RA4W1 Group

High Data throughput sample application

Introduction

This document describes the accompanying sample application which controls the Bluetooth® Low Energy communication module. In this document, the module which performs Bluetooth® Low Energy communication is referred to as the BLE module.

Target Device

RA4W1 Group

Related Documents

Bluetooth Core Specification (https://www.bluetooth.com)

RA4W1 Group User's Manual: Hardware (R01UH0883)

RA Flexible Software Package Documentation

e^2 studio Getting Started Guide (R20UT4204)

EK-RA4W1 – Quick Start Guide (R20QE0015)

EK-RA4W1 User's Manual (R20UT4683)

RA4W1 Group BLE sample application (R01AN5402)

RA4W1 Group Bluetooth Low Energy Profile Developer's Guide (R01AN5428)

Related Environments

Refer to section 2.1

Contents

1. Overview

Demo projects accompanying this document are shown in Table 1. These projects are provided as BLE sample application using BLE module.

Table 1. Demo Projects

Demo Project	Description
RA4W1_Throughput_Server	Server side of high data throughput demo project for EK-RA4W1.
RA4W1_Throughput_Client	Client side of high data throughput demo project for EK-RA4W1.

These projects demonstrate high data throughput BLE communication between one pair of EK-RA4W1 by using LE 2M PHY.

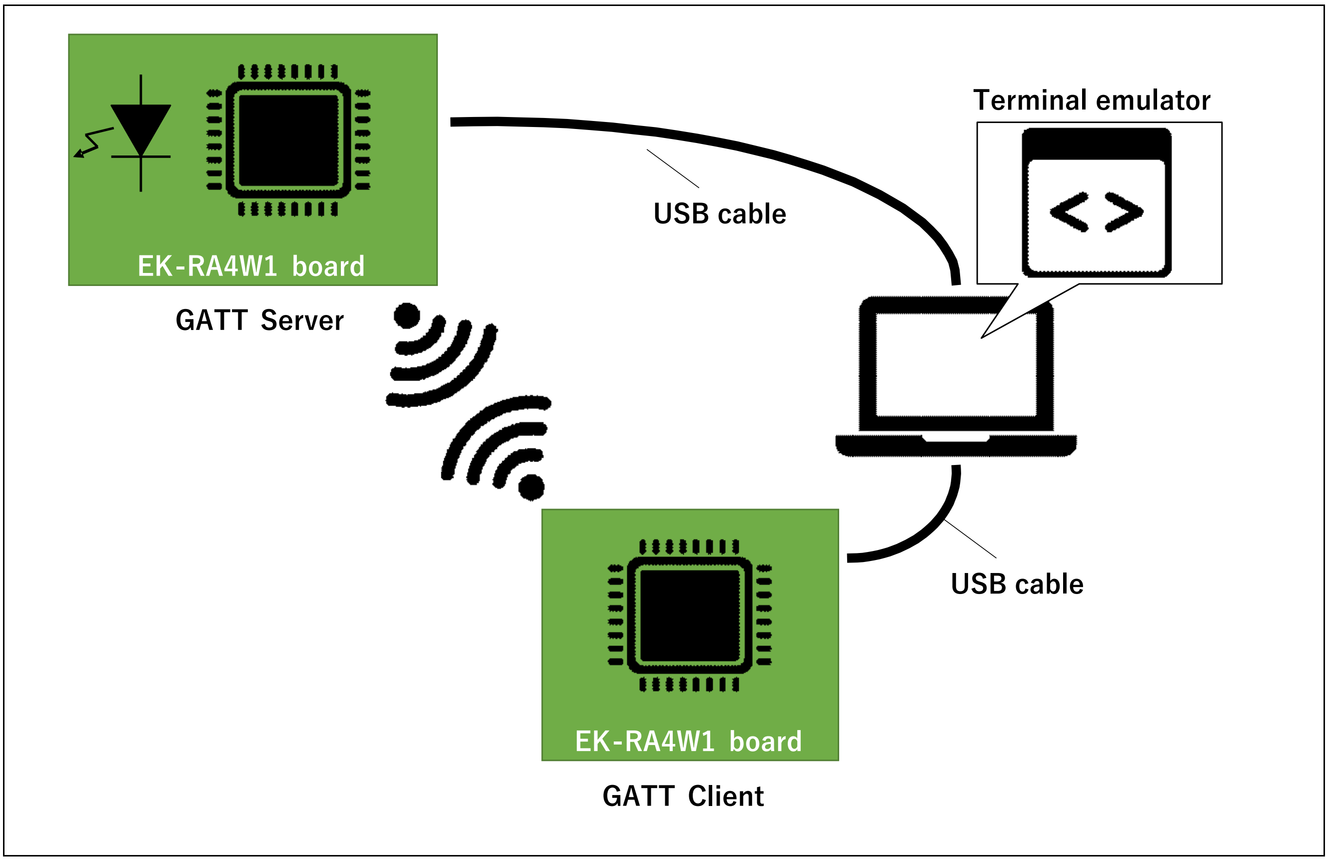

Figure 1. High data throughput demo project operating environment

1.1 BLE application software structure

This sample program has been created by using Flexible Software Package (FSP) BLE Abstraction Driver on rm_ble_abs (BLE Module) and QE for BLE. In general, it is necessary to design a profile for exchanging application data and user application itself when developing a Bluetooth Low Energy application. Profile and user application skeleton code can be generated by using QE for BLE. Figure 2 shows software structure of this sample application.

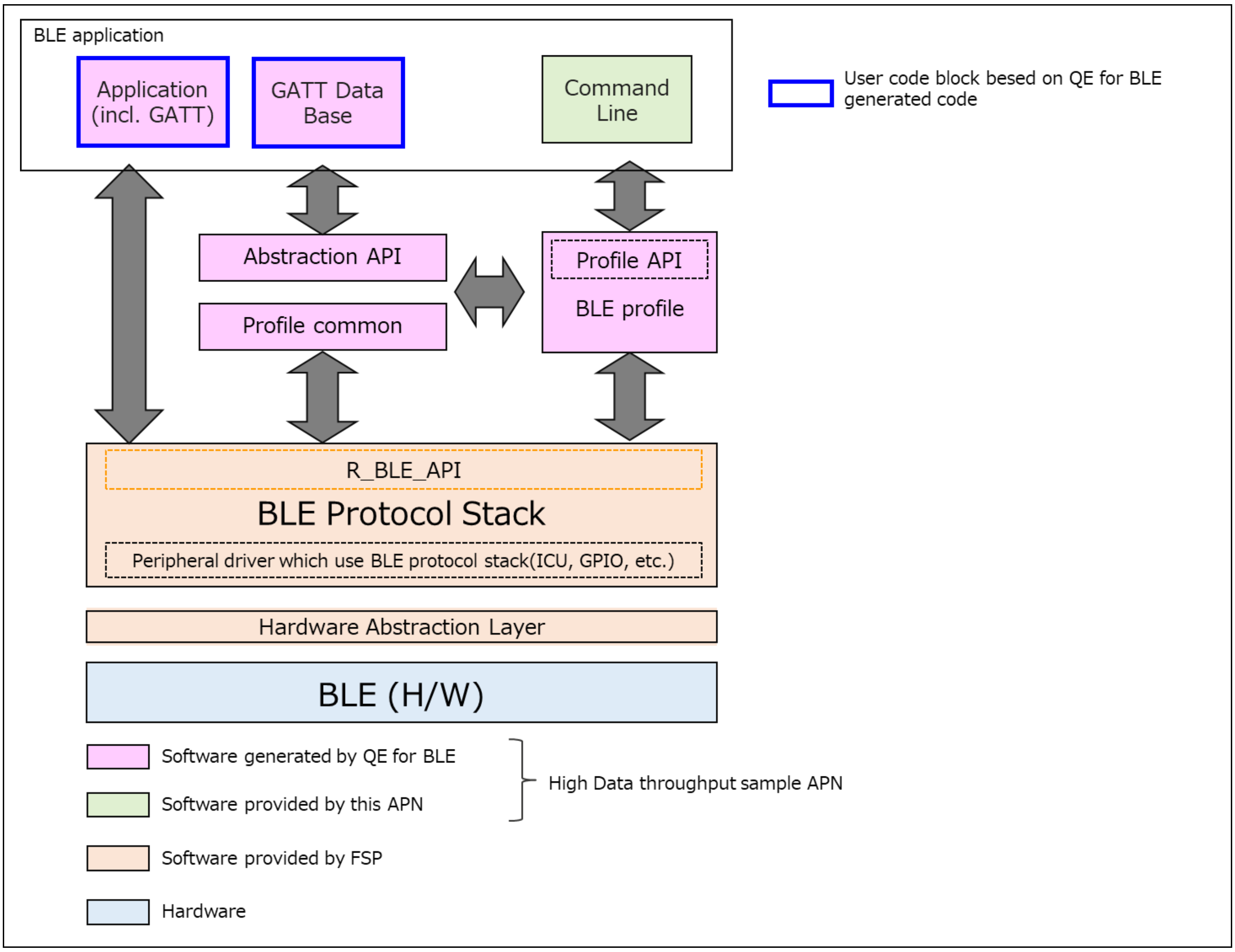

Figure 2. Software structure

2. How to use demo project

This chapter describes how to use demo project with this document.

2.1 Operating environment

Table 2 shows the hardware requirements for building and debugging BLE software.

Table 2. Hardware requirements

Hardware	Description
Host PC	Windows® 10 PC with USB interface.
MCU Board	The MCU used must support BLE functions. EK-RA4W1 [RTK7EKA4W1S00000BJ]
On-chip debugging emulators	The EK-RA4W1 has an on-board debugger (J-Link OB), therefore it is not necessary to prepare an emulator.
E2 lite emulator	Needed if user wants to write device-specific data (refer to *RA4W1 Group BLE sample application* section 4.2 (R01AN5402)) in custom board by using Renesas Flash Programmer.
USB cables	Used to connect to the MCU board. EK-RA4W1: 2 USB A-microB cable

Table 3 shows the software requirements for build and debug BLE software.

Table 3. Software requirements

Software		Version	Description
GCC environment	e^2 studio	2021-04	Integrated development environment (IDE) for Renesas devices.
	GCC ARM Embedded	V9	C/C++ Compiler. (download from e^2 studio installer)
	Renesas Flexible Software Package (FSP)	V3.0.0	Software package for making applications for the RA microcontroller series.
	QE for BLE[RA]	V1.2.0	Generates the source codes (BLE base skeleton program) as a base for the BLE Application and the BLE Profile. https://www.renesas.com/qe-ble
	SEGGER J-Flash Lite	V6.88	Tool for programming the on-chip flash memory of microcontrollers.
Header files			All API calls and their supporting interface definitions located in r_ble_api.h and rm_ble_abs_api.h.
Integer types			It uses ANSI C99 “Exact width integer types”. These types are defined in stdint.h.
Endian			Little endian
Terminan emulator			VT-100 compatible

2.2 Importing demo project

Refer to *RA4W1 Group BLE sample application* (R01AN5402) about how to import demo projects into e^{2}stidio. As a result of importation, following file structure will appear in e^2 studio project explorer.

Table 4. File structure about demo project

Directory/File structure				Description
qe_gen	ble	discovery		Service discovery related APIs
		profile_cmn		Profile common APIs
		app_main.c		Main code
		gatt_db.c		GATT Database
		gatt_db.h		GATT Database
		r_ble_thx.c		Profile APIs (x=s:server, x=c:client)
		r_ble_thx.h		Profile APIs (x=s:server, x=c:client)
ra	fsp	Inc	api	BLE API declaration r_ble_api.h rm_ble_abs_api.h
			instances	BLE module structure and macro definitions rm_ble_abs.h
		lib	r_ble	BLE Protocol Stack (All Features) static library
		src	rm_ble_abs	Abstraction API implementation rm_ble_abs.c
ra_gen	---	---		RA configuration generated.
ra_cfg	fsp_cfg	r_ble_cfg.h		Configuration option file
		rm_ble_abs_cfg.h		Configuration option file
src	---	hal_entry.c		User code entry point.
		r_ble_thc_cli.c		Throughput custom profile related command implementation (Client only)
		r_ble_thx_common.h		Throughput custom profile related common macros definitions (x=s:server, x=c:client)
	app_lib	cli		CLI functionality provided by this demo project
		cmd		Commands of CLI provided by this project
		logger		Logger functionality provided by this demo project

2.3 GATT Data Base

GATT data base included in gatt_db.c. Structure of GATT database is shown in following table. This demo project implements "Throughput" service as a custom service to achieve high speed communication. GATT data base in this sample program is generated by "QE for BLE". For detail of the custom service, refer to section 3.3.

Table 5. GATT Data Base

HANDLE	ATT Type	Properties	ATT Value	Definition
GAP Service				
0x0001	0x2800	Read	0x1800	GAP Service Declaration
0x0002	0x2803	Read	0x2A00	Device Name characteristic Declaration
0x0003	0x2A00	Read, Write	0x0000	Device Name characteristic value
0x0004	0x2803	Read	Property 0x02 Handle 0x0005 UUID 0x2A01	Appearance characteristic Declaration
0x0005	0x2A01	Read	0x0000	Appearance characteristic value
0x0006	0x2803	Read	Property 0x02 Handle 0x0007 UUID 0x2A04	Peripheral Preferred Connection Parameters characteristic Declaration
0x0007	0x2A04	Read	0x0000	Peripheral Preferred Connection Parameters characteristic value
0x0008	0x2803	Read	Property 0x02 Handle 0x0009 UUID 0x2AA6	Central Address Resolution characteristic Declaration
0x0009	0x2AA06	Read	0x0000	Central Address Resolution characteristic value
0x000A	0x2803	Read	Property 0x02 Handle 0x000B UUID 0x2AC9	Resolvable Private Address Only characteristic Declaration
0x000B	0x2AC9	Read	0x0000	Resolvable Private Address Only characteristic value
GATT Service				
0x000C	0x2800	Read	0x1801	GATT Service Declaration
0x000D	0x2803	Read	Property 0x20 Handle 0x0E UUID 0x2A05	Service Changed characteristic Declaration
0x000E	0x2A05	Indication	0x0000	Service Changed characteristic value
0x000F	0x2902	Read, Write	0x0000	Client Characteristic Configuration descriptor
Throughput Service				
0x0010	0x2800	Read	9CEF3D11-7FAB-49DC-AB89-762C9079FE96	Throughput Service Declaration

HANDLE	ATT Type	Properties	ATT Value	Definition
0x0011	0x2803	Read	Property 0x0C Handle 0x0012 UUID af6a66ad-7f11-4b77-b41d-9a8640f7a6a7	Throughput Data1 characteristic Declaration
0x0012	af6a66ad-7f11-4b77- b41d-9a8640f7a6a7	Write, Write without response	0x0000	Throughput Data1 characteristic value
0x0013	0x2803	Read	Property 0x30 Handle 0x14 UUID 6722005a-807a-43d4-9b94-1e6a08ea0478	Throughput Data2 characteristic Declaration
0x0014	6722005a-807a-43d4- 9b94-1e6a08ea0478	Notification Indication	0x00	Client Characteristic Configuration descriptor

2.4 Demo project behavior

2.4.1 Write demo project to the EK-RA4W1

This section describes how to program this demo project to EK-RA4W1.

1. Turn on the emulator terminal #2 of ESW1 on the EK board. And connect ECN1 of the EK board and your PC with a micro-B type USB cable.

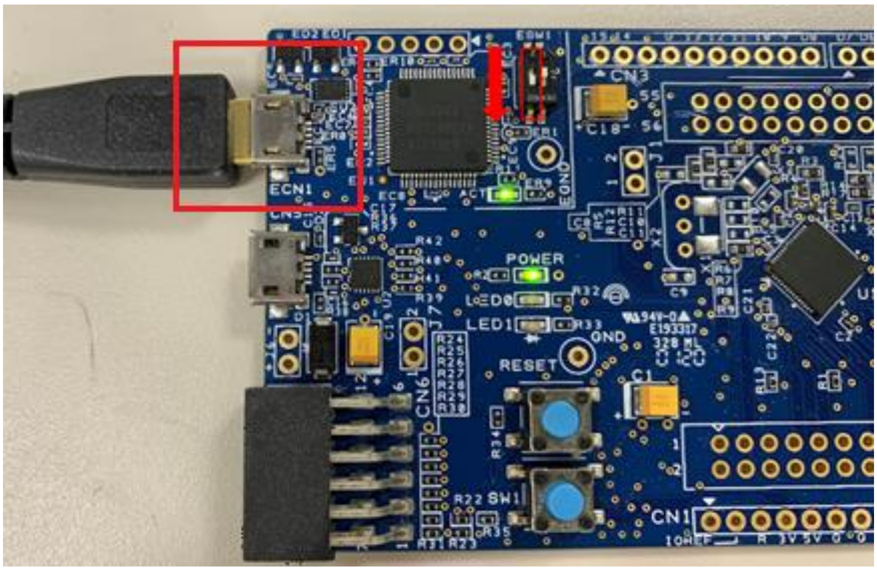

Figure 3. Connecting to the EK-board for program writing

2. Lunch J-Flash Light and click "**…**" button and select **R7FA4W1AD** as the device. Then click the **OK** button.

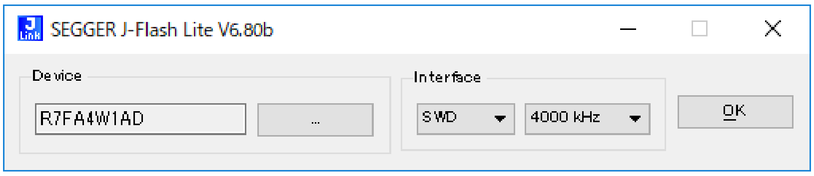

Figure 4. Select Device

3. Click "…" button and select **Throughput_Client.srec** for client or **Throughput_Server.srec** for server which included in the demo project. Then click on the **Program Device** button to write the program to the EK-RA4W1.

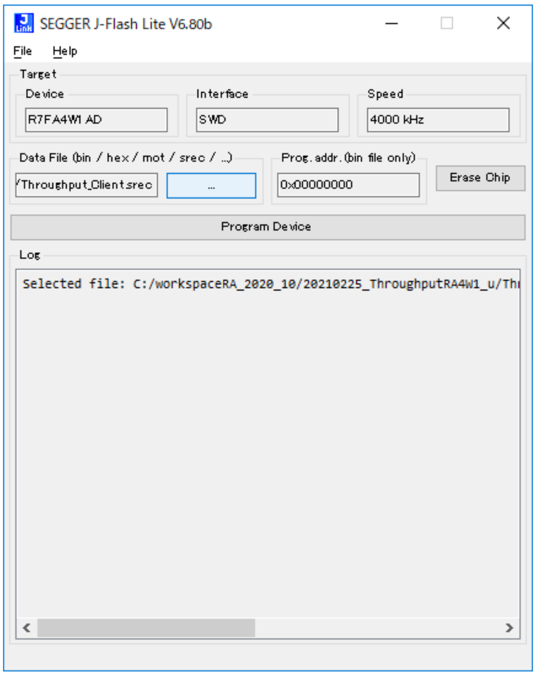

Figure 5. Select srec file

4. When the writing of the program is completed, "Done" will be displayed in the log.

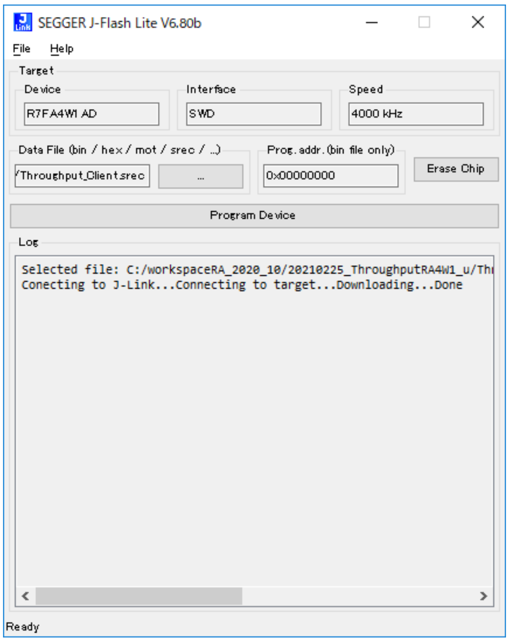

Figure 6. Program completed

5. Turn off the emulator switch (ESW1) on the EK-RA4W1.

2.4.2 How to work demo project

This section describes behavior of demo project.

1. Connect CN5 of the EK-RA4W1 and your PC with a micro-B type USB cable.

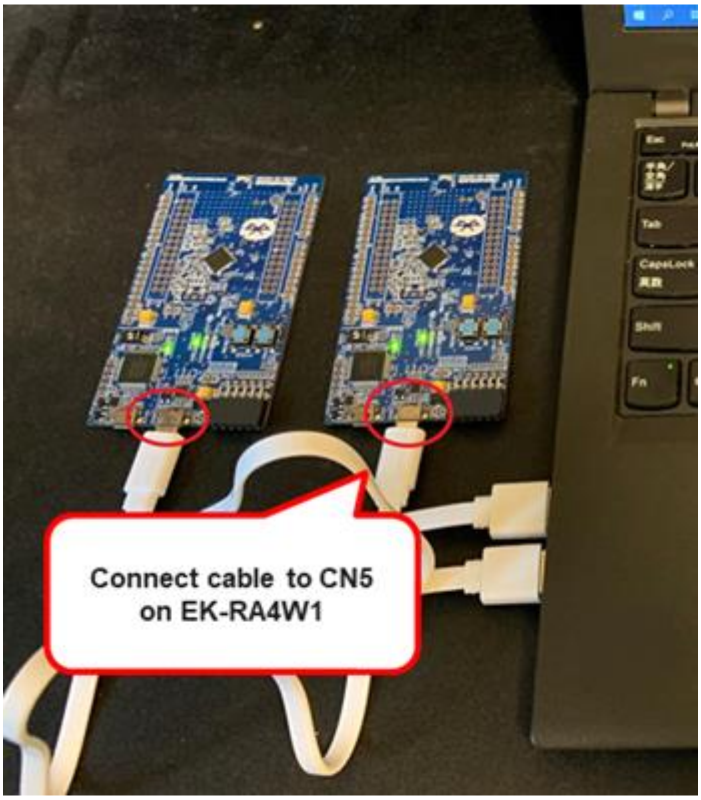

Figure 7. Demo project running

2. Launch VT-100 compatible terminal emulator with following configurations.

Table 6. Setting of the terminal software

Item	Value
New line (Receive)	LF
New line (Transmit)	CR
Terminal Mode	VT100
Baud rate	115200
Data bits	8bits
Parity	None
Stop bits	1bit
Flow control	None

3. Press “Reset” button on server and client side of EK-RA4W1.

4. Following characters will appear on terminal emulator connected with server and client side of EK-RA4W1(Figure 8)

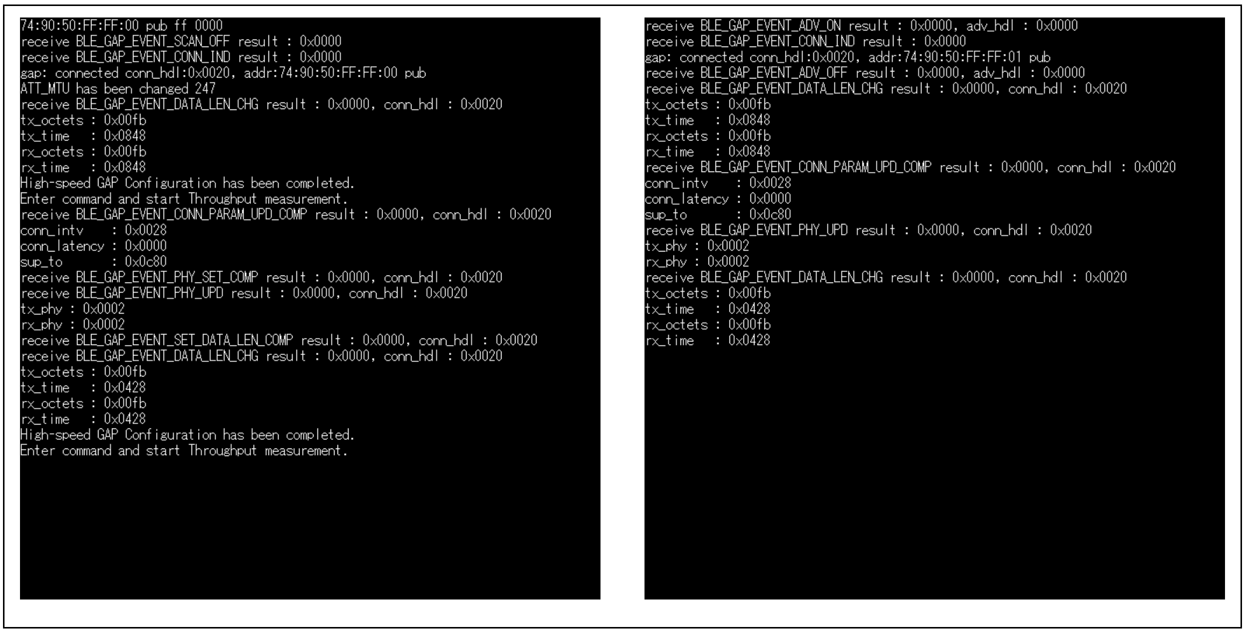

Figure 8. Console screens ready to start throughput measurement

(left : client, right : server)

5. Put “thc start notification” or “thc start indication” command on client side of EK-RA4W1. And then the measurement results will appear on client side (Figure 9)

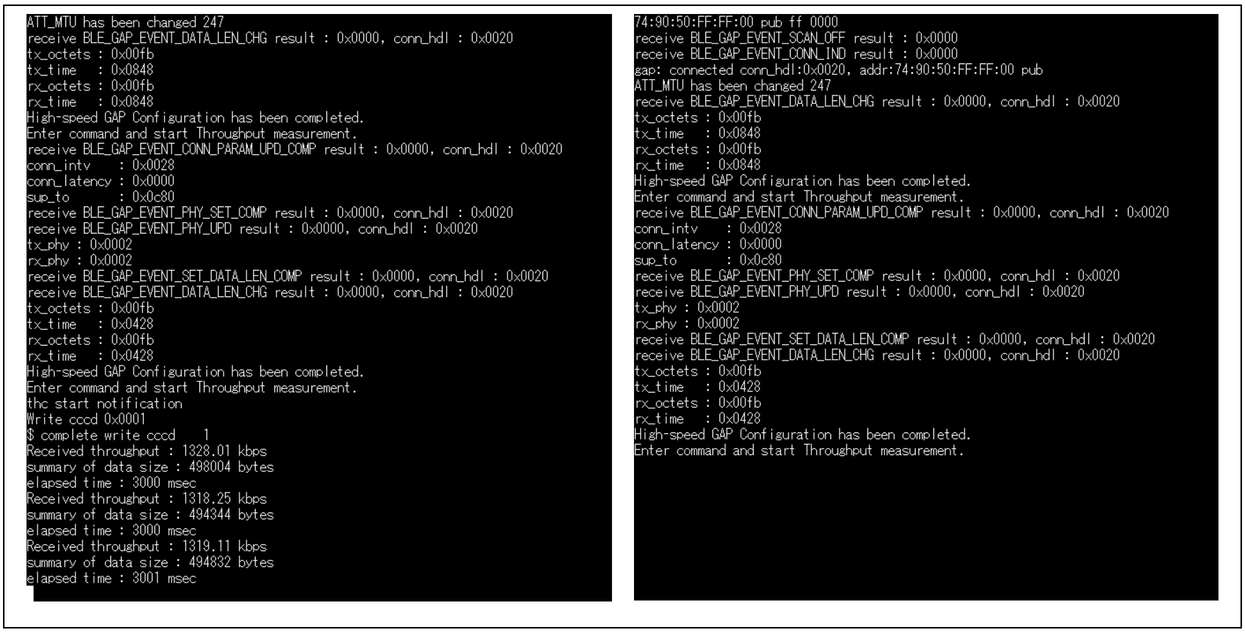

Figure 9. Console screens during throughput measurement

(left : client, right : server)

6. Put “thc start write_without_response” or “thc start write” command” on client side of EK-RA4W1. And then the measurement results will appear on server side (Figure 10).

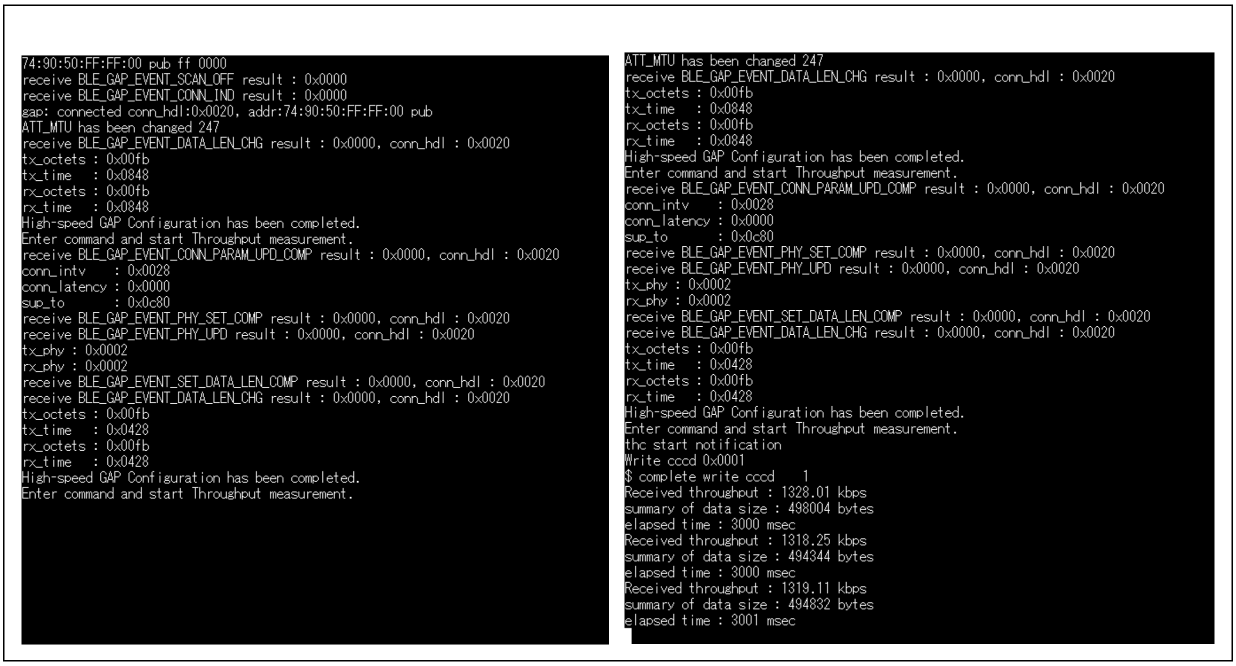

Figure 10. console screens during throughput measurement (left : client, right : server)

4. You can stop throughput measurement by using,

- Put “thc stop receive” command when you have measured throughput by using “thc start notification” or “thc start indication” command.
- Put “thc stop transmit” command when you have measured throughput by using “thc start write_without_response” or “thc start write” command.

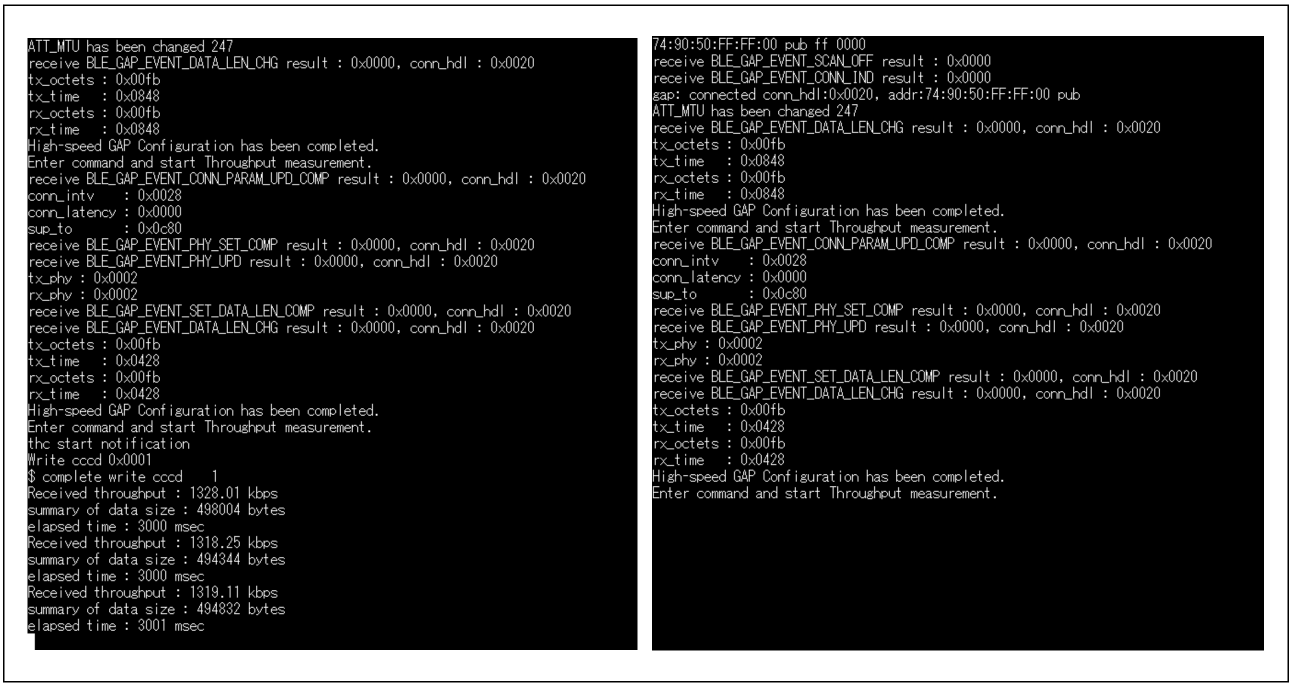

Figure 11. Server and client console screens after throughput measurement stopped

(left : client, right : server)

2.4.3 Message Sequence Chart

Figure 12 shows the message sequence chart when notification or indication sent from the server to the client.

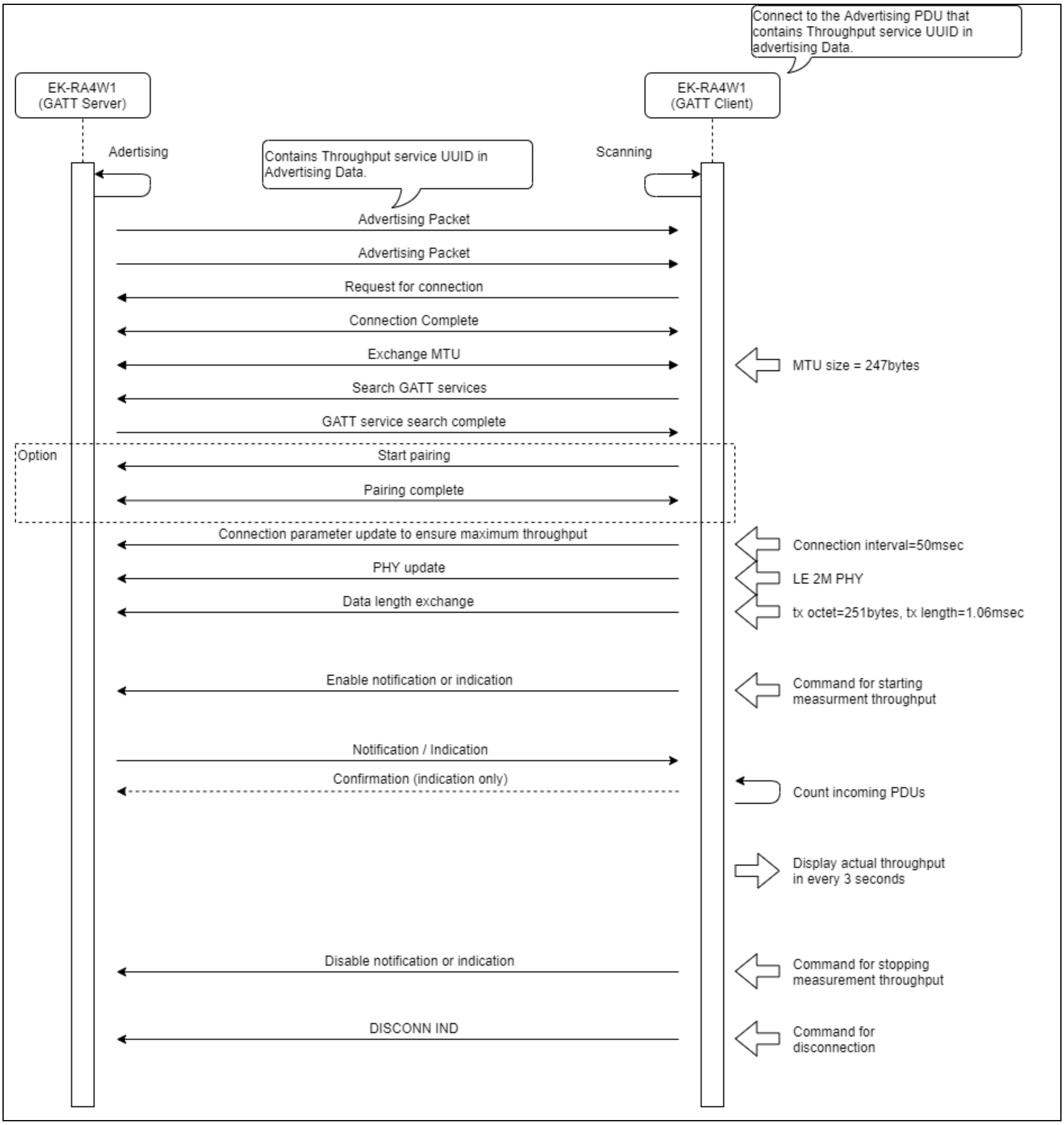

Figure 12. Message Sequence Chart of Notification and Indication

Figure 13 shows the message sequence chart when write or write without response sent from the client to the server.

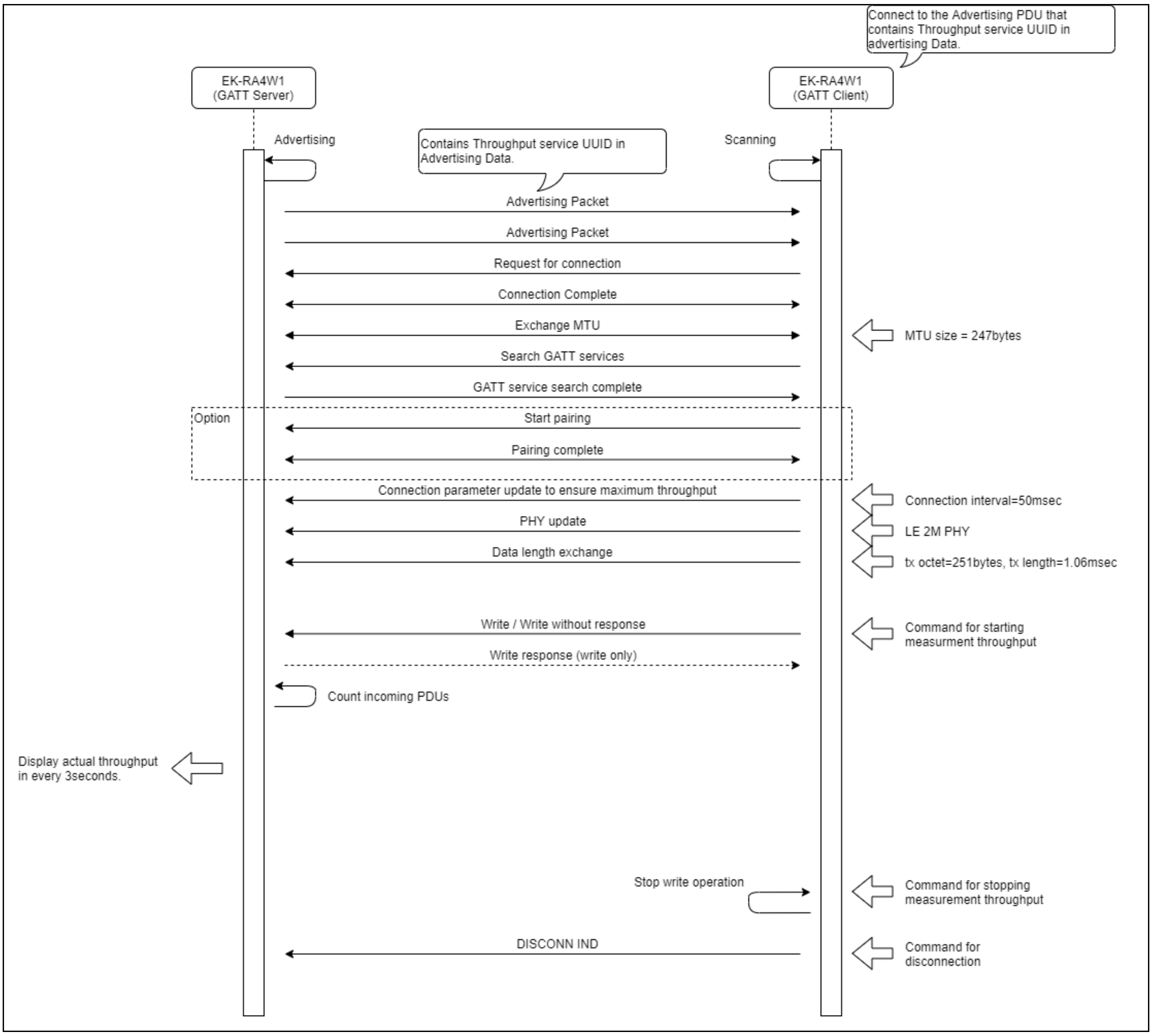

Figure 13. Message Sequence Chart of Write and Write without Response

3. Demo project implementation

This chapter describes relationship between the Bluetooth low energy communication mechanism and throughput. For details on communication standards, please refer to the Bluetooth Core Specification. Bluetooth Low Energy has three major layers Controller, Host and Host Controller Interface (HCI). Controller and Host are connected by Host Controller Interface (HCI). Application and Host are connected by API (*R_BLE* and *RM_BLE_ABS API* in BLE Abstraction Driver on rm_ble_abs in Flexible Software Package) (Figure 14).

In Bluetooth Low Energy, the Link Layer of the controller controls the actual communication path and transmission / reception interval. The operation of this Link Layer is important to achieve highspeed communication. The behavior of the Link Layer is set by the Generic Access Profile (GAP) of the Host Layer.

The Generic Attribute Profile (GATT) of the host layer used to send meaningful application data. The profile determines the application data transmission procedure and the data structure to be transmitted and received. The design of the profile is also important for achieving highspeed communication.

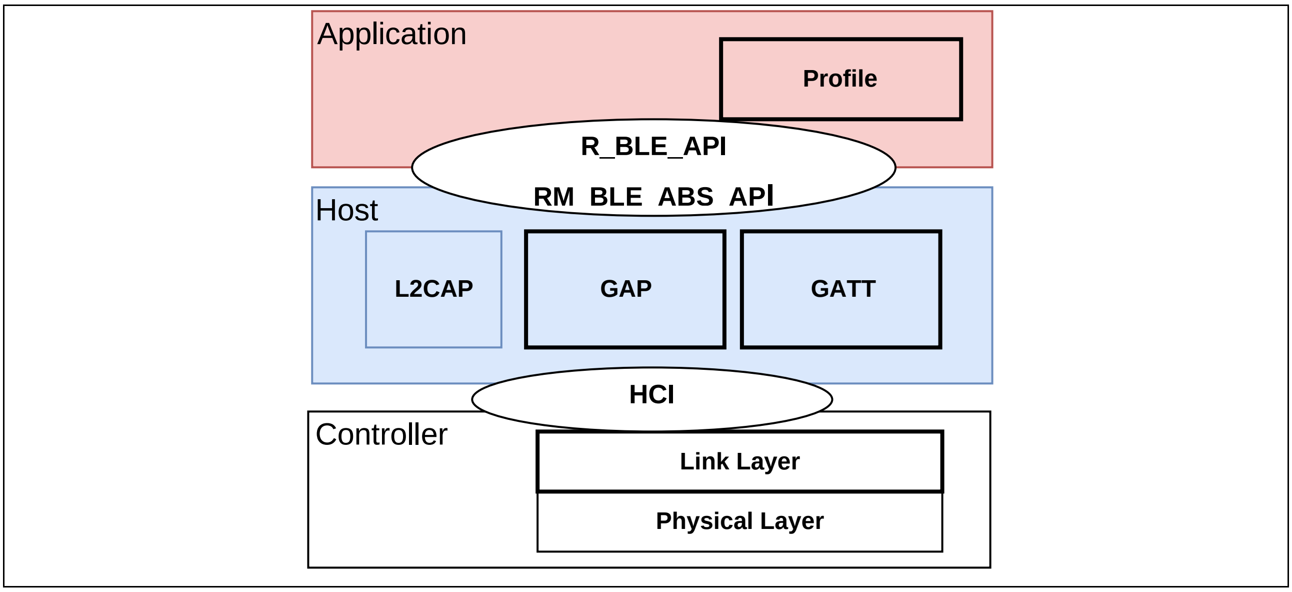

Figure 14. Three major layers in Bluetooth Low Energy

3.1 Generic Access Profile (GAP)

Generic Access Profile (GAP) defines the procedure for detecting connectable devices and establishing connections. GAP sets the operation of Link Layer and realizes these procedures.

3.1.1 Device detection and connection establishment

In Bluetooth Low Energy, a connection is established by one device transmitting (advertising) its own device information and the other device performing device detection (scanning) and connection request (initiating). The device that performs scanning and connection request is the central device, and the device that advertises is the peripheral device. Central determines parameters related to connection maintenance such as frequency map and communication interval (connection interval) after connection is established. The GAP will manage the following information.

- Connection Interval
- PHY
- Maximum Packet Length
- Information for Pairing
- etc.

3.1.2 Communication after establishing connection

In Bluetooth Low Energy, after the connection is established, the device exchanges radio frames with a connection event that occurs at each connection interval. In the Link Layer, the central is the master and the peripherals are the slaves. Radio frames are transmitted by the master in time with pre-shared connection events. (Figure 15)

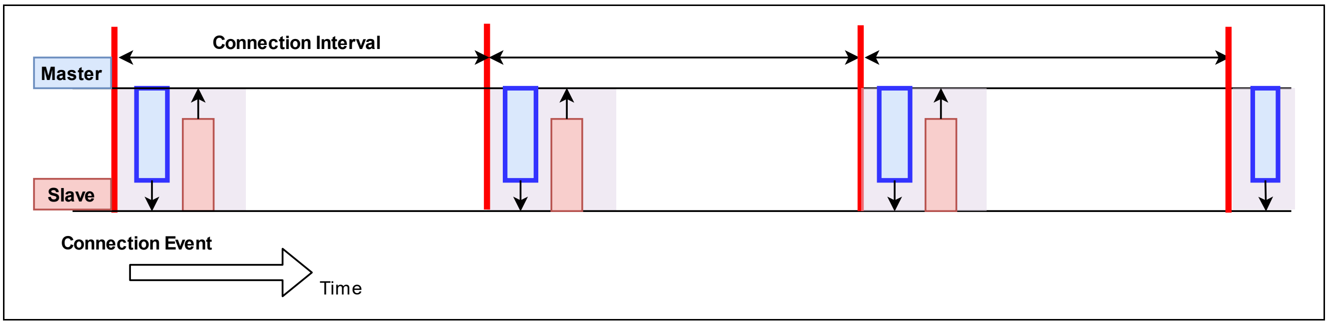

Figure 15. Exchange of connection event and radio frame

The connection is maintained by exchanging radio frames at the connection event. If there is additional data to send to either device, the More Data Bit in the radio frame will be set and the connection event will be extended. The connection event ends when the More Data Bit of each other is no longer set or when an error occurs in the received packet. Once the connection event ends, radio frames are not exchanged until the next connection event (Figure 16). In order to achieve highspeed communication, it is important to communicate using this More Data feature.

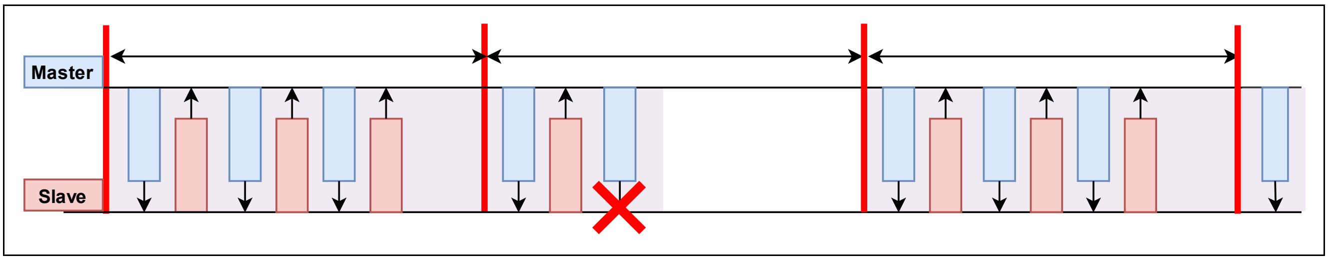

Figure 16. Communication by More Data feature

3.1.3 Setting the connection interval

Figure 17 shows Link Layer operation when the connection interval is changed. Even if the connection interval is changed, there is no significant change in throughput as long as the communication by the More Data feature is stable. Note that if the connection interval is extremely short, the throughput will become worse due to the overhead of waiting time for each connection interval.

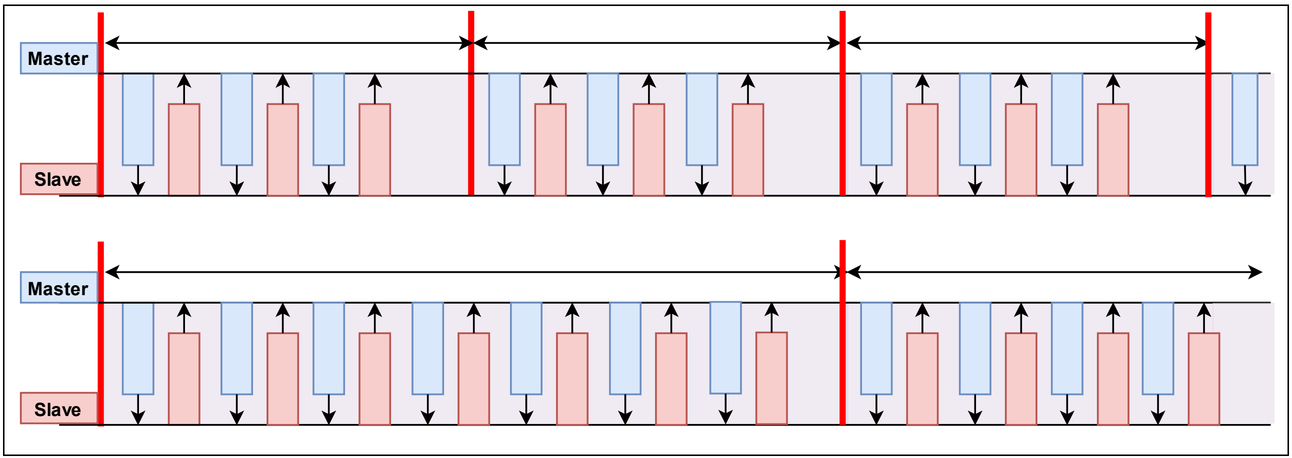

Figure 17. Change in connection interval and number of radio frames

Figure 18 shows the relationship between the connection interval and throughput when using notification case. And assume that the communication environment is good and frame exchange is always successful. The settings of GAP and GATT are PHY: 2M PHY, maximum packet length is 251 bytes, MTU is 247 bytes, and 244 bytes of application data are always notified. If the connection interval is 7.5msec, it will be about 1010 kbps due to overhead of waiting time for each connection interval.

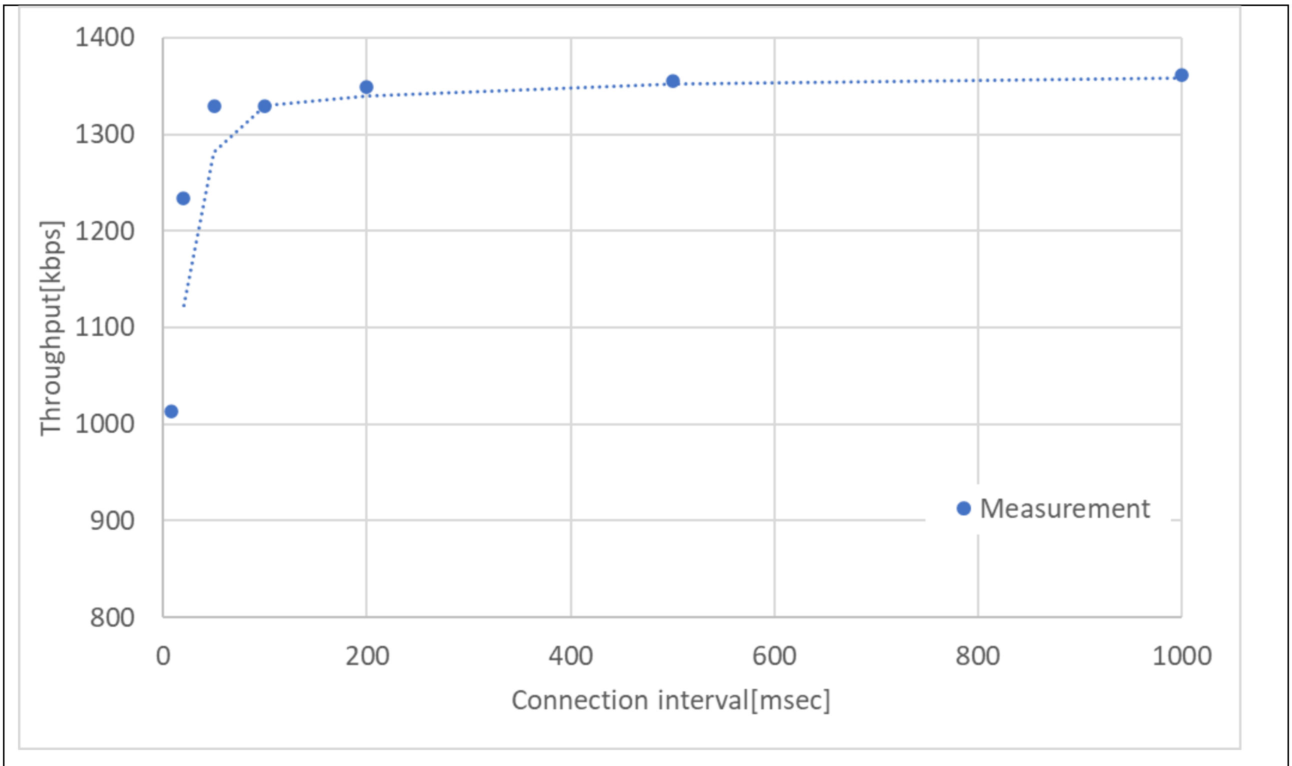

Figure 18. The relationship connection interval and throughput

The throughput per connection interval is calculated from the waiting time T_{overhead} immediately before the connection event, the minimum transfer time T_{frame} for the radio frame to make a round trip, and the application data length (L_{data}). If the packet length is 251 bytes, T_{frame} will be about 1.408 msec.

$$Throughput\ (kbps) = floor(\frac{connection\ interval - T_{overhead}}{T_{frame}}) * 8 * L_{\text{data}} * \frac{1}{connection\ interval}$$

If the communication environment is good, the throughput will not be affected even if the connection interval becomes long. However, if the communication by using the More Data bit is interrupted due to a communication error (e.g. interference), the effect of the connection interval on the throughput will be large (Figure 19). When a communication error occurs, each device waits until the next connection event. Therefore, throughput decreases as the connection interval increases.

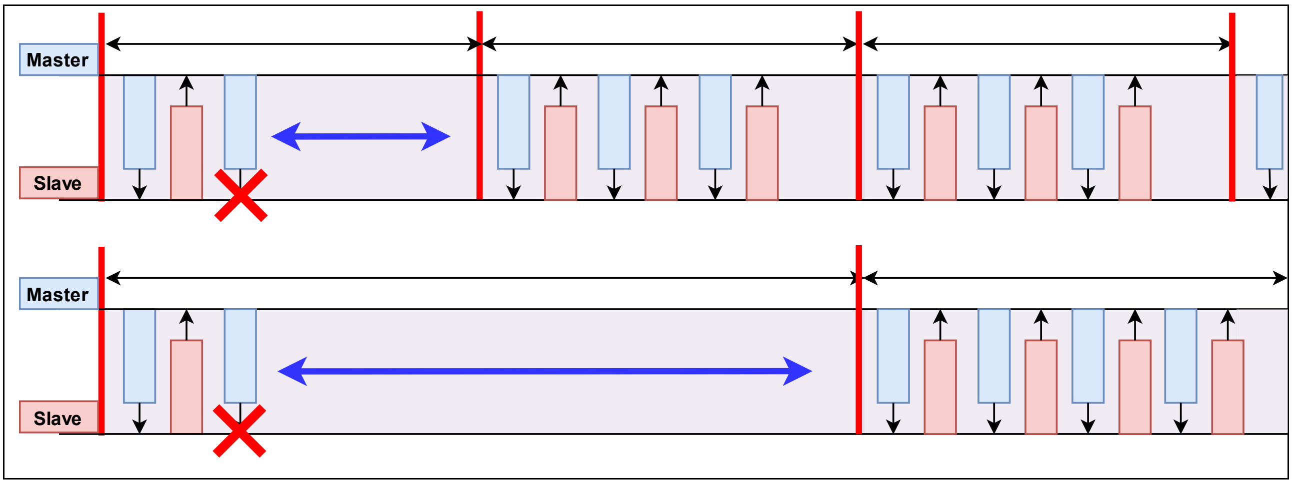

Figure 19 Connection interval and throughput when communication error occurs

Figure 20 shows the relationship between connection interval, probability of frame exchange failure, and throughput. GAP settings are PHY: 2M PHY, maximum packet length 251 bytes, MTU 247 bytes, and the value when 244 bytes of application data are always notified. The expected value of throughput per connection interval is plotted.

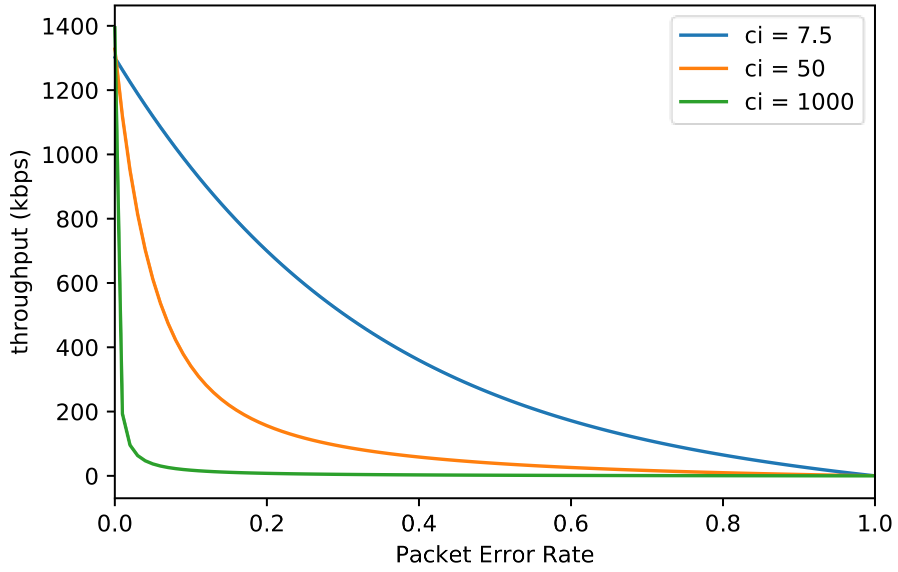

Figure 20. Relationship between frame exchange failure probability and throughput

To change the connection interval, execute the “gap conn_cfg update” command on terminal emurator. For details about the command, refer to *RA4W1 Group BLE sample application*(R01AN5402) . *R_BLE_GAP_UpdConn* API can also be used to change connection interval. For details about the API, refer to " *RA Flexible Software Package Documentation*”.

3.1.4 Setting the PHY

Figure 21 shows Link Layer operation when the physical layer (PHY) settings are changed. When the physical layer PHY is changed, the radio frame occupation time will be changed. If the data length is the same, the occupation time in 2M PHY is about half of that in 1M PHY. If the occupation time of one frame in the air is short, the number of packets transmitted / received per unit time increases, and the throughput improves.

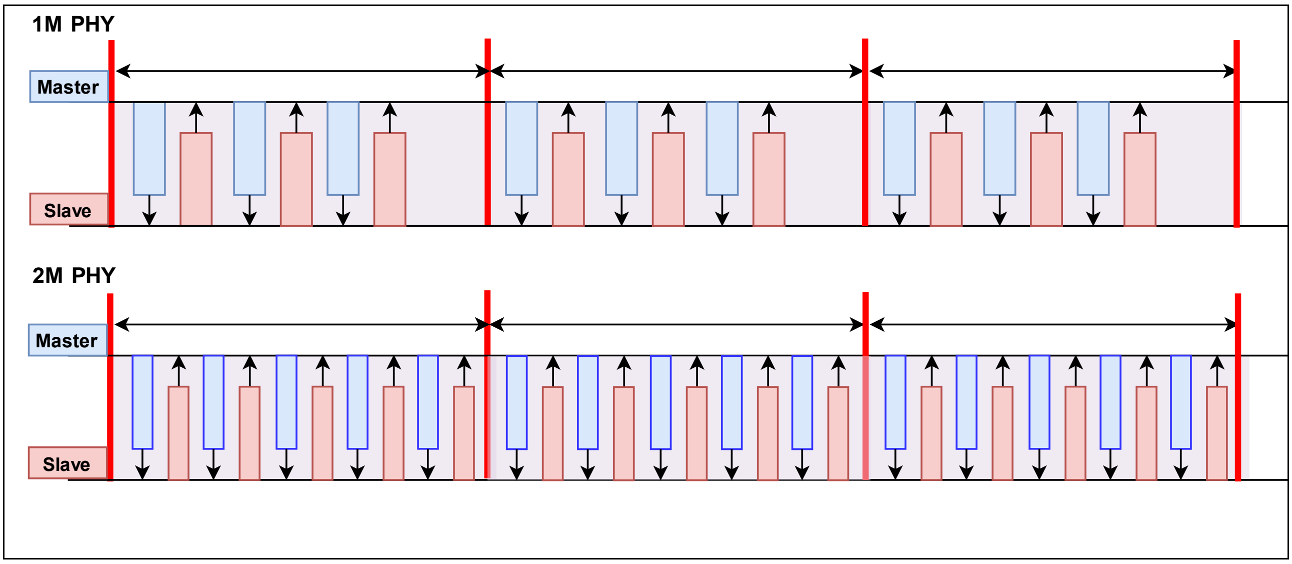

Figure 21. Schematic diagram when using 2M PHY

To change the PHY, execute the “gap conn_cfg phy” command on terminal emulator. For details about the command, refer to *RA4W1 Group BLE sample application* (R01AN5402). *R_BLE_GAP_SetPhy* API can also be used to change PHY. For details about the API, refer to " *RA Flexible Software Package Documentation*”.

3.1.5 Setting the Maximum packet length

Figure 22 shows Link Layer operation when the maximum packet length is set to long. The application information can be transmitted efficiently by minimizing the header information of the radio frame and the transmission / reception interval.

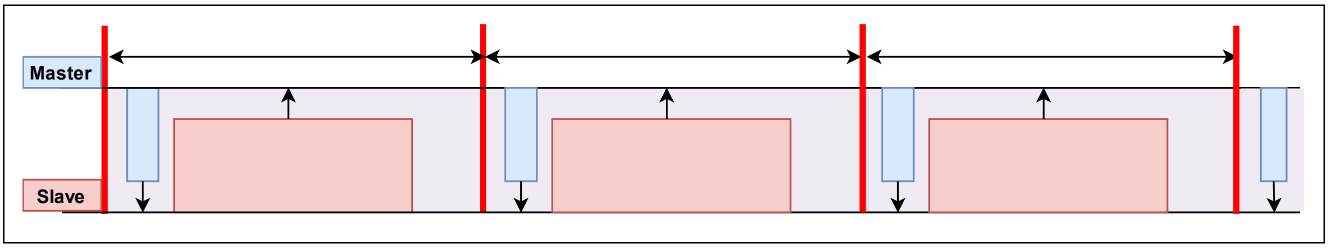

Figure 22. Schematic diagram of Link Layer when changing packet length

To change the Maximum packet length, execute the “gap conn_cfg data_len” command on terminal emulator. For details about the command, refer to *RA4W1 Group BLE sample application* (R01AN5402). *R_BLE_GAP_SetDataLen* API can also be used to change maximum packet length. For details about the API, refer to " *RA Flexible Software Package Documentation*”.

3.1.6 Setting the encryption of communication

Figure 23 shows Link Layer operation when communication is encrypted. Through encryption, the data for checking packet integrity (4 bytes) is carried in the radio frame, which may reduce the throughput.

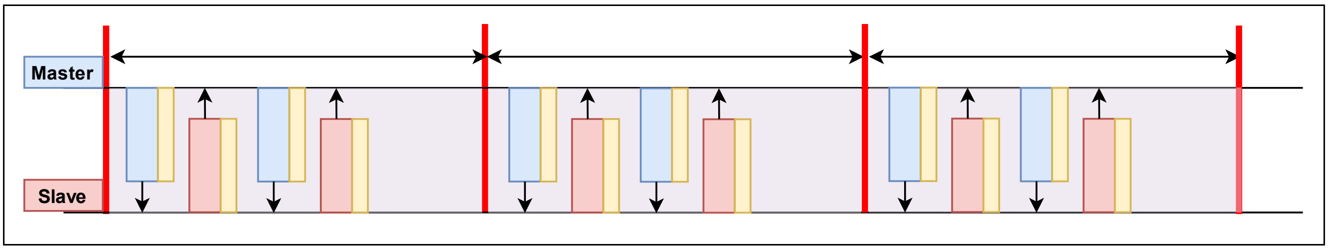

Figure 23. Schematic diagram of Link Layer in encrypted communication

To encrypt communication, execute the “gap auth start” command on terminal emulator. For details about the command, refer to *RA4W1 Group BLE sample application* (R01AN5402).
RM_BLE_ABS_StartAuthentication API can also be used to encrypt communication. For details about the API, refer to " *RA Flexible Software Package Documentation*”.

3.2 Generic Attribute Profile (GATT)

GATT determines the communication procedure for application data (e.g. sensor data). GATT implements a client-server architecture over the communication path established by GAP. The client reads / writes data from / to the GATT database held by the server using a predetermined procedure. The server returns a response to the client. And it is also possible for the server to notify to the client (Figure 24).

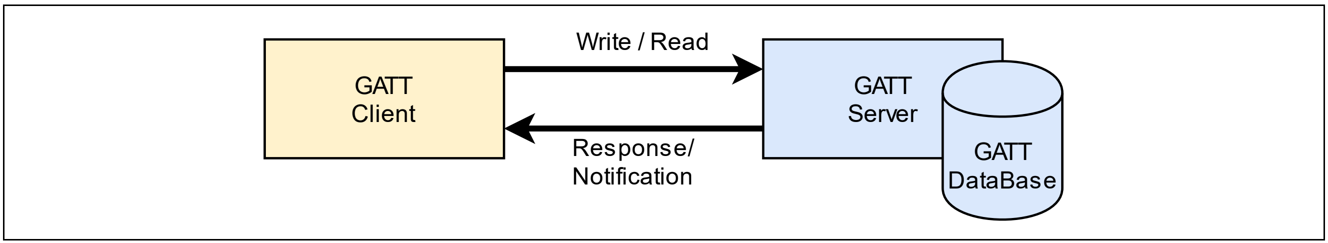

Figure 24. GATT architecture

The feature of an application is called a "service", and the data required for that feature is called a "characteristic." A "profile" is a set of services required to realize an application and defines the communication specifications of the application.

Figure 25 shows the relationship between profile, service, and characteristic when using a thermometer as an example. The features of the thermometer application are temperature measurement and device information. Features and data are kept on the GATT database as services and characteristics.

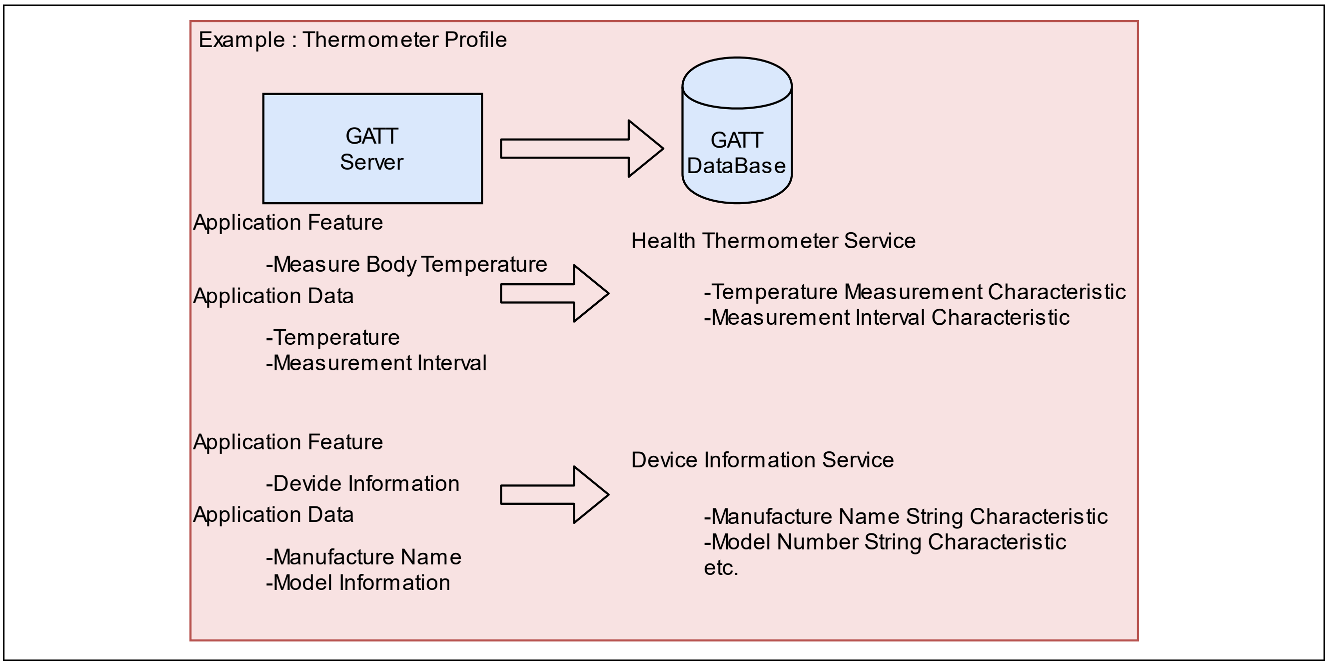

Figure 25. Application Data and GATT database

When performing GATT communication, it is necessary to share service and characteristic information between client and server. The service and characteristic information is store in database of server. Immediately after establishing a connection, the client does not have service information for the server. The client queries the server for a particular service by using service discovery procedure (Figure 26). The client obtains information about the particular service and handle of the service from the GATT database held by server. The client will use the handle to read and write to the GATT database.

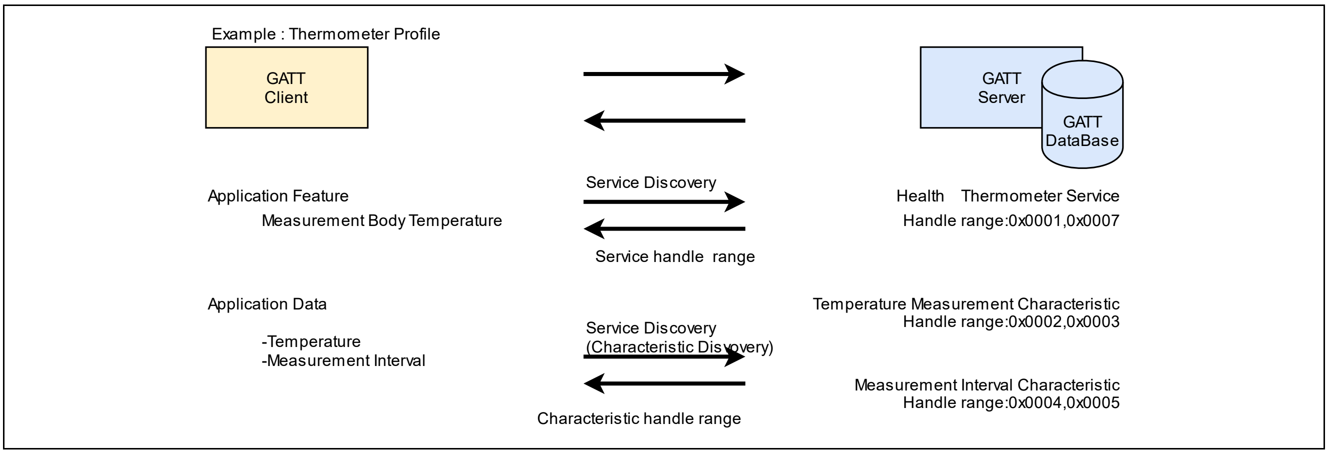

Figure 26. Service discovery operation

Characteristic defines the data, its structure, and the procedure for exchanging data between the client and server. The characteristic descriptor defines additional options for the data exchange procedure.

Table 7 summarizes the typical data exchange procedures. These procedures are categorized by the direction of data transmission and the presence or absence of a response. A procedure that requires a response cannot perform the same procedure before receiving a response.

Table 7. Typical communication procedure of GATT communication

Procedure name	Operation	Direction to transmit	Response required
Read	Read	From client to server	Yes
Write	Write	From client to server	Yes
Write Without Response	Write	From client to server	No
Indication	Notify	From server to client	Yes
Notification	Notify	From server to client	No

Figure 27 shows the Read operation in which the client reads the server data. Data communication by GATT is performed based on the handle information.

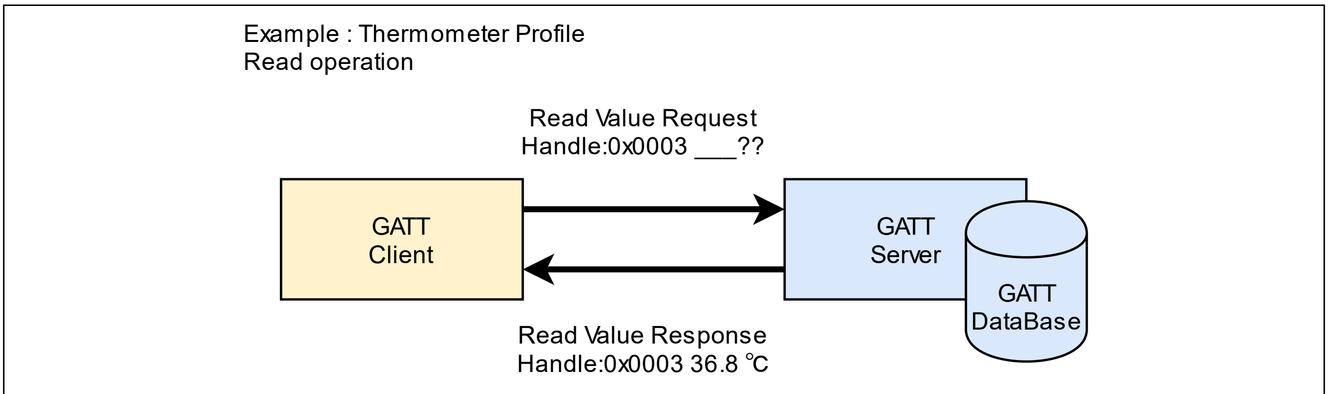

Figure 27. Read operation

3.2.1 No response operation (Notification / Write Without Response)

In Notification or Write Without Response operations, the next packet can be transmitted without waiting for the response. Therefore, it is possible to continuously send data by using the More Data feature. Figure 28 shows the Notification operation.

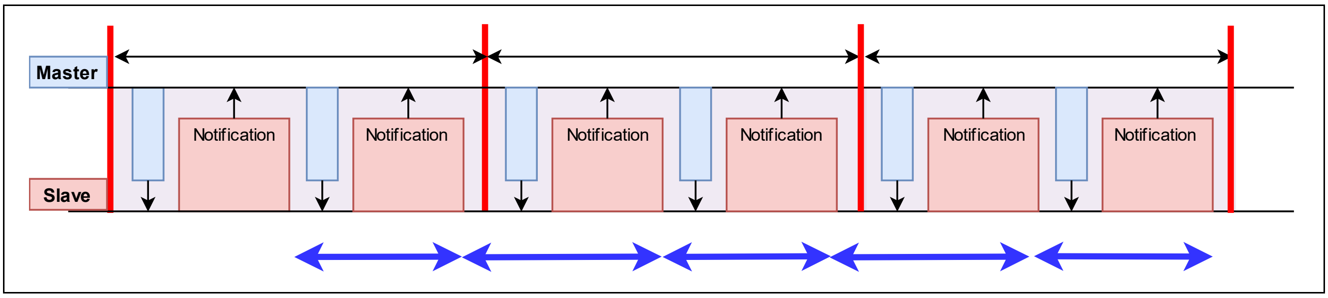

Figure 28. Schematic diagram of Link Layer during Notification operation

In the figure above, the slave acts as a server and performs a Notification.

3.2.2 Response operation (Indication / Write)

In Indication and Write operations, the next packet can be transmitted after the response. Therefore, a request to transmit the next data cannot be sent in one connection event, and it cannot be performed More Data feature. Figure 29 shows Link Layer operation for Indication. It takes twice as long as the connection interval to transmit one data packet.

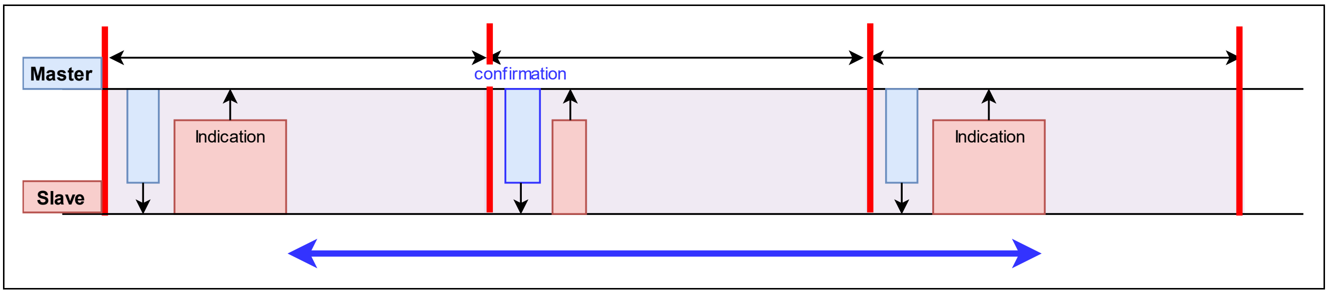

Figure 29. Schematic diagram of Link Layer during Indication operation

In the figure above, the slave acts as a server and performs Indication.

3.3 Profile design by using QE for BLE

This section describes how to create profiles by using QE for BLE, using the custom throughput service used in the demo project as an example. Throughput service has characteristics shown in Table 5. The purpose of these characteristics are data transmitted and received for throughput measurements.

To open QE for BLE, select Renesas views -> Renesas QE -> R_BLE Custom Profile RA, RE (QE) on e2studio menu bar.

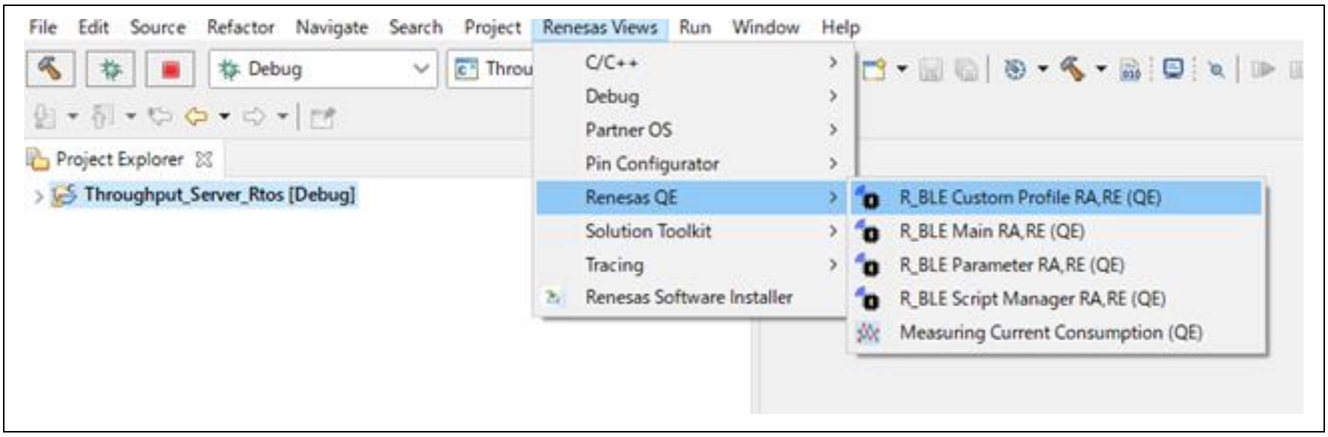

Figure 30. e2studio menu

And then, following screen will open.

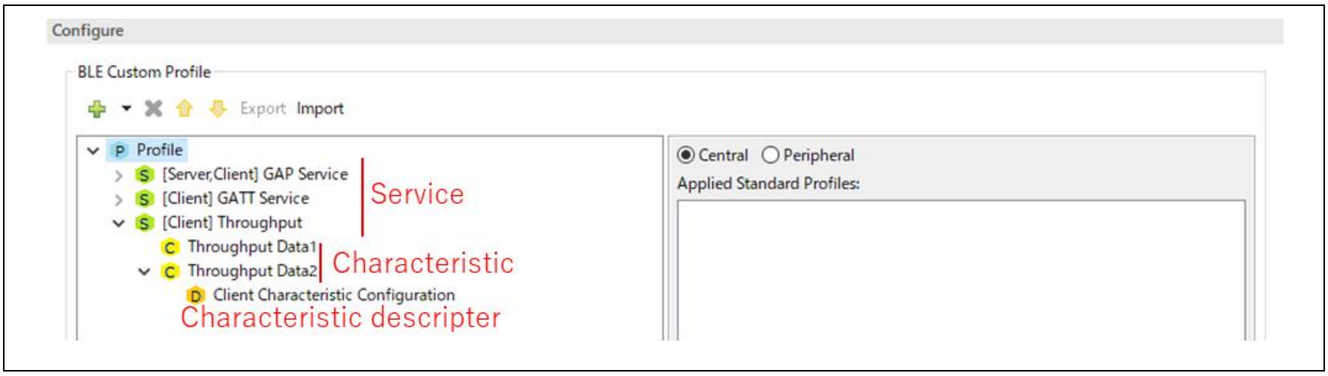

Figure 31. Profile configuration screen

Figure 32 shows the configuration screen of the Throughput Data 1 Characteristic of the throughput service.

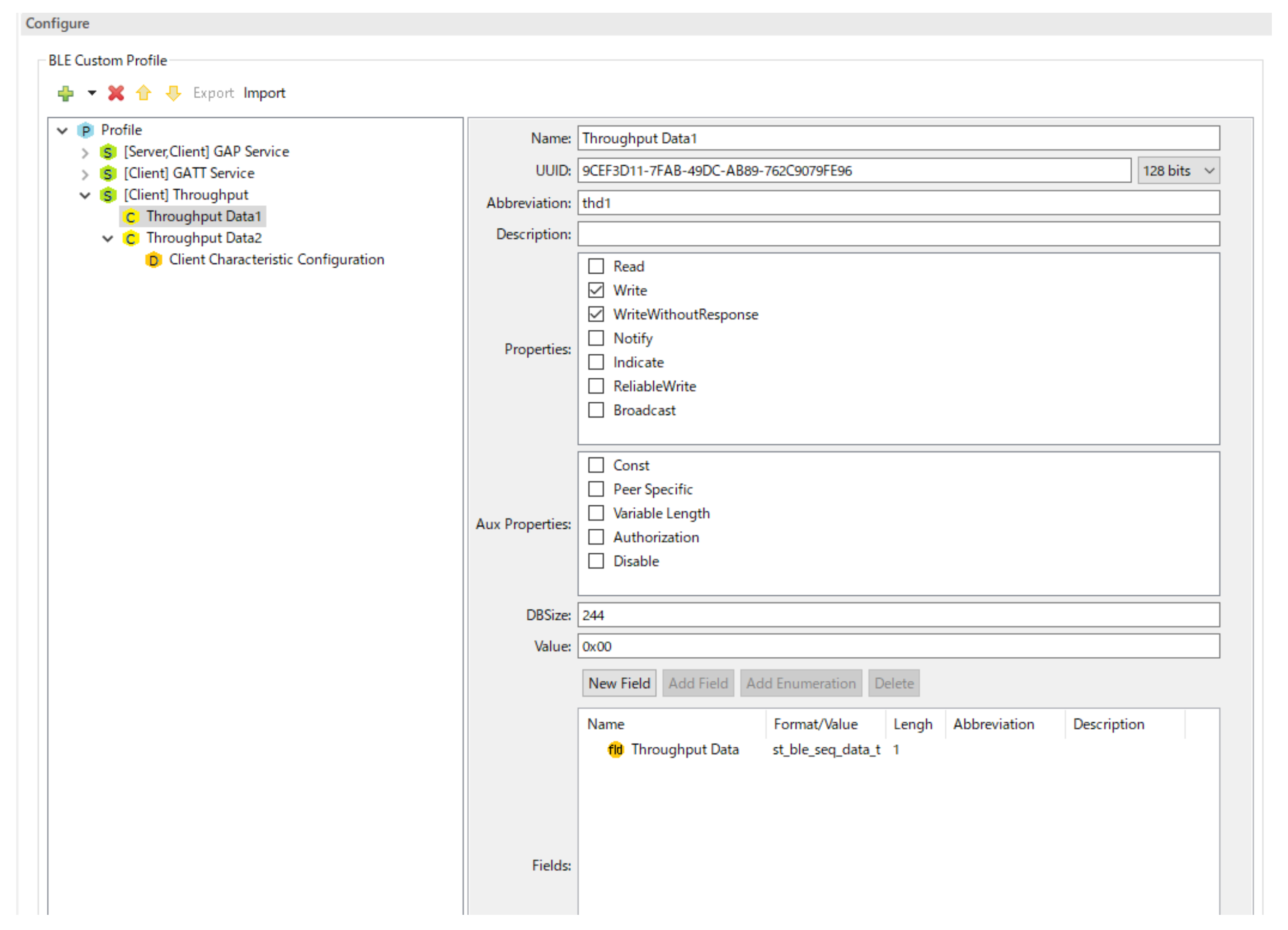

Figure 32. Characteristic configuration screen of Throughput Data 1 Characteristic

It is necessary to configure following properties in above screen.

1) Name of characteristic, UUID, Abbreviation and check Write and Write Without Response of Properties.
2) Aux Properties configure data management in the GATT database. There are no items to set for the throughput service.
3) Set DBSize and Value. DBSize sets the number of bytes reserved for storing data on the GATT database.
4) Finally, set Fields. Field defines the characteristic structure handled in the application. This characteristic structure is reciprocally converted by the encode / decode functions set in the service API to packet data or GATT database data.

The characteristics of the throughput service in sample application use the st_ble_seq_data_t structure. This structure has members that mean the start address of array data and length of array data. The encode / decode functions will automatically generate by QE for BLE. Therefore, you can transmit and receive array data without implementing the encode / decode functions.

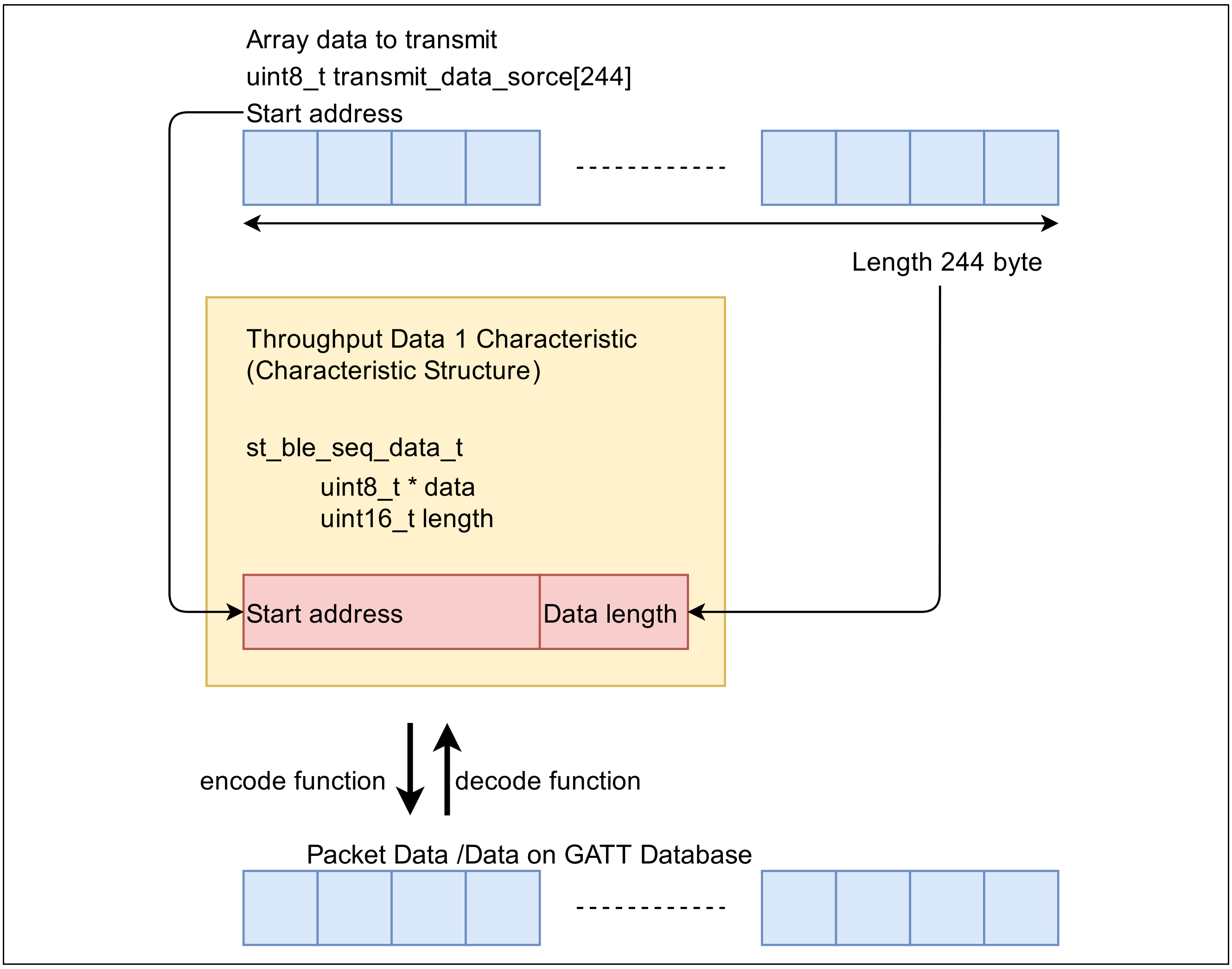

Figure 33. Transmitting array data using st_ble_seq_data_t structure

For details on how to develop a profile using QE for BLE, refer to "*RA4W1 Group Bluetooth Low Energy Profile Developer's Guide* (R01AN5428)".

4. Appendix

4.1 Command reference

(1) Throughput measurement command

<table>
<tr><td colspan="3">thc command</td></tr>
<tr><td rowspan="2">Format :</td><td colspan="2">thc [operation] {param}</td></tr>
<tr><td colspan="2">Start or stop throughput measurement</td></tr>
<tr><td rowspan="2">Parameters :</td><td>[operation]</td><td>Start or stop throughput measurement.
start : start throughput measurement
stop : stop throughput measurement</td></tr>
<tr><td>{param}</td><td>[operation] : start
notification : start notification from the server to the client.
indication : start indication from the server to the client.
write_without_response : start write without response from the client to the server
write : start write from the client to the server.
[operation] : stop
receive : stop notification, indication from the server
transmit : stop write, write without response from client</td></tr>
<tr><td>Example :</td><td colspan="2">thc start notification
Start notification from server to client.

thc stop receive
Stop notification or indication from server to client.</td></tr>
</table>

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	June.17.2021	—	First edition issued

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

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/.