Bluetooth® Low Energy Protocol Stack

Virtual UART Application

Introduction

This manual describes the software configuration, functions, operation check procedure, and implementation details of the Virtual UART Application that uses Bluetooth LE wireless technology.

The Virtual UART Application runs with Renesas Bluetooth® Low Energy Protocol Stack on a Renesas RL78/G1D device as embedded configuration and provides the following functions.

- Simple AT command function to control and configure Bluetooth LE connection
- Virtual UART function to send / receive characters or binary data to / from a remote device over Bluetooth LE communication
- A function to select communication with response or without response in Bluetooth LE communication

Target Device

RL78/G1D

Related Documents

Document Name	Document No.
Bluetooth® Low Energy Protocol Stack	
User's Manual	R01UW0095E
API Reference Manual: Basics	R01UW0088E
Quick Start Guide	R01AN2767E
GUI Tool	R01AN2469E
Bluetooth® Specification	
Vol 6. Low Energy Controller volume	Core_v4.2

Contents

1. Overview

This manual describes the software configuration, functions, operation check procedure, and implementation details of the Virtual UART Application (hereinafter referred to as "application") that uses Bluetooth LE wireless technology.

The application runs with Renesas Bluetooth® Low Energy Protocol Stack (hereinafter referred to as "BLE Protocol Stack") on a Renesas RL78/G1D device as embedded configuration and provides the following functions.

- Simple AT command function to control and configure Bluetooth LE connection
- Virtual UART function to send / receive characters or binary data to / from a remote device over Bluetooth LE communication
    - Character data transmission and reception operation
    - Binary data transmission and reception operation
- A function to select communication with response or without response in Bluetooth LE communication
    - With response: Write Request, Indication
    - Without response: Write Command, Notification

1.1 Application Behavior

Figure 1-1 shows the application execution environment setup.

Prepare two RL78/G1D evaluation boards and write the application onto them. Then connect them respectively to PC via USB cable. A user operates the application through a terminal software.

Simple AT command can be used to establish, disconnect and configure Bluetooth LE connection. After establishing a Bluetooth LE connection, data typed on the local terminal are sent to a remote device and are displayed on the remote terminal. In the contrary, data typed on the remote terminal are sent to the local device and are displayed on the local terminal.

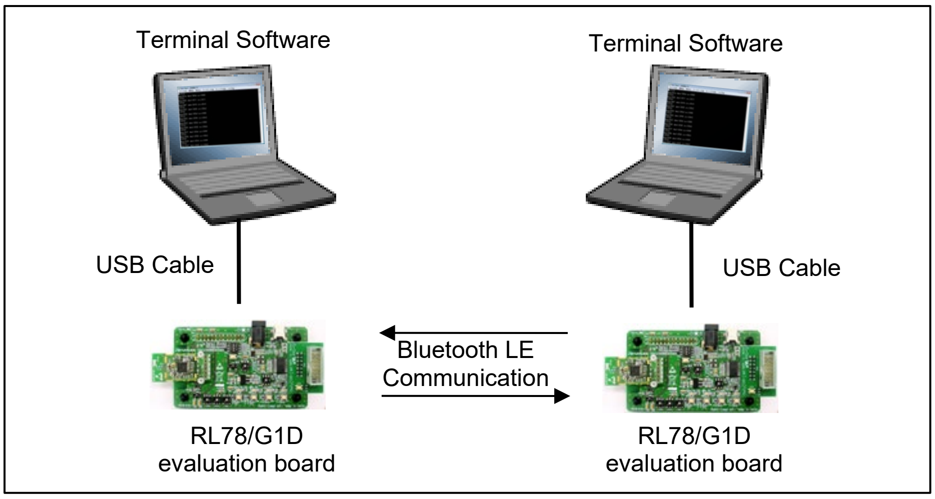

Terminal Software
Terminal Software
USB Cable
USB Cable
Bluetooth LE Communication
RL78/G1D evaluation board
RL78/G1D evaluation board

Figure 1-1: Application execution environment

2. Architecture

2.1 Software Architecture

Figure 2-1 shows software architecture of this application.

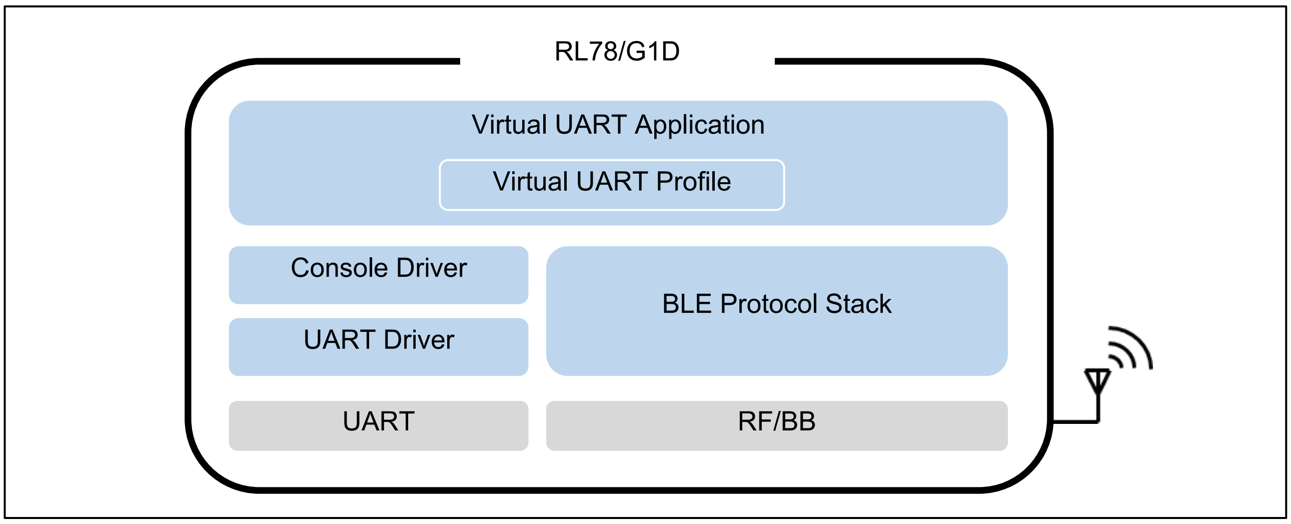


Figure 2-1: Software architecture

The software components are listed below.

Table 2-1: Software components

Component	Description
Virtual UART Application	The application is used for execution of simple AT command and transmission of characters or binary data. Virtual UART profile is defined to transfer characters.
Console Driver	The driver is used to relay data between a terminal software and Virtual UART application by using UART Driver functionality.
UART Driver	The device driver to control UART IP of RL78/G1D.
BLE Protocol Stack	Renesas Bluetooth® Low Energy Protocol Stack. Refer to "Bluetooth® Low Energy Protocol Stack User's manual" (R1UW0095E).

2.2 File Composition

The application is implemented based on BLE software which include BLE Protocol Stack. In this section, only files that have modified or added to the BLE software are listed. The modified files are marked with (M) and the added files are marked with (A).

```
r01an3130xx0114-rl78g1d-ble-vuart/
  ├── Project_Source/
  │   ├── rBLE/
  │   │   └── src/
  │   │       ├── sample_app/
  │   │       │   ├── r_vuart_app.c                   (A)   ┐ Virtual UART Application
  │   │       │   ├── r_vuart_app.h                   (A)   │
  │   │       │   ├── r_vuart_app_param.c             (A)   ┘
  │   │       │   ├── r_vuart_console.c               (A)   ┐ Console Driver
  │   │       │   └── r_vuart_console.h               (A)   ┘
  │   │       └── sample_profile/
  │   │           └── vuart/
  │   │               ├── vuart.h                     (A)   ┐ Virtual UART Profile
  │   │               ├── vuartc.c                    (A)   │
  │   │               ├── vuartc.h                    (A)   │
  │   │               ├── vuarts.c                    (A)   │
  │   │               └── vuarts.h                    (A)   ┘
  │   └── renesas/
  │       ├── src/
  │       │   ├── arch/
  │       │   │   └── rl78/
  │       │   │       ├── arch_main.c                 (M)   Modified to enable low power mode
  │       │   │       ├── db_handle.h                 (M)   ┐ Modified for Virtual UART Profiles
  │       │   │       ├── ke_conf.c                   (M)   │
  │       │   │       ├── prf_config.c                (M)   │
  │       │   │       └── prf_config.h                (M)   ┘
  │       │   └── driver/
  │       │       ├── dataflash/
  │       │       │   ├── eel_descriptor_t02.c        (M)   ┐ Modified to add definitions to
  │       │       │   └── eel_descriptor_t02.h        (M)   ┘ access the Data Flash
  │       │       └── uart/
  │       │           └── uart.c                      (M)   Modified to enable low power mode
  │       └── tools/
  │           └── project/
  │               ├── CS_CCRL/                        (M)   ┐
  │               ├── CS_CACX/                        (M)   │ Project files for
  │               └── e2studio/                       (M)   ┘ development environments
  ├── ROM_File/
  │   ├── ccrl/                                             ┐
  │   │     ├──RL78_G1D_CCE(VUART_CHAR).hex                 │ ccrl generated firmware
  │   │     └──RL78_G1D_CCE(VUART_BIN).hex            (A)   ┘
  │   └── ca78k0r/                                          ┐
  │         ├──RL78_G1D_CE(VUART_CHAR).hex                  │ ca78k0r generated firmware
  │         └──RL78_G1D_CE(VUART_BIN).hex             (A)   ┘
```

```
 │
 └── Macro/                                                      ┐
      ├── teraterm_debug_mode_on.ttl                             │
      ├── teraterm_debug_mode_off.ttl                            │Tera Term macro
      ├── tt_send_bin_1.ttl                                      │
      └── tt_send_bin_2.ttl                              (A)     ┘
```

3. Application Mode

3.1 mode

The application has two modes as shown in Table 3-1.

Table 3-1: Application mode

Application Mode	Description
Simple AT Command Mode	This mode is used for execution of the simple AT command to control and configure a Bluetooth LE connection. In this mode, characters are never sent to the remote device. Refer to "4. Simple AT Command Mode" for the detail.
Virtual UART Mode	This mode is used for transmission of character data or binary data over Bluetooth LE communication. In this mode, data typed during the application in disconnect state are not sent to the remote device. Refer to "5. Virtual UART Mode" for the detail.

3.2 Selection of data communication with response or without response

When data communication in virtual UART mode, you can select the communication method with or without response with the DIP switch on the RL78/G1D evaluation board.

Table 3-2: Setting of with response or without response

Number	Setting	Description
SW6-4	OFF (left side)	Data communication with response. from a client to a server : Write Request from a server to a client : Indication
	ON (right side)	Data communication without response. from a client to a server : Write Command from a server to a client : Notification

3.3 Selection of data to communicate

In the application, you can select "3.3.1 Character data transmission and reception" or "3.3.2 Binary data transmission and reception" in the macro settings. For macro settings, refer to "8.8.1 Character data/Binary data transmission and reception setting".

3.3.1 Character data transmission and reception

In the character data transmission and reception, ASCII printable characters and new-line character can be transmitted.

The application mode shown in Table 3-1 can be switched by entering the escape key (ASCII code: 0x1B) in the terminal software.

For an example of using character data transmission and reception, refer to "7. Build and Operational Check"-"7.4.1 Character data transmission and reception".

3.3.2 Binary data transmission and reception

In the binary data transmission and reception, all data including character data can be transmitted.

The application mode shown in Table 3-1 can be switched by the DIP switch on the RL78/G1D evaluation board.

For an example of using binary data transmission and reception, refer to "7. Build and Operational Check"-"7.4.2 Binary data transmission and reception".

Table 3-3: Operation settings for sending and receiving binary data

Number	Setting	Description
SW6-1	OFF (left side)	Simple AT Command Mode.
	ON (right side)	Virtual UART Mode. Note: In the virtual UART mode for sending and receiving binary data, local echo of the data input to the terminal software is prohibited.

4. Simple AT Command Mode

In Simple AT Command Mode, a user can control and configure Bluetooth LE connection by simple AT command.

Table 4-1 shows the simple AT commands that the application supports.

Table 4-1: List of simple AT commands

Simple AT command	Description
AT-C	Create a connection to the address specified by AT-AP=<addr>.
AT-C=<addr>	Create a connection to the address specified by <addr>.
AT-R	When the device is in connect state, disconnect the connection and start advertising. When the device is in disconnect state, start advertising.
AT-AS=<addr>	Set <addr> as the public device address of the local device.
AT-AS?	Display the public device address of the local device.
AT-AP=<addr>	Set the address used by AT-C.
AT-AP?	Display the address used by AT-C.
AT-DS	Display the address of devices which support virtual UART profile.
AT-S?	Display the application state. Connect state or Disconnect state.
AT-CI=<con_intv>	Change Connection Interval.
AT-CI?	Display Connection interval setting value.
ATE0	Disable local echo.
ATE1	Enable local echo.

4.1 Details of Simple AT Command

4.1.1 AT-C

Description	Create a connection to the address specified by AT-AP=<addr>	
Response	OK	Success
	ERROR	Failed due to the application is connect state
	CONNECT	Connection established
Command Example	AT-C OK CONNECT	

4.1.2 AT-C=<addr>

Description	Create a connection to the address specified by <addr>	
Response	OK	Success
	ERROR	Failed due to the application is connect state
	CONNECT	Connection established
Command Example	AT-C=CBA987654321 (Set CB:A9:87:65:43:21) OK CONNECT	

4.1.3 AT-R

Description	When the application is in connect state, disconnect the connection and start advertising. When the application is in disconnect state, start advertising.	
Response	OK	Success
	DISCONNECT	Disconnected
Command Example	[Connect state] AT-R OK DISCONNECT [Disconnect state] AT-R OK	

4.1.4 AT-AS=<addr>

Description	Set <addr> as the local device public device address. The address set by this command is preserved over power cycles. The address set by this command is reflected after reset. Please reset the system for example by pushing the reset button on the board.	
Response	OK	Success
	ERROR	Failed due to the application is connect state
Command Example	AT-AS=CCCCBBBBAAAA (Set CC:CC:BB:BB:AA:AA) OK	

4.1.5 AT-AS?

Description	Display the local device public address.	
Response	OK	Success
Command Example	AT-AS? -AS: CCCCBBBBAAAA (The address is CC:CC:BB:BB:AA:AA) OK	

4.1.6 AT-AP=<addr>

Description	Set <addr> as the public device address used by AT-C. The address set by this command is preserved over power cycles.	
Response	OK	Success
	ERROR	Failed due to the application is connect state
Command Example	AT-AP=CCCCBBBBAAAA (Set CC:CC:BB:BB:AA:AA) OK	

4.1.7 AT-AP?

Description	Display the public device address used by AT-C.	
Response	OK	Success
Command Example	AT-AP? -AP: CCCCBBBBAAAA (The address is CC:CC:BB:BB:AA:AA) OK	

4.1.8 AT-DS

Description	Display the address of the device which support virtual UART profile. Whether a device is supporting virtual UART profile is confirmed by checking the advertising data includes virtual UART service UUID.	
Response	OK	Success
	ERROR	Failed due to the application is connect state
Command Example	AT-DS -DS: CBA987654321 (The address is CB:A9:87:65:43:21) -DS: CCCCBBBBAAAA (The address is CC:CC:BB:BB:AA:AA) OK	

4.1.9 AT-S?

Description	Display the local device address connect state. The state is CONNECT or DISCONNECT.	
Response	OK	Success
Command Example	[Connect state] AT-S? CONNECT OK [Disconnect state] AT-S? DISCONNECT OK	

4.1.10 AT-CI=<con_intv>

Description	Change Connection Interval. Execute this command in Disconnect state, the application retains the con_intv value internally, and the value will be used for following connection. This command cannot change Connection Interval in Connect state. Connection Interval is calculated by following calculation. Connection Interval = con_intv * 1.25[ms] Ex) When you want to set 20[ms], execute AT-CI=16 The default value of Connection Interval is 30[ms]. Refer Figure 8-3 for Connection Interval Change sequence. After establishing the connection, Peripheral device requests Connection Interval parameter update to the Central device. Central device can decline the request depending on the restriction that Central device have. You can check whether requested Connection Interval is accepted by executing "AT-CI?" command.	
Response	OK	Success
	ERROR	Failed due to the application is connect state
		The setting value is out of range (Range: 6~3200)
Command Example	[Disconnect state] AT-CI=20 OK AT-CI=3201 [Connect state] AT-CI=20 ERROR	

4.1.11 AT-CI?

Description	When execute this command in Connect state, display Connection Interval of the connection. When execute this command in Disconnect state, display the retained Connection Interval. To compute the actual Connection Interval, multiply the response value by 1.25[ms]. Ex) If the response is "-CI: 20", Connection Interval is 20 * 1.25[ms] = 25[ms].	
Response	OK	Success
Command Example	AT-CI? -CI: 20 OK	

4.1.12 ATE0

Description	Disable local echo.	
Response	OK	Success
Command Example	ATE0 OK	

4.1.13 ATE1

Description	Enable local echo.	
Response	OK	Success
Command Example	ATE1 OK	

5. Virtual UART Mode

After establishing Bluetooth LE connection between two devices, a user can exchange data with the remote device.

In the character data transmission and reception, ASCII printable characters and new-line character can be transmitted. In the binary data transmission and reception, all data including character data can be transmitted.

5.1 Virtual UART Profile

Data transfer is enabled by GATT based virtual UART profile. Refer to 8.1.

The connection initiating device, which is executes AT-C command, works as GATT client and the remote device works as GATT server. Below is data transfer details.

5.1.1 Data communication with response

- To send data from the client to the server, send “Write Request” to the server. The server receives the data, and replies “Response” to the client.
- To send data from the server to the client, send “Indication” to the client. The client receives the data, and replies “Confirmation” to the server.

The data reception by the remote device can be confirmed by waiting for the "Response" or "Confirmation" reply from the remote device.

Figure 5-1 shows the data transfer sequence.

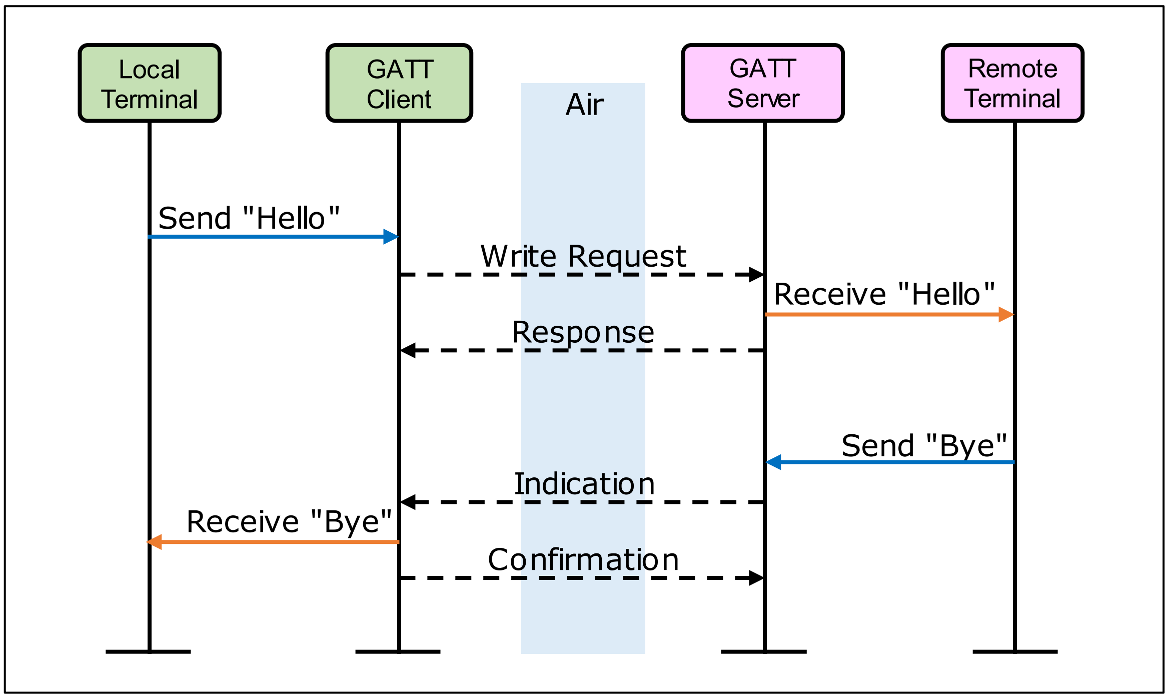


Figure 5-1: Data transfer sequence (with response)

5.1.2 Data communication without response

- To send data from the client to the server, send "Write Command" to the server.
- To send data from the server to the client, send "Notification" to the client.

Figure 5-2 shows the data transfer sequence.

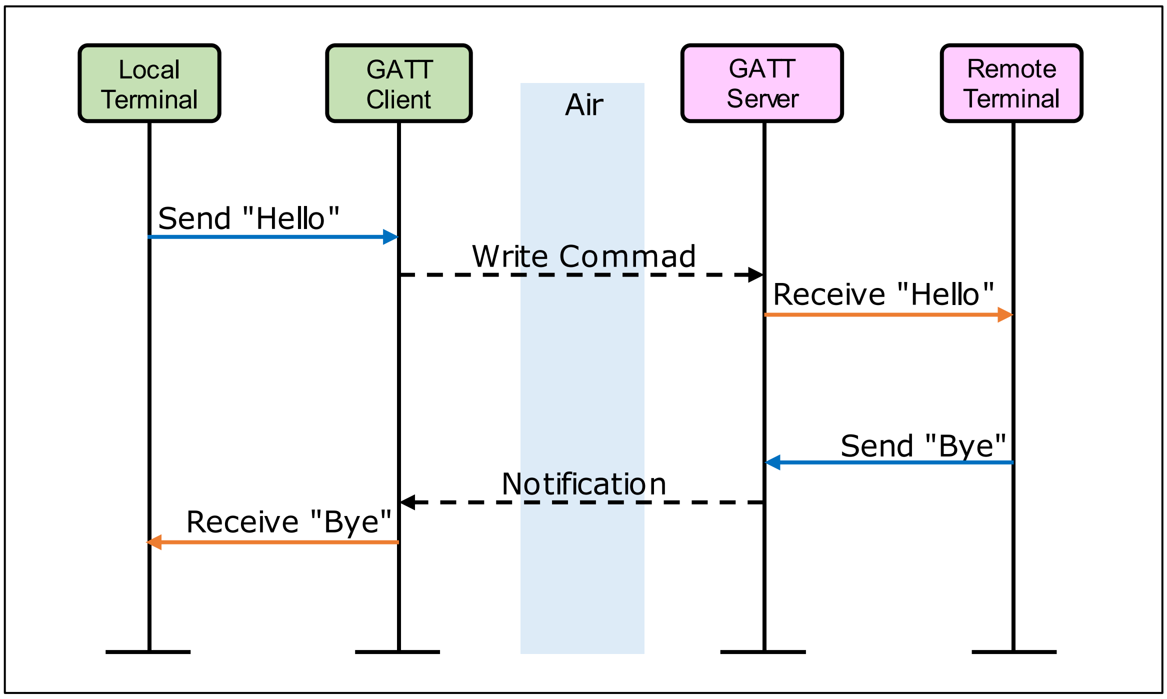


Figure 5-2: Data transfer sequence (without response)

5.2 Buffering of the Send Characters

In order to avoid the loss of send data, the application have a send data buffer.

5.2.1 Data communication with response

To send data within the period between "Write Request" and "Response" or "Indication" and "Confirmation" is not possible. So data typed within this period can be lost.

Virtual UART profile has a buffer and stores data typed within this period. If there are data in the buffer when receiving "Response" or "Confirmation" from the remote device, the profile sends it soon.

Figure 5-3 shows a send data buffering sequence.

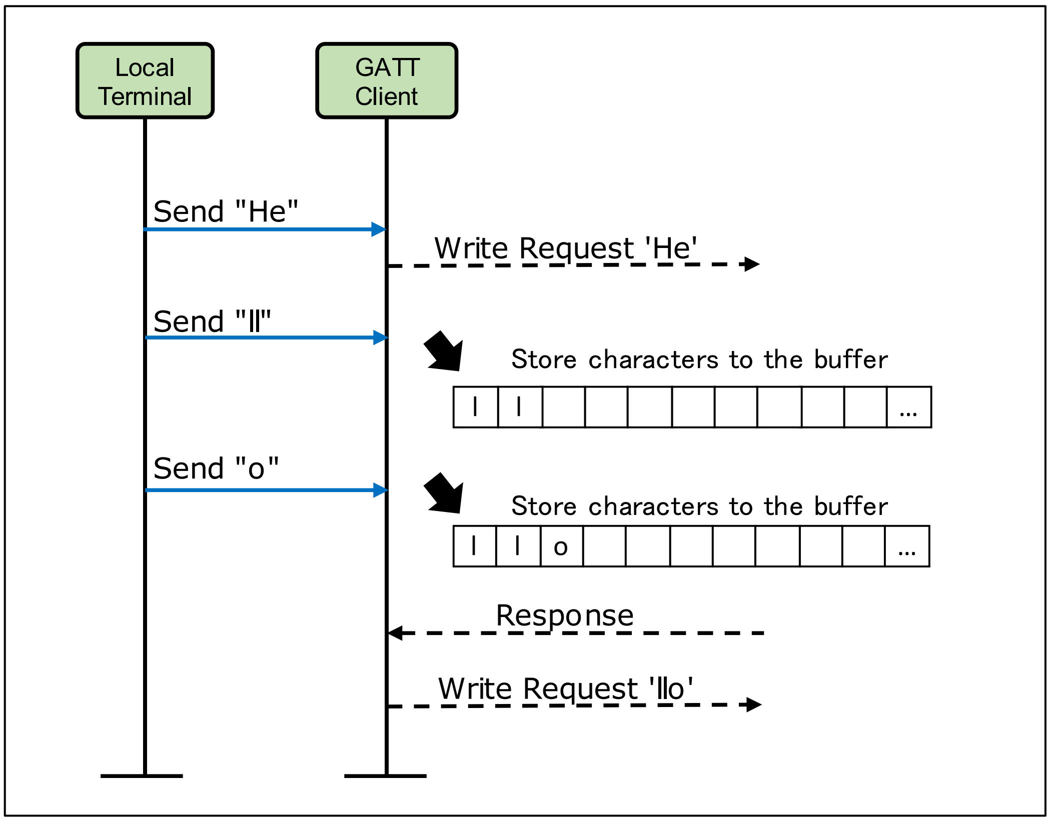


Figure 5-3: Send data buffering sequence (with response)

5.2.2 Data communication without response

After sending data with Write Command or Notification, use the timer function (ke_timer_set) of RWKE to wait for a certain period of time and buffer the data from the terminal software to some extent. Then send it after the timer has expired. This prevents the communication efficiency from dropping because it is sent in small units (1 byte or 2 bytes) if it is sent every time data is received from the terminal software. It also prevents the data entered by the user from being lost.

Figure 5-4 shows a send data buffering sequence.

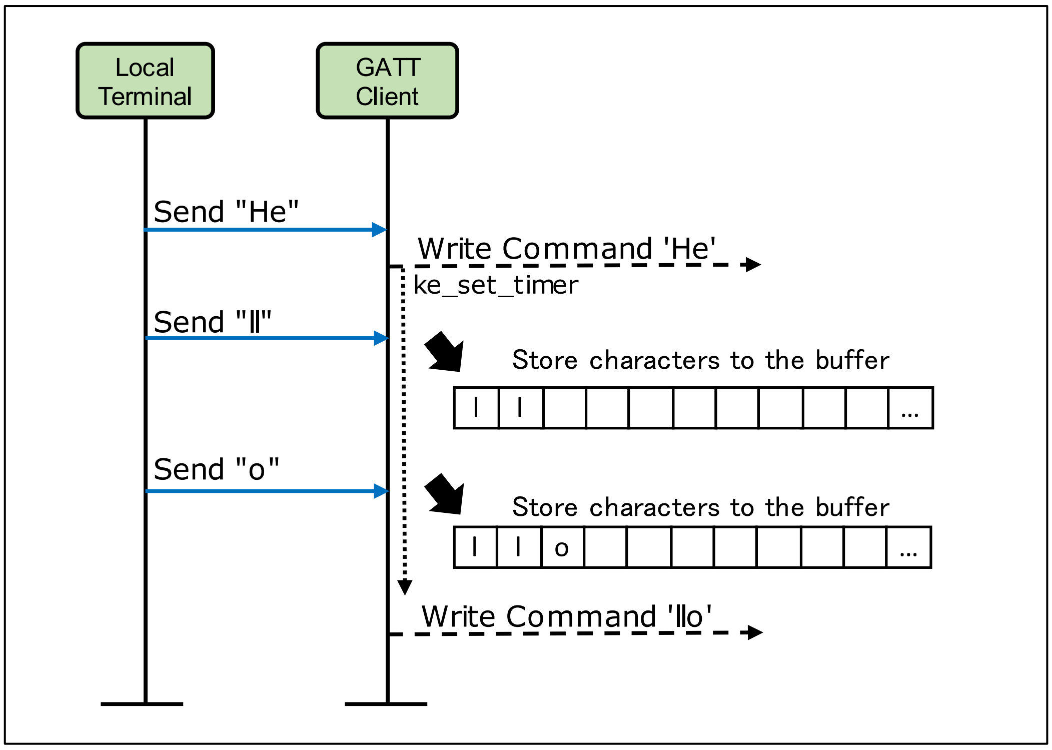


Figure 5-4: Send data buffering sequence (without response)

5.3 Encryption of BLE Connection

To protect the Bluetooth LE connection from such as eavesdropping, the encryption of the Bluetooth LE connection is enabled.

The application does not hold the pairing information, it performs pairing on each connection.

6. Power Saving Function

6.1 CPU STOP Mode

When 3 seconds have elapsed after the last data transfer, CPU enters STOP mode in order to reduce the power consumption. During STOP mode, the blinking of the LED1/LED2 becomes slower or stops. If data transfer is occurred when CPU is in STOP mode, CPU returns from STOP mode soon.

6.2 Changes of Advertising Interval

When 30 seconds has elapsed after advertising started, the application sets a longer advertising interval in order to reduce the power consumption. The longer advertising interval is reset by re-enabling advertising with connection and disconnection or AT-R.

The default advertising interval is 30 milliseconds and the longer advertising interval is 3 seconds.

If you do not want to change the advertising interval, comment out the code below.

r_vuart_app.c, line 60, line1009

```
60:     #define APP_ADV_LOW_POWER_DURATION (3000)

1009:   ke_timer_set(RBLE_APP_ADV_EVT, TASK_CON_APPL, APP_ADV_LOW_POWER_DURATION);
```

7. Build and Operational Check

7.1 Environment

Below is the environment to use for application build and operation check.

- Hardware
  - Host PC
    - PC/AT™ compatible machine
    - Processor Speed : 1.6GHz or higher
    - Main Memory : 1GB or more
    - Interface : USB2.0 (Used for E1 and RL78/G1D evaluation board)
  - Device
    - RL78/G1D evaluation board [RTK0EN0001010001BZ]

- Tool
  - Supports the following on-chip debuggers.
    ･E1 emulator
    ･E2 emulator
    ･E2 emulator Lite

  Note: The E1 emulator is used in this application note. We have already discontinued production of the E1 emulator due to components of the product having reached their EOL (end of life, i.e. end of production). Please click on the following link to confirm the details and our successor products.
  >> Tool News: [Notification] Advance Notice of E1 Emulator Product End of Life (EOL)

- Software
  - Windows® 7 or later
  - Supports the following integrated environments.
    ･e^2 studio 2020-07 (64-bit version) / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
    ･e^2 studio V7.8.0 (32-bit version) / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
    ･Renesas CS+ for CC V8.04.00 / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
    ･Renesas CS+ for CA, CX V4.04.00 / Renesas CA78K0R V1.72
  - Renesas Flash Programmer v3.06.02
  - Tera Term Version 4.105
  - UART-USB conversion device driver

  Note: It may be that device driver of UART-USB conversion IC “FT232RL” is requested is in the first connection with host. In this case, you can get the device driver from below link.

    FTDI (Future Technology Device International) – Drivers
    https://ftdichip.com/drivers/d2xx-drivers/

7.2 Build Procedure

The application can be built using the following environment.

Since all the files required for build are included in this application note, there is no need to download the BLE protocol stack or EEPROM emulation library.

- e^2 studio 2020-07 (64-bit version) / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
- e^2 studio V7.8.0 (32-bit version) / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
- Renesas CS+ for CC V8.04.00 / RL78 Family C Compiler Package V1 (without IDE) V1.09.00
- Renesas CS+ for CA, CX V4.04.00 / Renesas CA78K0R V1.72

Note: Please refer to "8.8 Macro Settings" and set the application before building.

7.2.1 e^2 studio (32-bit version/64-bit version)

1. Launch e^2 studio.
2. Right click on “Project Explorer” and select “Import…” from the dropdown menu.
3. “Import” window is popped up and select “Existing Projects into Workspace” and click “Next >”.
4. Fill “Select root directory:” form with the project folder shown in Table 7-1 and make sure that the project you selected is displayed in “Projects:” and click “Finish”. Then the window is closed.
5. Right click on the project just imported on “Project Explorer” and Select “Build Project” from the dropdown menu.
6. Refer Table 7-1 for the Hex file firmware path.

Table 7-1: Project file and Hex file Location (e^2 studio)

e^2 studio with CC-RL	
Project Folder	\Project_Source\renesas\tools\project\e2studio\BLE_Embedded\rBLE_Emb
Firmware	\Project_Source\renesas\tools\project\e2studio\BLE_Embedded\rBLE_Emb\DefaultBuild\rBLE_Emb_CCRL.hex

7.2.2 CS+

1. Double click the project file shown in Table 7-2.
2. Right click on “BLE_Emb” in “Project Tree” and select “Build BLE_Emb” from the dropdown menu.
3. Refer Table 7-2 for the Hex file firmware path.

Table 7-2: Project file and Hex file Location (CS+)

CS+ for CC	
Project File	\Project_Source\renesas\tools\project\CS_CCRL\BLE_Embedded\BLE_Embedded.mtpj
Firmware	\Project_Source\renesas\tools\project\CS_CCRL\BLE_Embedded\rBLE_Emb\DefaultBuild\rBLE_Emb_CCRL.hex
CS+ for CA, CX	
Project File	\Project_Source\renesas\tools\project\CS_CACX\BLE_Embedded\BLE_Embedded.mtpj
Firmware	\Project_Source\renesas\tools\project\CS_CACX\BLE_Embedded\BLE_Emb\DefaultBuild\BLE_Emb.hex

7.2.3 BLE Protocol Stack / EEPROM emulation library

The BLE protocol stack and EEPROM emulation library used in this application can be download from the Renesas Web page.

- BLE protocol stack
    - https://www.renesas.com/software-tool/bluetooth-low-energy-protocol-stack-rl78-family
- EEPROM emulation library
    - EEPROM Emulation Library Pack02 Package Ver.2.00(for CA78K0R/CC-RL Compiler) for RL78 Family
        - https://www.renesas.com/software-tool/data-flash-libraries

NOTE: The link address can be changed without notice.

7.3 Preparation for Execution Environment

1. Write the firmware onto two RL78/G1D evaluation boards. Refer to "Bluetooth® Low Energy Protocol Stack Quick Guide" (R01AN2767E) Section 5.

   You can use the pre-built HEX file included in this application note. This application note uses the CC-RL HEX file.

Table 7-3: Pre-built HEX file

Function	File
Character data transmission and reception	\ROM_File\ccrl\RL78_G1D_CCE(VUART_CHAR).hex
Binary data transmission and reception	\ROM_File\ccrl\RL78_G1D_CCE(VUART_BIN).hex

2. As shown in Figure 1-1, connect both RL78/G1D evaluation boards to PCs respectively.
3. Launch a terminal software on both PCs and configure them as Table 7-4. This application note uses Tera Term as the terminal software.

Table 7-4: Terminal software configuration

Setting	Value
New-line Receive	LF
New-line Send	CR
Baud rate	4800 [bps]
Data length	8 [bit]
Parity bit	none
Stop bit	1 [bit]
Flow control	none

4. Set the DIP switches on the two RL78/G1D evaluation boards according to Table 7-5.

Table 7-5: DIP switch setting

DIP switch	Setting
SW6-1	OFF (left side: Simple at command mode) Note: Used in "7.4.2 Binary data transmission and reception"
SW6-4	OFF (left side: Data communication with response)

7.4 Usage Example

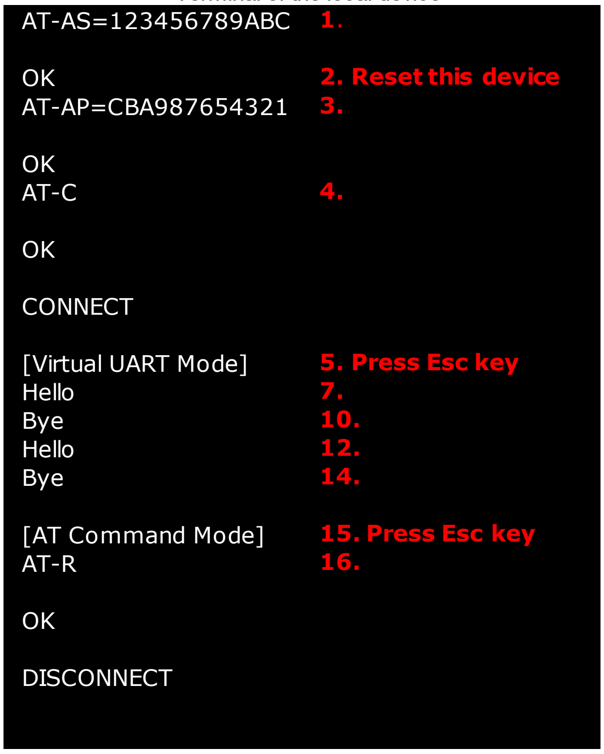
7.4.1 Character data transmission and reception

In this example, set device addresses, establish a Bluetooth LE connection, transfer characters and disconnect the connection.

Figure 7-1 shows execution results of the terminal. Figure 7-2 and Figure 7-3 shows the sequence diagram of this example usage. The red numbers in figures are corresponding to the numbers in the following procedures.

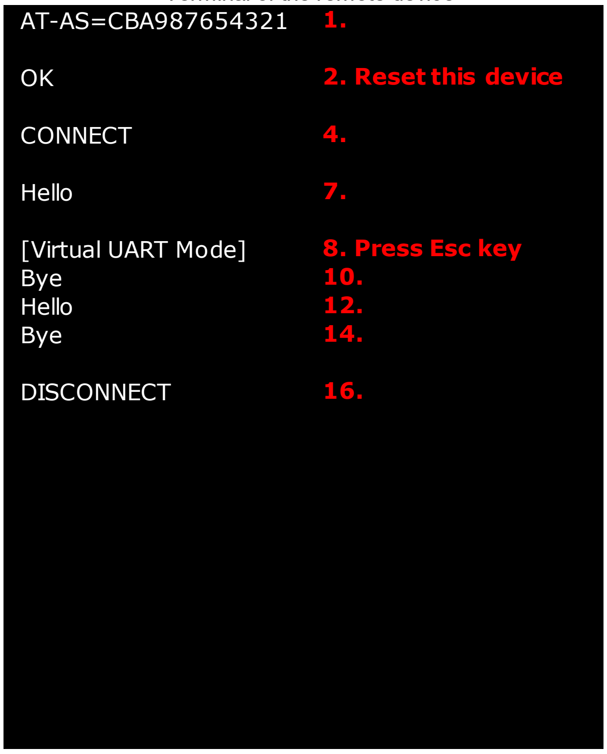
1. Set local and remote device address. To set the device address use “AT-AS=<addr>” command. For example, to set 12:34:56:78:9A:BC, execute “AT-AS=123456789ABC”. To display current device address settings, use “AT-AS?”.
   If a device address is 00:00:00:00:00:00 or the local device and the remote device have the same addresses, you need to change the device address. In the following example, we assume that you set 12:34:56:78:9A:BC to the local device address, CB:A9:87:65:43:21 to the remote device address.
2. When you change a device address by “AT-AS=<addr>” command, you need to reset the device to reflect the change by pushing RESET button (SW5) on the board.
3. Execute “AT-AP=CBA987654321” to set the target device address for a connection.
4. Execute “AT-C” on the local terminal. This command start the connection to the device which have the address CB:A9:87:65:43:21. After the connection established, “CONNECT” response is displayed on both of the local and remote terminal.
5. Type ESC key on the local terminal to switch the application mode to Virtual UART mode.
6. Set the DIP switch (SW6-4) of the local device to OFF (left side: communication with response).
7. Type “Hello” to the local terminal. Then “Hello” is displayed on the remote terminal.
8. Type ESC key on the remote terminal to switch the application mode to Virtual UART mode.
9. Set the DIP switch (SW6-4) of the remote device to OFF (left side: communication with response).
10. Type “Bye” on the remote terminal. Then “Bye” is displayed on the local terminal.
11. Set the DIP switch (SW6-4) of the local device to ON (right side: communication without response).
12. Type “Hello” to the local terminal. Then “Hello” is displayed on the remote terminal.
13. Set the DIP switch (SW6-4) of the remote device to ON (right side: communication without response).
14. Type “Bye” on the remote terminal. Then “Bye” is displayed on the local terminal.
15. Type ESC key on the local terminal to switch the application mode to AT command mode.
16. Execute “AT-R” on the local terminal to disconnect the connection. After completing the disconnection, “DISCONNECT” response is displayed on both of the local and remote terminal.

Terminal of the local device



```
AT-AS=123456789ABC   1.

OK                   2. Reset this device
AT-AP=CBA987654321   3.

OK
AT-C                 4.

OK

CONNECT

[Virtual UART Mode]  5. Press Esc key
Hello                7.
Bye                  10.
Hello                12.
Bye                  14.

[AT Command Mode]    15. Press Esc key
AT-R                 16.

OK

DISCONNECT
```

Terminal of the remote device



```
AT-AS=CBA987654321   1.

OK                   2. Reset this device

CONNECT              4.

Hello                7.

[Virtual UART Mode]  8. Press Esc key
Bye                  10.
Hello                12.
Bye                  14.

DISCONNECT           16.
```

Figure 7-1: Terminal result

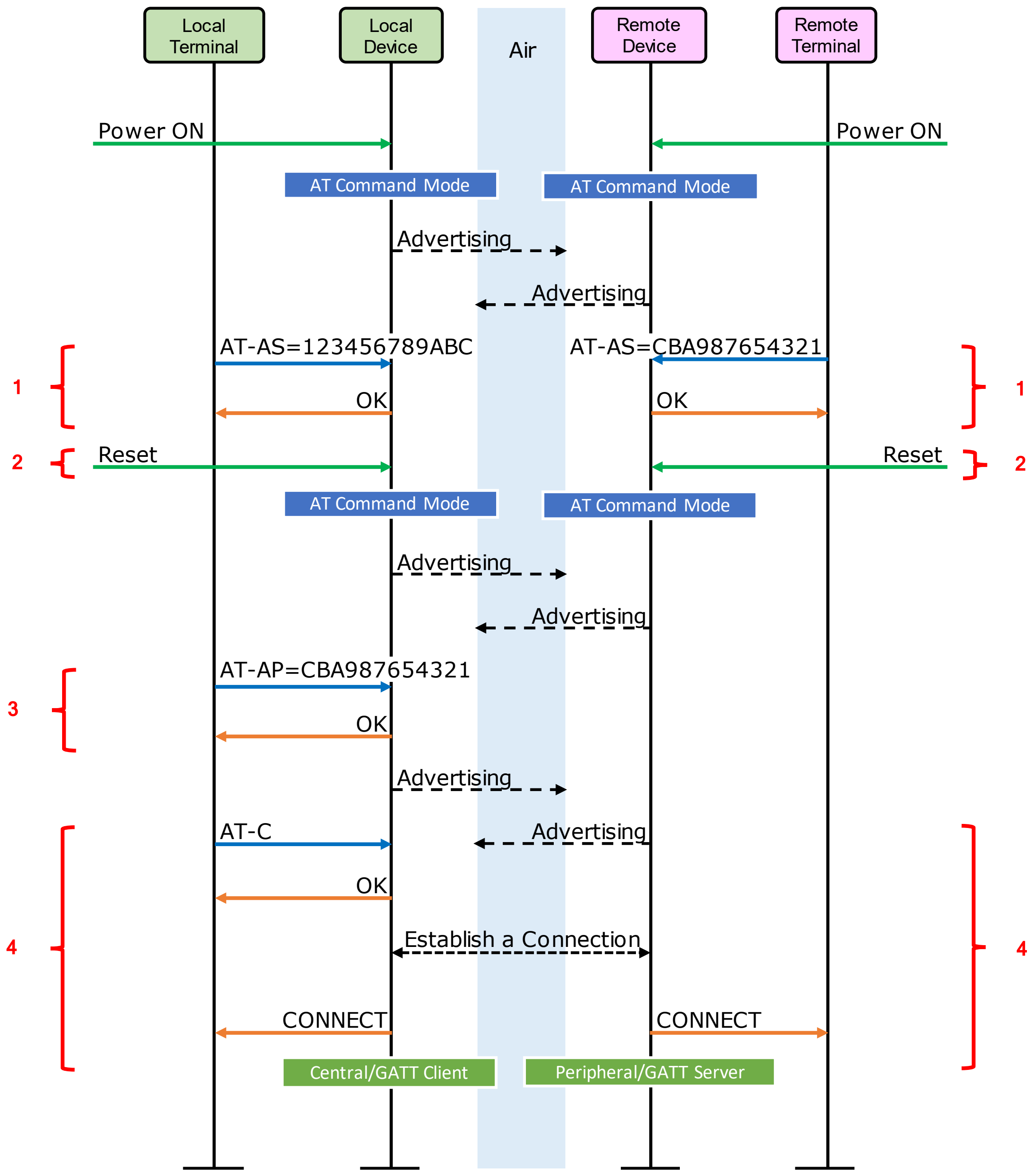


Figure 7-2: Example usage sequence of character data transmission and reception (1/2)

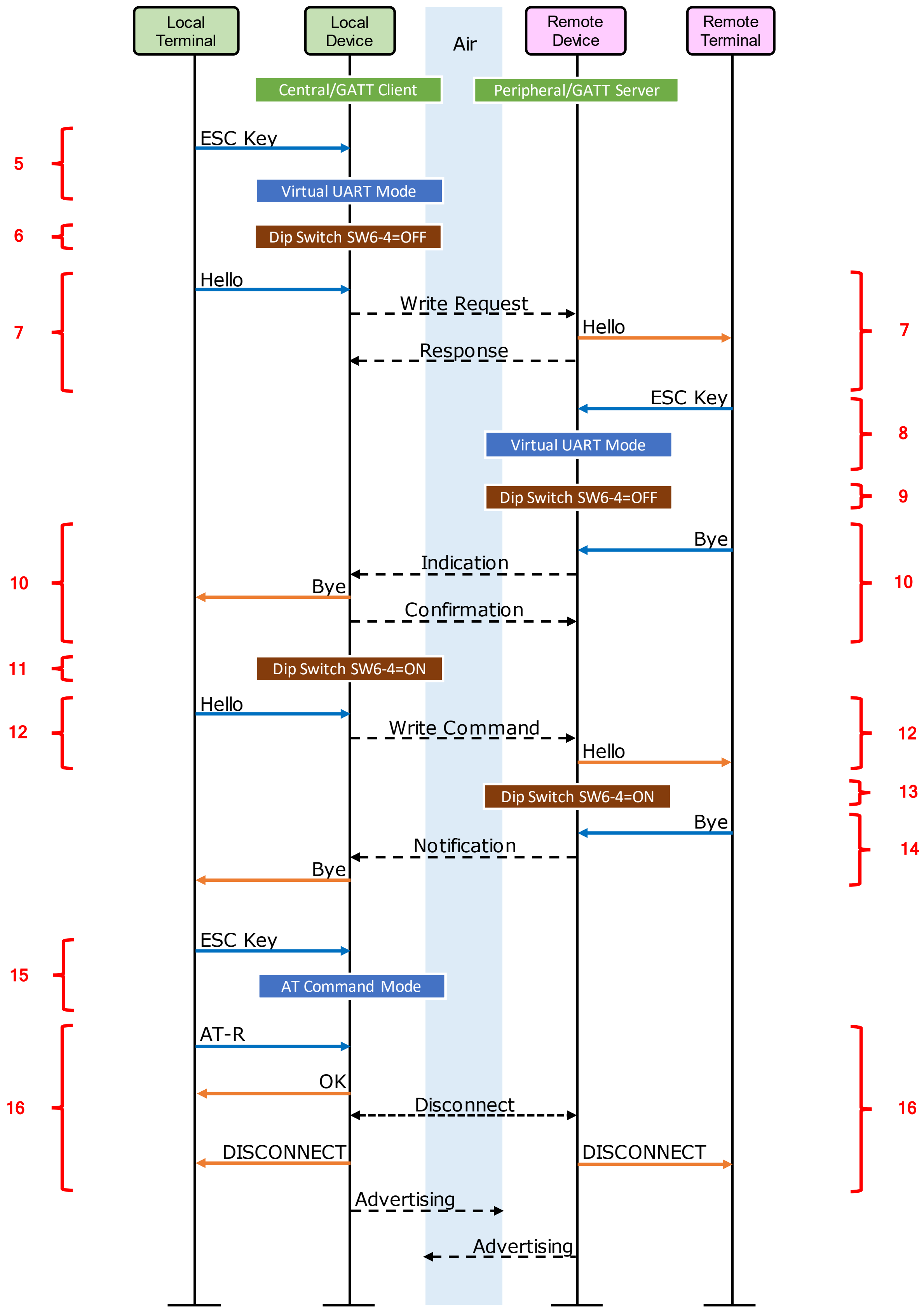


Figure 7-3: Example usage sequence of character data transmission and reception (2/2)

7.4.2 Binary data transmission and reception

In this example, set device addresses, establish a Bluetooth LE connection, transfer binary data and disconnect the connection.

Also, use the macro function of the terminal when binary data transmission and reception. The macro files to be used is shown below.

Table 7-6: Macro file (\r01an3130xx0114-rl78g1d-ble-vuart\Macro\)

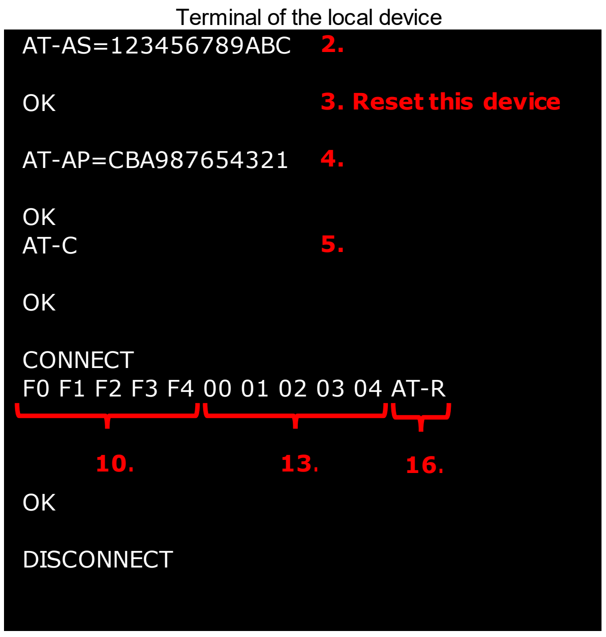
File name	Description
tt_debug_mode_on.ttl	Turn on the debug mode of Tera Term. The terminal display will be in hexadecimal.
tt_debug_mode_off.ttl	Turn off the debug mode of Tera Term. The terminal display will be in the initial state.
tt_send_bin_1.ttl	Sends binary data (0x00 0x01 0x02 0x03 0x04).
tt_send_bin_2.ttl	Sends binary data (0xF0 0xF1 0xF2 0xF3 0xF4).

Figure 7-4 shows execution results of the terminal. Figure 7-5 and Figure 7-6 shows the sequence diagram of this example usage. The red numbers in figures are corresponding to the numbers in the following procedures.

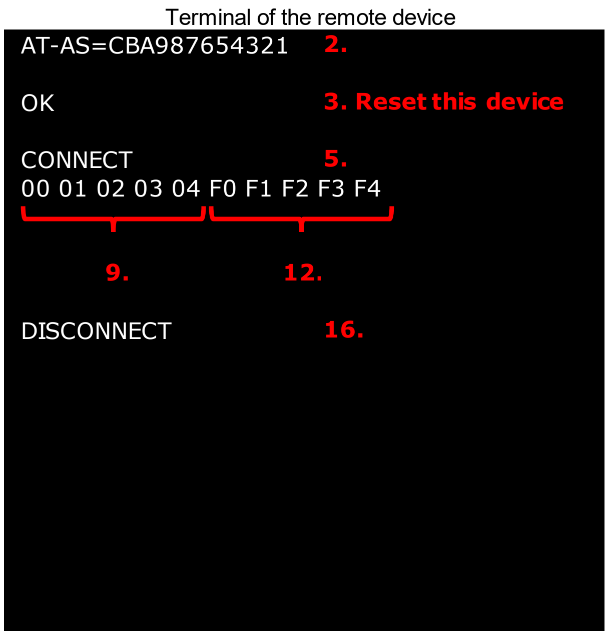
1. Set the DIP switch (SW6-1) of local and remote device to OFF (left side: Simple AT command mode).
2. Set local and remote device address. To set the device address use “AT-AS=<addr>” command. For example, to set 12:34:56:78:9A:BC, execute “AT-AS=123456789ABC”. To display current device address settings, use “AT-AS?”.
   If a device address is 00:00:00:00:00:00 or the local device and the remote device have the same addresses, you need to change the device address. In the following example, we assume that you set 12:34:56:78:9A:BC to the local device address, CB:A9:87:65:43:21 to the remote device address.
3. When you change a device address by “AT-AS=<addr>” command, you need to reset the device to reflect the change by pushing RESET button (SW5) on the board.
4. Execute “AT-AP=CBA987654321” to set the target device address for a connection.
5. Execute “AT-C” on the local terminal. This command start the connection to the device which have the address CB:A9:87:65:43:21. After the connection established, “CONNECT” response is displayed on both of the local and remote terminal.
6. Set the DIP switch (SW6-1) of local and remote device to ON (right side: Virtual UART mode).
7. Set the DIP switch (SW6-4) of the local device to OFF (left side: communication with response).
8. Run the macro on the local and remote device terminals to put the terminal in debug mode.
   Select "Control" from the terminal menu and select "Macro" from the dropdown menu. "MACRO" window is popped up and select the macro file "tt_debug_mode_on.ttl" in Table 7-6. click "Open".
9. Send binary data from the local device.
   Select "Control" from the terminal menu and select "Macro" from the dropdown menu. "MACRO" window is popped up and select the macro file "tt_send_bin_1.ttl" in Table 7-6. click "Open".
   Then "00 01 02 03 04" is displayed on the remote terminal.
10. Send binary data from the remote device.
    Select "Control" from the terminal menu and select "Macro" from the dropdown menu. "MACRO" window is popped up and select the macro file "tt_send_bin_2.ttl" in Table 7-6. click "Open".
    Then "F0 F1 F2 F3 F4" is displayed on the remote terminal.
11. Set the DIP switch (SW6-4) of the local and remote device to ON (right side: communication without response).
12. Send binary data from the local device.
    Select "Control" from the terminal menu and select "Macro" from the dropdown menu. "MACRO"

window is popped up and select the macro file "tt_send_bin_2.ttl" in Table 7-6. click "Open".
Then "F0 F1 F2 F3 F4" is displayed on the remote terminal.

13. Send binary data from the remote device.
    Select "Control" from the terminal menu and select "Macro" from the dropdown menu. "MACRO" window is popped up and select the macro file "tt_send_bin_1.ttl" in Table 7-6. click "Open".
    Then "00 01 02 03 04" is displayed on the local terminal.
14. Run the macro on the local and remote device terminals to put the terminal in initial mode.
    Select "Control" from the terminal menu and select "Macro" from the dropdown menu. "MACRO" window is popped up and select the macro file "tt_debug_mode_off.ttl" in Table 7-6. click "Open".
15. Set the DIP switch (SW6-1) of local and remote device to OFF (left side: Simple AT command mode).
16. Execute “AT-R” on the local terminal to disconnect the connection. After completing the disconnection, “DISCONNECT” response is displayed on both of the local and remote terminal.



```
AT-AS=123456789ABC

OK

AT-AP=CBA987654321

OK
AT-C

OK

CONNECT
F0 F1 F2 F3 F4 00 01 02 03 04 AT-R

OK

DISCONNECT
```



```
AT-AS=CBA987654321

OK

CONNECT
00 01 02 03 04 F0 F1 F2 F3 F4

DISCONNECT
```

Figure 7-4: Terminal result

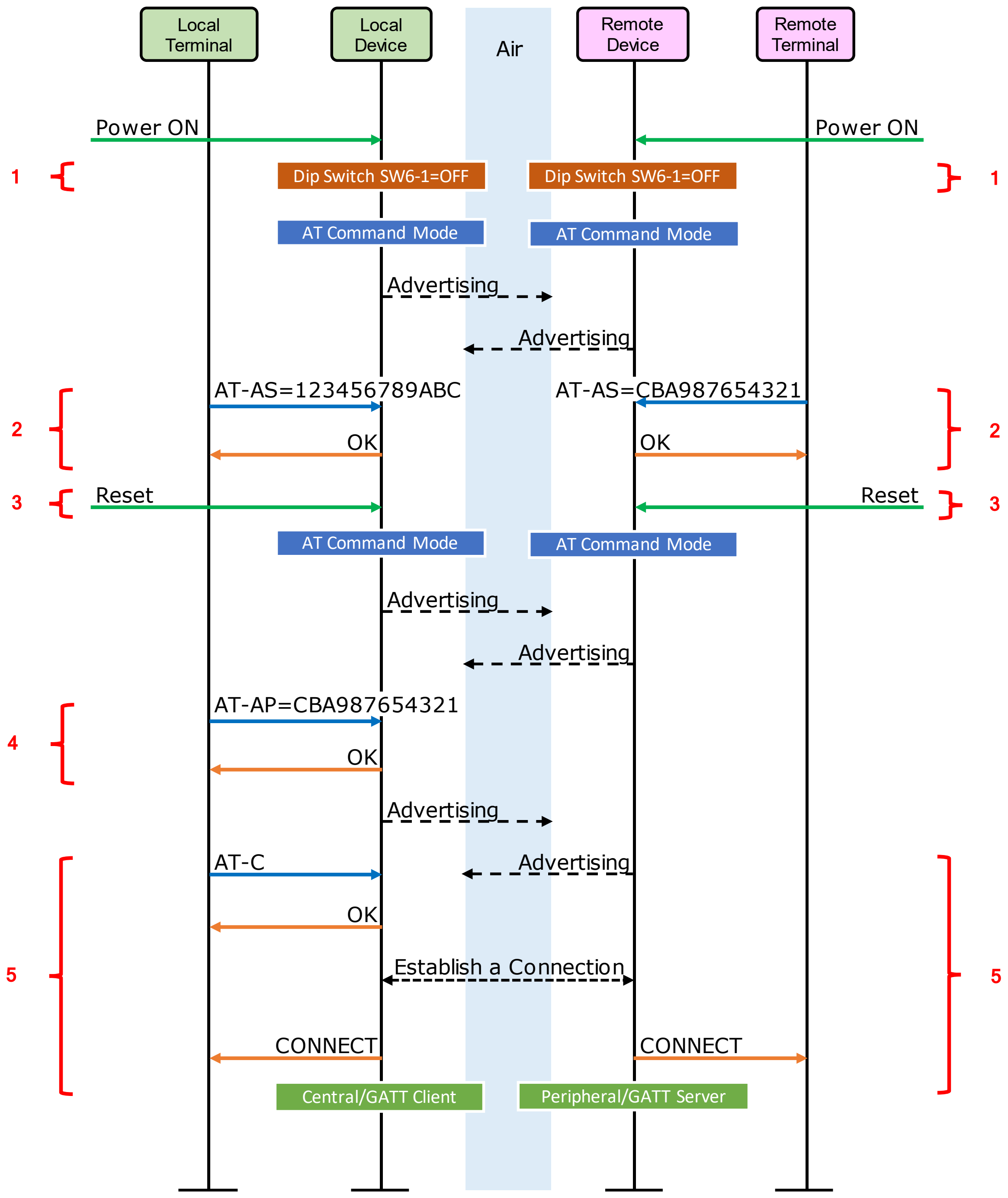


Figure 7-5: Example usage sequence of binary data transmission and reception (1/2)

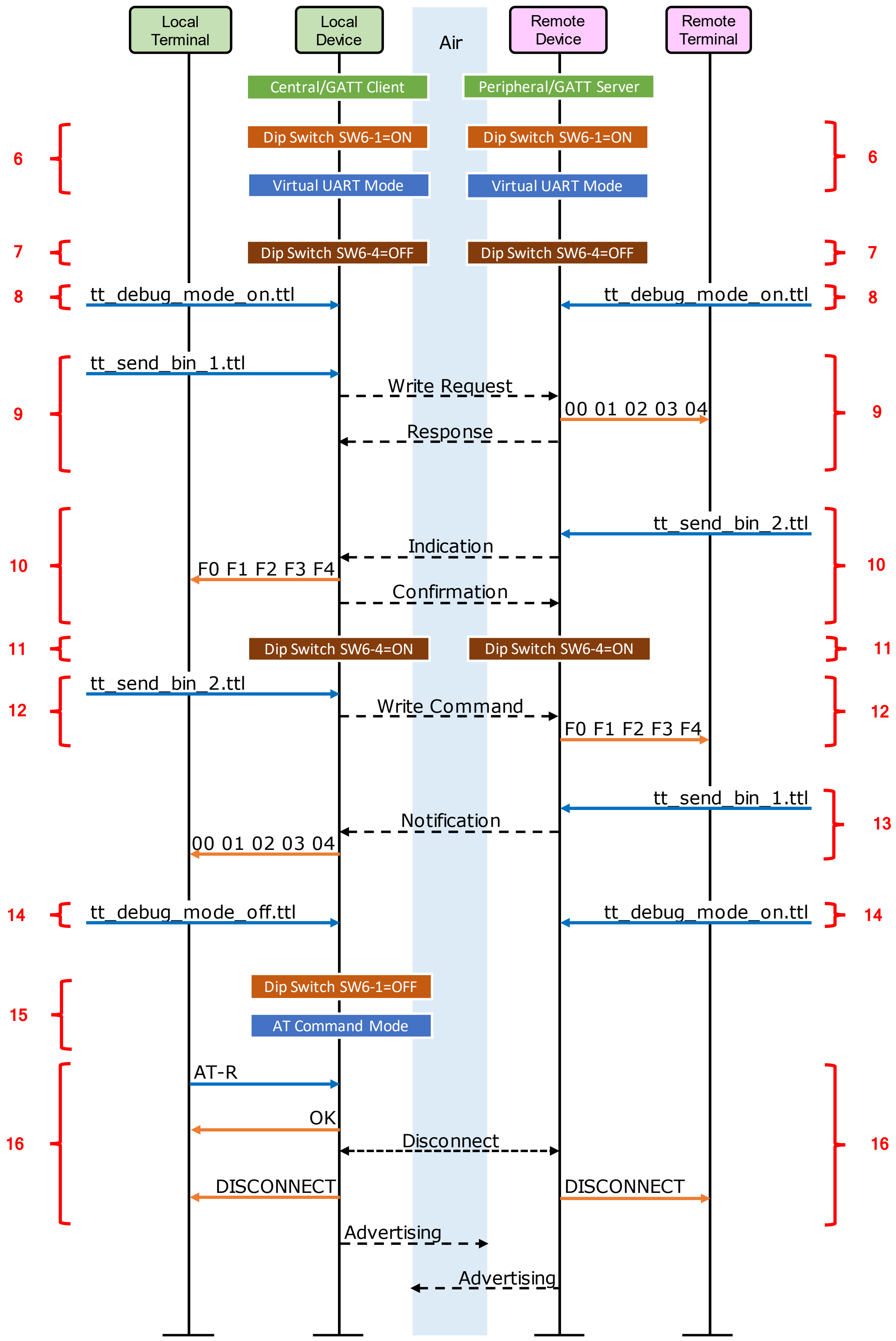


Figure 7-6: Example usage sequence of binary data transmission and reception (2/2)

8. Implementation Details

8.1 Virtual UART Profile

Table 8-1 and Table 8-2 shows the specification of Virtual UART Profile.

Table 8-1: Virtual UART Profile specification (1/2)

Attribute Handle	Attribute type and the value
VUART_HDL_SVC 0x000C	Type: Primary Service Declaration UUID: D68C0001-A21B-11E5-8CB8-0002A5D5C51B UUID for virtual UART service
VUART_HDL_INDICATION_CHAR 0x000D	Type: Characteristic Declaration UUID: D68C0002-A21B-11E5-8CB8-0002A5D5C51B Property: Indicate Used for data transfer from the server to the client
VUART_HDL_INDICATION_VAL 0x000E	Type: Indication Value By setting data to this characteristic and send Indication, the data are sent from the server to the client. Max 20 bytes.
VUART_HDL_INDICATION_CFG 0x000F	Type: Client Characteristic Configuration Descriptor Used for Indication enable / disable of the server from the client
VUART_HDL_WRITE_CHAR 0x0010	Type: Characteristic Declaration UUID: D68C0003-A21B-11E5-8CB8-0002A5D5C51B Property: Write Used for data transfer from the client to the server.
VUART_HDL_WRITE_VAL 0x0011	Type: Write Value By writing data to this characteristic with “Write Request”, the data are sent from the client to the server. Max 20 bytes.

Note: The hex value of attribute handle can be changed depends on profiles included in the firmware.

Table 8-2: Virtual UART Profile specification (2/2)

Attribute Handle	Attribute type and the value
VUART_HDL_NOTIFICATION_CHAR 0x0012	Type: Characteristic Declaration UUID: D68C0004-A21B-11E5-8CB8-0002A5D5C51B Property: Notify Used for data transfer from the server to the client
VUART_HDL_NOTIFICATION_VAL 0x00013	Type: Notification Value By setting data to this characteristic and send Notification, the data are sent from the server to the client. Max 20 bytes.
VUART_HDL_NOTIFICATION_CFG 0x0014	Type: Client Characteristic Configuration Descriptor Used for Notification enable / disable of the server from the client
VUART_HDL_WRITE_NORESP_CHAR 0x0015	Type: Characteristic Declaration UUID: D68C0005-A21B-11E5-8CB8-0002A5D5C51B Property: Write Used for data transfer from the client to the server.
VUART_HDL_WRITE_NORESP_VAL 0x0016	Type: Write Value By writing data to this characteristic with “Write Command”, the data are sent from the client to the server. Max 20 bytes.

Note: The hex value of attribute handle can be changed depends on profiles included in the firmware.

8.2 Advertising

Table 8-3 shows the default settings of advertising.

Table 8-3: Advertising specification

Advertising Type		Connectable undirected advertising (ADV_IND)
Advertising Interval Min		Default: 20 [ms], Low Power: 1.5 [s]
Advertising Interval Max		Default: 30 [ms], Low Power: 3 [s]
Advertising Channel Map		All Channels (37, 38, 39 ch)
Advertising Data		-
	Length of this Data	2 [bytes]
	Data Type	<<Flags>> (0x01)
	Flags	LE General Discoverable Mode BR/EDR Not Supported
	Length of this Data	8 [bytes]
	Data Type	<<Complete Local Name>> (0x09)
	Local Name	REL-BLE
	Length of this Data	17 [bytes]
	Data Type	<<Complete List of 128-bit Service Class UUIDs>> (0x07)
	UUID	D68C0001-A21B-11E5-8CB8-0002A5D5C51B
	Scan Response Data	none

8.3 Connection

Table 8-4 shows the default settings of connection.

Table 8-4: Connection specification

Scan Interval	30 [ms]
Scan Window Size	30 [ms]
Initiator Filter Policy	Ignore Accept List
Peer Address Type	Public Address
Peer BD Address	Specified by AT-C or AT-AP
Own Address Type	Public Address
Minimum of Connection Interval	30 [ms]
Maximum of Connection Interval	30 [ms]
Connection Latency	0 [ms]
Link Supervision Timeout	5 [s]
Minimum CE Length	0 [ms]
Maximum CE Length	50 [ms]

8.4 Pairing

Table 8-5 shows the pairing default settings.

Table 8-5: Pairing specification

Bonding	Bondable Mode
Security Mode	Unauthenticated pairing with encryption
Pairing Method	Just Works
IO capability	No Input No Output
OOB flag	OOB Data not present
Authentication Requirements	No MITM Bonding
Encryption key size	128 [bit]
Initiator key distribution	None
Responder key distribution	Encryption key

8.5 Virtual UART Function API

This section describes virtual UART function definitions and APIs.

8.5.1 Virtual UART Definitions

- Event type enumeration declaration

```
typedef enum {
  // Server Role
  RBLE_VUART_EVENT_SERVER_ENABLE_CMP = 0x01,
  RBLE_VUART_EVENT_SERVER_WRITE_REQ,
  RBLE_VUART_EVENT_SERVER_WRITE_NORESP_REQ,
  RBLE_VUART_EVENT_SERVER_INDICATION_CFM,
  RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP,
  // Client Role
  RBLE_VUART_EVENT_CLIENT_ENABLE_CMP = 0x81,
  RBLE_VUART_EVENT_CLIENT_INDICATION,
  RBLE_VUART_EVENT_CLIENT_NOTIFICATION,
  RBLE_VUART_EVENT_CLIENT_WRITE_RSP,
  RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP,
} RBLE_VUART_EVENT_TYPE;
```

- Event callback function declaration

```
typedef void (*RBLE_VUART_EVENT_HANDLER)(RBLE_VUART_EVENT *event);
```

- Event parameter structure

```
typedef struct RBLE_VUART_EVENT_t {
  RBLE_VUART_EVENT_TYPE type;     Virtual UART event type
  union Event_Vuart_Paramter_u {

    Server role enable completion event
    struct {
      RBLE_STATUS status;           Status
    } server_enable_cmp;

    Server role data receive event (Write Request)
    struct {
      RBLE_STATUS status;           Status
      char value[20];               Received data
      uint16_t len;                 Received data length
    } server_write_req;

    Server role data receive event (Write Command)
    struct {
      RBLE_STATUS status;           Status
      char value[20];               Received data
      uint16_t len;                 Received data length
    } server_write_noresp_req;
```

Server role data send completion event (Indication)
```
struct {
  RBLE_STATUS status;        Status
} server_indicate_cnf;
```

Server role data send completion event (Notification)
```
struct {
  RBLE_STATUS status;        Status
} server_notify_cmp;
```

Client role enable completion event
```
struct {
  RBLE_STATUS status;        Status
} client_enable_cmp;
```

Client role data receive event (Indication)
```
struct {
  RBLE_STATUS status;        Status
  char value[20];            Received data
  uint16_t len;              Received data length
} client_indication;
```

Client role data receive event (Notification)
```
struct {
  RBLE_STATUS status;        Status
  char value[20];            Received data
  uint16_t len;              Received data length
} client_notification;
```

Client role data send completion event (Write Request)
```
struct {
  RBLE_STATUS status;        Status
} client_write_rsp;
```

Client role data send completion event (Write Command)
```
    struct {
      RBLE_STATUS status;        Status
    } client_write_norsp;
  } param;
} RBLE_VUART_EVENT;
```

8.5.2 Function

8.5.2.1 RBLE_VUART_Server_Enable

RBLE_STATUS RBLE_VUART_Server_Enable(uint16_t conhdl, RBLE_VUART_EVENT_HANDLER callback)		
This function enables server role of virtual UART function. The result is informed by RBLE_VUART_EVENT_SERVER_ENABLE_CMP event.		
Parameters:		
	conhdl	Connection handle
	callback	Callback for virtual UART event
Return:		
	RBLE_OK	Success
	RBLE_STATUS_ERROR	Failed due to rBLE mode is in RBLE_MODE_ACTIVE

8.5.2.2 RBLE_VUART_Server_Disable

RBLE_STATUS RBLE_VUART_Server_Disable(void)		
This function disables server role of virtual UART function.		
Parameters:		
	-	-
Return:		
	RBLE_OK	Success

8.5.2.3 RBLE_VUART_Server_Send_Indication

RBLE_STATUS RBLE_VUART_Server_Send_Indication(const char *chars, uint16_t len)		
This function sends data from the server to the client. The data sent by the server are received by the client. After the reception, the client responses with Confirmation. The confirmation is informed to the server by RBLE_VUART_EVENT_SERVER_INDICATION_CFM event.		
Parameters:		
	chars	Received data
	len	Received data length
Return:		
	RBLE_OK	Success
	RBLE_STATUS_ERROR	Failed due to rBLE mode is in RBLE_MODE_ACTIVE

8.5.2.4 RBLE_VUART_Server_Send_Notification

RBLE_STATUS RBLE_VUART_Server_Send_Notification(const char *chars, uint16_t len)		
This function sends data from the server to the client. Executing this function will notify the server of RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP. Notes: This event does not guarantee the sending.		
Parameters:		
	chars	Received data
	len	Received data length
Return:		
	RBLE_OK	Success
	RBLE_STATUS_ERROR	Failed due to rBLE mode is in RBLE_MODE_ACTIVE

8.5.2.5 RBLE_VUART_Client_Enable

RBLE_STATUS RBLE_VUART_Client_Enable(uint16_t conhdl, RBLE_VUART_EVENT_HANDLER callback)		
This function enables client role of virtual UART function. The result is informed by RBLE_VUART_EVENT_CLIENT_ENABLE_CMP event.		
Parameters:		
	conhdl	Connection handle
	callback	Callback for virtual UART event
Return:		
	RBLE_OK	Success
	RBLE_STATUS_ERROR	Failed due to rBLE mode is in RBLE_MODE_ACTIVE

8.5.2.6 RBLE_VUART_Client_Disable

RBLE_STATUS RBLE_VUART_Client_Disable(void)		
This function disables client role of virtual UART function.		
Parameters:		
	-	-
Return:		
	RBLE_OK	Success

8.5.2.7 RBLE_VUART_Client_Send_Chars

RBLE_STATUS RBLE_VUART_Client_Send_Chars(const char *chars, uint16_t len)		
This function sends data from the client to the server. The data sent by the client are received by server. After the reception, the server responses with "Response" to the client. The response is informed to the client by RBLE_VUART_EVENT_CLIENT_WRITE_RSP event.		
Parameters:		
	chars	Received data
	len	Received data length
Return:		
	RBLE_OK	Success
	RBLE_STATUS_ERROR	Failed due to rBLE mode is in RBLE_MODE_ACTIVE

8.5.2.8 RBLE_VUART_Client_Send_Chars_Noresp

RBLE_STATUS RBLE_VUART_Client_Send_Chars_Noresp(const char *chars, uint16_t len)		
This function send data from the client to the server. Executing this function will notify the server of RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP. Notes: This event does not guarantee the sending.		
Parameters:		
	chars	Received data
	len	Received data length
Return:		
	RBLE_OK	Success
	RBLE_STATUS_ERROR	Failed due to rBLE mode is in RBLE_MODE_ACTIVE

8.5.3 Event

This section describes the events defined by virtual UART function.

8.5.3.1 RBLE_VUART_EVENT_SERVER_ENABLE_CMP

RBLE_VUART_EVENT_SERVER_ENABLE_CMP		
This event informs completion of server role enable.		
Parameters:		
	status	server role enable status

8.5.3.2 RBLE_VUART_EVENT_SERVER_WRITE_REQ

RBLE_VUART_EVENT_SERVER_WRITE_REQ		
This event informs that the server has received the data that the client sent using the RBLE_VUART_Client_Send_Chars function.		
Parameters:		
	status	The result of receiving data
	value	Received data
	len	Received data length

8.5.3.3 RBLE_VUART_EVENT_SERVER_WRITE_NORESP_REQ

RBLE_VUART_EVENT_SERVER_WRITE_NORESP_REQ		
This event informs that the server has received the data that the client sent using the RBLE_VUART_Client_Send_Chars_Noresp function.		
Parameters:		
	status	The result of receiving data
	value	Received data
	len	Received data length

8.5.3.4 RBLE_VUART_EVENT_SERVER_INDICATION_CFM

RBLE_VUART_EVENT_SERVER_INDICATION_CFM		
This event informs the completion of sending the data sent by the server using the RBLE_VUART_Server_Send_Indication function.		
Parameters:		
	status	The result of data send

8.5.3.5 RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP

RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP		
This event informs that the server has sent data using the RBLE_VUART_Server_Send_Notification function. Notes: This event does not guarantee the sending.		
Parameters:		
	status	The result of data send

8.5.3.6 RBLE_VUART_EVENT_CLIENT_ENABLE_CMP

RBLE_VUART_EVENT_CLIENT_ENABLE_CMP		
This event informs client role enable completion.		
Parameters:		
	status	

8.5.3.7 RBLE_VUART_EVENT_CLIENT_INDICATION

RBLE_VUART_EVENT_CLIENT_INDICATION		
This event informs that the client has received the data that the server sent using the RBLE_VUART_Server_Send_Indication function.		
Parameters:		
	status	The result of data receive
	value	Received data
	len	Received data length

8.5.3.8 RBLE_VUART_EVENT_CLIENT_NOTIFICATION

RBLE_VUART_EVENT_CLIENT_NOTIFICATION		
This event informs that the client has received the data that the server sent using the RBLE_VUART_Server_Send_Notification function.		
Parameters:		
	status	The result of data receive
	value	Received data
	len	Received data length

8.5.3.9 RBLE_VUART_EVENT_CLIENT_WRITE_RSP

RBLE_VUART_EVENT_CLIENT_WRITE_RSP		
This event informs the completion of sending the data that the client sent using the RBLE_VUART_Client_Send_Chars function.		
Parameters:		
	status	The result of data sending

8.5.3.10 RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP

RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP		
This event signals that the client has sent data using the RBLE_VUART_Client_Send_Chars_Noresp function. Notes: This event does not guarantee the sending.		
Parameters:		
	status	The result of data sending

8.6 Application State Change

Figure 8-1 shows the application state transition diagram. The application changes the state depends on connection and disconnection event and simple AT command execution.

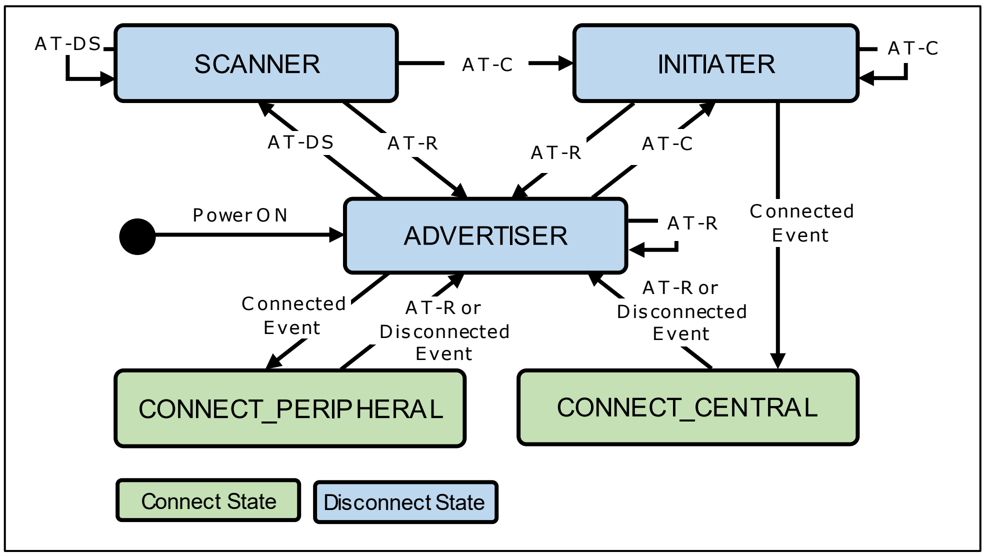


Figure 8-1: Application state diagram

Table 8-6 shows the application state list.

Table 8-6: Application state list

Application State	Description
ADVERTISER	The application is advertising.
SCANNER	The application is scanning neighbor devices by executing AT-DS command. After AT-DS has finished, the application remains in this state.
INITIATER	The application creates a connection to a remote device by executing AT-C.
CONNECT_CENTRAL	Bluetooth LE connection is established as master role. CONNECT_CENTRAL is GATT client.
CONNECT_PERIPHERAL	Bluetooth LE connection is established as slave role. CONNECT_PERIPHERAL is GATT sever.

8.7 Application Detailed Sequence

This section shows the sequence of boot, connection, character transfer and disconnection. Refer to "Bluetooth® Low Energy Protocol Stack API Reference: Basics" (R01UW0088E).

8.7.1 Boot Sequence

Figure 8-2 shows the boot sequence.

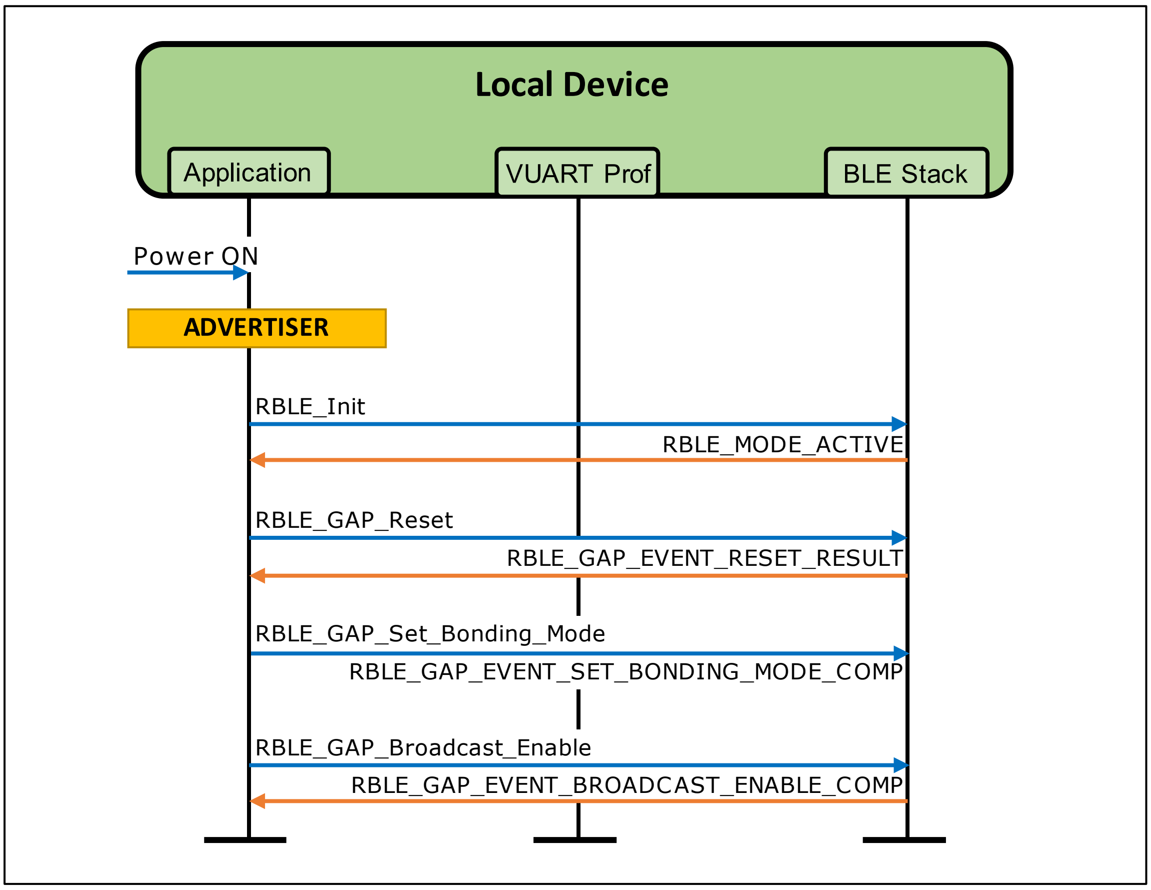

Local Device
Application
VUART Prof
BLE Stack
Power ON
ADVERTISER
RBLE_Init
RBLE_MODE_ACTIVE
RBLE_GAP_Reset
RBLE_GAP_EVENT_RESET_RESULT
RBLE_GAP_Set_Bonding_Mode
RBLE_GAP_EVENT_SET_BONDING_MODE_COMP
RBLE_GAP_Broadcast_Enable
RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP

Figure 8-2: Boot Sequence

8.7.2 Connection Sequence

Figure 8-3 shows the connection sequence.

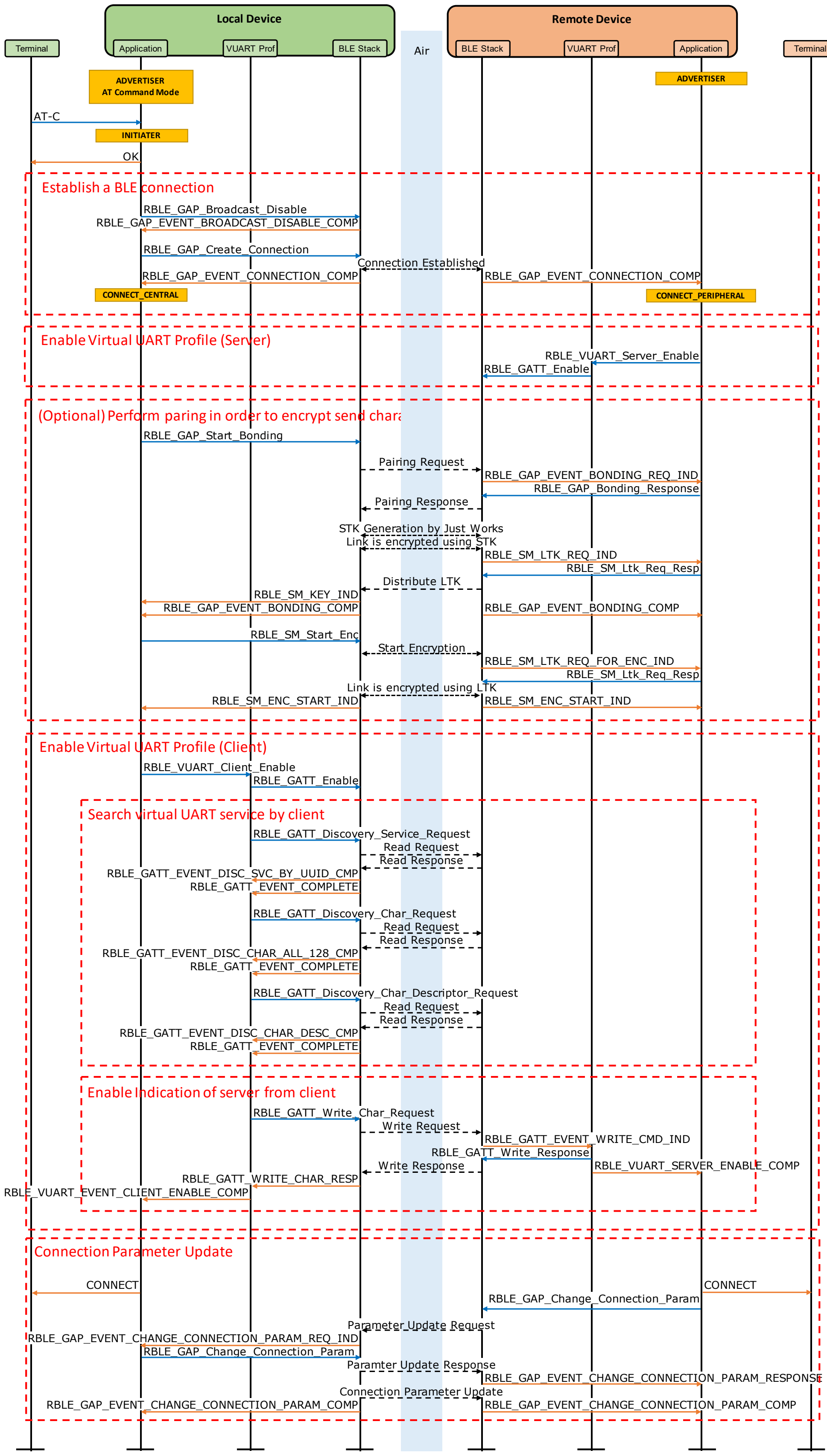

Local Device
Remote Device
Terminal
Application
VUART Prof
BLE Stack
Air
BLE Stack
VUART Prof
Application
Terminal
ADVERTISER AT Command Mode
ADVERTISER
AT-C
INITIATER
OK
Establish a BLE connection
RBLE_GAP_Broadcast_Disable
RBLE_GAP_EVENT_BROADCAST_DISABLE_COMP
RBLE_GAP_Create_Connection
Connection Established
RBLE_GAP_EVENT_CONNECTION_COMP
RBLE_GAP_EVENT_CONNECTION_COMP
CONNECT_CENTRAL
CONNECT_PERIPHERAL
Enable Virtual UART Profile (Server)
RBLE_VUART_Server_Enable
RBLE_GATT_Enable
(Optional) Perform paring in order to encrypt send chara
RBLE_GAP_Start_Bonding
Pairing Request
RBLE_GAP_EVENT_BONDING_REQ_IND
RBLE_GAP_Bonding_Response
Pairing Response
STK Generation by Just Works
Link is encrypted using STK
RBLE_SM_LTK_REQ_IND
RBLE_SM_Ltk_Req_Resp
Distribute LTK
RBLE_SM_KEY_IND
RBLE_GAP_EVENT_BONDING_COMP
RBLE_GAP_EVENT_BONDING_COMP
RBLE_SM_Start_Enc
Start Encryption
RBLE_SM_LTK_REQ_FOR_ENC_IND
RBLE_SM_Ltk_Req_Resp
Link is encrypted using LTK
RBLE_SM_ENC_START_IND
RBLE_SM_ENC_START_IND
Enable Virtual UART Profile (Client)
RBLE_VUART_Client_Enable
RBLE_GATT_Enable
Search virtual UART service by client
RBLE_GATT_Discovery_Service_Request
Read Request
Read Response
RBLE_GATT_EVENT_DISC_SVC_BY_UUID_CMP
RBLE_GATT_EVENT_COMPLETE
RBLE_GATT_Discovery_Char_Request
Read Request
Read Response
RBLE_GATT_EVENT_DISC_CHAR_ALL_128_CMP
RBLE_GATT_EVENT_COMPLETE
RBLE_GATT_Discovery_Char_Descriptor_Request
Read Request
Read Response
RBLE_GATT_EVENT_DISC_CHAR_DESC_CMP
RBLE_GATT_EVENT_COMPLETE
Enable Indication of server from client
RBLE_GATT_Write_Char_Request
Write Request
RBLE_GATT_EVENT_WRITE_CMD_IND
RBLE_GATT_Write_Response
RBLE_VUART_SERVER_ENABLE_COMP
Write Response
RBLE_GATT_WRITE_CHAR_RESP
RBLE_VUART_EVENT_CLIENT_ENABLE_COMP
Connection Parameter Update
CONNECT
CONNECT
RBLE_GAP_Change_Connection_Param
Parameter Update Request
RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_REQ_IND
RBLE_GAP_Change_Connection_Param
Paramter Update Response
RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_RESPONSE
Connection Parameter Update
RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP
RBLE_GAP_EVENT_CHANGE_CONNECTION_PARAM_COMP

Figure 8-3: Connection sequence

8.7.3 Data Transfer Sequence (Write Request/Indication)

Figure 8-4 shows the data transfer sequence.

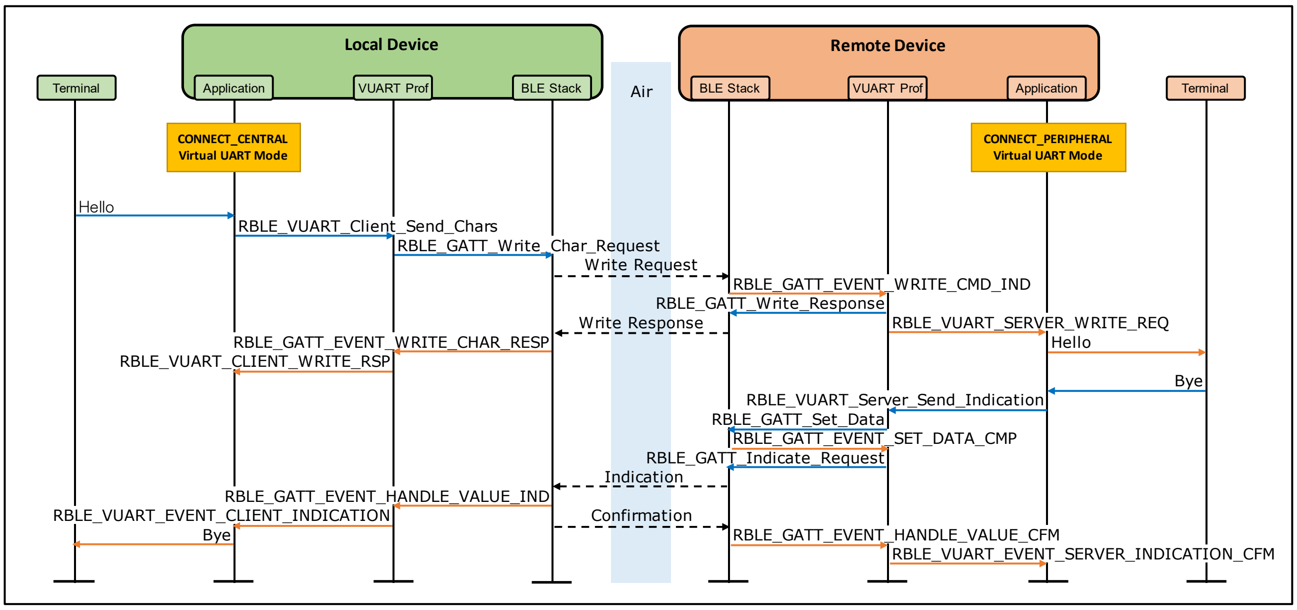

Local Device
Remote Device
Terminal
Application
VUART Prof
BLE Stack
Air
BLE Stack
VUART Prof
Application
Terminal
CONNECT_CENTRAL Virtual UART Mode
CONNECT_PERIPHERAL Virtual UART Mode
Hello
RBLE_VUART_Client_Send_Chars
RBLE_GATT_Write_Char_Request
Write Request
RBLE_GATT_EVENT_WRITE_CMD_IND
RBLE_GATT_Write_Response
Write Response
RBLE_VUART_SERVER_WRITE_REQ
Hello
RBLE_GATT_EVENT_WRITE_CHAR_RESP
RBLE_VUART_CLIENT_WRITE_RSP
Bye
RBLE_VUART_Server_Send_Indication
RBLE_GATT_Set_Data
RBLE_GATT_EVENT_SET_DATA_CMP
RBLE_GATT_Indicate_Request
Indication
RBLE_GATT_EVENT_HANDLE_VALUE_IND
RBLE_VUART_EVENT_CLIENT_INDICATION
Confirmation
Bye
RBLE_GATT_EVENT_HANDLE_VALUE_CFM
RBLE_VUART_EVENT_SERVER_INDICATION_CFM

Figure 8-4: Data transfer sequence (Write Request/Indication)

8.7.4 Data Transfer Sequence (Write Command/Notification)

Figure 8-5 shows the data transfer sequence.

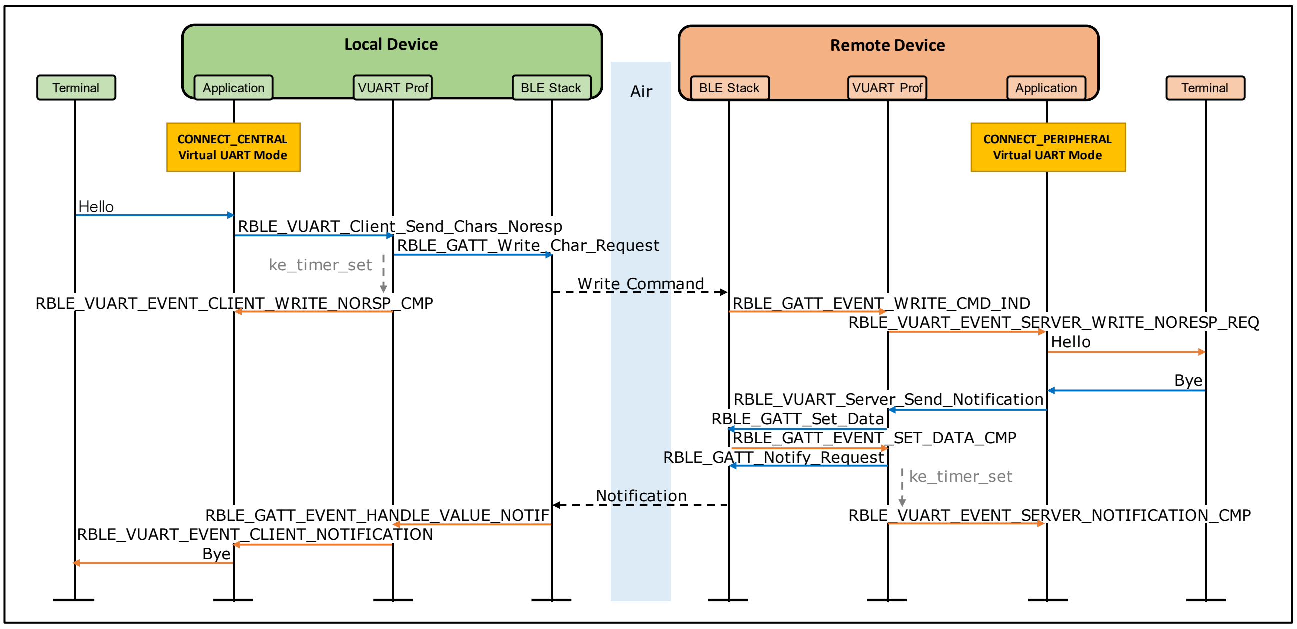

Local Device
Remote Device
Terminal
Application
VUART Prof
BLE Stack
Air
BLE Stack
VUART Prof
Application
Terminal
CONNECT_CENTRAL Virtual UART Mode
CONNECT_PERIPHERAL Virtual UART Mode
Hello
RBLE_VUART_Client_Send_Chars_Noresp
RBLE_GATT_Write_Char_Request
ke_timer_set
Write Command
RBLE_VUART_EVENT_CLIENT_WRITE_NORSP_CMP
RBLE_GATT_EVENT_WRITE_CMD_IND
RBLE_VUART_EVENT_SERVER_WRITE_NORESP_REQ
Hello
Bye
RBLE_VUART_Server_Send_Notification
RBLE_GATT_Set_Data
RBLE_GATT_EVENT_SET_DATA_CMP
RBLE_GATT_Notify_Request
ke_timer_set
Notification
RBLE_GATT_EVENT_HANDLE_VALUE_NOTIF
RBLE_VUART_EVENT_SERVER_NOTIFICATION_CMP
RBLE_VUART_EVENT_CLIENT_NOTIFICATION
Bye

Figure 8-5: Data transfer sequence (Write Command/Notification)

8.7.5 Disconnection Sequence

Figure 8-6 shows the disconnection sequence.

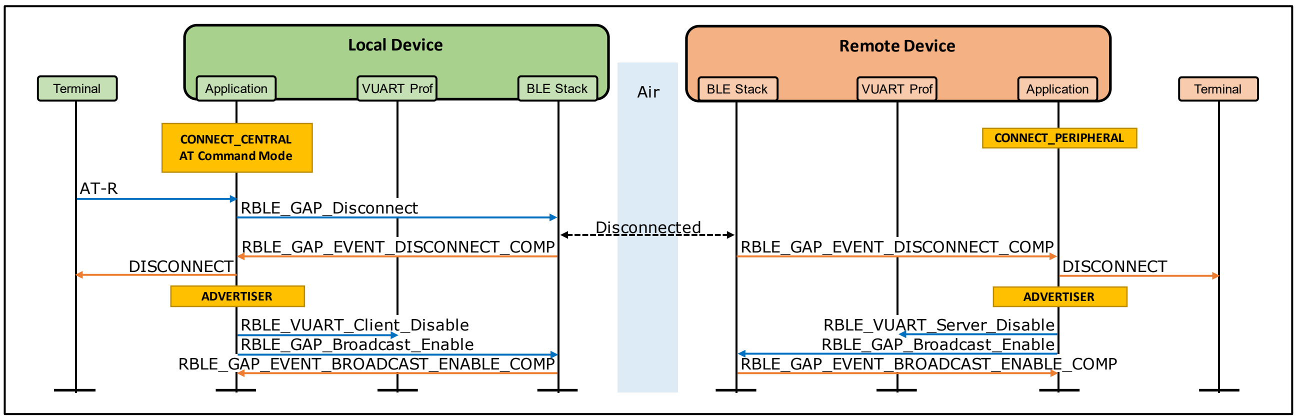

Local Device
Remote Device
Terminal
Application
VUART Prof
BLE Stack
Air
BLE Stack
VUART Prof
Application
Terminal
CONNECT_CENTRAL
AT Command Mode
CONNECT_PERIPHERAL
AT-R
RBLE_GAP_Disconnect
Disconnected
RBLE_GAP_EVENT_DISCONNECT_COMP
RBLE_GAP_EVENT_DISCONNECT_COMP
DISCONNECT
DISCONNECT
ADVERTISER
ADVERTISER
RBLE_VUART_Client_Disable
RBLE_GAP_Broadcast_Enable
RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP
RBLE_VUART_Server_Disable
RBLE_GAP_Broadcast_Enable
RBLE_GAP_EVENT_BROADCAST_ENABLE_COMP

Figure 8-6: Disconnection sequence

8.8 Macro Settings

This section describes macros that set the behavior of this application.

8.8.1 Character data/Binary data transmission and reception setting

Set the type of data (character data / binary data) to transmit / received in virtual UART mode. The default setting is 1 (character data).

Note: In the case of binary data transmission / reception setting, local echo in virtual UART mode is disabled.

r_vuart_app.h, line 47-49

```
47:     #define CFG_VUART_CHAR          (1)                 /* Switch between AT mode and VUART mode. */
48:                                                         /* (0): Binary mode                       */
49:                                                         /* (1): Character mode (default)          */
```

8.8.2 Local echo setting

Set the default setting for local echo to "disable local echo". The default setting is 0 (local echo enabled).

Note: In the case of binary data transmission / reception setting, local echo in virtual UART mode is disabled.

r_vuart_app.h, line 51-53

```
51:     #define CFG_DISABLE_LOCAL_ECHO_BY_DEFAULT   (0)     /* Disable local echo                    */
52:                                                         /* (0): Enable local echo (default)      */
53:                                                         /* (1): Disable local echo               */
```

8.9 Others

8.9.1 Caution when implementing the program to connect to the application

As described in 8.7.2, when AT-C command is executed, the following processing are executed in order. After these steps have finished successfully the application responses with “CONNECT” message on both of the devices. If you implement the program connect to the application also follows these steps. The pairing is optional.

- Establish a BLE connection
- Perform paring in order to encrypt send characters
- Search virtual UART service by client
- Enable Indication of server from client

8.9.2 Read processing of DIP switch that select mode in binary data transmission / reception

The DIP switch state read processing used by the application to select the mode for binary data transmission / reception is shown below.

r_vuart_app.c, line 1916-1921

```
1916:    BOOL read_dipsw1(void)
1917:    {
1918:        /* TRUE (1)  : AT command mode */
1919:        /* FALSE(0)  : VUART mode      */
1920         return ((BOOL)read1_sfr(P1, 0));
1921:    }
```

8.9.3 Read processing of DIP switch that select whether to respond in data transmission / reception

The DIP switch state read processing used by the application to select whether to respond to data transmission / reception is shown below.

r_vuart_app.c, line 1931-1936

```
1931:    BOOL read_dipsw4(void)
1932:    {
1933:        /* TRUE (1)  : with response.    Indication, Write                    */
1934:        /* FALSE(0)  : without response. Notification, Write without response */
1935         return ((BOOL)read1_sfr(P0, 2));
1936     }
```

8.9.4 CFG_CON macro

The CFG_CON macro sets the heap memory of the RF section of the RL78/G1D at the same time as setting the maximum number of connections. Setting CFG_CON 4 (default) is recommended.

8.9.5 Data transmission from the terminal or host microcomputer

The maximum size of data transmitted by UART from a PC terminal or host microcomputer should be 20 bytes.

Examples of RL78/G1D receiving the data transmitted from the terminal via UART and transmitting it via Bluetooth LE communication are shown in 8.9.5.1 and 8.9.5.2.

The settings of this application are baud rate 4800bps, data length 20 bytes, and connection interval 30ms.

8.9.5.1 Example of data transmission during communication with response

Figure 8-7 shows an example of Indication transmission as communication with response.

Indication communication is a communication that combines data transmission by Indication and response reception by Confirmation. Data cannot be sent with the next Indication until a Confirmation is received.

Indication sends the amount of data (2 bytes) received from the terminal in the first Bluetooth LE communication. The next Indication cannot be sent until the Confirmation is received, so the remaining data (18 bytes) received from the terminal is suspended. When Confirmation is received, the remaining data (18 bytes) will be sent by Indication at the next Bluetooth LE communication timing. When Confirmation is received, the remaining data (18 bytes) will be sent by Indication at the next Bluetooth LE communication timing.

At this time, if data of 21 bytes or more is sent, the packet size (20 bytes) of RL78 / G1D may be exceeded and invalid data may be transmitted. When sending data from the terminal, send it in 20 bytes units at intervals larger than the time of the connection interval x3 (100ms in this example).

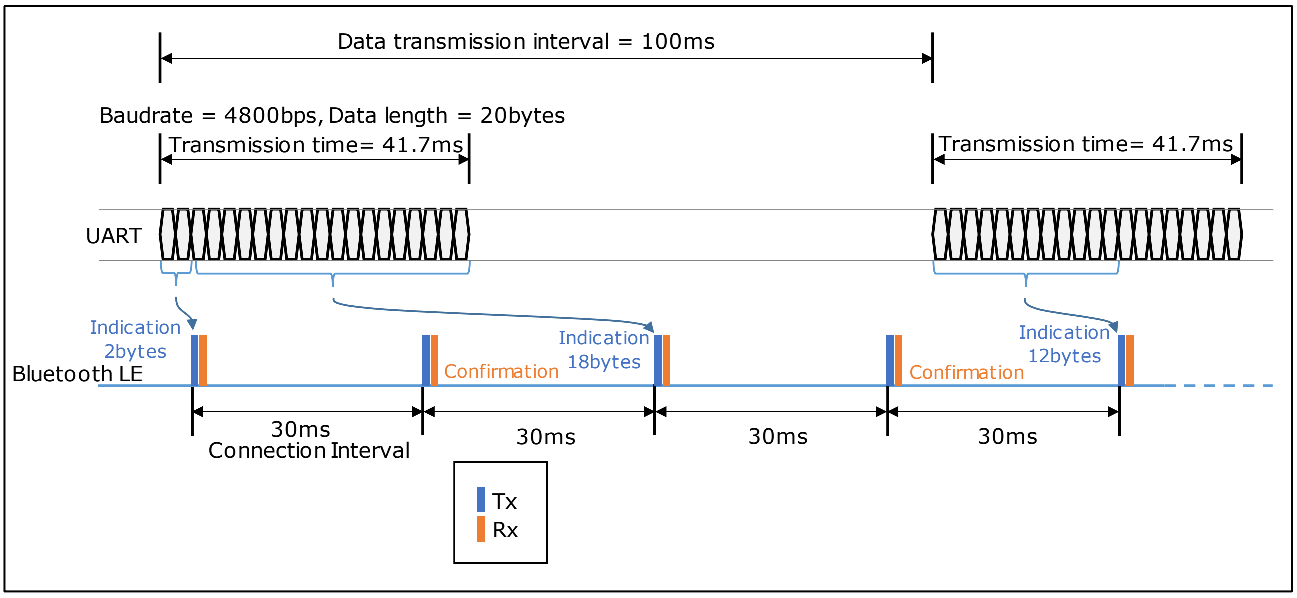


Figure 8-7: Data reception from terminal and Indication communication example

8.9.5.2 Example of data transmission during communication without response

Figure 8-8 shows an example of Notification transmission as communication without response.

Notification communication can be sent data at each timing of Bluetooth LE communication.

Notification will send the amount of data (2 bytes) received from the terminal in the first Bluetooth LE communication. If there is data (15 bytes) received from the terminal before the next Bluetooth LE communication timing, it will be sent by Notification. In this way, Notification allows you to send data at Bluetooth LE communication timing if there is data to send.

When sending data from the terminal via UART, send it in 20 bytes units as in response communication. Also, if data is sent continuously without any gaps, an error may occur in the UART. The data should be sent at an interval larger than the connection interval (50ms in this example).

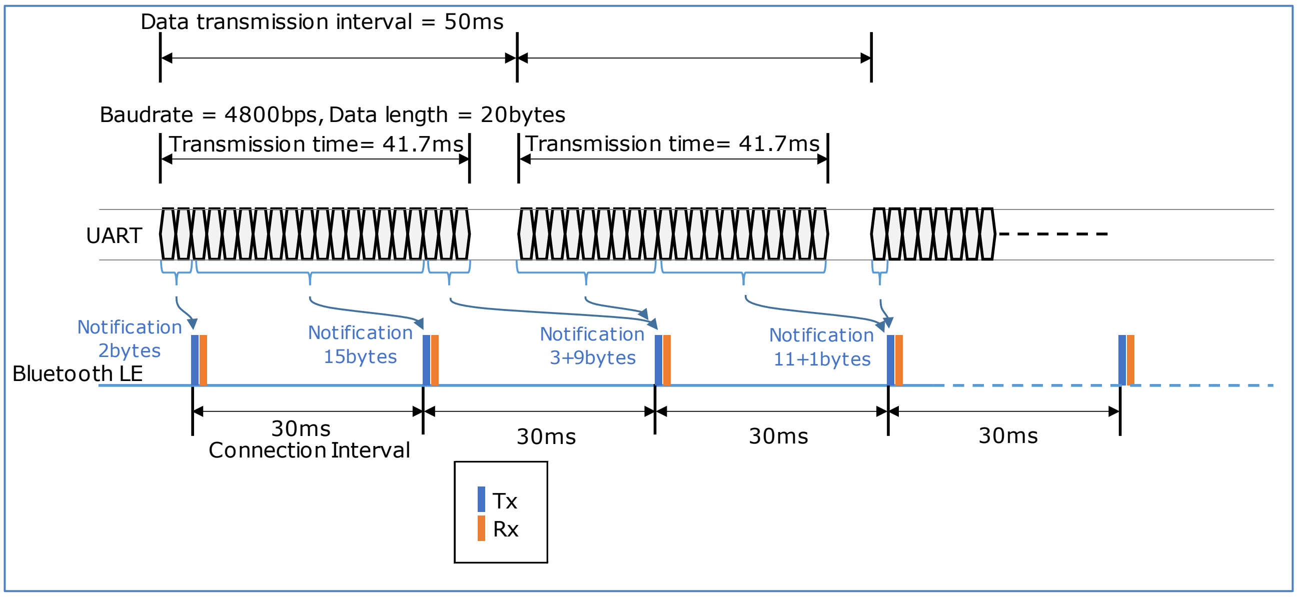


Figure 8-8: Data reception from terminal and Notification communication example

9. Appendix

9.1 ROM size, RAM size

Table 9-1 and Table 9-2 show the ROM size and RAM size when the virtual UART application is used with the BLE protocol stack V1.21.

Table 9-1: ROM size, RAM size (Character data transmission / reception)

Compiler	ROM size	RAM size
RL78 Family C Compiler Package V1 V1.09.00	127,002	11,535
Renesas CA78K0R V1.72	155,849	11,609

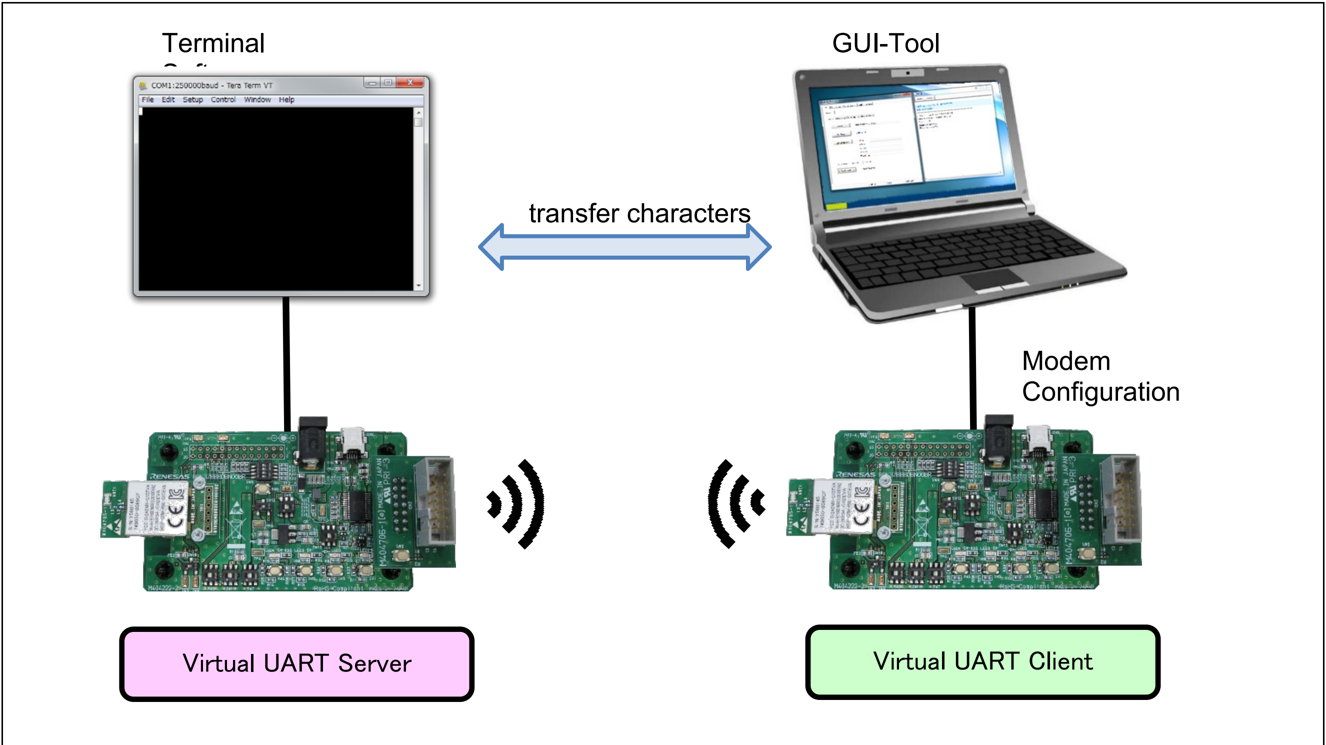
Table 9-2: ROM size, RAM size (Binary data transmission / reception)

Compiler	ROM size	RAM size
RL78 Family C Compiler Package V1 V1.09.00	126,684	11,535
Renesas CA78K0R V1.72	155,676	11,609

9.2 Operational Check by Using the GUI-Tool

This section describes the operation check procedure of Virtual UART by using the GUI-Tool (R01AN2469).

Figure 9-1 shows overview diagram of operational check by using the GUI-Tool. This application operates as a Virtual UART server, and the GUI-Tool operates as a Virtual UART client. It is possible to transfer characters to each other.



Note: GUI-Tool Version 1.12 (R01AN2469XX0112) or later is required in order to check the operation.

Figure 9-1: Operation check by using the GUI-Tool

Hereafter, the combination of evaluation board (Virtual UART application was written) and terminal software is mentioned as “Virtual UART Server”. And the combination of evaluation board (Modem Configuration Hex file was written) and the GUI-Tool is mentioned as “Virtual UART Client”.

9.2.1 Preparation

- Virtual UART Server
  In accordance with the procedures described in the following section, write the firmware onto RL78/G1D evaluation board and then launch a terminal software on PC.
  7.2 Build Procedure
  7.3 Preparation for Execution Environment
  This section uses character data transmission / reception (RL78_G1D_CCE(VUART_CHAR).hex).

- Virtual UART Client
  Write a Modem configuration Hex file (any of the build environment) that is included in the package of BLE protocol stack onto RL78/G1D evaluation board, then launch the GUI-Tool.
  Notes: 1. In order to access the service on the Virtual UART Server by using the GATT APIs, it does not matter the profile type of Hex file to be written.
  2. Refer to "Bluetooth Low Energy Protocol Stack GUI Tool" (R01AN2469) “6. Utilization” about how to launch the GUI-Tool.

9.2.2 Operation

It is possible to transfer characters by operating Virtual UART Server and Virtual UART Client in the following procedure.

1. Discoverable Mode (Virtual UART Server)
   Push the RESET button (SW5) on RL78/G1D evaluation board that operates as Virtual UART Server.
   After the reset, the virtual UART application transitions to discoverable mode automatically, and then start the Advertising.

2. Device Discovery (Virtual UART Client)
   Search discoverable mode devices by operating the GUI-Tool.
   (1) Activate [Scanning] tab of [GAP] tab.
   (2) Select "General Discovery" in the Discovery group.
   (3) Press [Discover] button.

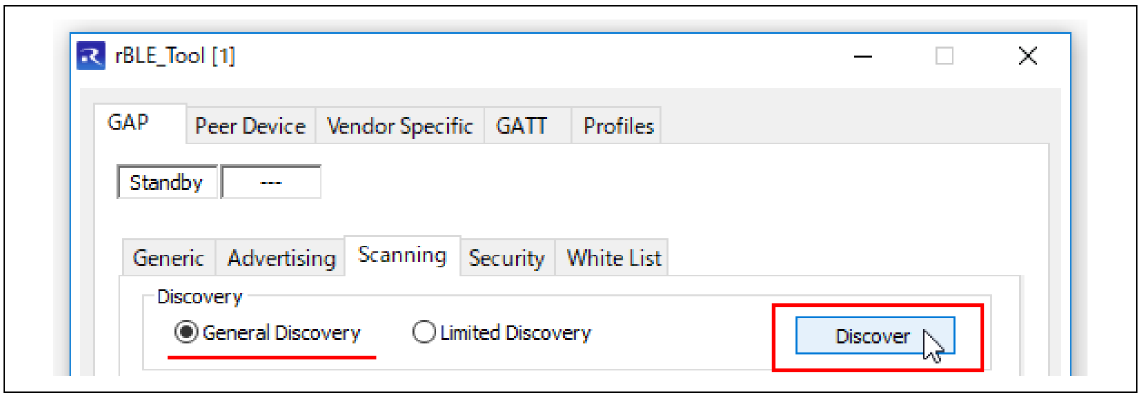
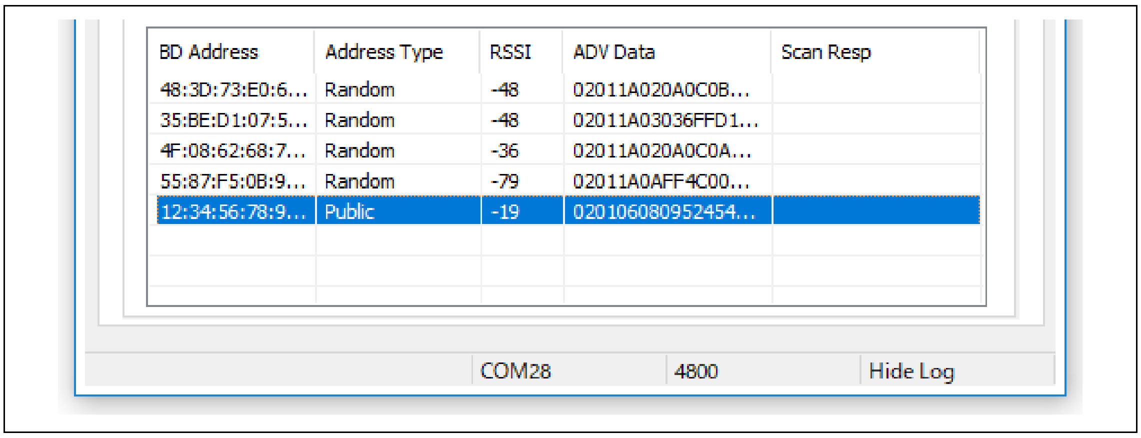


Figure 9-2: Device Discovery

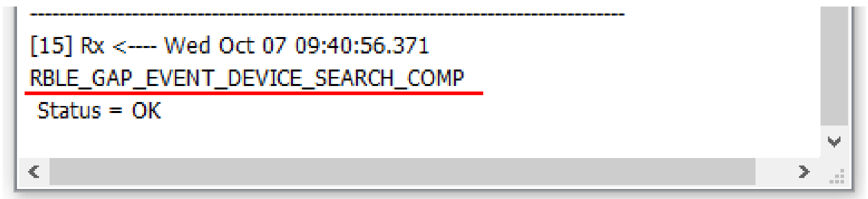
   (4) Discoverable mode devices will display in the list of Received Advertising data.



BD Address	Address Type	RSSI	ADV Data	Scan Resp
48:3D:73:E0:6...	Random	-48	02011A020A0C0B...	
35:BE:D1:07:5...	Random	-48	02011A03036FFD1...	
4F:08:62:68:7...	Random	-36	02011A020A0C0A...	
55:87:F5:0B:9...	Random	-79	02011A0AFF4C00...	
12:34:56:78:9...	Public	-19	020106080952454...	

Figure 9-3: Result of Device Discovery

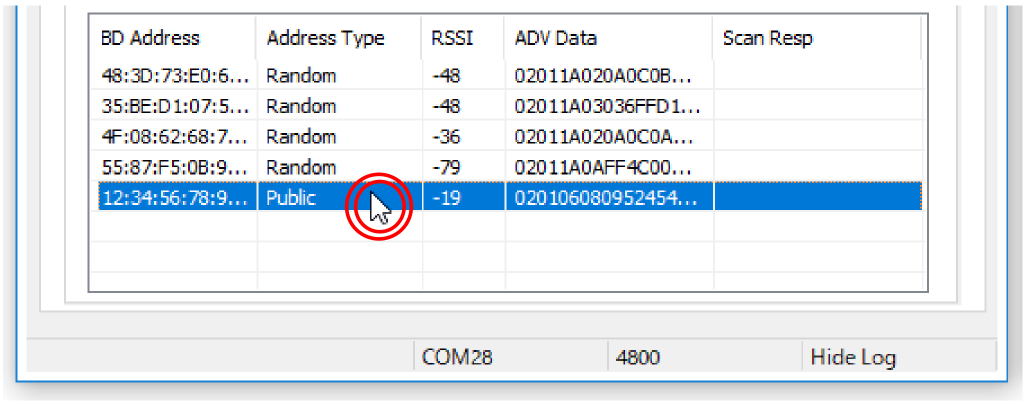
3. Connection (Virtual UART Client)
   Initiate connection to Virtual UART Server by operating the GUI-Tool.
   (1) Confirm that the RBLE_GAP_EVENT_DEVICE_SEARCH_COMP event has occurred in the log dialog.



```
--------------------------------------------------------------------------------
[15] Rx <---- Wed Oct 07 09:40:56.371
RBLE_GAP_EVENT_DEVICE_SEARCH_COMP
 Status = OK
```

Figure 9-4: Confirmation of RBLE_GAP_EVENT_DEVICE_SEARCH_COMP event

   (2) In [Scanning] tab, Double-click onto the target device in the list of Received Advertising data.

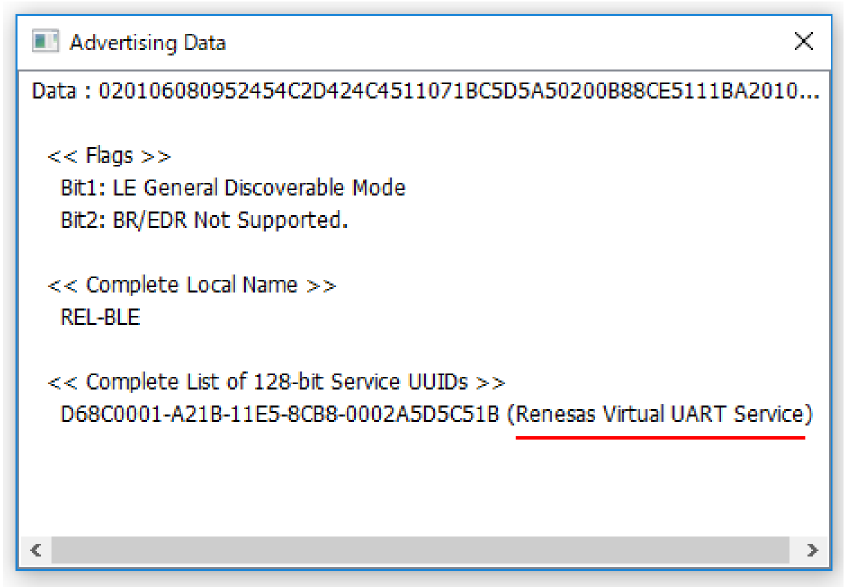


BD Address	Address Type	RSSI	ADV Data	Scan Resp
48:3D:73:E0:6...	Random	-48	02011A020A0C0B...	
35:BE:D1:07:5...	Random	-48	02011A03036FFD1...	
4F:08:62:68:7...	Random	-36	02011A020A0C0A...	
55:87:F5:0B:9...	Random	-79	02011A0AFF4C00...	
12:34:56:78:9...	Public	-19	020106080952454...	

COM28 4800 Hide Log

Figure 9-5: Select Device

Tips: It will be displayed the Advertising data analysis dialog by [Ctrl] key + double-clicking arbitrary row in the list of Received Advertising data.
The device which operates as Virtual UART Server contains “Renesas Virtual UART Service” to the <<Complete List of 128-bit Service UUIDs>>



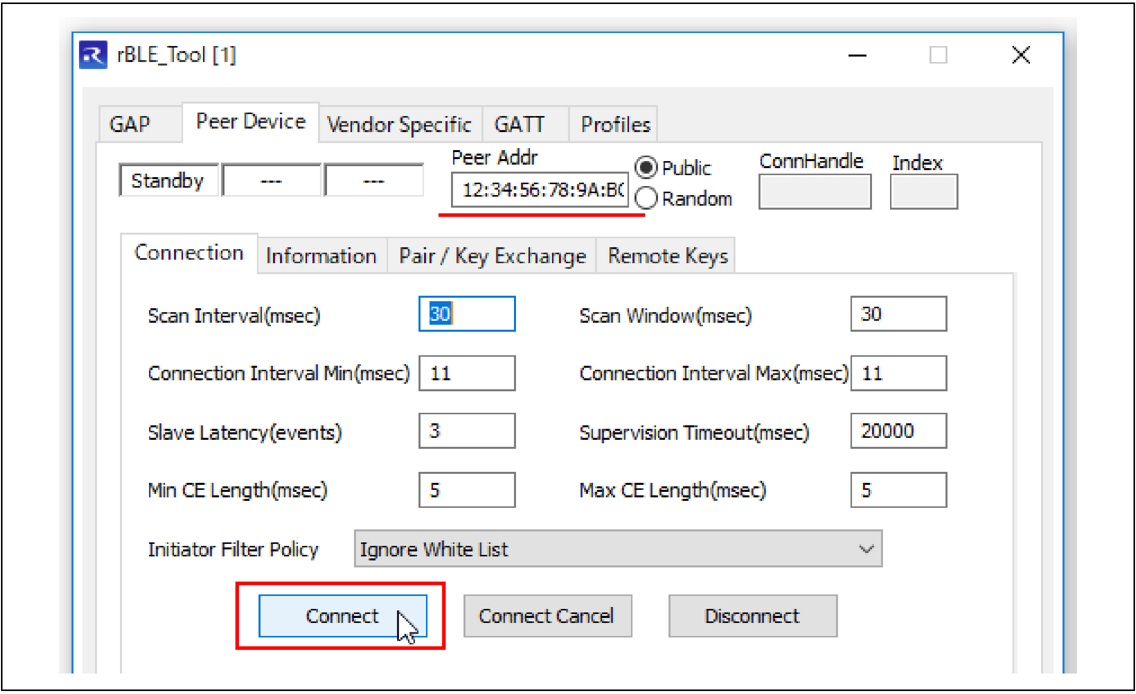
Advertising Data

```
Data : 020106080952454C2D424C4511071BC5D5A50200B88CE5111BA2010...

  << Flags >>
    Bit1: LE General Discoverable Mode
    Bit2: BR/EDR Not Supported.

  << Complete Local Name >>
    REL-BLE

  << Complete List of 128-bit Service UUIDs >>
    D68C0001-A21B-11E5-8CB8-0002A5D5C51B (Renesas Virtual UART Service)
```

Figure 9-6: Advertising Data Analysis Dialog

(3) Activate [Connection] tab of [Peer Device] tab.
At this time, make sure that the target device address will reflect to “Peer Addr” field in top of [Peer Device] tab.

(4) Initiate a connection to Virtual UART Server by pressing [Connect] button.

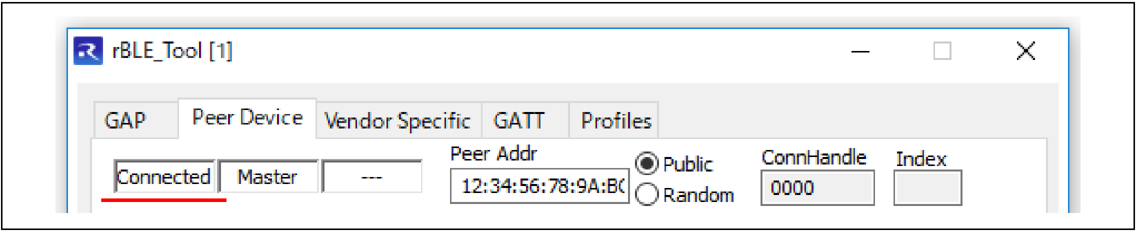


Standby | --- | ---
Peer Addr: 12:34:56:78:9A:B(
[✓ Public] Random
ConnHandle:
Index:

Scan Interval(msec): 30
Scan Window(msec): 30
Connection Interval Min(msec): 11
Connection Interval Max(msec): 11
Slave Latency(events): 3
Supervision Timeout(msec): 20000
Min CE Length(msec): 5
Max CE Length(msec): 5
Initiator Filter Policy: Ignore White List

Figure 9-7: Initiate Connection

(5) When a connection is established, the State display in top of [Peer Device] tab is changed to “Connected”.



Connected | Master | ---
Peer Addr: 12:34:56:78:9A:B(
[✓ Public] Random
ConnHandle: 0000
Index:

Figure 9-8: Established Connection

4. Service Discovery (Virtual UART Client)
   Discover services and characteristics on Virtual UART Server by operating the GUI-Tool.
   - Service Discovery
     Discover all services on Virtual UART Server.
     (1) Activate [Service Discovery] tab of [GATT]→[Client] tab.
     (2) Select “Discover All Primary Services” in the Discovery Type drop-down list, and press [Discover] button.

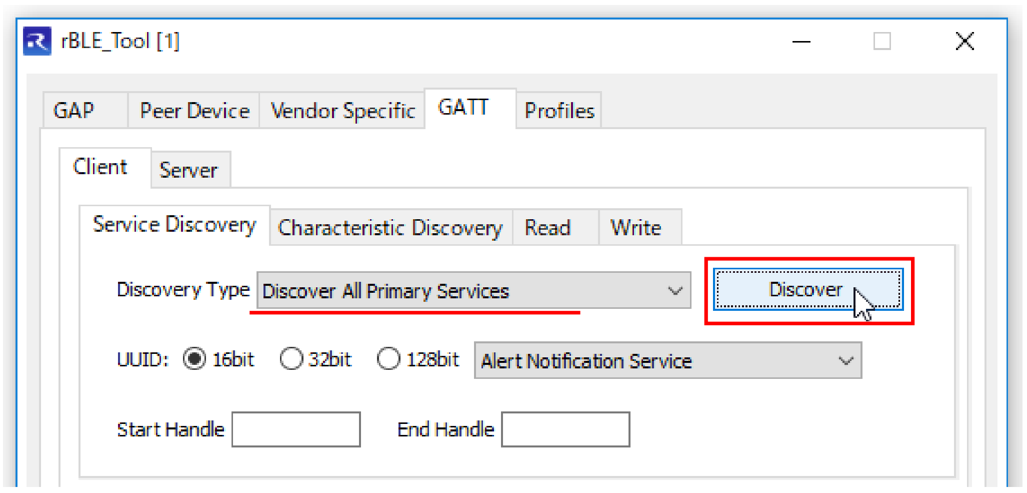
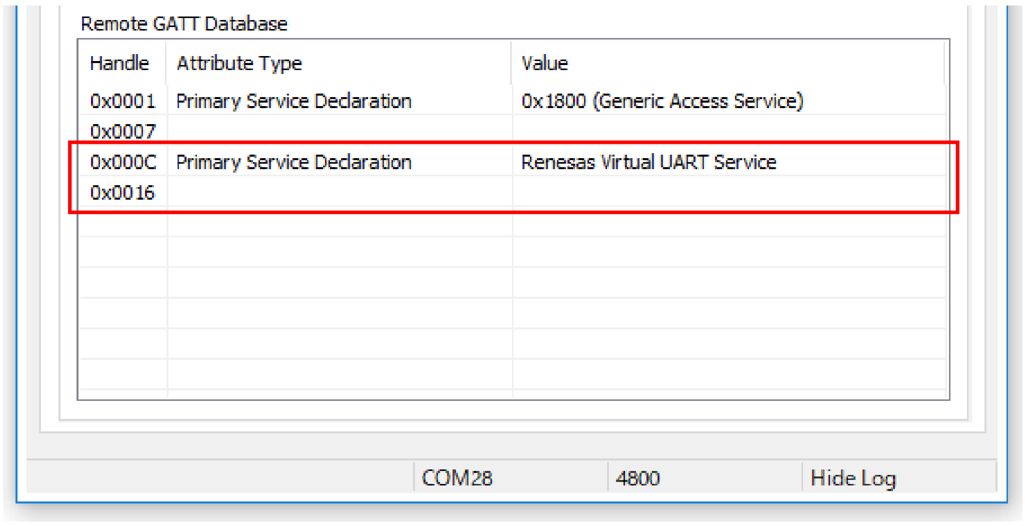


Figure 9-9: Service Discovery

(3) Acquired service information is displayed in the list of “Remote GATT Database”.



Remote GATT Database

Handle	Attribute Type	Value
0x0001	Primary Service Declaration	0x1800 (Generic Access Service)
0x0007		
0x000C	Primary Service Declaration	Renesas Virtual UART Service
0x0016		

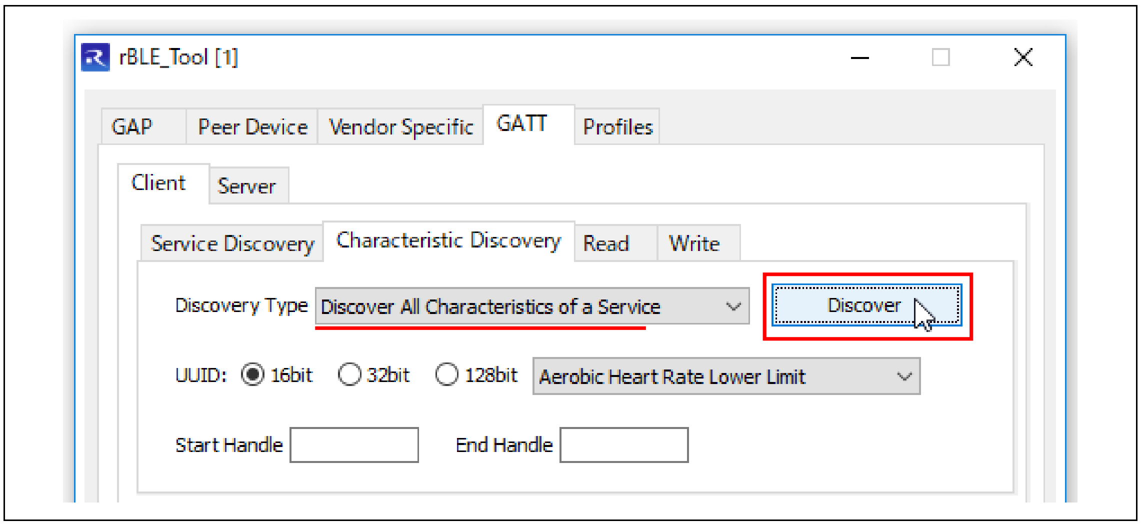
Figure 9-10: Result of Service Discovery

- Characteristic Discovery

Discover all service characteristics on Virtual UART Server.

(1) Activate [Characteristic Discovery] tab of [GATT]→[Client] tab.

(2) Select "Discover Characteristics of a Service" in the Discovery Type drop-down list, and press [Discover] button.



Discovery Type: Discover All Characteristics of a Service

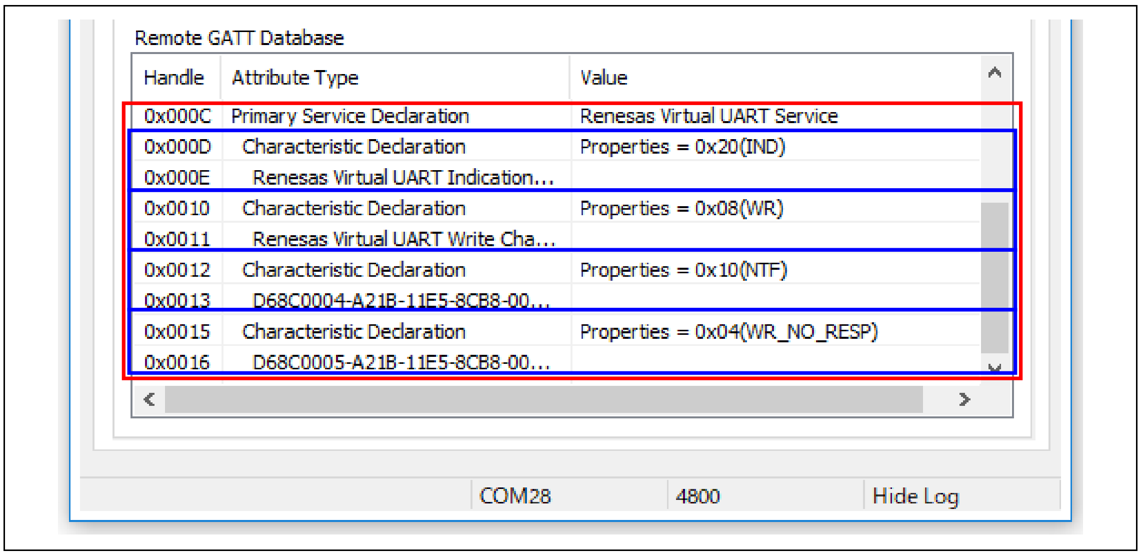
UUID: (● 16bit) ○ 32bit ○ 128bit — Aerobic Heart Rate Lower Limit

Start Handle:

End Handle:

Figure 9-11: Characteristic Discovery

(3) Acquired characteristic information is displayed in the list of "Remote GATT Database".



Remote GATT Database

Handle	Attribute Type	Value
0x000C	Primary Service Declaration	Renesas Virtual UART Service
0x000D	Characteristic Declaration	Properties = 0x20(IND)
0x000E	Renesas Virtual UART Indication...	
0x0010	Characteristic Declaration	Properties = 0x08(WR)
0x0011	Renesas Virtual UART Write Cha...	
0x0012	Characteristic Declaration	Properties = 0x10(NTF)
0x0013	D68C0004-A21B-11E5-8CB8-00...	
0x0015	Characteristic Declaration	Properties = 0x04(WR_NO_RESP)
0x0016	D68C0005-A21B-11E5-8CB8-00...	

COM28 4800 Hide Log

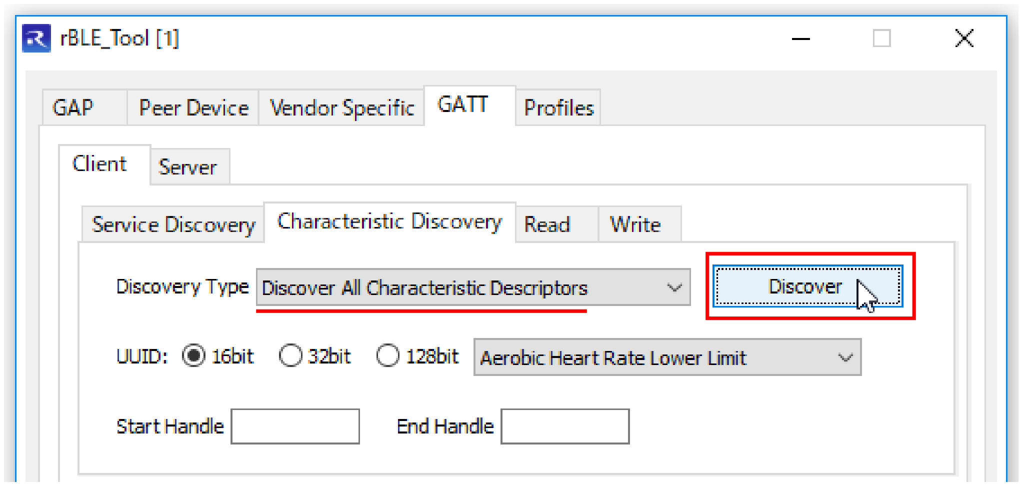
Figure 9-12: Result of Characteristic Discovery

- Characteristic Descriptor Discovery

Discover characteristic descriptors of a characteristic on Virtual UART Server.

(1) Activate [Characteristic Discovery] tab of [GATT]→[Client] tab.

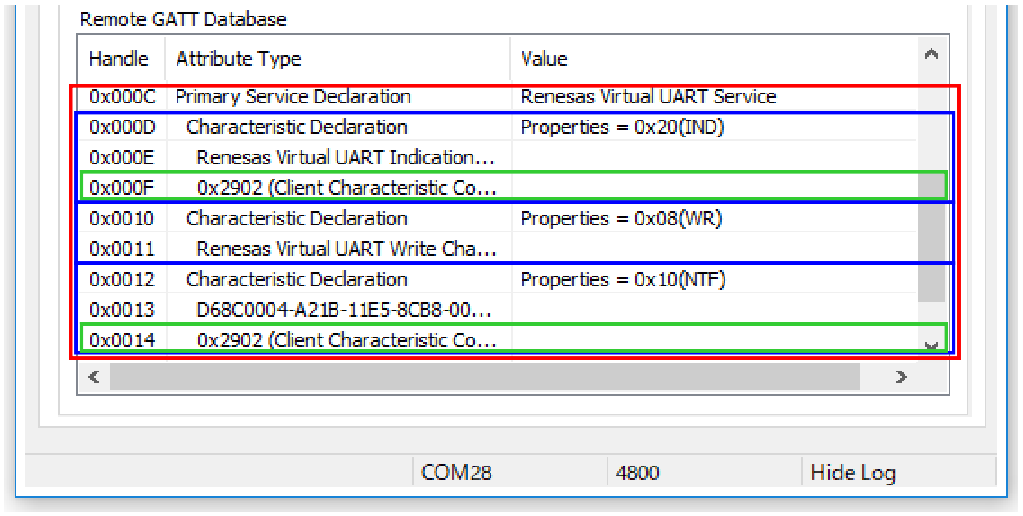
(2) Select “Discover All Characteristic Descriptors” in the Discovery Type drop-down list, and press [Discover] button.



Discovery Type: Discover All Characteristic Descriptors
UUID: (● 16bit) ○ 32bit ○ 128bit — Aerobic Heart Rate Lower Limit
Start Handle:
End Handle:

Figure 9-13: Characteristic Descriptor Discovery

(3) Acquired characteristic descriptor information is displayed in the list of “Remote GATT Database”.

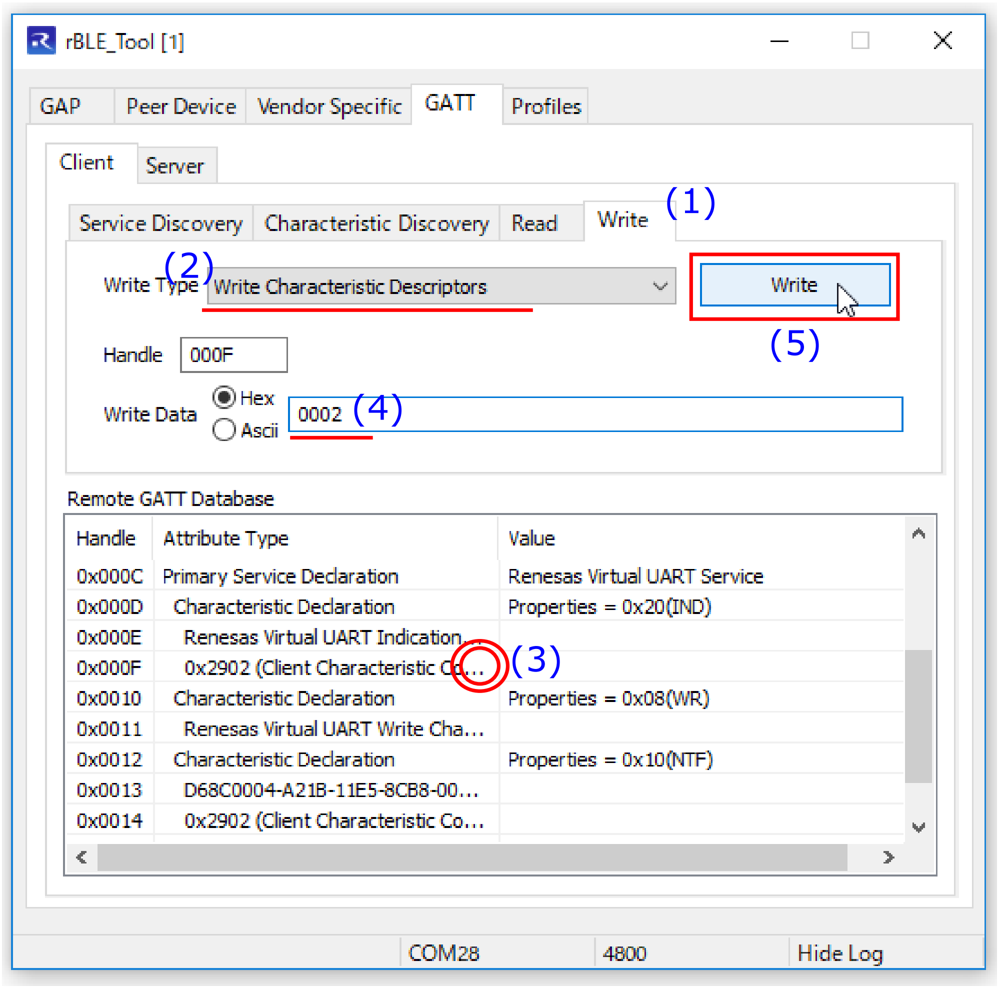


Remote GATT Database

Handle	Attribute Type	Value
0x000C	Primary Service Declaration	Renesas Virtual UART Service
0x000D	Characteristic Declaration	Properties = 0x20(IND)
0x000E	Renesas Virtual UART Indication…	
0x000F	0x2902 (Client Characteristic Co…	
0x0010	Characteristic Declaration	Properties = 0x08(WR)
0x0011	Renesas Virtual UART Write Cha…	
0x0012	Characteristic Declaration	Properties = 0x10(NTF)
0x0013	D68C0004-A21B-11E5-8CB8-00…	
0x0014	0x2902 (Client Characteristic Co…	

Figure 9-14: Result of Characteristic Descriptor Discovery

5. Enable Indication (Virtual UART Client)
Enable character transfer from Virtual UART Server to Virtual UART Client (Indication, Notification) by operating the GUI-Tool.
(1) Activate [Write] tab of [GATT]→[Client] tab.
(2) Select "Write Characteristic Descriptors" in the Write Type drop-down list.
(3) In the list of "Remote GATT Database", double-click the Client Characteristic Configuration Descriptor within "Renesas Virtual UART Indication Characteristic".
By double-clicking, the handle value of Client Characteristic Configuration Descriptor will reflect to "Handle" field in [Write] tab.
(4) Enter the value of "0002" (it means that "Indications enabled") in "Write Data" field.
(5) By pressing [Write] button, write the characteristic descriptor value to Virtual UART Server.



Write Type: Write Characteristic Descriptors
Handle: 000F
Write Data: 0002

Remote GATT Database

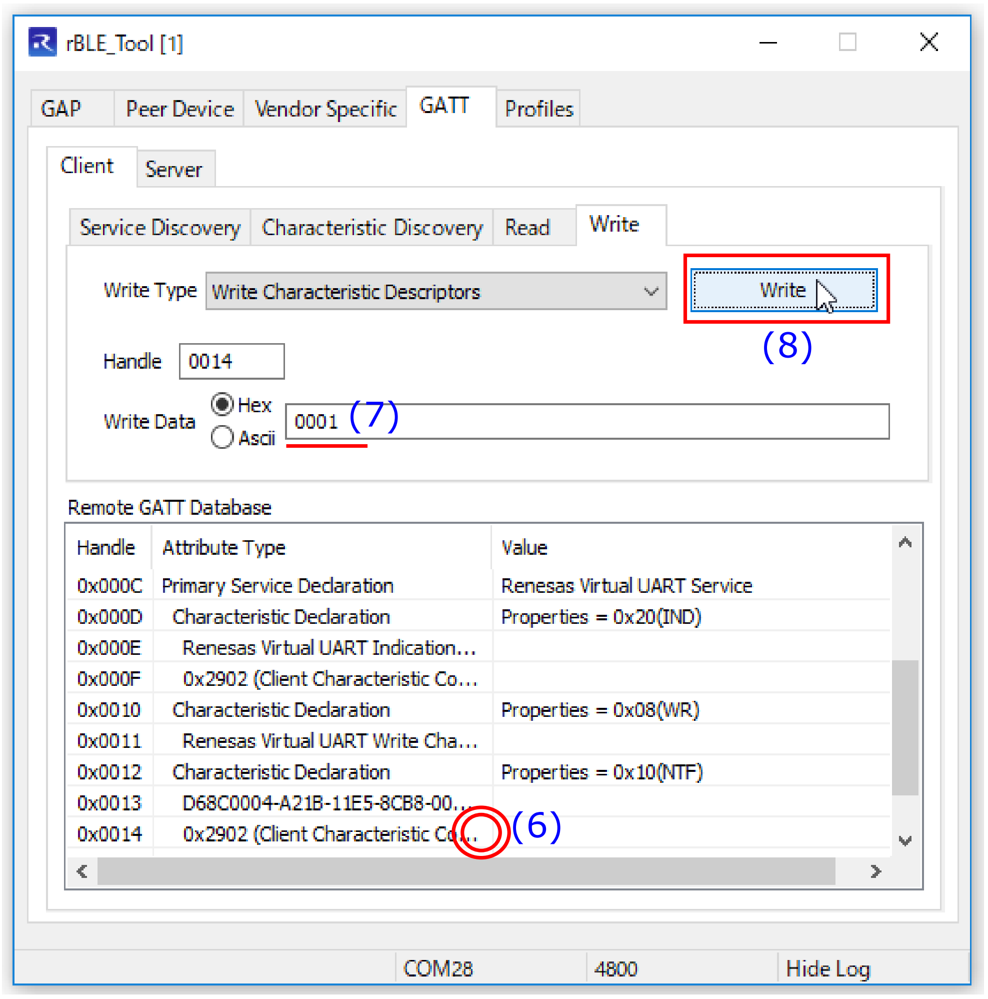
Handle	Attribute Type	Value
0x000C	Primary Service Declaration	Renesas Virtual UART Service
0x000D	Characteristic Declaration	Properties = 0x20(IND)
0x000E	Renesas Virtual UART Indication...	
0x000F	0x2902 (Client Characteristic Co...	
0x0010	Characteristic Declaration	Properties = 0x08(WR)
0x0011	Renesas Virtual UART Write Cha...	
0x0012	Characteristic Declaration	Properties = 0x10(NTF)
0x0013	D68C0004-A21B-11E5-8CB8-00...	
0x0014	0x2902 (Client Characteristic Co...	

Figure 9-15: Enable Indication

(6) In the list of “Remote GATT Database”, double-click the Client Characteristic Configuration Descriptor within “D68C0004-A21B-11E5-8CB8-0002A5D5C51B” (Renesas Virtual UART Notification Characteristic).
By double-clicking, the handle value of Client Characteristic Configuration Descriptor will reflect to “Handle” field in [Write] tab.

(7) Enter the value of “0001” (it means that “Notification enabled”) in “Write Data” field.

(8) By pressing [Write] button, write the characteristic descriptor value to Virtual UART Server.

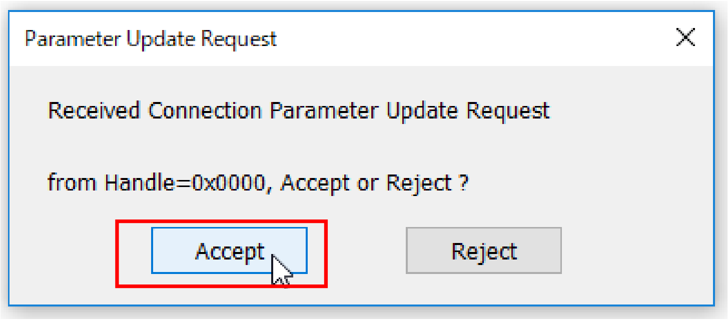


Write Type: Write Characteristic Descriptors
Handle: 0014
Write Data: 0001

Handle	Attribute Type	Value
0x000C	Primary Service Declaration	Renesas Virtual UART Service
0x000D	Characteristic Declaration	Properties = 0x20(IND)
0x000E	Renesas Virtual UART Indication…	
0x000F	0x2902 (Client Characteristic Co…	
0x0010	Characteristic Declaration	Properties = 0x08(WR)
0x0011	Renesas Virtual UART Write Cha…	
0x0012	Characteristic Declaration	Properties = 0x10(NTF)
0x0013	D68C0004-A21B-11E5-8CB8-00…	
0x0014	0x2902 (Client Characteristic Co…	

Figure 9-16: Enable Notification

(9) The Parameter Update Request dialog is displayed. Press "Accept".

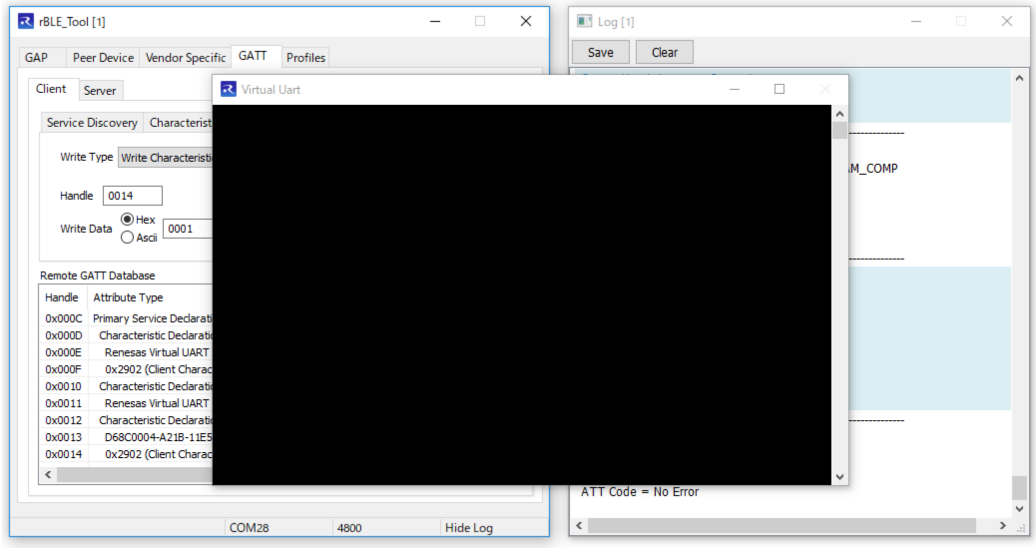


Received Connection Parameter Update Request

from Handle=0x0000, Accept or Reject ?

Figure 9-17: Accept Parameter Update Request

(10) Console window is displayed when the response is received from Virtual UART Server.



Write Type: Write Characteristi

Handle: 0014

Write Data: (● Hex / ○ Ascii) 0001

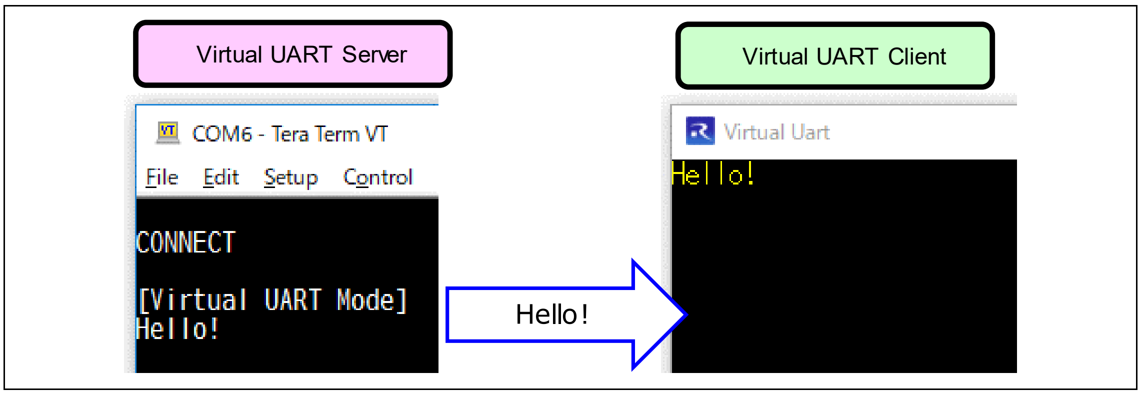
Remote GATT Database

Handle	Attribute Type
0x000C	Primary Service Declarati
0x000D	Characteristic Declarati
0x000E	Renesas Virtual UART
0x000F	0x2902 (Client Charac
0x0010	Characteristic Declarati
0x0011	Renesas Virtual UART
0x0012	Characteristic Declarati
0x0013	D68C0004-A21B-11E5
0x0014	0x2902 (Client Charac

```
---------------
M_COMP
---------------
---------------
ATT Code = No Error
```

Figure 9-18: Console Window

6. Character Transfer (Virtual UART Server / Virtual UART Client)
   - Character transfer from Virtual UART Server
     (1) Input ESC key on the terminal software in order to switch the application mode to Virtual UART mode.
     (2) Type arbitrary characters (e.g. "Hello!") on the terminal software.
     (3) Input characters are displayed by yellow characters in the console window of Virtual UART Client.

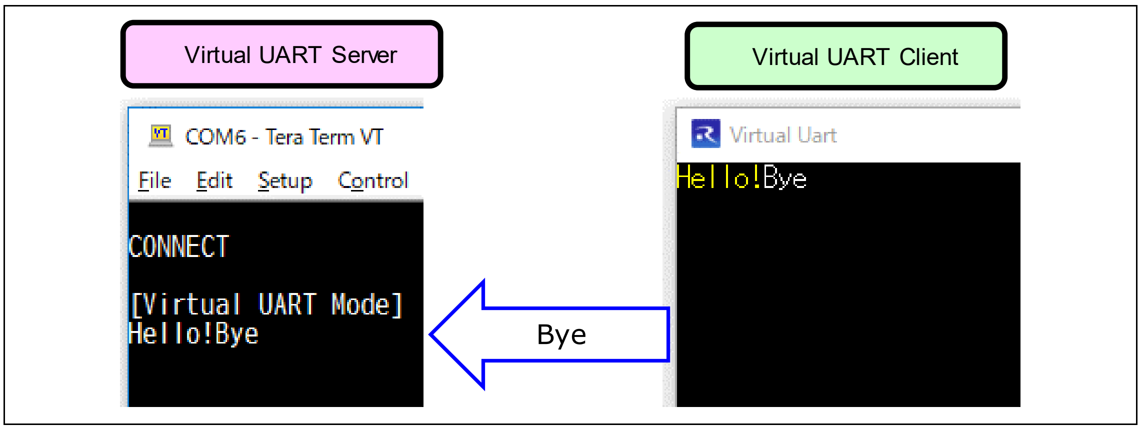


```
CONNECT

[Virtual UART Mode]
Hello!
```

```
Hello!
```

Figure 9-19: Character Transfer (Server→Client)

   - Character transfer from Virtual UART Client
     (1) Type arbitrary characters (e.g. "Bye") on the console window.
     (2) Input characters are displayed in the terminal software of Virtual UART Server.



```
CONNECT

[Virtual UART Mode]
Hello!Bye
```

```
Hello!Bye
```

Figure 9-20: Character Transfer (Client→Server)

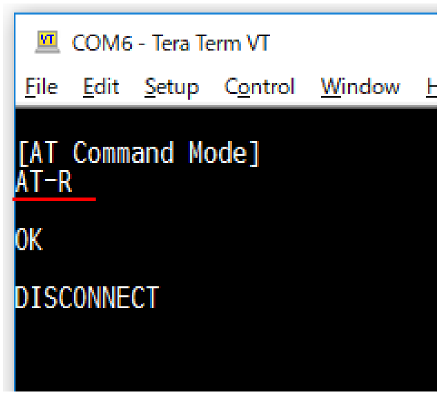
7. Disconnection (Virtual UART Server / Virtual UART Client)

- Disconnect from Virtual UART Server

  (1) Input ESC key on the terminal software in order to switch the application mode to Simple AT command mode.

  (2) Execute "AT-R" on the terminal software (Disconnect the established connection).

  (3) When the connection is terminated, it will be displayed "DISCONNECT" on the terminal software.



```
[AT Command Mode]
AT-R

OK

DISCONNECT
```

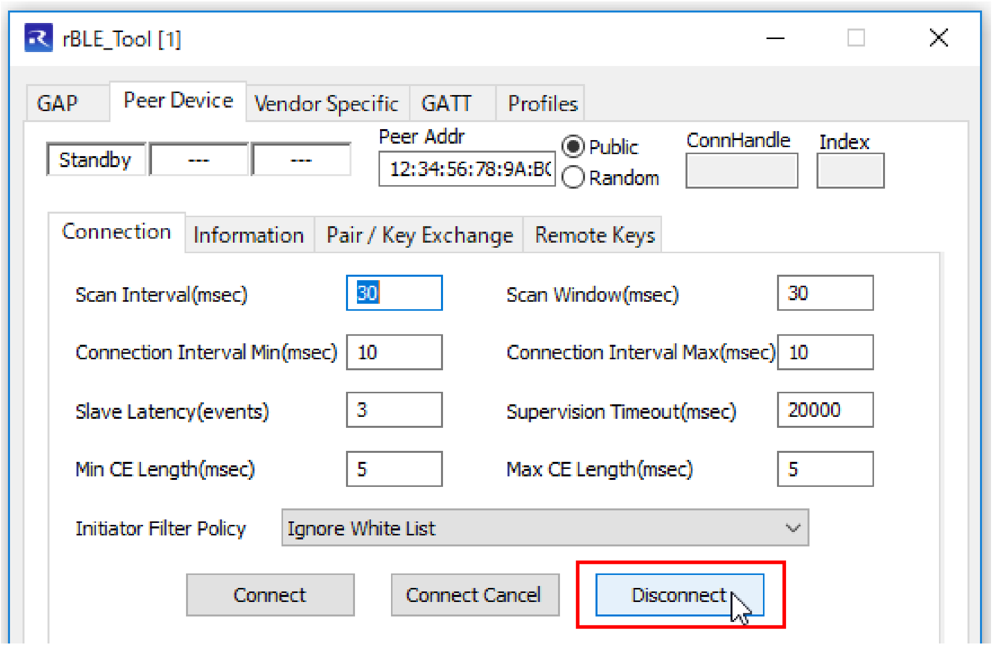
Figure 9-21: Disconnect from Virtual UART Server

- Disconnect from Virtual UART Client

  (1) Activate [Connection] tab of [Peer Device] tab.

  (2) Disconnect the established connection by pressing [Disconnect] button.

  (3) When the connection is terminated, the State display in top of [Peer Device] tab is changed to "Standby".



Standby --- ---
Peer Addr: 12:34:56:78:9A:B(
● Public ○ Random
ConnHandle:
Index:

Scan Interval(msec): 30
Scan Window(msec): 30
Connection Interval Min(msec): 10
Connection Interval Max(msec): 10
Slave Latency(events): 3
Supervision Timeout(msec): 20000
Min CE Length(msec): 5
Max CE Length(msec): 5
Initiator Filter Policy: Ignore White List

Figure 9-22: Disconnect from Virtual UART Client

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Feb 24, 2016	—	Initial issue
1.10	Oct 7, 2016	7	2.2 File Composition : Add the file composition for development environments and firmware.
		11	4 Simple AT Command Mode : Add commands AT-CI=<con_intv>, AT-CI?, ATE0, ATE1.
		22	7 Build and Operational Check : Add Environment setup and build procedure descriptions for development environments.
		26	7.4 Usage Example: Add the procedure to set a device address.
		37	8.3 Connection : Change Connection Interval Default setting value.
		49	8.7.2 Connection Sequence : Add the procedure for Connection Interval change.
		57	9.2 Operational Check by Using the GUI-Tool : Newly added.
1.11	Nov 22, 2016	-	Revision change (No document update).
1.12	Oct 20, 2017	24	7.2.3 BLE Protocol Stack / EEPROM : Update EEPROM emulation library download path.
1.13	Jul 12, 2019	-	Fix the AT-AS command. (No document update)
1.20	Oct 23, 2020	-	This is a revised version that supports sending and receiving binary data and non-response communication (write command, notification).
		5	1. Overview : Added an overview of sending and receiving binary data and non-response communication.
		7	2.2 File Composition : Added HEX file for sending and receiving binary data. Added macro file of Tera Term.
		9	3. Application Mode : Added function explanations for sending and receiving binary data and non-response communication.
		17	5. Virtual UART Mode : Added explanation of sending and receiving binary data and communication without response.
		21	6.2 Changes of Advertising Interval : Posted the code to change the advertising interval.
		22	7. Build and Operational Check : Updated build environment. Added build procedure and usage example for sending and receiving binary data and non-response communication.
		35	8. Implementation Details : Added profiles, functions, and events for sending and receiving binary data and non-response communication.
		56	9. Appendix : Updated ROM size and RAM size. Updated GUI tool operation method by supporting non-response communication.
1.20	Jan 31, 2022	-	Fixed due to the end of IAR support in Bluetooth Low Energy Protocol Stack.

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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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.