BLE Virtual UART Application

Android programming guide

R01AN3842EJ0101
Rev.1.01
Jun 30, 2017

Introduction

This document describes the programming method of an Android application which performs data communication between PC (terminal software) and Android device by using the BLE Virtual UART Application of RL78/G1D.

Target Device

- RL78/G1D Evaluation Board（RTK0EN0001D01001BZ）
- Android 5.0 or later

Related Documents

Document Name	Document No.
Bluetooth® Low Energy Protocol Stack	
BLE Virtual UART Application	R01AN3130E
GATTBrowser for Android	
Smartphone Application Instruction manual	R01AN3802E

Contents

1. Overview

This document describes the programming method of an Android application which performs data communication between PC (terminal software) and Android device by using the BLE Virtual UART Application of RL78/G1D.

Figure 1-1 shows the overall configuration. Connect a RL78/G1D evaluation board to a PC via USB cable. A user operates a RL78/G1D evaluation board through a terminal software. A RL78/G1D evaluation board communicate with Android devices using BLE.

From the perspective of PC (terminal software), data is exchanged between PC (terminal software) and Android devices with virtual UART connection via RL78/G1D.

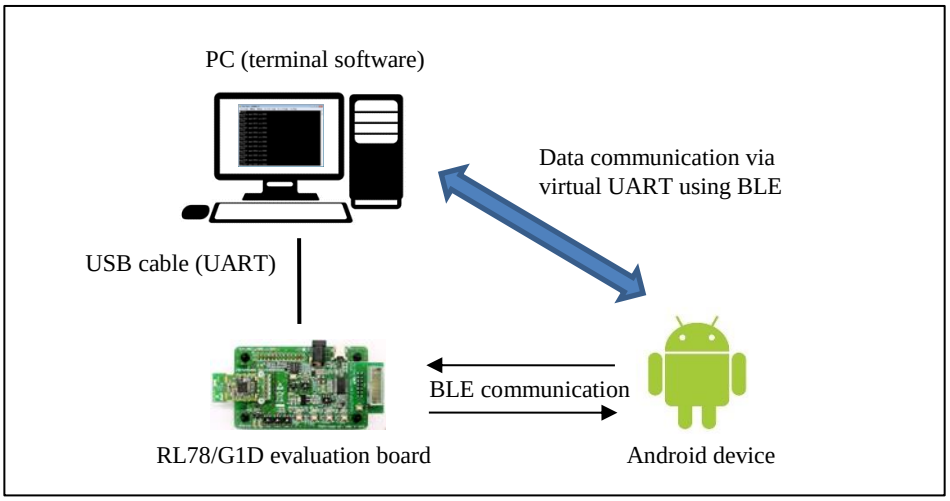

Figure 1-1 Overview

Note: The source code described in this document is not to guarantee the actual operation but to explain the programming method. Also, a sample application for Android using them will be not provided.

2. Evaluation environment

- Windows®7 Service Pack1 or later
- Android 5.0 or later
- RL78/G1D Evaluation Board (RTK0EN0001D01001BZ)

  https://www.renesas.com/products/software-tools/boards-and-kits/evaluation-demo-solution-boards/rtk0en0001d01001bz.html
- Renesas On-chip debug emulator E1 (used for writing firmware)

  https://www.renesas.com/products/software-tools/tools/emulator/e1.html
- Renesas Flash Programmer (used for writing firmware)

  https://www.renesas.com/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html
- Bluetooth® low energy Protocol Stack BLE Virtual UART application (R01AN3130)

  https://www.renesas.com/search/keyword-search.html#q=r01an3130&genre=tooldownload
- GATTBrowser for Android

  https://play.google.com/store/apps/details?id=com.renesas.ble.gattbrowser
- Tera Term

  https://ttssh2.osdn.jp/
- 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 Devices International) - Drivers

http://www.ftdichip.com/Drivers/D2XX.htm

3. Reference material

- Android Bluetooth Low Energy API Guide

  https://developer.android.com/guide/topics/connectivity/bluetooth-le.html
- Android BluetoothLeGatt Sample Source Code

  https://developer.android.com/samples/BluetoothLeGatt/index.html
- Bluetooth Core Specification v4.2 (PDF)

  Bluetooth Core Specification v4.2

4. Operation check using GATTBrowser

4.1 Preparation of RL78/G1D

Download the BLE Virtual UART Application (R01AN3130) from the following website.

Bluetooth® low energy Protocol Stack BLE Virtual UART Application (R01AN3130)

https://www.renesas.com/search/keyword-search.html#q=r01an3130&genre=tooldownload

Expand the zip file and write the following hex file to RL78/G1D.

ROM_File/cc_rl/RL78_G1D_CCE(VUART).hex

Start the terminal software on the PC and set up the configuration according to Table 4-1. This document uses Tera Term for terminal software.

Table 4-1 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

Figure 4-1 shows the setup screen of Tera Term.

- Select "Serial Port..." from "Setup" menu and set "Baud rate", "Data", "Parity", "Stop", and "Flow control"
- Select “Terminal...” from “Setup” menu and set ”New-line Receive” and “New-line Transmit”

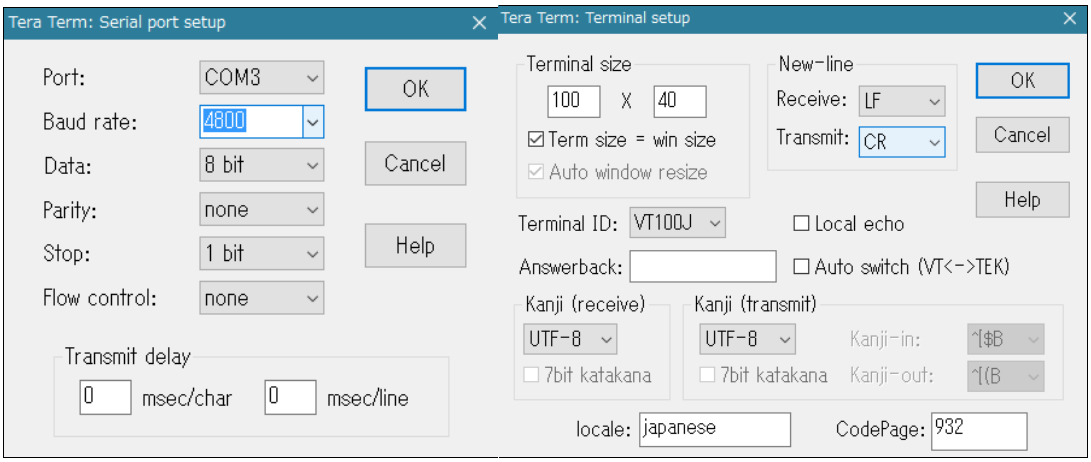

Figure 4-1 Terminal software (Tera Term) settings

When the connection between the RL78/G1D and the terminal software is completed, enter the ESC key on the terminal software and switch the RL78/G1D to the "Virtual UART Mode".

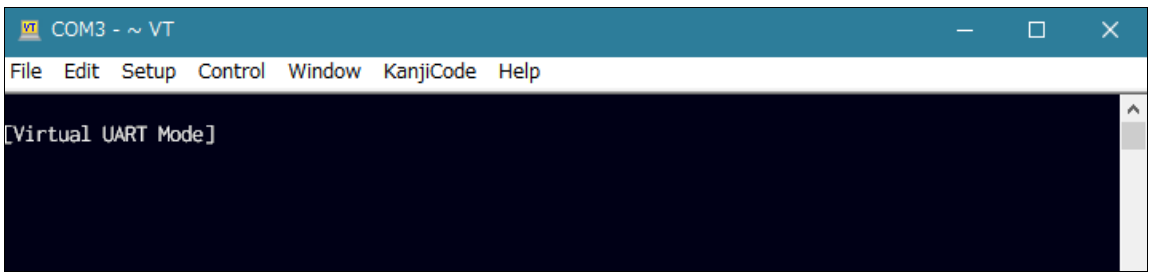

Figure 4-2 Virtual UART Mode

Reference: BLE Virtual UART Application (R01AN3130) “7.3 Preparation for Execution Environment”

4.2 Preparation of GATTBrowser

GATTBrowser is a general-purpose application for Android that can scan BLE devices, connect with those devices, and perform GATT-based communication. In this section, GATTBrowser will be used to check the operation of BLE Virtual UART function of RL78/G1D.

GATTBrowser for Android

https://play.google.com/store/apps/details?id=com.renesas.ble.gattbrowser

Start GATTBrowser and receive the advertising of RL78/G1D. (Figure 4-3)

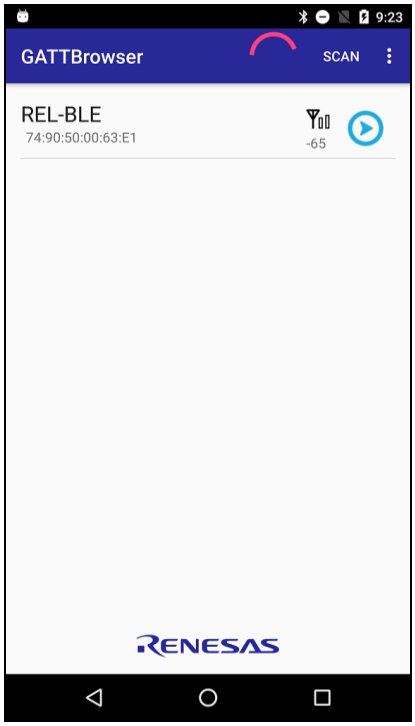

Figure 4-3 Scanning (GATTBrowser)

Receive the advertising of RL78/G1D and connect to the RL78/G1D. When the connection is successful, the “Service and Characteristic list” screen is displayed.

After the “Service and Characteristic list” screen is displayed, tap the "Indication Characteristic" item in the "Renesas Virtual UART Service" item to move to the "Indication Characteristic" operation screen. (Figure 4-4)

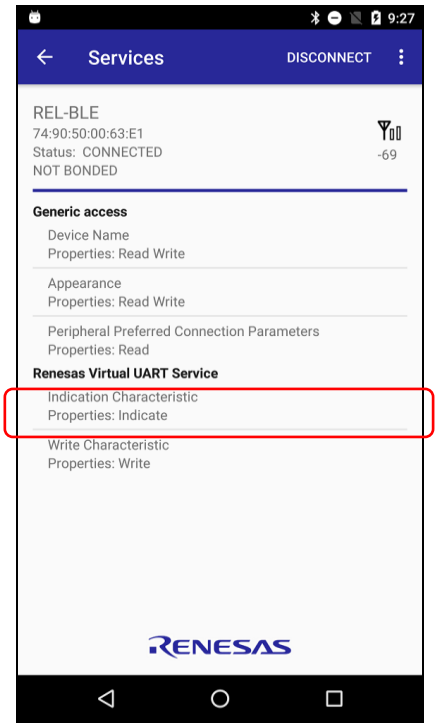

Figure 4-4 Service and Characteristic list (GATTBrowser)

After moving to the "Indication Characteristic" operation screen, tap the "Indication Off" toggle button and put it in the "Indication On" state. (Figure 4-5)

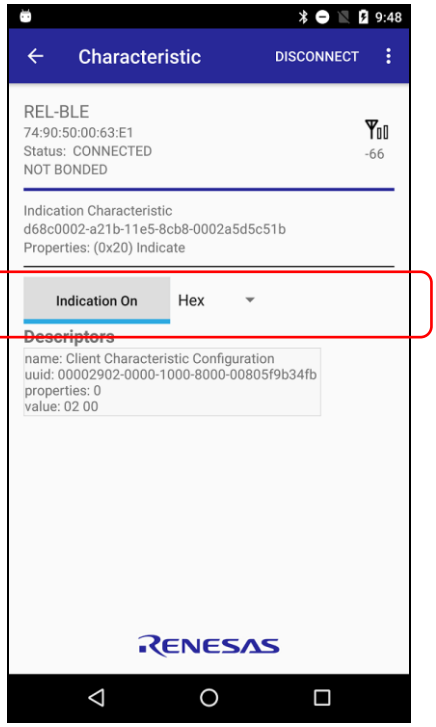

Figure 4-5 Characteristic operation (GATTBrowser)

When the toggle button becomes the "Indication On", The character string "CONNECT" is displayed in the terminal software. (Figure 4-6)

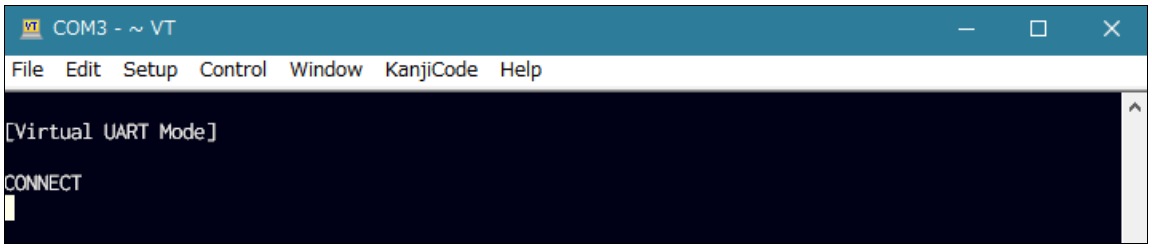

Figure 4-6 Ready for the Indication (Terminal software)

Up to this point, preparation for data communication between RL78/G1D and GATTBrowser has been completed.

4.3 Sending data from RL78/G1D to GATTBrowser

This section describes how to send data from RL78/G1D to GATTBrowser.

In the terminal software, enter the character string "abc". (Figure 4-7)

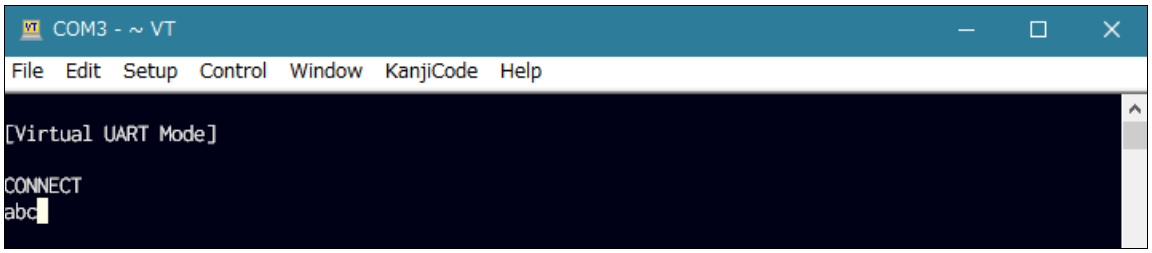

Figure 4-7 Sending the data to Android (Terminal software)

The character string "abc" entered by the terminal software is sent to the GATTBrowser with the BLE Virtual UART function of RL78/G1D. Figure 4-8 shows the screen where the GATTBrowser received the character string "abc".

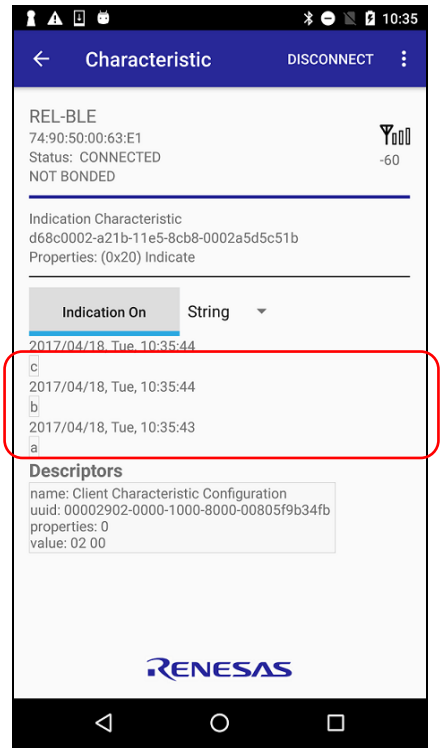

Figure 4-8 Receiving the data from RL78/G1D (GATTBrowser)

4.4 Sending data from GATTBrowser to RL78/G1D

This section describes how to send data from GATTBrowser to RL78/G1D.

Tap the Back button (" ◁ ") or the Up button (" ← ") in the GATTBrowser and return to the “Service and Characteristic list” screen from the "Indication Characteristic" operation screen. (Figure 4-9)

After moving to the “Service and Characteristic list” screen, tap the "Write Characteristic" item in the "Renesas Virtual UART Service" item and go to the "Write Characteristic" operation screen.

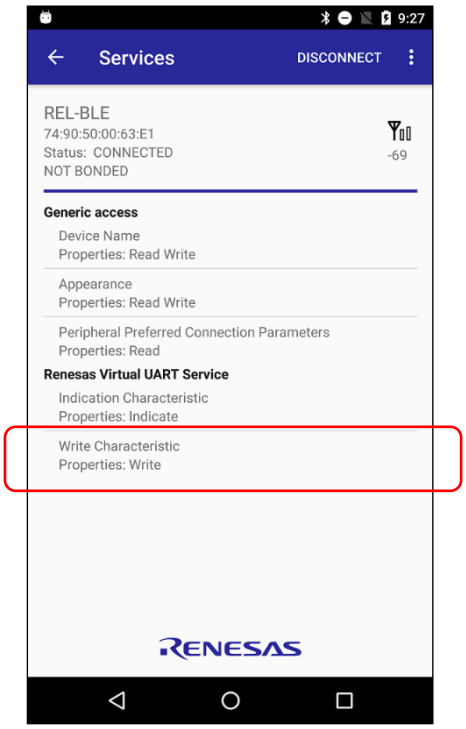

Figure 4-9 Service and Characteristic list (GATTBrowser)

After moving to the "Write Characteristic" operation screen (Figure 4-10), execute the following procedure.

1. Select "String" with the "Write mode selection" spinner next to the "Write" button
2. Enter the character string “Renesas-BLE” in the sending text field.
3. Tap the “Write” button

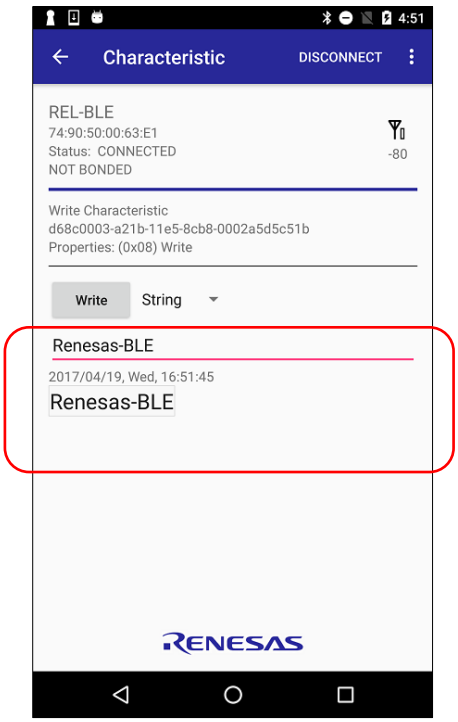

Figure 4-10 Sending the data to RL78/G1D (GATTBrowser)

When the "Write" button is tapped, the sending data is sent from the GATTBrowser to the RL78/G1D using the BLE Virtual UART function of RL78/G1D. Figure 4-11 is the screen that the RL78/G1D outputs the data ("Renesas-BLE") received from the GATTBrowser to the terminal software.

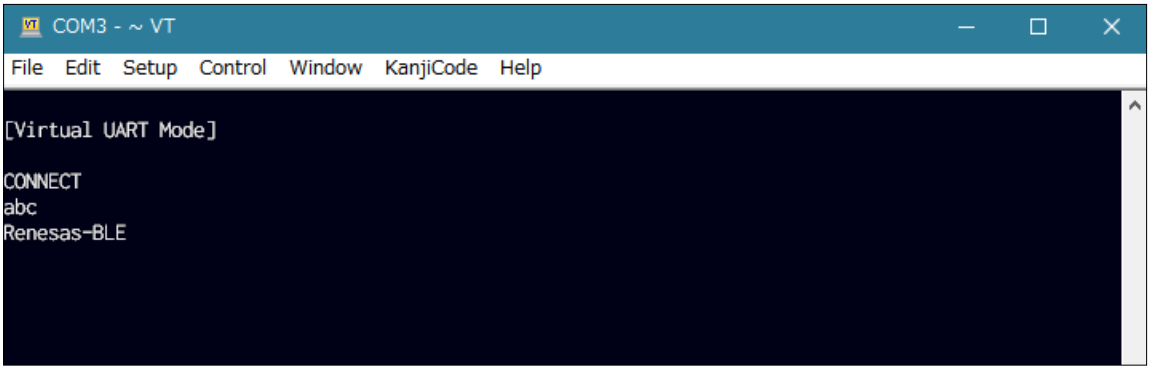

Figure 4-11 Receiving the data from Android (Terminal software)

5. Virtual UART Profile

Table 5-1 shows the Virtual UART Profile specification. This table is an excerpt from Bluetooth® Low Energy Protocol Stack BLE Virtual UART Application (R01AN3130) '8.1 Virtual UART Profile'

Table 5-1 Virtual UART Profile specification

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 character transfer from the server to the client
VUART_HDL_INDICATION_VAL 0x000E	Type: Indication Value By setting characters to this characteristic and send Indication, the characters are sent from the server to the client. Max 20 characters.
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 character transfer from the client to the server.
VUART_HDL_WRITE_VAL 0x0011	Type: Write Value By writing characters to this characteristic with "Write Request", the characters are sent from the client to the server. Max 20 characters.

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

Table 5-2 and Table 5-3 show the service and characteristics that an Android device needs to operate.

Table 5-2 Renesas Virtual UART Service

Service name	UUID
Renesas Virtual UART Service	D68C0001-A21B-11E5-8CB8-0002A5D5C51B

Table 5-3 Characteristics

Characteristic name	UUID	Properties
Indication Characteristic	D68C0002-A21B-11E5-8CB8-0002A5D5C51B	Indication
Write Characteristic	D68C0003-A21B-11E5-8CB8-0002A5D5C51B	Write

5.1 How to send data from Android to RL78/G1D

Data is sent from Android to RL78/G1D by writing a sending value to the “Write Characteristic” using the "Write Request". To write a value to a characteristic, use BluetoothGatt#writeCharacteristic().

Write Request:

BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part F], 3.4.5.1 Write Request

BluetoothGatt#writeCharacteristic() method:

https://developer.android.com/reference/android/bluetooth/BluetoothGatt.html#writeCharacteristic(android.bluetooth.BluetoothGattCharacteristic)

5.2 How to send data from RL78/G1D to Android

From Android, enable the Characteristic Value Indication (Indication) of the "Indication Characteristic".

If the Indication of the "Indication Characteristic" is enabled, when RL78/G1D changes the value of the "Indication Characteristic", it will be notified that the value of the "Indication Characteristic" has been changed to Android. By using this mechanism, data is sent from RL78/G1D to Android.

The actual method of enabling the Indication of the "Indication Characteristic" is described in 6.3.1.

Characteristic Value Indication:

BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part G], 3.3.3.3 Client Characteristic Configuration

BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part G], 4.11 CHARACTERISTIC VALUE INDICATIONS

6. Programming guide

- Android BluetoothLeGatt Sample Source Code

  https://developer.android.com/samples/BluetoothLeGatt/project.html
- Android Bluetooth Low Energy API Guide

  https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

This chapter describes how to program an Android application that communicates with RL78/G1D using the BLE Virtual UART function of RL78/G1D. The description is based on the BluetoothLeGatt sample application provided by the Android Developers website and the API guide of the Android Bluetooth Low Energy.

6.1 Scanning

Figure 6-1 is a screen of the BluetoothLeGatt sample application, receiving an advertising of RL78/G1D.

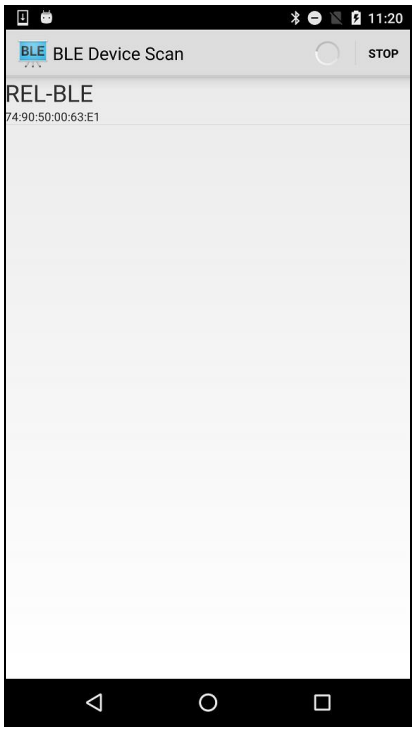

Figure 6-1 Scanning (BluetoothLeGatt sample application)

To run the BluetoothLeGatt sample application using Android 6.0 or later, it is necessary to support for the Runtime Permission introduced from Android 6.0

Requesting Permissions at Run Time

https://developer.android.com/training/permissions/requesting.html

If the Runtime Permission is not supported, the following exception occurs and the BLE function can't be used.

```
W/Binder: Caught a RuntimeException from the binder stub implementation.
        java.lang.SecurityException: Need ACCESS_COARSE_LOCATION or ACCESS_FINE_LOCATION permission to
get scan results
            at android.os.Parcel.readException(Parcel.java:1620)
            at android.os.Parcel.readException(Parcel.java:1573)
            ....
```

6.1.1 Support for Runtime Permission

■ Added support for Runtime Permission (Recommended)

1. Add the following declaration to AndroidManifext.xml

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

2. Add the request of the ACCESS_COARSE_LOCATION permission to DeviceScanActivity.java

For details on how to add processing, please refer to the following website etc.

Requesting Permissions at Run Time

https://developer.android.com/training/permissions/requesting.html

ACCESS_COARSE_LOCATION

https://developer.android.com/reference/android/Manifest.permission.html#ACCESS_COARSE_LOCATION

■ Exclude Runtime Permission (Not recommended)

In Android 6.0 and later, to run the BluetoothLeGatt sample application without supporting the Runtime Permission, indicate the "targetSdkVersion" value in the build.gradle file to be less than Android 6.0, exclude the Runtime Permission introduced from Android 6.0

```
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"

    defaultConfig {
        minSdkVersion 18
        targetSdkVersion 25
    }
    ….
```

Change the “targetSdkVersion” value from 25 to 21.

6.1.2 Search for BLE devices providing BLE Virtual UART service

By filtering the BLE Virtual UART service UUID from the advertising data, it is possible to display only BLE devices providing the BLE Virtual UART function on the scanning screen.

Add the following declaration to the build.gradle file.

```
dependencies {
    ….
    compile 'com.google.guava:guava:20.0'   Add
}
```

This declaration is the use of Google Guava library. For the Google Guava, please see below.

https://github.com/google/guava

In this sample source code, Bytes#indexOf(byte [] array, byte [] target) method in the Google Guava library is used.

https://google.github.io/guava/releases/20.0/api/docs/com/google/common/primitives/Bytes.html#indexOf-byte:A-byte:A-

Add the following source code to DeviceScanActivity.java

```
import com.google.common.primitives.Bytes;

    private final byte[] RENESAS_VIRTUAL_UART_SERVICE = {
            (byte)0x11, // length
            (byte)0x07, // AD Type (Complete List of 128-bit Service Class UUIDs)
            // Renesas Virtual UART Service UUID
            (byte)0x1b, (byte)0xc5, (byte)0xd5, (byte)0xa5, (byte)0x02, (byte)0x00,
            (byte)0xb8, (byte)0x8c, (byte)0xe5, (byte)0x11, (byte)0x1b, (byte)0xa2,
            (byte)0x01, (byte)0x00, (byte)0x8c, (byte)0xd6,
    };

    // Device scan callback.
    private BluetoothAdapter.LeScanCallback mLeScanCallback = new BluetoothAdapter.LeScanCallback() {
        @Override
        public void onLeScan(final BluetoothDevice device, int rssi, final byte[] scanRecord) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if (Bytes.indexOf(scanRecord, RENESAS_VIRTUAL_UART_SERVICE) != -1) {
                        mLeDeviceListAdapter.addDevice(device);
                        mLeDeviceListAdapter.notifyDataSetChanged();
                    }
                }
            });
        }
    ....
```

Add source code

Add source code

Add source code

The onLeScan method is invoked when the Android device receives an advertising data from BLE devices. In the above sample source code, if the UUID of the “Renesas Virtual UART Service” (Table 5-2) is included in the advertising data, the information of the discovered BLE device is registered in the display list.

For details of advertising data, refer to the following documents.

- Bluetooth® low energy Protocol Stack BLE Virtual UART Application (R01AN3130) ‘8.2 Advertising’
- BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part C], 11 ADVERTISING AND SCAN RESPONSE DATA FORMAT

6.2 Discovering Services and Characteristics

Add the following source code to SampleGattAttributes.java

```
static {
    ....
    // Renesas custom services
    attributes.put("d68c0001-a21b-11e5-8cb8-0002a5d5c51b", "Renesas Virtual UART Service");
    attributes.put("d68c0002-a21b-11e5-8cb8-0002a5d5c51b", "Indication Characteristic");
    attributes.put("d68c0003-a21b-11e5-8cb8-0002a5d5c51b", "Write Characteristic");
}
```

Add source code

Although it is not necessary, operation on the “Service and Characteristic list” screen becomes easy. (Figure 6-2)

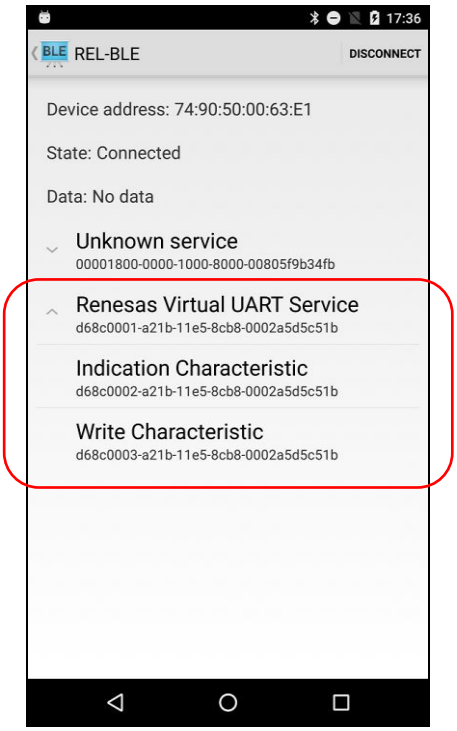

Figure 6-2 Service and Characteristic list (BluetoothLeGatt sample application)

6.3 Characteristic operation

Add the following source code to BluetoothLeService.java. These methods are invoked from the source code to add to DeviceControlActivity.java.

```
/**
 * Requests a write on a given {@code BluetoothGattCharacteristic}.
 *
 * @param characteristic The characteristic to write from.
 */
public void writeCharacteristic(BluetoothGattCharacteristic characteristic) {
    if (mBluetoothAdapter == null || mBluetoothGatt == null) {
        Log.w(TAG, "BluetoothAdapter not initialized");
        return;
    }
    mBluetoothGatt.writeCharacteristic(characteristic);
}

/**
 * Requests a write on a given {@code BluetoothGattDescriptor}.
 *
 * @param descriptor The descriptor to write to.
 */
public void writeDescriptor(BluetoothGattDescriptor descriptor) {
    if (mBluetoothAdapter == null || mBluetoothGatt == null) {
        Log.w(TAG, "BluetoothAdapter not initialized");
        return;
    }
    mBluetoothGatt.writeDescriptor(descriptor);
}
```

Add the following source code to DeviceControlActivity.java The explanation of the source code will be described later.

```
private static final String INDICATION_CHARACTERISTIC = "d68c0002-a21b-11e5-8cb8-0002a5d5c51b";
private static final String WRITE_CHARACTERISTIC      = "d68c0003-a21b-11e5-8cb8-0002a5d5c51b";

private StringBuilder mBuff = new StringBuilder();

private enum NotificationType {
        NOTIFY, INDICAT
}

private boolean setGattNotification(NotificationType type, boolean enable) {
    BluetoothGattDescriptor descriptor = mNotifyCharacteristic.getDescriptor(
            UUID.fromString(SampleGattAttributes.CLIENT_CHARACTERISTIC_CONFIG));
    if (descriptor != null) {
        if (enable) {
            if (type == NotificationType.NOTIFY) {
                descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
            } else {
                descriptor.setValue(BluetoothGattDescriptor.ENABLE_INDICATION_VALUE);
            }
        } else {
            descriptor.setValue(BluetoothGattDescriptor.DISABLE_NOTIFICATION_VALUE);
        }
        mBluetoothLeService.writeDescriptor(descriptor);
    } else {
        Log.w(TAG, "Couldn't get CLIENT_CHARACTERISTIC_CONFIG.");
        return false;
    }
    return true;
}

private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        ....
        } else if (BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action)) {
            displayData(intent.getStringExtra(BluetoothLeService.EXTRA_DATA));

            String value = intent.getStringExtra(BluetoothLeService.EXTRA_DATA);
            mBuff.append(value.substring(0, value.lastIndexOf('¥n')));
            int index = mBuff.toString().indexOf('¥n');
            if (index != -1) {
                String line = mBuff.toString().substring(0, index);
                mBuff = mBuff.delete(0, index + 1);
                Toast.makeText(DeviceControlActivity.this, line, Toast.LENGTH_SHORT).show();
            }
        }
    }
};

private final ExpandableListView.OnChildClickListener servicesListClickListner =
    new ExpandableListView.OnChildClickListener() {
        @Override
        public boolean onChildClick(ExpandableListView parent, View v, int groupPosition,
                                    int childPosition, long id) {
                ....
```

Add source code

Add source code

```
            //if ((charaProp | BluetoothGattCharacteristic.PROPERTY_NOTIFY) > 0) {
              if ((charaProp & (BluetoothGattCharacteristic.PROPERTY_INDICATE |
                                BluetoothGattCharacteristic.PROPERTY_NOTIFY)) > 0){
                  mNotifyCharacteristic = characteristic;
                  mBluetoothLeService.setCharacteristicNotification(characteristic, true);
              }

              if (characteristic.getUuid().toString().equalsIgnoreCase(INDICATION_CHARACTERISTIC)) {
                  setGattNotification(NotificationType.INDICATE, true);
              }

              if (characteristic.getUuid().toString().equalsIgnoreCase(WRITE_CHARACTERISTIC)) {
                  final EditText editText = new EditText(DeviceControlActivity.this);
                  new AlertDialog.Builder(DeviceControlActivity.this)
                          .setTitle("Enter the data to send")
                          .setView(editText)
                          .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                              public void onClick(DialogInterface dialog, int whichButton) {
                                  characteristic.setValue(editText.getText().toString());
                                  mBluetoothLeService.writeCharacteristic(characteristic);
                              }
                          }).show();
              }
              ....
```

Modify source code

Add source code

6.3.1 Sending data from RL78/G1D to Android

Enter the ESC key on the terminal software and change to the "Virtual UART Mode". (Figure 6-3)

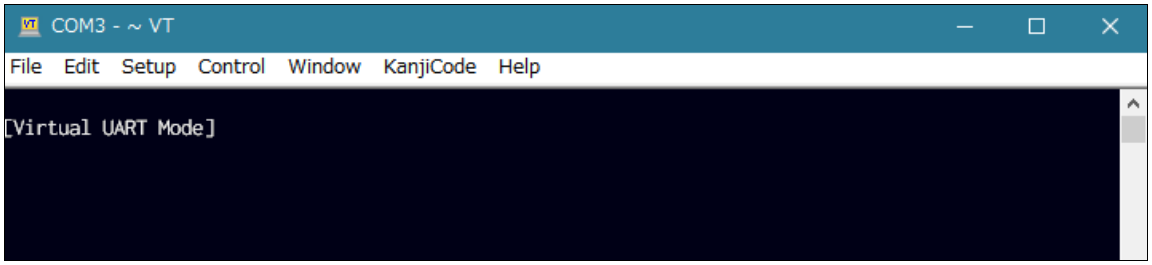

Figure 6-3 Virtual UART Mode

After the character string "[Virtual UART Mode]" is displayed in the terminal software, tap the "Indication Characteristic" item in the “Service and Characteristic list” screen of the BluetoothLeGatt sample application. (Figure 6-4)

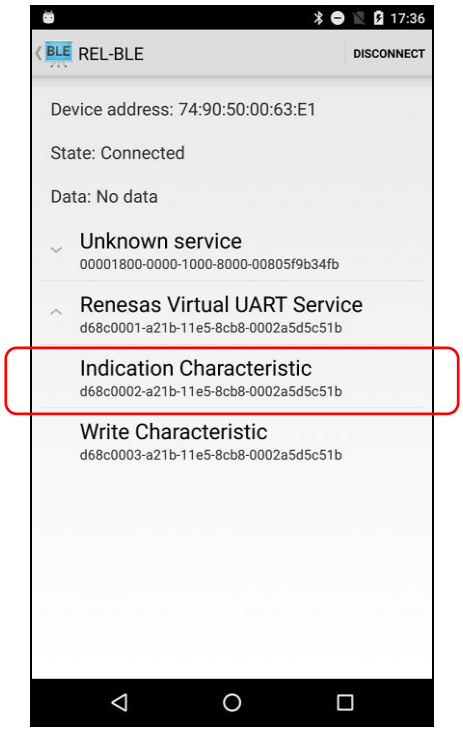

Figure 6-4 Service and Characteristic list (BluetoothLeGatt sample application)

By tapping the "Indication Characteristic" item, the character string "CONNECT" will be displayed in the terminal software. When the character string "Renesas-BLE" is entered in the terminal software (Enter the Enter key at the end), the character string "Renesas-BLE" will be sent to the Android application from the RL78/G1D. (Figure 6-5)

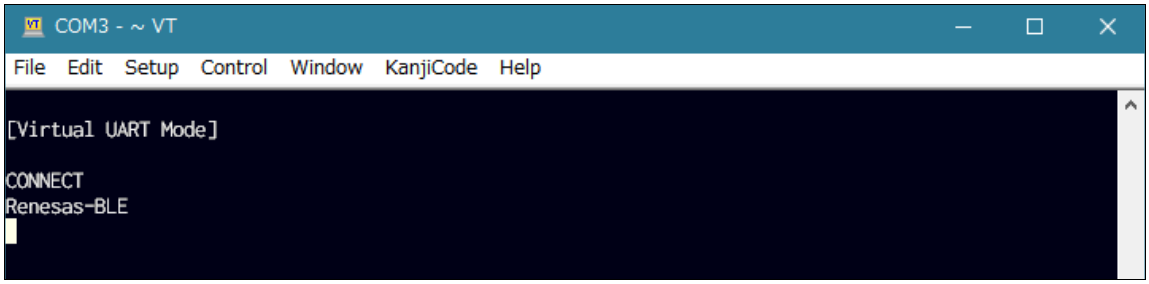

Figure 6-5 Sending the data to Android (Terminal software)

Figure 6-6 is the screen where the Android application received the character string "Renesas-BLE" sent by the RL78/G1D.

When entering characters in the terminal software, the input characters are notified to the Android application using the Indication of Characteristic for one character or every few characters. In the Android application, the data received by the Indication is displayed in the "Data:" part. If the received data contains a line feed code, the contents of the data received so far are displayed by the Toast.

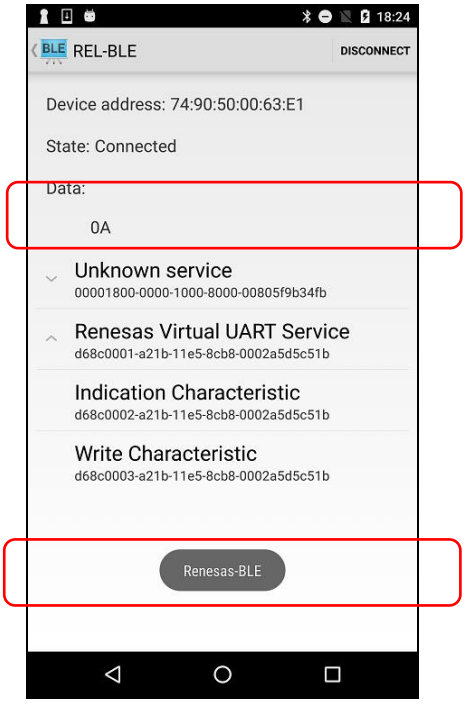

Figure 6-6 Receiving the data from RL78/G1D (BluetoothLeGatt sample application)

Below is a description of the source code.

Sending data from RL78/G1D to Android is executed using the Indication of the "Indication Characteristic". The following source code performs to enable the Indication of the “Indication Characteristic”.

```
//if ((charaProp | BluetoothGattCharacteristic.PROPERTY_NOTIFY) > 0) {
if ((charaProp & (BluetoothGattCharacteristic.PROPERTY_INDICATE |
                  BluetoothGattCharacteristic.PROPERTY_NOTIFY)) > 0){
    mNotifyCharacteristic = characteristic;
    mBluetoothLeService.setCharacteristicNotification(characteristic, true);
}

if (characteristic.getUuid().toString().equalsIgnoreCase(INDICATION_CHARACTERISTIC)) {
    setGattNotification(NotificationType.INDICATE, true);
}
```

To enable the Indication of the "Indication Characteristic", follow the procedure below.

1. Enable the Indication of the "Indication Characteristic" using BluetoothLeService#setCharacteristicNotification()

2. Acquire the Characteristic Descriptor corresponding to the attribute type «Client Characteristic Configuration» (0x2902) of the "Indication Characteristic" (Table 6-1)

3. Write the Configuration Value Indication (0x0002) to the acquired Characteristic Descriptor (Table 6-2)

Note: 2. and 3. is executed with the setGattNotification method. For the implementation of the setGattNotification method, see the source code listed in 6.3

Table 6-1 Client Characteristic Configuration declaration

Attribute Type	UUID	Description
«Client Characteristic Configuration»	0x2902	Client Characteristic Configuration Descriptor

(BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part G] page 543)

Table 6-2 Client Characteristic Configuration bit field definition

Configuration	Value	Description
Notification	0x0001	The Characteristic Value shall be notified. This value can only be set if the characteristic's property has the notify bit set.
Indication	0x0002	The Characteristic Value shall be indicated. This value can only be set if the characteristic's property has the indicate bit set.

(BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part G] page 537)

Once these processes have been completed, the Android application can receive the Indication from RL78/G1D.

For detailed specifications for enabling the Indication, refer to the following document.

BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part G], 3.3.3.3 Client Characteristic Configuration

To get the characteristic value notified by the Indication, follow the procedure below.

When RL78/G1D updates a value of the "Indication Characteristic" (it equals to send the data from RL78/G1D to Android), the following method defined in BluetoothLeService.java is invoked.

```
@Override
public void onCharacteristicChanged(BluetoothGatt gatt,
                                    BluetoothGattCharacteristic characteristic) {
    broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic);
}
```

Invoking the broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic) method invokes the onReceive method defined in DeviceControlActivity.java The onReceive method processes the data received from RL78/G1D.

```
private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        ....
        } else if (BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action)) {
            displayData(intent.getStringExtra(BluetoothLeService.EXTRA_DATA));
        }
    }
};
```

In the above source code, a value of the data received by the onReceive method is displayed every reception, but in an actual application, processing such as buffering the received data is necessary depending on the purpose. The following source code is an example of buffering the received data and displaying the received data using the Toast when receiving a line feed code.

```
private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        ....
        } else if (BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action)) {
            displayData(intent.getStringExtra(BluetoothLeService.EXTRA_DATA));

            String value = intent.getStringExtra(BluetoothLeService.EXTRA_DATA);
            mBuff.append(value.substring(0, value.lastIndexOf('¥n')));
            int index = mBuff.toString().indexOf('¥n');
            if (index != -1) {
                String line = mBuff.toString().substring(0, index);
                mBuff = mBuff.delete(0, index + 1);
                Toast.makeText(DeviceControlActivity.this, line, Toast.LENGTH_SHORT).show();
            }
        }
    }
};
```

The data received with the onReceive method is broadcasted from the following source code in BluetoothLeService.java

Note: "Line feed code + hexadecimal binary string of received data" is added to the end of the data. The format of the data to be broadcasted needs to be changed depending on the purpose of the application.

```
private void broadcastUpdate(final String action,
                             final BluetoothGattCharacteristic characteristic) {
    ....
    } else {
        // For all other profiles, writes the data formatted in HEX.
        final byte[] data = characteristic.getValue();
        if (data != null && data.length > 0) {
            final StringBuilder stringBuilder = new StringBuilder(data.length);
            for(byte byteChar : data)
                stringBuilder.append(String.format("%02X ", byteChar));
            intent.putExtra(EXTRA_DATA, new String(data) + "¥n" + stringBuilder.toString());
        }
    }
    sendBroadcast(intent);
}
```

6.3.2 Sending data from Android to RL78/G1D

Tap the "Write Characteristic" item in the “Service and Characteristic list” screen of the BluetoothLeGatt sample application. (Figure 6-7)

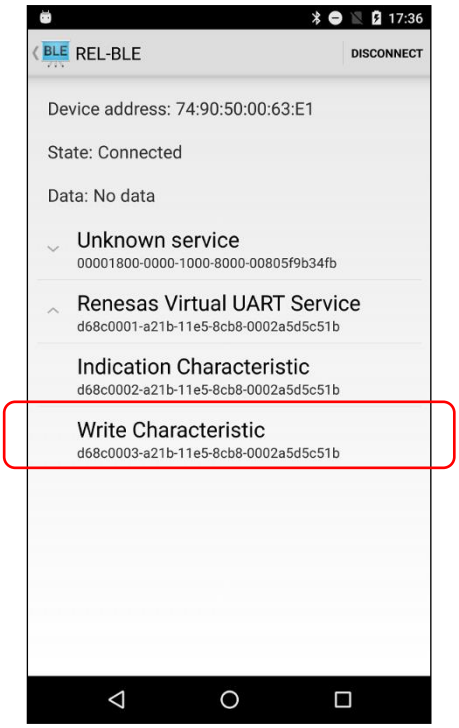

Figure 6-7 Service and Characteristic list (BluetoothLeGatt sample application)

By tapping the “Write Characteristic” item, the sending data input dialog will be displayed. Enter the sending data and tap the "OK" button. (Figure 6-8).

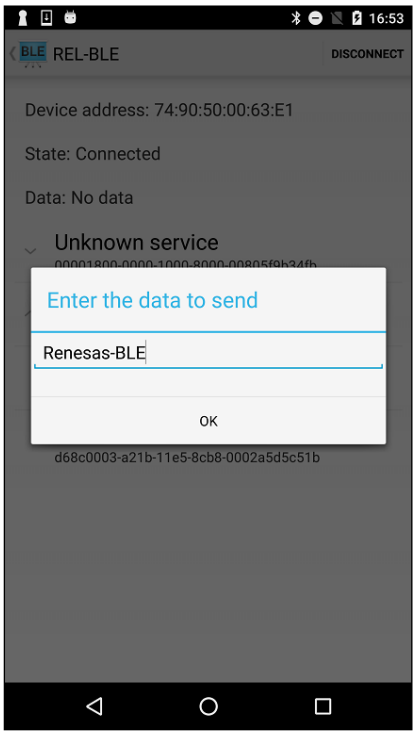

Figure 6-8 Entering the sending data (BluetoothLeGatt sample application)

When the "OK" button in the sending data input dialog is tapped, the input "Renesas-BLE" character string is sent to the RL78/G1D, and the result received at the RL78/G1D is output to the terminal software. (Figure 6-9)

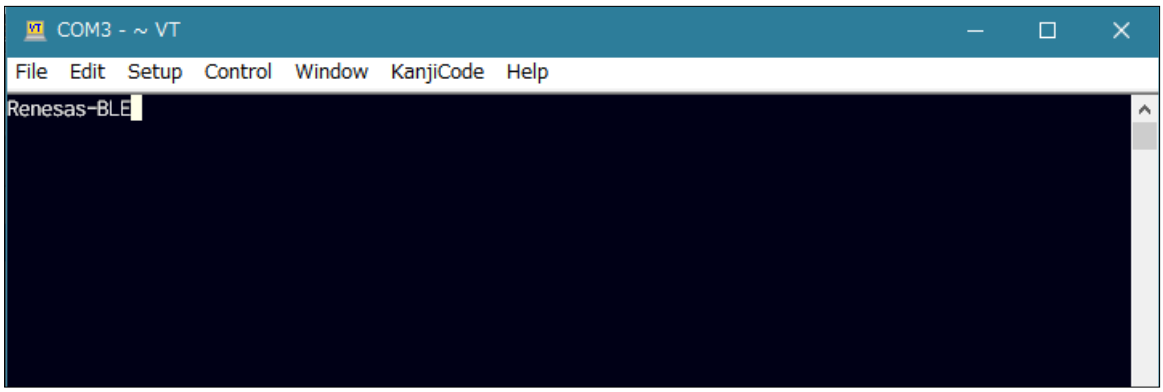

Figure 6-9 Receiving the data from Android (Terminal software)

Sending data is performed in the following source code.

```
final EditText editText = new EditText(DeviceControlActivity.this);
    new AlertDialog.Builder(DeviceControlActivity.this)
            .setTitle("Enter the data to send")
            .setView(editText)
            .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int whichButton) {
                    characteristic.setValue(editText.getText().toString());
                    mBluetoothLeService.writeCharacteristic(characteristic);
                }
            }).show();
```

Write the sending data to the “Write Characteristic” using BluetoothLeService#writeCharacteristic().

In the above source code, a dialog (AlertDialog) to enter a sending data is displayed. But in an actual application, it is necessary to prepare sending data according to the purpose. Also, since the maximum data length is 20 bytes, to send data exceeding 20 bytes, it is necessary to divide the sending data and write it to the "Write Characteristic". (Refer to Table 5-1 Virtual UART Profile specification)

Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr 27, 2017	-	First edition issued.
1.01	Jun 30, 2017	16	Add the source code of the methods that needs to be added to BluetoothLeService.java.

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

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

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

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

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

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

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

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

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

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

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

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

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

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0