Bluetooth® Low Energy Protocol Stack

Sample Program

R01AN1375EJ0120
Rev.1.20
Jul 31, 2017

Introduction

This manual describes the installation, configuration and usage of sample program, which is included in the Bluetooth Low Energy software (the BLE software).

The BLE software refers to the set of software that includes the Bluetooth Low Energy protocol stack (the BLE protocol stack) compliant with the Bluetooth Low Energy specification (Bluetooth specification v4.2). The BLE protocol stack is designed to run on the Bluetooth Low Energy microcontroller RL78/G1D.

Target Device

RL78/G1D

Contents

1. Overview

This manual describes the installation, configuration and usage of sample program, which is included in the Bluetooth Low Energy software (the BLE software).

The BLE software refers to the set of software that includes the Bluetooth Low Energy protocol stack (the BLE protocol stack) compliant with the Bluetooth Low Energy specification (Bluetooth specification v4.2). The BLE protocol stack is designed to run on the Bluetooth Low Energy microcontroller RL78/G1D.

For details about the BLE protocol stack APIs, see Bluetooth Low Energy Protocol Stack API Reference Manual.

2. Applicability

The descriptions in this manual apply to the BLE protocol stack Version 1.20 and later.

3. Installation

The sample program of the BLE software is included in the BLE protocol stack package.

3.1. Contents

The BLE software package includes the following:

Documents

- Bluetooth Low Energy Protocol Stack User's Manual
- Bluetooth Low Energy Protocol Stack API Reference Manual
- Bluetooth Low Energy Protocol Stack Sample Program Application Note (this document)
- rBLE Command Specification

Files used for building the executable file

- Executable file
- BLE software library
- Sample source code
- Source code that configures parameters
- CS+ for CA, CX project file
- CS+ for CC project file
- IAR Embedded Workbench workspace file
- e^2 studio project file

Sample program for computer

- Executable file
- Source code
- Microsoft Visual Studio Express 2015 for Desktop project file

HCI packet monitor application for computer

- Executable file
- INI file

3.2. Installation Procedure

Copy the decompressed contents to any folder in your computer.

[Note] If using the e^2 studio, cannot be include multi-byte characters and blank in the BLE software installation folder path.

4. Sample Program

This sample program shows how to use the BLE software. The BLE software contains two sample programs.

- Console-based Sample Program Section 5
- Simple Sample Program Section 6

This section describes the common concept of a sample program. Regarding the details of sample programs, refer respective dedicated section.

Caution

Sample programs in this application note shall be handled as a sample, whose quality and reliability are not guaranteed. When you use the sample program in the final products or systems manufactured by you, evaluate the safety of them at your own risk.

4.1. Operating Environment and Development Environment

The BLE software supports two different system configurations, the modem configuration and the embedded configuration. This section describes the operating environment and development environment of the sample program in each configuration.

Modem Configuration

In the modem configuration, the controller stack, host stack and profiles are implemented together on the BLE MCU (RL78/G1D), while the application is implemented on the APP MCU separately.
The BLE software provides the sample program running on the computer as the APP MCU. You can easily evaluate the BLE software using the computer.
The sample program in the modem configuration runs on the following operating environment.

Hardware

- PC/AT™ compatible computer
  - Processor : 1.6GHz and greater
  - Memory : 1.0GB and more
  - Display : 1024×768 (XGA) and higher resolution
    65536 and more colors
  - Interface : USB 2.0 (E1 emulator and USB TTL serial cable)

Software

- Windows 7 or later
- Microsoft Visual Studio Express 2015 for Desktop
- Microsoft .NET Framework 4 + Language Pack

Embedded Configuration

In the embedded configuration, the controller stack, host stack, profiles and the application are implemented together on the BLE MCU (RL78/G1D).
The BLE software also provides the sample program running on the BLE MCU.
The sample program in the embedded configuration runs on the following operating environment.

Hardware

- RL78/G1D Test Board

Development tools and utilities

- Renesas on-chip debugging emulator E1
- Terminal Emulator for Windows

Software

- Renesas Integrated Development Environment CS+ for CA, CX or CS+ for CC or e^2 studio or IAR Embedded Workbench
- Renesas Flash Programmer V3
  (You can download it from https://www.renesas.com/software-tool/renesas-flash-programmer-programming-gui)

4.2. Structure

Figure 4-1 shows the structure of the BLE software.

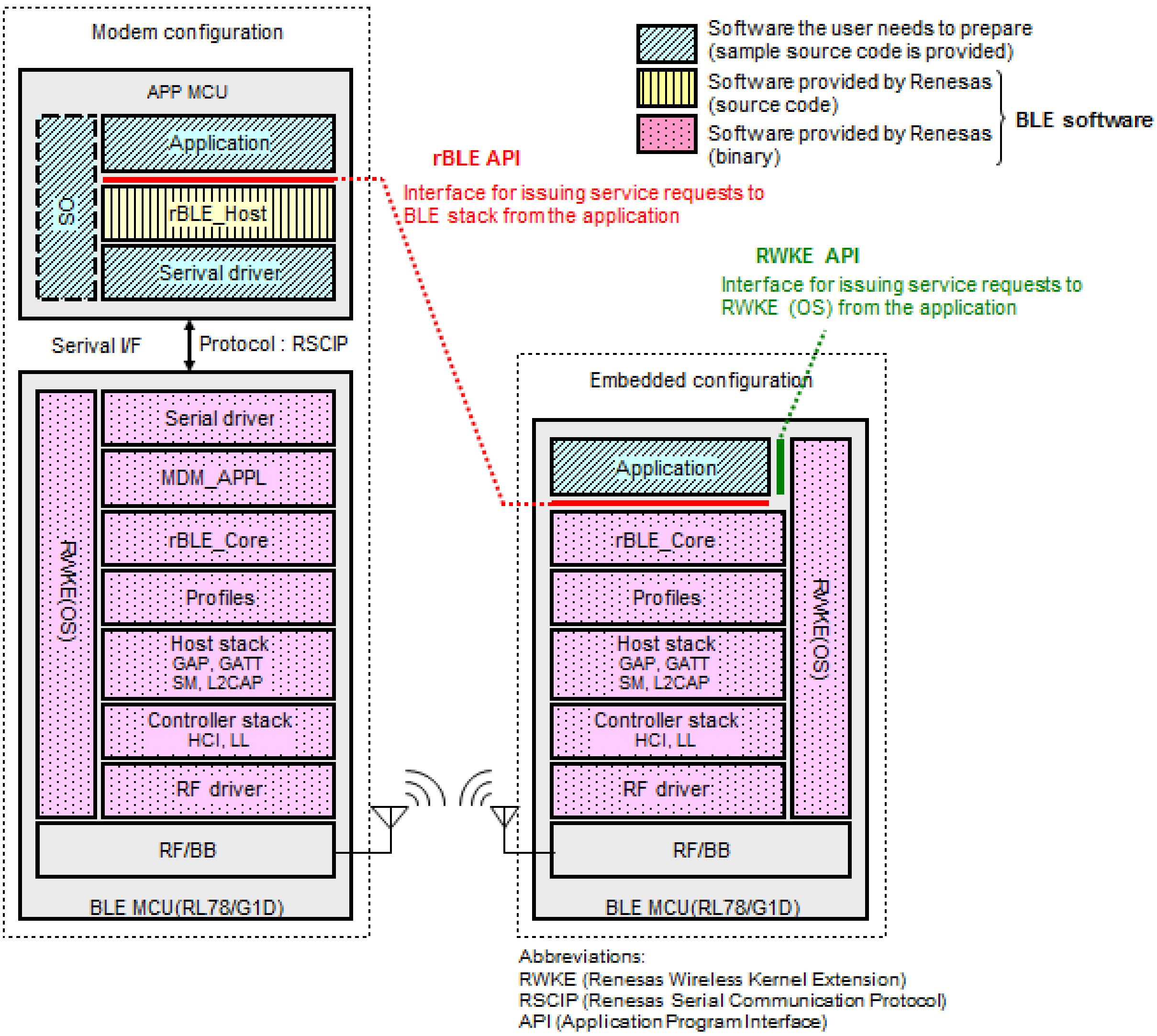


Figure 4-1 Structure of the BLE software

The BLE software in the modem configuration runs on two MCUs that are APP MCU and BLE MCU, and consists of ‘rBLE_Host’ block (block in the figure) running on the APP MCU and the software blocks (blocks in the figure) running on the BLE MCU.

In addition, the software blocks (blocks in the figure) that you need to prepare is ‘application’, ‘serial communication driver’ and ‘OS’ (Operating System) blocks. However, ‘rBLE_Host’ block does not use any OS specific resources, ‘OS’ block is not required if it does not run on the APP MCU.

On the other hand, the BLE software in the embedded configuration runs on the BLE MCU (RL78/G1D) only. The software block that you need to prepare is ‘application’ block running on the BLE MCU.

5. Usage of Console-based Sample Program

5.1. How to Change Parameters

The console-based sample program has the ability to change the parameters for rBLE API, and you will be able to execute it by selecting the parameters prepared in advance.

Parameters selection is performed as follows.

```
menu-number [blank] parameter-number
```

In the function which is called at the time of execution of the menu, treats the given arguments separated by a space and calls rBLE function.

5.2. Start the Sample Program in Modem Configuration

The console-based sample program in the modem configuration is started by executing the EXE file 'rBLE_Sample.exe' that is stored in the folder '\Renesas\BLE_Software_Ver_X_XX\BLE_Sample\project\windows\Exe'.

The sample program 'rBLE_Sample.exe' requires arguments at its start time, please edit the contents of the batch file "run.bat" stored in the same folder as the EXE file and execute it. The arguments required at the start time are explained below.

Table 5-1 Arguments required at the start time

<table>
<tr><th>Arguments</th><th>Description</th></tr>
<tr><td>COM Port Number</td><td>Specify the COM port number in the computer (e.g., COM1, COM2, …)</td></tr>
<tr><td>Baud rate</td><td>Specify between 4,800 and 250,000 to match the settings of the BLE software
<table>
<tr><th>items</th><th>settings</th></tr>
<tr><td>Baud rate</td><td>4,800 ~ 250,000 bps</td></tr>
<tr><td>data length</td><td>8 bit</td></tr>
<tr><td>parity</td><td>none</td></tr>
<tr><td>stop bit</td><td>1bit</td></tr>
<tr><td>flow control</td><td>none</td></tr>
</table></td></tr>
<tr><td>BD Address of remote device (public address)</td><td>Set the BD address (Bluetooth device address) of the remote device to be connected to. With this address, it is not required to obtain the BD address of remote device using device search, and connection procedure can be started immediately.
Use public address as BD address.</td></tr>
<tr><td>UART 2-wire Branch Connection</td><td>UART 2-wire with Branch Connection : -div2wire
UART 2 wire : none</td></tr>
</table>

Write the program into the BLE-MCU using the HEX file 'RL78_G1D_CM(*).hex' or 'RL78_G1D_IM(*).hex' or 'RL78_G1D_CCM(*).hex' stored in the following folder after the BLE software installation from package

'\Renesas\BLE_Software_Ver_X_XX\RL78_G1D\ROM_File' after installation.

These HEX files are used in 4800 bps with baud rate of serial communication.

5.3. Start the Sample Program in Embedded Configuration

Before starting the sample program in the embedded configuration, write the program into the RL78/G1D Test Board using the HEX file 'RL78_G1D_CE(*).hex' or 'RL78_G1D_IE(*).hex' or 'RL78_G1D_CCE(*).hex' stored in the following folder after the BLE software installation from package.

'\Renesas\BLE_Software_Ver_X_XX\RL78_G1D\ROM_File' after installation.

To start the sample program, reset the RL78/G1D Test Board.

However, to use this sample program, the RL78/G1D Test Board and computer should be connected each other by the USB TTL serial cable, and you should enter commands to the sample program from the terminal emulator running on the computer.

Please setup the serial port of terminal emulator as shown below. In addition, the new-line code on receive for the terminal emulator is set to LF (LF only).

Table 5-2 UART port settings

Port Setting	Setting value
Baud rate	250,000 bps
Data length	8 bit

Port Setting	Setting value
Parity	None
Stop bit	1 bit
Flow control	None

Note that the BD Address of remote device is not required for the sample program in the embedded configuration, it does the device search automatically.

Figure 5-1 shows the screen shot of terminal setup window in the terminal emulator (Tera Term).

In the following, it is described in the screenshot when the EXE file is executed.

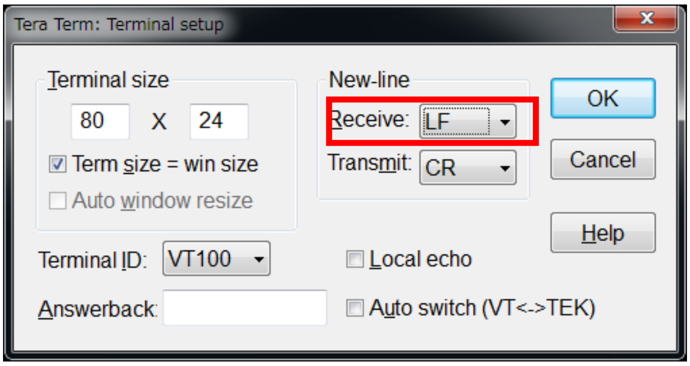


Figure 5-1 Terminal Setup window (Tera Term)

5.4. Usage of Console-based Sample Program

When you start the console-based sample program at the command prompt, Table 5-2 shows the main menu.

[Note] When the number of the implementation profile is changes, the command number may change.

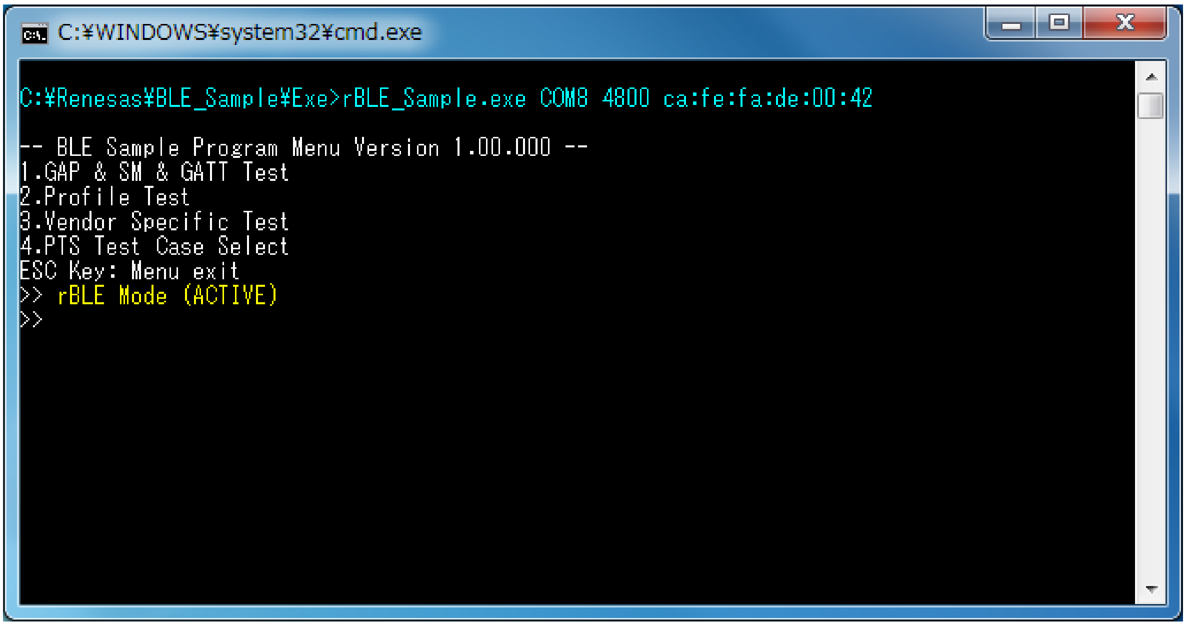


Figure 5-2 Sample Program Start Screen

Please confirm that the message "rBLE Mode (ACTIVE)" is displayed. If this message is not displayed, there is some problem and the sample program does not start successfully. Please check the cable connection or settings again.

The console-based sample program executes the operation which you may choose the menu item by its number. It shows the following main menu at the start time.

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

Figure 5-3 Main menu at the start time

At the main menu, there are three menu items. You can choose the menu item by its number.

In this screen, the menu items from 1 to 4 are displayed.
When you want to select the menu item, type its number and ENTER key.

When you want to go back to the previous menu, type ESC key.

When you want to see the current menu list again, type ENTER key.

When you want to exit the sample program, go back to the main menu by ESC key and enter ESC key again to terminate the sample program.

In addition, log output is displayed in different colors (using ANSI escape sequence).

The cyan notation means command execution (it called rBLE API), the yellow notation means event notification (its rBLE callback function is called), as shown in the following figure.

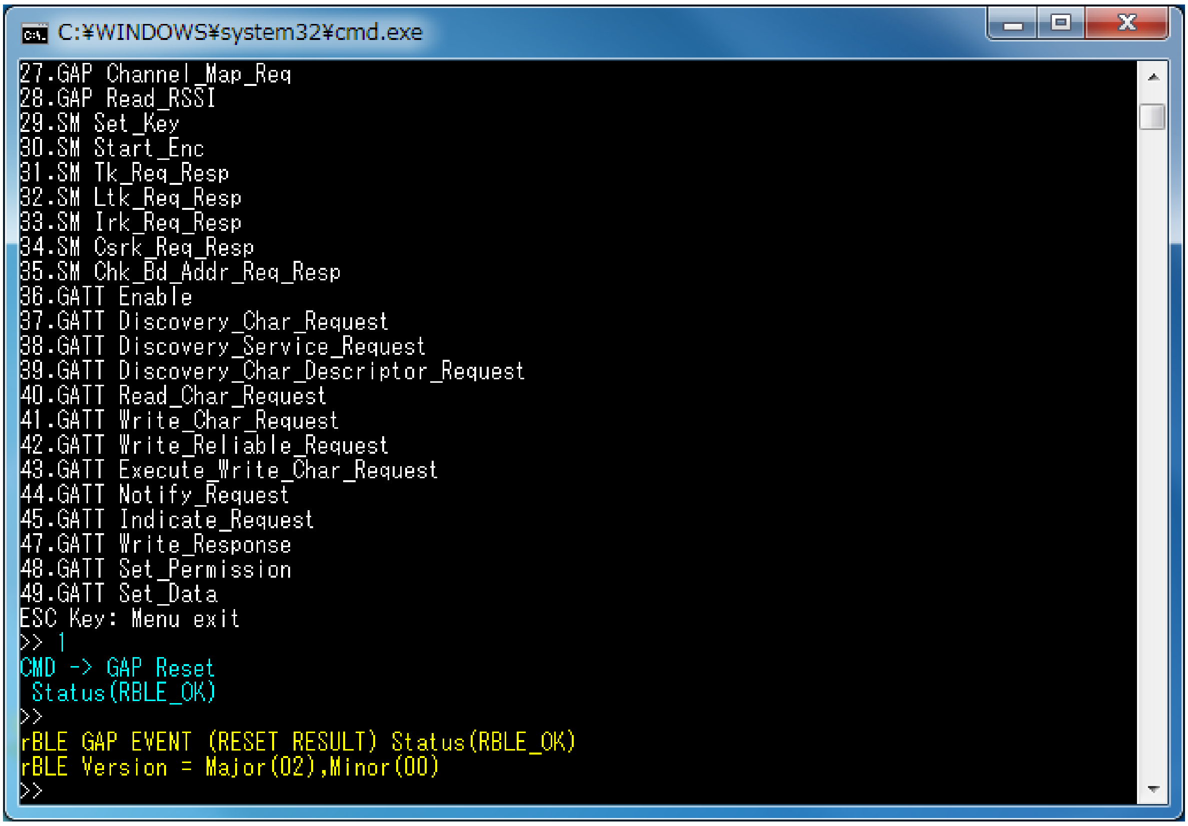


Figure 5-4 Execution example of RBLE_GAP_Reset function

In the following sections, basic usage of each layer is explained.

5.5. Generic Access Profile (GAP)

Commands and events for connecting device without security are shown in the following table as basic operations of the GAP. In addition, Figure 5-5 shows the log of the master device and Figure 5-6 shows the log of the slave device when you do the following operations in the table.

Operation	Master (Command & Event)	Slave (Command & Event)
Initialize	GAP Reset RESET_RESULT	GAP Reset RESET_RESULT
Send Advertising		GAP Broadcast_Enable BROADCAST_ENABLE_COMP
Search device (optional)	GAP Device_Search DEVICE_SEARCH_RESULT_IND DEVICE_SEARCH_COMP	
Establish connection	GAP Create_Connection CONNECTION_COMP	CONNECTION_COMP

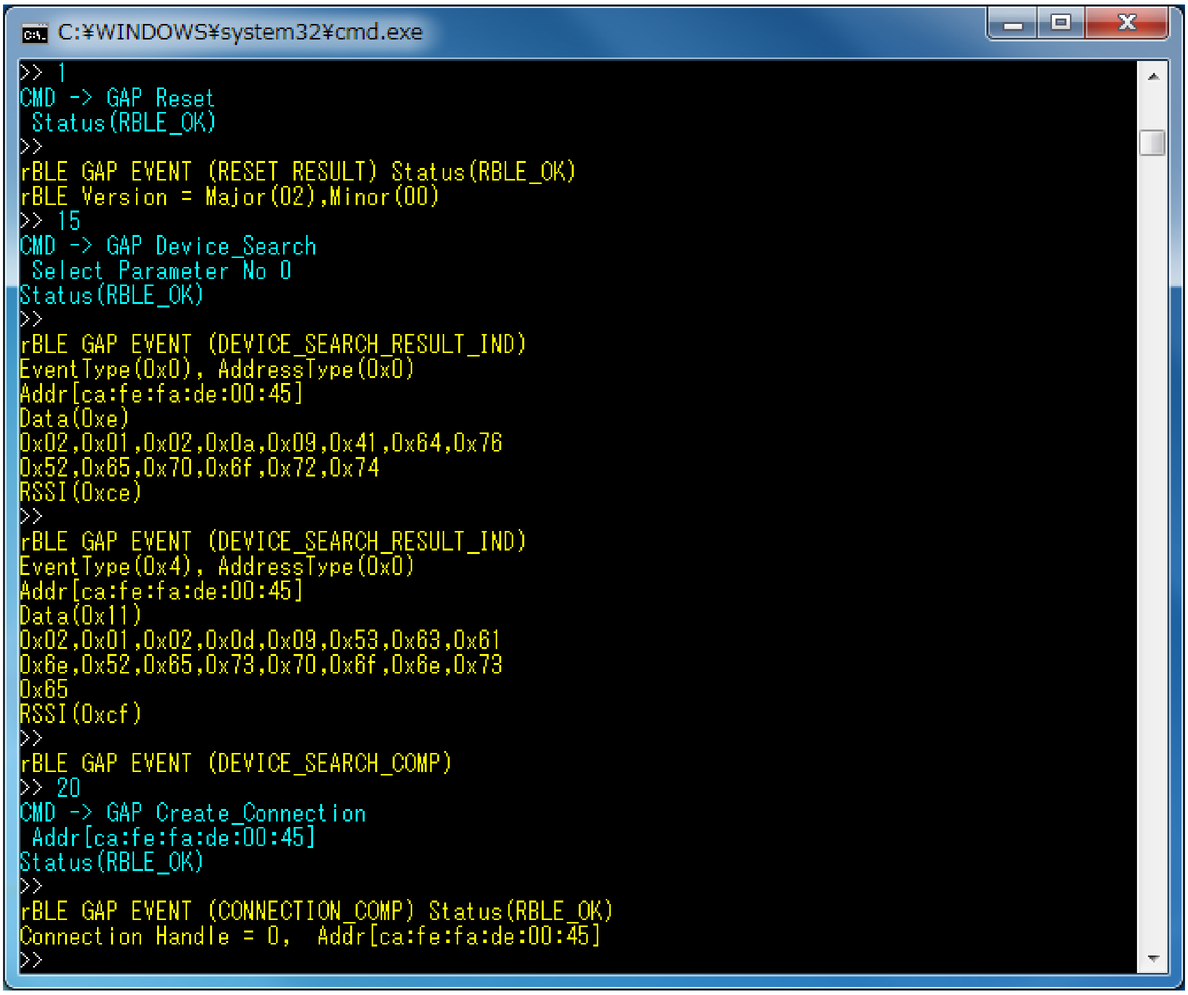


Figure 5-5 Log of Master Device (when connecting device without security)

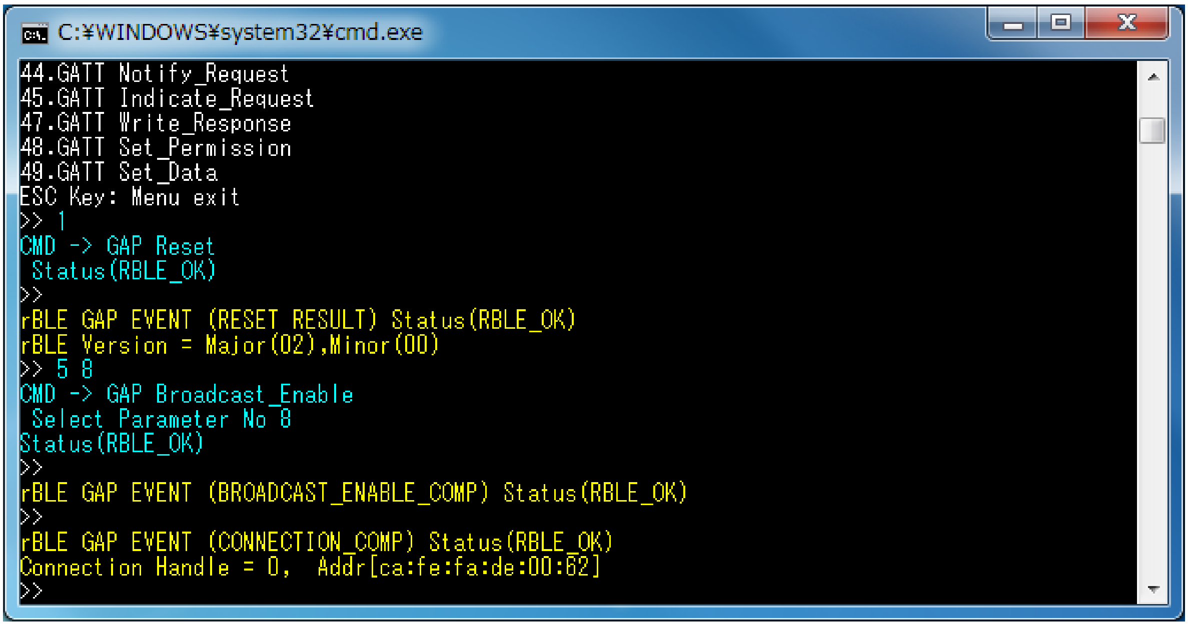


Figure 5-6 Log of Slave Device (when connecting device without security)

5.6. Security Manager (SM)

Commands and events for connecting device with security are shown in the following table as basic operations of the SM. In addition, Figure 5-7 and Figure 5-8 show the log of the master device and Figure 5-9 and Figure 5-10 show the log of the slave device when you do the following operations in the table. The device search operation is omitted in the each log.

Operation	Master (Command & Event)	Slave (Command & Event)
Initialize	GAP Reset	GAP Reset
	RESET_RESULT	RESET_RESULT
Set security	GAP Set_Security_Request	GAP Set_Security_Request
	SET_SECURITY_REQUEST_COMP	SET_SECURITY_REQUEST_COMP
	GAP_Set_Bonding_Mode	GAP_Set_Bonding_Mode
	SET_BONDING_MODE_COMP	SET_BONDING_MODE_COMP
Send Advertising		GAP Broadcast_Enable
		BROADCAST_ENABLE_COMP
Search device (optional)	GAP Device_Search	
	DEVICE_SEARCH_RESULT_IND	
	DEVICE_SEARCH_COMP	
Establish connection	GAP Create_Connection	
	CONNECTION_COMP	CONNECTION_COMP
Confirm device	BD_ADDR_REQ_IND	BD_ADDR_REQ_IND
	SM Chk_Bd_Addr_Req_Resp	SM Chk_Bd_Addr_Req_Resp
Start bonding	GAP Start_Bonding	
Bonding request and response		BONDING_REQ_IND
		GAP Bonding_Response
TK request and response	TK_REQ_IND	
	SM Tk_Req_Resp	
		TK_REQ_IND
		SM Tk_Req_Resp
LTK delivery		LTK_REQ_IND
		SM Ltk_Req_Resp
Key Indication	KEY_IND	KEY_IND
Bonding completion	BONDING_COMP	BONDING_COMP

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

Figure 5-7 Log of Master Device (when connecting device with security)

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

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

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

Figure 5-8 Log of Master device (when connecting device with security) (continued).

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

Figure 5-9 Log of Slave device (when connecting device with security)

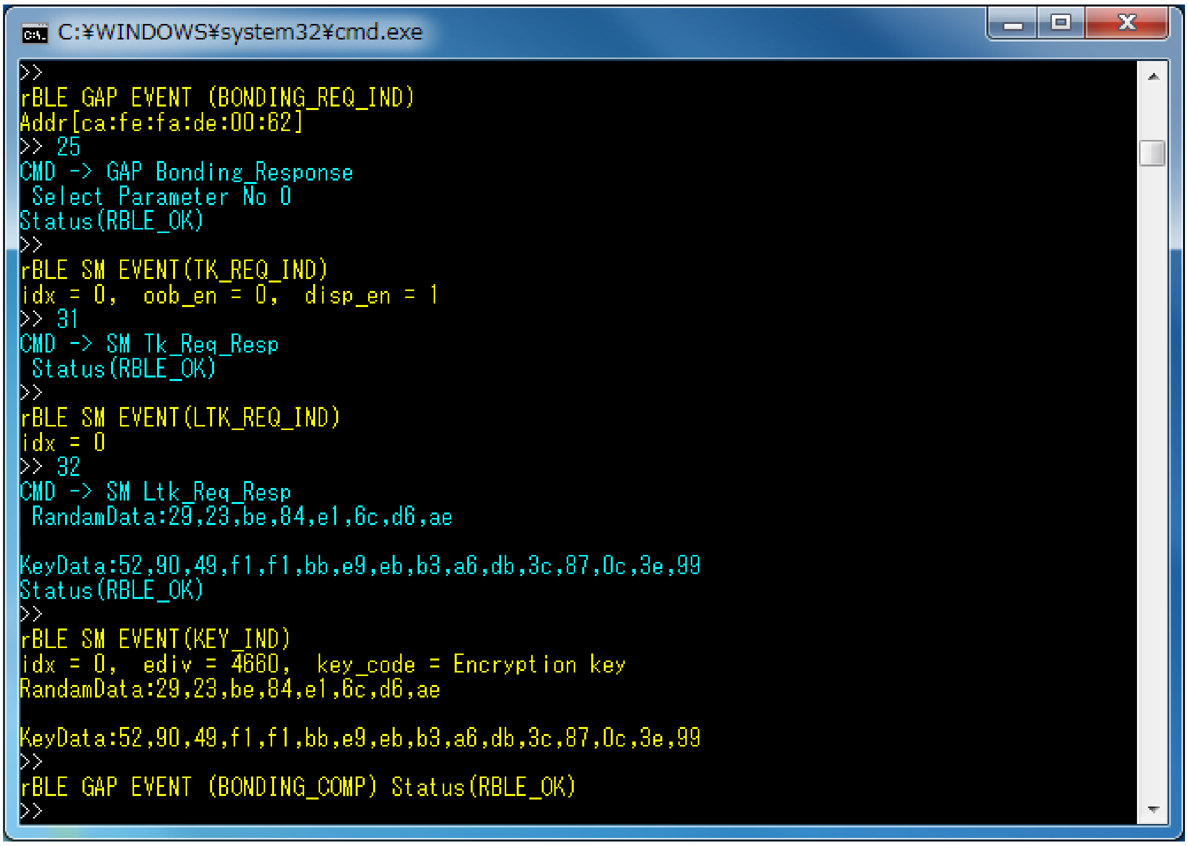


Figure 5-10 Log of Slave Device (when connecting device with security) (continued)

5.7. Generic Attribute Profile (GATT)

Commands and events for obtaining the characteristic handle grouped in service of remote device are shown in the following table as basic operations of the GATT. In addition, Figure 5-11 show the log of the Master device and Figure 5-12 shows the log of the Slave device when you do the following operations in the table.

Operation	Master (Command & Event)	Slave (Command & Event)
Connecting to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable GATT	GATT Enable	
Read characteritics	GATT Discovery_Char_Request DISC_CHAR_BY_UUID_CMP DISC_CHAR_BY_UUID_CMP COMPLETE	

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

Figure 5-11 Log of Master (Read Characteristic using GATT)

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

Figure 5-12 Log of Slave (Read Characteristic using GATT)

5.8. Find Me Profile (FMP)

Commands and events for writing alert level are shown in the following table as basic operations of the FMP. In addition, Figure 5-13 show the log of the Locator device and Figure 5-14 shows the log of the Target device when you do the following operations in the table.

Operations	Locator (Command & Event)	Target (Command & Event)
Connecting to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable target		FMP Target_Enable TARGET_ENABLE_COMP
Enable locator	FMP Locator_Enable LOCATOR_ENABLE_COMP	
Set alert	FMP Locator_Set_Alert	TARGET_ALERT_IND

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.

C:¥WINDOWS¥system32¥cmd.exe

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

   alert_char_hdl   = 0x0016
   alert_val_hdl    = 0x0017
   alert_char_prop  = 0x04

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

Figure 5-13 Log of FMP Locator

```
C:¥WINDOWS¥system32¥cmd.exe
>>
rBLE GAP EVENT (CONNECTION_COMP) Status(RBLE_OK)
Connection Handle = 0,  Addr[ca:fe:fa:de:00:62]
>> ↵
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 1
-- BLE Sample Program Find Me Profile Test Menu --
1.FMP Target_Enable
2.FMP Target_Disable
3.FMP Locator_Enable
4.FMP Locator_Disable
5.FMP Locator_Set_Alert
ESC Key: Menu exit
>> 1
CMD -> FMP Target_Enable
 Status(RBLE_OK)
>>
rBLE FMP EVENT (TARGET_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE FMP EVENT (TARGET_ALERT_IND)
Connection Handle = 0
alert_lvl = 2
>>
```

Figure 5-14 Log of FMP Target

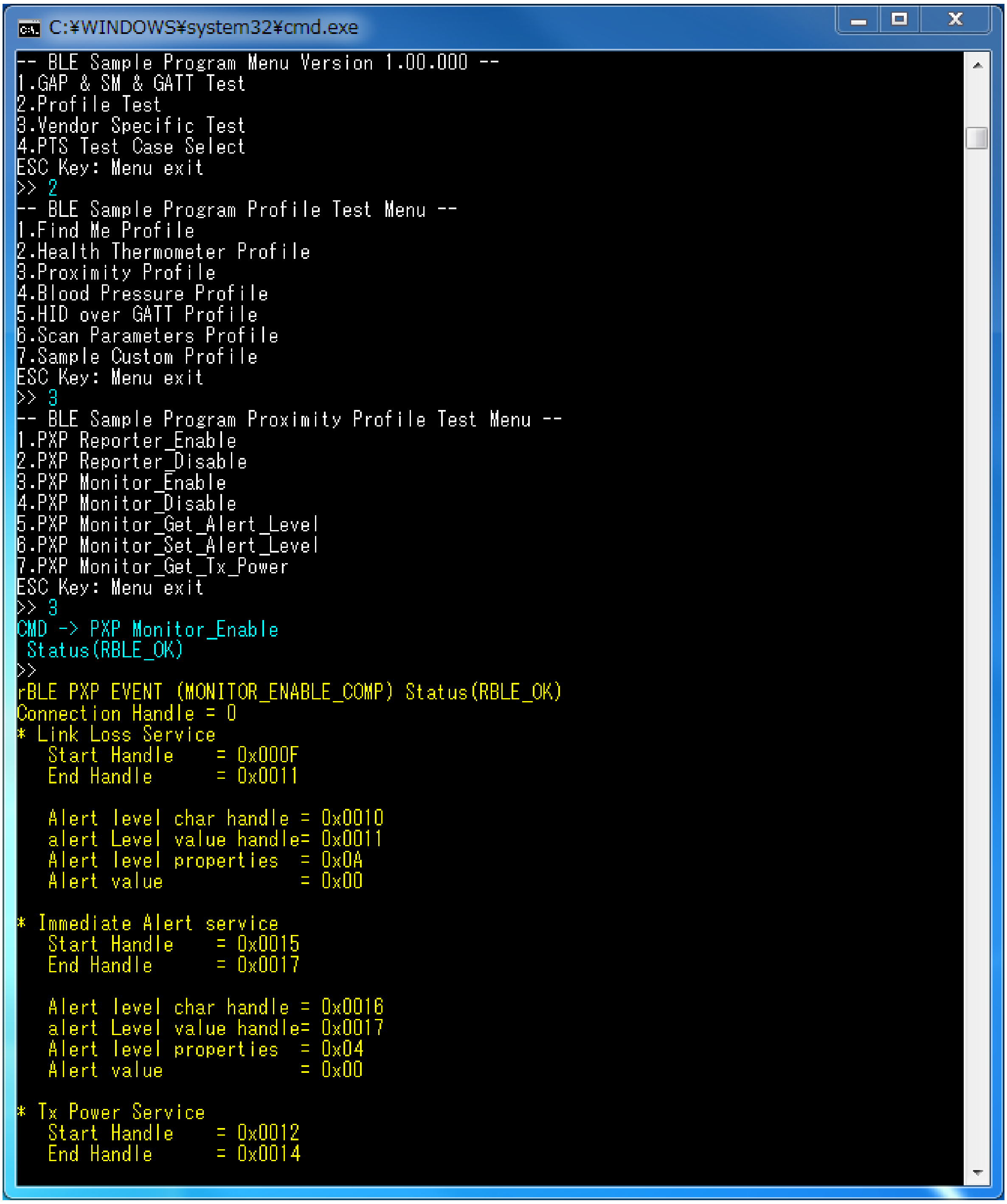
5.9. Proximity Profile (PXP)

Commands and events for reading and writing alert level are shown in the following table as basic operations of the PXP. In addition, Figure 5-15 and Figure 5-16 show the log of the Monitor device and Figure 5-17 shows the log of the Reporter device when you do the following operations in the table.

Operation	Monitor (Command & Event)	Reporter (Command & Event)
Connecting to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable reporter		PXP Reporter_Enable REPORTER_ENABLE_COMP
Enable monitor	PXP Monitor_Enable MONITOR_ENABLE_COMP	
Read alert level	PXP Monitor_Get_Alert_Level MONITOR_READ_CHAR_RESPONSE	
Write alert level	PXP Monitor_Set_Alert_Level MONITOR_WRITE_CHAR_RESPONSE	

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.



```
C:¥WINDOWS¥system32¥cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 3
-- BLE Sample Program Proximity Profile Test Menu --
1.PXP Reporter_Enable
2.PXP Reporter_Disable
3.PXP Monitor_Enable
4.PXP Monitor_Disable
5.PXP Monitor_Get_Alert_Level
6.PXP Monitor_Set_Alert_Level
7.PXP Monitor_Get_Tx_Power
ESC Key: Menu exit
>> 3
CMD -> PXP Monitor_Enable
 Status(RBLE_OK)
>>
rBLE PXP EVENT (MONITOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Link Loss Service
   Start Handle     = 0x000F
   End Handle       = 0x0011

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

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

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

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

Figure 5-15 Log of PXP Monitor

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

   txpw_lvl_char_hdl= 0x0013
   txpw_lvl_val_hdl = 0x0014
   txpw_lvl_cfg_hdl = 0x0000
   txpw_lvl_prop    = 0x02
   txpw_lvl         = 0x00

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

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

Figure 5-16 Log of PXP Monitor (continued)

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

Figure 5-17 Log of PXP Reporter

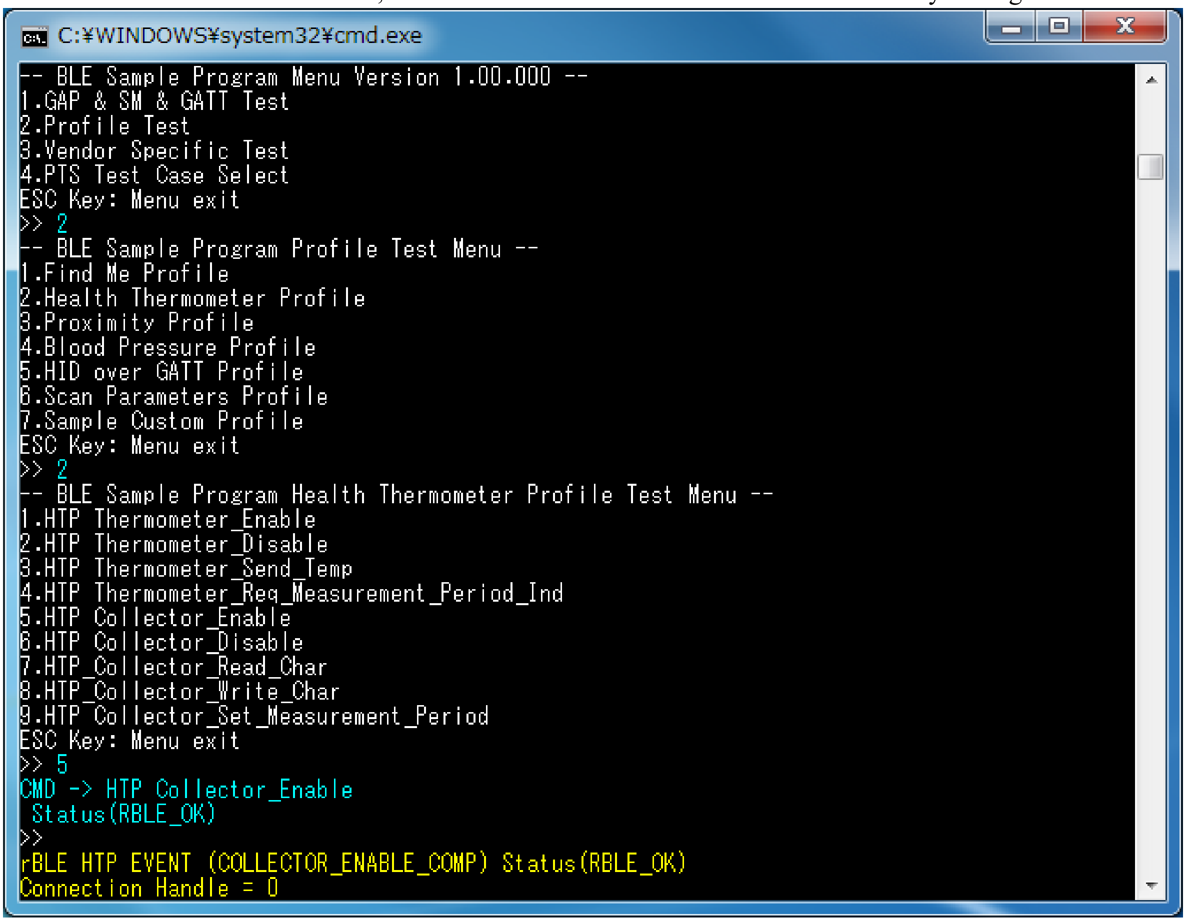
5.10. Health Thermometer Profile (HTP)

Commands and events for sending thermometer data are shown in the following table as basic operations of the HTP. In addition, Figure 5-18 and Figure 5-19 show the log of the Collector device and Figure 5-20 and Figure 5-21 shows the log of the Thermometer device when you do the following operations in the table.

Operation	Collector (Command & Event)	Thermometer (Command & Event)
Connecting to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable Thermometer		HTP Thermometer_Enable THERMOMETER_ENABLE_COMP
Enable Collector	HTP Collector_Enable COLLECTOR_ENABLE_COMP	
Enable Indication	HTP Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	THERMOMETER_CFG_INDNTF_IND
Transmit and receive thermometer data	COLLECTOR_TEMP_IND	HTP Thermometer_Send_Temp THERMOMETER_SEND_TEMP_COMP

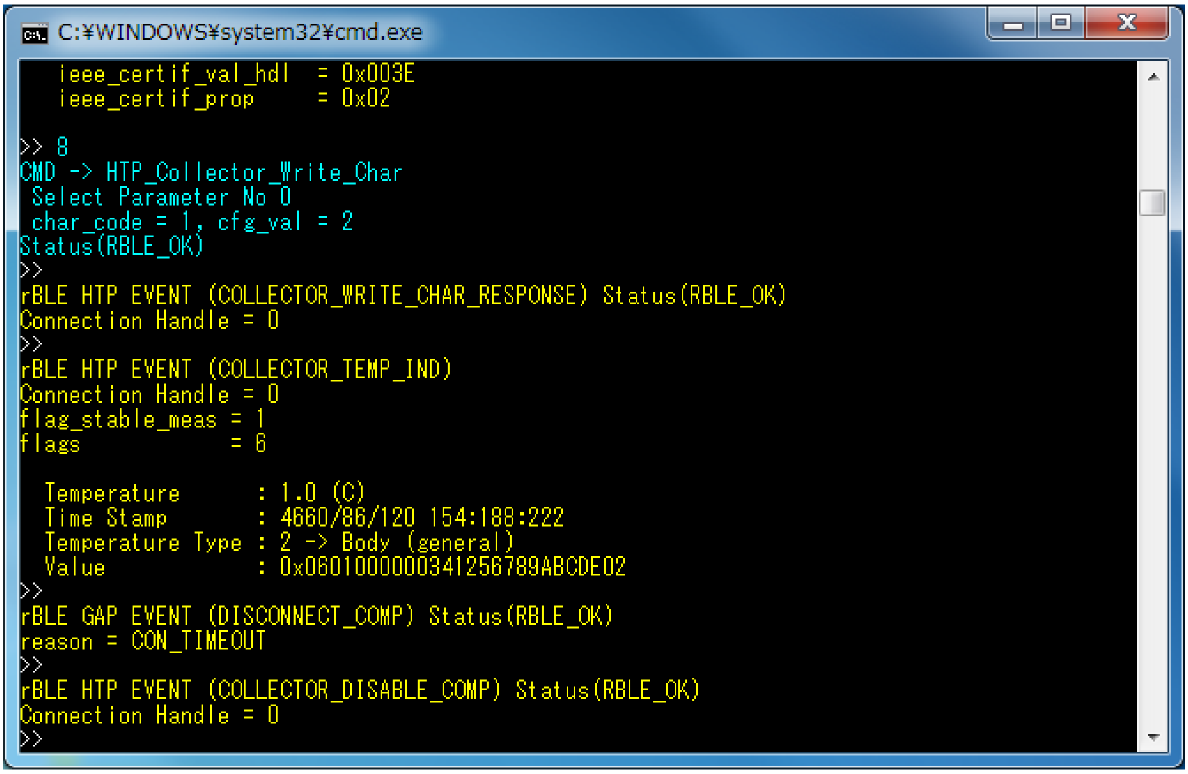
[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.



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

Figure 5-18 Log of HTP Collector



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

>> 8
CMD -> HTP_Collector_Write_Char
 Select Parameter No 0
 char_code = 1, cfg_val = 2
Status(RBLE_OK)
>>
rBLE HTP EVENT (COLLECTOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE HTP EVENT (COLLECTOR_TEMP_IND)
Connection Handle = 0
flag_stable_meas = 1
flags            = 6

  Temperature      : 1.0 (C)
  Time Stamp       : 4660/86/120 154:188:222
  Temperature Type : 2 -> Body (general)
  Value            : 0x06010000003412567 89ABCDE02
>>
rBLE GAP EVENT (DISCONNECT_COMP) Status(RBLE_OK)
reason = CON_TIMEOUT
>>
rBLE HTP EVENT (COLLECTOR_DISABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
```

Figure 5-19 Log of HTP Collector (continued)

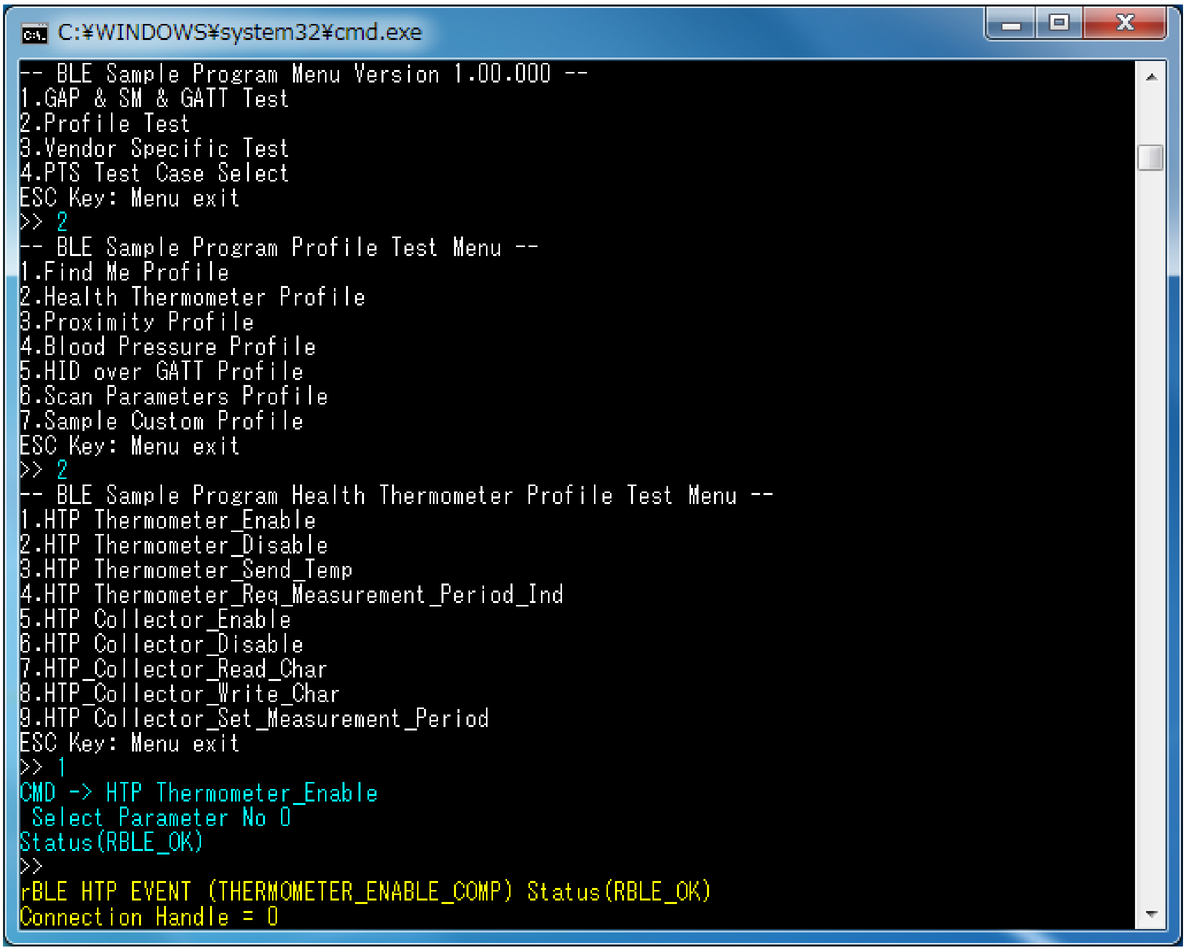


Figure 5-20 Log of HTP Thermometer

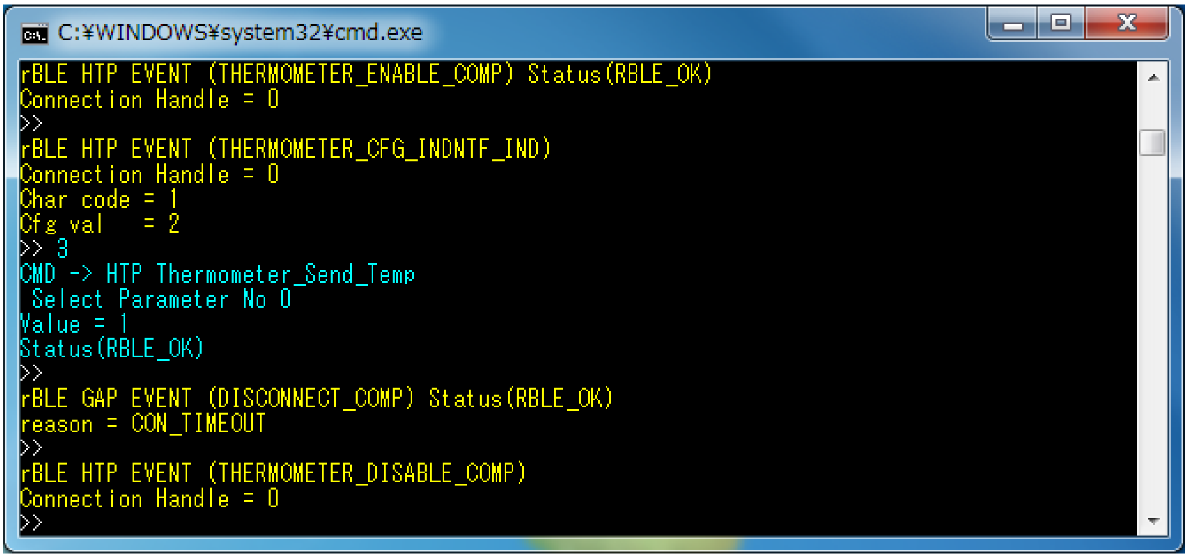


Figure 5-21 Log of HTP Thermometer (continued)

5.11. Blood Pressure Profile (BLP)

Commands and events for sending measurement data are shown in the following table as basic operations of the BLP. In addition, Figure 5-22 and Figure 5-23 show the log of the Collector device and Figure 5-24 shows the log of the Sensor device when you do the following operations in the table.

Operation	Collector (Command & Event)	Sensor (Command & Event)
Connecting to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable Sensor		BLP Sensor_Enable
		SENSOR_ENABLE_COMP
Enable Collector	BLP Collector_Enable	
	COLLECTOR_ENABLE_COMP	
Enable Indication	BLP Collector_Write_Char	
	COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_INDNTF_IND
Transmit and receive measurement data		BLP Sensor_Send_Measurements
	COLLECTOR_MEASUREMENTS_IND	SENSOR_SEND_MEASUREMENTS_COMP

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.

```
C:¥WINDOWS¥system32¥cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 4
-- BLE Sample Program Blood Pressure Profile Test Menu --
1.BLP Sensor_Enable
2.BLP Sensor_Disable
3.BLP Sensor_Send_Measurements
4.BLP Collector_Enable
5.BLP Collector_Disable
6.BLP Collector_Read_Char
7.BLP Collector_Write_Char
ESC Key: Menu exit
>> 4
CMD -> BLP Collector_Enable
 Status(RBLE_OK)
>>
rBLE BLP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Blood Pressure Service
   Start Handle = 0x0025
   End Handle   = 0x002D
```

Figure 5-22 Log of BLP Collector

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

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

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

Figure 5-23 Log of BLP Collector (continued)

```
C:¥WINDOWS¥system32¥cmd.exe
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Sample Custom Profile
ESC Key: Menu exit
>> 4
-- BLE Sample Program Blood Pressure Profile Test Menu --
1.BLP Sensor_Enable
2.BLP Sensor_Disable
3.BLP Sensor_Send_Measurements
4.BLP Collector_Enable
5.BLP Collector_Disable
6.BLP Collector_Read_Char
7.BLP Collector_Write_Char
ESC Key: Menu exit
>> 1
CMD -> BLP Sensor_Enable
 Status(RBLE_OK)
>>
rBLE BLP EVENT (SENSOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE BLP EVENT (SENSOR_CFG_INDNTF_IND)
  Char Code = BLDPRS_MEAS
  Cfg Value = START_NTF_IND
>> 3 1
CMD -> BLP Sensor_Send_Measurements
 Select Parameter -> Stable
Status(RBLE_OK)
>>
rBLE BLP EVENT (SENSOR_SEND_MEASUREMENTS_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
```

Figure 5-24 Log of BLP Sensor

5.12. HID over GATT Profile (HOGP)

Commands and events for transmitting the input report data are shown in the following table as basic operations of the HOGP. In addition, Figure 5-25 and Figure 5-26 shows the log of the Report Host device and Figure 5-27 and Figure 5-29 shows the log of the HID Device when you do the following operations in the table.

Operation	Report Host (Command & Event)	HID Device (Command & Event)
Connecting to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable HID device		HGP_HDevice_Enable HDEVICE_ENABLE_COMP
Enable report host	HGP_RHost_Enable RHOST_ENABLE_COMP	
Transmit and receive input report data	HGP_RHost_Set_Report RHOST_WRITE_CHAR_RESPONSE	HDEVICE_REPORT_IND

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.

About commands and events for profiles are described after connecting to the remote device.

To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.

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

Figure 5-25 Log of Report Host

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

Figure 5-26 Log of Report Host (continued)

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

Figure 5-27 Log of HID Device

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

Figure 5-28 Log of HID Device (continued)

5.13. Scan Parameters Profile (ScPP)

Commands and events for transmitting the scan interval window data are shown in the following table as basic operations of the ScPP. In addition, Figure 5-29 shows the log of the Scan Client device and Figure 5-30 shows the log of the Scan Server device when you do the following operations in the table.

Operation	Scan Client (Command & Event)	Scan Server (Command & Event)
Connect to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable server		SPP_Server_Enable SPPS_ENABLE_COMP
Enable client	SPP_Client_Enable SPPC_ENABLE_COMP	
Transmit and receive scan interval window data	SPP_Client_Write_Interval	SPPS_INTERVAL_WINDOW_CHG_EVT

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.

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

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

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

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

Figure 5-29 Log of Scan Client

C:¥WINDOWS¥system32¥cmd.exe

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

Figure 5-30 Log of Scan Server

5.14. Heart Rate Profile (HRP)

Commands and events for sending measurement data are shown in the following table as basic operations of the HRP. In addition, Figure 5-31 and Figure 5-32 show the log of the Collector device and Figure 5-33 shows the log of the Sensor device when you do the following operations in the table.

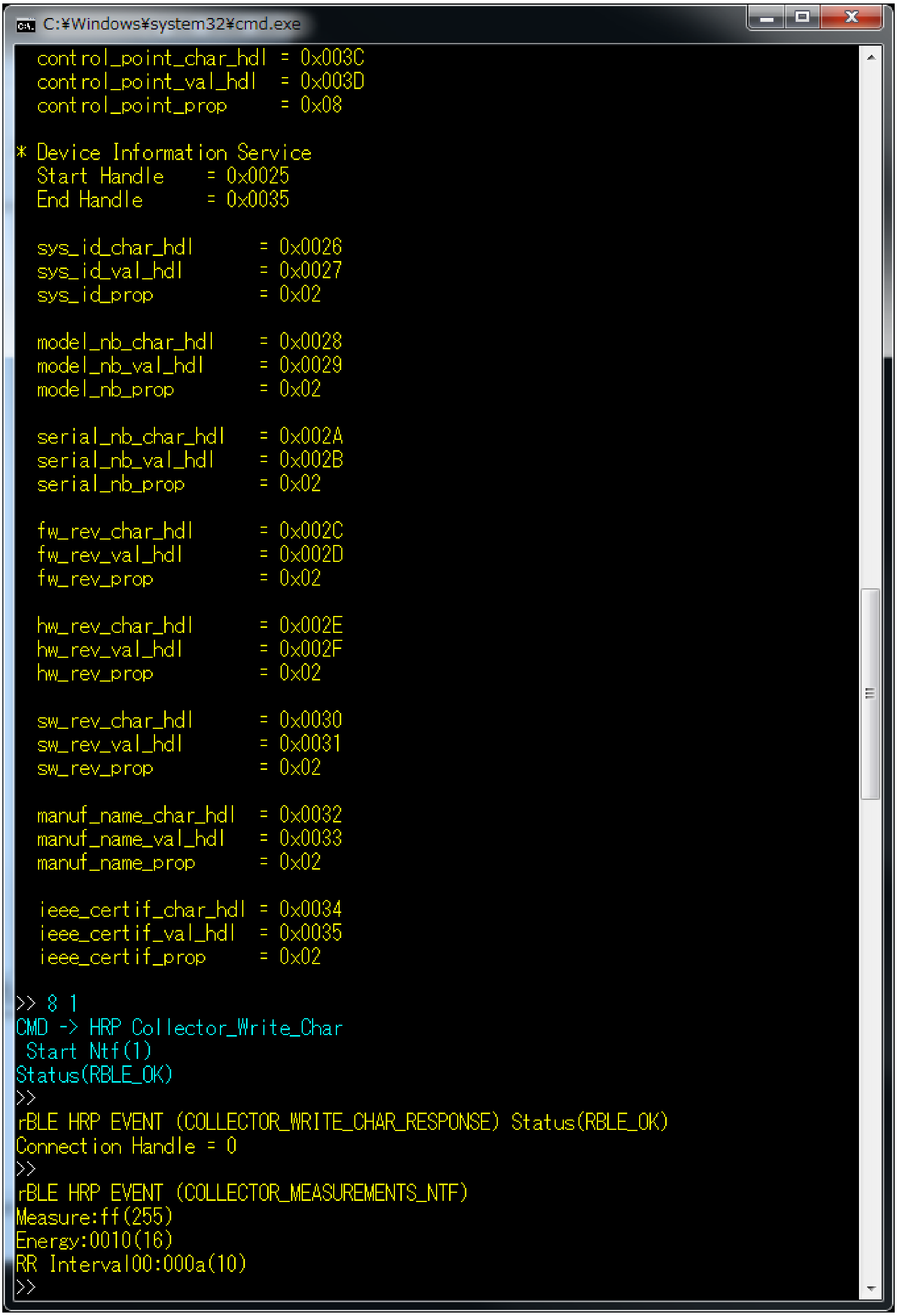
Operation	Heart Rate Collector (Command & Event)	Heart Rate Sensor (Command & Event)
Connect to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable Sensor		HRP Sensor_Enable SENSOR_ENABLE_COMP
Enable Collector	HRP Collector_Enable COLLECTOR_ENABLE_COMP	
Enable Indication	HRP Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_NTF_IND
Transmit and receive measurement data	COLLECTOR_MEASUREMENTS_NTF	HRP Sensor_Send_Measurements SENSOR_SEND_MEASUREMENTS_COMP

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.

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

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

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

Figure 5-31 Log of HRP Collector



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

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

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

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

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

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

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

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

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

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

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

Figure 5-32 Log of HRP Collector (continued)

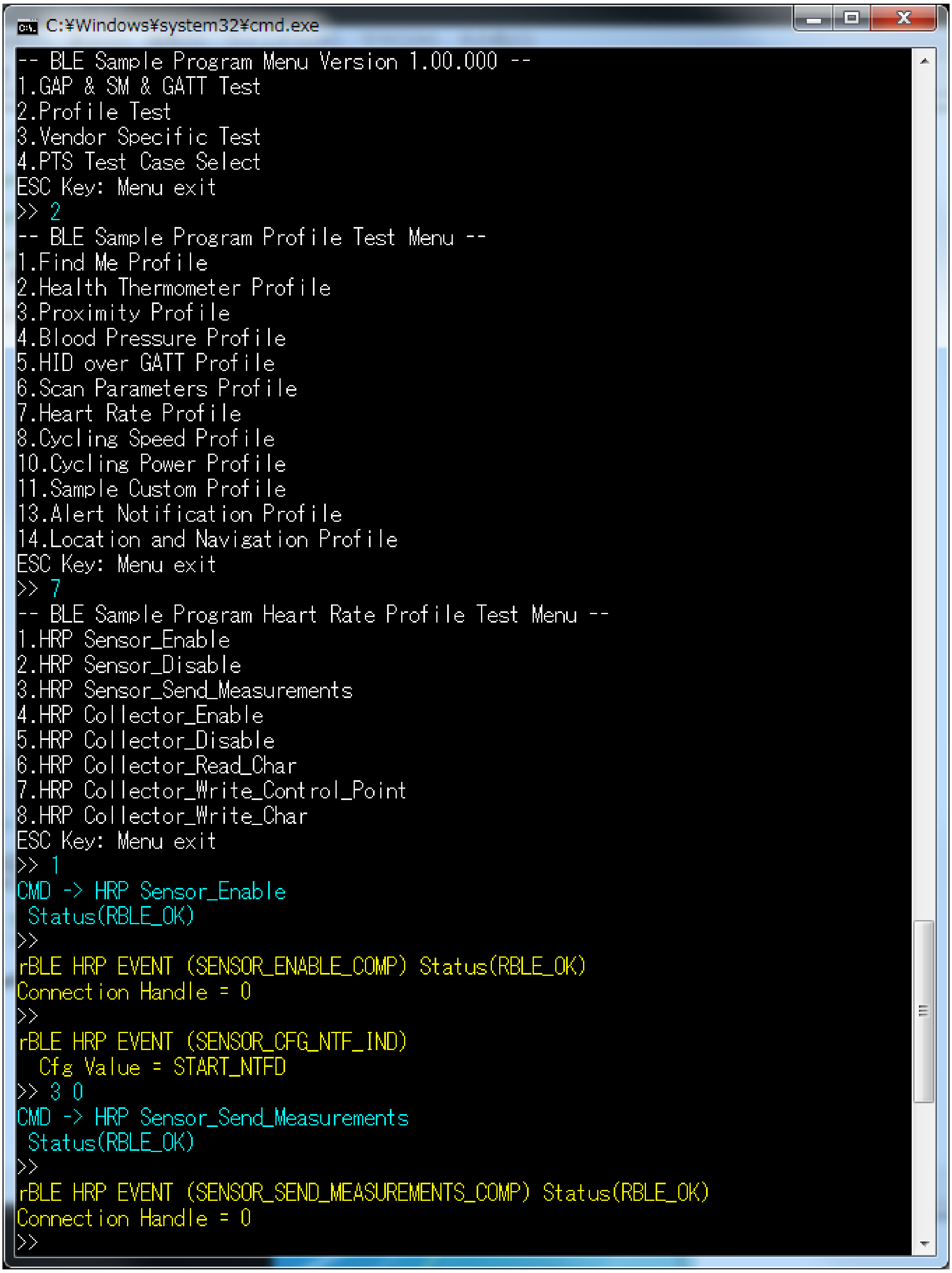


Figure 5-33 Log of HRP Sensor

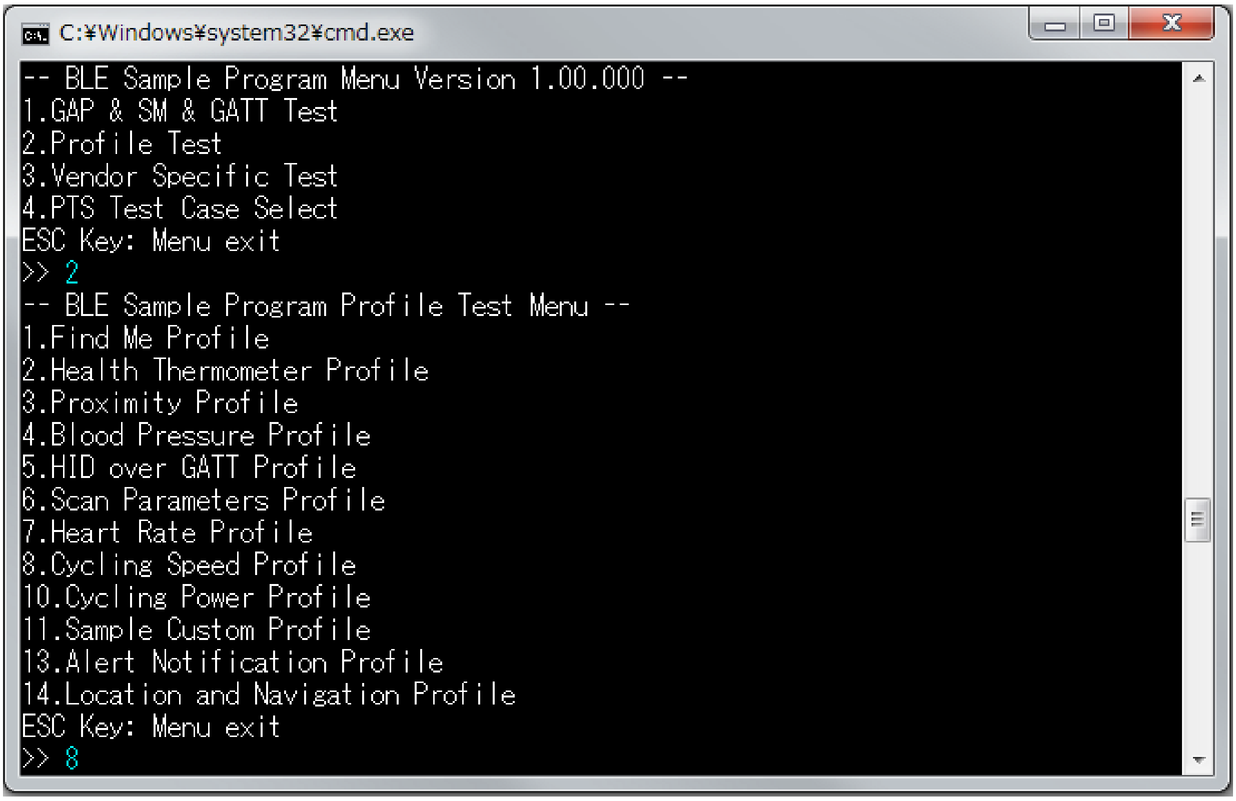
5.15. Cycling Speed and Cadence Profile (CSCP)

Commands and events for sending CSC measurement data are shown in the following table as basic operations of the CSCP. In addition, Figure 5-34 , Figure 5-35 and Figure 5-36 show the log of the Collector device and Figure 5-37 shows the log of the Sensor device when you do the following operations in the table.

Operation	Cycling Speed and Cadence Collector (Command & Event)	Cycling Speed and Cadence Sensor (Command & Event)
Connect to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable Sensor		CSCP Sensor_Enable SENSOR_ENABLE_COMP
Enable Collector	CSCP Collector_Enable COLLECTOR_ENABLE_COMP	
Enable Indication	CSCP Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_INDNTF_IND
Transmit and receive CSC measurement data	COLLECTOR_MEASUREMENTS_NTF	CSCP Sensor_Send_Measurements SENSOR_SEND_MEASUREMENTS_COMP

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.



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

Figure 5-34 Log of CSCP Collector

```
C:¥Windows¥system32¥cmd.exe
-- BLE Sample Program Cycling Speed and Cadence Profile Test Menu --
1.CSCP Sensor_Enable
2.CSCP Sensor_Disable
3.CSCP Sensor_Send_Measurements
4.CSCP Sensor_Send_Sc_Control_Point
5.CSCP Collector_Enable
6.CSCP Collector_Disable
7.CSCP Collector_Read_Char
8.CSCP Collector_Write_Sc_Control_Point
9.CSCP Collector_Write_Char
ESC Key: Menu exit
>> 5
CMD -> CSCP Collector_Enable
 Status(RBLE_OK)
>>
rBLE CSCP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Cycling Speed and Cadence Service
  Start Handle = 0x0025
  End Handle   = 0x002F

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

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

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

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

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

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

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

Figure 5-35 Log of CSCP Collector (continued -1)

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

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

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

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

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

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

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

Figure 5-36 Log of CSCP Collector (continued -2)

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

Figure 5-37 Log of CSCP Sensor

5.16. Cycling Power Profile (CPP)

Commands and events for sending Cycling Power measurement data are shown in the following table as basic operations of the CPP. In addition, Figure 5-38, Figure 5-39 and Figure 5-40 show the log of the Collector device and Figure 5-41 shows the log of the Sensor device when you do the following operations in the table.

Operation	Cycling Power Collector (Command & Event)	Cycling Power Sensor (Command & Event)
Connect to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable Sensor		CPP Sensor_Enable SENSOR_ENABLE_COMP
Enable Collector	CPP Collector_Enable COLLECTOR_ENABLE_COMP	
Enable Indication	CPP Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_INDNTFBRD_IND
Transmit and receive Cycling Power measurement data	COLLECTOR_MEASUREMENTS_NTF	CPP Sensor_Send_Measurements SENSOR_SEND_MEASUREMENTS_COMP

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.

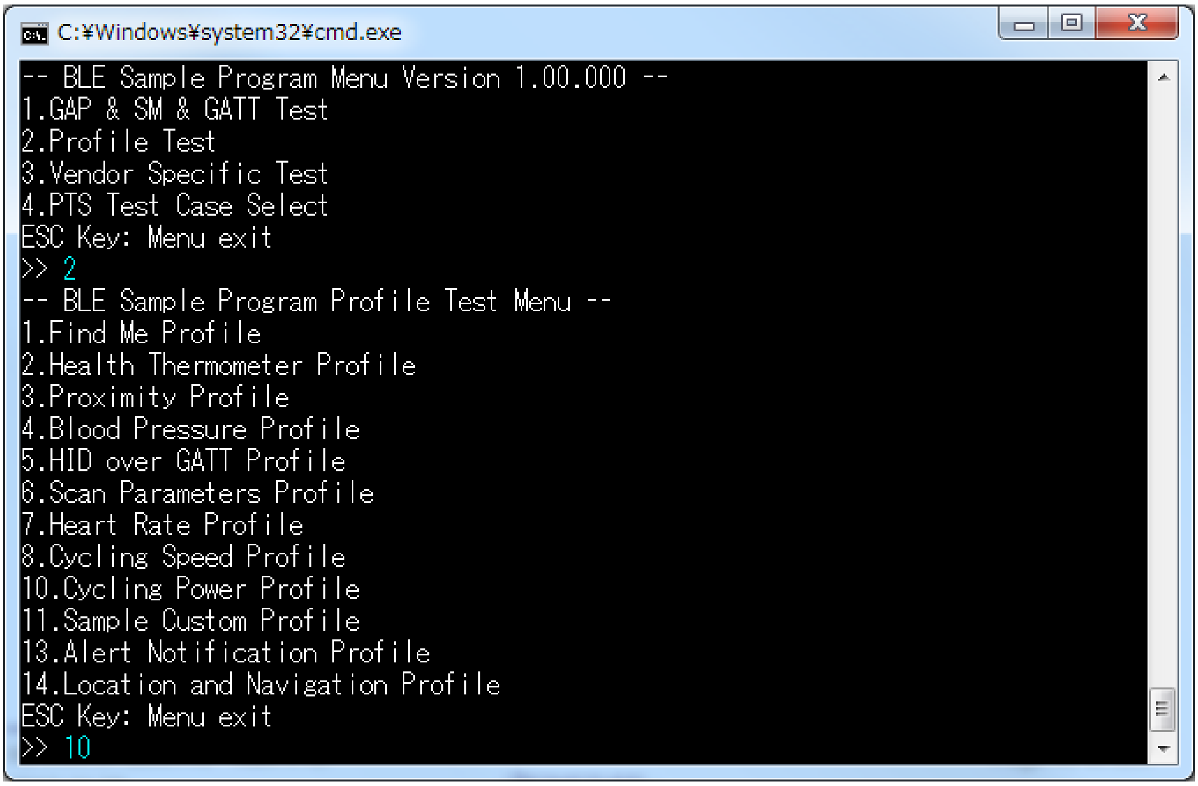


Figure 5-38 Log of CPP Collector

```
C:¥Windows¥system32¥cmd.exe
-- BLE Sample Program Cycling Power Profile Test Menu --
1.CPP Sensor_Enable
2.CPP Sensor_Disable
3.CPP Sensor_Send_Measurements
4.CPP Sensor_Broadcast_Measurements
5.CPP Sensor_Send_Vector
6.CPP Sensor_Send_CP_Control_Point
7.CPP Sensor_Send_Battery_Level
8.CPP Sensor_Send_Write_Response
9.CPP Collector_Enable
10.CPP Collector_Disable
11.CPP Collector_Read_Char
12.CPP Collector_Write_CP_Control_Point
13.CPP Collector_Write_Char
ESC Key: Menu exit
>> 9
CMD -> CPP Collector_Enable
 Status(RBLE_OK)
>>
rBLE CPP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
* Cycling Power Service
   Start Handle = 0x0030
   End Handle   = 0x003E

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

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

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

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

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

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

Figure 5-39 Log of CPP Collector (continued -1)

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

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

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

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

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

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

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

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

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

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

Figure 5-40 Log of CPP Collector (continued -2)

```
C:¥Windows¥system32¥cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 10
-- BLE Sample Program Cycling Power Profile Test Menu --
1.CPP Sensor_Enable
2.CPP Sensor_Disable
3.CPP Sensor_Send_Measurements
4.CPP Sensor_Broadcast_Measurements
5.CPP Sensor_Send_Vector
6.CPP Sensor_Send_CP_Control_Point
7.CPP Sensor_Send_Battery_Level
8.CPP Sensor_Send_Write_Response
9.CPP Collector_Enable
10.CPP Collector_Disable
11.CPP Collector_Read_Char
12.CPP Collector_Write_CP_Control_Point
13.CPP Collector_Write_Char
ESC Key: Menu exit
>> 1
CMD -> CPP Sensor_Enable
 Status(RBLE_OK)
>>
rBLE CPP EVENT (SENSOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE CPP EVENT (SENSOR_CFG_INDNTFBRD_IND)
  Char Code = 1
  Cfg Value = START
>> 3 0 0x3
CMD -> CPP Sensor_Send_Measurements
 Status(RBLE_OK)
>>
rBLE CPP EVENT (SENSOR_SEND_MEASUREMENTS_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
```

Figure 5-41 Log of CPP Sensor

5.17. Alert Notification Profile (ANP)

Commands and events for sending New Alert data are shown in the following table as basic operations of the ANP. In addition, Figure 5-42 and Figure 5-43 show the log of the Client device and Figure 5-44 shows the log of the Server device when you do the following operations in the table.

Operation	Alert Notification Client (Command & Event)	Alert Notification Server (Command & Event)
Connect to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable Sensor		ANP Server_Enable SERVER_ENABLE_COMP
Enable Collector	ANP Client_Enable CLIENT_ENABLE_COMP	
Enable Indication	ANP Client_Write_Char CLIENT_WRITE_CHAR_RESPONSE	SERVER_CFG_NTF_IND
Transmit and receive New Alert data	CLIENT_NEW_ALERT_NTF	ANP Sensor_Send_New_Alert SERVER_SEND_NEW_ALERT_COMP

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.

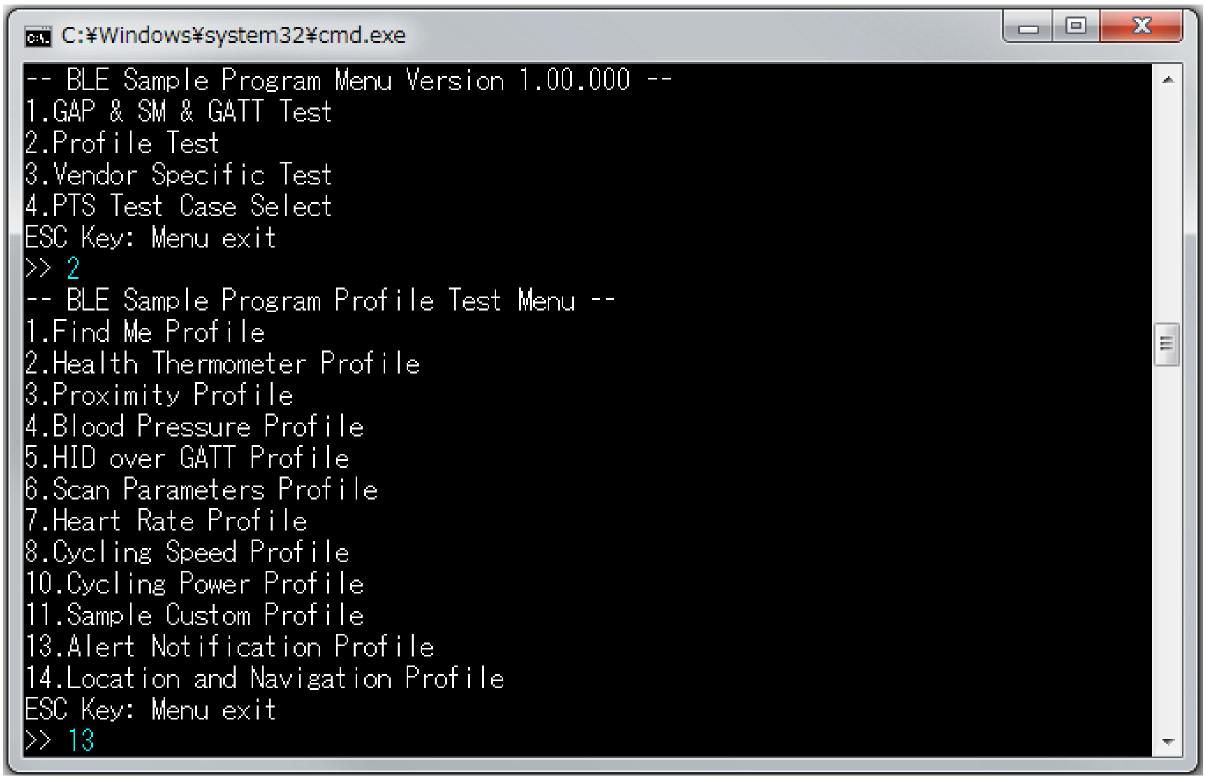


Figure 5-42 Log of ANP Client

C:¥Windows¥system32¥cmd.exe

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

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

Figure 5-43 Log of ANP Client (continued)

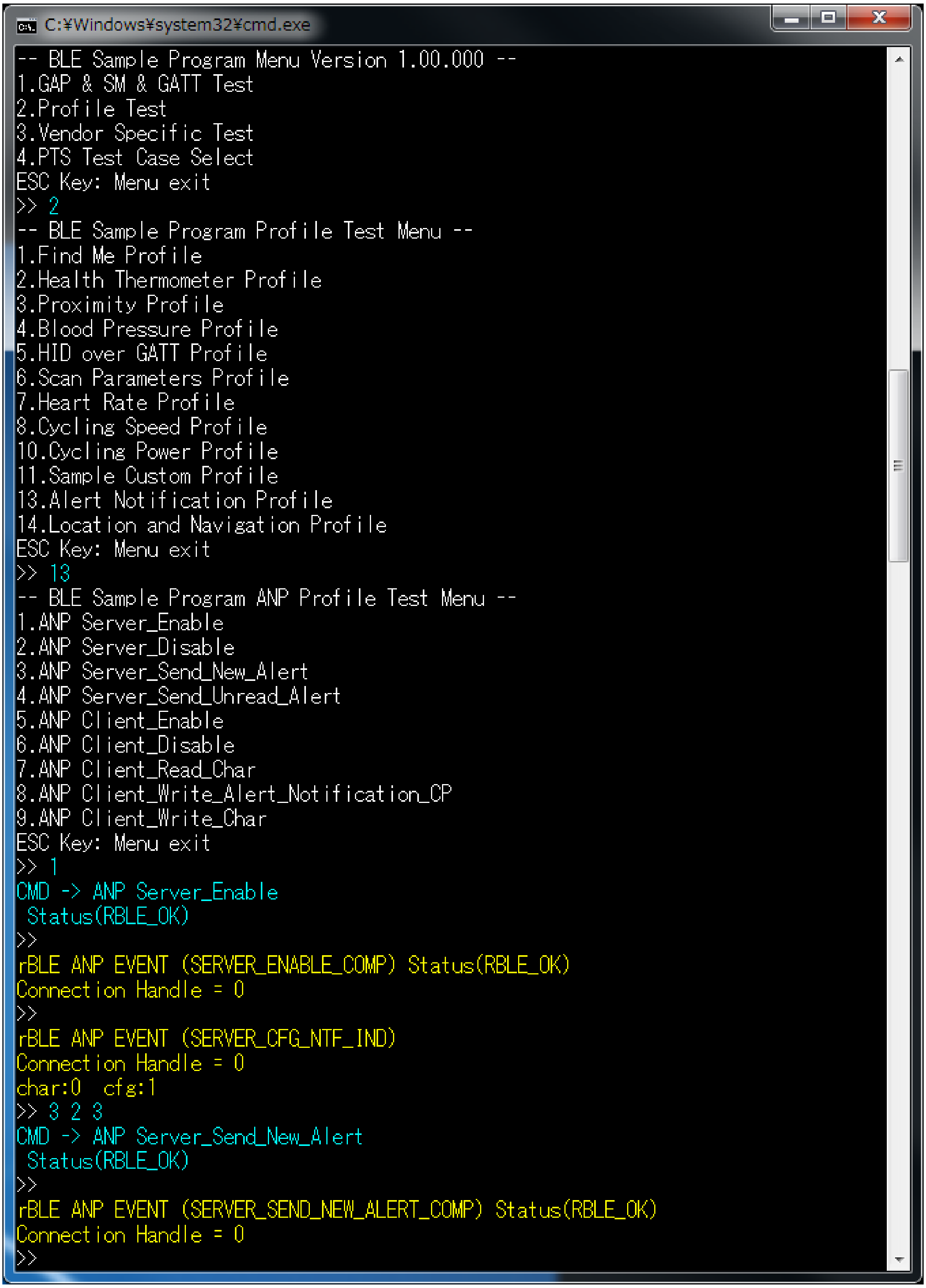


Figure 5-44 Log of ANP Server

5.18. Location and Navigation Profile (LNP)

Commands and events for sending Location Speed data are shown in the following table as basic operations of the LNP. In addition, Figure 5-45, Figure 5-46 and Figure 5-47 show the log of the Collector device and Figure 5-48 shows the log of the Sensor device when you do the following operations in the table.

Operation	Location and Navigation Collector (Command & Event)	Location and Navigation Sensor (Command & Event)
Connect to the remote device	Refer to 5.5 Generic Access Profile (GAP) and 5.6 Security Manager (SM)	
Enable Sensor		LNP Sensor_Enable SENSOR_ENABLE_COMP
Enable Collector	LNP Collector_Enable COLLECTOR_ENABLE_COMP	
Enable Indication	LNP Collector_Write_Char COLLECTOR_WRITE_CHAR_RESPONSE	SENSOR_CFG_INDNTF_IND
Transmit and receive Location Speed data	COLLECTOR_LOCATION_SPEED_NTF	LNP Sensor_Send_Location_Speed SENSOR_SEND_LOCATION_SPEED_COMP

[Note]

All profiles are connected to the remote device using GAP and SM commands, and use the handle that has been notified at the time of connection.
About commands and events for profiles are described after connecting to the remote device.
To connect to the remote device, refer to 5.5 Generic Access Profile and 5.6 Security Manager.

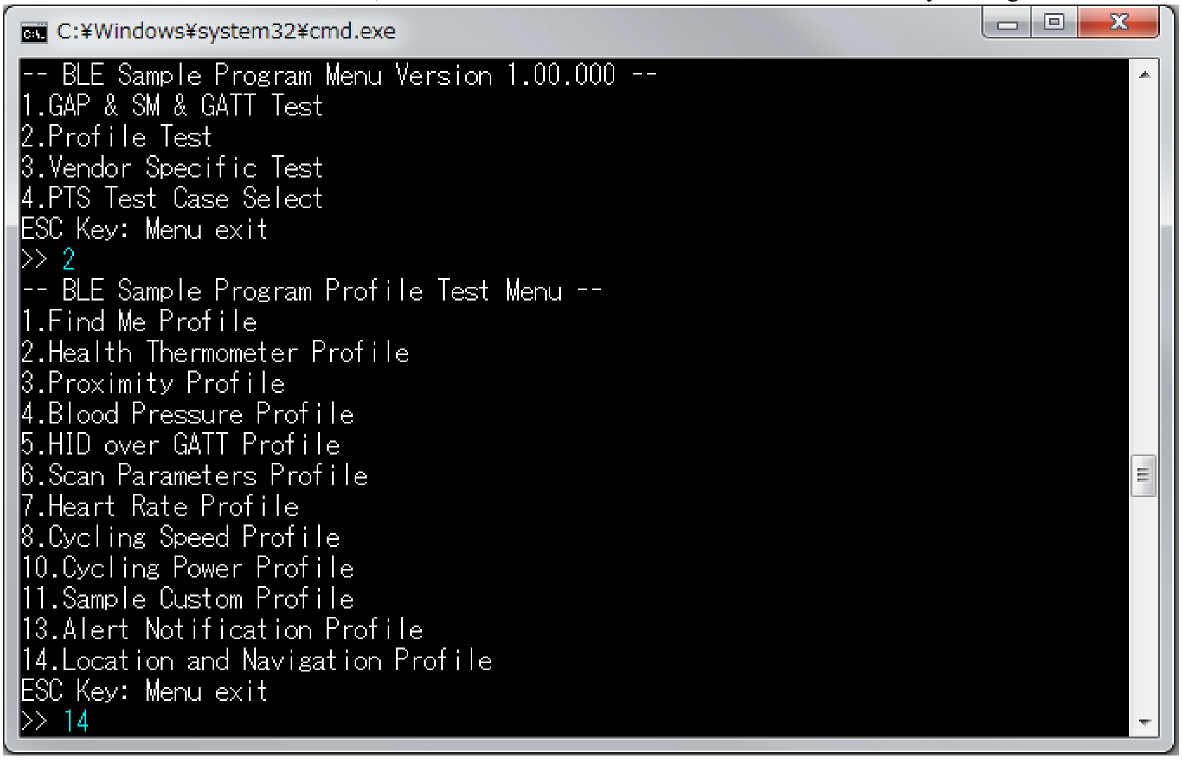


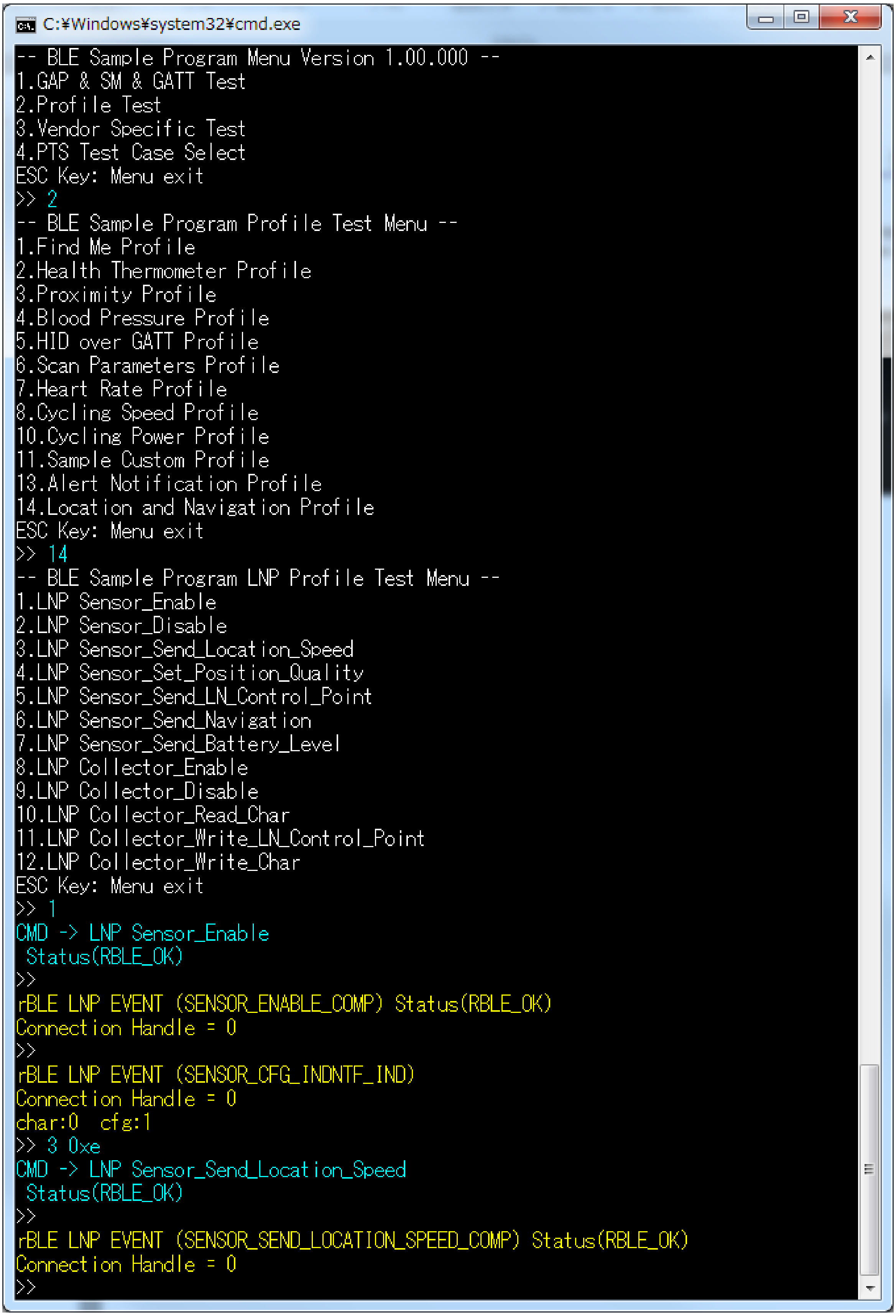
Figure 5-45 Log of LNP Collector

```
C:¥Windows¥system32¥cmd.exe
-- BLE Sample Program LNP Profile Test Menu --
1.LNP Sensor_Enable
2.LNP Sensor_Disable
3.LNP Sensor_Send_Location_Speed
4.LNP Sensor_Set_Position_Quality
5.LNP Sensor_Send_LN_Control_Point
6.LNP Sensor_Send_Navigation
7.LNP Sensor_Send_Battery_Level
8.LNP Collector_Enable
9.LNP Collector_Disable
10.LNP Collector_Read_Char
11.LNP Collector_Write_LN_Control_Point
12.LNP Collector_Write_Char
ESC Key: Menu exit
>> 8
CMD -> LNP Collector_Enable
 Status(RBLE_OK)
>>
rBLE LNP EVENT (COLLECTOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0

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

Figure 5-46 Log of LNP Collector (continued -1)

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

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

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

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

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

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

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

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

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

   battery_lvl_char_hdl = 0x0021
   battery_lvl_val_hdl  = 0x0022
   battery_lvl_cfg_hdl  = 0x0023
   battery_lvl_prop     = 0x12
>> 12 0 1
CMD -> LNP Collector_Write_Char
 Status(RBLE_OK)
>>
rBLE LNP EVENT (COLLECTOR_WRITE_CHAR_RESPONSE) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE LNP EVENT (COLLECTOR_LOCATION_SPEED_NTF)
Connection Handle = 0
flags:0x000e
total_distance:200000(0x00030d40)
latitude       :1050000000(0x3e95ba80)
longitude      :1430000000(0x553c1180)
elevation      :-59(0xffffffc5)
>>
```

Figure 5-47 Log of LNP Collector (continued -2)



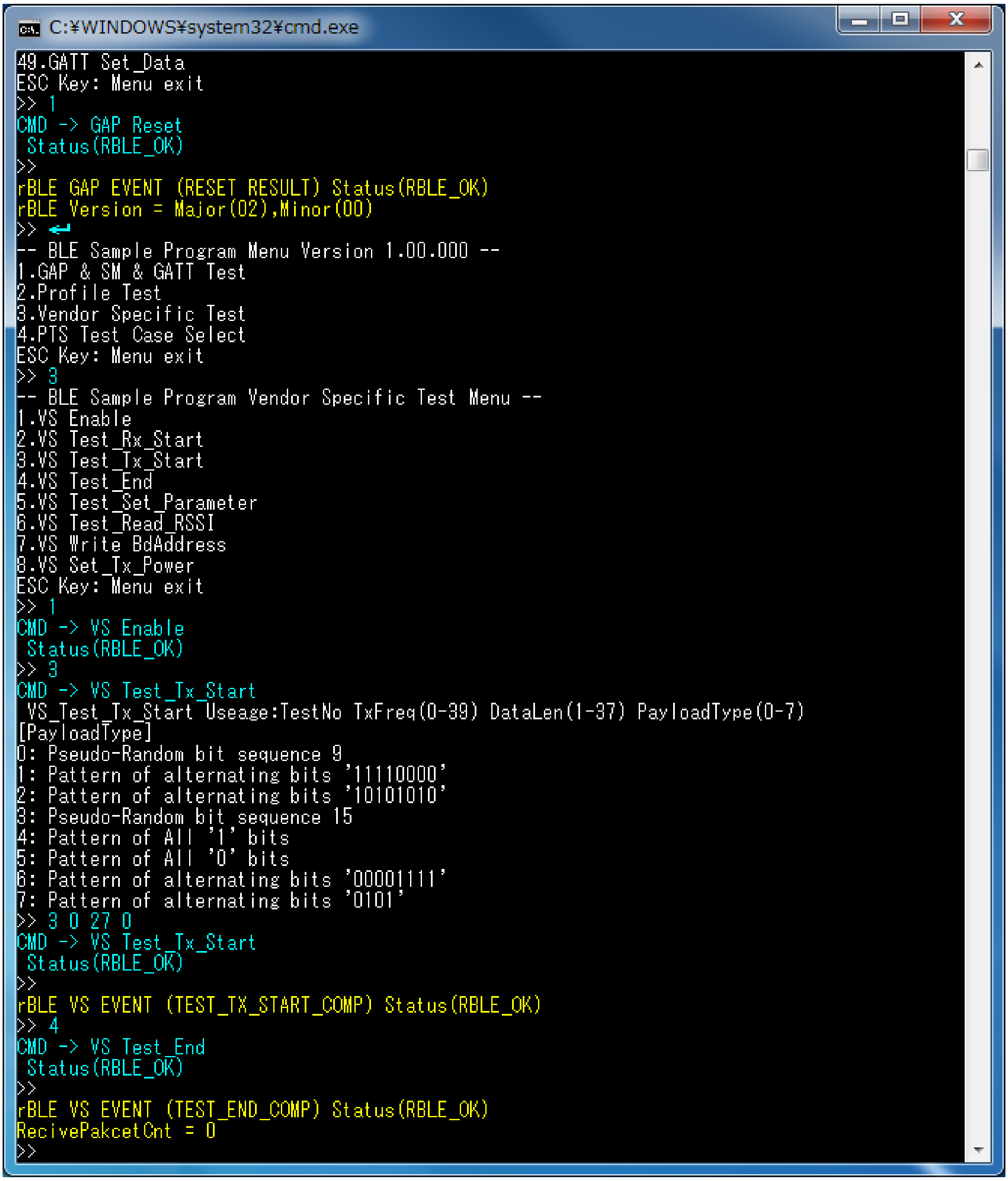
```
C:¥Windows¥system32¥cmd.exe
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 2
-- BLE Sample Program Profile Test Menu --
1.Find Me Profile
2.Health Thermometer Profile
3.Proximity Profile
4.Blood Pressure Profile
5.HID over GATT Profile
6.Scan Parameters Profile
7.Heart Rate Profile
8.Cycling Speed Profile
10.Cycling Power Profile
11.Sample Custom Profile
13.Alert Notification Profile
14.Location and Navigation Profile
ESC Key: Menu exit
>> 14
-- BLE Sample Program LNP Profile Test Menu --
1.LNP Sensor_Enable
2.LNP Sensor_Disable
3.LNP Sensor_Send_Location_Speed
4.LNP Sensor_Set_Position_Quality
5.LNP Sensor_Send_LN_Control_Point
6.LNP Sensor_Send_Navigation
7.LNP Sensor_Send_Battery_Level
8.LNP Collector_Enable
9.LNP Collector_Disable
10.LNP Collector_Read_Char
11.LNP Collector_Write_LN_Control_Point
12.LNP Collector_Write_Char
ESC Key: Menu exit
>> 1
CMD -> LNP Sensor_Enable
 Status(RBLE_OK)
>>
rBLE LNP EVENT (SENSOR_ENABLE_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
rBLE LNP EVENT (SENSOR_CFG_INDNTF_IND)
Connection Handle = 0
char:0  cfg:1
>> 3 0xe
CMD -> LNP Sensor_Send_Location_Speed
 Status(RBLE_OK)
>>
rBLE LNP EVENT (SENSOR_SEND_LOCATION_SPEED_COMP) Status(RBLE_OK)
Connection Handle = 0
>>
```

Figure 5-48 Log of LNP Sensor

5.19. Vendor Specific (VS)

Commands and events for using the Direct Test Mode are shown in the following table as basic operations of the VS. In addition, Figure 5-49 shows the log of the transmitter device and Figure 5-50 shows the log of the receiver device when you do the following operations in the table.

Operation	Transmitter (Command & Event)	Receiver (Command & Event)
Enable VS	VS Enable	VS Enable
Test start	VS Test_Tx_Start	VS Test_Rx_Start
	TEST_TX_START_COMP	TEST_RX_START_COMP
Test end	VS Test_End	VS Test_End
	TEST_END_COMP	TEST_END_COMP



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

Figure 5-49 Log of Direct Test Mode (Transmitter)

```
C:¥WINDOWS¥system32¥cmd.exe
40.GATT Read_Char_Request
41.GATT Write_Char_Request
42.GATT Write_Reliable_Request
43.GATT Execute_Write_Char_Request
44.GATT Notify_Request
45.GATT Indicate_Request
47.GATT Write_Response
48.GATT Set_Permission
49.GATT Set_Data
ESC Key: Menu exit
>> 1
CMD -> GAP Reset
 Status(RBLE_OK)
>>
rBLE GAP EVENT (RESET RESULT) Status(RBLE_OK)
rBLE Version = Major(02),Minor(00)
>> ↵
-- BLE Sample Program Menu Version 1.00.000 --
1.GAP & SM & GATT Test
2.Profile Test
3.Vendor Specific Test
4.PTS Test Case Select
ESC Key: Menu exit
>> 3
-- BLE Sample Program Vendor Specific Test Menu --
1.VS Enable
2.VS Test_Rx_Start
3.VS Test_Tx_Start
4.VS Test_End
5.VS Test_Set_Parameter
6.VS Test_Read_RSSI
7.VS Write BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
 Status(RBLE_OK)
>> 2
CMD -> VS Test_Rx_Start
 VS_Test_Rx_Start Useage:TestNo RxFreq(0-39)
>> 2 0
CMD -> VS Test_Rx_Start
 Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLE_OK)
>> 4
CMD -> VS Test_End
 Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
RecivePakcetCnt = 2745
>>
```

Figure 5-50 Log of Direct Test Mode (Receiver)

6. Usage of Simple Sample Program

This simple sample program shows how to use the BLE software. Contrary to the Sample Program, this simple sample program includes small functions, thus you can understand the behavior and the implementation easily.

This simple sample program is identical with "Embedded Configuration Sample Application (r01an3319)" Peripheral role sample application. Refer "Embedded Configuration Sample Application (r01an3319)" application note for the detail.

6.1. Configuration

This simple sample program works with embedded configuration only. Not works with modem configuration.

6.2. HEX File Preparation

There are two methods to prepare HEX file.

First one is to use the pre-built HEX file. The pre-built HEX files are located in /Renesas/BLE_Software_Ver_X_XX/RL78_G1D/ROM_File. You can find the pre-built HEX files compiled with each supported compiler (CC-RL, IAR, CA78K0R).

Second one is to build the HEX file from source codes. The project files are located in /Renesas/BLE_Software_Ver_X_XX/RL78_G1D/renesas/tools/simple_sample. You can find the project files for each supported development environment (e^2 studio, CS+, IAR Embedded Workbench).

6.3. Behavior

After writing the HEX file prepared in Section 6.2 onto RL78/G1D Test Board, reset the board by pressing the reset button. After the reset, make sure LED1 and LED2 on the board start blinking.

The simple sample program starts advertising automatically. You can perform following functions after establishing connection between the board and a peer device.

- A peer device receives SW4 state (PUSH/RELEASE) from the board
- A peer device controls LED4 state (ON/OFF) on the board

You need a peer device for the simple sample program behavior checking. From next section describes the procedure to use Android Device or iOS Device as a peer device.

6.4. Check with Android Device

This section describes procedures to check the simple sample program behavior with Android Device. We use “BLE Scanner Version 3.6”. Check following URL for details of BLE Scanner.

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

1) Launch BLE Scanner on Android Device and scan nearby device. Select the device the name is “REL-BLE” from the discovered device list (Figure a). After selecting the device, a connection between the board and Android Device is established.
2) Select CUSTOM SERVICE (UUID: 5BC1B9F7-A1F1-40AF-9043-C43692C18D7A) from the service list (Figure b).
3) Procedures for “Android Device receives SW4 state (PUSH/RELEASE) from the board”
You use CUSTOM CHARACTERISTIC (UUID: 5BC18D80-A1F1-40AF-9043-C43692C18D7A) for the LED4 control. When you tap on (N) button (Figure c, upside arrow), the board starts sending SW4 state to Android Device. Depends on the board SW4 state, you will see “HEX” value is changed (Figure c, downside arrow). When SW4 state is RELEASE you will see 0x00, when SW4 state is PUSH you will see 0x01. To stop sending SW4 state, re-tap the (N) button.
4) Procedures for “Android Device controls LED4 state (ON/OFF) on the board”.
You use CUSTOM CHARACTERISTIC (UUID: 5BC143EE-A1F1-40AF-9043-C43692C18D7A) for the LED4 control. When you tap (W) button (Figure d), a dialog is opened. Then select “Byte Array”, input “01” and tap “OK” (Figure e), then you will see LED4 is ON. To turn OFF LED4 writes 0x01.

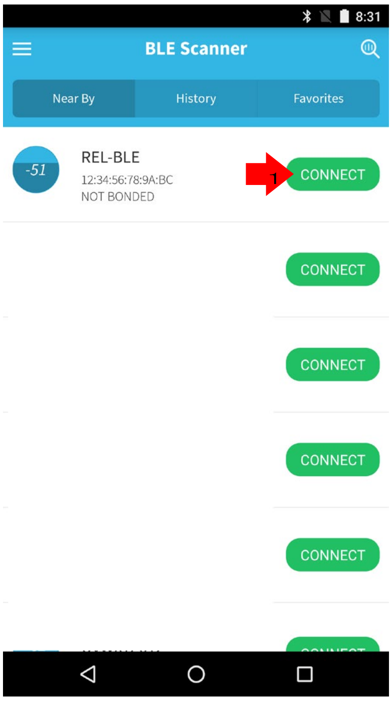


Figure a

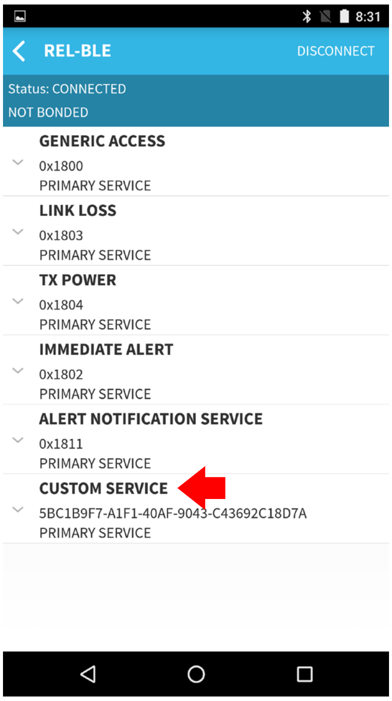


Figure b

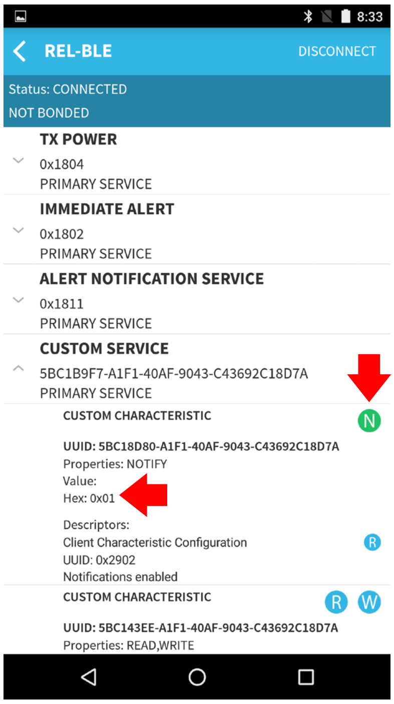


Figure c

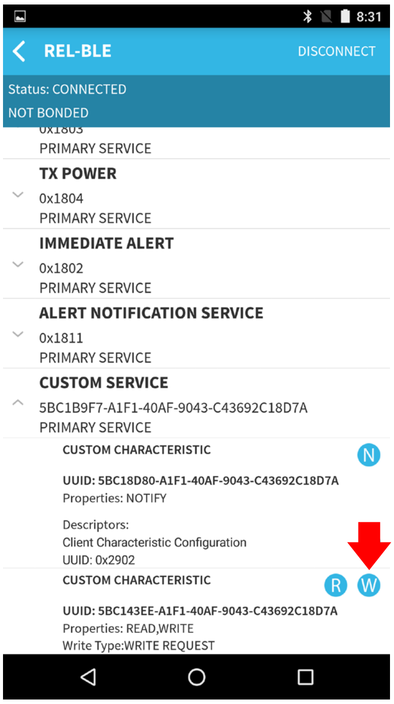


Figure d

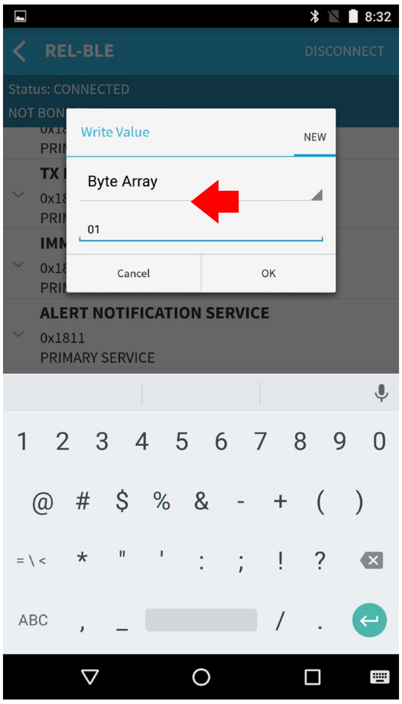


Figure e

6.5. Check with iOS Device

This section describes the process to check the behavior of the simple sample program with iOS Device. We use "LightBlue Version 2.4.0". Regarding LightBlue, see following URL.

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

1) Launch BLE Scanner on Android Device and scan nearby device. Select the device the name is "REL-BLE" from the discovered device list (Figure a). After selecting the device, a connection between the board and Android Device is established.
2) Procedures for "iOS Device receives SW4 state (PUSH/RELEASE) from the board"
   Select Characteristic (UUID:5BC18D80-A1F1-40AF-9043-C43692C18D7A) (Figure b, upside arrow). Tap on "Listen for notifications" button (Figure c), and then the board start sending SW4 state to iOS Device. Depends on the board SW4 state, you will see "NOTIFIED VALUES" is changed (Figure d, downside arrow). When SW4 state is RELEASE you will see 0x00, when SW4 state is PUSH you will see 0x01. To stop the SW4 state sending, tap on "Stop Listening" (Figure d, upside arrow).
3) Procedures for "iOS Device controls LED4 state (ON/OFF) on the board".
   Select Characteristic (UUID:5BC1B9F7-A1F1-40AF-9043-C43692C18D7A) (Figure b, downside arrow). When you tap on "write new value" (Figure e), then new dialog is opened. Input "01" and tap on "Done" on the dialog (Figure f), then you will see LED4 on the board is ON. To turn off the LED4, write 0x00.

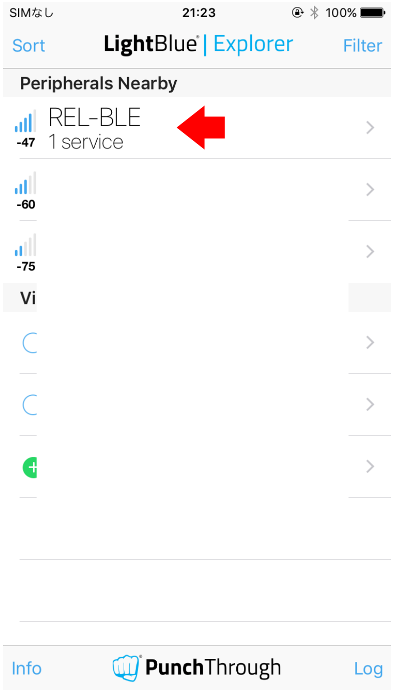


Figure a

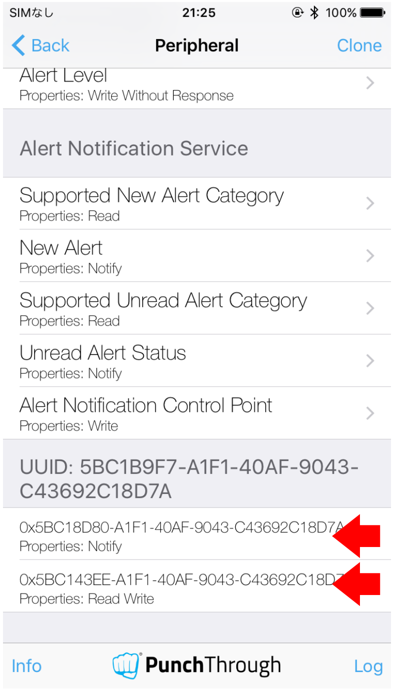


Figure b

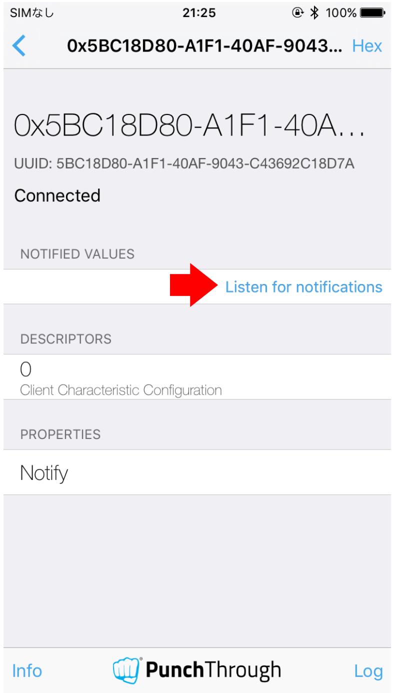


Figure c

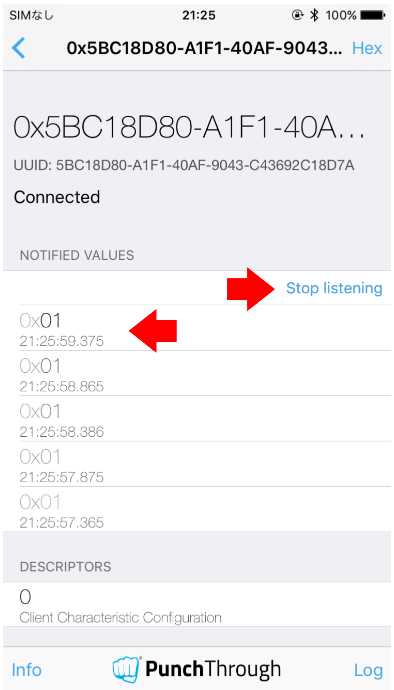


Figure d

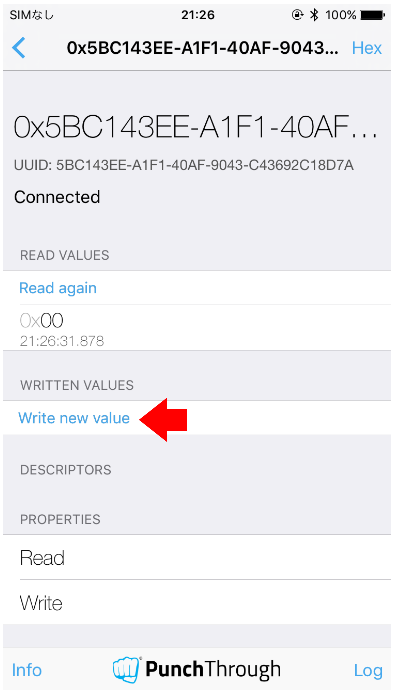


Figure e

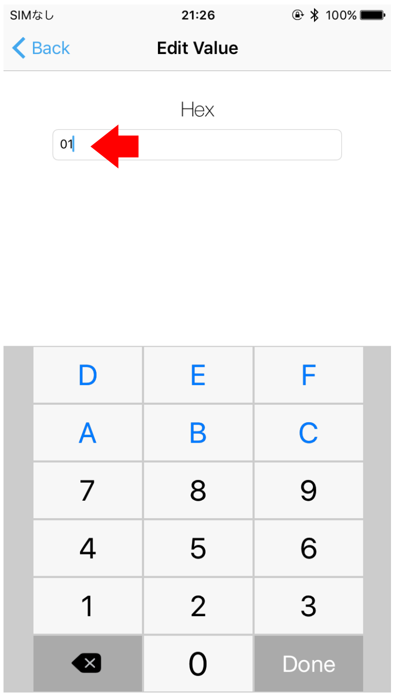


Figure f

7. Appendix

7.1. Transmit and Receive Operations in the Sample Program for the Computer

The application running on the APP MCU is provided the BLE services from the BLE MCU via rBLE_Host. APP MCU and BLE MCU are physically connected by the UART or CSI or IIC and communicate each other using RSCIP (Renesas Serial Communication Protocol) under the control of rBLE_Host.

Figure 7-1 shows the internal structure of the sample program for computer. The sample program for computer works by calling the command I/O function from the main processing and rBLE software blocks as shown Figure 7-1.

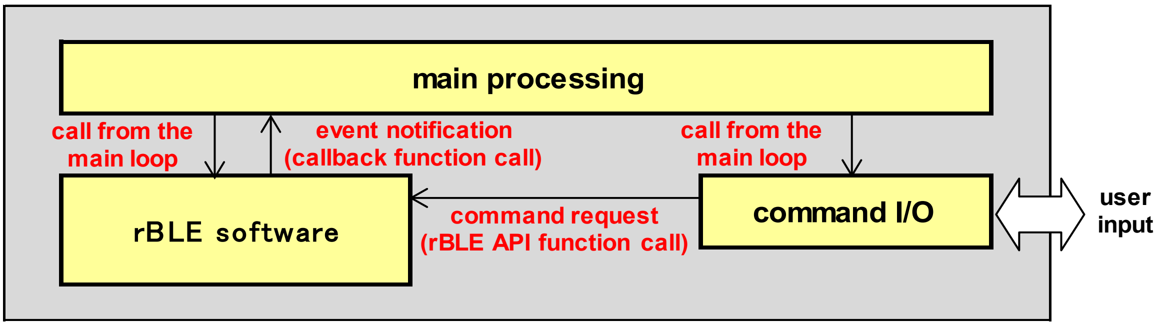


Figure 7-1 Internal structure of sample program

The process of transmitting and receiving in rBLE software block, are handled by calling the rBLE_Run function from the main processing block.

The rBLE_Run function checks the transmit buffer to the BLE MCU and calls the transmit function in RSCIP driver if there is the transmit data. It also analyzes received data in the receive buffer from BLE-MCU, if any, and calls registered application function based on the event information.

Also, if there is an event notification, it calls RSCIP function corresponding to the event. Figure 7-2 shows the sequence of internal processing.

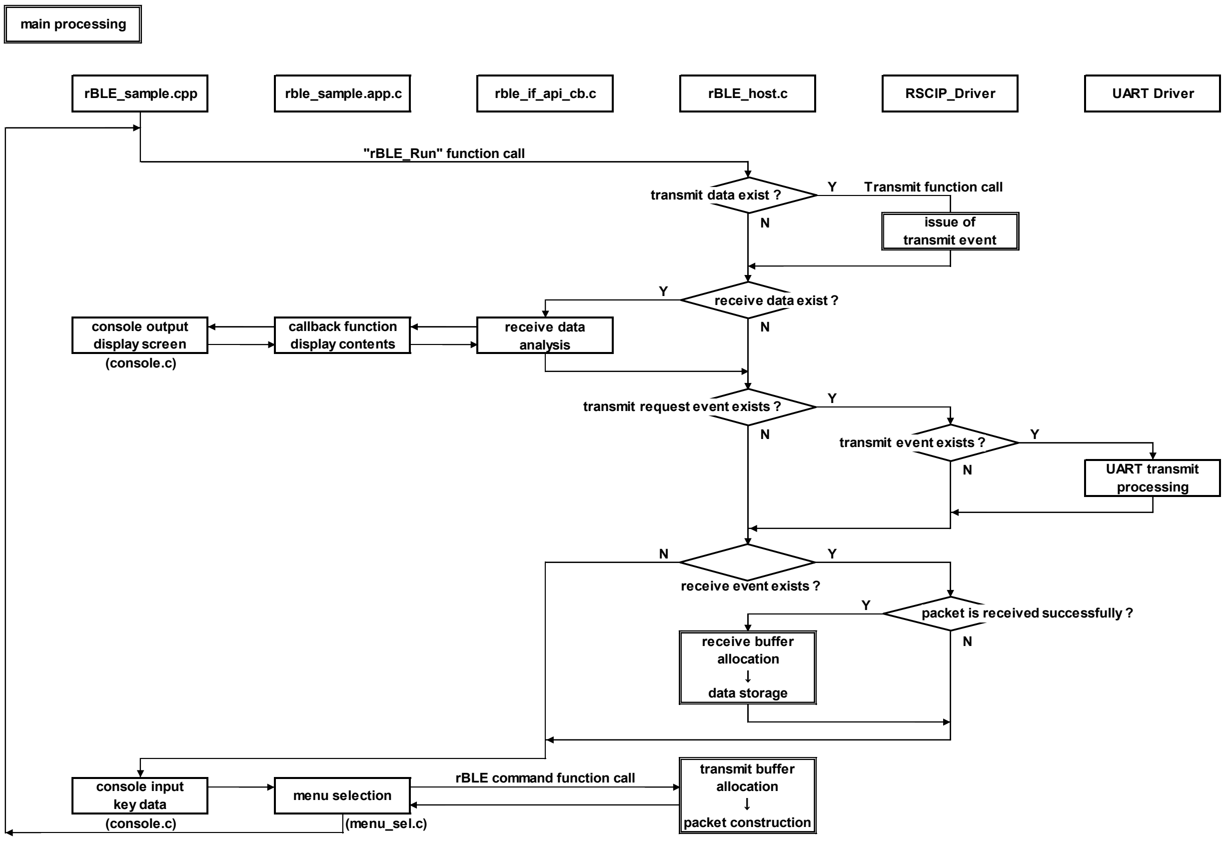


Figure 7-2 Internal processing of the sample program (main processing)

Figure 7-3 shows the sequence of events at the time of issuance of the transmit events from RSCIP. The RSCIP processes transmit requests both from the retransmit processing block and from the application in one place, so it issues transmit event request to the rBLE when transmit events from both side. The rBLE calls RSCIP transmit function as shown in Figure 7-2 if transmit request is generated from both sides.

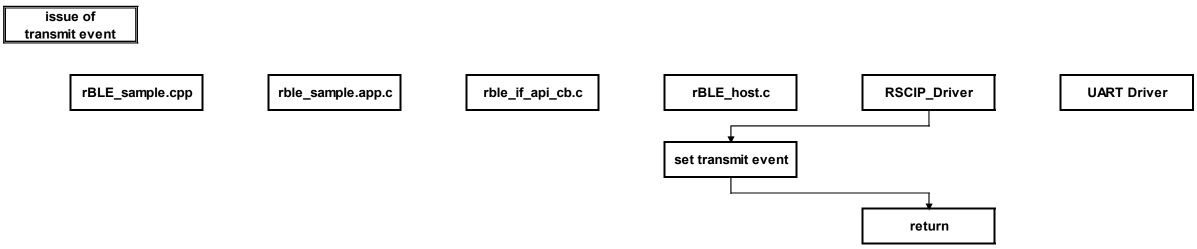


Figure 7-3 Internal processing of sample program (issue of transmit event)

Figure 7-4 shows the sequence of events at the time of issuance of the receive events from RSCIP. Considering that the data receive notification from serial communication driver is called from an interrupt, RSCIP issues the receive event request to the rBLE when a packet has been received. The rBLE calls RSCIP packet receive function as shown in Figure 7-2 if receive event request is generated.

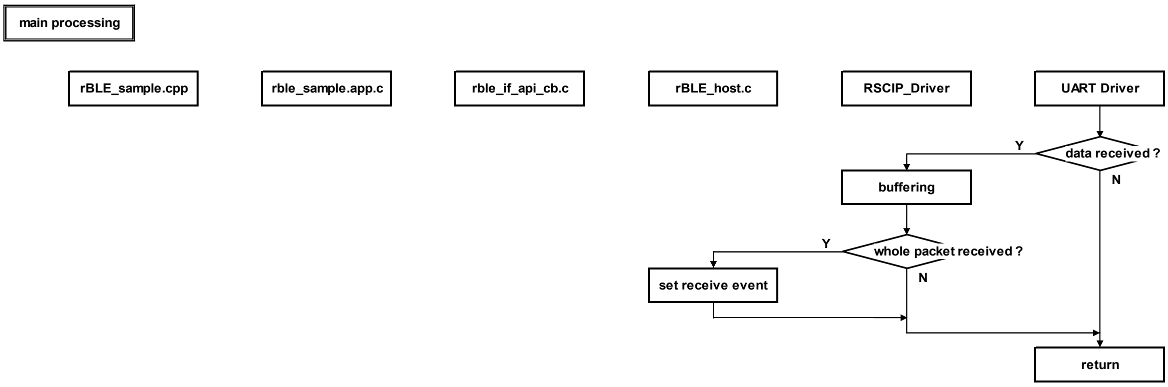


Figure 7-4 Internal processing of sample program (issue of receive event)

7.2. Requirements and Flow Chart of Serial Communication Driver on APP MCU

The requirements on the APP MCU in the application development of modem configuration are summarized below.

H/W resource

1 channel of UART or CSI (Clocked Serial Interface) or IIC(Inter-Integrated Circuit) for serial communication is required for the communication with BLE MCU.

Timer

The timeout function is required in the RSCIP driver. (Refer to the RSCIP implementation in rBLE_Host).

Serial communication driver

The serial communication driver using UART or CSI or IIC should be prepared by the user.
In addition, the following functions are required in the serial communication driver as an interface between the RSCIP driver and serial communication driver.

•

Function	BOOL serial_init (void)	
Overview	Serial communication driver initialization function	
Description	This function initializes the serial communication driver. Initialize the serial communication driver in settings that are described in the Bluetooth Low Energy Protocol Stack User's Manual.	
Arguments	None	
Return value	TRUE	Initialization is completed successfully
	FALSE	Initialization is completed with some errors

Function	BOOL serial_write (uint8_t *bufptr, uint16_t size)	
Overview	Serial communication driver transmit function	
Description	This function is a non-blocking function that transmits specified size of data via the serial communication line. The transmit data size is specified by the argument 'size' and transmit data is stored in the area pointed by the argument '*bufprt'. When the transmission of data is completed, call following transmit0 completion notification function (RSCIP_Uart_Tx_Done) to RSCIP driver. void RSCIP_Uart_Tx_Done(void); If the method other than the two-wire UART connection is used, follow the transmit handshake procedure that is described in the Bluetooth Low Energy Protocol Stack User's Manual.	
Arguments	uint8_t *bufptr	Pointer to the transmit data buffer
	uint16_t size	Data size to be transmitted
Return Value	TRUE	Transmission is completed successfully
	FALSE	Transmission is completed with some errors
	This function may be called from an interrupt. In the sample program, transmit processing is performed by rBLE_Run function, which is called from main loop, except the minimum required processing that should be done in this function.	

Function	BOOL serial_read (uint8_t *bufptr, uint16_t size)	
Overview	Serial communication driver receive function	
Description	This function is a non-blocking function that receives specified size of data via the serial communication line. The receive data size is specified by the argument 'size'. Store received data into the area pointed by the argument '*bufprt'. When the reception of data is completed, call following receive completion notification function (RSCIP_Uart_Rx_Done) to RSCIP driver. void RSCIP_Uart_Rx_Done (void); If the method other than the two-wire UART connection is used, follow the receive handshake procedure that is described in the Bluetooth Low Energy Protocol Stack User's Manual. In addition, after calling the RSCIP receive completion notification function, call the following RSCIP get receive status function and check its return value in order to determine whether or not to receive data continuously. BOOL RSCIP_Uart_Rx_Idle (void); FALSE indicates that the packet reception is not completed. TRUE indicates that the packet reception is completed, and the driver is waiting for the beginning of the next packet.	
Arguments	uint8_t *bufptr	pointer to the receive data buffer

	uint16_t size	data size to be received
Return Value	TRUE	Reception is completed successfully
	FALSE	Reception is completed with some errors
Supplement	This function may be called from an interrupt. In the sample program, receive processing is performed by rBLE_Run function, which is called from main loop, except the minimum required processing that should be done in this function.	

Function	void serial_exit (void)
Overview	Serial communication driver exit function
Description	This function does the exit procedure of the serial communication driver. Do the exit procedure of the serial communication driver.
Arguments	none
Return Value	none

In the Modem configuration, the following connection methods are available as the serial communication line. Refer to the Bluetooth Low Energy Protocol Stack User's Manual for more details of the connection method.

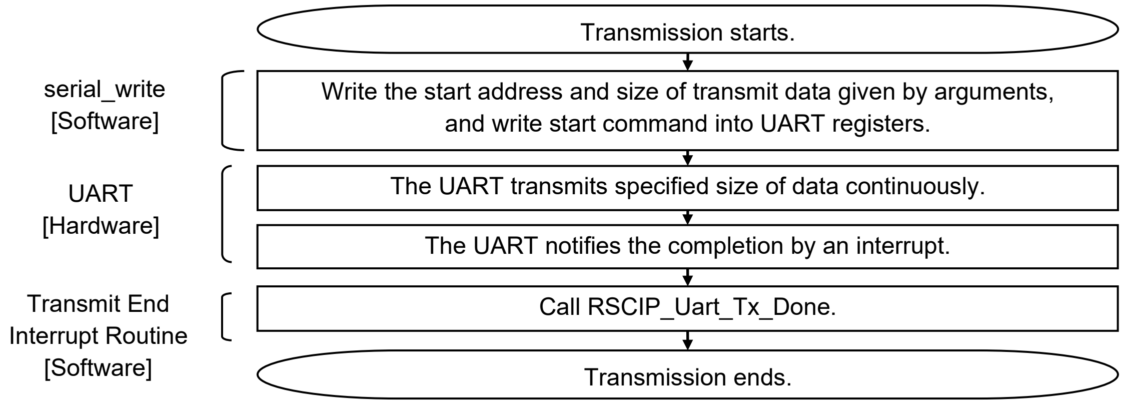
Implement the driver that fits with your system resources, refer to the flow chart of the serial communication procedure examples,

Hardware	Connection method	Example of transmit procedure	Example of receive procedure
UART	2-wire	Refer to 7.2.1 Transmit Procedure Example using the UART 2-wire Connection Method, below.	Refer to (2) Receive Procedure Example using the UART Two-wire Connection Method, below.
	3-wire	Refer to 7.2.3 Transmit Procedure Example using the UART 3-wire Connection Method, below.	Refer to 7.2.5 Receive Procedure Example using the UART 3-wire and 2-wire with Branch Connection Methods, below.
	2-wire with branch	Refer to 7.2.4 Transmit Procedure Example using the UART 2-wire with Branch Connection Method, below.	
CSI	4-wire	Refer to 7.2.6 Transmit Procedure Example using the CSI 4-wire Connection Method below.	Refer to 7.2.8 Receive Procedure Example using the CSI 4-wire and 5-wire Connection Method, below.
	5-wire	Refer to 7.2.7 Transmit Procedure Example using the CSI 5-wire Connection Method, below.	
IIC	3-wire	Refer to 7.2.6 Transmit Procedure Example using the IIC 3-wire Connection Method below.	Refer to 7.2.8 Receive Procedure Example using the IIC 3-wire Connection Method, below.

7.2.1. Transmit Procedure Example using the UART 2-wire Connection Method

The following flowchart shows an example of transmit procedure using the UART 2-wire connection method.

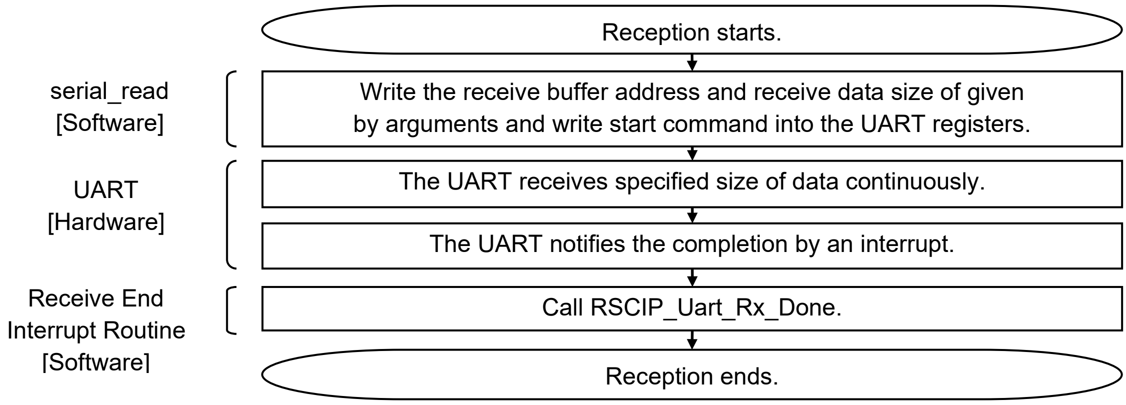
As a prerequisite, UART hardware in this example starts transmission by writing the transmit data address, transmit data size and start command into UART registers, and generates an interrupt after the transmission of specified data is completed.



7.2.2. Receive Procedure Example using the UART Two-wire Connection Method

The following flowchart shows an example of receive procedure using the UART 2-wire connection, 3-wire connection or 2-wire with branch connection methods.

As a prerequisite, UART hardware in this example starts reception by writing the receive buffer address, receive data size and start command into UART registers, and generates an interrupt after the reception of specified size of data is completed.



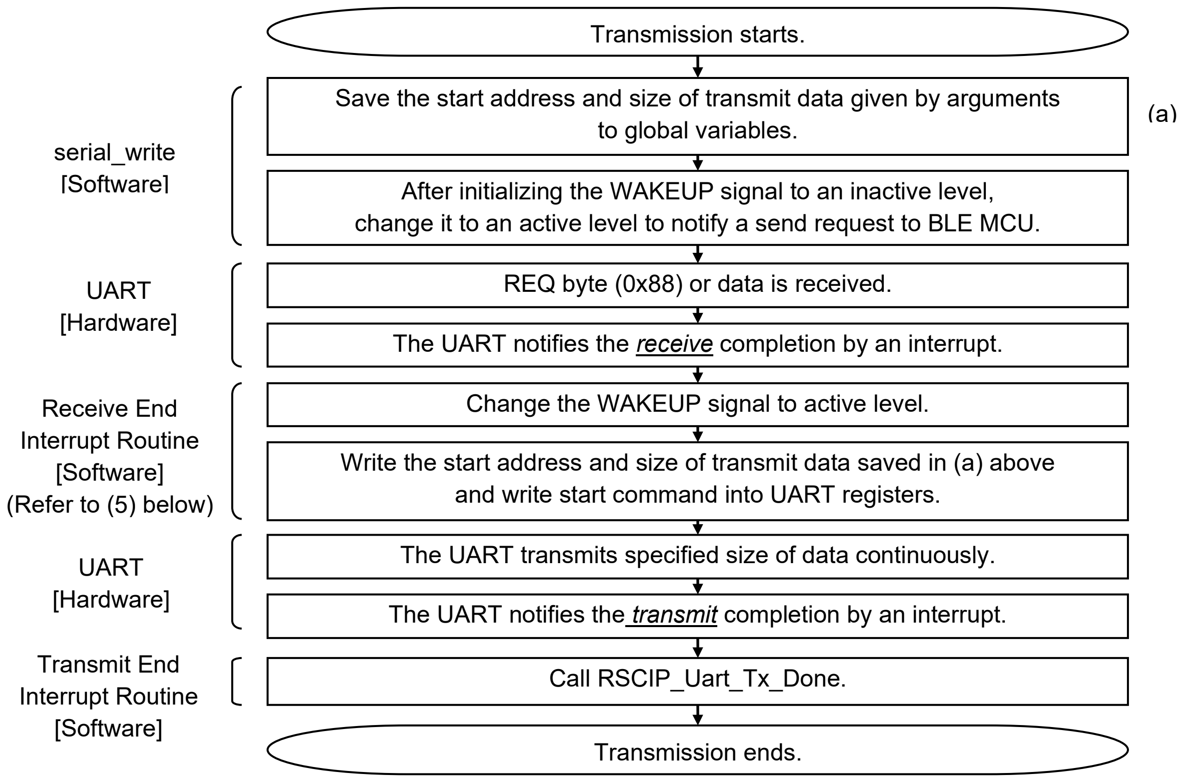
* RSCIP_Uart_Rx_Done function calls the serial_read function to start the next reception operation.

7.2.3. Transmit Procedure Example using the UART 3-wire Connection Method

The following flowchart shows an example of transmit procedure using the UART 3-wire connection method.

As a prerequisite, UART hardware in this example starts transmission by writing the transmit data address, transmit data size and start command into UART registers, and generates an interrupt after the transmission of specified data is completed. Also, the receive procedure example described in 7.2.5 below is used.

In addition, for reliable communication, it is necessary to add the timeout process, in which carry out monitoring during handshake procedure and re-execute the handshake procedure if a timeout occurs..



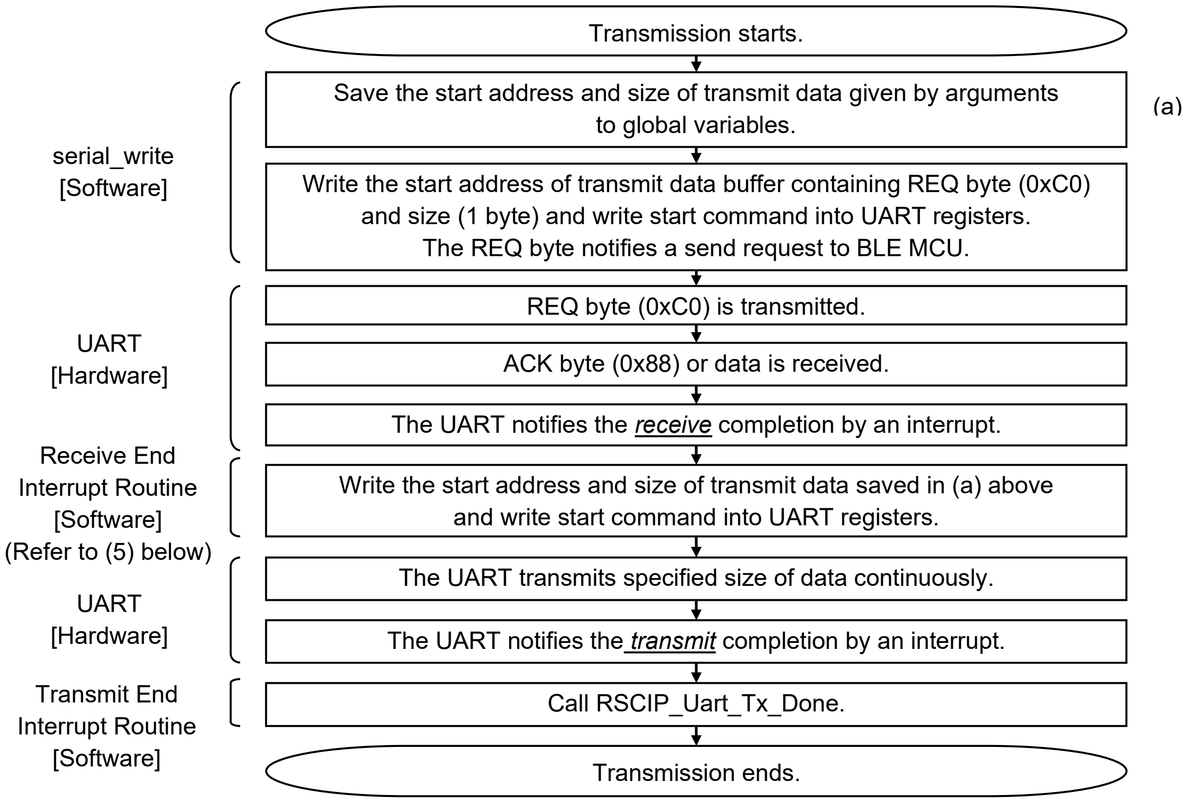
* Receive End Interrupt Routine is used in receive procedure example.

7.2.4. Transmit Procedure Example using the UART 2-wire with Branch Connection Method

The following flow chart shows an example of transmit procedure using the UART 2-wire with branch connection method.

As a prerequisite, UART hardware in this example starts transmission by writing the transmit data address, transmit data size and start command into UART registers, and generates an interrupt after the transmission of specified data is completed. Also, the receive procedure example described in 7.2.5 below is used.

In addition, for reliable communication, it is necessary to add the timeout process, in which carry out monitoring during handshake procedure and re-execute the handshake procedure if a timeout occurs.

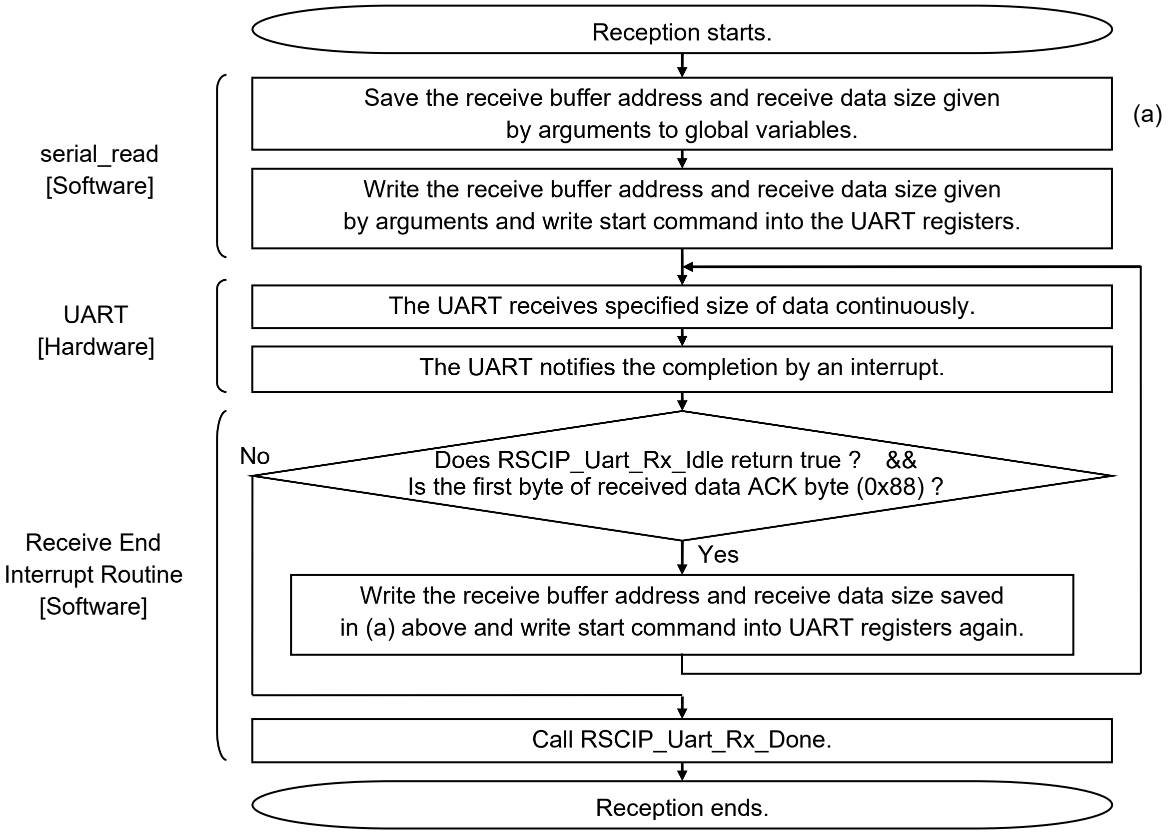


* Receive End Interrupt Routine is used in receive procedure example.

7.2.5. Receive Procedure Example using the UART 3-wire and 2-wire with Branch Connection Methods

The following flowchart shows an example of receive procedure using the UART 3-wire connection and 2-wire with branch connection methods.

As a prerequisite, UART hardware in this example starts reception by writing the receive buffer address, receive data size and start command into UART registers, and generates an interrupt after the reception of specified size of data is completed.



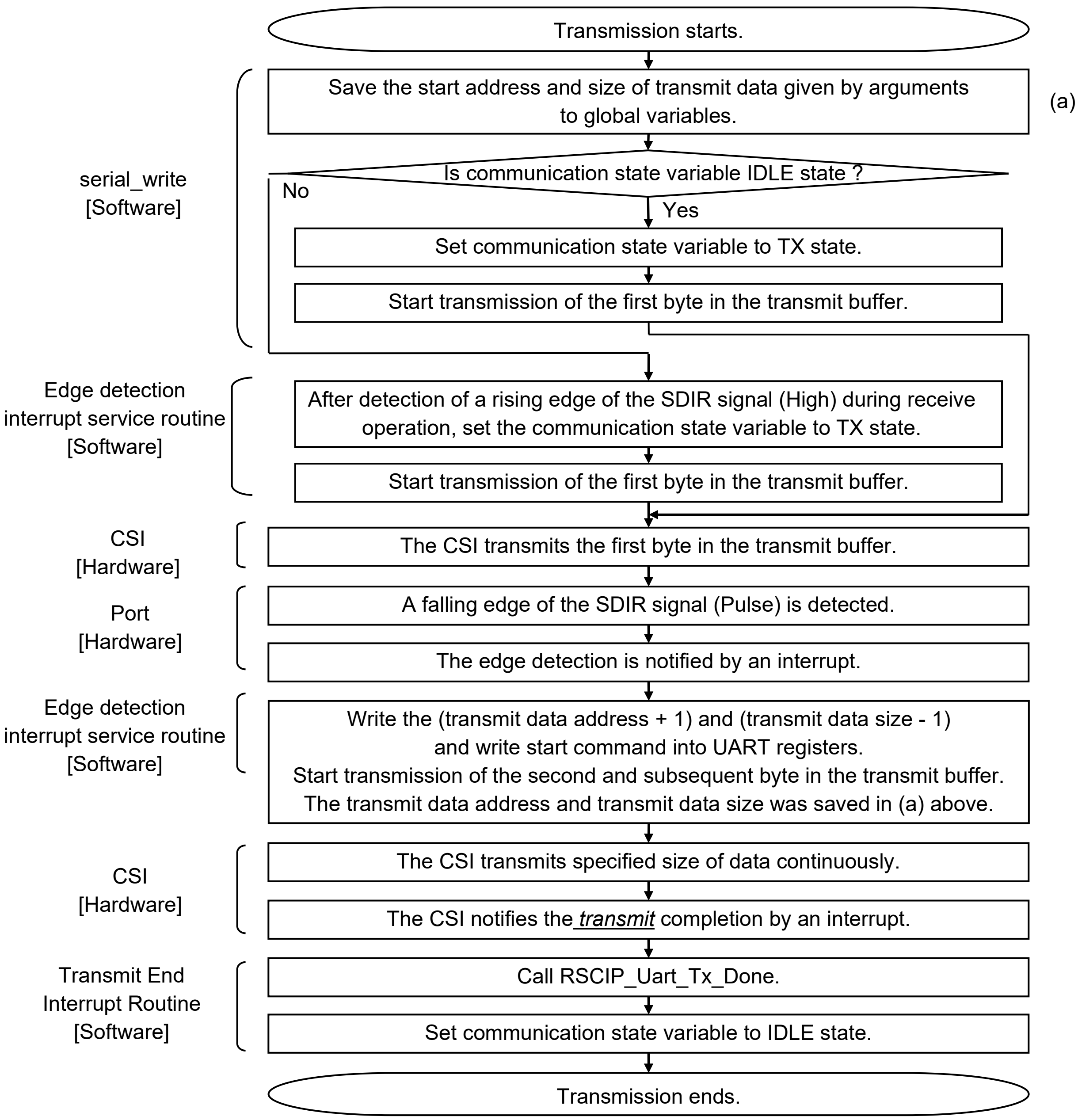
* RSCIP_Uart_Rx_Done function calls the serial_read function to start the next reception operation.

7.2.6. Transmit Procedure Example using the CSI 4-wire Connection Method

The following flowchart shows an example of transmit procedure using the CSI 4-wire connection method.

As a prerequisite, CSI hardware in this example starts transmission by writing the transmit data address, transmit data size and start command into CSI registers, and generates an interrupt after the transmission of specified data is completed. It is assumed that an input port of APP MCU are connected to the SDIR signal and also an interrupt is generated by dual edge detection (both the falling edge and rising edge) of the SDIR signal.

In addition, for reliable communication, it is necessary to add the timeout process, in which carry out monitoring during handshake procedure and re-execute the handshake procedure if a timeout occurs.



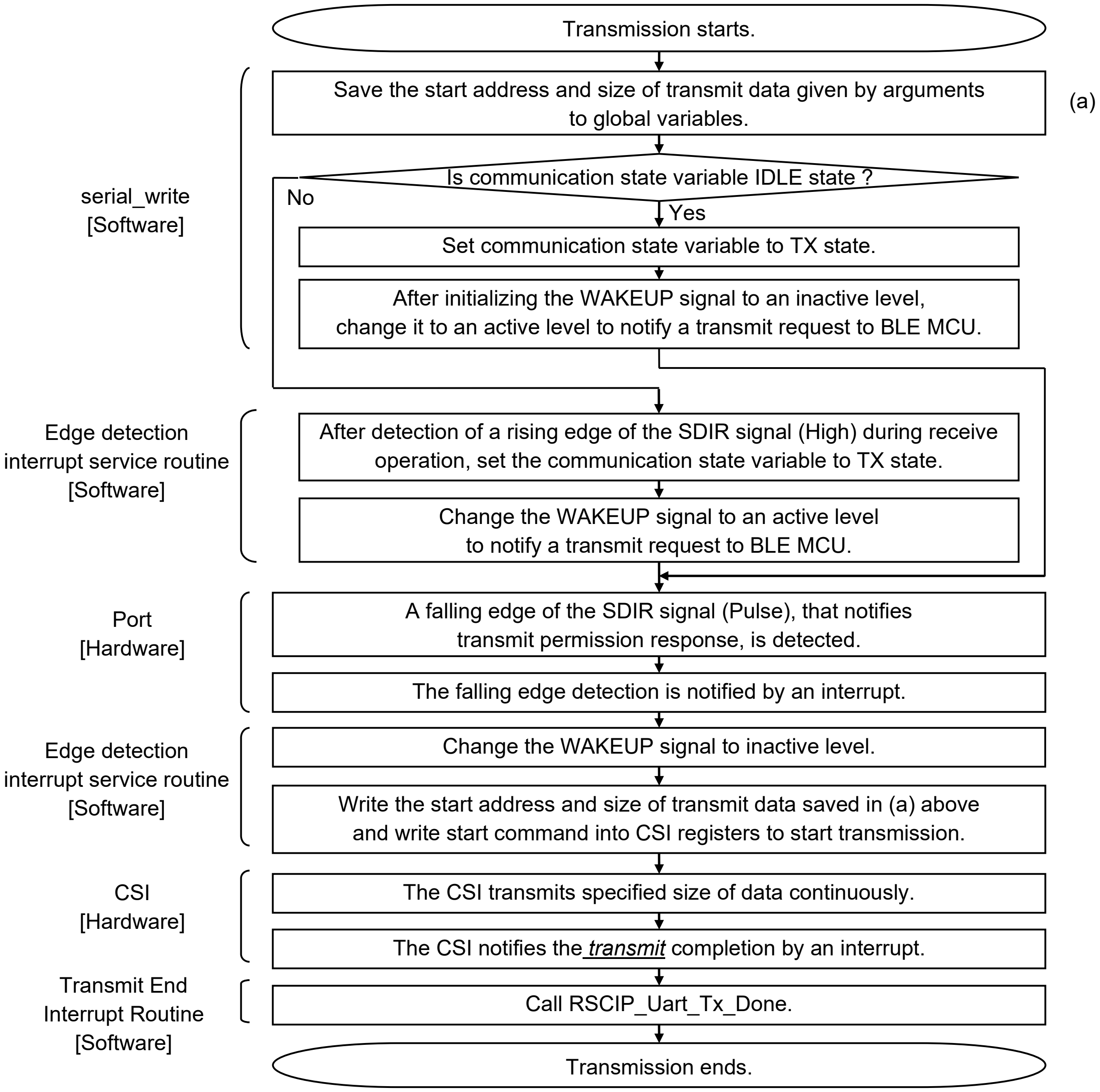
* The edge detection interrupt service routine is used more than once in the transmission operation.
* The edge detection interrupt service routine is also used in the receive operation.

7.2.7. Transmit Procedure Example using the CSI 5-wire Connection Method

The following flowchart shows an example of transmit procedure using the CSI 4-wire connection method.

As a prerequisite, CSI hardware in this example starts transmission by writing the transmit data address, transmit data size and start command into CSI registers, and generates an interrupt after the transmission of specified data is completed. It is assumed that an input port of APP MCU are connected to the SDIR signal and also an interrupt is generated by dual edge detection (both the falling edge and rising edge) of the SDIR signal.

In addition, for reliable communication, it is necessary to add the timeout process, in which carry out monitoring during handshake procedure and re-execute the handshake procedure if a timeout occurs.

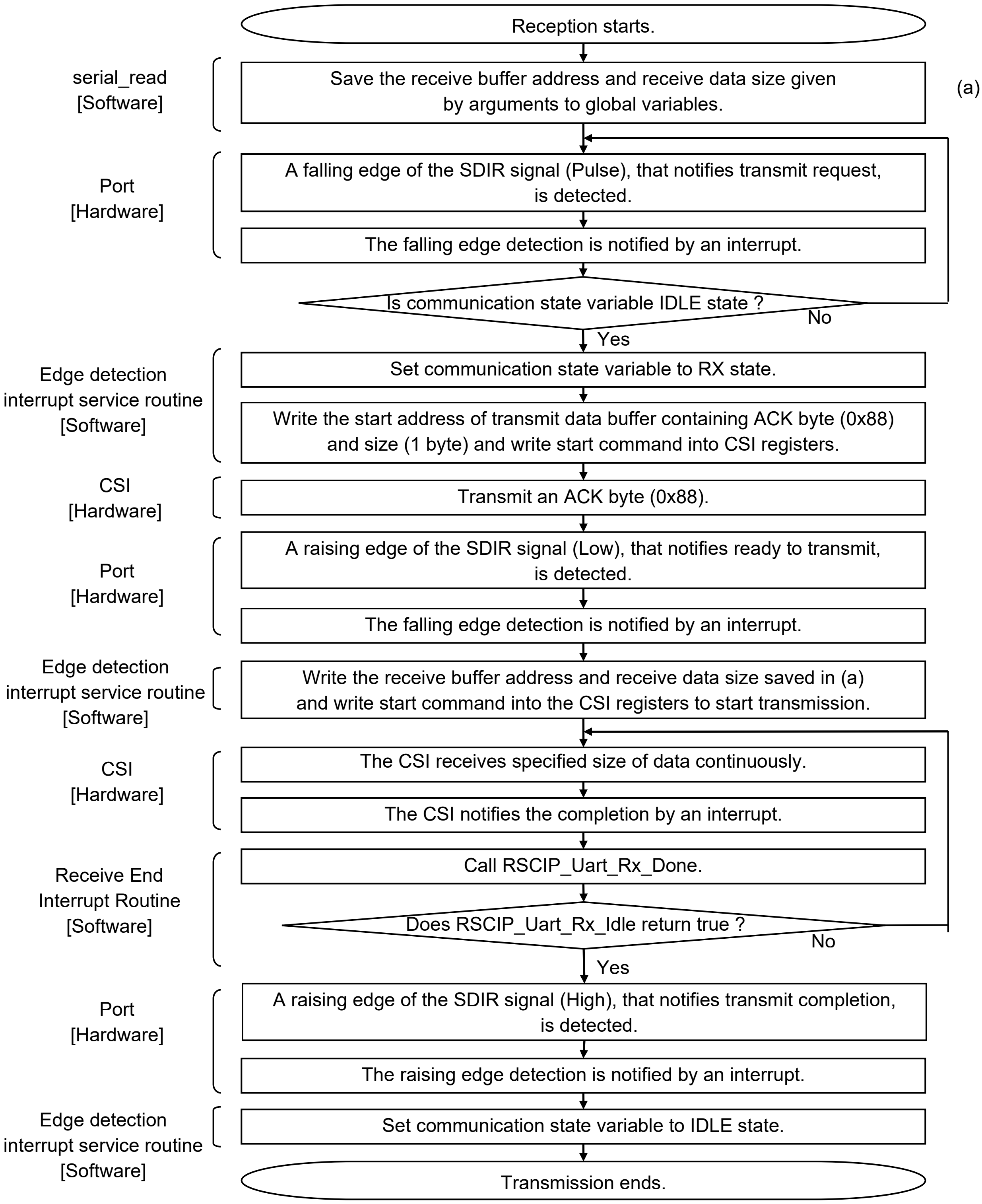


* The edge detection interrupt service routine is used more than once in the transmission operation.
* The edge detection interrupt service routine is also used in the receive operation.

7.2.8. Receive Procedure Example using the CSI 4-wire and 5-wire Connection Method

The following flowchart shows an example of receive procedure using the CSI 4-wire and 5-wire connection method.

As a prerequisite, CSI hardware in this example start reception by writing the receive buffer address, receive data size and start command into CSI registers, and generates an interrupt after the reception of specified data is completed. It is assumed that an input port of APP MCU are connected to the SDIR signal and also an interrupt is generated by dual edge detection (both the falling edge and rising edge) of the SDIR signal.



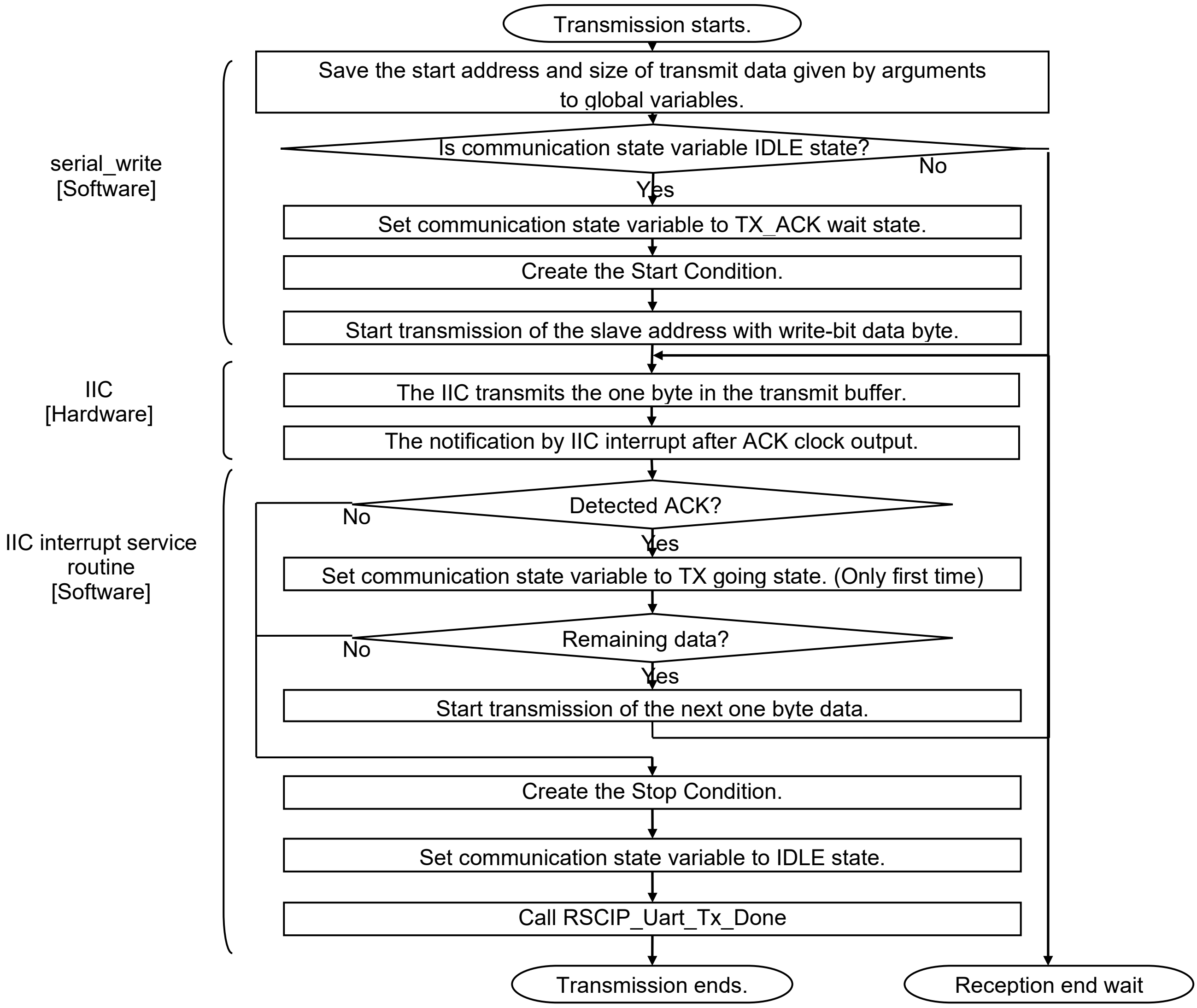
* RSCIP_Uart_Rx_Done function calls the serial_read function to start the next reception operation.
* The edge detection interrupt service routine is used more than once in the transmission operation.
* The edge detection interrupt service routine is also used in the receive operation.

7.2.9. Transmit Procedure Example using the IIC 3-wire Connection Method

The following flowchart shows an example of receive procedure using the IIC 3-wire connection method.

As a prerequisite, IIC hardware to generate an interrupt after reception of 1 byte data.

In addition, for reliable communication, it is necessary to add the timeout process, in which carry out monitoring during

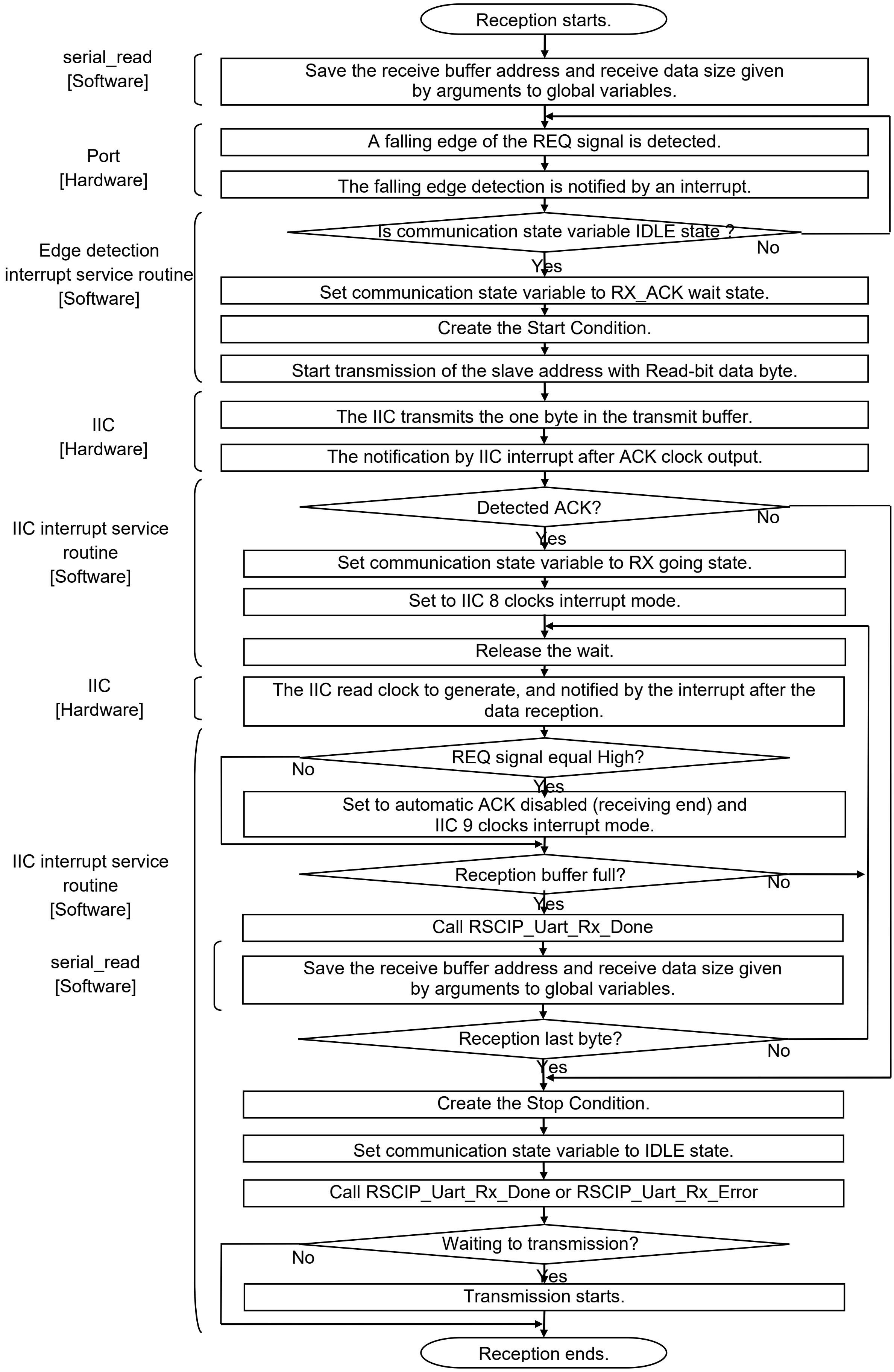


handshake procedure and re-execute the handshake procedure if a timeout occurs.

7.2.10. Receive Procedure Example using the IIC 3-wire Connection Method

The following flowchart shows an example of transmit procedure using the IIC 3-wire connection method.

As a prerequisite, IIC hardware to generate an interrupt after transmission of 1 byte data. It is assumed that an input port of APP MCU are connected to the REQ signal and also an interrupt is generated by falling edge detection of the REQ signal.


Reception starts.
serial_read
[Software]
Save the receive buffer address and receive data size given by arguments to global variables.
Port
[Hardware]
A falling edge of the REQ signal is detected.
The falling edge detection is notified by an interrupt.
Edge detection
interrupt service routine
[Software]
Is communication state variable IDLE state ?
No
Yes
Set communication state variable to RX_ACK wait state.
Create the Start Condition.
Start transmission of the slave address with Read-bit data byte.
IIC
[Hardware]
The IIC transmits the one byte in the transmit buffer.
The notification by IIC interrupt after ACK clock output.
IIC interrupt service
routine
[Software]
Detected ACK?
No
Yes
Set communication state variable to RX going state.
Set to IIC 8 clocks interrupt mode.
Release the wait.
IIC
[Hardware]
The IIC read clock to generate, and notified by the interrupt after the data reception.
IIC interrupt service
routine
[Software]
REQ signal equal High?
No
Yes
Set to automatic ACK disabled (receiving end) and IIC 9 clocks interrupt mode.
Reception buffer full?
No
Yes
Call RSCIP_Uart_Rx_Done
serial_read
[Software]
Save the receive buffer address and receive data size given by arguments to global variables.
Reception last byte?
No
Yes
Create the Stop Condition.
Set communication state variable to IDLE state.
Call RSCIP_Uart_Rx_Done or RSCIP_Uart_Rx_Error
Waiting to transmission?
No
Yes
Transmission starts.
Reception ends.

7.3. Porting of the Sample Program

When porting the sample program to APP MCU, there are modules that should be newly developed by the user and modules that can be reused directly.

Table 7-1 shows the classification of them.

Table 7-1 How to port a sample program

Folder Name	Classification	Porting Details
BLE_Sample\src\Platform\G1D_cs_iar\	new development	With reference to the sample program, this module should be newly developed by the user, to meet the resources in the APP MCU.
BLE_Sample\src\rBLE\src\host	reuse	This module can be reused directly.
BLE_Sample\src\rBLE\src\include	reuse	This module can be reused directly.
BLE_Sample\src\rBLE\src\rscip	reuse	This module can be reused directly.
BLE_Sample\src\rBLE\src\sample_app	new development	With reference to API usage in the sample program, this module should be newly developed by the user.

It is to be noted that the reference value of the size of the reusable sample program are shown in Table 7-2. These values are the result of compiling for the RL78/G1D.

Build Environment: CS+ for CC V4.00.00 / RL78 compiler CC-RL V1.03.00

Table 7-2 ROM size / RAM size

Components	ROM size	RAM size
rBLE (BLE_Sample\src\rBLE\src\host)	52,519 bytes	2,898 bytes
RSCIP (BLE_Sample\src\rBLE\src\rscip	4,279 bytes	1,268 bytes

When you implement the following measures, it is possible to reduce the RAM size about 2KB.

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

Before changing: #define MAX_BUFF_NUM 8
(Follows)
After changing: #define MAX_BUFF_NUM 2

[Note] You cannot call the command more than MAX_BUFF_NUM continuously.

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

Before changing: static uint8_t rBLE_Over_Packet_Temp[0x256];
(Follows)
After changing: static uint8_t rBLE_Over_Packet_Temp[1];

[Note] You cannot handle data more than 128-byte in RBLE_VS_Flash_Access API.
If you call the API, illegal memory access occurs.

7.4. How to use the Direct Test Mode

Direct Test Mode is performed by the Vendor Specific (VS) command. Figure 7-5 shows the Vendor Specific (VS) command menu. The menu items from 2 to 5 are related to the Direct Test Mode.

After this section, the commands related to the Direct Test Mode are explained.

[Note]

About the details of Direct Test Mode, refer to the ‘Chapter 8. Vendor Specific’ in the ‘API Reference Manual: Basics’

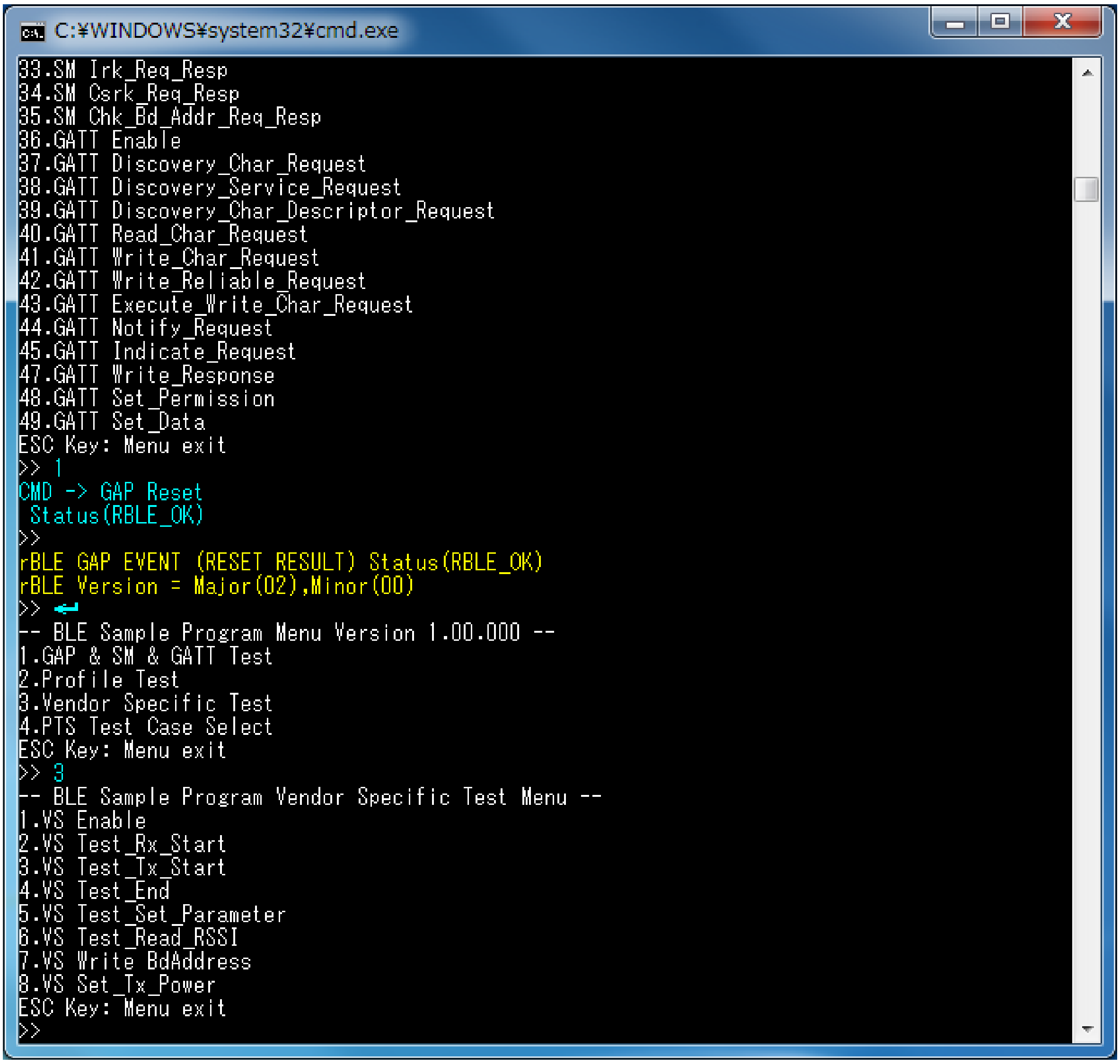


Figure 7-5 Vendor Specific (VS) command menu

7.4.1. Direct Test Mode (Receiver)

Using the VS menu number 2 'VS Test_Rx_Start', you can start the Direct Test Mode (Receiver).

```
C:¥WINDOWS¥system32¥cmd.exe
-- BLE Sample Program Vendor Specific Test Menu --
1.VS Enable
2.VS Test_Rx_Start
3.VS Test_Tx_Start
4.VS Test_End
5.VS Test_Set_Parameter
6.VS Test_Read_RSSI
7.VS Write BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
 Status(RBLE_OK)
>> 2
CMD -> VS Test_Rx_Start
 VS_Test_Rx_Start Useage:TestNo RxFreq(0-39)
>> 2 39
CMD -> VS Test_Rx_Start
 Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLE_OK)
>>
```

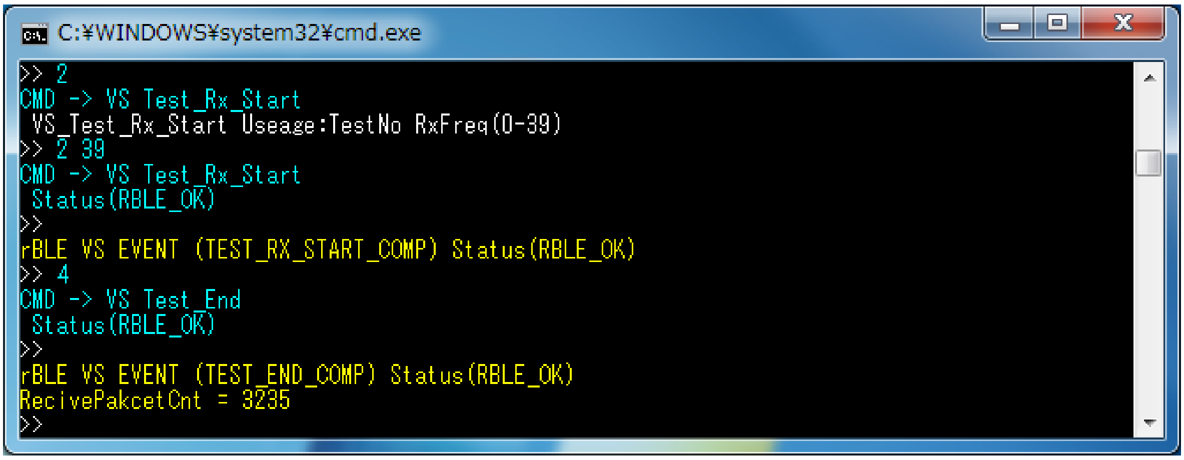
Figure 7-6 Log of Direct Test Mode (Receiver) Start

Using the VS menu number 2 'VS Test_Rx_Start', you can set the receive frequency (channel number). If no argument is given, it displays the usage of this command. Figure 7-6 shows the log of execution, when the receive frequency is channel 39 (2,480MHz).

If you want to terminate the execution of Direct Test Mode (Receiver), use the VS menu number 4 'VS Test_End'. Figure 7-7 shows the log after execution of Direct Test Mode (Receiver). The number of received packets is displayed after this test. The data have been received zero times in Figure 7-7 and 3,235 times in Figure 7-8.

```
C:¥WINDOWS¥system32¥cmd.exe
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
 Status(RBLE_OK)
>> 2
CMD -> VS Test_Rx_Start
 VS_Test_Rx_Start Useage:TestNo RxFreq(0-39)
>> 2 39
CMD -> VS Test_Rx_Start
 Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLE_OK)
>> 4
CMD -> VS Test_End
 Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
RecivePakcetCnt = 0
>>
```

Figure 7-7 Log of Direct Test Mode (Receiver) End

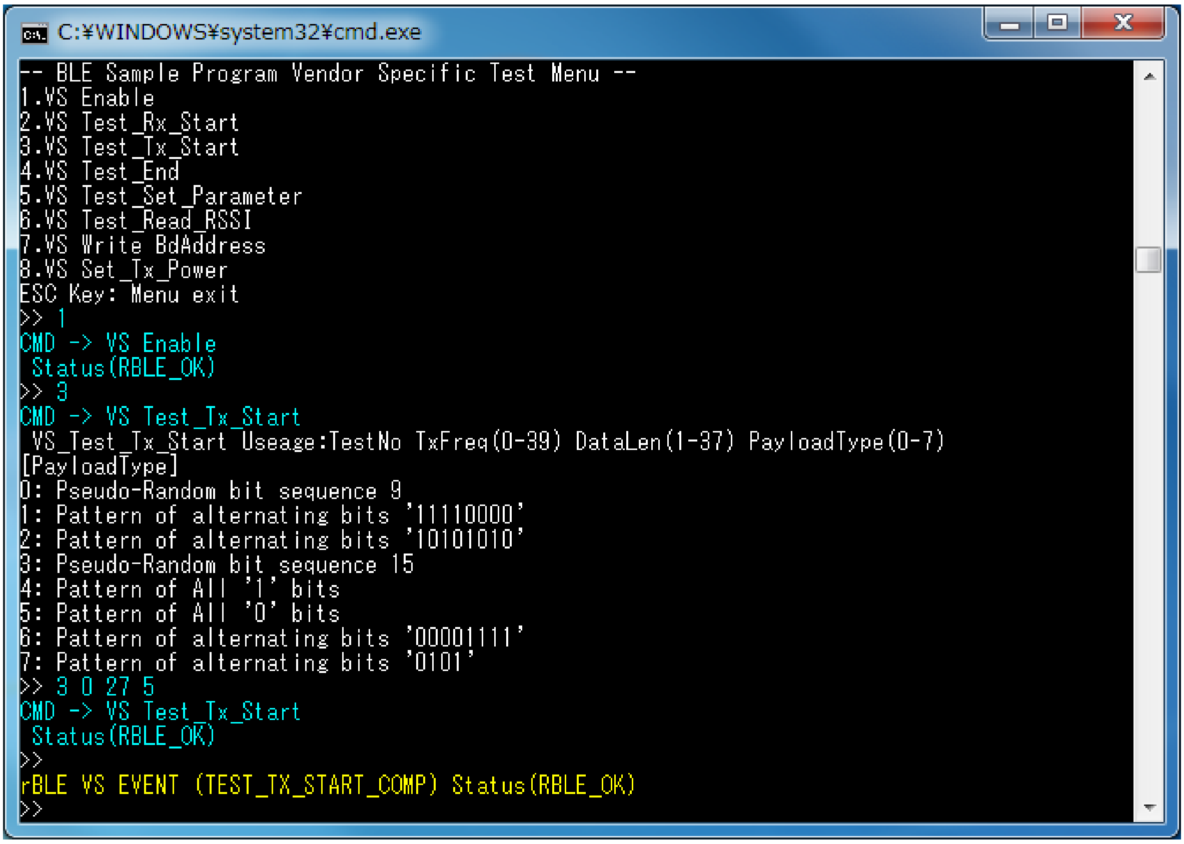


```
C:¥WINDOWS¥system32¥cmd.exe
>> 2
CMD -> VS Test_Rx_Start
 VS_Test_Rx_Start Useage:TestNo RxFreq(0-39)
>> 2 39
CMD -> VS Test_Rx_Start
 Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_RX_START_COMP) Status(RBLE_OK)
>> 4
CMD -> VS Test_End
 Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_END_COMP) Status(RBLE_OK)
RecivePakcetCnt = 3235
>>
```

Figure 7-8 Log of Direct Test Mode (Receiver) End (cont.)

7.4.2. Direct Test Mode (Transmitter)

Using the VS menu number 3 ‘VS Test_Tx_Start’, you can start the Direct Test Mode (Transmitter).



```
C:¥WINDOWS¥system32¥cmd.exe
-- BLE Sample Program Vendor Specific Test Menu --
1.VS Enable
2.VS Test_Rx_Start
3.VS Test_Tx_Start
4.VS Test_End
5.VS Test_Set_Parameter
6.VS Test_Read_RSSI
7.VS Write BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
 Status(RBLE_OK)
>> 3
CMD -> VS Test_Tx_Start
 VS_Test_Tx_Start Useage:TestNo TxFreq(0-39) DataLen(1-37) PayloadType(0-7)
[PayloadType]
0: Pseudo-Random bit sequence 9
1: Pattern of alternating bits '11110000'
2: Pattern of alternating bits '10101010'
3: Pseudo-Random bit sequence 15
4: Pattern of All '1' bits
5: Pattern of All '0' bits
6: Pattern of alternating bits '00001111'
7: Pattern of alternating bits '0101'
>> 3 0 27 5
CMD -> VS Test_Tx_Start
 Status(RBLE_OK)
>>
rBLE VS EVENT (TEST_TX_START_COMP) Status(RBLE_OK)
>>
```

Figure 7-9 Log of Direct Test Mode (Transmitter) Start

Using the VS menu number 3 ‘VS Test_Tx_Start’, you can set the transmit frequency (channel number), data size and data type as arguments. If no argument is given, it displays the usage of this command. Figure 7-9 shows the log of execution, when the transmit frequency is channel 0 (2,420MHz), data size is 27 bytes and data type is ALL0.

If you want to terminate the execution of Direct Test Mode (Transmitter), use the VS menu number 4 ‘VS Test_End’. Figure 7-10 shows the log after execution of Direct Test Mode (Transmitter). The number of received packets is displayed after this test and it is always 0.

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

Figure 7-10 Log of Direct Test Mode (Transmitter) End

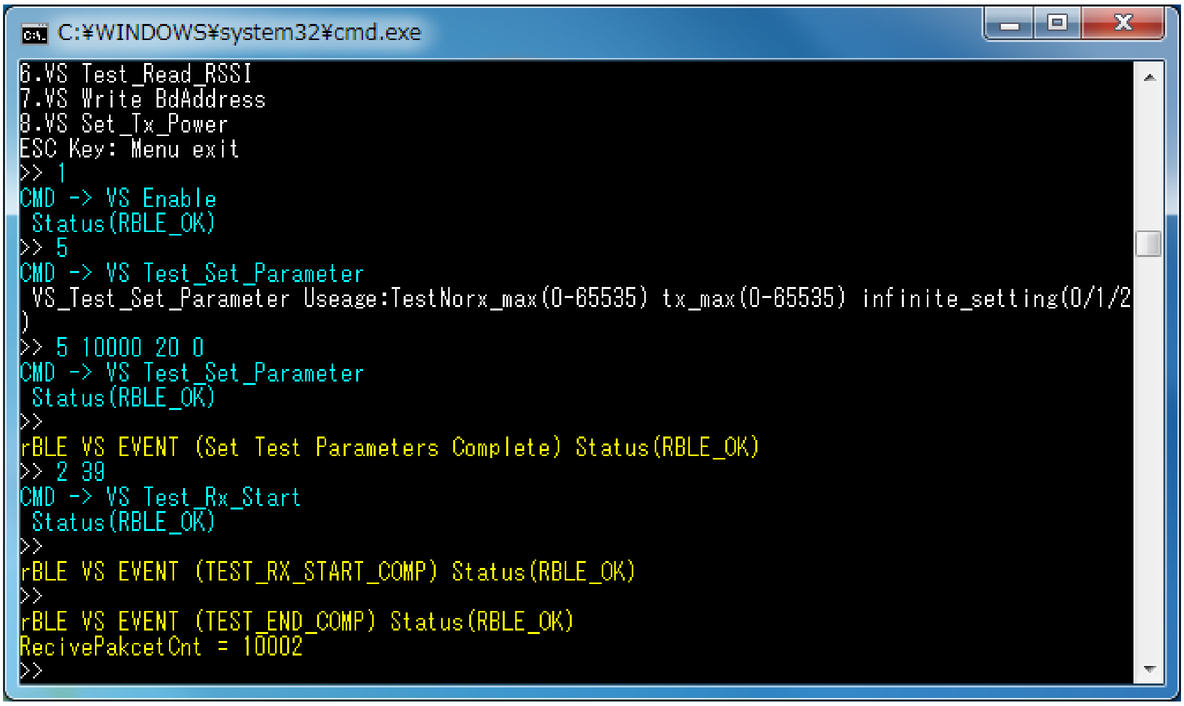
7.4.3. Direct Test Mode (Parameter Set)

Using the VS menu number 5 'VS Test_Set_Parameter', you can set the parameters for the Direct Test Mode (Receiver) and Direct Test Mode (Transmitter) menu items.

```
C:¥WINDOWS¥system32¥cmd.exe
-- BLE Sample Program Vendor Specific Test Menu --
1.VS Enable
2.VS Test_Rx_Start
3.VS Test_Tx_Start
4.VS Test_End
5.VS Test_Set_Parameter
6.VS Test_Read_RSSI
7.VS Write BdAddress
8.VS Set_Tx_Power
ESC Key: Menu exit
>> 1
CMD -> VS Enable
 Status(RBLE_OK)
>> 5
CMD -> VS Test_Set_Parameter
 VS_Test_Set_Parameter Useage:TestNorx_max(0-65535) tx_max(0-65535) infinite_setting(0/1/2
)
>> 5 10000 20 0
CMD -> VS Test_Set_Parameter
 Status(RBLE_OK)
>>
rBLE VS EVENT (Set Test Parameters Complete) Status(RBLE_OK)
>>
```

Figure 7-11 Log of Direct Test Mode Parameter Set

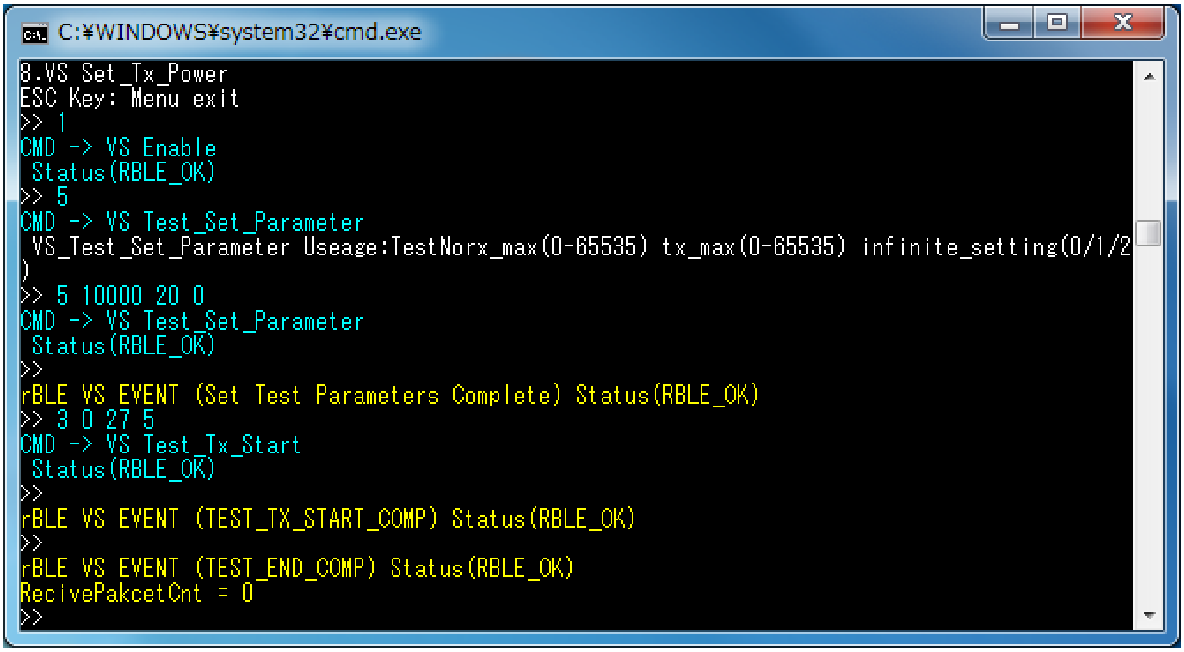
Using the VS menu number 5 'VS Test_Set_Parameter', you can set the number of packet receptions, the number of packet transmissions, enable or disable of burst transfer as arguments. If no argument is given, it displays the usage of this command. Figure 7-11 shows the log of execution, when the number of packet receptions is 10000 times, the number of packet transmissions is 20 time and burst transfer is disabled.



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

Figure 7-12 Log of Direct Test Mode (Receiver) after Direct Test Mode Parameter Set

Figure 7-13 shows the log of the Direct Test Mode (Transmitter) after setting of the above parameters. The direct test mode has automatically finished after sending 20 packets.



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

Figure 7-13 Log of Direct test mode (transmitter) after Direct Test Mode Parameter Set

7.5. Sample Custom Profile

This section explains the Sample Custom Profile (SCP) by using the GATT API below.

To use Sample Custom Profile (SCP), add "USE_SAMPLE_PROFILE" to the macro definition in the compile option of a project.

7.5.1. Sample Custom Profile Specification

Sample Custom Profile (SCP) defines two roles: Client Role and Server Role.

Table 7-3 shows the service characteristics of the SCP.

Table 7-3 Sample Custom Profile Characteristic/Descriptor

Characteristic Name	Properties	Format	Description
Notify Characteristic	Notify	uint8_t[ ]	This characteristic is used to send any notification. The length of notification is from 0 to 20 bytes and can be specified by the Notify Length Characteristic.
Notify Characteristic - Client Characteristic Configuration descriptor	Read/Write	uint16_t	This characteristic descriptor is used to specify ON/OFF of notification.
Indicate Characteristic	Indicate	uint8_t[ ]	This characteristic is used to send any indication. The length of indication is from 0 to 20 bytes and can be specified by the Indication Length Characteristic.
Indicate Characteristic - Client Characteristic Configuration descriptor	Read/Write	uint16_t	This characteristic descriptor is used to specify ON/OFF of indication.
Interval Characteristic	Read/Write	uint16_t	This characteristic is used to specify the transmit interval of indication/notification. (unit: 10 ms)
Notify Length Characteristic	Read/Write	uint8_t	This characteristic is used to specify the transmit data size of notification.
Indicate Length Characteristic	Read/Write	uint8_t	This characteristic is used to specify the transmit data size of indication.

7.5.2. File Structure Corresponding to Sample Custom Profile

The following figure shows the file structure corresponding to the sample custom profile.

```
Renesas
└ BLE_Software_Ver_X_XX
   ├ BLE_Sample                                   Sample program folder for PC
   │ └ src
   │    └ rBLE                                    BLE sample program folder
   │       └ src
   │          ├ include
   │          │ ├ rble_api_custom.h               Custom profile additional API header file
   │          │ └ rble_app.h                      Sample program header file
   │          ├ sample_profile                    Sample profile folder
   │          │ ├ db_handle.h                     Attribute database handles header file
   │          │ └ scp                             Sample custom profile folder
   │          │   ├ scpc.c                        Sample custom profile client file
   │          │   └ scps.c                        Sample custom profile server file
   │          └ sample_app
   │             ├ rble_sample_app.c              Sample program file
   │             └ rble_sample_custom.c           Sample program file (Sample Custom Profile)
   └ RL78_G1D                                     BLE software folder for BLE MCU
      └ Project_Source
         ├ rBLE                                   rBLE folder
         │ └ src
         │    ├ include
         │    │ ├ rble_api_custom.h               Custom profile additional API header file
         │    │ └ rble_app.h                      Sample program header file
         │    ├ sample_profile                    Sample profile folder
         │    │ └ scp                             Sample custom profile folder
         │    │   ├scpc.c                         Sample custom profile client file
         │    │   └scps.c                         Sample custom profile server file
         │    └ sample_app
         │       ├ rble_sample_app.c              Sample program file
         │       └ rble_sample_custom.c           Sample program file (Sample Custom Profile)
         └ renesas
            └ src
               └ arch
                  └ rl78
                     ├ prf_config.c               Parameter file for profile
                     ├ prf_config.h               Parameter header file for profile
                     ├ prf_sel.h                  Profile selection configuration header file
                     └ db_handle.h                Attribute database handles header file
```

7.5.3. API Functions defined for Sample Custom Profile

This section describes the API functions defined for the SCP (Sample Custom Profile) in detail.

7.5.3.1. RBLE_SCP_Clinet_Enable

RBLE_STATUS RBLE_SCP_Client_Enable (uint16_t conhdl, uint8_t con_type, RBLE_SCS_CONTENT *scs, RBLE_SCPC_EVENT_HANDLER call_back)	
This function is used to enable the SCP Client role. When connecting to the SCP Server device for the first time, set con_type to RBLE_SCP_CON_CFG, and perform the configuration connection to discover service on the SCP Server device. The result is notified by the client role enable completion event RBLE_SCP_EVENT_CLIENT_ENABLE_COMP, save the obtained service information at this time. When connecting to the SCP Server device for the second or subsequent time, set con_type to RBLE_SCP_CON_NORMAL, and perform the normal connection by using saved service information. The service discovery is skipped and the Client role can be enabled in shorter time.	
Parameters:	
conhdl	Connection handle
con_type	Connection type
scs	SCP handle information (This parameter is valid if setting RBLE_SCP_CON_NORMAL to con_type.)
call_back	Callback function for event notification
Return:	
RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the SCP Client is not “Disabled”)

7.5.3.2. RBLE_SCP_Clinet_Disable

RBLE_STATUS RBLE_SCP_Client_Disable (uint16_t conhdl)	
This function is used to disable the SCP Client role. The result is notified by the client role disable completion event RBLE_SCP_EVENT_CLIENT_DISABLE_COMP.	
Parameters:	
conhdl	Connection handle
Return:	
RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the SCP Client is not “Enabled”)

7.5.3.3. RBLE_SCP_Clinet_Read_Char

RBLE_STATUS RBLE_SCP_Client_Read_Char (uint16_t conhdl, uint8_t char_code)		
This function is used to read characteristic value or descriptor specified by char_code. The result is notified by the read characteristic response event RBLE_SCP_EVENT_CLIENT_READ_CHAR_RESPONSE.		
Parameters:		
	conhdl	Connection handle
	char_code	Characteristic value or configuration descriptor to be read: RBLE_SCP_SCS_NTF_CFG Read Notify ClientConfiguration descriptor RBLE_SCP_SCS_IND_CFG Read Indicate ClientConfiguration descriptor RBLE_SCP_SCS_INTERVAL Read Interval characteristic value RBLE_SCP_SCS_NTF_LEN Read Notify Length characteristic value RBLE_SCP_SCS_IND_LEN Read Indicate Length characteristic value
Return:		
	RBLE_OK	Success
	RBLE_PARAM_ERR	Failure (Wrong parameter)
	RBLE_STATUS_ERROR	Failure (The state of the SCP Client is not “Enabled”)

7.5.3.4. RBLE_SCP_Clinet_Write_Char

RBLE_STATUS RBLE_SCP_Client_Write_Char (uint16_t conhdl, uint8_t char_code, uint8_t *write_value)		
This function is used to write characteristic value or descriptor specified by char_code. The result is notified by the write characteristic response event RBLE_SCP_EVENT_CLIENT_WRITE_CHAR_RESPONSE.		
Parameters:		
	conhdl	Connection handle
	char_code	Characteristic value or configuration descriptor to be written: RBLE_SCP_SCS_NTF_CFG Write Notify ClientConfiguration descriptor RBLE_SCP_SCS_IND_CFG Write Indicate ClientConfiguration descriptor RBLE_SCP_SCS_INTERVAL Write Interval characteristic value RBLE_SCP_SCS_NTF_LEN Write Notify Length characteristic value RBLE_SCP_SCS_IND_LEN Write Indicate Length characteristic value
Return:		
	RBLE_OK	Success
	RBLE_PARAM_ERR	Failure (Wrong parameter)
	RBLE_STATUS_ERROR	Failure (The state of the SCP Client is not “Enabled”)

7.5.3.5. RBLE_SCP_Server_Enable

<table>
<tr><td colspan="4">RBLE_STATUS RBLE_SCP_Server_Enable (uint16_t conhdl, uint8_t con_type,
RBLE_SCP_SERVER_PARAM *param, RBLE_SCPS_EVENT_HANDLER call_back)</td></tr>
<tr><td colspan="4">This function is used to enable the SCP Server role.
If the Client will write the notification/indication configuration descriptor later, set RBLE_SCP_CON_CFG to the con_type and perform the configuration connection.
If the Server writes (initializes) the notification/indication configuration descriptor, set RBLE_SCP_CON_NORMAL to the con_type, set the initial value to the param and perform the normal connection.
The result is notified by the server role enable completion event
RBLE_SCP_EVENT_SERVER_ENABLE_COMP.</td></tr>
<tr><td colspan="4">Parameters:</td></tr>
<tr><td></td><td>Conhdl</td><td colspan="2">Connection handle</td></tr>
<tr><td></td><td>con_type</td><td colspan="2">Connection type</td></tr>
<tr><td></td><td rowspan="3">Param</td><td colspan="2">Initial value (This parameter is valid if the con_type is RBLE_SCP_CON_NORMAL)</td></tr>
<tr><td></td><td>data_ntf_en</td><td>Initial value for Notify ClientConfiguration descriptor</td></tr>
<tr><td></td><td>data_ind_en</td><td>Initial value for Indicate ClientConfiguration descriptor</td></tr>
<tr><td></td><td>call_back</td><td colspan="2">Callback function for event notification</td></tr>
<tr><td colspan="4">Return:</td></tr>
<tr><td></td><td>RBLE_OK</td><td colspan="2">Success</td></tr>
<tr><td></td><td>RBLE_PARAM_ERR</td><td colspan="2">Failure (Wrong parameter)</td></tr>
<tr><td></td><td>RBLE_STATUS_ERROR</td><td colspan="2">Failure (The state of the SCP Server is not “Disabled”)</td></tr>
</table>

7.5.3.6. RBLE_SCP_Server_Disable

<table>
<tr><td colspan="3">RBLE_STATUS RBLE_SCP_Server_Disable(uint16_t conhdl,)</td></tr>
<tr><td colspan="3">This function is used to disable the SCP Server role.
The result is notified by the server role disable completion event
RBLE_SCP_EVENT_SERVER_DISABLE_COMP.</td></tr>
<tr><td colspan="3">Parameters:</td></tr>
<tr><td></td><td>conhdl</td><td>Connection handle</td></tr>
<tr><td colspan="3">Return:</td></tr>
<tr><td></td><td>RBLE_OK</td><td>Success</td></tr>
<tr><td></td><td>RBLE_PARAM_ERR</td><td>Failure (Wrong parameter)</td></tr>
<tr><td></td><td>RBLE_STATUS_ERROR</td><td>Failure (The state of the SCP Server is not “Enabled”)</td></tr>
</table>

7.5.3.7. RBLE_SCP_Server_Send_Notify

RBLE_STATUS RBLE_SCP_Server_Send_Notify (uint16_t conhdl, RBLE_SCP_NOTIFY_INFO *notify_info)		
This function is used for the Server to send the notification data. The result is notified by the server role send notification completion event RBLE_SCP_EVENT_SERVER_SEND_NOTIFY_COMP.		
Parameters:		
conhdl	Connection handle	
notify_info	Notification data	
	data_len	Data size
	data[ ]	Data
Return:		
RBLE_OK	Success	
RBLE_PARAM_ERR	Failure (Wrong parameter)	
RBLE_STATUS_ERROR	Failure (The state of the SCP Server is not “Enabled”)	

7.5.3.8. RBLE_SCP_Server_Send_Indicate

RBLE_STATUS RBLE_SCP_Server_Send_Indicate (uint16_t conhdl, RBLE_SCP_IND_INFO *ind_info)		
This function is used for the Server to send the indication data. The result is notified by the server role send indication completion event RBLE_SCP_EVENT_SERVER_SEND_IND_COMP.		
Parameters:		
conhdl	Connection handle	
ind_info	Indication data	
	data_len	data_len
	data[ ]	data[ ]
Return:		
RBLE_OK	Success	
RBLE_PARAM_ERR	Failure (Wrong parameter)	
RBLE_STATUS_ERROR	Failure (The state of the SCP Server is not “Enabled”)	

7.5.4. Events defined for Sample Custom Profile

This section describes the events defined for the SCP (Sample Custom Profile) in detail

Table 7-4 Events Used by the SCP

Role	Event Name	Description	Parameter Structure
Server	RBLE_SCP_EVENT_SERVER _ENABLE_COMP	Enable Completion Event	struct RBLE_SCP_Server_Enable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; } server_enable;
	RBLE_SCP_EVENT_SERVER _DISABLE_COMP	Disable Completion Event	struct RBLE_SCP_Server_Disable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; RBLE_SCP_SERVER_PARAM server_info; } server_disable;
	RBLE_SCP_EVENT_SERVER _ERROR_IND	Error Indication Event (Unused)	struct RBLE_SCP_Server_Error_Ind_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }error_ind;
	RBLE_SCP_EVENT_SERVER _SEND_NOTIFY_COMP	Notification Send Completion Event	struct RBLE_SCP_Server_Send_Notify_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }send_notify;
	RBLE_SCP_EVENT_SERVER _SEND_IND_COMP	Indication Send Completion Event	struct RBLE_SCP_Server_Send_Indicate_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }send_ind;
	RBLE_SCP_EVENT_SERVER _CHG_INDNTF_IND	Client Configuration Changed Event	struct RBLE_SCP_Server_Cfg_Indntf_Ind_t{ uint16_t conhdl; uint8_t char_code; uint8_t reserved; uint16_t cfg_val; }cfg_indntf;
	RBLE_SCP_EVENT_SERVER _CHG_CHAR_IND	Characteristic Changed Event	struct RBLE_SCP_Server_Write_Chara_Ind_t{ uint16_t conhdl; uint8_t char_code; uint8_t reserved; uint8_t value[RBLE_SCPC_WRITE_CHAR_MAX]; }write_char;
	RBLE_SCP_EVENT_SERVER _COMMAND_DISALLOWED_IND	Command Disallowed Notification Event (Unused)	struct RBLE_SCP_Server_Command_Disallowed_Ind_t{ RBLE_STATUS status; uint8_t reserved; uint16_t opcode; }cmd_disallowed_ind;

Client	RBLE_SCP_EVENT_CLIENT _ENABLE_COMP	Enable Completion Event	struct RBLE_SCP_Client_Enable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; RBLE_SCS_CONTENT scs; }client_enable;
	RBLE_SCP_EVENT_CLIENT _DISABLE_COMP	Disable Completion Event	struct RBLE_SCP_Client_Disable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }client_disable;
	RBLE_SCP_EVENT_CLIENT _ERROR_IND	Error Indication Event (Unused)	struct RBLE_SCP_Client_Error_Ind_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; }error_ind;
	RBLE_SCP_EVENT_CLIENT _NOTIFY	Notification Received Event	struct RBLE_SCP_Client_Notify_Ind_t{ uint16_t conhdl; uint8_t data_len; uint8_t data[]; }notify;
	RBLE_SCP_EVENT_CLIENT _INDICATE	Indicatation Received Event	struct RBLE_SCP_Client_Indicate_Ind_t{ uint16_t conhdl; uint8_t data_len; uint8_t data[]; }ind;
	RBLE_SCP_EVENT_CLIENT _READ_CHAR_RESPONSE	Read Characteristic Response Event	struct RBLE_SCP_Client_Read_Char_Response_t{ uint16_t conhdl; uint8_t att_code; RBLE_ATT_INFO_DATA data; }rd_char_resp;
	RBLE_SCP_EVENT_CLIENT _WRITE_CHAR_RESPONSE	Write Characteristic Response Event	struct RBLE_SCP_Client_Write_Char_Response_t{ uint16_t conhdl; uint8_t att_code; }wr_char_resp;
	RBLE_SCP_EVENT_CLIENT _COMMAND_DISALLOWED_IND	Command Disallowed Notification Event (Unused)	struct RBLE_SCP_Client_Command_Disallowed_Ind_t{ RBLE_STATUS status; uint8_t reserved; uint16_t opcode; }cmd_disallowed_ind;

7.5.5. Usage of the Sample Program for Sample Custom Profile

This section explains usage of the Sample Program for Sample Custom Profile (SCP).

By default, the sample program for Server role is intended to run in the Embedded configuration and the sample program for Client role is intended to run in the Modem configuration. Therefore, the Sample program for Server role operates without any external command control.

Refer to Usage of the Sample Program for Server role in detail.

If you want to run the Sample Program for Client role in Embedded configuration, disable the definition of USE_CUSTOM_DEMO macro in prf_sel.h file.

7.5.5.1. Usage of the Sample Program for Client role

This section explains usage of the Sample Program for Client role.

After connecting to the Server device using GAP command, the following steps allow you to use commands for SCP.

(1) Select Profile Test (In case of Figure 7-14, Enter 2)

Figure 7-14 Initial Menu (the Sample Program for Client role)

(2) Select Sample Custom Profile (In case of Figure 7-15, Enter 7)

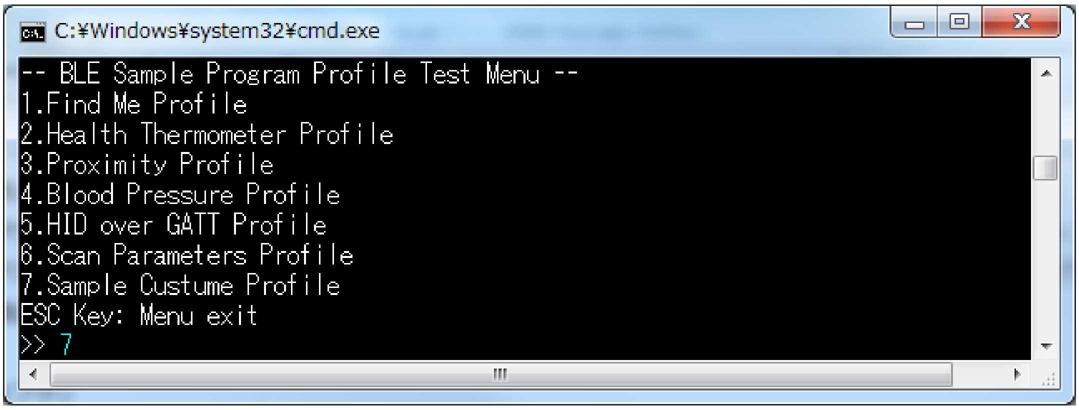


Figure 7-15 Profile Test Menu (the Sample Program for Client role)

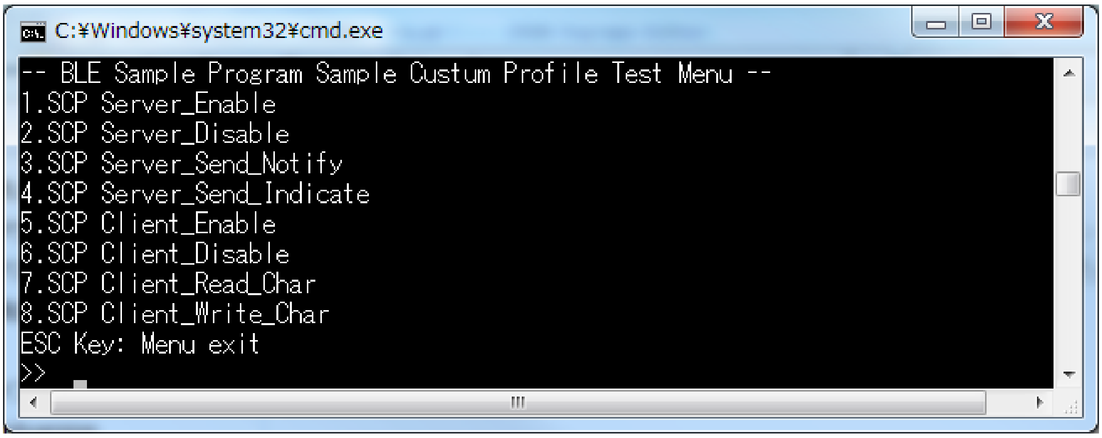


Figure 7-16 Sample Custom Test Menu (the Sample Program for Client role)

The following table explains commands provided by the Sample Program for Client role.

Command No.	Operation	Parameters	Description	Example
1	Server Enable	-	Controls the Server.	-
2	Server Disable	-	Controls the Server.	-
3	Server Send Notify	-	Controls the Server.	-
4	Server Send Indicate	-	Controls the Server.	-
5	Client Enable	-	Enables Client role, by calling RBLE_SCP_Client_Enable API. (This command always performs configuration connection using SCP_CON_CFG parameter.)	5
6	Client Disable	-	Disables Client role, by calling RBLE_SCP_Client_Disable API.	6
7	Client Read Char	char_code	Reads characteristic value, by calling RBLE_SCP_Client_Read_Char API. The parameter specifies characteristic value or characteristic descriptor to be read. 0: Client Characteristic Configuration of Notify (RBLE_SCP_SCS_NTF_CFG) 1: Client Characteristic Configuration of Indicate (RBLE_SCP_SCS_IND_CFG) 2: Characteristic value of Interval (RBLE_SCP_SCS_INTERVAL) 3: Characteristic value of Notify Length (RBLE_SCP_SCS_NTF_LEN) 4: Characteristic value of Indicate Length (RBLE_SCP_SCS_IND_LEN)	7 2
8	Client Write Char	char_code data	Writes characteristic value, by calling RBLE_SCP_Client_Write_Char API. The first parameter specifies characteristic value or characteristic descriptor to be writen, and the second parameter specifies data to be written. 0: Client Characteristic Configuration of Notify (RBLE_SCP_SCS_NTF_CFG) 1: Client Characteristic Configuration of Indicate (RBLE_SCP_SCS_IND_CFG) 2: Characteristic value of Interval (RBLE_SCP_SCS_INTERVAL) 3: Characteristic value of Notify Length (RBLE_SCP_SCS_NTF_LEN) 4: Characteristic value of Indicate Length (RBLE_SCP_SCS_IND_LEN)	8 1 2

7.5.5.2. Usage of the Sample Program for Server role

This section explains usage of the Sample Program for Server role.

When you power on the Server device, it waits for a connect request from the Client device automatically.

When the Client device has been connected to the Server device, the Server device enables the Server role of SCP automatically and becomes ready to accept requests from the Client device.

Write the Notify and/or Indicate characteristic from the Client device. Then, if you push SW2 switch on the RL78/G1D evaluation board, it takes effect and the Server starts sending the notification and /or indication. If you push the SW2 switch again, the Server stops sending.

The notification and/or Indication are sent depending on the Interval, Notify Length and Indicate Length characteristics respectively.

Note that the unit of Interval characteristic value is 10 milliseconds.

7.6. Simple Sample Profile

This section describes about the simple sample profile. To use the profile, you need to add "USE_SIMPLE_SAMPLE_PROFILE" macro definition to a project configuration.

7.6.1. Characteristic Specification

Table 7-5 shows the simple sample profile characteristic specification.

Table 7-5 Simple Sample Profile Characteristics

Characteristic Name	Properties	Format	Note
Switch State Characteristic UUID: 5BC18D80-A1F1-40AF-9043-C43692C18D7A	Notify	uint8_t	Notify SW4 state (PUSH/RELEASE). 0x00 is RELEASE, 0x01 is PUSH.
- Client Characteristic Configuration	Read/Write	uint16_t	Enable/Disable Notify.
LED Control Characteristic UUID: 5BC143EE-A1F1-40AF-9043-C43692C18D7A	Read/Write	uint8_t	Set/Get LED4 state (ON/OFF). 0x00 is OFF, 0x01 is ON.

7.6.2. File Structure

Following shows the simple sample profile related files.

```
Renesas
└ BLE_Software_Ver_X_XX
   └ RL78_G1D
      └ Project_Source
         ├ rBLE
         │ └ src
         │    └ sample_simple                      Simple Sample Profile, Simple Sample Program folder
         │       ├ sam                             Simple Sample Profile folder
         │       │ ├ sams.c                        Simple Sample Profile source file
         │       │ └ sams.h                        Simple Sample Profile header file
         │       ├ console.c                       Console Driver source file
         │       ├ console.h                       Console Driver header file
         │       ├ rble_sample_app_peripheral.c    Simple Sample Program source file
         │       └ rble_sample_app_peripheral.h    Simple Sample Program header file
         └ renesas
            └ src
               └ arch
                  └ rl78
                     ├ prf_config.c                Parameter source file for profile
                     ├ prf_config.h                Parameter header file for profile
                     ├ ke_conf_simple.c            RWKE task definition file
                     └ db_handle.h                 Attribute database handles header file
```

7.6.3. Details of Simple Sample Profile

The simple sample profile is identical with "Embedded Configuration Sample Application (r01an3319)" Peripheral role sample application. Refer "Embedded Configuration Sample Application (r01an3319)" application note for the details.

7.7. Sample Program for the Direct Test Mode with RF Tester

The BLE software includes the Sample Program that supports the Direct Test Mode (DTM).

The RL78/G1D evaluation board and the RF Conformance Tester, which is used for Bluetooth Qualification Test, are connected through a 2-wire UART interface.

To use the Sample Program for DTM:
(1) Change the following macro definition from 0 to 1
(2) Rebuild (recompile) the Sample Program.

```
#define __DTM2WIRE_UART_USE__          0
```

The file structure corresponding to this Sample Program is shown below.

```
Renesas
└ BLE_Software_Ver_X_XX
   └ RL78_G1D                                   BLE software folder for the BLE MCU
      └ Project_Source
         ├ bleip                                BLE stack folder
         │ └ src
         │    └ rwble
         │       └ rwble_config.h               BLE stack configuration header file
         └ renesas
            └ src
               ├ arch
               │ └ rl78
               │    ├ arch_main.c               BLE software main file
               │    └ ke_conf.c                 RWKE task management file
               └ driver
                  ├ DTM2Wire
                  │ ├ DTM2Wire.c                2-wire UART Direct Test Mode driver file
                  │ └ DTM2Wire.h                2-wire UART Direct Test Mode driver header file
                  └ uart
                     ├ uart.c                   UART driver file
                     └ uart.h                   UART driver header file
```

The Sample Program for DTM automatically determines its operating mode immediately after the system reset. There are two operating modes: DTM mode and Normal operating mode. When this sample program starts with DTM mode, the baud rate of 2-wire UART interface is set 9,600 baud.

Determination conditions of the operating mode is different depending on the configurations (Embedded or Modem).

(1) In the Modem configuration

The Sample Program operates in DTM mode, if the first data is received successfully (without any errors) through the two-wire UART interface after the system reset. Otherwise the Sample Program operates in normal operation mode.

(2) In the Embedded configuration

The Sample Program operates in DTM mode, if you power on the RL78/G1D evaluation board while pressing SW2 switch on the board. Otherwise the Sample Program operates in normal operation mode.

The startup sequence in each configuration is shown in Figure 7-17, Figure 7-18 and Figure 7-19.

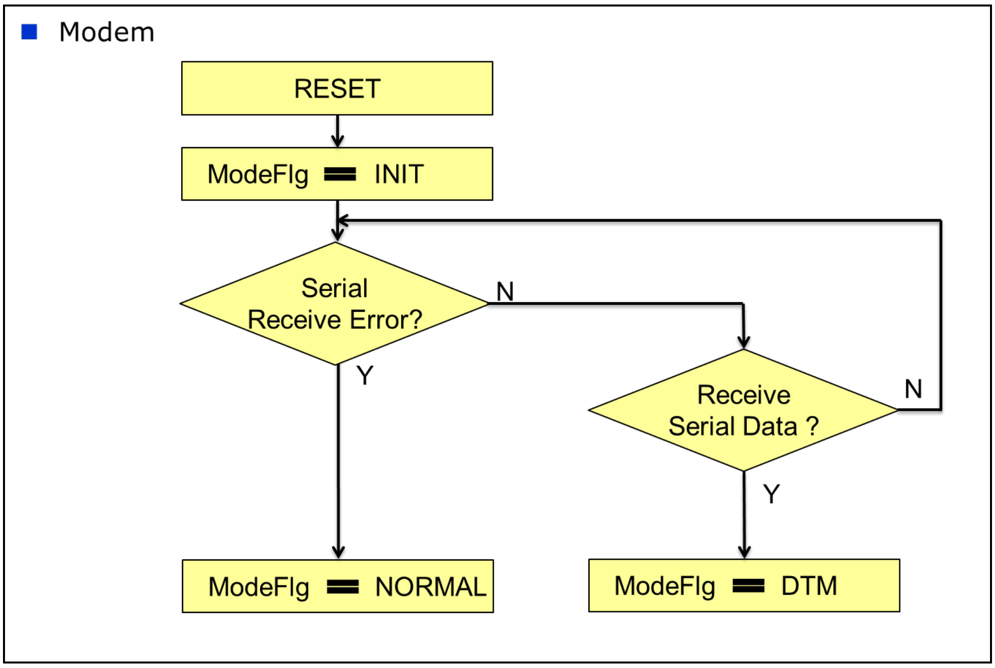


Figure 7-17 Startup sequence in Modem configuration

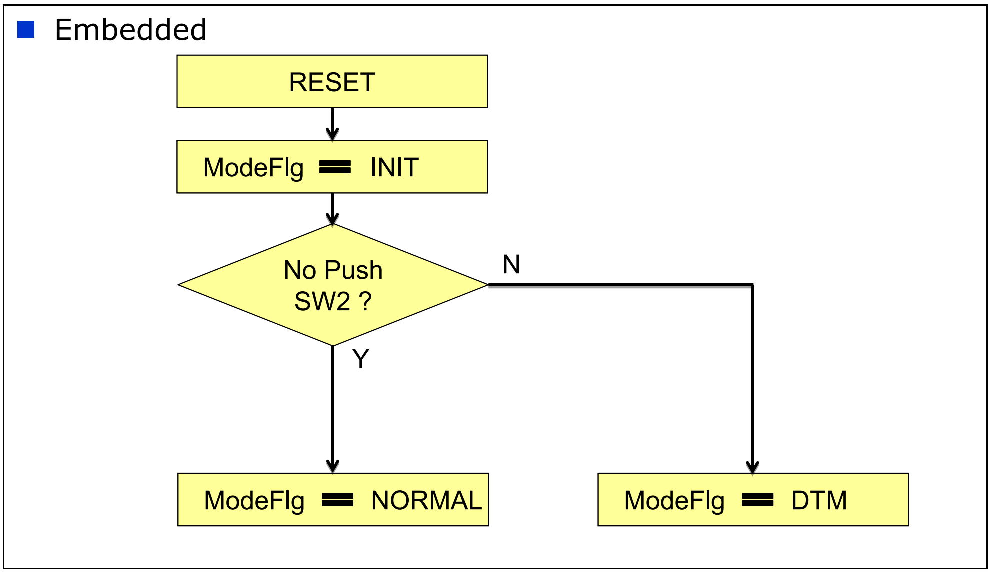


Figure 7-18 Startup sequence in Embedded configuration

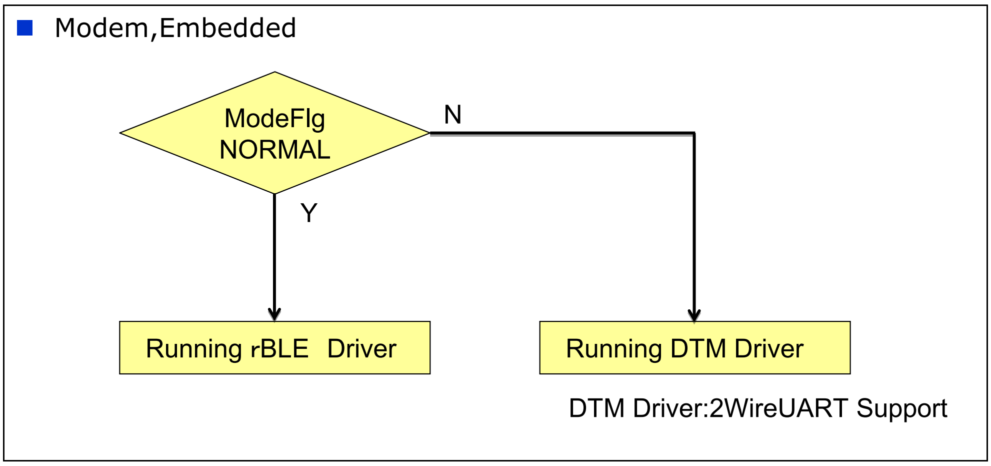


Figure 7-19 Operation after operating mode determination

7.8. Printf program in the Embedded configuration

At the Sample program in the Embedded Configuration, an access of Standard IO is materialized by the "console.c".

When "printf" function is used, "printf" function in a standard library is not called, but the "Printf" function defined in the "console.c" is called by the following macro definitions.

```
#define  printf        Printf
```

The "Printf" function writes a formatted string to the buffer and outputs this buffer to serial port. The size of this buffer is set by the following macro definitions.

```
#define   STREAM_MEMORY_MAX_LINE_SIZE     80
```

Therefore if you need output the data which is over 80 bytes, you should adjust buffer size by this macro definition.

7.9. FW Update Sample Program

This section explains the FW Update Sample Program.

In FW Update, One device sends FW Update data(Sender device). The other device receives FW Update data and update FW(Receiver device).

The operation image of FW Update is shown in Figure 7-20.

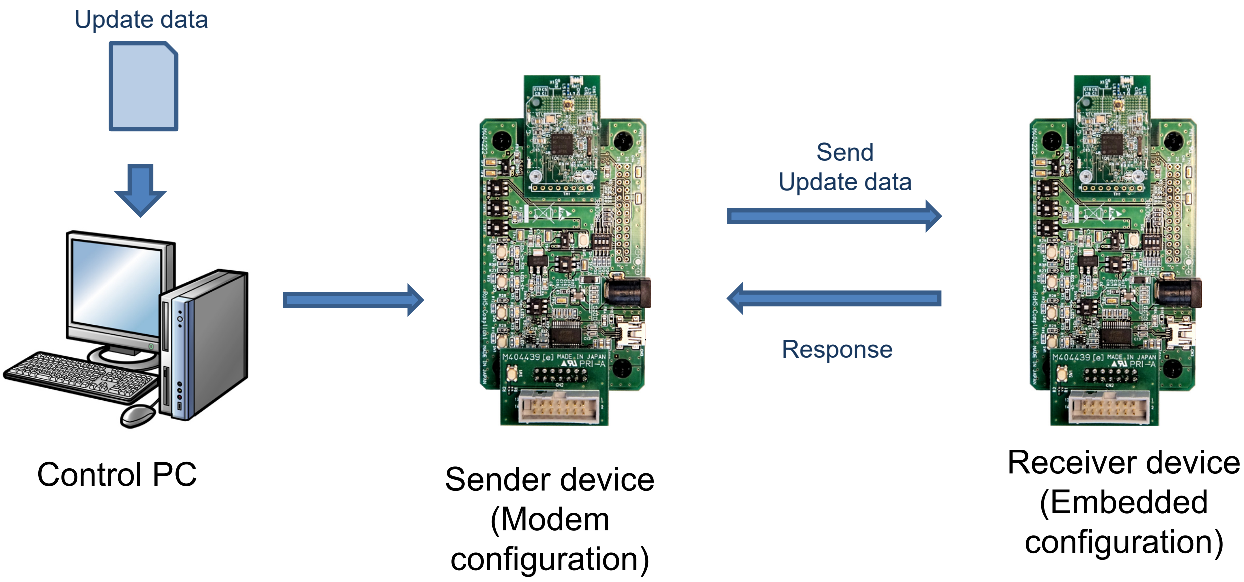


Figure 7-20 Operation image of FW Update

In the FW Update Sample Program, the Sender device is Modem configuration and the Receiver device is Embedded configuration. Table 7-1

7.9.1. FW Update Profile Specification

FW Update Profile defines two roles: Sender Role and Receiver Role.

Table 7-6 shows the service characteristics of the FW update profile.

Table 7-6 FW Update Profile Characteristic/Descriptor

Characteristic Name	Properties	format	Description
Data Control Characteristic	Write	uint8_t[ ]	Control information of data transmission is written by the Write Request.
Data Characteristic	Write without Response	uint8_t[ ]	Update data of 1 to 20 bytes are written by the Write Command.

7.9.2. File Structure Corresponding to FW Update Profile

The file structure corresponding to this FW Update Sample Program is shown below.

```
BLE_Software_Ver_X_XX
  ├ BLE_Sample                                  Sample Program folder for PC
  │ ├ src
  │ │ └ rBLE                                    rBLE folder
  │ │   └ src
  │ │     ├ include
  │ │     │ └ rble_api_fwup.h                   FW Update profile header file
  │ │     ├ sample_profile                      Sample profile folder
  │ │     │ └ fwup                              FW Update profile folder
  │ │     │   └ fwups.c                         FW Update profile sender file
  │ │     └ sample_app
  │ │       ├ rble_sample_app.c                 Sample Program file
  │ │       └ rble_fw_up_sender_app.c           Sample Program file for FW Update(Sender)
  │ └ Fwup                                      Sample folder for FW Update
  │   ├ bin                                     Binary data folder
  │   │ ├ ca78k0r                               Folder of Binary which Base hex file that was built with CA78K0R
  │   │ │ ├ RL78_G1D_CE(PXP,FMP,ANP).bin        Binary file for Embedded configuration (PXP/FMP/ANP)
  │   │ │ └ RL78_G1D_CE(HTP,BLP,HRP).bin        Binary file for Embedded configuration (HTP/BLP/HRP)
  │   │ ├ ccrl                                  Folder of Binary which Base hex file that was built with CC-RL
  │   │ │ ├ RL78_G1D_CCE(PXP,FMP,ANP).bin       Binary file for Embedded configuration (PXP/FMP/ANP)
  │   │ │ └ RL78_G1D_CCE(HTP,BLP,HRP).bin       Binary file for Embedded configuration (HTP/BLP/HRP)
  │   │ └ iar_v2                                Folder of Binary which Base hex file that was built with IAR Embedded
  │   │                                         Workbench v2
  │   │   ├ RL78_G1D_IE(PXP,FMP,ANP).bin        Binary file for Embedded configuration (PXP/FMP/ANP)
  │   │   └ RL78_G1D_IE(HTP,BLP,HRP).bin        Binary file for Embedded configuration (HTP/BLP/HRP)
  │   └ hex                                     Hex data folder
  │     ├ Sender                                ROM file for Sender device
  │     │ ├ RL78_G1D_CM(Sender).hex             ROM file for Embedded configuration that was built with CA78K0R
  │     │ ├ RL78_G1D_CCM(Sender).hex            ROM file for Embedded configuration that was built with CC-RL
  │     │ └ RL78_G1D_IM_V2(Sender).hex          ROM file for Embedded configuration that was built with IAR Embedded
  │     │                                       Workbench v2
  │     └ Receiver                              ROM file for Receiver device
  │       ├ ca78k0r                             Folder of ROM file that was built with CA78K0R
  │       │ └ Embedded                          ROM file folder for Embedded configuration
  │       │   ├ RL78_G1D_CE(PXP,FMP,ANP).hex    ROM file for Embedded configuration (PXP/FMP/ANP)
  │       │   └ RL78_G1D_CE(HTP,BLP,HRP).hex    ROM file for Embedded configuration (HTP/BLP/HRP)
  │       ├ ccrl                                Folder of ROM file that was built with CC-RL
  │       │ └ Embedded                          ROM file folder for Embedded configuration
  │       │   ├ RL78_G1D_CCE(PXP,FMP,ANP).hex   ROM file for Embedded configuration (PXP/FMP/ANP)
  │       │   └ RL78_G1D_CCE(HTP,BLP,HRP).hex   ROM file for Embedded configuration (HTP/BLP/HRP)
  │       └ iar_v2                              Folder of ROM file that was built with IAR Embedded Workbench v2
  │         └ Embedded                          ROM file folder for Embedded configuration
  │           ├ RL78_G1D_IE(PXP,FMP,ANP).hex    ROM file for Embedded configuration (PXP/FMP/ANP)
  │           └ RL78_G1D_IE(HTP,BLP,HRP).hex    ROM file for Embedded configuration (HTP/BLP/HRP)
  └ RL78_G1D                                    BLE software folder for the BLE MCU
    └ Project_Source
      └ rBLE                                    rBLE folder
        └ src
          ├ include
          │ └ rble_api_fwup.h                   FW Update profile header file
          ├ sample_profile                      Sample profile folder
          │ └ fwup                              FW Update profile folder
          │   └ fwupr.c                         FW Update profile receiver file
          └ sample_app                          Sample Program file
            └ rble_fw_up_receiver_app.c         Sample Program file for FW Update(Receiver)
```

Procedures of operating the FW Update Sample Program are shown in the following.

(1) Write one of the following HEX file to the RL78/G1D Evaluation Board which operate as the Sender device.

Stored folder: BLE_Software_Ver_X_XX\BLE_Sample\Fwup\hex\Sender

File name:

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

(2) Write the HEX file that are stored in following folder to the RL78/G1D Evaluation Board which operate as the Receiver device.

Stored folder: BLE_Software_Ver_X_XX\BLE_Sample\Fwup\hex\Receiver\<development environment>

CS+ for CA, CX (CA78K0R) : ca78k0r

e^2 studio / CS+ for CC (CC-RL) : ccrl

IAR Embedded Workbench V2 : iar_v2

[Note] Please choose the Hex file suitable for development environment.

(3) Store the FW updates data in the following folder. The FW update data uses what converted HEX file to binary format.

Folder name: BLE_Software_Ver_X_XX\BLE_Sample\project\windows\Exe

[Note] HEX file uses the same development environment as what was written to Receiver device. For example, when you write 'RL78_G1D_CE(PXP,FMP,ANP).hex' to the Receiver device, you store the data which converted 'RL78_G1D_CE(HTP,BLP,HRP).hex' to binary data(RL78_G1D_CE(HTP,BLP,HRP).bin).

Binary data which already converted are stored following folder for sample binary data.

Folder name: BLE_Software_Ver_X_XX\BLE_Sample\Fwup\bin

(4) Start the Sample Program of the Sender device. At this time, the baud rate specifies the 76800bps according to the HEX file written to Sender device. How to start the Sample Program is shown in 5.1.

(5) Start the Sample Program of the Receiver device. How to start the Sample Program is shown in 5.3.

7.9.3. API Functions defined for FW Update Profile

This section describes the API functions defined for the FWUP (FW Update Profile) in detail.

(1) RBLE_FWUP_Sender_Enable

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

This function is used to enable the FWUP Sender role.
When connecting to the FWUP Receiver device for the first time, set con_type to RBLE_FWUP_CON_CFG, and perform the configuration connection to discover service on the FWUP Receiver device.
The result is notified by the Sender role enable completion event RBLE_FWUP_EVENT_SENDER_ENABLE_COMP, save the obtained service information at this time.
When connecting to the FWUP Receiver device for the second or subsequent time, set con_type to RBLE_FWUP_CON_NORMAL, and perform the normal connection by using saved service information. The service discovery is skipped and the Sender role can be enabled in shorter time.

Parameters:

conhdl	Connection handle
con_type	Connection type
fwus	FWUP handle information (This parameter is valid if setting RBLE_FWUP_CON_NORMAL to con_type)
call_back	Callback function for event notification

Return:

RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the FWUP Sender is not “Disabled”)

(2) RBLE_FWUP_Sender_Disable

```
RBLE_STATUS RBLE_FWUP_Sender_Disable ( uint16_t   conhdl )
```

This function is used to disable the FWUP Sender role.
The result is notified by the client role disable completion event RBLE_FWUP_EVENT_SENDER_DISABLE_COMP.

Parameters:

conhdl	Connection handle

Return:

RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the FWUP Sender is not “Enabled”)

(3) RBLE_FWUP_Sender_Write_Data_Cntl

```
RBLE_STATUS RBLE_FWUP_Sender_Write_Cntl ( uint16_t   conhdl,
                                          uint8_t    type,
                                          uint8_t    block_num,
                                          uint16_t   data_size )
```

Data Control Characteristic is set.
The block_num and the data_size are effective only if the type is set to RBLE_FWUP_DATA_SEND_START.
The result is notified by the write characteristic data response event RBLE_FWUP_EVENT_SENDER_WRITE_CHAR_RES.

Parameters:

conhdl	Connection handle
type	Specifying a control command type RBLE_FWUP_DATA_SEND_START Data transmission start RBLE_FWUP_DATA_SEND_COMP Data transmission completion (with specified size) RBLE_FWUP_DATA_CHECK_WRITE Data write confirmation RBLE_FWUP_DATA_SEND_FINISH Data transmission completion (all data) RBLE_FWUP_DATA_CHECK_UPDATE FW Update completion confirmation
block_num	Specifying the write block number of the code flash (0 to 255) This parameter is effective only if the type is set to BLE_FWUP_DATA_SEND_START.
data_size	Specifying the write data size to the code flash (4 to 1024 in increments of 4 bytes) This parameter is effective only if the type is set to BLE_FWUP_DATA_SEND_START.

Return:

RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the FWUP Sender is not “Enabled”)

(4) RBLE_FWUP_Sender_Write_Data

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

Data Characteristic is set.

Parameters:

conhdl	Connection handle
**data*	Specifying the beginning address of the write data to Receiver
data_size	Specifying the setting data size (1 to 20 bytes)

Return:

RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the FWUP Sender is not “Enabled”)

(5) RBLE_FWUP_Receiver_Enable

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

This function is used to enable the FWUP Receiver role.
The result is notified by the Receiver role enable completion event
RBLE_FWUP_EVENT_RECEIVER_ENABLE_COMP.

Parameters:

conhdl	Connection handle
call_back	Callback function for event notification

Return:

RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the FWUP Receiver is not “Disabled”)

(6) RBLE_FWUP_Receiver_Disable

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

This function is used to disable the FWUP Receiver role.
The result is notified by the Receiver role disable completion event
RBLE_FWUP_EVENT_RECEIVER_DISABLE_COMP.

Parameters:

conhdl	Connection handle

Return:

RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the FWUP Receiver is not “Enabled”)

(7) RBLE_FWUP_Receiver_Send_Data_Cntl_Res

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

This function sends response to write request of Data Control characteristic.
The status is set the result in accordance with the control command that is set in Data Control characteristic.

RBLE_FWUP_DATA_SEND_START	If block number and the size is correct, set to RBLE_OK. Otherwise RBLE_ERR.
RBLE_FWUP_DATA_SEND_COMP and RBLE_FWUP_DATA_SEND_FINISH	If specified size data is received, set to RBLE_OK. Otherwise RBLE_ERR.
RBLE_FWUP_DATA_CHECK_WRITE	If flash write is successfully finished, set to RBLE_OK. Otherwise RBLE_ERR.
RBLE_FWUP_DATA_CHECK_UPDATE	If FW Update is successfully finished, set to RBLE_OK. Otherwise RBLE_ERR.

Parameters:

conhdl	Connection handle	
status	The result for received command	
	RBLE_OK	Success
	RBLE_ERR	Failure

Return:

RBLE_OK	Success
RBLE_PARAM_ERR	Failure (Wrong parameter)
RBLE_STATUS_ERROR	Failure (The state of the FWUP Receiver is not “Enabled”)

7.9.4. Events defined for FW Update Profile

This section describes the events defined for the FWUP (FW Update Profile) in detail.

Table 7-7 Events Used by the FWUP

Role	Event Name	Description	Parameter Structure
Receiver	RBLE_FWUP_EVENT_RECEIVER_ENABLE_COMP	Enable Completion Event	struct RBLE_FWUP_Receiver_Enable_t{ uint16_t conhdl; RBLE_STATUS status; }receiver_enable;
	RBLE_FWUP_EVENT_RECEIVER_DISABLE_COMP	Disable Completion Event	struct RBLE_FWUP_Receiver_Disable_t{ uint16_t conhdl; RBLE_STATUS status; }receiver_disable;
	RBLE_FWUP_EVENT_RECEIVER_CHG_DATA_CNTL_IND	Data Control Change Event	struct RBLE_FWUP_Receiver_Chg_Data_Cntl_Ind_t{ uint16_t conhdl; uint8_t type; uint8_t block_num; uint16_t data_size; }data_cntl_ind;
	RBLE_FWUP_EVENT_RECEIVER_CHG_DATA_IND	Data Change Event	struct RBLE_FWUP_Receiver_Chg_Data_Ind_t{ uint16_t conhdl; uint8_t data_size; uint8_t data[RBLE_FWUP_DATA_MAX]; }data_ind;
Sender	RBLE_FWUP_EVENT_SENDER_ENABLE_COMP	Enable Completion Event	struct RBLE_FWUP_Sender_Enable_t{ uint16_t conhdl; RBLE_STATUS status; uint8_t reserved; RBLE_FWUS_CONTENT fwus; }sender_enable;
	RBLE_FWUP_EVENT_SENDER_DISABLE_COMP	Disable Completion Event	struct RBLE_FWUP_Sender_Disable_t{ uint16_t conhdl; RBLE_STATUS status; }sender_disable;
	RBLE_FWUP_EVENT_SENDER_WRITE_CHAR_RES	Write Characteristic Response Event	struct RBLE_FWUP_Sender_Write_Char_Res_t{ uint16_t conhdl; uint8_t att_code; }wr_char_resp;

7.9.5. Usage of the Sample Program for FW Update Profile

When started the Sample Program of the Sender device or the Receiver device according to 7.9.2, the following content is displayed at console.

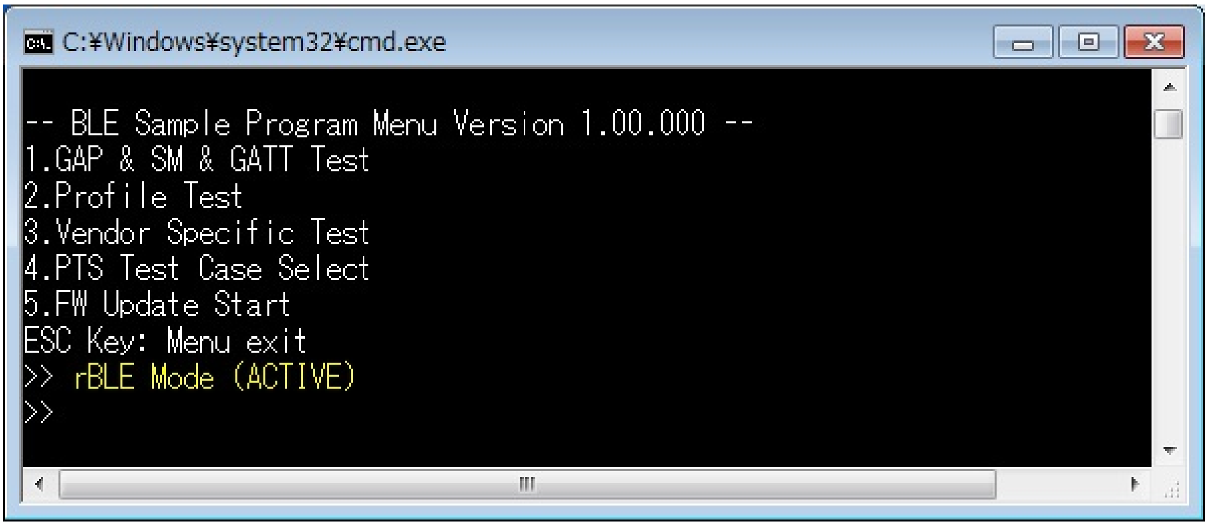


Figure 7-21 Console of Sample Program

[Note] '5.FW Update Start' command is not displayed on a console of the Receiver device.

Procedures of control the Sample Program are shown in the following.

(1) The Sender device(Master) gets BD address of the Receiver device by using procedure of 5.5.

(2) Push the SW2(red frame of Figure 7-22) for the Receiver device become FW Update mode.

[Note]Until FW Update is complete after press the SW2, the Receiver device will not be able to receive command from console.

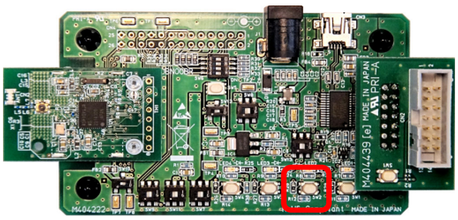


Figure 7-22 Selector switch to the FW Update mode

(3) Send ‘5. FW Update Start’ command to Sender device with binary file num.

Following table is correspondence of file num and file name.

num	file name
0	RL78_G1D_CE(PXP,FMP,ANP).bin
1	RL78_G1D_CE(HTP,BLP,HRP).bin
2	RL78_G1D_IE(PXP,FMP,ANP).bin
3	RL78_G1D_IE(HTP,BLP,HRP).bin
4	RL78_G1D_CCE(PXP,FMP,ANP).bin
5	RL78_G1D_CCE(HTP,BLP,HRP).bin

Figure 7-23 is example of sending binary file of ‘RL78_G1D_CE(PXP,FMP,ANP).bin.

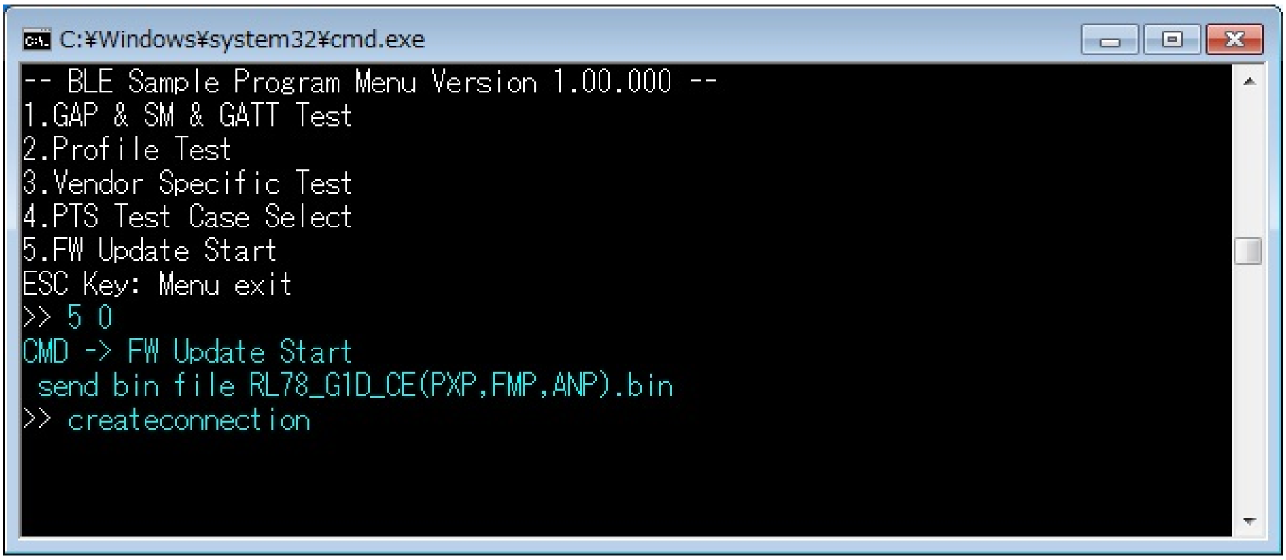


Figure 7-23 Console log when sending ‘FW update Start’ command.(Sender device)

(4) After sending ‘5.FW Update Start’ command, the Sample Program operates automatically until FW Update is completed.

Figure 7-24 is console log when FW Update is in operation.

[Note] While FW Update is in operation, the Sample Program repeats create connection, data send and disconnect.

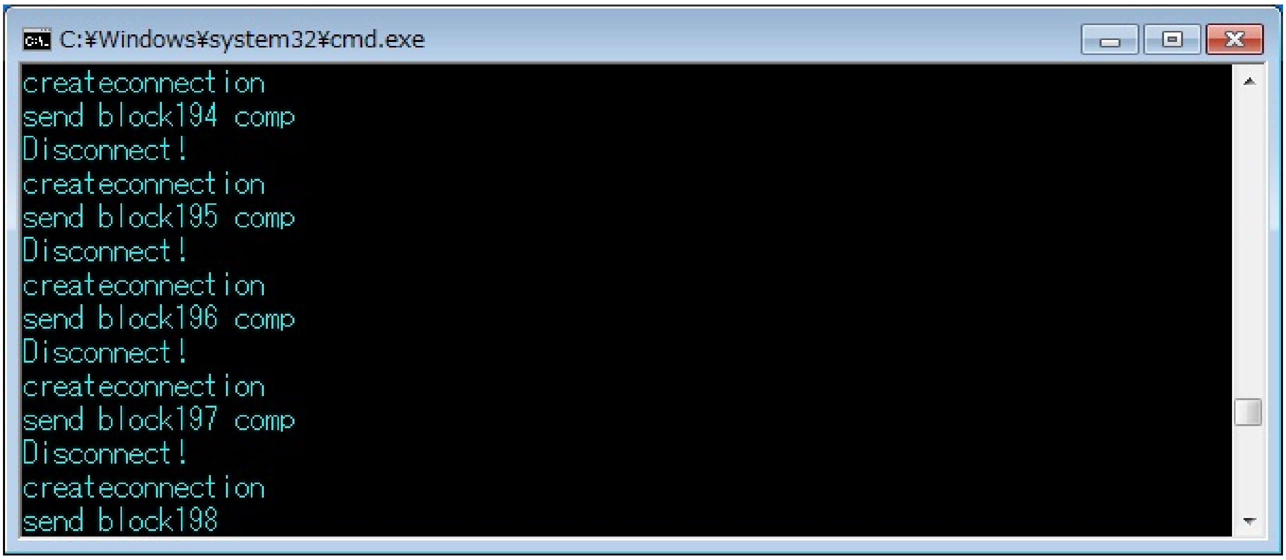


Figure 7-24 Console log when FW Update is in operation.

(5) When FW Update is completed, ‘fw update finish’ is displayed on a console of Sender device.

The Receiver device is reset and can send a command from console.

7.10. Project Setting to use FW Update Sample Program

Procedures of setting project to use FW update sample program are shown in the following.

7.10.1. Receiver device

7.10.1.1. Project Settings of IAR Embedded Workbench V2.20.1

The setup procedures of the project in IAR Embedded Workbench V2.20.1 are shown in the following.

(1) Starting the project of Embedded or Modem configuration.

(2) Select [Project]→[Option]→[C/C++Compiler]→[Preprocessor].

Change the Defined symbol form ‘noUSE_FW_UPDATE_PROFILE’ to ‘USE_FW_UPDATE_PROFILE’.

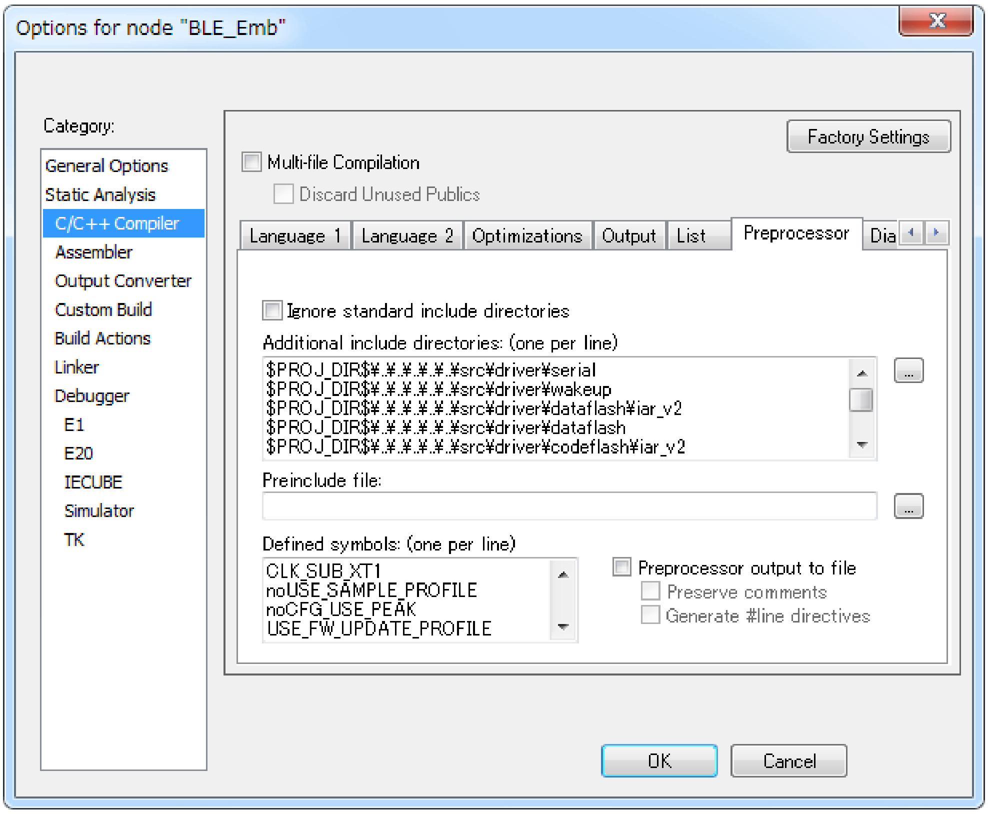


Figure 7-25 Setting of Defined symbols.

(3) Select [Linker]→[Config]

- Embedded Configuration
  Change the linker configuration file from 'lnkr5f11agj.icf' to 'lnkr5f11agj_fw.icf'.
- Modem Configuration
  Change the linker configuration file from 'lnkr5f11agj.icf' to 'lnkr5f11agj_fw_mdm.icf'.

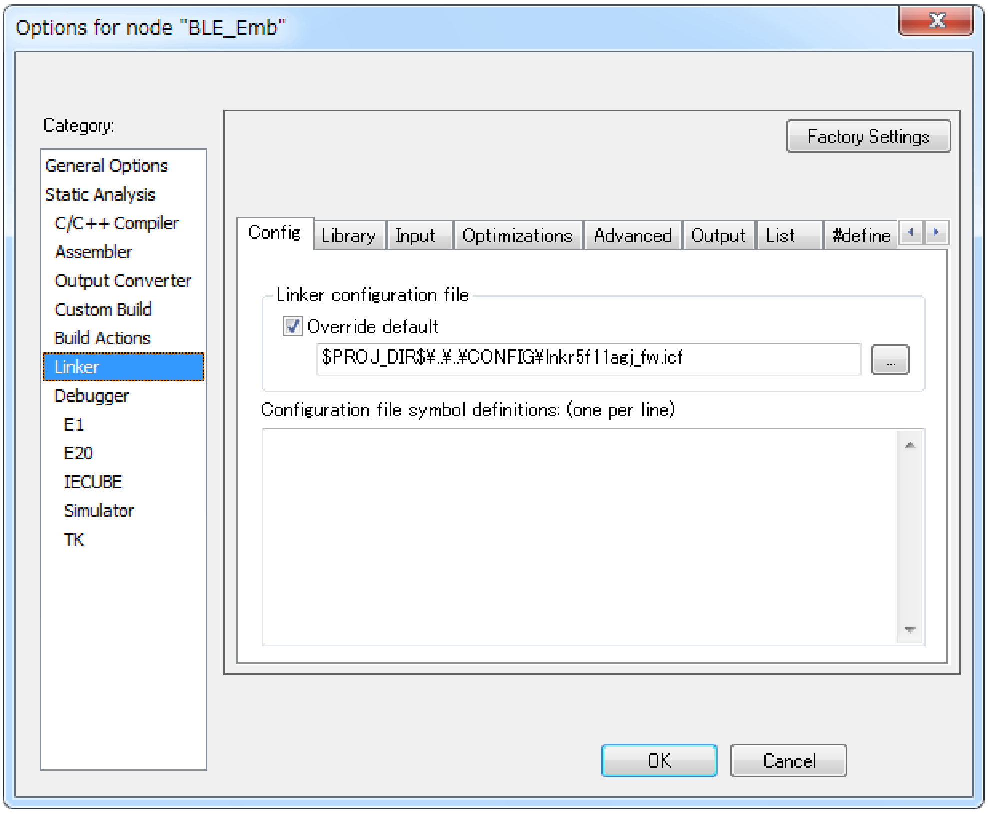


Figure 7-26 Setting of Linker configuration file(Embedded Configuration)

(4) If need setting of force link, Select [Input] and set function which need force link.

About force link is shown 7.10.3.

If change profile by using FW Update function, Must set following function at 'Keep symbols'.

Embedded :
```
?F_DIV
?F_MUL
?F_SL2F
?F_UL2F
?SL_RSH_L03
?UL_RSH_L03
?0EI_VSWITCH_L10
?0SI_VSWITCH_L10
?1EC_VSWITCH_L10
?1SI_VSWITCH_L10
?2SI_VSWITCH_L10
?3SI_VSWITCH_L10
?I_VSWITCH_L10
___iar_copy_init2
___iar_packbits_init_near_single2
```

Modem :
```
?I_VSWITCH_L10
?3SI_VSWITCH_L10
?2SI_VSWITCH_L10
?0SI_VSWITCH_L10
___iar_copy_init2
___iar_packbits_init_near_single2
```

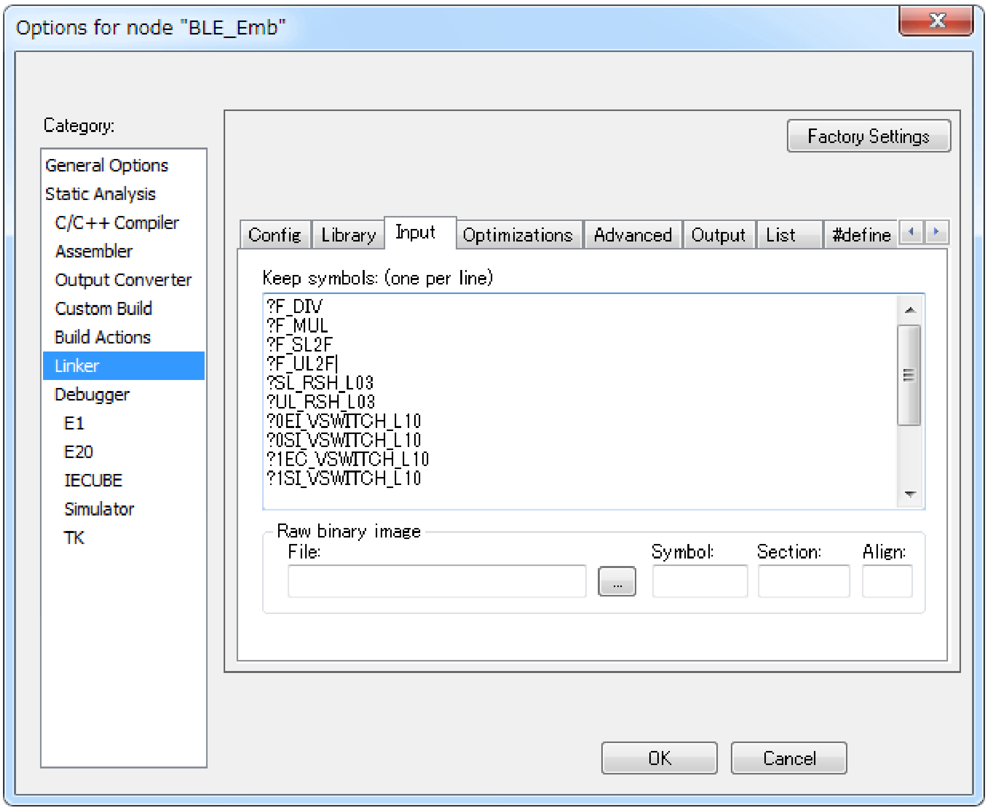


Figure 7-27 Setting of Keep symbols.

(5) Select [General Options]→[Library Options]

- Embedded Configuration
  Change the Printf formatter from 'Large' to 'Small'.

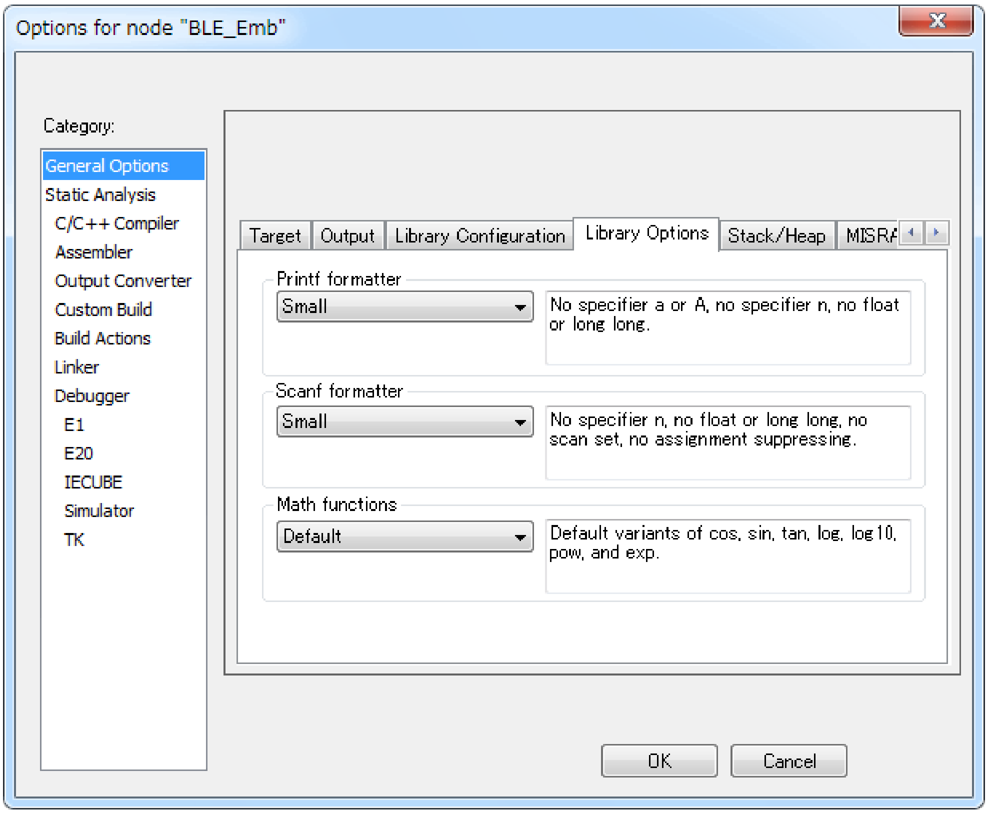


Figure 7-28 Setting of Library Options (Embedded Configuration)

(6) Click ‘OK’ and finish option setting.

(7) Run Build.

7.10.1.2. Project Settings of e^2 studio

The setup procedures of the project in e^2 studio are shown in the following.

(1) Launch the e^2 studio, and open the workspace.

(2) From the [Project Explorer], right-click the project of rBLE_Emb or rBLE_Mdm, select the [Renesas Tool Settings] in the context menu.

(3) From the left tree of [Tool Settings] tab, select [Compiler]→[Source], and change the following definitions from the right pane of the [Macro definition].

```
noUSE_FW_UPDATE_PROFILE → USE_FW_UPDATE_PROFILE
```

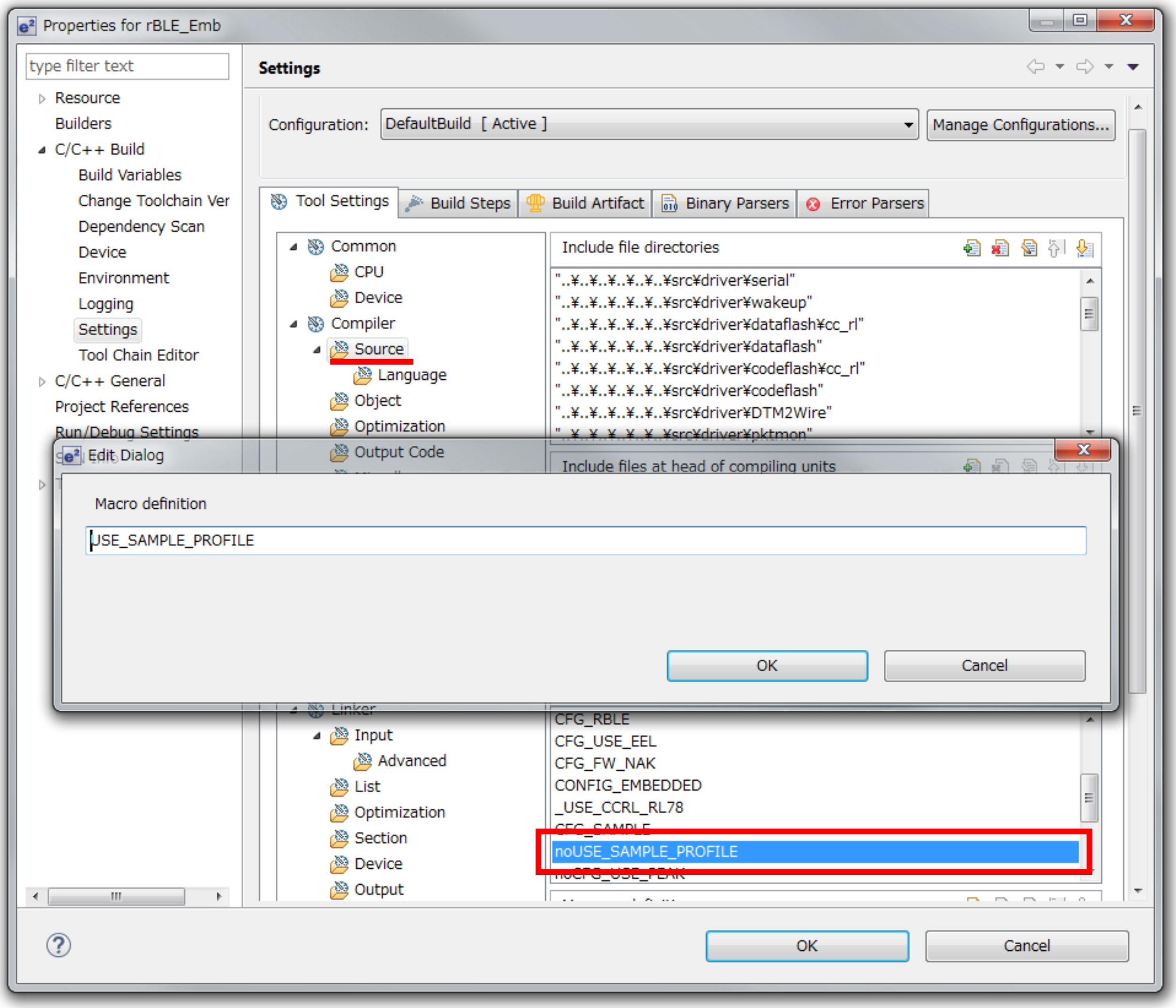


Figure 7-29 Setting of Macro definition.

(4) From the left tree of [Tool Settings] tab, select [Linker]→[Section], and click the [Import] button of the right pane, then select the following section information file for FW update.

Embedded : renesas\tools\project\e2studio\BLE_Embedded \rBLE_Emb\sect_emb_fwup.esi

Modem : renesas\tools\project\e2studio\BLE_Modem \rBLE_Mdm\sect_mdm_fwup.esi

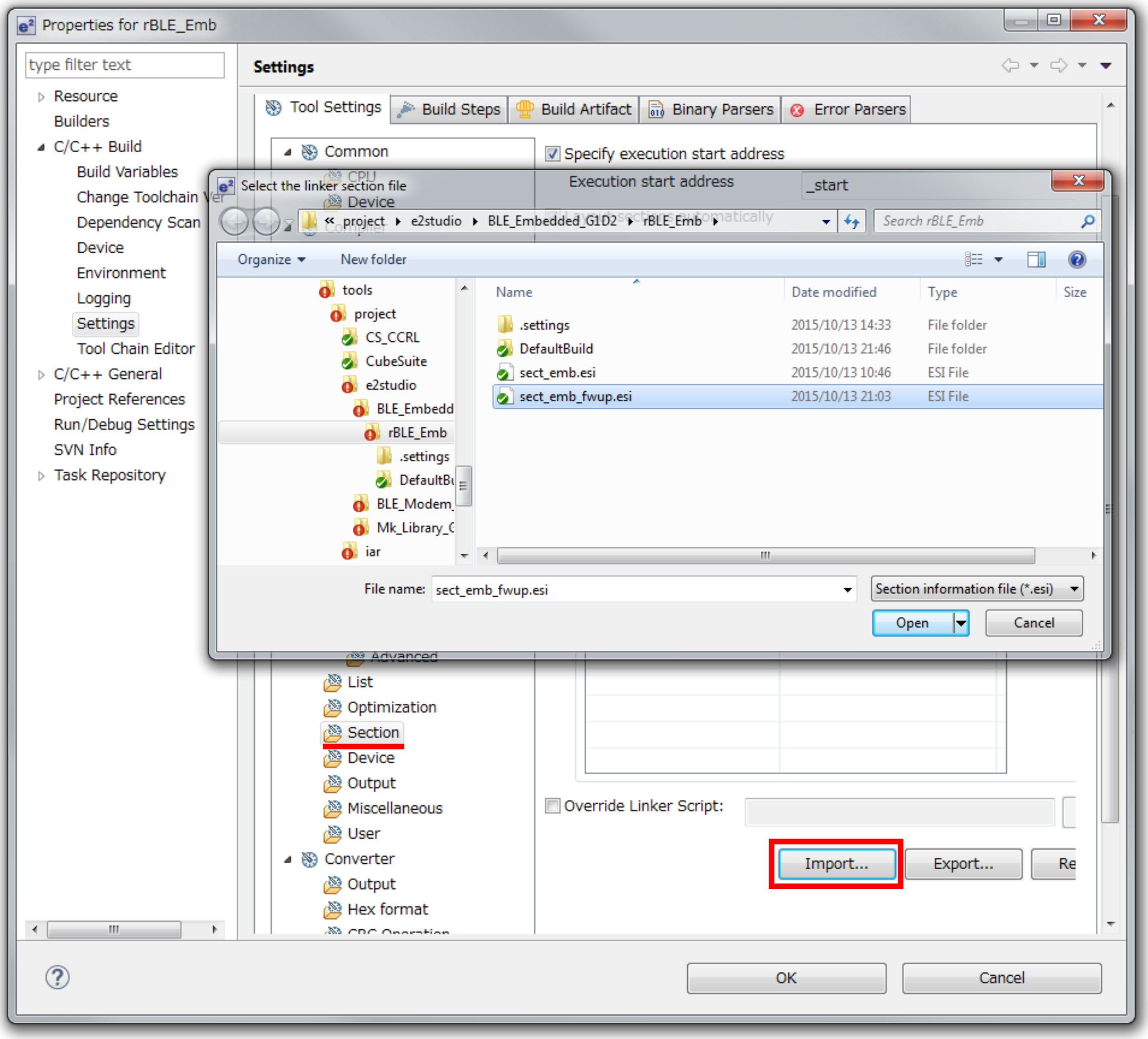


Figure 7-30 Import of section information file

(5) Click 'OK' and finish tool settings.

(6) Run Build.

7.10.2. Sender device

The setup procedures of the project for Sender device are shown in the following.

The Sender device received FW Update data from the Sample Program at Windows.

So change operating frequency to 32MHz and change UART baud rate to 76,800bps.

[Note] It is possible to do FW Update at low clock, but FW Update time will be long.

(1) Change UART baud rate

To change the UART driver in order to 76,800bps the UART baud rate.

The serial_init() function in 'Renesas\RL78_G1D\Project_Source\renesas\src\driver\uart\uart.c' is changed following processing.

[Note] red word is a changing point.

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

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

(2) Change operating frequency

To change operating frequency is shown in Bluetooth Low Energy Protocol Stack User's Manual.

7.10.3. Notes of making FW Update Environment

・force link of function

In FW Update, update code area of Application and profile.

So it is impossible to change link of runtime library or standard library before and after the FW Update.

[Note] If link of runtime library or standard library is changed, can't use runtime library or standard library from excluded area of FW Update.

If link of runtime library or standard library is changed, you need to link to the required function of runtime library or standard library using forced link in the FW of before FW Update.

Way of forcing link is shown in 7.10.1.

7.11. References

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

7.12. Terminology

Term	Description
Service	A service is provided from a GATT server to a GATT client. The GATT server exposes some characteristics as the interface. The service prescribes how to access the exposed characteristics.
Profile	A profile enables implementation of a use case by using one or more services. The services used are defined in the specifications of each profile.
Characteristic	A characteristic is a value used to identify services. The characteristics to be exposed and their formats are defined by each service.
Role	Each device takes the role prescribed by the profile or service in order to implement the specified use case.
Client Characteristic Configuration Descriptor	This is used to control the transmission (notification / indication) of the characteristic values from the GATT server with a client characteristic configuration descriptor.
Connection Handle	This is the handle determined by the controller stack and is used to identify connection with a remote device. The valid handle range is between 0x0000 and 0x0EFF.
Universally Unique Identifier (UUID)	This is an identifier for uniquely identifying an item. In the BLE standard, a 16-bit UUID is defined for identifying services and their characteristics.
Bluetooth Device Address (BD Address)	This is a 48-bit address for identifying a Bluetooth device. The BLE standard defines both public and random addresses, and at least one or the other must be supported
Public Address	This is an address that includes an allocated 24-bit OUI (Organizationally Unique Identifier) registered with the IEEE.
Random Address	This is an address that contains a random number and belongs to one of the following three categories: • Static Address • Non-Resolvable Private Address Resolvable Private Address
Static Address	This is an address whose 2 most significant bits are both 1, and whose remaining 46 bits form a random number other than all 1's or all 0's. This static address cannot be changed until the power is switched off.
Non-resolvable private Address	This is an address whose 2 most significant bits are both 0, and whose remaining 46 bits form a random number other than all 1's or all 0's. Static addresses and public addresses must not be equal. This type of address is used to make tracking by an attacker difficult by changing the address frequently.
Resolvable private Address	This is an address generated from an IRK and a 24-bit random number. Its 2 most significant bits are 0 and 1, and the remaining higher 22 bits form a random number other than all 1's or all 0's. The lower 24 bits are calculated based on an IRK and the higher random number. This type of address is used to make tracking by an attacker difficult by changing the address frequently. By allocating an IRK to the peer device, the peer device can identify the communicating device by using that IRK.
Broadcaster	This is one of the roles of GAP. It is used to transmit advertising data.
Observer	This is one of the roles of GAP. It is used to receive advertising data.
Central	This is one of the roles of GAP. It is used to establish a physical link. In the link layer, it is called Master.
Peripheral	This is one of the roles of GAP. It is used to accept the establishment of a physical link. In the link layer, it is called Slave.
Advertising	Advertising is used to transmit data on a specific channel for the purpose of establishing a connection or performing data transmission.
Scan	Scans are used to receive advertising data. There are two types of scans: Passive scan, in which data is simply received, and active scan, in which additional information is requested by sending SCAN_REQ.
White List	By registering known devices that are connected or bonded to a White List, it is possible to filter devices that can accept advertising data or connection requests.

Device Name	This is a user-friendly name freely assigned to a Bluetooth device to identify it. In the BLE standard, the device name is exposed to the peer device by the GATT server as a GAP characteristic.
Reconnection Address	If a non-resolvable private address is used and the address is changed frequently, not only attackers but also the peer device will have difficulty identifying the device. Therefore, the address to be used at reconnection is reported by setting a new reconnection address as the exposed reconnection address characteristic.
Scan Interval	This is the interval for receiving advertising data.
Scan Window	This is the period of time during which advertising data is received at the scan interval.
Connection Interval	This is the interval for transmitting and receiving data periodically following connection establishment.
Connection Event	This is the period of time during which data is transmitted and received at the connection interval.
Slave Latency	This is the period of time during which data is transmitted and received at the connection interval.
Supervision Timeout	This is the timeout interval after which the link is considered to have been lost when no response is received from the peer device.
Passkey Entry	This is a pairing method whereby a six-digit number is input by each device to the other, or a six-digit number is displayed by one of the devices and that number is input to the other device.
Just Works	This is a pairing method that does not require user action.
OOB	This is a pairing method whereby pairing is performed by using data obtained by a communication method other than Bluetooth.
Identity Resolving Key (IRK)	This is a 128-bit key used to generate and resolve resolvable private addresses.
Connection Signature Resolving Key (CSRK)	This is a 128-bit key used to create data signatures and verify the signature of incoming data.
Long Term Key (LTK)	This is a 128-bit key used for encryption. The key size to be used is the size agreed on during pairing.
Short Term Key (STK)	This is a 128-bit key used for encryption during key exchange. It is generated using TK.
Temporary Key (TK)	This is a 128-bit key used required for STK generation. In the case of Just Works, the TK value is 0. In the case of Passkey Entry, it is the 6-digit number that was input, and in the case of OOB, it is the OOB data.

Website and support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision Record

Rev.	Issued on	Description	
		Page	Summary
1.01	Feb 15, 2013	-	First edition issued
1.10	Mar 27, 2013	-	The descriptions on the following topics are added: * Compliant with the BLE S/W Ver.2.0 * Serial communication * Custom profile * 2-wire DTM
1.11	Apr 12, 2013	-	Replace the captured images from the command prompt screen
1.12	Jun 28, 2013	2	Update folder organization
1.13	Nov 29, 2013	- 2 10 - 37 41 45 49 52 101 122	Compliant with the BLE S/W Ver.2.3 Update folder organization Added note to Usage of Sample Program The usages of the following profiles are added * Heart Rate Profile * Cycling Speed and Cadence Profile * Cycling Power Profile * Alert Notification Profile * Location and Navigation Profile Printf program in the Embedded configuration is added References specifications are added
1.14	Sep 19, 2014	- - 4 4 5 8 102 113 -	Compliant with the Bluetooth specification v4.1 Compliant with the BLE S/W Ver0.5 Update version of VC++ Delete folder organization Update Operating Environment and Development Environment Update folder pass of EXE file. FW Update Sample Program is added FW Update Environment is added Clerical error correction
1.15	Jan 30, 2015	- 120	Compliant with the BLE S/W Ver0.9 Changed the UART baud rate for FW Update.
1.16	Apr 17, 2015	- -	Compliant with the BLE S/W Ver1.0 Add IIC interface of serial communication
1.17	Jul 10, 2015	- 97	Compliant with the BLE S/W Ver1.01 Change the mode switching specification in the Modem Configuration
1.18	Oct 30, 2015	- 5, 118 97	Compliant with the BLE S/W Ver1.1 The description related CS+ for CC / e^2 studio (CC-RL) are added. The description for the baud rate information of DTM mode is added.
1.19	Aug 31, 2016	- -	Compliant with the BLE S/W Ver1.2 Description is changed from CD to package. Version information for Renesas Flash Programmer is

		5	changed.
		8	“5.1 How to Change Parameters” is moved from Appendix.
		9	“UART 2-wire Branch Connection” is added in Table 5-1
		67	Heading level of “7.2 Requirements and Flow Chart of Serial Communication Driver on APP MCU” is changed.
		79	Reference value of ROM/RAM size is added.
		86	File structure for Sample Custom Profile is updated.
		103	File structure for F/W update is updated.
		113	The description for F/W update of IAR V2 is added.
1.20	Jul 31, 2017	-	Update supported Windows / Visual Studio version. Remove IAR V1 description.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   ¾ The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   ¾ The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   ¾ The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   ¾ When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   ¾ The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

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

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

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

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

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

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

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

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.1