Bluetooth® Low Energy Protocol Stack

Case studies for good connectivity with smartphones

Introduction

This document provides good connectivity with smartphones when using "Bluetooth Low Energy Protocol Stack" (referred to as "BLE Software") used for developing Bluetooth application products using the Renesas Bluetooth Low Energy microprocessor RL78/G1D the corresponding case for doing it is described.

Target Device

RL78/G1D

Android device

Related Documents

The documents referred in this document may be preliminary version but might not marked as such version.

Document	Document No.
Bluetooth Low Energy Protocol Stack	—
User's Manual	R01UW0095E
API Reference Manual Basic	R01UW0088E
rBLE Command Specification	R01AN1376E
Quick Start	R01AN2767E

Contents

1. Overview ... 3
1.1 Equipment used in this document ... 3
1.2 Case list ... 3

2. Case that the connection cannot be established again by turning on, after turning off, the terminal device. ... 4
2.1 Outline ... 4
2.2 State explanation ... 5
2.3 Improvement plan ... 6
2.3.1 Improvement plan on terminal device side ... 6
2.3.2 Improvement plan for Android device side ... 8

3. The connection may not be established due to turning on the power of the terminal device (failure of the pairing sequence) ... 9
3.1 Outline ... 9
3.2 State explanation ... 10
3.2.1 HCI snoop log (Normal) ... 11
3.2.2 HCI snoop log (Symptom occurrence) ... 11
3.3 Improvement plan ... 12
3.4 Example of terminal device program implementation ... 13
3.4.1 Add message ID for delay ... 13
3.4.2 Added message processing for LTK response ... 13

4. Connection may not be established due to turning on the power of the terminal device (Feature exchange sequence failure) ... 15
4.1 Outline ... 15
4.2 State explanation ... 16
4.3 Improvement plan ... 18

5. Unable to maintain connection with Android device (packet reception failure on the terminal side) ... 19
5.1 Outline ... 19
5.2 State explanation ... 20
5.3 Improvement plan ... 20
5.4 Example of terminal device program implementation ... 21
5.4.1 Add Library Function/Variable References ... 21
5.4.2 Add interoperability improvement processing ... 21

6. Appendix ... 23
6.1 Analysis environment ... 23
6.1.1 Packet Sniffer log ... 24
6.1.2 Bluetooth HCI snoop log ... 24

1. Overview

1.1 Equipment used in this document

- ➢ Commercially available smartphones and tablets with Android OS

  (referred to as "Android device")

  - The android device will be our master device. It encrypts pairing and communication when making a connection request to the slave device.

- ➢ Equipment using RL78/G1D

  (referred to as "Terminal device")

  - The terminal device is a slave device. The terminal device automatically starts advertising when turning on the power. When connecting with an Android device, it performs pairing and communication encryption.

1.2 Case list

1. Case that the connection cannot be established again by turning on, after turning off, the terminal device.
If you turn on the power immediately after turning off the terminal, and make a connection request from the Android device from the state where the connection is established between the Android device and the terminal, the connection may not be established.
2. The connection may not be established due to turning on the power of the terminal device (failure of the pairing sequence)
When turning on the terminal and making a connection request from the Android device, the pairing sequence fails, and the connection cannot be established in some cases.
3. Connection may not be established due to turning on the power of the terminal device (Feature exchange sequence failure)
When turning on the terminal and issuing a connection request from the Android device, the Feature exchange sequence may fail, and the connection cannot be established in some cases.
4. Unable to maintain connection with Android device (packet reception failure on the terminal side)
When connecting with an Android device, it disconnects without pairing or GATT communication.

2. Case that the connection cannot be established again by turning on, after turning off, the terminal device.

2.1 Outline

- Phenomenon

If you turn on the power immediately after turning off the terminal device, and then make a connection request from the Android device from the state where the connection is established between the Android device and the terminal device, the connection may not be established.

- Assumed cause

Since a disconnection request is not issued from the terminal device when the terminal device is powered off, the Android device cannot communicate with the terminal device and waits for timeout (supervision timeout).

When the terminal device is turned on again, advertising is started, and the Android device tries to establish a new connection. But a connection waiting for the supervision timeout before the power is turned off remains for the terminal device, it is presumed to be caused by the unexpected state of "Dual Connection".

- Measures

Send disconnect command to Android device before turning off terminal device. Make sure that you cannot make a connection request to the same terminal device until disconnection is notified by the Android device application.

- Symptom confirmation device

Some Android devices with Android 7.1.1

2.2 State explanation

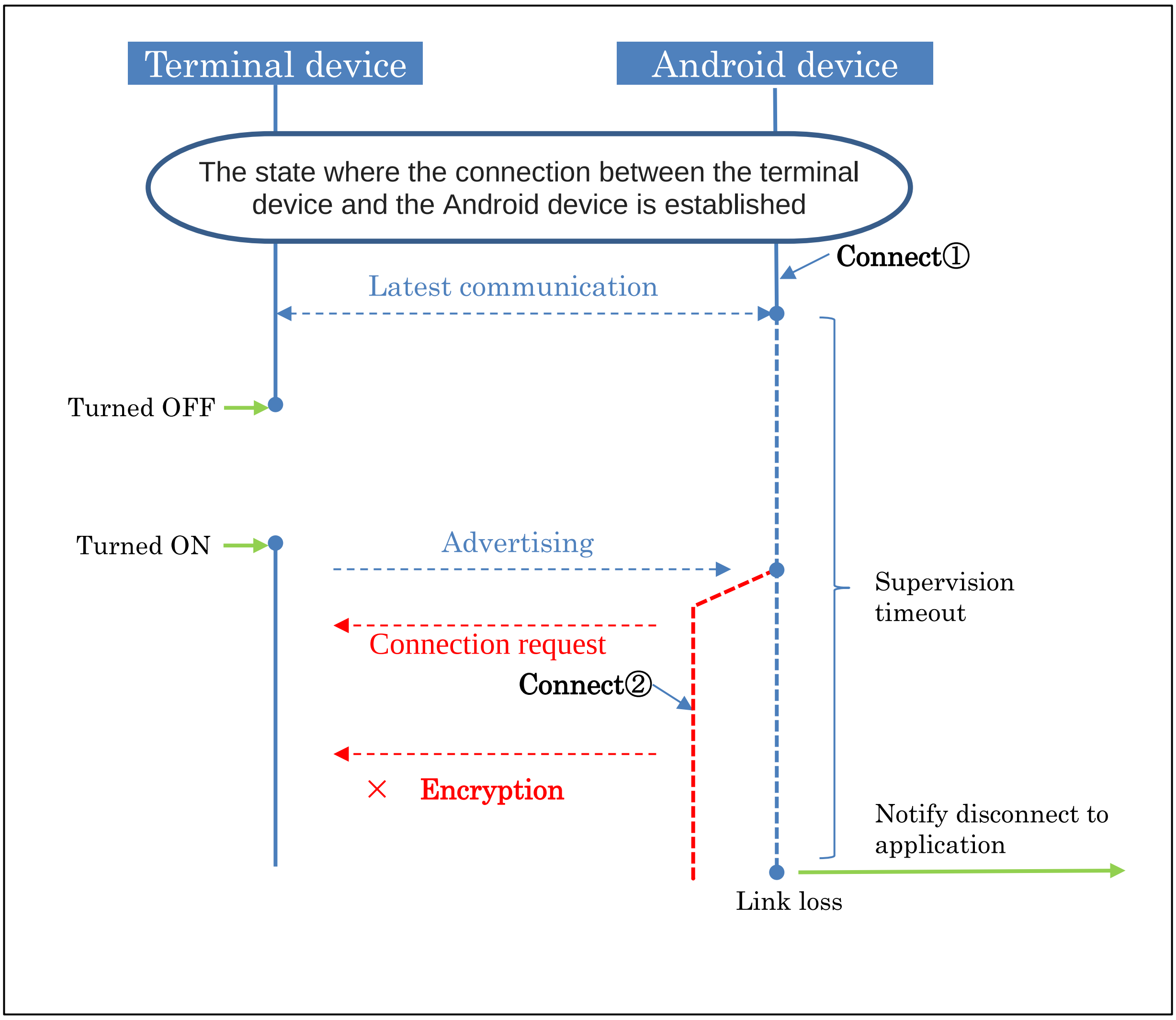


Figure 2-1 State when the terminal device cannot be reconnected from power OFF to ON

- When pairing is completed between the terminal device and the Android device, and the terminal device is turned off "Connection ①" of the Android device is supervision timeout (20 seconds for some Android devices with Android 7.1.1.).

- When the terminal device is powered on while in the above state, the Android device tries to establish a connection by issuing a connection request of "Connection ②" to the advertisement from the terminal device. Now the connection of the android device is doubly connected to the terminal device due to the presence, encryption is not started, and an abnormal connection state is established.

2.3 Improvement plan

2.3.1 Improvement plan on terminal device side

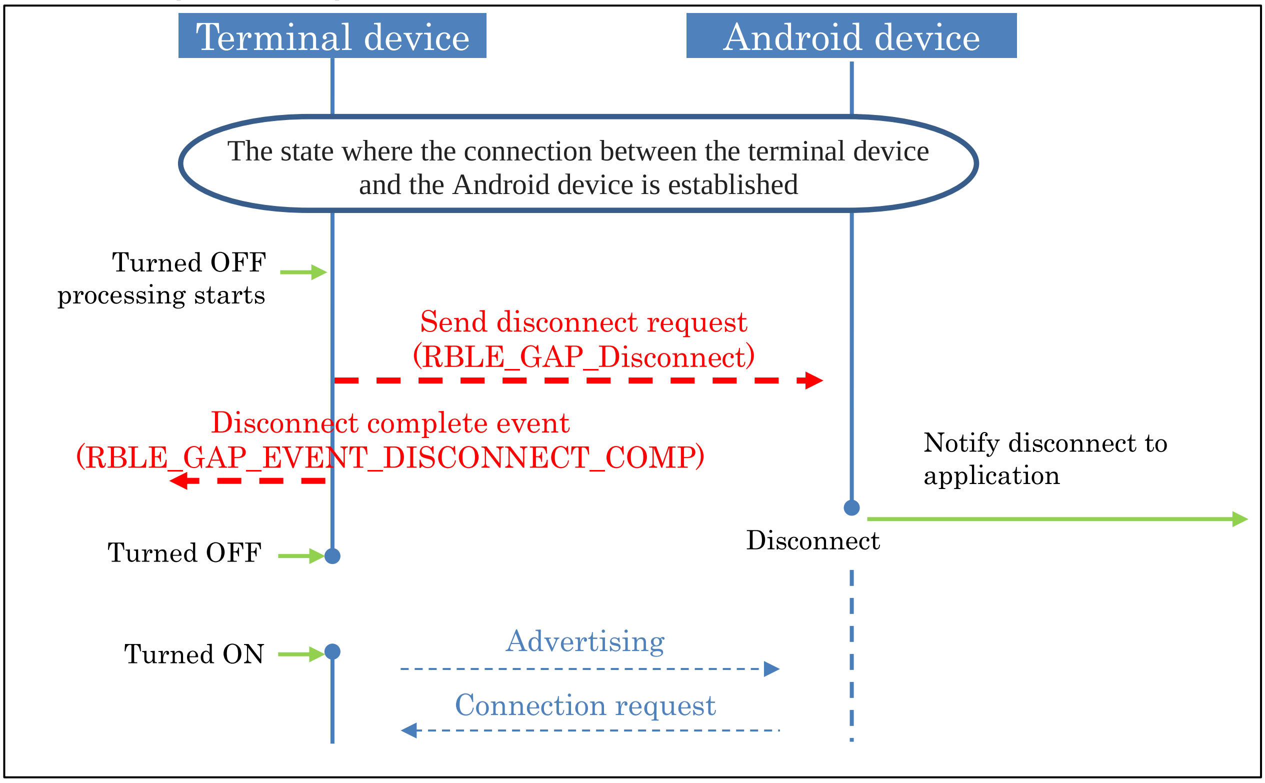


Figure 2-2 Improvement plan for the case where the terminal device cannot be reconnected from the power OFF to ON (Terminal device side: No.1)

- Before turning off the power of the terminal device, execute a disconnection command (RBLE_GAP_Disconnect) to send a disconnection to the Android device. The terminal device waits for a disconnection completion event (RBLE_GAP_EVENT_DISCONNECT_COMP) and turns off the power. By explicitly disconnecting the connection with the Android device, it is possible to normally connect to the advertising by turning on the power of the terminal device.

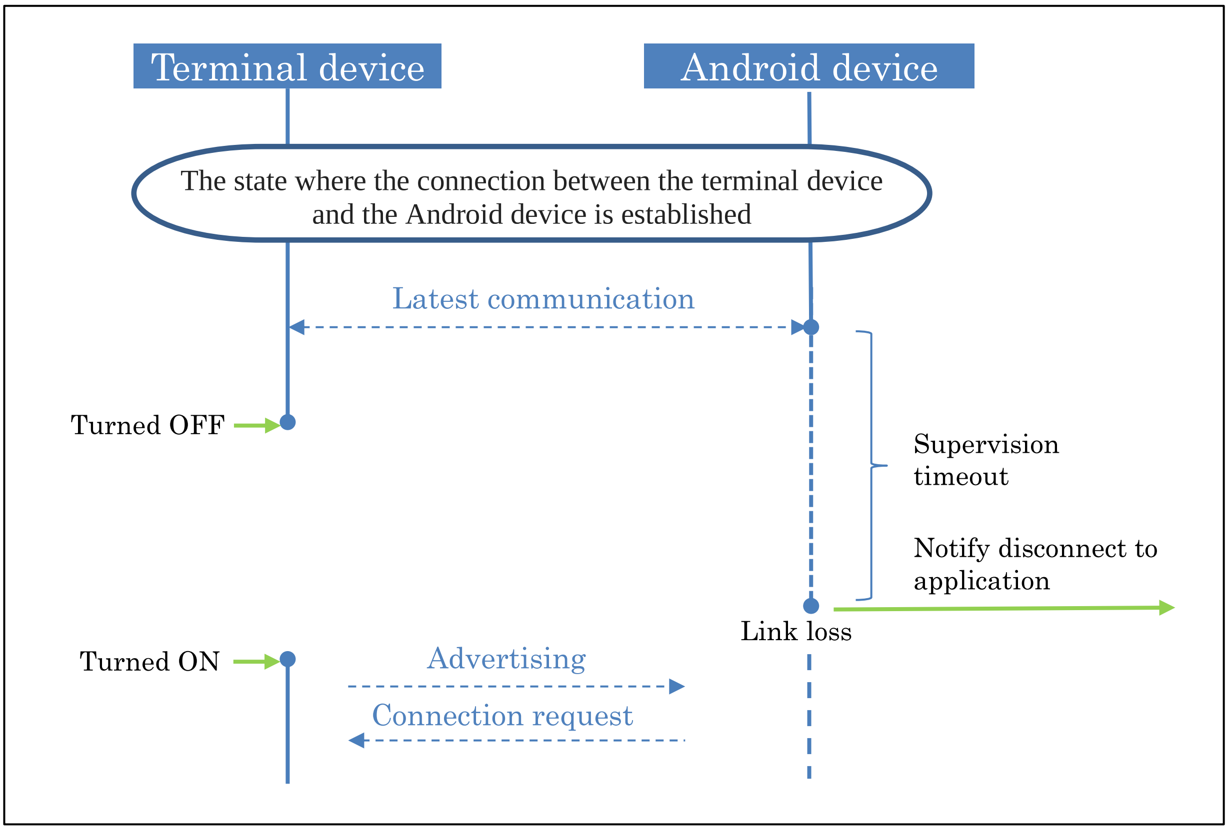


Figure 2-3 Improvement plan for the case where the terminal device cannot be reconnected from the power OFF to ON (Terminal device side: No.2)

- Turn on the power of the terminal device after the time of supervision timeout elapses. Since the connection between the Android device and the terminal device has been disconnected, the Android device issues a connection request to the advertisement from the terminal device, so that the connection is established normally.

2.3.2 Improvement plan for Android device side

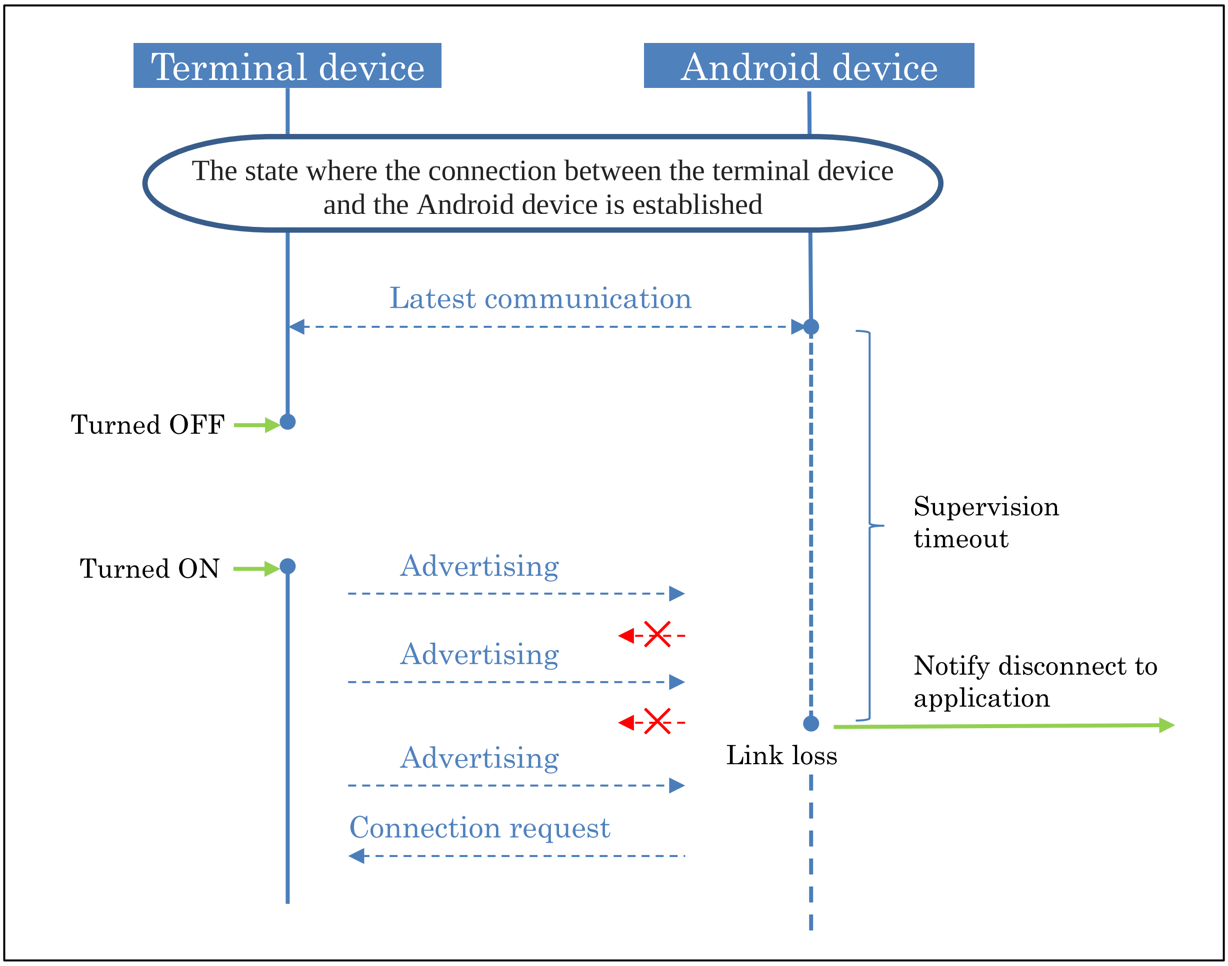


Figure 2-4 Improvement plan for the case where the terminal device cannot be reconnected from the power OFF to ON (Android device side)

- Avoid an abnormal connection state caused by the Android device not issuing a connection request for advertising from the terminal device received during the period of judging that the connection with the terminal device is continuing (until the supervision timeout expires).

Note: If possible, at the time of receiving advertisement from the connected terminal device, judge the link loss, issue a disconnection request from the application to the controller, shift to the disconnected state, and issue a connection request to the terminal device.

3. The connection may not be established due to turning on the power of the terminal device (failure of the pairing sequence)

3.1 Outline

- Phenomenon

When turning on the power of the terminal device and making a connection request from the Android device, the pairing sequence may fail, and connection may not be established in some cases.

- Assumed cause

In the pairing sequence of the Android device, the Encryption Information and Master Identification are notified from the controller layer of the Android device to the host layer prior to the Encryption Change event, whereby inconsistency occurs in the order of the pairing sequence and processing stops.

- Measures

Delay the execution of the function RBLE_SM_Ltk_Req_Resp responding to the LTK request with the RL78 / G1D program so that the Encryption Information and Master Identification will be notified after the Encryption Change event on the Android device.

Note: For delay time, "connection interval × 2" is recommended.

- Symptom confirmation device

Some Android devices with Android 7.0

3.2 State explanation

- Symptom occurrence status

A phenomenon in which connection cannot be established, and an error message with content saying "Cannot connect to Bluetooth" is displayed on the application of the smartphone.

- Analysis method

Analyze the HCI snoop log, which is the communication log in the Android device, using the Android device and the terminal device where the symptoms occur.

- Analysis result

When an error message of symptom occurrence status is displayed on the Android device, the pairing sequence after the connection between the terminal and the Android device is stopped. After the terminal transmits Master Identification, since the Android device does not transmit Identify Information, the pairing sequence is not completed, and the terminal and the Android device remain in an incomplete connection state. This is because the controller layer of the Android device notifies the host layer of Encryption Information and Master Identification prior to the Encryption Change event, thereby causing a discrepancy in the order of the pairing sequence and halting the processing.

3.2.1 HCI snoop log (Normal)

No.	Time	Source	Destination	Protocol	Length	Info
145	28.886628	host	controller	HCI_CMD	32	Sent LE Start Encryption
146	28.892502	controller	host	HCI_EVT	7	Rcvd Command Status (LE Start Encryption)
147	29.177797	controller	host	HCI_EVT	7	Rcvd Encryption Change
148	29.178117	RenesasE_00:7f…	localhost ()	SMP	26	Rcvd Encryption Information
149	29.178819	RenesasE_00:7f…	localhost ()	SMP	20	Rcvd Master Identification
150	29.180198	localhost ()	RenesasE_00:7f…	SMP	26	Sent Identity Information
151	29.180302	host	controller	HCI_CMD	43	Sent LE Add Device to Resolving List
152	29.180370	localhost ()	RenesasE_00:7f…	SMP	17	Sent Identity Address Information
153	29.180429	localhost ()	RenesasE_00:7f…	ATT	16	Sent Read By Group Type Request, GATT Pri…

①
②

[Source / Destination]
host : Top driver of smartphone
localhost : Top driver of smartphone
controller : Subordinate driver of smartphone
RenesasE_00 : RL78/G1D

Figure 3-1 HCI snoop log (Normal)

① When an Encryption Change event occurs immediately after the LE Start Encryption event from the controller layer in the Android device, Identity Information is transmitted from the host layer.

② Encryption information and Master Identification transmitted from the terminal device are notified from the controller layer in the Android device. Identity Information is transmitted from the host layer, after that, the pairing sequence is completed.

3.2.2 HCI snoop log (Symptom occurrence)

No.	Time	Source	Destination	Protocol	Length	Info
168	29.038978	host	controller	HCI_CMD	32	Sent LE Start Encryption
169	29.049877	controller	host	HCI_EVT	7	Rcvd Command Status (LE Start Encryption)
170	29.331540	RenesasE_00:7f…	localhost ()	SMP	26	Rcvd Encryption Information
171	29.331783	RenesasE_00:7f…	localhost ()	SMP	20	Rcvd Master Identification
172	29.331918	controller	host	HCI_EVT	7	Rcvd Encryption Change
173	55.516396	remote ()	localhost ()	L2CAP	488	Rcvd Connection oriented channel

①

[Source / Destination]
host : Top driver of smartphone
localhost : Top driver of smartphone
controller : Subordinate driver of smartphone
RenesasE_00 : RL78/G1D

Figure 3-2 HCI snoop log (Symptom occurrence)

① The Encryption Information and Master Identification transmitted from the terminal device are notified from the controller layer of the Android device before the Encryption Change event. When this state occurs, since the host layer of the Android device does not transmit the Identify Information, the pairing sequence is not completed, and the incomplete connection is maintained.

Note: When the occurrence of the Encryption Change event is delayed after the LE Start Encryption event from the controller layer in the Android device, the Identify Information is not transmitted from the host layer.

3.3 Improvement plan

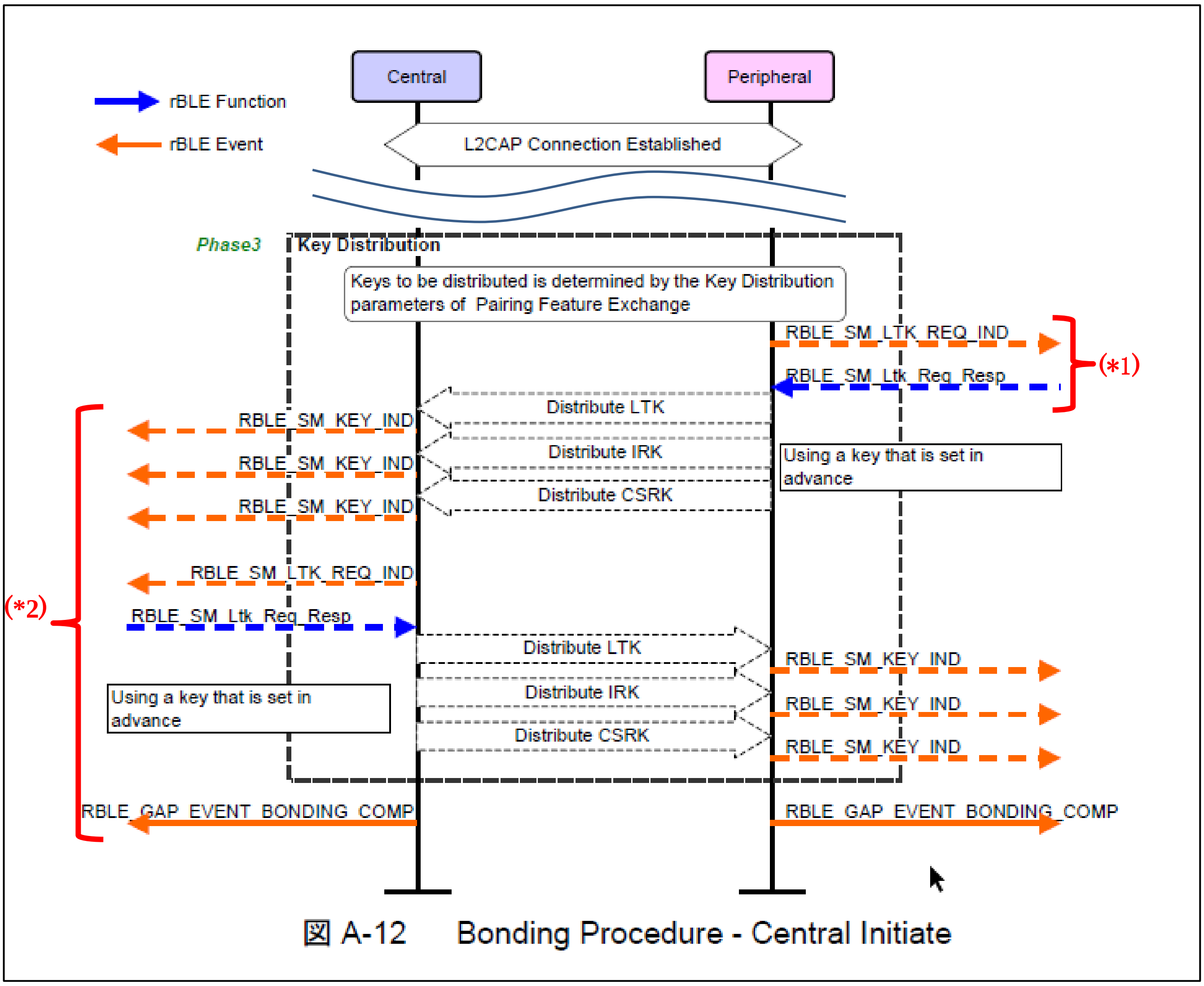


Figure 3-3 Improvement plan for cases where connection cannot be established by turning on terminal device (Pairing sequence failure)

- In pairing sequence processing of the terminal device program, wait insertion (**(*1)** in Figure 3-3) is performed while calling the RBLE_SM_Ltk_Req_Resp function since the occurrence of the RBLE_SM_LTK_REQ_IND event, thereby intentionally sending Encryption Information and Master Identification It can be delayed.

  Thus, after an Encryption Change of the HCI event occurs, in the Android device, Encryption Information and Master Identification from the terminal device can be received, and a normal pairing sequence can be executed.

- On the smartphone side which is the Central side, in the case of Android, the OS automatically performs the processing of **(*2)** in Figure 3-3, so there is no processing done on the application side.

3.4 Example of terminal device program implementation

An implementation example for delaying the invocation of the RBLE_SM_Ltk_Req_Resp function using the kernel timer is shown below.

3.4.1 Add message ID for delay

Adds the delay message ID to the enum enumeration of the message ID of the kernel used in the terminal device program.

Note: In our sample program, there is a definition in “r_ble_sample_app_peripheral.h”.

```
typedef enum {
    APP_MSG_BOOTUP = KE_FIRST_MSG(APP_TASK_ID) + 1,
    APP_MSG_RESET_COMP,
    APP_MSG_SECLIB_SET_PARAM_COMP,
    APP_MSG_CONNECTED,
    APP_MSG_SECLIB_CHK_ADDR_COMP,
    APP_MSG_SECLIB_PASSKEY_IND,
    APP_MSG_SECLIB_ENC_COMP,
    APP_MSG_DISCONNECTED,
    APP_MSG_PROFILE_ENABLED,
    APP_MSG_PROFILE_DISABLED,
    APP_MSG_TIMER_EXPIRED,
    APP_MSG_LTK_REQ_DELAY,   // ①Message ID for delay
} APP_MSG_ID;
```

Figure 3-4 Add message ID for delay

3.4.2 Added message processing for LTK response

When the RBLE_SM_LTK_REQ_IND event occurs, add a message function process to set the kernel timer and set the wait time and perform the LTK response.

Note: In our sample program, add message function processing to “r_ble_sample_app_peripheral.c”.

```
// ②Added Prototype Declaration of Message Function for LTK Response
static int_t app_ltk_req_delay(ke_msg_id_t const msgid, void const *param,
                            ke_task_id_t const dest_id, ke_task_id_t const
src_id);
```

Figure 3-5 Added Prototype Declaration of Message Function for LTK Response

```
const struct ke_msg_handler app_connect_handler[] = {
    { APP_MSG_CONNECTED,        (ke_msg_func_t)app_profile_enable },
    /* { APP_MSG_PROFILE_ENABLED, (ke_msg_func_t)NULL }, */
    { APP_MSG_TIMER_EXPIRED,    (ke_msg_func_t)app_timer_expired },
    // ③Register message function after elapse of timer time
    { APP_MSG_LTK_REQ_DELAY,    (ke_msg_func_t)app_ltk_req_delay },
};
```

Figure 3-6 Add message function after connection of timer time to connected message handler

```
/* ##### SM Event Handler ##### */
void app_sm_callback(RBLE_SM_EVENT *event)
{
    switch (event->type) {
    case RBLE_SM_LTK_REQ_IND:
        req_result = event->param.ltk_req;

        // ④Set the kernel timer for the delay time wait(unit time is 10 msec)
        //   -> Delay the response of LTK (Long Term Key)
        ke_timer_set(APP_MSG_LTK_REQ_DELAY, APP_TASK_ID, 50);//Wait 500 msec
        break;

    default:
        break;
    }
}
```

Figure 3-7 Added kernel timer setting process when RBLE_SM_LTK_REQ_IND event occurs

```
// ⑤Additional message function for LTK response → Execute after delay time
static int_t app_ltk_req_delay(ke_msg_id_t const msgid, void const *param,
                               ke_task_id_t const dest_id, ke_task_id_t const
src_id)
{
    /* Generate LTK/EDIV/NB. */
    seclib_generate_key(&ld_ltk.val);
    seclib_generate_nb(&ld_ltk.nb);
    ld_ltk.ediv  = SecLib_Rand();
    ld_ltk.valid = SECDB_VALID_KEY;

    /* LE Long Term Key Request Reply */
    RBLE_SM_Ltk_Req_Resp(req_result.idx, RBLE_OK,
                         RBLE_SMP_KSEC_NONE,
                         ld_ltk.ediv,
                         &ld_ltk.nb,
                         &ld_ltk.val);

    return KE_MSG_CONSUMED;
}
```

Figure 3-8 Added message function for LTK response

4. Connection may not be established due to turning on the power of the terminal device (Feature exchange sequence failure)

4.1 Outline

- Phenomenon

When turning on the power of the terminal device and making a connection request from the Android device, the Feature exchange sequence may fail, and the connection cannot be established in some cases.

- Assumed cause

When another command or event is executed between the connection request command HCI_LE_Create_Connection executed from the host layer of the Android device and the connection completion event LE Connection Complete, the command after the command HCI_LE_Read_Remote_Used_Features after the command which reads the function supported by the remote device The sequence for establishing the connection is not executed.

- Measures

After executing the connection request in the Android device application, do not enter any other processing until receiving the connection completion event LE Connection Complete.

- Symptom confirmation device

Some Android devices with Android 5.0.1

4.2 State explanation

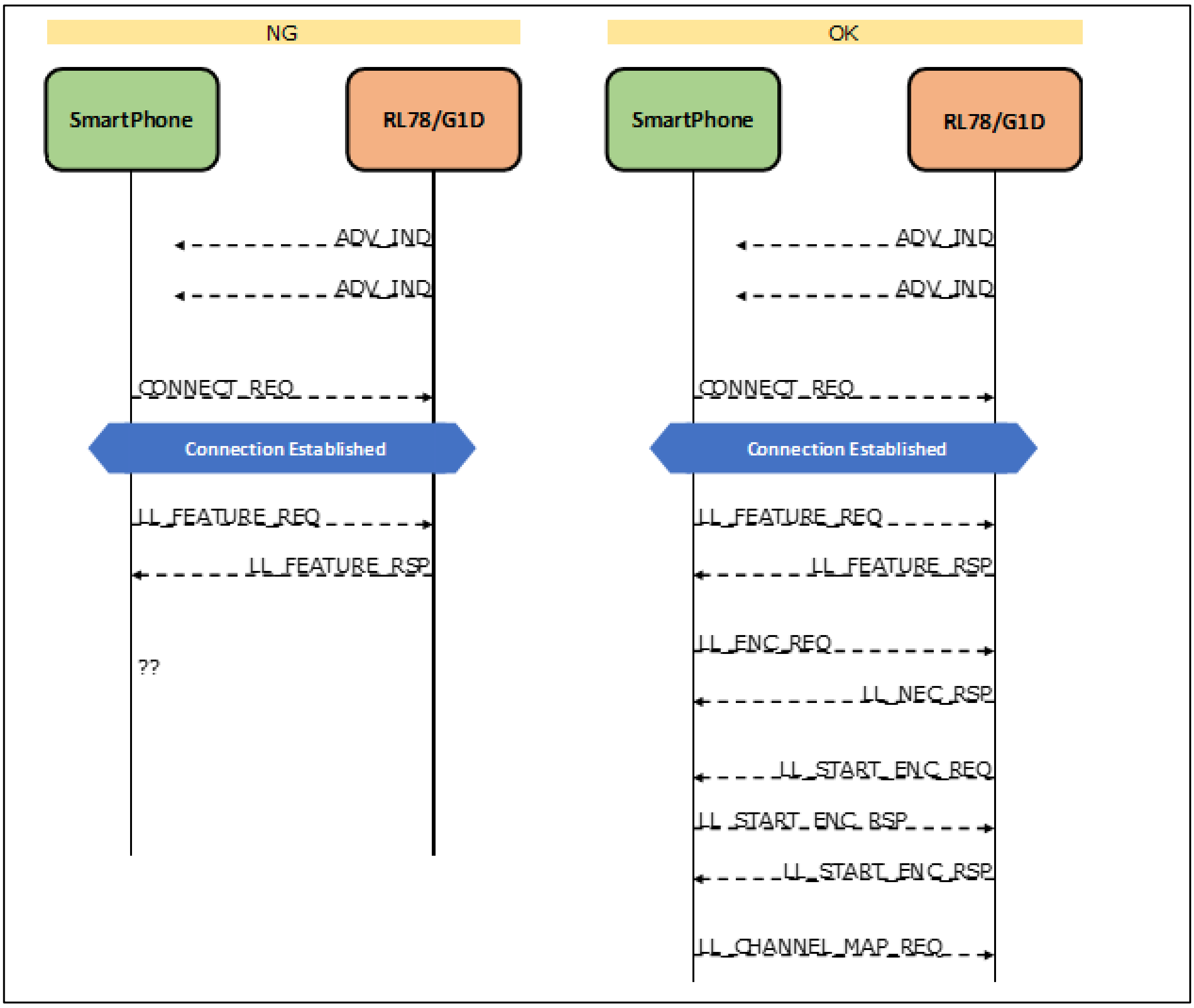


Figure 4-1 State when the terminal device cannot be connected due to power ON (Feature exchange sequence failure)

- Bluetooth Packet Sniffer log analysis result
  - In the case of NG, the Android device sends LL_FEATURE_REQ, and after the terminal device transmits LL_FEATURE_RSP, the Android device stops responding. Even in the case of NG, the Android device and the terminal device are connected. While maintaining the connection it will not proceed from the sequence of LL_FEATURE.
  - In case of OK, the Android device transmits LL_FEATURE_REQ, the sequence device is processed normally after the terminal device transmits LL_FEATURE_RSP.

Type	Opcode Command	Event
Command	HCI_LE_Create_Connection	
Event	HCI_LE_Create_Connection	Command Status
Command	HCI_LE_Set_Advertising_Parameters	
Event	HCI_LE_Set_Advertising_Parameters	Command Complete
Command	Write_Scan_Enable	
Event	Write_Scan_Enable	Command Complete
Command	Write_Scan_Enable	
Event	Write_Scan_Enable	Command Complete
Command	Write Extended Inquiry Response	
Event	Write Extended Inquiry Response	Command Complete
Command	Write Extended Inquiry Response	
Event	Write Extended Inquiry Response	Command Complete
Command	Write Extended Inquiry Response	
Event	Write Extended Inquiry Response	Command Complete
Command	Write Extended Inquiry Response	
Event	Write Extended Inquiry Response	Command Complete
Command	Write Extended Inquiry Response	
Event	Write Extended Inquiry Response	Command Complete
Command	Write_Class_of_Device	
Event	Write_Class_of_Device	Command Complete
Command	HCI_LE_Set_Advertising_Data	
Event	HCI_LE_Set_Advertising_Data	Command Complete
Command	HCI_LE_Set_Advertising_Parameters	
Event	HCI_LE_Set_Advertising_Parameters	Command Complete
Event		LE Connection Complete
Command	HCI_LE_Read_Remote_Used_Features	
Event	HCI_LE_Read_Remote_Used_Features	Command Status
Event		LE Read Remote Used Features Complete
Command	Change_Local_Name	
Event	Change_Local_Name	Command Complete
Command	Write Extended Inquiry Response	

252ms

Figure 4-2 HCI snoop log (Symptom occurrence)

Type	Opcode Command	Event
Command	HCI_LE_Create_Connection	
Event	HCI_LE_Create_Connection	Command Status
Event		LE Connection Complete
Command	HCI_LE_Read_Remote_Used_Features	
Event	HCI_LE_Read_Remote_Used_Features	Command Status
Event		LE Read Remote Used Features Complete
Command	HCI_LE_Start_Encryption	
Event	HCI_LE_Start_Encryption	Command Status
Event		Encryption Change

35ms

Figure 4-3 HCI snoop log (Normal)

- Analysis result of HCI snoop log
  - In the normal case, it is executed continuously from HCI_LE_Create_Connection to LE Connection Complete. Thereafter, commands of HCI_LE_Read_Remote_Used_Features and HCI_LE_Start_Encryption are executed.
  - When a symptom occurs, another command or event is executed during HCI_LE_Create_Connection ~ LE Connection Complete. After LE Connection Complete, the HCI_LE_Read_Remote_Used_Features command is executed, but the HCI_LE_Start_Encryption command is not executed.

4.3 Improvement plan

After executing the connection request in the Android device application, do not enter any other processing until receiving the connection completion event LE Connection Complete.

5. Unable to maintain connection with Android device (packet reception failure on the terminal side)

5.1 Outline

- Phenomenon

When connecting with an Android device, it disconnects without pairing or GATT communication.

- Assumed cause

The cause is presumed to be that the terminal failed to receive packets for some Android devices and was unable to respond with a connection event, resulting in disconnection.

- Measures

Modify the program on the terminal side so that the corresponding packet can be received.

- Symptom confirmation device

Some Android devices with Android 11 or later.

5.2 State explanation

- Symptom occurrence status

When connecting a terminal and an Android device, an error message stating "An app is needed to use this device" is displayed on the smartphone application.

Note: The error message may differ depending on the Android device used.

- Analysis method

Use the Android device and terminal where the symptom occurs and set a break at the LE link disconnection completion event (*RBLE_GAP_EVENT_DISCONNECT_COMP*) notified to the *gap_call_back* function registered by the *RBLE_GAP_Reset* API while the terminal side is connected for debugging using the on-chip debugging emulator (e.g., E2 emulator Lite). Then, when the LE link disconnection completion event occurs, check the disconnection reason value of the event parameter.

Note: Depending on the optimization level of the compiler, it may not be possible to watch event parameters with variable names. In that case, add a *uint8_t* type global variable for debug, assign a disconnection reason value to the global variable, and confirm the global variable in the watch window on the debugger.

- Analysis result

If the symptom occurrence operation is repeated and the disconnection reason of the terminal side LE link disconnection completion event is R*BLE_CONN_FAILED_TO_BE_ES* (0x3E) or *RBLE_CON_TIMEOUT* (0x08), it is considered that the terminal has failed to receive Android device packets and has disconnected.

5.3 Improvement plan

Add processing to improve interoperability in the main function on the terminal side so that packets from Android devices that cause symptoms can be received.

5.4 Example of terminal device program implementation

An implementation example for improving interoperability is shown below.

5.4.1 Add Library Function/Variable References

Add code [1] to use library functions/variables for use in terminal programs.

Note: For our sample program, add the code to rf.h.

```
<Project_Source\renesas\src\driver\rf\rf.h>
:
#ifndef RF_H_
#define RF_H_
:

void rf_init(const uint16_t rf_flg);
// [1] Add external reference declarations for library functions/variables
void rf_renesas_reg_wr(uint16_t addr, uint16_t value);
extern bool sleep_data_save;

/// @}

#endif // RF_H_
```

Figure 5-1 Add library function/variable reference

5.4.2 Add interoperability improvement processing

Add processing codes [2] to [5] to improve interoperability to the main function and arch_main_ent function of the terminal program.

Note: For our sample program, add code to main.c and arch_main.c.

```
<Project_Source\renesas\src\arch\rl78\main.c>
:
_MAINCODE void main( void )
{
:
#ifdef USE_FW_UPDATE_PROFILE
    /* during FW update? */
    if( true == check_fw_update() ) {
:
        // Disable the BLE core
        rwble_disable();

        // Initialize RF
        rf_init(CFG_RF_INIT);

        // [2] Add interoperability improvement processing
        rf_renesas_reg_wr(0x11A4,0x0B3A);
        rf_renesas_reg_wr(0x11A6,0x3A3A);

        // Initialize BLE stack
        rwble_init(&public_addr, CFG_SCA);
:
    }
    else
#endif
    {
        /* call arch main */
        arch_main();
    }
}
```

Figure 5-2 Add code to main.c

```
<Project_Source\renesas\src\arch\rl78\arch_main.c>
:
void arch_main_ent(void)
{
   struct bd_addr public_addr;     /* Public Device Address */
   bool app_reg_set = false; // [3] Add setting flag for interoperability improvement


    /* Disable parity error resets */
    RPECTL = 0x80;
:
:
    // And loop forever
    for (;;)
    {
// [4] Add setting flag check and interoperability improvement process
        if( (sleep_data_save == false) && (app_reg_set == false) )
        {
            rf_renesas_reg_wr(0x11A4,0x0B3A);
            rf_renesas_reg_wr(0x11A6,0x3A3A);
            app_reg_set = true;
        }

        //LED activity
        led_blink();

        // schedule the BLE stack
        rwble_schedule();

        // Checks for sleep have to be done with interrupt disabled
        GLOBAL_INT_DISABLE();
        // Check if the processor clock can be gated
        if ((uint16_t)rwble_sleep() != false)
        {
            // check CPU can sleep
            if ((uint16_t)sleep_check_enable() != false)
            {
                #ifndef CONFIG_EMBEDDED
                /* Before CPU enters stop mode, this function must be called */
                if ((uint16_t)wakeup_ready() != false)
                #endif // #ifndef CONFIG_EMBEDDED
                {
                    // Wait for interrupt
                    WFI();

                    #ifndef CONFIG_EMBEDDED
                    /* After CPU is released stop mode, this function must be called immediately */
                    wakeup_finish();
                    #endif // #ifndef CONFIG_EMBEDDED
                }
            }
        }
// [5] Add setting flag update processing for interoperability improvement
        if(sleep_data_save != false)
        {
            app_reg_set = false;
        }
        // Checks for sleep have to be done with interrupt disabled
        GLOBAL_INT_RESTORE();

        sleep_load_data();
    }
}
```

Figure 5-3 Add code to arch_main.c

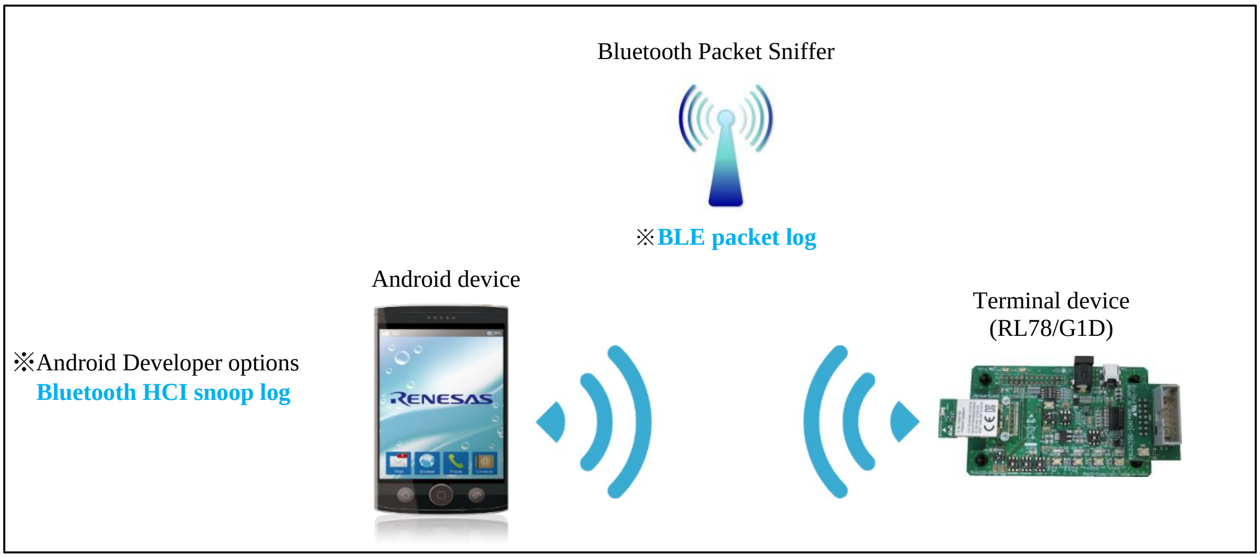
6. Appendix

The environment that can be used for analysis is shown below.

6.1 Analysis environment

There are two kinds of analysis environments as follows.

- Capture the communication between the terminal device (RL78/G1D) and Android device with Bluetooth Packet Sniffer and analyze with packet log
- Analyze BLE operation status of Android device with Bluetooth HCI snoop log



- **Figure 6-1 Analysis environment**

6.1.1 Packet Sniffer log

Capture communication on Air between devices and analyze logs.

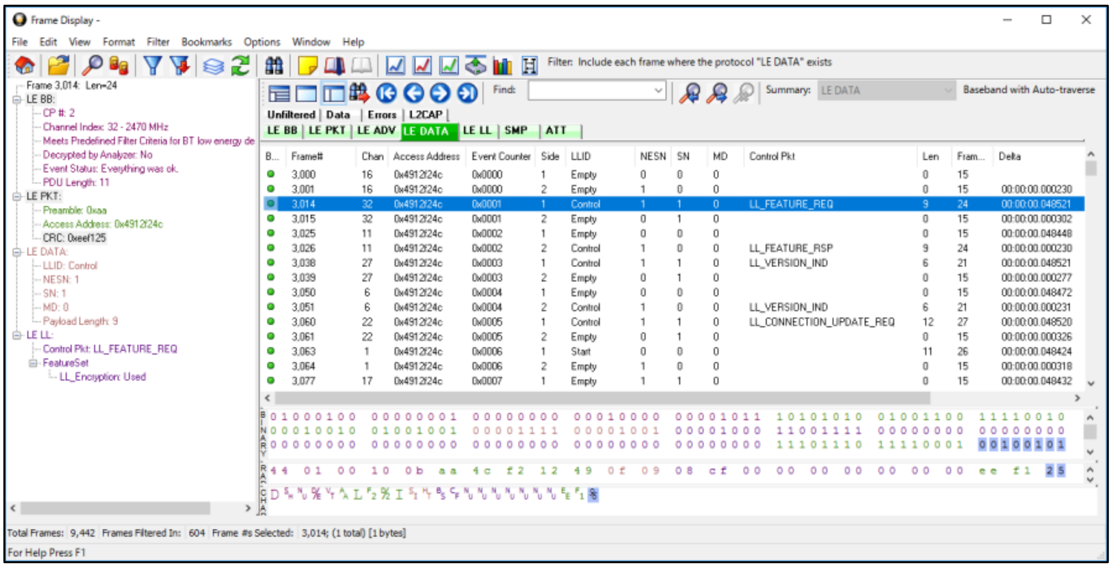


Figure 6-2 Packet Sniffer log

6.1.2 Bluetooth HCI snoop log

Record BLE HCI communication in Android smartphone and analyze log.

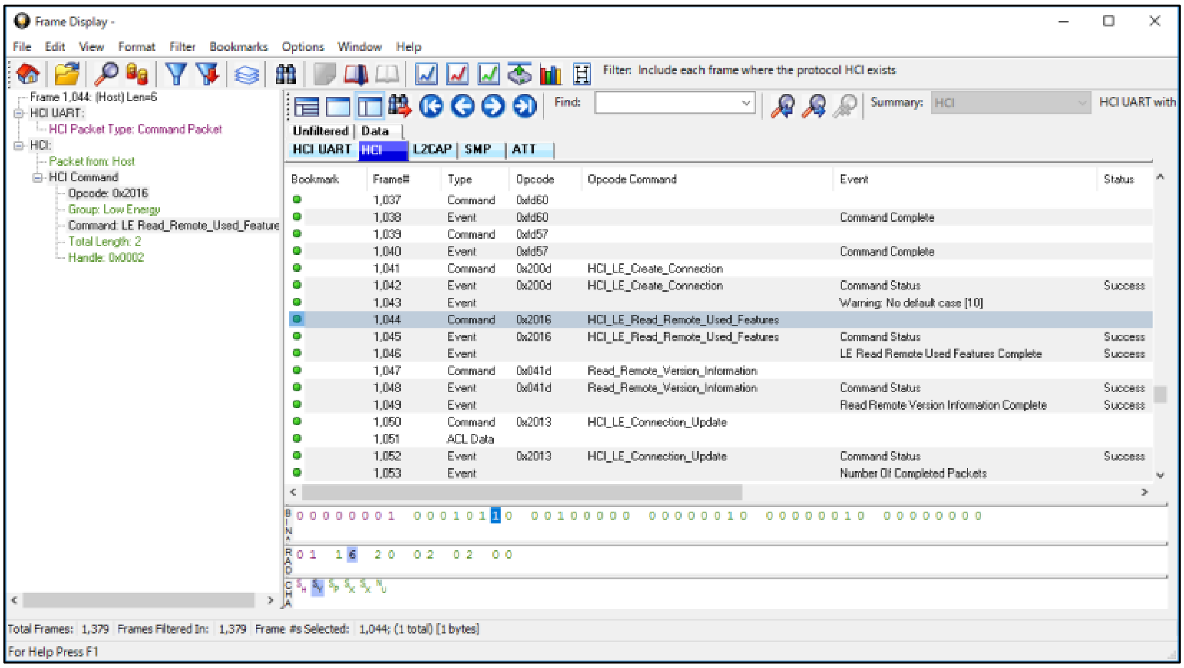


Figure 6-3 Bluetooth HCI snoop log

The recording method of Android's Bluetooth HCI snoop log is as follows.

1. Activate "Developer options" from "Settings" of Android smartphone and turn on "Enable Bluetooth HCI snoop log" setting.
2. When Bluetooth of Android smartphone is enabled, log recording is started.

   The log file name is "btsnoop_hci.log".

   Note: The save destination of the file depends on the model. For details, refer to the smartphone manual.

3. If you continue to record logs and the file size becomes large and it is difficult to see it, disable Bluetooth on smartphone once, disable "Developer options" and "Enable Bluetooth HCI snoop log" before starting recording, then again It is possible to reset and reset the log recording.

After acquiring the Bluetooth HCI snoop log, it can be viewed in the viewer.

- Reference viewer

- BPA software

  http://fte.com/support/CPAS-download.aspx?demo=BPA%20500&iid=1U

  After installation, use Capture File Viewer.

- Wireshark

  https://www.wireshark.org/

Once opened in the viewer, you can do the analysis in the following procedure.

1. Search by keywords such as "disconnection" or "response timeout". If found, the cause of the error is analyzed from the error occurrence point backward.
2. If an error occurrence location cannot be found by keyword, search for the point where connection was established with the keyword "create_connection". Check Command and Event one by one and analyze error occurrence points and cause.

53	Command	HCI_LE_Create_Connection			00:00:00.024035	2017/09/26 9:26:58.951985
54	Event	HCI_LE_Create_Connection	Command Status	Success	00:00:00.005141	2017/09/26 9:26:58.957126
55	Event		Warning: No default case [10]		00:00:00.550315	2017/09/26 9:26:59.507441
56	Command	HCI_LE_Read_Remote_Used_Features			00:00:00.000640	2017/09/26 9:26:59.508081
57	Event	HCI_LE_Read_Remote_Used_Features	Command Status	Success	00:00:00.002543	2017/09/26 9:26:59.510624
58	Event		LE Read Remote Used Features Complete	Success	00:00:00.109168	2017/09/26 9:26:59.619792
59	Command	Read_Remote_Version_Information			00:00:00.000234	2017/09/26 9:26:59.620026
60	Event	Read_Remote_Version_Information	Command Status	Success	00:00:00.004088	2017/09/26 9:26:59.624114
61	Event		Read Remote Version Information Complete	Success	00:00:00.093136	2017/09/26 9:26:59.717250
:	:	:	:	:	:	:
173	ACL Data				00:00:00.009306	2017/09/26 9:27:07.672764
174	Command	HCI_LE_Connection_Update			00:00:00.006043	2017/09/26 9:27:07.678807
175	Event	HCI_LE_Connection_Update	Command Status	Success	00:00:00.006914	2017/09/26 9:27:07.685721
176	Event		LE Connection Update Complete	LMP Response Timeout	00:00:00.550251	2017/09/26 9:27:08.235972
177	Event		Disconnection Complete	Success	00:00:00.000642	2017/09/26 9:27:08.236614
178	Command				00:02:00.298772	2017/09/26 9:29:08.535386

Figure 6-4 **Viewer display example**

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr 27, 2018	-	First edition issued
1.10	Dec 23, 2022	19	Added section “5.Unable to maintain connection with Android device (packet reception failure on the terminal side)”
1.20	Apr 7, 2023	21	Updated the setting values of processing codes [2] and [4] in “5.4.2 Add interoperability improvement processing” to ensure interoperability for iOS devices.

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.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.