)
+                    {
+                        g_rf_state++;
+                        g_rrtp_moredata = 3U;
+                    }
+                    else
+                    {
```

```
+                    /* Do nothing */
+                    __nop();
+                }
+            }
+            break;
+        case E_RF_STATE_CONNECTED:
+            {
+                /* Do nothing */
+                __nop();
+            }
+            break;
+        default:
+            {
+                /* Do nothing */
+                __nop();
+            }
+            break;
+    }
+}
```

(2) db_handle.h

(a) 413 行目に追加（サンプルコードの 413～429 行目）

高速転送プロファイルのハンドル定義を追加します。

```
+    #if (PRF_SEL_RTPS)
+    /* Renesas Rapid Transfer Service */
+    RRTS_HDL_SVC,
+    RRTS_HDL_NOTIFY_CHAR1,
+    RRTS_HDL_NOTIFY_VAL1,
+    RRTS_HDL_NOTIFY_CHAR2,
+    RRTS_HDL_NOTIFY_VAL2,
+    RRTS_HDL_NOTIFY_CHAR3,
+    RRTS_HDL_NOTIFY_VAL3,
+    RRTS_HDL_NOTIFY_CHAR4,
+    RRTS_HDL_NOTIFY_VAL4,
+    RRTS_HDL_NOTIFY_CFG,
+    RRTS_HDL_IND_CHAR,
+    RRTS_HDL_IND_VAL,
+    RRTS_HDL_IND_CFG,
+    #endif /* #if (PRF_SEL_RTPS) */
+
```

(3) main.c

(a) 101 行目に追加（サンプルコードの 101,102 行目）

More Data 制御用の変数の参照設定を行います。

```
+extern uint16_t more_data_count;
+
```

(b) 476 行目に追加（サンプルコードの 478 行目）

peak 割り込みで More Data 制御用変数をリセットします。

```
+    more_data_count = 0;
```

(4) prf_config.c

(a) 1238 行目に追加（サンプルコードの 1238～1294 行目）

高速転送プロファイルのサービスデータを定義します。

```
+#if (PRF_SEL_RTPS)
+/*********************************
+ * Renesas Rapid Transfer Service *
+ *********************************/
+/* Service (rtps) */
+static const uint8_t rtps_svc[RBLE_GATT_128BIT_UUID_OCTET] =
RBLE_SVC_RAPID_TRANSFER;
+
+/* Renesas Rapid Transfer Service Notify characteristic 1 */
+static const struct atts_char128_desc rtps_notify_char1 =
{ RBLE_GATT_CHAR_PROP_NTF,
+
{(uint8_t)(RRTS_HDL_NOTIFY_VAL1 & 0xff), (uint8_t)((RRTS_HDL_NOTIFY_VAL1 >> 8)
& 0xff)},
+                                                    RBLE_CHAR_RRTP_NOTIFY };
+uint8_t rtps_notify_char_val1[RBLE_ATTM_MAX_VALUE] = {0};
+struct atts_elmt_128 rtps_notify_char_val_elmt1 = { RBLE_CHAR_RRTP_NOTIFY,
+                                                    RBLE_GATT_128BIT_UUID_OCTET,
+                                                    &rtps_notify_char_val1[0] };
+
+/* Renesas Rapid Transfer Service Notify characteristic 2 */
+static const struct atts_char128_desc rtps_notify_char2 =
{ RBLE_GATT_CHAR_PROP_NTF,
+
{(uint8_t)(RRTS_HDL_NOTIFY_VAL2 & 0xff), (uint8_t)((RRTS_HDL_NOTIFY_VAL2 >> 8)
& 0xff)},
+                                                    RBLE_CHAR_RRTP_NOTIFY };
+uint8_t rtps_notify_char_val2[RBLE_ATTM_MAX_VALUE] = {0};
+struct atts_elmt_128 rtps_notify_char_val_elmt2 = { RBLE_CHAR_RRTP_NOTIFY,
+                                                    RBLE_GATT_128BIT_UUID_OCTET,
+                                                    &rtps_notify_char_val2[0] };
+
+/* Renesas Rapid Transfer Service Notify characteristic 3 */
+static const struct atts_char128_desc rtps_notify_char3 =
{ RBLE_GATT_CHAR_PROP_NTF,
+
{(uint8_t)(RRTS_HDL_NOTIFY_VAL3 & 0xff), (uint8_t)((RRTS_HDL_NOTIFY_VAL3 >> 8)
& 0xff)},
+                                                    RBLE_CHAR_RRTP_NOTIFY };
+uint8_t rtps_notify_char_val3[RBLE_ATTM_MAX_VALUE] = {0};
+struct atts_elmt_128 rtps_notify_char_val_elmt3 = { RBLE_CHAR_RRTP_NOTIFY,
+                                                    RBLE_GATT_128BIT_UUID_OCTET,
+                                                    &rtps_notify_char_val3[0] };
+
+/* Renesas Rapid Transfer Service Notify characteristic 4 */
+static const struct atts_char128_desc rtps_notify_char4 =
{ RBLE_GATT_CHAR_PROP_NTF,
+
{(uint8_t)(RRTS_HDL_NOTIFY_VAL4 & 0xff), (uint8_t)((RRTS_HDL_NOTIFY_VAL4 >> 8)
& 0xff)},
+                                                    RBLE_CHAR_RRTP_NOTIFY };
+uint8_t rtps_notify_char_val4[RBLE_ATTM_MAX_VALUE] = {0};
+struct atts_elmt_128 rtps_notify_char_val_elmt4 = { RBLE_CHAR_RRTP_NOTIFY,
+                                                    RBLE_GATT_128BIT_UUID_OCTET,
+                                                    &rtps_notify_char_val4[0] };
+
+uint16_t rtps_notify_en = 0x0000u;
+
+/* Renesas Rapid Transfer Service Indicate characteristic */
```

```
+static const struct atts_char128_desc rtps_ind_char =
{ RBLE_GATT_CHAR_PROP_IND,
+                                                {(uint8_t)(RRTS_HDL_IND_VAL &
0xff), (uint8_t)((RRTS_HDL_IND_VAL >> 8) & 0xff)},
+                                                RBLE_CHAR_RRTP_INDICATE };
+uint8_t rtps_ind_char_val[RBLE_ATTM_MAX_VALUE] = {0};
+struct atts_elmt_128 rtps_ind_char_val_elmt = { RBLE_CHAR_RRTP_INDICATE,
+                                        RBLE_GATT_128BIT_UUID_OCTET,
+                                        &rtps_ind_char_val[0] };
+
+uint16_t rtps_ind_en = 0x0000u;
+#endif /* (PRF_SEL_RTPS) */
```

(b)　2135 行目に追加（サンプルコードの 2193～2236 行目）

高速転送プロファイルのサービスデータをデータベースに登録します。

```
+#if (PRF_SEL_RTPS)
+    /**********************************
+     * Renesas Rapid Transfer Service *
+     **********************************/
+    { RBLE_DECL_PRIMARY_SERVICE,
+       sizeof(rtps_svc), sizeof(rtps_svc), TASK_ATTID(TASK_RBLE,
RRTS_IDX_SVC), RBLE_GATT_PERM_RD, (void *)&rtps_svc },
+    /* Renesas Rapid Transfer Notify1 Char */
+    { RBLE_DECL_CHARACTERISTIC,
+       sizeof(rtps_notify_char1), sizeof(rtps_notify_char1),
TASK_ATTID(TASK_RBLE, RRTS_IDX_NOTIFY_CHAR1), RBLE_GATT_PERM_RD, (void
*)&rtps_notify_char1 },
+    /* Renesas Rapid Transfer Notify1 Value */
+    { DB_TYPE_128BIT_UUID,
+       sizeof(rtps_notify_char_val1), sizeof(rtps_notify_char_val1),
TASK_ATTID(TASK_RBLE, RRTS_IDX_NOTIFY_VAL1),
(RBLE_GATT_PERM_NI|RBLE_GATT_PERM_NOTIFY_COMP_EN), (void
*)&rtps_notify_char_val_elmt1 },
+    /* Renesas Rapid Transfer Notify2 Char */
+    { RBLE_DECL_CHARACTERISTIC,
+       sizeof(rtps_notify_char2), sizeof(rtps_notify_char2),
TASK_ATTID(TASK_RBLE, RRTS_IDX_NOTIFY_CHAR2), RBLE_GATT_PERM_RD, (void
*)&rtps_notify_char2 },
+    /* Renesas Rapid Transfer Notify2 Value */
+    { DB_TYPE_128BIT_UUID,
+       sizeof(rtps_notify_char_val2), sizeof(rtps_notify_char_val2),
TASK_ATTID(TASK_RBLE, RRTS_IDX_NOTIFY_VAL2),
(RBLE_GATT_PERM_NI|RBLE_GATT_PERM_NOTIFY_COMP_EN), (void
*)&rtps_notify_char_val_elmt2 },
+    /* Renesas Rapid Transfer Notify3 Char */
+    { RBLE_DECL_CHARACTERISTIC,
+       sizeof(rtps_notify_char3), sizeof(rtps_notify_char3),
TASK_ATTID(TASK_RBLE, RRTS_IDX_NOTIFY_CHAR3), RBLE_GATT_PERM_RD, (void
*)&rtps_notify_char3 },
+    /* Renesas Rapid Transfer Notify3 Value */
+    { DB_TYPE_128BIT_UUID,
+       sizeof(rtps_notify_char_val3), sizeof(rtps_notify_char_val3),
TASK_ATTID(TASK_RBLE, RRTS_IDX_NOTIFY_VAL3),
(RBLE_GATT_PERM_NI|RBLE_GATT_PERM_NOTIFY_COMP_EN), (void
*)&rtps_notify_char_val_elmt3 },
+    /* Renesas Rapid Transfer Notify4 Char */
+    { RBLE_DECL_CHARACTERISTIC,
```

```
+       sizeof(rtps_notify_char4), sizeof(rtps_notify_char4),
TASK_ATTID(TASK_RBLE, RRTS_IDX_NOTIFY_CHAR4), RBLE_GATT_PERM_RD, (void
*)&rtps_notify_char4 },
+    /* Renesas Rapid Transfer Notify4 Value */
+    { DB_TYPE_128BIT_UUID,
+       sizeof(rtps_notify_char_val4), sizeof(rtps_notify_char_val4),
TASK_ATTID(TASK_RBLE, RRTS_IDX_NOTIFY_VAL4),
(RBLE_GATT_PERM_NI|RBLE_GATT_PERM_NOTIFY_COMP_EN), (void
*)&rtps_notify_char_val_elmt4 },
+    /* Renesas Rapid Transfer Notify Cfg Value */
+    { RBLE_DESC_CLIENT_CHAR_CONF,
+       sizeof(rtps_notify_en), sizeof(rtps_notify_en), TASK_ATTID(TASK_RBLE,
RRTS_IDX_NOTIFY_CFG), (RBLE_GATT_PERM_RD|RBLE_GATT_PERM_WR), (void
*)&rtps_notify_en },
+    /* Renesas Rapid Transfer Indicate Char */
+    { RBLE_DECL_CHARACTERISTIC,
+       sizeof(rtps_ind_char), sizeof(rtps_ind_char), TASK_ATTID(TASK_RBLE,
RRTS_IDX_IND_CHAR), RBLE_GATT_PERM_RD, (void *)&rtps_ind_char },
+    /* Renesas Rapid Transfer Indicate Value */
+    { DB_TYPE_128BIT_UUID,
+       sizeof(rtps_ind_char_val), sizeof(rtps_ind_char_val),
TASK_ATTID(TASK_RBLE, RRTS_IDX_IND_VAL), RBLE_GATT_PERM_NI, (void
*)&rtps_ind_char_val_elmt },
+    /* Renesas Rapid Transfer Indicate Cfg Value */
+    { RBLE_DESC_CLIENT_CHAR_CONF,
+       sizeof(rtps_ind_en), sizeof(rtps_ind_en), TASK_ATTID(TASK_RBLE,
RRTS_IDX_IND_CFG), (RBLE_GATT_PERM_RD|RBLE_GATT_PERM_WR), (void
*)&rtps_ind_en },
+#endif /* (PRF_SEL_RTPS) */
+
```

(5)　prf_config.h

(a)　33 行目に追加（サンプルコードの 33 行目）

インクルードするヘッダファイルを追加します。

```
+#include "rble_api_rrtp.h"
```

(b)　502 行目を変更（サンプルコードの 503～518 行目）

高速転送プロファイルのインデックス定義を追加します。

```
- SAMS_IDX_LED_CONTROL_VAL
+    SAMS_IDX_LED_CONTROL_VAL,
+
+    /* Renesas Rapid Transfer Service */
+    RRTS_IDX_SVC,
+    RRTS_IDX_NOTIFY_CHAR1,
+    RRTS_IDX_NOTIFY_VAL1,
+    RRTS_IDX_NOTIFY_CHAR2,
+    RRTS_IDX_NOTIFY_VAL2,
+    RRTS_IDX_NOTIFY_CHAR3,
+    RRTS_IDX_NOTIFY_VAL3,
+    RRTS_IDX_NOTIFY_CHAR4,
+    RRTS_IDX_NOTIFY_VAL4,
+    RRTS_IDX_NOTIFY_CFG,
+    RRTS_IDX_IND_CHAR,
+    RRTS_IDX_IND_VAL,
+    RRTS_IDX_IND_CFG,
```

(6)　prf_sel.h

(a)　38 行目を変更（サンプルコードの 38～41 行目）

不要なプロファイルを無効化します。

```
-#define PRF_SEL_PXPM    1   /* Proximity Profile Monitor role */
-#define PRF_SEL_PXPR    1   /* Proximity Profile Reporter role */
-#define PRF_SEL_FMPL    1   /* Find Me Profile Locator role */
-#define PRF_SEL_FMPT    1   /* Find Me Profile Target role */
+#define PRF_SEL_PXPM    0   /* Proximity Profile Monitor role */
+#define PRF_SEL_PXPR    0   /* Proximity Profile Reporter role */
+#define PRF_SEL_FMPL    0   /* Find Me Profile Locator role */
+#define PRF_SEL_FMPT    0   /* Find Me Profile Target role */
```

(b)　61 行目を変更（サンプルコードの 61,62 行目）

不要なプロファイルを無効化します。

```
-#define PRF_SEL_ANPC    1   /* Alert Notification Profile Client role */
-#define PRF_SEL_ANPS    1   /* Alert Notification Profile Server role */
+#define PRF_SEL_ANPC    0   /* Alert Notification Profile Client role */
+#define PRF_SEL_ANPS    0   /* Alert Notification Profile Server role */
```

(c)　102 行目に追加（サンプルコードの 103～106 行目）

高速転送プロファイル（サーバロール）を有効化します。

```
+/* Renesas original custom profile selection */
+#define PRF_SEL_RTPC    0   /* Renesas Rapid Transfer Profile Client role */
+#define PRF_SEL_RTPS    1   /* Renesas Rapid Transfer Profile Server role */
+
```

2.4.3　プロジェクトファイル

BLE ソフトウェアに複数の環境向けプロジェクトが存在しています。本アプリケーションノートの Peripheral 機器では、下記階層のプロジェクトを使用します。

WORKSPACE Folder の階層については、「2.3.1 プロジェクト階層」を参照してください。

```
Project_Source　（WORKSPACE Folder）
    └renesas
        └tools
            └project
                └CS_CCRL
                    └BLE_Embedded　（CS+ CC-RL プロジェクト: BLE_Embedded.mtpj）
                        └rBLE_Emb
                            └DefaultBuild　（出力ファイル:ROM）
```

図 2-6　使用するプロジェクト

2.4.4　ビルド設定

サブプロジェクトのビルド・ツールプロパティを変更します。

(1)　マクロ定義

表 2-4 にコンパイル・オプションタブの定義マクロ一覧を示します。下記リストの定義マクロは、変更・削除しないでください。

表 2-4　変更マクロ定義

デフォルトマクロ名	変更後マクロ名	変更内容
CFG_FULLEMB	CFG_FULLEMB	
CFG_CON=4	CFG_CON=4	【注 1】
CFG_EXMEM_NOT_PRESENT	CFG_EXMEM_NOT_PRESENT	
CFG_BLECORE_10	noCFG_BLECORE_10	変更（無効化）
CFG_PROFEMB	noCFG_PROFEMB	変更（無効化）
CFG_SECURITY_ON	noCFG_SECURITY_ON	変更（無効化）
CFG_RBLE	noCFG_RBLE	変更（無効化）
CFG_USE_EEL	CFG_USE_EEL	
CFG_FW_NAK	noCFG_FW_NAK	変更（無効化）
CONFIG_EMBEDDED	CONFIG_EMBEDDED	
_USE_CCRL_RL78	_USE_CCRL_RL78	
CFG_SAMPLE	noCFG_SAMPLE	変更（無効化）
noUSE_SAMPLE_PROFILE	noUSE_SAMPLE_PROFILE	
noCFG_USE_PEAK	CFG_USE_PEAK	変更（有効化）
noUSE_FW_UPDATE_PROFILE	noUSE_FW_UPDATE_PROFILE	
CLK_HOCO_8MHZ	CLK_HOCO_32MHZ	変更（クロック変更）
CLK_SUB_XT1	CLK_SUB_XT1	
noCFG_PKTMON	noCFG_PKTMON	

注 1：「Bluetooth Low Energy プロトコルスタックユーザーズマニュアル」(https://www.renesas.com/ja-jp/doc/products/mpumcu/doc/rl78/r01uw0095jj0122-g1dum.pdf)の「6.1.1 最大同時接続台数」では Slave (Peripheral)として動作する場合は「1」に設定するとありますが、More Data を使用して通信する場合、More Data の個数と同一もしくはそれ以上の値を「CFG_CON」に設定してください。

(2)　ユーザ・オプション・バイト

動作クロックを変更します。「表 2-4　変更マクロ定義」のクロック設定と必ず同期させてください。

表 2-5　変更ユーザ・オプション・バイト

ユーザ・オプション・バイト	変更前	変更後
アドレス 0x000C2	EFFFAA（8MHz）	EFFFE8（32MHz）

2.5　Central 機器

Central 機器は、Peripheral 機器の BD Address をサーチし自動的に接続した後、Peripheral 機器から送信されたデータと 1 秒間辺りのスループット値について、ターミナル・ソフトウェア画面を通して出力表示します。

2.5.1　変更差分対象ファイル一覧

Central 機器の BLE ソフトウェアについて、ベースソフトウェアからの変更差分を以下に示します。

表 2-6　変更差分ファイル一覧

階層	ファイル名	処置	備考
r_BLE/src/include	rble_api_rrtp.h	追加	RRTP で使用する定数や構造体の定義
r_BLE/src/sample_app	rble_app_rrtpc.c	追加	RRTP のクライアント API とコールバック関数
r_BLE/src/sample_app	rble_app_rrtpc.h	追加	rble_app_rrtpc.c のヘッダファイル
r_BLE/src/sample_profile	rtpc.c	追加	RRTP クライアントのサービス処理関数群
bleip/src/common	co.bt.h	変更	
r_BLE/src/include	rble_app.h	変更	
r_BLE/src/sample_app	Console.c	変更	
r_BLE/src/sample_app	Console.h	変更	
r_BLE/src/sample_app	menu_sel.c	変更	
r_BLE/src/sample_app	rble_sample_app.c	変更	
r_BLE/src/sample_app	rble_sample_app_gap_sm_gatt.c	変更	
renesass/src/arch/rl78	arch_main.c	変更	
renesass/src/arch/rl78	config.h	変更	
renesass/src/arch/rl78	main.c	変更	
renesass/src/arch/rl78	prf_sel.h	変更	
renesass/src/driver/dataflash	dataflash.c	変更	
renesass/src/driver/dataflash	dataflash.h	変更	
renesass/src/driver/uart	uart.c	変更	

2.5.2　変更差分詳細

BLE ソフトウェアからの差分情報について、以下に示します。変更指定行は変更後の行位置を主に示します。行頭の「－」は行の削除、「＋」は行の追加を表します。

(1)　co_bt.h

(a)　42 行目に追加（サンプルコードの 42 行目）

自動接続の定義を追加します。

```
+ #define __AUTO_CONNECT_DEMO__
```

(b)　681 行目に追加（サンプルコードの 682～688 行目）

自動接続時の Connection Interval を格納する型を追加します。

```
+ #ifdef __AUTO_CONNECT_DEMO__
+ /* connection interval variable */
+ struct con_intval
+ {
+  uint8_t     val;
+ };
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(2)　rble_app.h

(a)　18 行目に追加（サンプルコードの 18～19 行目）

接続処理分岐用マクロ定義を追加します。

```
+ #define __THROUGHPUT_TEST__
+
```

(b)　22 行目に追加（サンプルコードの 24～32 行目）

接続処理用に組み込むヘッダファイルを追加します。

```
+ #ifdef __THROUGHPUT_TEST__
+ #if !defined(_USE_RWBLE_SOURCE)
+ #include "arch.h"
+ #include "rwke_api.h"
+ #else /* !defined(_USE_RWBLE_SOURCE) */
+ #include  "ke_task.h"
+ #endif
+ #endif
+
```

(c)　34 行目に追加（サンプルコードの 45～52 行目）

接続処理用にマクロ定義を追加します。

```
+ #ifdef __THROUGHPUT_TEST__
+ /* Task Infomation */
+ #define DEMO_STATE_MAX    1   /* Max State Num */
+ #define DEMO_IDX_MAX      1   /* Max ID Num */
+
+ #define DEMO_DATA_SEND    1   /* Task API ID */
+ #endif
+
```

(d)　81 行目に追加（サンプルコードの 100～109 行目）

接続処理用に extern 宣言を追加します。

```
+
+ #ifdef __THROUGHPUT_TEST__
+ /* Status Handler */
```

```
+ extern const struct ke_state_handler Demo_state_handler[ DEMO_STATE_MAX ];
+ /* Default Handler */
+ extern const struct ke_state_handler Demo_default_handler;
+ /* Status */
+ extern ke_state_t Demo_State[ DEMO_IDX_MAX ];
+ #endif
+
```

(3) Console.c

(a) 18 行目に追加（サンプルコードの 18～19 行目）

接続処理用の分岐用マクロ定義を追加します。

```
+ #define __THROUGHPUT_TEST__
+
```

(b) 41 行目のコメントアウト（サンプルコードの 43 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(c) 45 目のコメントアウト（サンプルコードの 47 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(d) 50 行目のコメントアウト（サンプルコードの 52 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(e) 54 行目のコメントアウト（サンプルコードの 56 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(f) 82 行目のコメントアウト（サンプルコードの 84 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(g) 84 行目のコメントアウト（サンプルコードの 86 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(h) 244 行目に追加（サンプルコードの 246～249 行目）

接続処理用に変数の extern 宣言を追加します。

```
+ #ifdef __THROUGHPUT_TEST__
+ extern uint8_t      RBLE_Test_Start_Flg;
+ #endif /* __THROUGHPUT_TEST__ */
+
```

(i) 608 行目に追加（サンプルコードの 614～618 行目）

接続処理時の条件分岐処理を追加します。

```
+
+ #ifdef __THROUGHPUT_TEST__
+ if ( false == RBLE_Test_Start_Flg ) {
```

```
+ #endif /* __THROUGHPUT_TEST__ */
+
```

(j)　630 行目に追加（サンプルコードの 641～643 行目）

接続処理時の条件分岐終了を追加します。

```
+ #ifdef __THROUGHPUT_TEST__
+ }
+ #endif /* __THROUGHPUT_TEST__ */
```

(k)　674 行目のコメントアウト（サンプルコードの 688 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(l)　788 行目のコメントアウト（サンプルコードの 802 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif /* USE_CUSTOM_DEMO */
+ /*#endif*/ /* USE_CUSTOM_DEMO */
```

(4)　Console.h

(a)　83 行目のコメントアウト（サンプルコードの 83 行目）

RRTP サンプルで使用するためのマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(b)　89 行目のコメントアウト（サンプルコードの 89 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(c)　100 行目のコメントアウト（サンプルコードの 100 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(d)　105 行目のコメントアウト（サンプルコードの 105 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(5)　menu_sel.c

(a)　206 行目に追加（サンプルコードの 206 行目）

自動接続時に処理しないようマクロ分岐処理を追加します。

```
+ #ifndef __AUTO_CONNECT_DEMO__
```

(b)　209 行目に追加（サンプルコードの 210 行目）

マクロ分岐終了を追加します。

```
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(6)　rble_sample_app.c

(a)　22 行目に追加（サンプルコードの 22～24 行目）

接続処理用の分岐用マクロ定義を追加します。

```
+ #define __THROUGHPUT_TEST__
```

```
+ #define __AUTO_CONNECT_DEMO__
+
```

(b) 37 行目の変更（サンプルコードの 40～44 行目）

不要なヘッダの取り込みをコメントアウトし、必要なヘッダを追加します。

```
- #include    "push_sw.h"
+ /* #include    "push_sw.h" */
+ #endif
+ #ifdef __AUTO_CONNECT_DEMO__
+ #include    "co_bt.h"
+ #include    "dataflash.h"
```

(c) 61 行目のコメントアウト（サンプルコードの 68 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(d) 64 行目のコメントアウト（サンプルコードの 71 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(e) 70 行目に追加（サンプルコードの 77～80 行目）

自動接続で使用する関数の extern 宣言を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+ extern BOOL RBLE_Test_Select( void );
+ #endif  /* __THROUGHPUT_TEST__ */
+
```

(f) 97 行目に追加（サンプルコードの 108～119 行目）

接続処理で使用する変数の extern 宣言を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+ #define           DEVICE_SEARCH_MAX            10
+
+ extern RBLE_BD_ADDR Remote_Device;
+ extern RBLE_BD_ADDR       Device_Search_Result[ DEVICE_SEARCH_MAX ];
+ extern uint16_t          Device_Search_Cnt;
+ #endif
+
+ #ifdef __AUTO_CONNECT_DEMO__
+ extern struct bd_addr remote_bda_addr;     /* Remote Device Address */
+ extern struct con_intval df_intval;       /* connection interval */
+ #else /* __AUTO_CONNECT_DEMO__ */
```

(g) 165 行目に追加（サンプルコードの 188～190 行目）

接続処理の設定値を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+     {  6,   RBLE_Test_Select,    NULL,                          "6.Test Case
Select¥n",  },
+ #endif  /* __THROUGHPUT_TEST__ */
```

(h) 173 行目に追加（サンプルコードの 199 行目）

自動接続のマクロ分岐終了を追加します。

```
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(i) 215 行目のコメントアウト（サンプルコードの 242 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(j)　218 行目のコメントアウト（サンプルコードの 245 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(k)　256 行目に追加（サンプルコードの 283～286 行目）

接続処理のためのデータ設定処理を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+        Device_Search_Result[ 0 ] = Remote_Device;
+        Device_Search_Cnt = 1;
+ #endif
```

(l)　265 行目の変更（サンプルコードの 296 行目）

不要な関数コールをコメントアウトします。

```
-        push_sw2_start( &sw_int );
+        /* push_sw2_start( &sw_int ); */
```

(m)　267 行目に追加（サンプルコードの 298～304 行目）

自動接続のための dataflash からの取得関数コールとシリアル出力処理を追加します。

```
+ #ifdef __AUTO_CONNECT_DEMO__
+        flash_get_remote_bda();
+  printf("Dataflash 0xF2800 set Remote BLE Addr
= %02X:%02X:%02X:%02X:%02X:%02X¥n",
+     remote_bda_addr.addr[0],remote_bda_addr.addr[1],remote_bda_addr.addr[2],
+
  remote_bda_addr.addr[3],remote_bda_addr.addr[4],remote_bda_addr.addr[5]);
+        flash_get_intval();
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(n)　287 行目に追加（サンプルコードの 325 行目）

自動接続のマクロ分岐処理を追加します。

```
+ #ifndef __AUTO_CONNECT_DEMO__
```

(o)　288 行目に追加（サンプルコードの 327 行目）

マクロ分岐終了を追加します。

```
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(p)　400 行目に追加（サンプルコードの 440 行目）

自動接続のマクロ分岐処理を追加します。

```
+ #ifndef __AUTO_CONNECT_DEMO__
```

(q)　401 行目に追加（サンプルコードの 442 行目）

マクロ分岐終了を追加します。

```
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(7)　rble_sample_app_gap_sm_gatt.c

(a)　22 行目に追加（サンプルコードの 22～24 行目）

接続処理用マクロ定義を追加します。

```
+ #define __THROUGHPUT_TEST__
+ #define __AUTO_CONNECT_DEMO__
+
```

(b)　35 行目に追加（サンプルコードの 38～41 行目）

自動接続用に組み込むヘッダファイルを追加します。

```
+ #ifdef __AUTO_CONNECT_DEMO__
+ #include    "co_bt.h"
+ #include "rble_app_rrtpc.h"
+ #endif
```

(c)　102 行目の変更（サンプルコードの 109 行目）

他のソースで参照するため static を削除します。

```
- static BOOL RBLE_GAP_Device_Search_Test( void );
/* A GAP_Device_Search command is executed. */
+       BOOL RBLE_GAP_Device_Search_Test( void );
/* A GAP_Device_Search command is executed. */
```

(d)　146 行目に追加（サンプルコードの 153～155 行目）

接続処理で他ソースの関数を使用するため extern 宣言を追加します。

```
+ extern void send_data(void);
+ #ifdef  __THROUGHPUT_TEST__
+ extern BOOL RBLE_SCP_Client_Enable_Test(void);
```

(e)　147 行目に追加（サンプルコードの 157～163 行目）

接続処理で他ソースの関数を使用するため extern 宣言を追加します。

```
+ #endif
+
+ #ifdef  __THROUGHPUT_TEST__
+ extern void RBLEL_Throughput_Disp( void );
+ #endif  /* __THROUGHPUT_TEST__ */
+
+ extern BOOL RBLE_SCP_Client_Write_Char_Test( void );
/* A SCP_Client_Write_Char command is executed. */
```

(f)　354 行目に追加（サンプルコードの 374～405 行目）

接続処理で使用する宣言を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+ #define           DEVICE_SEARCH_MAX            20
+
+ extern RBLE_BD_ADDR       Device_Search_Result[];
+ extern uint16_t          Device_Search_Cnt;
+ typedef struct {
+     BOOL       Notify_en;
+     BOOL       Indicate_en;
+     BOOL       Timer_en;
+     uint16_t    Timer_interval;
+     uint8_t     Notify_len;
+     uint8_t     Indicate_len;
+ } RBLE_SCP_SAMPLE_INFO;
+ //extern RBLE_SCP_SAMPLE_INFO    scp_sample_info;
+ extern RBLE_RRTP_INFO rrtp_info;
+ extern uint8_t     RBLE_Test_Data;
+ extern int_t       RBLE_Test_Type;
+ extern uint8_t     RBLE_Test_Start_Flg;
+ extern uint8_t send_packet_start;
+ extern uint32_t packet_count;
+ #endif  /* __THROUGHPUT_TEST__ */
+
+ #ifdef __AUTO_CONNECT_DEMO__
+ uint8_t d_serch_1st_flg = 0;
+ uint8_t d_search_remote_device = 0;
```

```
+ extern struct bd_addr remote_bda_addr;     /* Remote Device Address */
+ extern struct con_intval df_intval;        /* connection interval */
+ #endif /* __AUTO_CONNECT_DEMO__ */
+ #ifdef __USE_REPEAT_CONNECTION__
+ extern uint8_t g_hdl_set_flg;
+ #endif /* __USE_REPEAT_CONNECTION__ */
+
```

(g)　434 行目のコメントアウト（サンプルコードの 483 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(h)　435 行目に追加（サンプルコードの 484～490 行目）

接続処理の処理分岐を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+           if ( 0 == RBLE_Test_Type ) {
+           RBLE_GAP_Broadcast_Enable_Test();
+           } else if ( (1 == RBLE_Test_Type) || (2 == RBLE_Test_Type) ) {
+              RBLE_GAP_Create_Connection_Test();
+           }
+ #else
```

(i)　436 行目に追加（サンプルコードの 492 行目）

マクロ分岐終了を追加します。

```
+ #endif  /* __THROUGHPUT_TEST__ */
```

(j)　436 行目のコメントアウト（サンプルコードの 493 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(k)　437 行目に追加（サンプルコードの 494～499 行目）

自動接続で peripheral 側を発見するためにサーチを繰り返すための処理を追加します。

```
+ #ifdef __AUTO_CONNECT_DEMO__
+           if(d_serch_1st_flg == 0){
+              d_serch_1st_flg = 1;
+              RBLE_GAP_Device_Search_Test();
+      }
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(l)　501 行目に追加（サンプルコードの 564 行目）

自動接続で処理を行わないようマクロ分岐処理を追加します。

```
+ #ifndef __AUTO_CONNECT_DEMO__
```

(m)　502 行目に追加（サンプルコードの 566～599 行目）

自動接続で peripheral 側を発見するためにサーチを繰り返すための処理を追加します。

```
+ #endif /* __AUTO_CONNECT_DEMO__ */
+ #ifdef  __THROUGHPUT_TEST__
+ {
+           uint16_t       i;
+
+           for ( i = 0;i < Device_Search_Cnt;i++ ) {
+ #ifdef __AUTO_CONNECT_DEMO__
+              if((Device_Search_Result[ i ].addr[5] ==
remote_bda_addr.addr[0]) &&
+                 (Device_Search_Result[ i ].addr[4] ==
remote_bda_addr.addr[1]) &&
```

```
+                  (Device_Search_Result[ i ].addr[3] ==
remote_bda_addr.addr[2]) &&
+                  (Device_Search_Result[ i ].addr[2] ==
remote_bda_addr.addr[3]) &&
+                  (Device_Search_Result[ i ].addr[1] ==
remote_bda_addr.addr[4]) &&
+                  (Device_Search_Result[ i ].addr[0] ==
remote_bda_addr.addr[5])) {
+                     d_search_remote_device = 1;
+                     Remote_Device = Device_Search_Result[ i ];
+           RBLE_Test_Type = 1;
+  Console_SetTextAttribute( CONSOLE_COLOR );
+  printf( "                          " );
+  Console_SetTextAttribute( COMMAND_COLOR );
+           RBLE_GAP_Reset_Test();
+               }
+ #else /* __AUTO_CONNECT_DEMO__ */
+               printf( "%d/%d:", i + 1, Device_Search_Cnt );
+               BdAddress_Disp( &Device_Search_Result[ i ] );
+ #endif /* __AUTO_CONNECT_DEMO__ */
+           }
+ #ifdef __AUTO_CONNECT_DEMO__
+           if(d_search_remote_device == 0){
+               Device_Search_Cnt = 1;
+               RBLE_GAP_Device_Search_Test();
+      }
+ #endif /* __AUTO_CONNECT_DEMO__ */
+ }
+ #endif
```

(n)　504 行目に追加（サンプルコードの 602～625 行目）

自動接続でサーチ結果を取得する処理を追加します。

```
+ #ifdef __AUTO_CONNECT_DEMO__
+          Adv_Rep_p = &event->param.dev_search_result.adv_resp;
+           Peer_Addr_Type = Adv_Rep_p->adv_addr_type;
+           /* The last device is saved. */
+           Remote_Device = Adv_Rep_p->adv_addr;
+
+           Add_White_List_Dev_info.dev_addr_type = Peer_Addr_Type;
+           Add_White_List_Dev_info.dev_addr = Remote_Device;
+ #ifdef  __THROUGHPUT_TEST__
+ {
+           uint16_t          i;
+
+           for ( i = 0;i < Device_Search_Cnt && Device_Search_Cnt <
DEVICE_SEARCH_MAX;i++ ) {
+               if ( 0 == memcmp( &Device_Search_Result[ i ], &Remote_Device,
sizeof( RBLE_BD_ADDR ) ) ) {
+                   break;
+               }
+           }
+           if ( i == Device_Search_Cnt && DEVICE_SEARCH_MAX !=
Device_Search_Cnt ) {
+               Device_Search_Result[ i ] = Remote_Device;
+               Device_Search_Cnt++;
+           }
+ }
+ #endif
+ #else /* __AUTO_CONNECT_DEMO__ */
```

(o)　513 行目に追加（サンプルコードの 635～650 行目）

自動接続でサーチ結果を取得する処理を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+ {
+           uint16_t          i;
+
+           for ( i = 0;i < Device_Search_Cnt && Device_Search_Cnt <
DEVICE_SEARCH_MAX;i++ ) {
+               if ( 0 == memcmp( &Device_Search_Result[ i ], &Remote_Device,
sizeof( RBLE_BD_ADDR ) ) ) {
+                   break;
+               }
+           }
+           if ( i == Device_Search_Cnt && DEVICE_SEARCH_MAX !=
Device_Search_Cnt ) {
+               Device_Search_Result[ i ] = Remote_Device;
+               Device_Search_Cnt++;
+           }
+ }
+ #endif
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(p)　530 行目に追加（サンプルコードの 668～669 行目）

接続時の Connection Interval 値をシリアル出力する処理を追加します。

```
+     /* For Connection Interval */
+     printf("Connection Interval = %f msec, ",(float)(1.25 * Con_Info_p-
>con_interval));
```

(q)　537 行目のコメントアウト（サンプルコードの 677 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #ifdef USE_CUSTOM_DEMO
+ /*#ifdef USE_CUSTOM_DEMO*/
```

(r)　538 行目に追加（サンプルコードの 678～682 行目）

接続処理の処理分岐を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+           if ( (1 == RBLE_Test_Type) || (2 == RBLE_Test_Type) ) {
+               RBLE_SCP_Client_Enable_Test();
+           }
+ #else
```

(s)　539 行目に追加（サンプルコードの 684 行目）

マクロ分岐終了を追加します。

```
+ #endif  /* __THROUGHPUT_TEST__ */
```

(t)　539 行目のコメントアウト（サンプルコードの 685 行目）

RRTP サンプルで使用するためマクロ分岐をコメントアウトします。

```
- #endif
+ /*#endif*/
```

(u)　545 行目に追加（サンプルコードの 691～694 行目）

接続処理の変数設定を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+           RBLE_Test_Start_Flg = false;
+           send_packet_start = false;
+ #endif  /* __THROUGHPUT_TEST__ */
```

(v)　547 行目に追加（サンプルコードの 697～703 行目）

切断終端処理と再接続処理を追加します。

```
+ #ifdef __USE_REPEAT_CONNECTION__
+        RBLE_RTP_Client_Disable_Test();
+        d_serch_1st_flg = 0;
+        RBLE_Test_Type = -1;
+        g_hdl_set_flg = 0;
+        RBLE_GAP_Reset_Test();
+ #endif /* __USE_REPEAT_CONNECTION__ */
```

(w)　572 行目に追加（サンプルコードの 729～737 行目）

接続処理の処理分岐を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+           if ( (1 == RBLE_Test_Type) || (2 == RBLE_Test_Type) ) {
+              uint16_t result = 0x0000;
+              if ( 500 < event-
>param.chg_connect_param_req.conn_param.latency ) {
+                 result = 0x0001;
+              }
+              RBLE_GAP_Change_Connection_Param(event-
>param.chg_connect_param_req.conhdl, result, &event-
>param.chg_connect_param_req.conn_param, Role_Status);
+           }
+ #endif  /* __THROUGHPUT_TEST__ */
```

(x)　577 行目に追加（サンプルコードの 743～775 行目）

接続処理の変数設定を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+           if ( 0 == RBLE_Test_Type ) {
+           printf("Server!¥n");
+              RBLE_Test_Data = 0;
+              RBLE_Test_Start_Flg = true;
+              if ( TRUE == scp_sample_info.Notify_en ) {
+                 RBLE_Test_Start_Flg = false;
+                 printf("Notify_en!¥n");
+                 RBLE_Test_Start_Flg = true;
+
+                 if ( 0 == packet_count ) {
+                    //send_packet_start = true;
+                     Console_Send_Timer_msg( RBLEL_Throughput_Disp );
+                     Console_Set_Timer( 100 ); // 100 * 10ms => 1S
+                 }
+ #ifndef CFG_USE_PEAK
+                 RBLE_Test_Start_Flg = false;
+                 RBLE_Test_Start_Flg = true;
+ #else
+ #ifndef CLK_HOCO_32MHZ
+                 RBLE_Test_Start_Flg = false;
+                 RBLE_Test_Start_Flg = true;
+ #endif
+ #endif
+              } else if ( TRUE == scp_sample_info.Indicate_en ) {
+                 printf("Indication_en!¥n");
+                 send_data();
+              }
+     else {
+        printf("Error! No Notify & Indication!");
```

```
+      }
+           }
+ #endif  /* __THROUGHPUT_TEST__ */
```

(y)　1058 行目の変更（サンプルコードの 1257 行目）

他のソースで参照するため static を削除します。

```
- static BOOL RBLE_GAP_Device_Search_Test( void )
+ BOOL RBLE_GAP_Device_Search_Test( void )
```

(z)　1060 行目に追加（サンプルコードの 1259 行目）

自動接続で処理をしないようマクロ分岐処理を追加します。

```
+ #ifndef __AUTO_CONNECT_DEMO__
```

(aa) 1061 行目の変更（サンプルコードの 1261 行目）

マクロ分岐終了を追加します。

```
-
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(bb) 1074 行目の変更（サンプルコードの 1274 行目）

自動接続で処理をしないようマクロ分岐処理を追加します。

```
-
+ #ifndef __AUTO_CONNECT_DEMO__
```

(cc) 1076 行目の変更（サンプルコードの 1276 行目）

マクロ分岐終了を追加します。

```
-
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(dd) 1187 行目に追加（サンプルコードの 1387～1389 行目）

接続処理の変数設定を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+     param.peer_addr_type    = RBLE_ADDR_PUBLIC;               /* Peer address
type */
+ #else
```

(ee) 1188 行目に追加（サンプルコードの 1391 行目）

マクロ分岐終了を追加します。

```
+ #endif
```

(ff)　1190 行目に追加（サンプルコードの 1394～1404 行目）

自動接続処理の Connection Interval 値設定を追加します。

```
+ #ifdef  __THROUGHPUT_TEST__
+     // For Connection Interval
+ #ifdef __AUTO_CONNECT_DEMO__
+     param.con_intv_min      = df_intval.val;
+     param.con_intv_max      = df_intval.val;
+ #else /* __AUTO_CONNECT_DEMO__ */
+     // 7.5msec (Minimum value for BLE)
+     param.con_intv_min      = 0x6;                            /* Minimum of
connection interval ( 7.5msec = 0x6 * 1.25msec ) Range:0x0006-0x0c80 */
+     param.con_intv_max      = 0x6;                            /* Maximum of
connection interval ( 7.5msec = 0x6 * 1.25msec ) Range:0x0006-0x0c80 */
+ #endif /* __AUTO_CONNECT_DEMO__ */
+ #else
```

(gg) 1192 行目に追加（サンプルコードの 1407 行目）

マクロ分岐終了を追加します。

```
+ #endif
```

(8) arch_main.c

(a) 31 行目に追加（サンプルコードの 31～32 行目）

接続処理のマクロ定義を追加します。

```
+ #define __THROUGHPUT_TEST__
+
```

(b) 43 行目コメントアウト（サンプルコードの 45 行目）

使用しないヘッダの取り込みをコメントアウトします。

```
- #include "push_sw.h"
+ /* #include "push_sw.h" */
```

(c) 251 行目に追加（サンプルコードの 253～261 行目）

接続処理でのピーク処理を有効にします。

```
+ #ifdef __THROUGHPUT_TEST__
+ #ifdef  CLK_HOCO_32MHZ
+ #ifdef CFG_USE_PEAK
+    peak_init( 4 );     /* 4msec before peak */
+ #else  /* noCFG_USE_PEAK */
+    peak_init( 0 );     /* not peak */
+ #endif /* CFG_USE_PEAK */
+ #endif /* CLK_HOCO_32MHZ */
+ #else  /* no__THROUGHPUT_TEST__ */
```

(d) 255 行目の変更（サンプルコードの 266～267 行目）

コメントの追加とマクロ分岐終了を追加します。

```
- #endif
+ #endif /* CFG_USE_PEAK */
+ #endif /* __THROUGHPUT_TEST__ */
```

(9) config.h

(a) 22 行目に追加（サンプルコードの 22～23 行目）

自動接続の定義を追加します。

```
+ #define __AUTO_CONNECT_DEMO__
+
```

(b) 81 行目に追加（サンプルコードの 83～87 行目）

自動接続で参照する dataflash 領域のアドレスを追加します。

```
+ #ifdef __AUTO_CONNECT_DEMO__
+ #define df_bleinfo_top     (0xF2800) /* for R5F11AGJ(BLOCK262) */
+ #define p_remote_bda_ptr    ((__far const struct
bd_addr*    )(df_bleinfo_top+0))
+ #define p_con_intval_ptr    ((__far const struct
con_intval* )(df_bleinfo_top+6))
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(10) main.c

(a) 36 行目に追加（サンプルコードの 36～37 行目）

接続処理の定義を追加します。

```
+ #define __THROUGHPUT_TEST__
+
```

(b) 59 行目に追加（サンプルコードの 61～65 行目）

接続処理で使用するヘッダの取り込みを追加します。

```
+ #define __THROUGHPUT_TEST__
+ #ifndef CONFIG_MODEM
+ #include "rble_app.h"
+ #endif
+ #endif
```

(c)　464 行目に追加（サンプルコードの 471 行目）

接続処理で使用しないようにマクロ分岐を追加します。

```
+ #ifndef __THROUGHPUT_TEST__
```

(d)　468 行目に追加（サンプルコードの 476 行目）

マクロ分岐終了を追加します。

```
+ #endif
```

(e)　473 行目に追加（サンプルコードの 482～491 行目）

接続処理で使用する変数の extern 宣言の追加とタスク処理を追加します。

```
+ #ifdef __THROUGHPUT_TEST__
+ {
+ extern uint8_t     RBLE_Test_Start_Flg;
+ extern uint8_t     RBLE_Test_Notify_en;
+
+     if ( (true == RBLE_Test_Start_Flg) && (true == RBLE_Test_Notify_en) ) {
+         ke_msg_send_basic( DEMO_DATA_SEND, TASK_USR_0, TASK_RBLE );
+     }
+ }
+ #else
```

(f)　476 行目に追加（サンプルコードの 495 行目）

マクロ分岐終了を追加します。

```
+ #endif
```

(11) prf_sel.h

(a)　38～41 行目の変更（サンプルコードの 38～41 行目）

使用しないプロファイルを無効化します。

```
- #define PRF_SEL_PXPM    1   /* Proximity Profile Monitor role */
- #define PRF_SEL_PXPR    1   /* Proximity Profile Reporter role */
- #define PRF_SEL_FMPL    1   /* Find Me Profile Locator role */
- #define PRF_SEL_FMPT    1   /* Find Me Profile Target role */
+ #define PRF_SEL_PXPM    0   /* Proximity Profile Monitor role */
+ #define PRF_SEL_PXPR    0   /* Proximity Profile Reporter role */
+ #define PRF_SEL_FMPL    0   /* Find Me Profile Locator role */
+ #define PRF_SEL_FMPT    0   /* Find Me Profile Target role */
```

(b)　61～62 行目の変更（サンプルコードの 61～62 行目）

使用しないプロファイルを無効化します。

```
- #define PRF_SEL_ANPC    1   /* Alert Notification Profile Client role */
- #define PRF_SEL_ANPS    1   /* Alert Notification Profile Server role */
+ #define PRF_SEL_ANPC    0   /* Alert Notification Profile Client role */
+ #define PRF_SEL_ANPS    0   /* Alert Notification Profile Server role */
```

(c)　102 行目に追加（サンプルコードの 102～105 行目）

RRTP を追加し、有効化します。

```
+
+ /* Renesas original custom profile selection */
+ #define PRF_SEL_RTPC    1   /* Renesas Rapid Transfer Profile Client role */
```

```
+ #define PRF_SEL_RTPS    0   /* Renesas Rapid Transfer Profile Server role */
```

(12) dataflash.c

(a)　831 行目に追加（サンプルコードの 831～872 行目）

dataflash から BD Address と Connection Interval 値を読み出す処理を追加します。

```
+
+ #ifdef __AUTO_CONNECT_DEMO__
+ #define D_SAMPLE_CONNECTION_INTVAL_MAX (40)
+ #define D_SAMPLE_CONNECTION_INTVAL_MIN (6)
+ struct bd_addr remote_bda_addr;
+ struct con_intval df_intval;
+ /**
+  ***********************************************************************************
+  * @brief   get remote device address from DataFlash
+  *
+  * @param[out] remote_bda_addr     remote device address
+  *
+  ***********************************************************************************
+  */
+ _DFL_CODE void flash_get_remote_bda(void)
+ {
+     uint8_t ii;
+
+     /* try to get device address from DataFlash */
+     for(ii = 0; ii < BD_ADDR_LEN; ii++)
+     {
+         remote_bda_addr.addr[ii] = p_remote_bda_ptr->addr[ii];
+     }
+ }
+
+ /**
+  ***********************************************************************************
+  * @brief   get connection interval from DataFlash
+  *
+  * @param[out] df_intval    connection interval
+  *
+  ***********************************************************************************
+  */
+ _DFL_CODE void flash_get_intval(void)
+ {
+     df_intval.val = p_con_intval_ptr->val;
+     if((df_intval.val < D_SAMPLE_CONNECTION_INTVAL_MIN) || (df_intval.val >
D_SAMPLE_CONNECTION_INTVAL_MAX))
+     {
+         df_intval.val = D_SAMPLE_CONNECTION_INTVAL_MIN;
+     }
+ }
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(13) dataflash.h

(a)　225 行目に追加（サンプルコードの 225～247 行目）

dataflash.h で追加した関数のプロトタイプ宣言を追加します。

```
+ #ifdef __AUTO_CONNECT_DEMO__
+ /**
+  ***********************************************************************************
```

```
+  * @brief   get remote device address from DataFlash
+  *
+  * @param[out] bda      remote device address
+  *
+  ***********************************************************************
+  */
+ void flash_get_remote_bda(void);
+
+ /**
+  ***********************************************************************
+  * @brief   get connection interval from DataFlash
+  *
+  * @param[out] df_intval    connection interval
+  *
+  ***********************************************************************
+  */
+ void flash_get_intval(void);
+
+ #endif /* __AUTO_CONNECT_DEMO__ */
```

(14) uart.c

(a)　131 行目に追加（サンプルコードの 131 行目）

32MHz 動作定義を追加します。

```
+ #define UART_VAL_SPS_32MHZ  0x00U
```

(b)　390 行目を変更（サンプルコードの 391 行目）

有効な処理分岐を変更します。

```
- #if (1)
+ #if (0)
```

(c)　408,410～411 行目を変更（サンプルコードの 409、411～416 行目）

通信速度を 1Mbps に変更します。

```
-       write_sfr(SPS0L, (uint8_t)((read_sfr(SPS0L) | UART_VAL_SPS_2MHZ)));
+ /*       write_sfr(SPS0L, (uint8_t)((read_sfr(SPS0L) |
UART_VAL_SPS_2MHZ)));*/
         /* baudrate 250000bps(when MCK = 2MHz) */
-       write_sfrp(UART_TXD_SDR, (uint16_t)0x0600U);
+ /*       write_sfrp(UART_TXD_SDR, (uint16_t)0x0600U);*/
-       write_sfrp(UART_RXD_SDR, (uint16_t)0x0600U);
+ /*       write_sfrp(UART_RXD_SDR, (uint16_t)0x0600U);*/
+        /* MCK = fclk/n = 32MHz */
+        write_sfr(SPS0L, (uint8_t)((read_sfr(SPS0L) | UART_VAL_SPS_32MHZ)));
+        write_sfrp(UART_TXD_SDR, (uint16_t)0x1E00U); /* 1/32(1,000,000bps) */
+        write_sfrp(UART_RXD_SDR, (uint16_t)0x1E00U); /* 1/32(1,000,000bps) */
```

2.5.3　プロジェクトファイル

BLE ソフトウェアに複数の環境向けプロジェクトが存在しています。本アプリケーションノートの Central 機器では、下記階層のプロジェクトを使用します。

WORKSPACE Folder の階層については、「2.3.1 プロジェクト階層」を参照してください。

図 2-7　使用するプロジェクト

2.5.4　ビルド設定

サブプロジェクトのビルド・ツールプロパティを変更します。

(1)　マクロ定義

表 2-7 にコンパイル・オプションタブの定義マクロ一覧を示します。下記リストの定義マクロは、変更・削除しないでください。

表 2-7　変更マクロ定義

デフォルトマクロ名	変更後マクロ名	追加・変更
CFG_FULLEMB	CFG_FULLEMB	
CFG_CON=4	CFG_CON=4	【注 1】
CFG_EXMEM_NOT_PRESENT	CFG_EXMEM_NOT_PRESENT	
CFG_BLECORE_10	noCFG_BLECORE_10	
CFG_PROFEMB	noCFG_PROFEMB	
CFG_SECURITY_ON	noCFG_SECURITY_ON	
CFG_RBLE	noCFG_RBLE	
CFG_USE_EEL	CFG_USE_EEL	
CFG_FW_NAK	noCFG_FW_NAK	
CONFIG_EMBEDDED	CONFIG_EMBEDDED	
_USE_CCRL_RL78	_USE_CCRL_RL78	
CFG_SAMPLE	CFG_SAMPLE	
noUSE_SAMPLE_PROFILE	noUSE_SAMPLE_PROFILE	
noCFG_USE_PEAK	CFG_USE_PEAK	変更（有効化）
noUSE_FW_UPDATE_PROFILE	noUSE_FW_UPDATE_PROFILE	
CLK_HOCO_8MHZ	CLK_HOCO_32MHZ	変更（クロック変更）
CLK_SUB_XT1	CLK_SUB_XT1	
noCFG_PKTMON	noCFG_PKTMON	

注 1：More Data を使用して通信する場合、More Data の個数と同一もしくはそれ以上の値を「CFG_CON」に設定してください。

(2)　ユーザ・オプション・バイト

表 2-8　変更ユーザ・オプション・バイト

ユーザ・オプション・バイト	変更前	変更後
アドレス 0x000C2	EFFFAA（8MHz）	EFFFE8（32MHz）

2.6　動作確認用簡易デモ

「図 1-1 サンプルプログラムの動作環境概要」の構成で、動作確認のための簡易デモを実行できます。

2.6.1　動作確認用簡易デモシステム構成

Peripheral 機器として RL78/G1D 評価ボードを利用し、生成したランダムデータから送信するパケットを作成します。

対向機の Central 機器として、RL78/G1D 評価ボードを利用し、測定データを受信します。受信したデータおよび受信スループット等の通信状況の情報は、内蔵の UART⇔USB 変換デバイス経由で PC のターミナル・ソフトウェアに表示することができます。

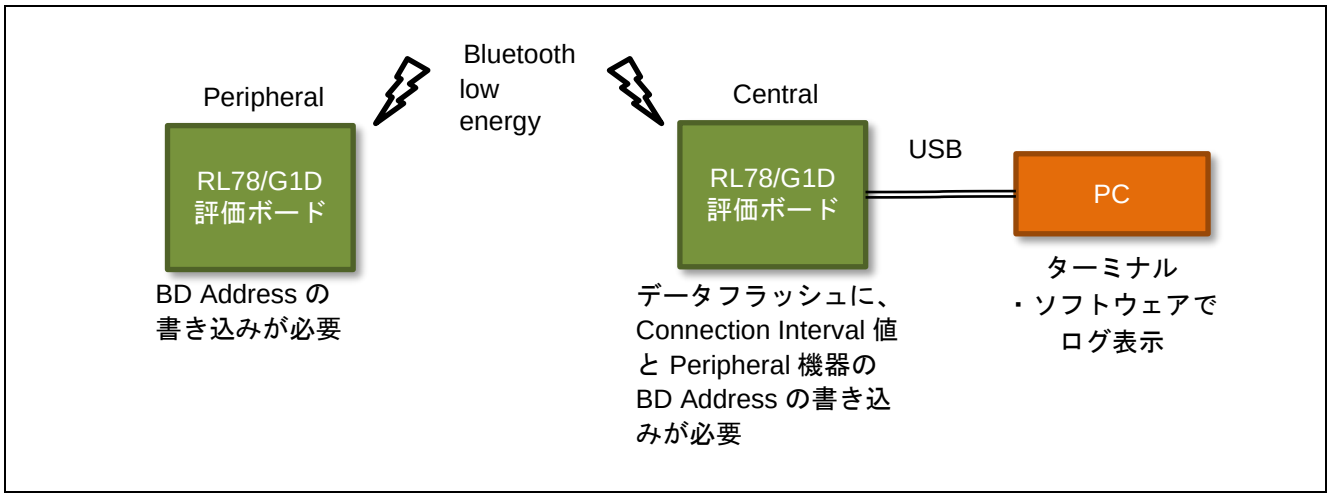


図 2-8　動作確認用簡易デモシステム構成

このデモで送信されるユーザデータ 20 バイトのデータ（パケット）のフォーマットを以下に示します。

Data 内容はランダムで、特に意味はありません。

表 2-9　送信データの内容

データ	サイズ（バイト）	内容
SN	1	シリアル番号：0x00 ～ 0xFF　【注 1】 パケットの送信の度にインクリメントします。 0xFF の次は 0x00 に戻ります。
DATA	19	内容はランダムです。

注 1：Peripheral 機器－Central 機器間の電波等の状況により、パケットの欠損や受信順番の入れ替わりが発生する場合があります。

2.6.2　Peripheral 機器（RL78/G1D 評価ボード）の準備

簡易デモのための準備内容を以下に示します。

(1)　ハードウェア構成

RL78/G1D 評価ボードを単体で使用します。

PC からの USB 給電で動作させるため、表 2-10 に示すように RL78/G1D 評価ボード上のスライドスイッチ SW7、SW8、SW11 を設定してください。スライドスイッチ設定の詳細については、RL78/G1D 評価ボードのユーザーズマニュアルを参照してください。

表 2-10　RL78/G1D 評価ボード上のスライドスイッチ SW7、SW8、SW11 の設定

スライドスイッチ	設定値
SW7	“2-3 接続”側（全て 3pin 側）
SW8	“2-3 接続”側（全て 3pin 側）
SW11	“2-3 接続”側（全て 3pin 側）

(2)　ファームウェアの書き込み

開発環境 CS+で BLE_Embedded.mtpj（「2.4.3 プロジェクトファイル」を参照）を開きます。

ファームウェアのプロジェクトが立ち上がったら、メニュー「デバッグ」→「リビルド＆デバッグ・ツールへダウンロード」を選択し、評価ボードへファームウェアをダウンロードします。

なお、ファームウェアを MOT ファイル形式で、以下のフォルダに格納済です。そのため開発環境を使用せずに、Renesas Flash Programmer を使って、このファイルを RL78/G1D のコードフラッシュに書き込むことも可能です。

表 2-11　ファームウェア格納場所

```
├──ROM_Files
│　　└──RL78_G1D_peripheral
│　　│　　└──rBLE_Emb_CCRL.mot
```

注意：RL78/G1D のコードフラッシュの書き換えの際に、ブロック#255 を削除しないでください。BD Address を書き込み済の場合、削除すると BD Address が失われます。

(3)　BD Address の書き込み

RL78/G1D 評価ボードには RL78/G1D 単体が搭載されているため、BD Address が格納されていません。RL78/G1D に、エンディアンに注意し BD Address を書き込んでください。書き込み後、SmartPhone 等を使ってスキャンし、設定とおりの BD Address が表示されるかを確認することを推奨します。

BD Address の詳細については、「Bluetooth low energy　プロトコルスタック　ユーザーズマニュアル (R01UW0095JJ)」を参照してください。

2.6.3　Central 機器（RL78/G1D 評価ボード）の準備

簡易デモのための準備内容を以下に示します。

(1)　ハードウェア構成

RL78/G1D 評価ボードと通信の結果を表示する PC を USB で接続します。「図 2-8　動作確認用簡易デモシステム構成」を参照してください。

PC からの USB 給電で動作させるため、表 2-12 に示すように RL78/G1D 評価ボード上のスライドスイッチ SW7、SW8、SW11 を設定してください。スライドスイッチ設定の詳細については、RL78/G1D 評価ボードのユーザーズマニュアルを参照してください。

表 2-12　RL78/G1D 評価ボード上のスライドスイッチ SW7、SW8、SW11 の設定

スライドスイッチ	設定値
SW7	“2-3 接続”側（全て 3pin 側）
SW8	“2-3 接続”側（全て 3pin 側）
SW11	“2-3 接続”側（全て 3pin 側）

(2)　PC 用ターミナル・ソフトウェアの準備

ログ出力用ターミナル・ソフトウェアとして、Tera Term を使用します。「設定」メニューの「端末」と「シリアルポート」の設定を以下に示します。

表 2-13　「設定」メニュー　「シリアルポート」設定時

「Tera Term: シリアルポート設定」画面	設定値
ボーレート	1,000,000　（直接値を入力）
データビット	8bit
パリティ	なし
ストップビット	1bit
フロー制御	なし
送信遅延（ミリ秒/字）	1
送信遅延（ミリ秒/行）	100

表 2-14　「設定」メニュー　「端末」設定時

「Tera Term: 端末の設定」画面	設定値
改行コード　受信	LF
改行コード　送信	CR

シリアルコンソール上に Tera Term の機能を用いて出力ログにタイムスタンプを載せることができます。タイムスタンプを出力するためには、「設定」→「その他の設定」→「ログ」タブ内の「オプション」項目の「タイムスタンプ」チェックボックスを有効にしてください。（Tera Term Version 4.86 で確認済です。）

表 2-15　「設定」メニュー　「その他の設定」設定時

「Tera Term: その他の設定」画面 「ログ」タブ	設定値
オプション　タイムスタンプ	有効

ログ保存の設定方法を以下に示します。

表 2-16　「ファイル」メニュー　「ログ」設定時

「Tera Term: ログ」画面	設定値
ファイル名	ログ出力ファイル名

(3)　コードフラッシュへの書き込み

Renesas Flash Programmer（以下 RFP と略します）を使って、RL78/G1D のコードフラッシュを書き換えてください。デモ専用のファームウェアを MOT ファイル形式で提供しています。以下のフォルダに格納済です。

表 2-17　ファームウェア格納場所

```
├──ROM_Files
│　　├──RL78_G1D_central
│　　│　　└──rBLE_Emb_CCRL.mot
```

注意：RL78/G1D のコードフラッシュの書き換えの際に、ブロック#255 を削除しないでください。BD Address を書き込み済の場合、削除すると BD Address が失われます。

(4)　データフラッシュへの書き込み

RFP を使って、デモ固有情報を書き込んでください。デモ固有情報書き込みのため、RFP のユニークコードを利用します。以下に、デモ固有情報のデータ構成とユニークコードの作成方法を示します。

コードフラッシュを書き込み済で、データフラッシュのみを書き換えする場合の書き込み方法を以下に示します。

(a)　デモ固有情報のデータ構成

データフラッシュのブロック#6（アドレス：0xF2800）に Peripheral 機器の BD Address と Connection Interval 設定値を格納する必要があります。

Peripheral 機器の BD Address は、Central 機器が Peripheral 機器をサーチした際、接続対象を特定するために使用されます。

表 2-18　Central 機器のデータフラッシュのブロック#6

Offset Address	サイズ（バイト）	内容
0x00 – 0x05	6	接続対象の Peripheral 機器の BD Address
0x06	1	Connection Interval 値（0x06 ～ 0x28）　【注 1】 設定値の 1.25 倍が Connection Interval 時間（ミリ秒）です。

注 1：設定値は 0x06 ～ 0x28（7.5 ミリ秒 ～ 50 ミリ秒）の間が有効値で、設定範囲外の場合、7.5 ミリ秒で動作します。高速転送を実現したい場合は 7.5 ミリ秒を推奨します。

(b)　ユニークコードの準備

テキストエディタでユニークコードファイル（sample_central.ruc）を編集し、接続先の Peripheral 機器の BD Address と Connection Interval 値（推奨値：0x06（7.5 ミリ秒））を設定してください。

RL78/G1D 評価ボードの場合、RL78/G1D 搭載モジュール（RTK0EN0002C01001BZ）上に貼付シールに記載の”749050-XXXXXX”（XXXXXX は数字）を BD Address として利用できます。

```
//Sample unique code file
// Peripheral BD Address(6-byte), connection interval(1-byte)
format hex
address 0xf2800
size 7
index data
000001 749050800c7606
```

接続先 Peripheral 機器の BD Address

Connection Interval 値

図 2-9　ユニークコードの記載例

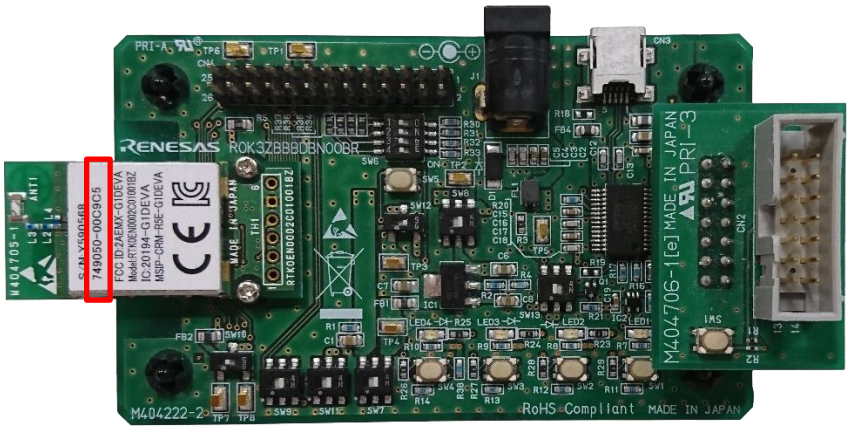


図 2-10　RL78/G1D 搭載モジュール(RTK0EN0002C01001BZ)　BD Address

(c)　ユニークコードの書き込み

1. RFP を起動し、「操作」タブで Central 機器のデモ専用のファームウェアの MOT ファイル（rBLE_Emb_CCRL.mot：RL78_G1D_peripheral フォルダ内）を選択してください。

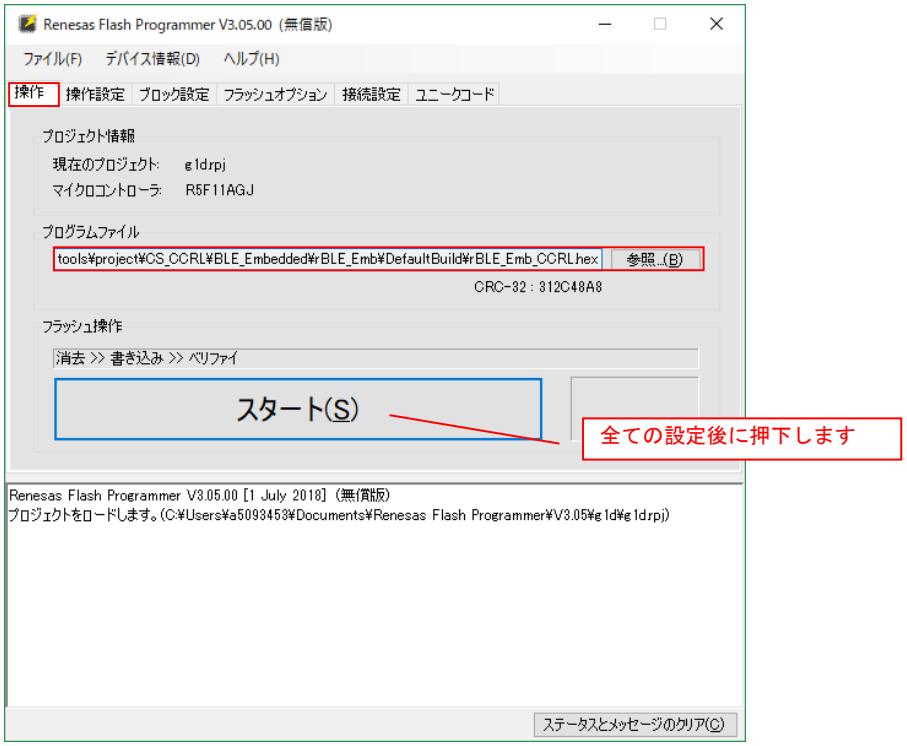


図 2-11　RFP 操作タブ

2. 「操作設定」タブで、「コマンド」に「消去」「書き込み」「ベリファイ」のチェックが入っていることを確認してください。また、「消去オプション」を「ブロック選択消去」に変更してください。

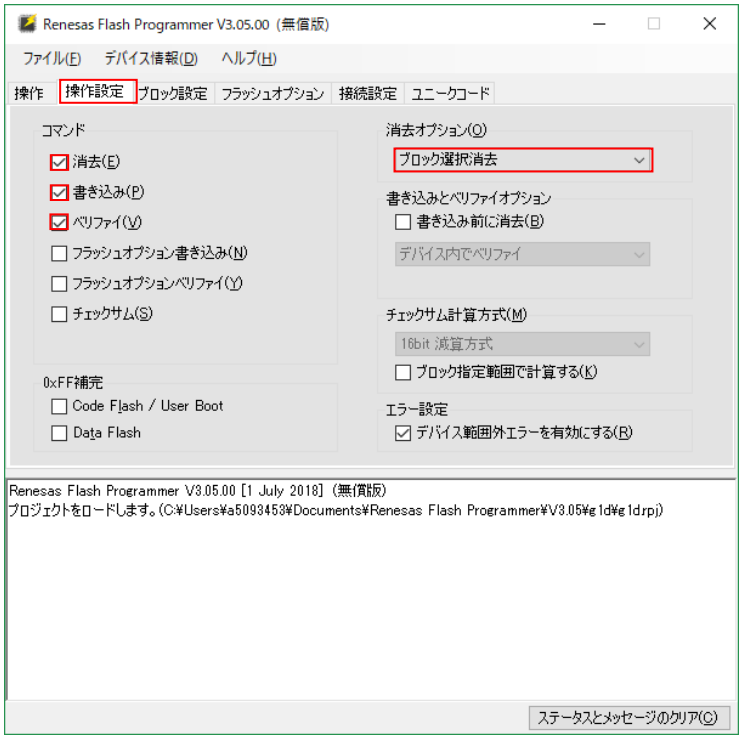


図 2-12　RFP 操作設定タブ

3. 「ブロック設定」タブで、コードフラッシュを操作対象外とするため、データフラッシュ#6（アドレス 0x000F2800）のチェックボックスのみをチェックしてください。

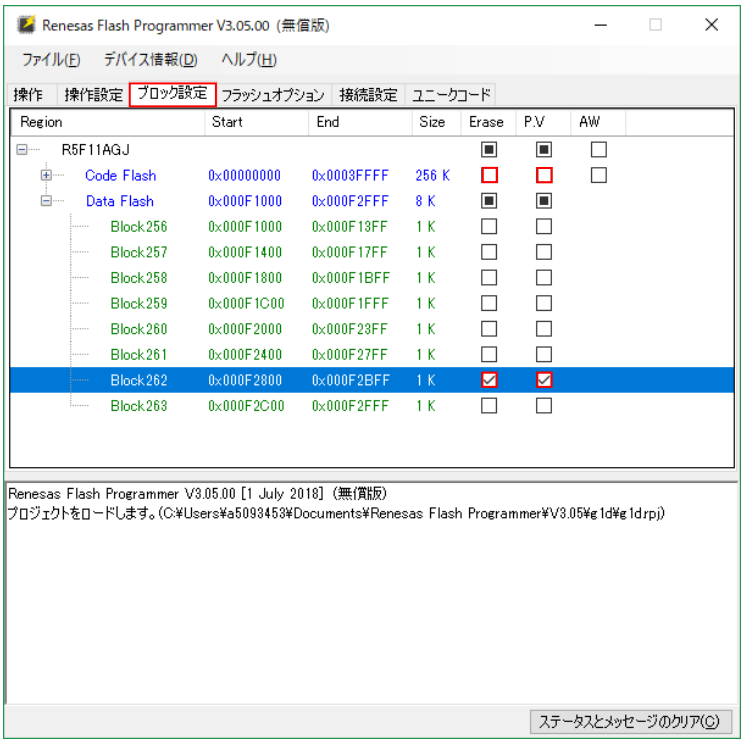


図 2-13　RFP ブロック設定タブ

4. 「ユニークコード」タブで、「有効にする」をチェックし、「範囲指定」を「全範囲」に設定し、作成したユニークコードファイル（sample_central.ruc）を選択してください。

「フラッシュオプション」タブと「接続設定」タブはデフォルト設定のまま変更の必要はありません。

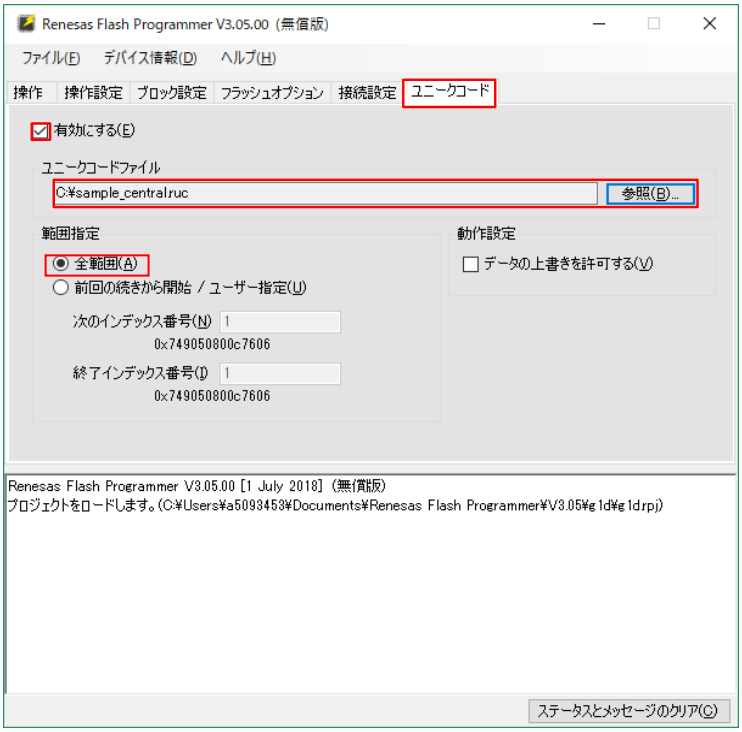


図 2-14　RFP ユニークコードタブ

5. 「操作」タブで、「スタート」を押して書き込みを実行してください（図 2-11 を参照）。
   正常終了したらウィンドウを閉じて終了します。

2.6.4　デモソフトウェア

(1)　Peripheral 側

Peripheral 側のデモソフトの状態遷移を以下に示します。

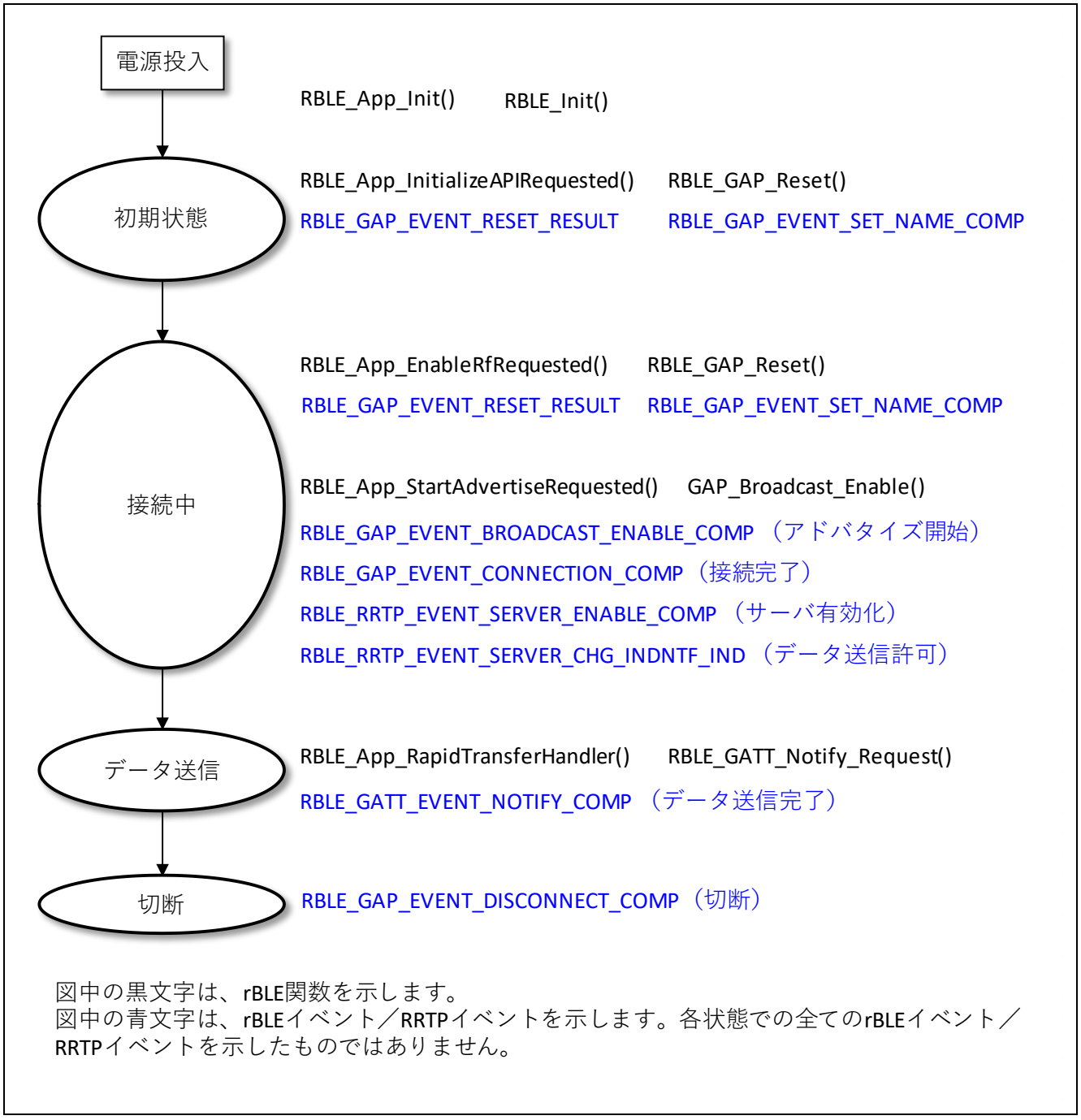


図 2-15　Peripheral 側の状態遷移

(2)　Central 側

Central 側のデモソフトの状態遷移を以下に示します。

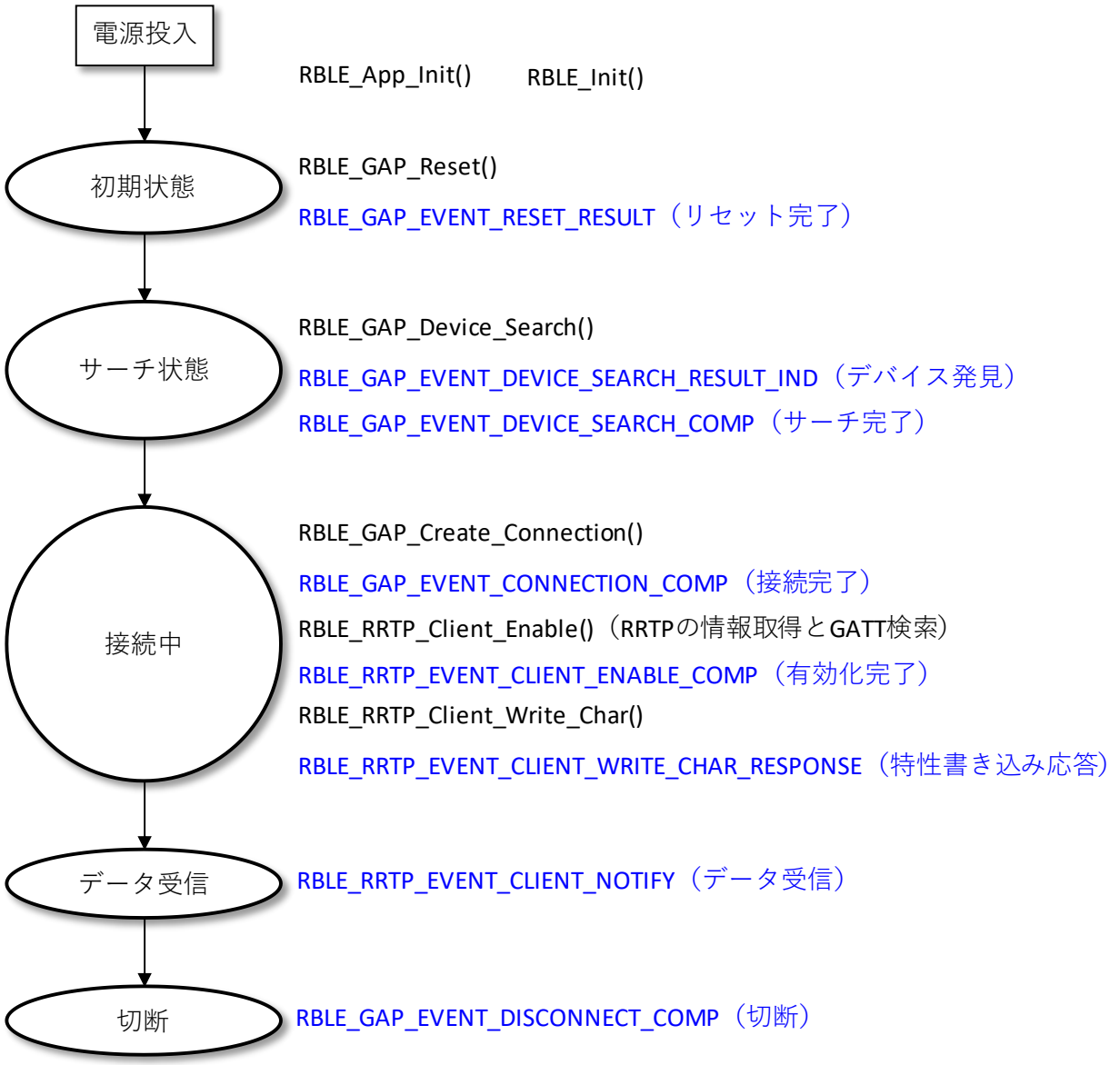


図 2-16　Central 側の状態遷移

2.6.5　デモ手順

以下の順番で操作してください。

1. デモのセットアップ

「2.6.2 Peripheral 機器（RL78/G1D 評価ボード）の準備」「2.6.3 Central 機器（RL78/G1D 評価ボード）の準備」を参照してください。
Peripheral 機器（RL78/G1D 評価ボード）に電源を供給しないでください。事前に Central 機器を設定するためです。

2. Central 機器（RL78/G1D 評価ボード）に接続された PC 上で、ターミナル・ソフトウェアを起動

RL78/G1D 評価ボードと PC を接続し、PC のターミナル・ソフトウェアの立ち上げ、「シリアル」を選択し、「ポート」に RL78/G1D 評価ボードが接続された COM 番号を設定してください。
ターミナル・ソフトウェアの設定に関しては、「2.6.3(2) PC 用ターミナル・ソフトウェアの準備」を参照してください
シリアルポートからのログを取得するように設定してください。Central 機器がリセット解除後に接続手続きを開始し、ログを出力するためです。

3. Central 機器（RL78/G1D 評価ボード）をリセット

Central 機器（RL78/G1D 評価ボード）のリセット SW (SW5)を押し、リセット状態にしてください。
リセット解除後、ターミナル・ソフトウェアにログ表示が開始されます。Peripheral 機器が Advertising 状態になるまで待ちます。

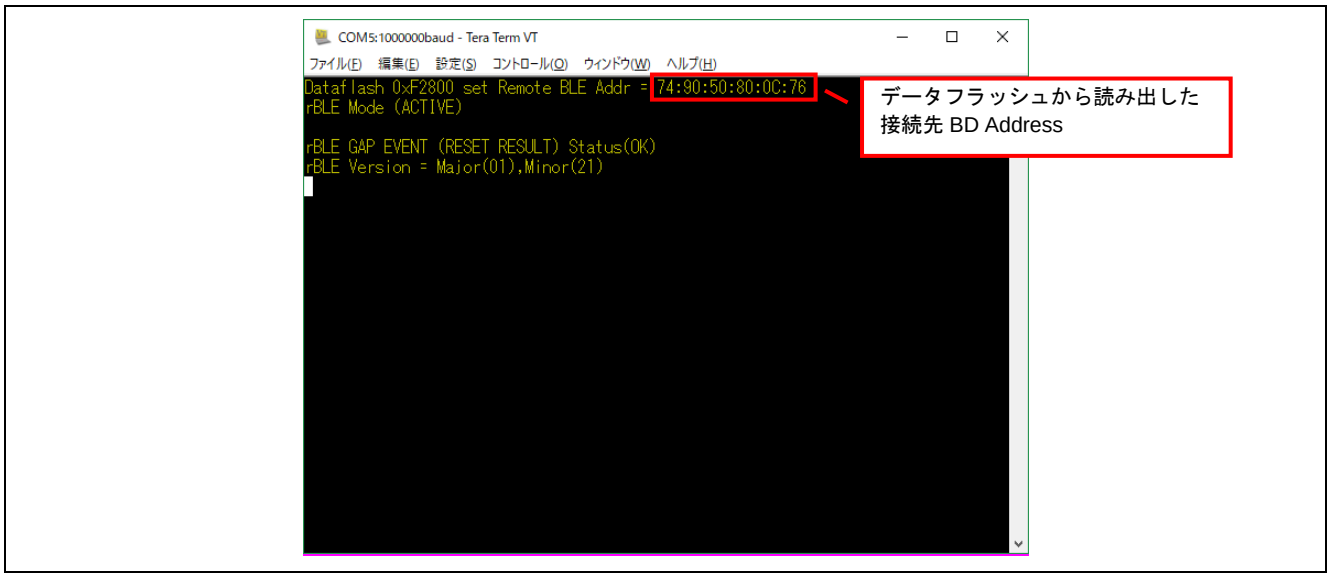


図 2-17　Peropheral 機器のアドバタイズ待ち状態のログ出力

4. Pheripehral 機器（RL78/G1D 評価ボード）に電源投入

電源投入後、自動でアドバタイズを開始します。なお、Peripheral 機器は接続後、データを送信開始します。

5. Central 機器（RL78/G1D 評価ボード）が Peripheral 機器（RL78/G1D 評価ボード）と接続

接続が完了すると、Central 機器に接続されたターミナル・ソフトウェアに受信したデータおよび受信スループット等の情報が表示されます。

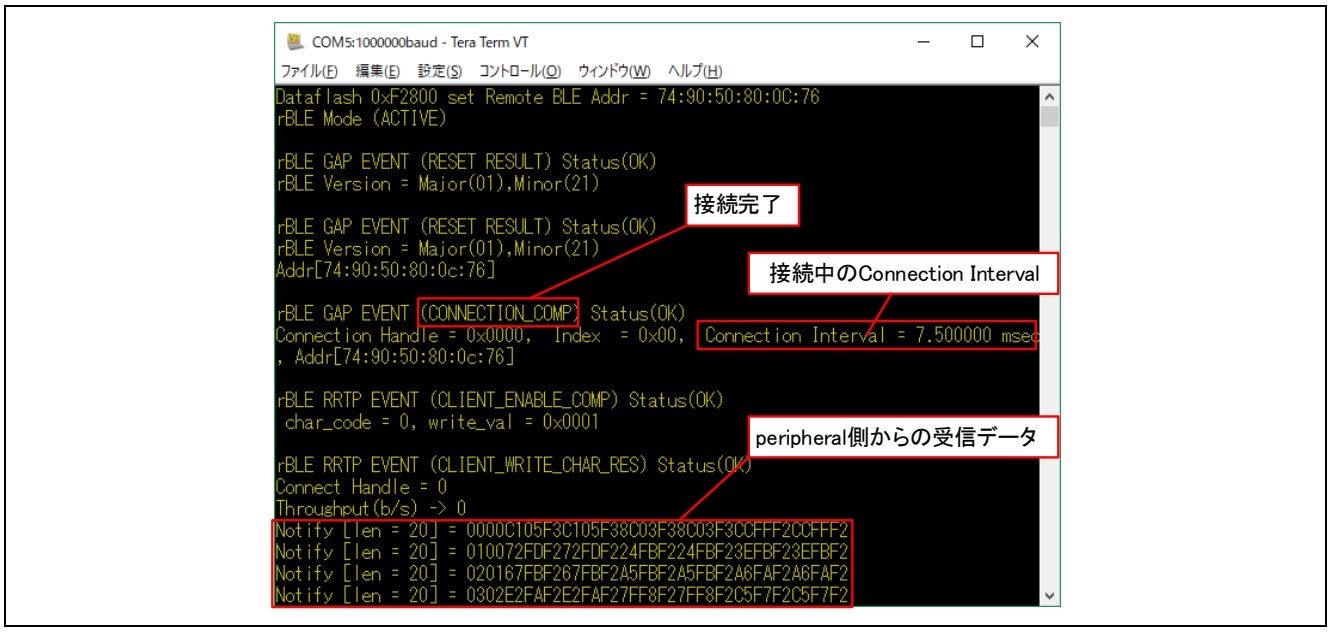


図 2-18　自動接続完了とデータ受信のログ出力

6. デモを停止させるの操作

Peripheral 機器（RL78/G1D 評価ボード）のリセット SW (SW5)を押してください。切断状態になります。リセット解除後も同じ状態です。

7. ログのファイルをクローズ

ターミナル・ソフトウェアの「ファイル」メニューの「ログダイアログを表示」を選択し、「閉じる」ボタンを押してください。

8. デモを再開させる場合の操作

以下の手順を実施してください。

① ターミナル・ソフトウェアの立ち上げ、「ファイル」メニューの「ログ」を選択しログ出力ファイル名を設定してください。ファイル名を変えない場合、ファイルは上書きされます。

② Central 機器（RL78/G1D 評価ボード）と Peripheral 機器（RL78/G1D 評価ボード）共にリセット SW (SW5)を押し続けてください。

③ Central 機器（RL78/G1D 評価ボード）のみリセットを解除してください。ターミナル・ソフトウェアにログ表示が開始されます。Peripheral 機器が Advertising 状態になるまで待ちます。

④ Peripheral 機器（RL78/G1D 評価ボード）のみリセットを解除してください。自動でアドバタイズを開始します。Peripheral 機器は接続後、データを送信開始します。

2.6.6　参考データ

表 2-19 に簡易デモ（ユーザアプリケーション負荷：小）の評価結果を示します。なお、電波状況に依存するため、下記評価結果は参考値です。

評価結果から、以下を推奨します。

- システムクロック周波数 16MHz 以上が必要です。ユーザアプリケーションの負荷を想定した場合、システムクロック周波数 32MHz を推奨します。

- 1Connection Interval 中のパケット数が 4 個の場合、欠損データパケットが多く発生する場合あります。1Connection Interval 中のパケット数が 3 個以下（ユーザデータのサンプリングレート：64 kbps）での利用を推奨します。

- 送信データ順番と受信データ順番が入れ替わることがあります。ユーザデータの一部にシリアル番号を割り当て、データ受信後にシリアル番号を基にデータを再構成することを推奨します。

表 2-19　送信評価結果（参考）

1Connection Interval 中のパケット数	RL78/G1D システムクロック周波数 [MHz]	Central 機器（クライアント）でのユーザデータの受信スループット [kbps]	送信データパケット数	欠損データパケット数【注 1】	再送データパケット数【注 2】
4	32	70.94	133,896	27,328	0
	16	69.51	132,510	28,093	8,738
	8	49.79【注 3】	93,041	0	141
3	32	64.02	122,036	1	0
	16	64.01	122,822	0	0
	8	49.84【注 3】	94,376	0	156

注 1：受信側で受信できなかったパケット数を示します。再送データパケット数を含みません。

注 2：BLE 間の再送手続き等により、受信側で受信順番の入れ替わりが発生したパケット数を示します。受信側でシリアル番号のパケット欠損が発生したが、後で欠損したシリアル番号のパケットを受信した場合を指します。

注 3：欠損パケット無しにも関わらず、受信スループットが理論値に達していないため、送信データレート（BLE ソフトウェアへのデータ送信処理速度）が律速しています。

3.　高速データ転送動作デモについて

センサデバイスからの測定データを送信し、対向機を経由し PC で受信データを表示するデモを説明しています。

3.1　高速データ転送動作デモシステム

3.1.1　システム構成

Peripheral 機器として血圧測定評価キットを利用し、RL78/H1D 上の 24bitΔΣAD コンバータで取得したデータから送信するパケットを作成します。

対向機の Central 機器として、RL78/G1D 評価ボードを利用し、測定データを受信します。受信したデータおよび受信スループット等の通信状況の情報は、内蔵の UART⇔USB 変換デバイス経由で PC のターミナル・ソフトウェアに表示することができます。

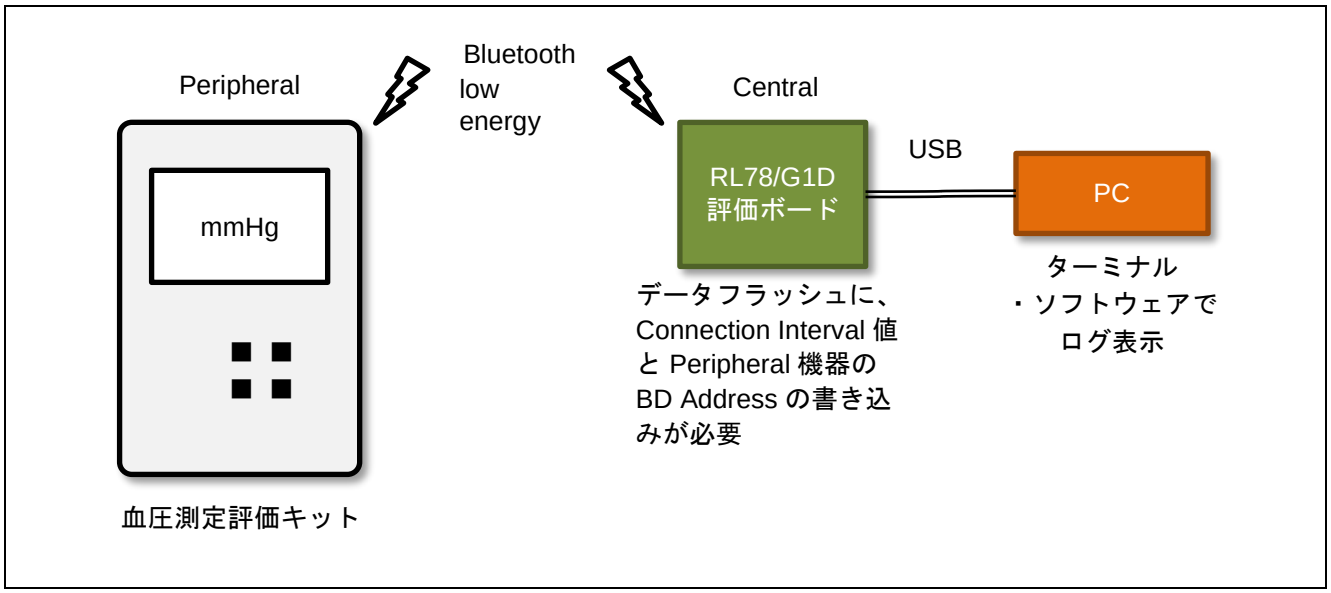


図 3-1　高速データ転送動作デモシステム構成

3.1.2　送信データ（パケット）フォーマット

以下に送信データの内容と送信データ（パケット）フォーマットを示します。

血圧測定評価キットのΔΣAD コンバータを使って取得した測定データと、それに内蔵するデジタルフィルタ（IIR フィルタ）を適用したデータ（各 3 バイトで 1 セット）とし、1 パケット内に最大 3 セットの測定データを格納します。

また、各パケットにはシリアル番号が格納されており、パケット欠損等の通信異常の発生状況を受信側で確認することができます。

血圧測定評価キットを使ったデモでは、測定データのサンプリングレートを約 1K サンプル/s に設定し、約 52kbps のデータ通信を行っています。

表 3-1　送信データの内容

データ	サイズ（バイト）	内容
SN	1	シリアル番号：0x00 ～ 0xFF パケットの送信の度にインクリメントします。 0xFF の次は 0x00 に戻ります。
DN	1	データ数：0x01 ～ 0x03 1 パケットに含まれる測定データセットの数を示します。
DATA_AD	3 【注 1】	測定データ 24 ビット AD 変換値
DATA_DF	3 【注 1】	測定データをデジタルフィルタ処理した後のデータ

注 1：それぞれのデータはリトルエンディアンで格納されます。

送信順番　→

SN	DN	DATA_AD ①	DATA_DF ①	DATA_AD ② 注 1	DATA_DF ② 注 1	DATA_AD ③ 注 2	DATA_DF ③ 注 2

注 1：DN が 0x01 の場合は 0x000000 が格納されます。
注 2：DN が 0x01 か 0x02 の場合は 0x000000 が格納されます。

図 3-2　送信データ（パケット）フォーマット

3.2　高速データ転送動作デモの準備

3.2.1　Peripheral 機器（血圧測定評価キット）の準備

高速データ転送デモのための準備内容を以下に示します。

(1)　ハードウェア構成

血圧測定評価キットのユーザーズマニュアルを参照してください。

(2)　ファームウェアの書き込み

高速データ転送動作デモ専用のファームウェアを MOT ファイル形式で提供しています。以下のフォルダに格納済です。RFP を使って、RL78/H1D と RL78/G1D のコードフラッシュのみを書き換えてください。書き込み方法は血圧測定評価キットのユーザーズマニュアルを参照してください。

注意：RL78/G1D のコードフラッシュの書き換えの際に、ブロック#254 と#255 を削除しないでください。
詳細は、血圧測定評価キットのユーザーズマニュアルを参照してください。

表 3-2　ファームウェア格納場所

```
├──ROM_Files
│    ├──BPMEK_RL78_G1D
│    │    └──rBLE_Emb_CCRL.mot              ：血圧測定評価キット RL78/G1D 用
│    ├──BPMEK_RL78_H1D
│    │    └──BPM_solution_H1D.mot           ：血圧測定評価キット RL78/H1D 用
```

(3)　Peripheral 機器（血圧測定評価キット）に対して IIR フィルタ設定

血圧測定評価キット用 GUI ツールを用いて血圧測定評価キットの IIR フィルタを設定してください。設定方法の詳細は「BPMEK 用 GUI Tool 操作マニュアル(R01AN4396JJ)」を参照してください。また設定後、血圧測定評価キットの電源を落とさないでください。

高速データ転送動作デモ専用の IIR フィルタ係数の例を以下のフォルダに格納済です。高速データ転送動作デモ用に測定データのサンプリングレートを変更しているため、通常の血圧測定評価キットを使った評価用途に使用しないでください。

表 3-3　IIR フィルタ係数例の格納場所

```
├──tools
│    ├──IIR_filter_coefficient_OSR1024_reference.csv
```

3.2.2　Central 機器（RL78/G1D 評価ボード）の準備

Central 機器のファームウェアは、共通のため、「2.6.3(3) コードフラッシュへの書き込み」を参照してください。

なお、血圧測定評価キット用の BD Address と Connection Interval 値をデータフラッシュに書き込む必要があります。テキストエディタでユニークコードファイル（sample_central.ruc）を編集し、接続先の血圧測定評価キットの BD Address と Connection Interval 値（推奨値：0x06（7.5 ミリ秒））を設定してください。データの詳細については、「2.6.3(4) データフラッシュへの書き込み」を参照してください。

3.3　高速データ転送動作デモ

3.3.1　デモ手順

以下の順番で操作してください。

1. デモのセットアップ

   「3.2 高速データ転送動作デモの準備」を参照してください。

2. Central 機器（RL78/G1D 評価ボード）に接続された PC 上で、ターミナル・ソフトウェアを起動

   RL78/G1D 評価ボードと PC を接続し、PC のターミナル・ソフトウェアの立ち上げ、「シリアル」を選択し、「ポート」に RL78/G1D 評価ボードが接続された COM 番号を設定してください。
   ターミナル・ソフトウェアの設定に関しては、「2.6.3(2) PC 用ターミナル・ソフトウェアの準備」を参照してください
   シリアルポートからのログを取得するように設定してください。Central 機器がリセット解除後に接続手続きを開始し、ログを出力するためです。

3. Central 機器（RL78/G1D 評価ボード）をリセット

   Central 機器（RL78/G1D 評価ボード）のリセット SW (SW5)を押し、リセット状態にしてください。
   リセット解除後、ターミナル・ソフトウェアにログ表示が開始されます。Peripheral 機器が Advertising 状態になるまで待ちます。

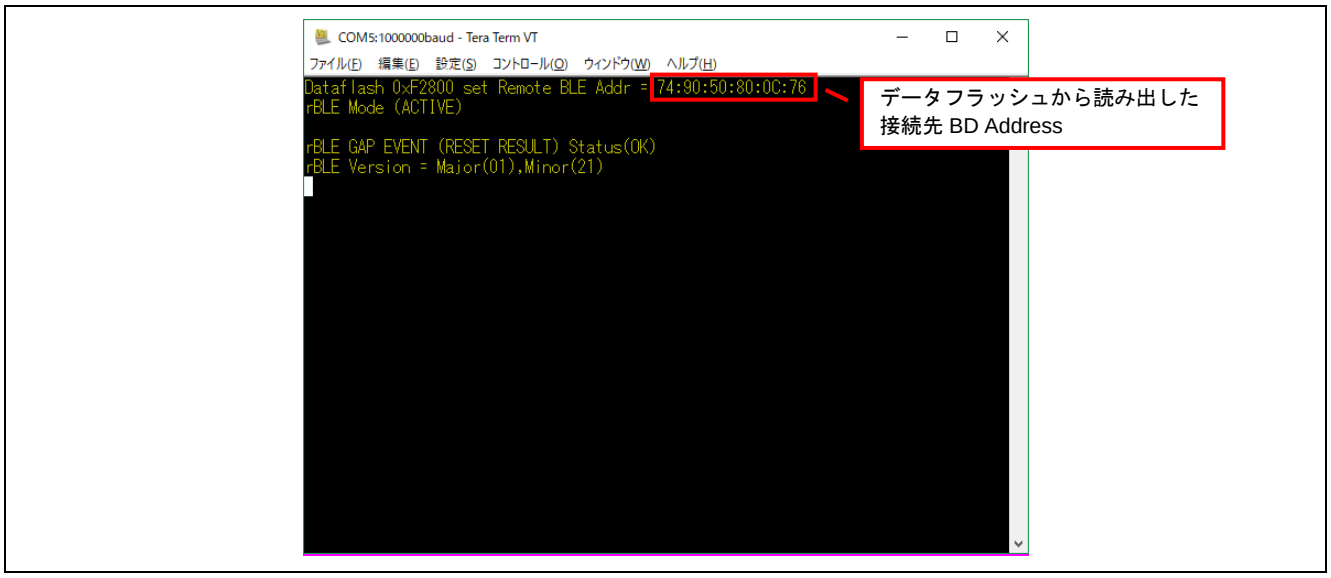


図 3-3　Peropheral 機器のアドバタイズ待ち状態のログ出力

4. Central 機器（RL78/G1D 評価ボード）が Peripheral 機器（血圧測定評価キット）と接続

以下に示す血圧測定評価キットのスイッチ(SW)操作によって、血圧測定評価キットを Advertising 状態に設定します。

① 血圧測定評価キットの電源投入後、左下 SW「Down キー」を 1 回短押ししてください。LCD 画面に“RRTP”が点灯し、RRTP モードに遷移します。
② 右上 SW「Enter キー」を 3 回短押ししてください。LCD 画面に“ADVT”が点滅し、Advertising 状態になります。（“BLEC”点灯→“ENAB”点灯→“ADVT”点滅の順で LCD 画面上に表示されます。）

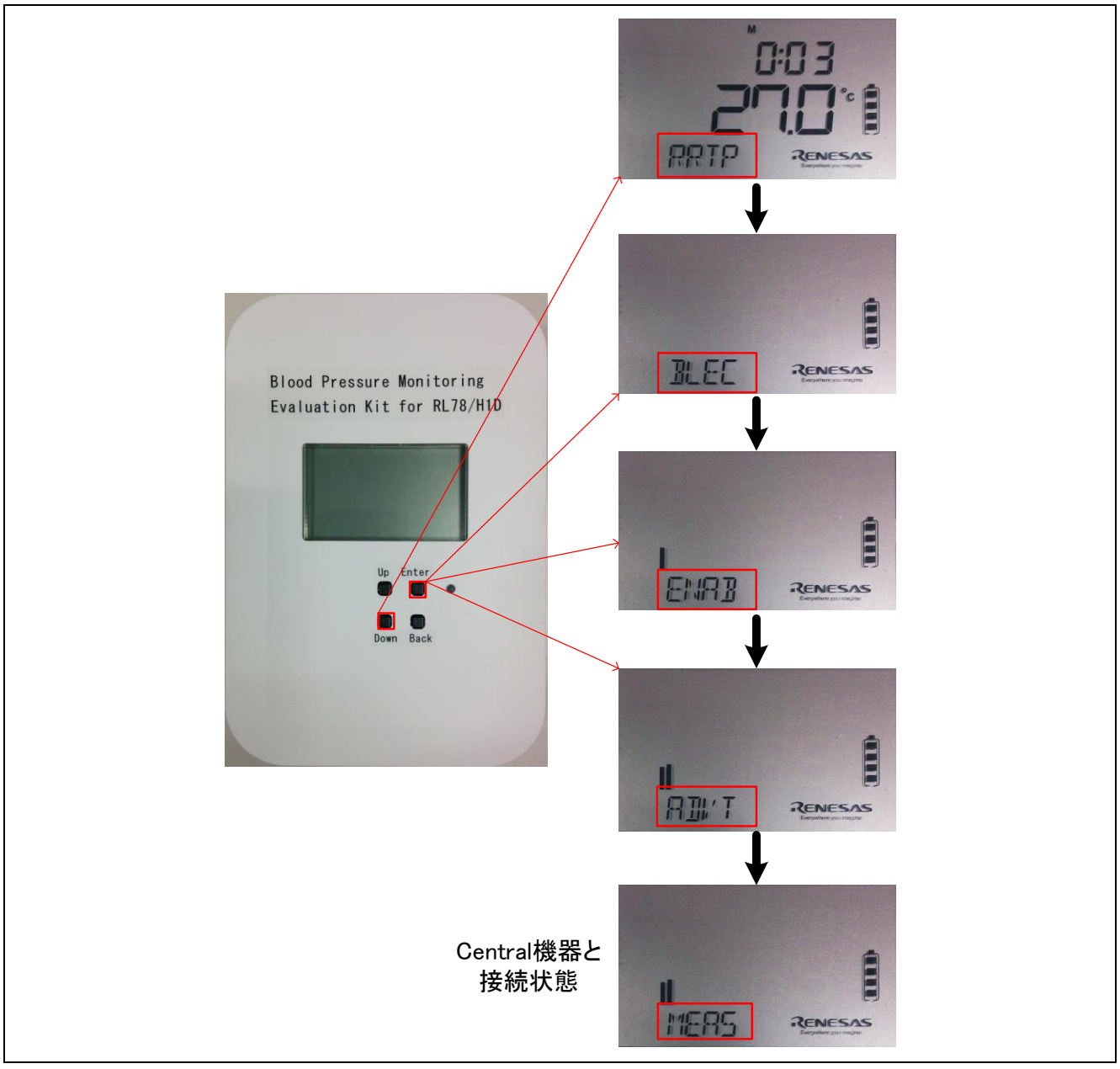


図 3-4　LCD 画面表示遷移

③ Central 機器（RL78/G1D 評価ボード）から自動で接続処理が実施されます。接続が完了すると血圧測定評価キットの LCD 画面に“MEAS”が点灯します。

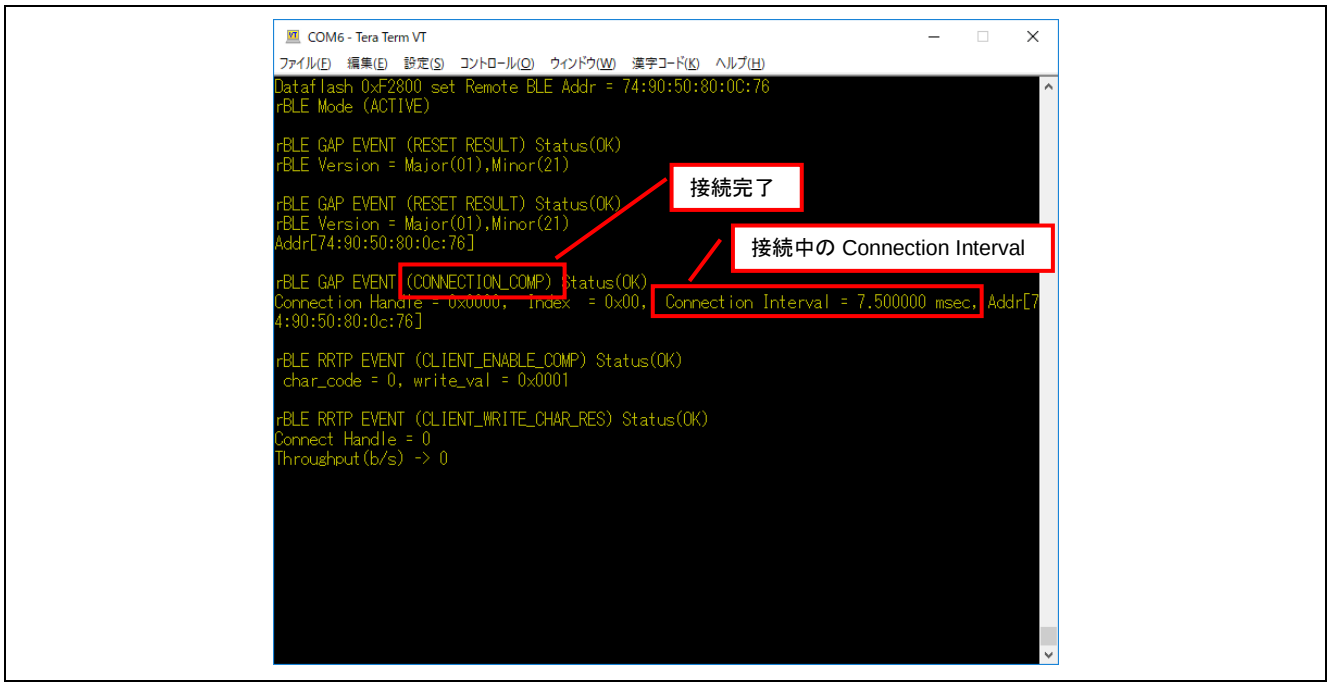


図 3-5　自動接続完了のログ出力

5.　Peripheral 機器（血圧測定評価キット）での測定開始とデータ送信

血圧測定評価キットの右上 SW「Enter キー」を 1 回短押ししてください。血圧測定評価キットのエアポンプモータが起動し測定が開始され、測定データが送信されます。

6.　Central 機器（RL78/G1D 評価ボード）に接続された PC 上でモニタ

Central 機器（RL78/G1D 評価ボード）で受信されたデータがターミナル画面に表示されます。
測定が終了すると、血圧測定評価キットの LCD 画面に“RRTP”が表示され、ログ出力が停止します。
途中で測定を停止する場合は右下 SW「Back キー」を長押ししてください。血圧測定評価キットの LCD 画面に“RRTP”が表示され、ログ出力が停止します。

図 3-6　Tera Term のログ出力保存データ（テキスト出力）

7. ログのファイルをクローズ

ターミナル・ソフトウェアの「ファイル」メニューの「ログダイアログを表示」を選択し、「閉じる」ボタンを押してください。

8. 再測定したい場合の操作

以下の手順を実施してください。

① ターミナル・ソフトウェアの立ち上げ、「ファイル」メニューの「ログ」を選択しログ出力ファイル名を設定してください。ファイル名を変えない場合、ファイルは上書きされます。

② Central 機器（RL78/G1D 評価ボード）をリセットしてください。
「3. Central 機器（RL78/G1D 評価ボード）をリセット」を参照してください。

③ Peripheral 機器（血圧測定評価キット）の右上 SW「Enter キー」を 3 回短押してください。接続状態になります。
「4. Central 機器（RL78/G1D 評価ボード）が Peripheral 機器（血圧測定評価キット）と接続」を参照してください。既に RRTP モード状態であるため、「右上 SW「Enter キー」を 3 回短押してください。」から実施することで、接続状態になります。

④ 血圧測定評価キットの右上 SW「Enter キー」を 1 回短押ししてください。測定が開始され、データが送信されます。
「5. Peripheral 機器（血圧測定評価キット）での測定とデータ送信」を参照してください。

3.3.2　グラフ化によるデータ確認

ログ出力保存データより、「測定データ 24 ビット AD 変換値」と「測定データをデジタルフィルタ処理した後のデータ」に分離し、それぞれのデータをグラフ化できます。グラフ化に同梱の Microsoft® Excel®で作成したル“RRTP_graph_sample_rev100.xlsm”ファイルのマクロを使用します。ターミナル・ソフトウェアで表示した RL78/G1D 評価ボードのログを使用するため、ログを保存してください。

Excel ファイルのマクロを有効にして、以下の手順でグラフを描画します。

1. ログ出力保存データを「graph」シートの A 列に張り付け

   「3.3.1　6 Central 機器（RL78/G1D 評価ボード）に接続された PC 上でモニタ」で取得した出力保存ログをコピーし、エクセルファイルの「graph」シートの A 列に張り付けてください。

2. 「graph」シート S1 セルにあるマクロの「実行」ボタンを押下

   「実行」ボタンを押してください。“測定データの AD 変換値とフィルタ処理後の波形”、”Central 側で受信データから計測した BLE 通信のスループット値”の 2 つのグラフが生成されます。
   注意：マクロの実行処理時間はログデータ量に依存します。

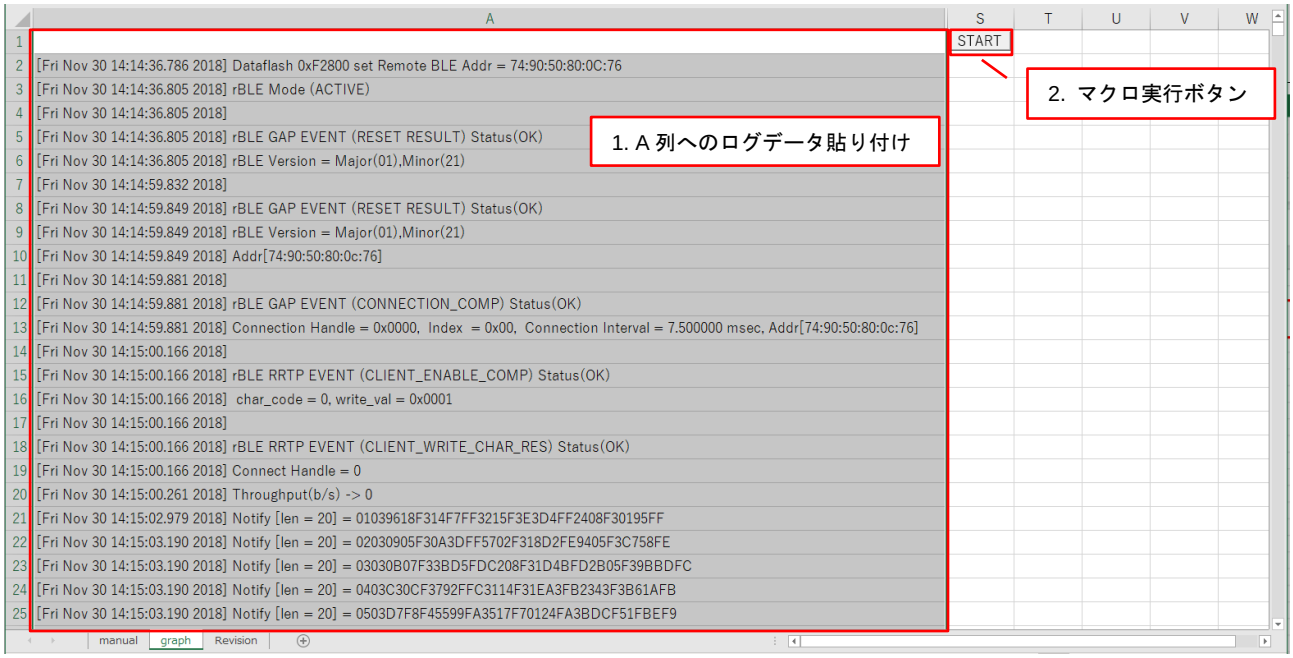


図 3-7　ログデータのグラフ化操作

ログ出力保存データのサンプルを以下フォルダに格納しています。

表 3-4　ログ出力保存データのサンプルファイル格納場所

```
├──tools
|    ├──RRTP sample.log
```

サンプルのログ出力保存データをグラフ化したものを図 **3-8** に示します。

上のグラフ上の青のグラフが“測定データ 24 ビット AD 変換値”で、赤のグラフが“測定データをデジタルフィルタ処理した後のデータ”です。また、下のグラフはスループットの結果を示しています。

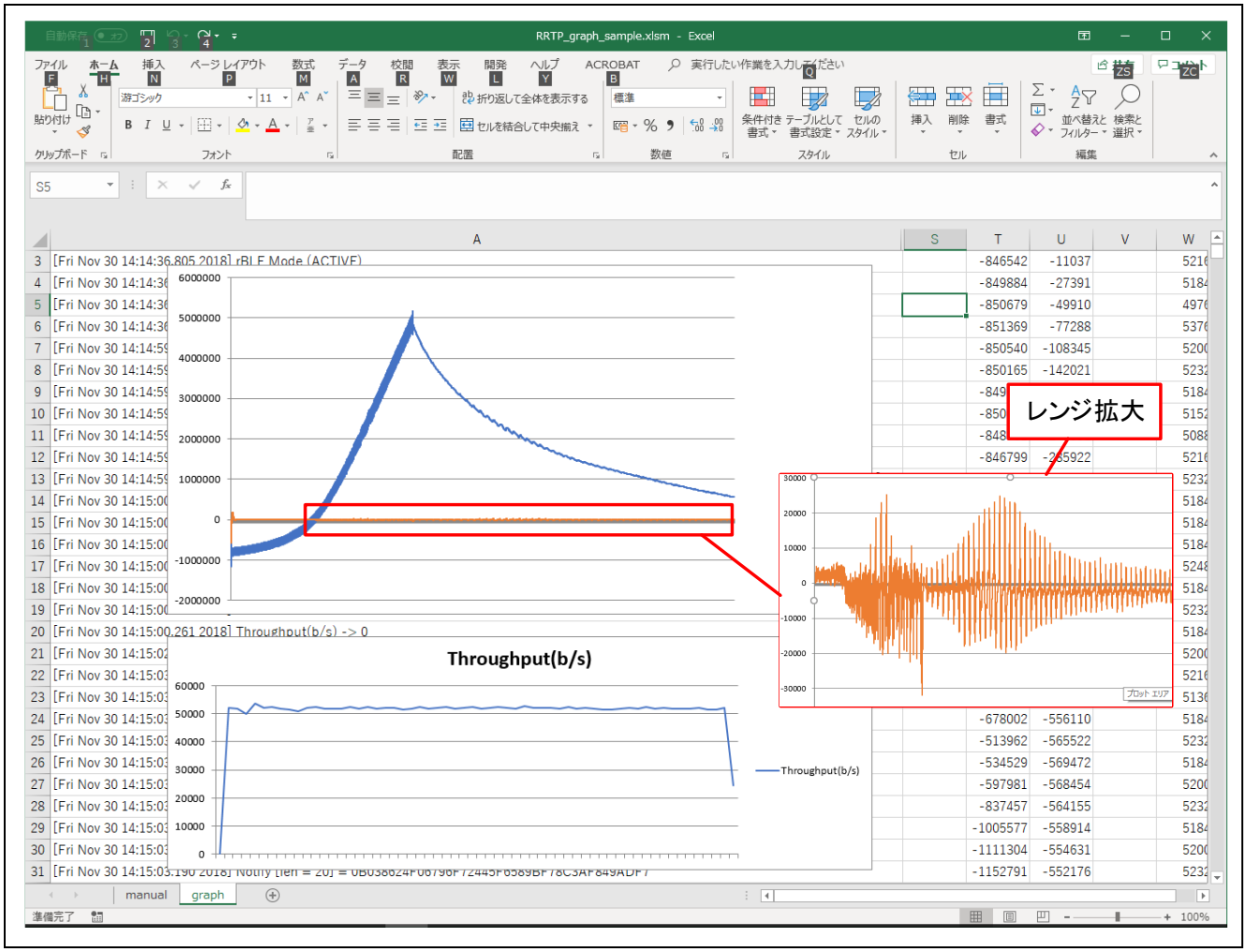


図 3-8　受信ログデータのマクロ実行結果の例

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2019.02.28	ー	新規作成
1.01	2019.05.31	6	1.5 ファイル構造　において 表 1 3　ファイル構造　の内容を更新した。
		25	2.4.1 変更差分対象ファイル一覧　において 表 2-3 変更差分ファイル一覧　の内容を更新した。
		26	2.4.2 変更差分詳細　において (1)(c) 77 行目に追加（サンプルコードの 97～99 行目）　の「自動接続用変数の初期化および BLE ソフトウェアの初期化を行います。」の文字色を赤から黒に変更した。
		33	2.4.2 変更差分詳細　において 「(6)(a) 38～41 行目を変更（サンプルコードの 38～41 行目）」を「(6)(a) 38 行目を変更（サンプルコードの 38～41 行目）」に変更した。
		33	2.4.2 変更差分詳細　において 「(6)(b) 61,62 行目を変更（サンプルコードの 61,62 行目）」を「(6)(b) 61 行目を変更（サンプルコードの 61,62 行目）」に変更した。
		33	2.4.2 変更差分詳細　において 「(6)(c) 102 行目に追加（サンプルコードの 103～106 行目）「サバロール」を「サーバロール」に修正した。
		35	2.5.1 変更差分対象ファイル一覧　において 表 2-6 変更差分ファイル一覧　のファイル名の誤記を修正更新した。
		60	2.6.4 デモソフトウェア　において 図 2-15 Peripheral 側の状態遷移　を更新した。
		70	3.3.1 デモ手順　において 6. Central 機器（RL78/G1D 評価ボード）に接続された PC 上でモニタ　の「測定評価キット」を「血圧測定評価キット」に修正した。「「Back キー」」を「右下「Back キー」」に修正した。
		71	3.3.1 デモ手順　において 8. 再測定したい場合の操作　の③ 「②右上 SW「Enter キー」を「右上 SW「Enter キー」に修正した。
		72	3.3.2 グラフ化によるデータ確認　において 「図 3-7 ログデータのグラフ化操作」を更新した。
		73	3.3.2 グラフ化によるデータ確認　において 「赤のブラフ」を「赤のグラフ」に修正した。

製品ご使用上の注意事項

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

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

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1　2017.11)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

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

Microsoft および Excel は米国 Microsoft Corporation の登録商標です。

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。