BLE 仮想 UART アプリケーション
Android プログラミングガイド

R01AN3842JJ0101
Rev.1.01
2017.06.30

要旨

本ドキュメントは、RL78/G1D の BLE 仮想 UART 機能を使用して、PC（ターミナルソフト）と Android 端末でデータ通信を行う Android アプリケーションのプログラミングの方法について記載します。

対象デバイス

- RL78/G1D 評価ボード（RTK0EN0001D01001BZ）
- Android 端末（5.0 以降）

関連資料

資料名	資料番号	
	和文	英文
Bluetooth® Low Energy プロトコルスタック		
BLE 仮想 UART アプリケーション	R01AN3130J	R01AN3130E
GATTBrowser for Android		
スマートフォンアプリ取扱説明書	R01AN3802J	R01AN3802E

目次

1. 要旨

本ドキュメントは、RL78/G1D の BLE 仮想 UART 機能を使用して、PC（ターミナルソフト）と Android 端末でデータ通信を行う Android アプリケーションのプログラミングの方法について記載します。

全体の構成を図 1-1 に示します。RL78/G1D 評価ボードを USB ケーブルで PC に接続します。ユーザは、PC 上に起動したターミナルソフト経由で RL78/G1D を操作します。RL78/G1D は、Android 端末と BLE 通信を使用してデータのやり取りを行います。

PC（ターミナルソフト）から見ると、RL78/G1D を通して仮想的な UART 接続で Android 端末とデータ通信を行います。

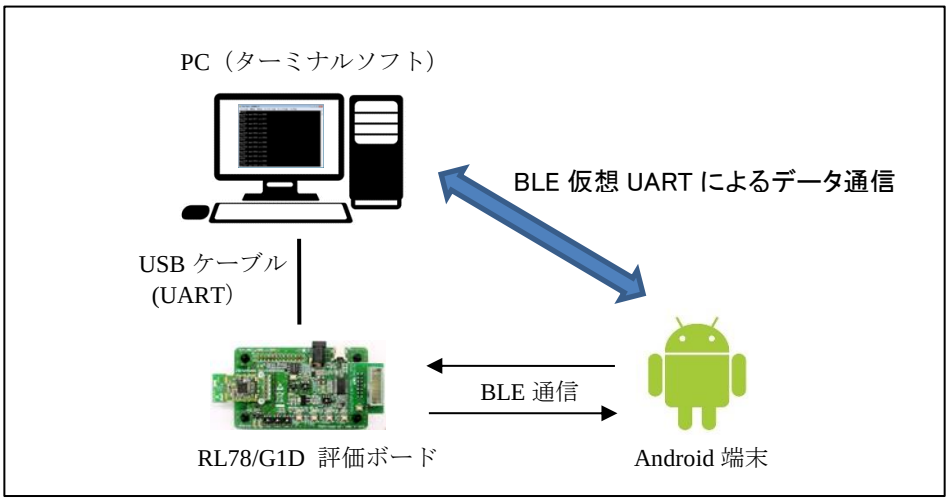

図 1-1 全体構成図

注意: 本ドキュメントに記載されているソースコードは、実際の動作を保証するものではなくプログラミングの方法を説明するためのものです。また、これらを用いた Android 用のアプリケーションは提供していません。

2. 評価環境

- Windows®7 Service Pack1 以降
- Android 端末（5.0 以降）
- RL78/G1D 評価ボード（RTK0EN0001D01001BZ）

  https://www.renesas.com/products/software-tools/boards-and-kits/evaluation-demo-solution-boards/rtk0en0001d01001bz.html
- E1 エミュレータ（ファームウェアの書き込みに使用）

  https://www.renesas.com/products/software-tools/tools/emulator/e1.html
- Renesas Flash Programmer（ファームウェアの書き込みに使用）

  https://www.renesas.com/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html
- Bluetooth® low energy プロトコルスタック BLE 仮想 UART アプリケーション(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 変換デバイスドライバ（※）

※ 評価ボードと PC を接続する際に、UART-USB 変換 IC「FT232RL」のデバイスドライバを要求される場合があります。その場合はデバイスドライバを下記から入手してください。

FTDI (Future Technology Devices International) - Drivers

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

3. 参考資料

- 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. GATTBrowser を使用した接続確認

4.1 RL78/G1D の準備

下記の Web サイトより BLE 仮想 UART アプリケーション(R01AN3130)を取得します。

Bluetooth® low energy プロトコルスタック BLE 仮想 UART アプリケーション(R01AN3130)

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

zip ファイルを展開して、下記の hex ファイルを RL78/G1D に書き込みます。

ROM_File/cc_rl/RL78_G1D_CCE(VUART).hex

PC でターミナルソフトを起動し、表 4-1 に従ってターミナルソフトの設定を行います。本ドキュメントでは、ターミナルソフトは Tera Term を使用しています。

表 4-1 ターミナルソフトの設定値

設定項目	設定値
改行コード(受信)	LF
改行コード(送信)	CR
ボー・レート	4800 [bps]
データ長	8 [bit]
パリティ	none
ストップビット	1 [bit]
フロー制御	none

図 4-1 は、Tera Term の設定画面です。

- 「Setup」メニューから「Serial Port...」を選択して、「ボー・レート」、「データ長」、「パリティ」、「ストップビット」、「フロー制御」を設定します
- 「Setup」メニューから「Terminal...」を選択して、「改行コード(受信)」と「改行コード(送信)」を設定します

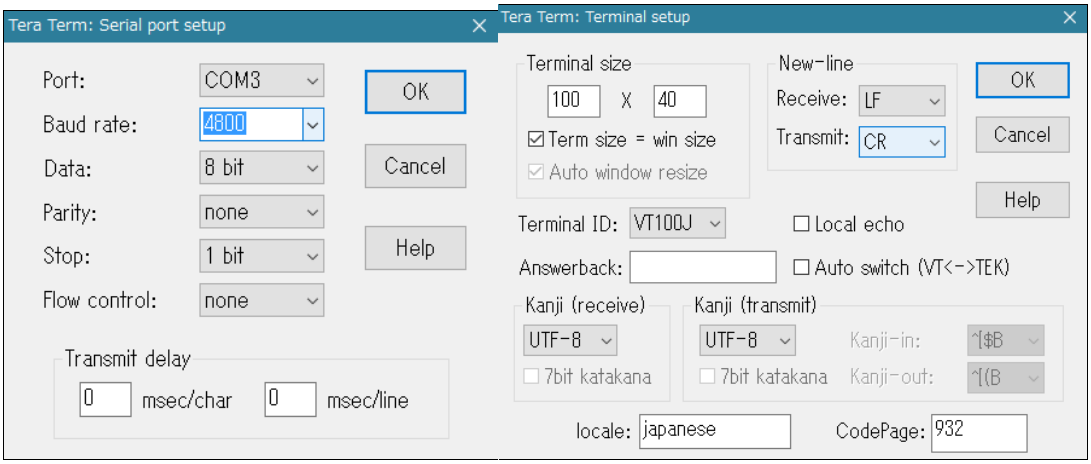

図 4-1 ターミナルソフト(Tera Term)の設定

RL78/G1D とターミナルソフトの接続が完了したら、ターミナルソフト上で ESC キーを入力して RL78/G1D に「Virtual UART Mode」への遷移を指示します。

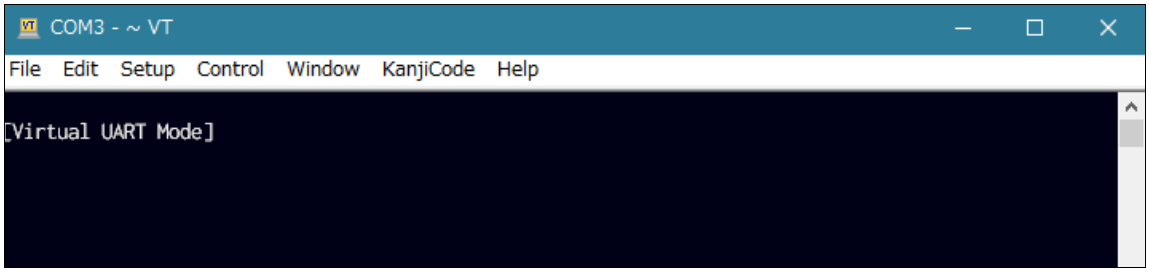

図　4-2 Virtual UART Mode (ターミナルソフト)

参考: BLE 仮想 UART アプリケーション(R01AN3130) 「7.3 実行環境の準備」

4.2　GATTBrowser の準備

GATTBrowser は、周辺で動作する BLE デバイスをスキャンし、それらのデバイスと接続を行って GATT ベースの通信を行うことのできる Android 向け汎用アプリケーションです。本章では、GATTBrowser を使用して、RL78/G1D の BLE 仮想 UART 機能の動作内容を確認します。

GATTBrowser for Android

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

GATTBrowser を起動して、RL78/G1D のアドバタイジングを受信します（図 4-3）。

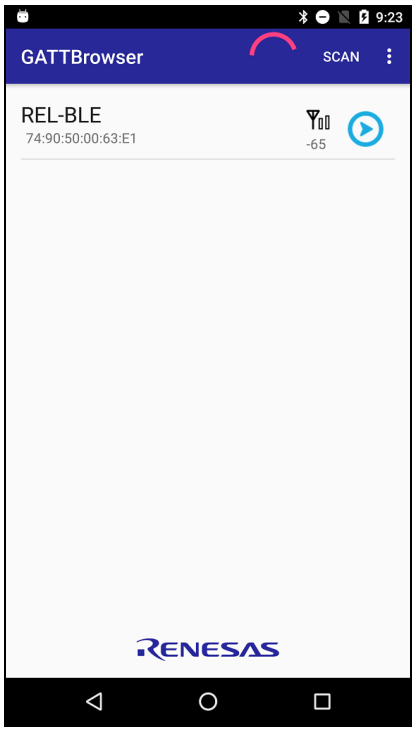

図 4-3　スキャニング (GATTBrowser)

RL78/G1D のアドバタイジングを受信したら、RL78/G1D に接続します。接続に成功すると、RL78/G1D の提供するサービスとキャラクタリスティックの一覧画面が表示されます（図 4-4）。

サービスとキャラクタリスティックの一覧画面が表示されたら、「Renesas Virtual UART Service」の「Indication Characteristic」をタップして、「Indication Characteristic」の操作画面に移動します。

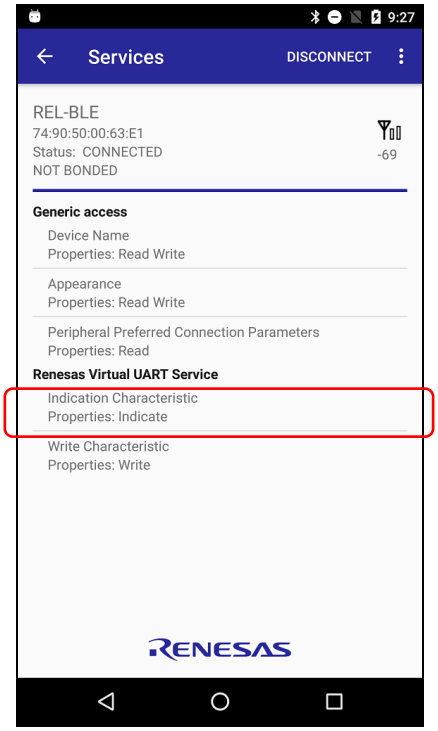

図 4-4　サービスとキャラクタリスティックの一覧画面 (GATTBrowser)

「Indication Characteristic」の操作画面に移動したら、「Indication Off」トグルボタンをタップして「Indication On」状態にします（図 4-5）。

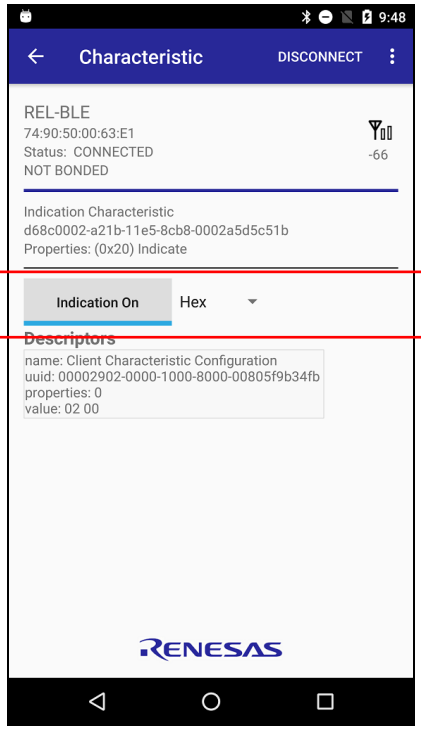

図 4-5　キャラクタリスティックの操作画面 (GATTBrowser)

「Indication Off」トグルボタンをタップして「Indication On」状態にすると、ターミナルソフトに”CONNECT”の文字列が表示されます（図 4-6）。

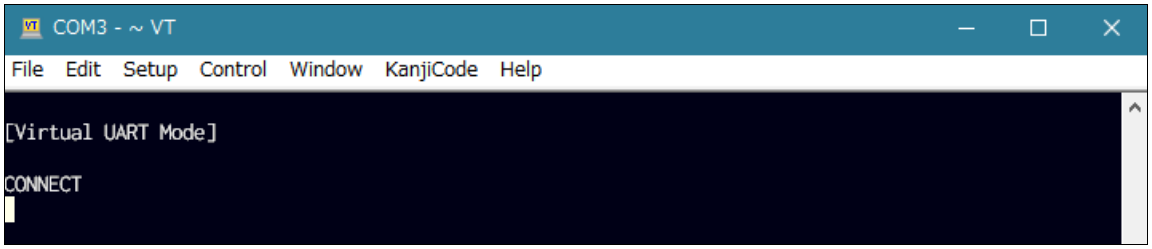

図 4-6　Indication 有効化の確認 (ターミナルソフト)

ここまでの作業で、RL78/G1D と GATTBrowser でデータのやり取りを行う準備が完了しました。

4.3　RL78/G1D から GATTBrowser へのデータ送信

RL78/G1D から GATTBrowser にデータを送信します。

ターミナルソフトで、"abc"を入力します（図 4-7）。

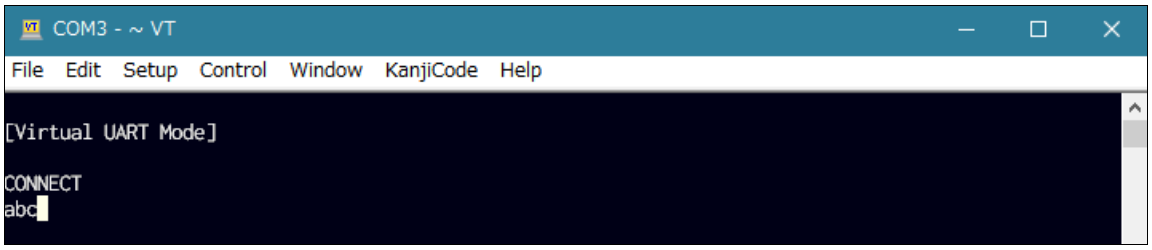

図 4-7　Android にデータを送信 (ターミナルソフト)

ターミナルソフトで入力された"abc"は、RL78/G1D の BLE 仮想 UART 機能で GATTBrowser に送信されます。図 4-8 は GATTBrowser で"abc"を受信した画面です。

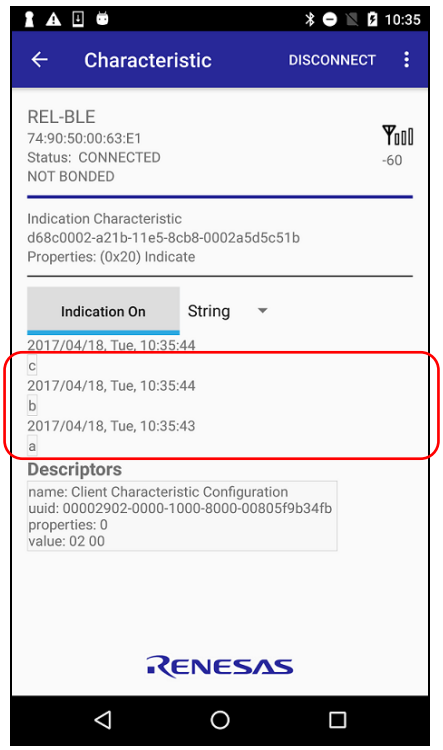

図 4-8　RL78/G1D からデータを受信 (GATTBrowser)

4.4　GATTBrowser から RL78/G1D へのデータ送信

GATTBrowser から RL78/G1D にデータを送信します。

GATTBrowser で「戻る」ボタンをタップして、「Indication Characteristic」の操作画面から、サービスとキャラクタリスティックの一覧画面に戻ります（図 4-9）。サービスとキャラクタリスティックの一覧画面に戻ったら、「Renesas Virtual UART Service」の「Write Characteristic」をタップして、「Write Characteristic」の操作画面に移動します。

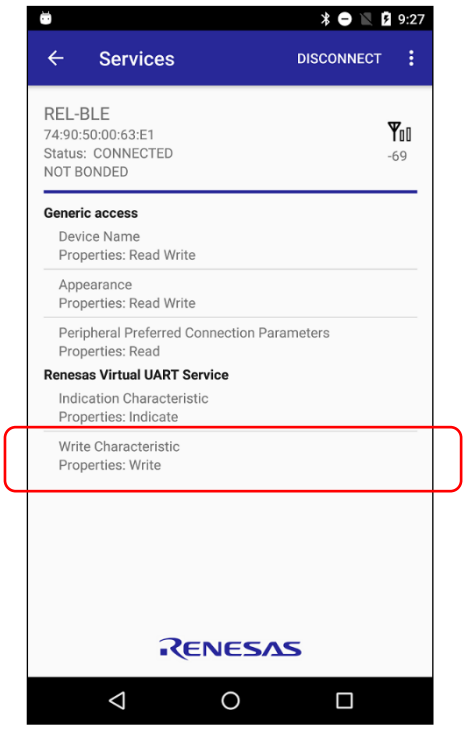

図 4-9　サービスとキャラクタリスティックの一覧画面 (GATTBrowser)

「Write Characteristic」の操作画面に移動したら（図 4-10）、下記の手順を行います。

1. 「Write」ボタンの横の「書き込みモード選択」スピナーで「String」を選択します
2. 「送信テキストィールド」に"Renesas-BLE"と入力します
3. 「Write」ボタンをタップします

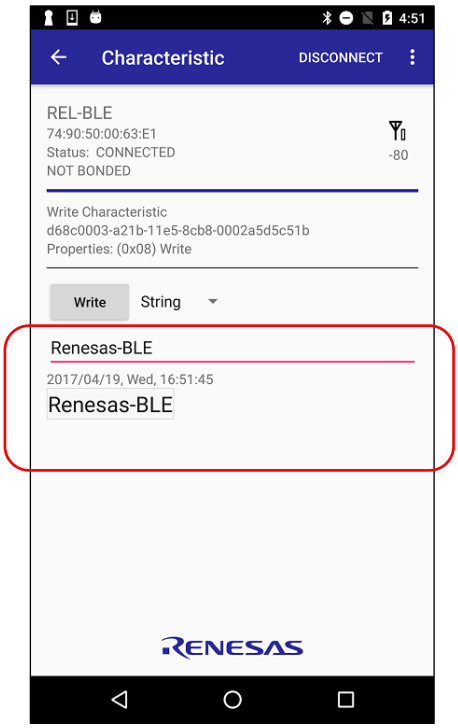

図 4-10　RL78/G1D にデータを送信 (GATTBrowser)

「Write」ボタンをタップすると、GATTBrowser から RL78/G1D へ、BLE 仮想 UART 機能を使用してデータが送信されます。図 4-11 は、RL78/G1D が GATTBrowser から受信したデータ（”Renesas-BLE”）をターミナルソフトに出力した画面です。

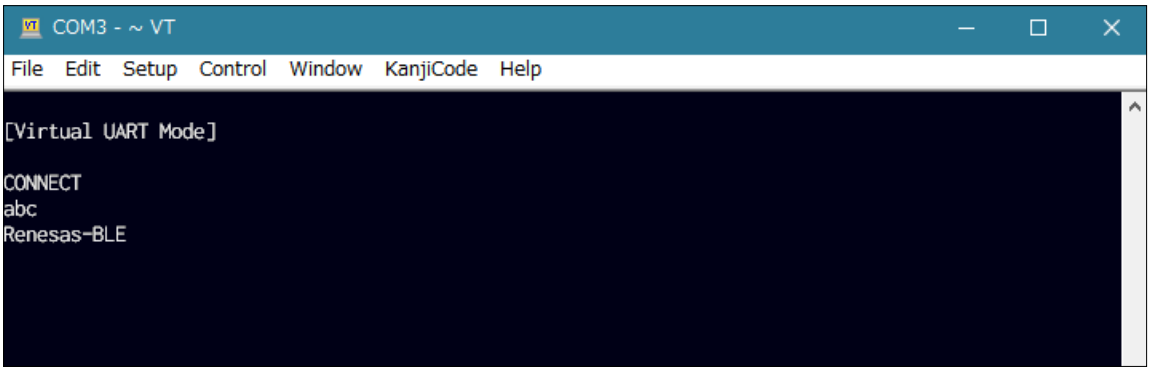

図 4-11　Android からデータを受信 (ターミナルソフト)

5. 仮想 UART プロファイル

仮想 UART プロファイルの仕様を表 5-1 に示します。この表は、Bluetooth® low energy プロトコルスタック BLE 仮想 UART アプリケーション(R01AN3130)「8.1 仮想 UART プロファイル」からの抜粋です。

表 5-1　仮想 UART プロファイルの仕様

アトリビュートハンドル	アトリビュートタイプとその値
VUART_HDL_SVC 0x0022 （※）	Type: Primary Service Declaration UUID: 0xD68C0001-A21B-11E5-8CB8-0002A5D5C51B 仮想 UART サービス
VUART_HDL_INDICATION_CHAR 0x0023 （※）	Type: Characteristic Declaration UUID: 0xD68C0002-A21B-11E5-8CB8-0002A5D5C51B Property: Indicate サーバからクライアントへの文字送信に使用します。
VUART_HDL_INDICATION_VAL 0x0024 （※）	Type: Indication Value 本 Characteristic にデータを設定後、Indication を送信することで、サーバからクライアントへの文字送信を行います。一度に送信可能な文字数は、最大 20 文字です。
VUART_HDL_INDICATION_CFG 0x0025 （※）	Type: Client Characteristic Configuration Descriptor クライアントが、サーバの Indication を許可・不許可を設定するために使用します。
VUART_HDL_WRITE_CHAR 0x0026 （※）	Type: Characteristic Declaration UUID: 0xD68C0003-A21B-11E5-8CB8-0002A5D5C51B Property: Write クライアントからサーバへの文字送信に使用します。
VUART_HDL_WRITE_VAL 0x0027 （※）	Type: Write Value 本 Characteristic に Write Request により文字を書き込むことで、クライアントからサーバへの文字送信を行います。一度に送信可能な文字数は、最大 20 文字です。

※ アトリビュートハンドルの 16 進値は、ファームウェア内に組み込むプロファイルによって変わります

Android 端末から操作する必要のあるサービスとキャラクタリスティックを、表 5-2 と表 5-3 に示します。

表 5-2 Renesas Virtual UART Service

サービス名	UUID
Renesas Virtual UART Service	D68C0001-A21B-11E5-8CB8-0002A5D5C51B

表 5-3 Characteristics

キャラクタリスティック名	UUID	Properties
Indication Characteristic	D68C0002-A21B-11E5-8CB8-0002A5D5C51B	Indication
Write Characteristic	D68C0003-A21B-11E5-8CB8-0002A5D5C51B	Write

5.1　Android から RL78/G1D へデータ送信を行う方法

Android 端末から、「Write Characteristic」に対して Write Request を使用して値を書き込みます。

キャラクタリスティックに値を書き込むには、BluetoothGatt#writeCharacteristic()を使用します。

Write Request:

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

BluetoothGatt#writeCharacteristic():

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

5.2　RL78/G1D から Android へデータ送信を行う方法

Android 端末から、「Indication Characteristic」の Indication を有効にします。

「Indication Characteristic」の Indication が有効の場合、RL78/G1D が「Indication Characteristic」の値を変更すると、Android 端末に「Indication Characteristic」の値が変更された事が通知されるようになります。この仕組みを利用して、RL78/G1D から Android 端末にデータの送信を行います。

実際に「Indication Characteristic」の Indication を有効にする方法は、6.3.1 に記載します。

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.　プログラミングガイド

- 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

本章は、RL78/G1D の BLE 仮想 UART 機能を使用して RL78/G1D とデータ通信を行う Android アプリケーションのプログラミングの方法を記載します。

プログラミングの方法は、上記の Android Developers の Web サイトから提供されている BluetoothLeGatt サンプルアプリケーションと BLE 機能の API ガイドを元に、それらで使用されているソースコードに補足追加する形式で記載します。

6.1　スキャニング

図 6-1 は、BluetoothLeGatt サンプルアプリケーションをビルドして実行した画面です。RL78/G1D のアドバタイジングを受信しています。

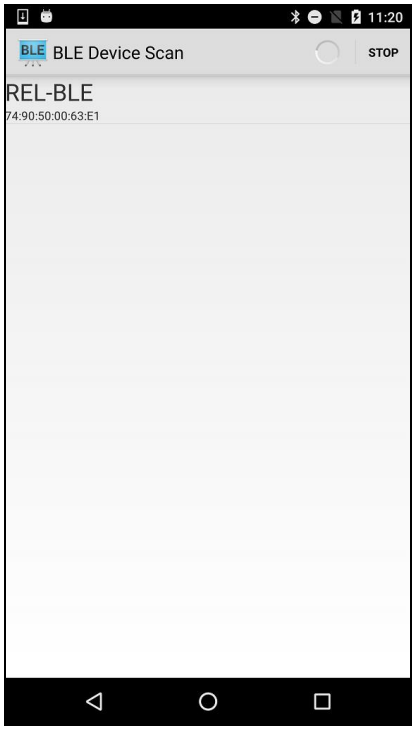

図 6-1　スキャニング (BluetoothLeGatt サンプルアプリ)

Android 6.0 以降で、BluetoothLeGatt サンプルアプリケーションを動作させるには、Android 6.0 から導入された Runtime Permission の対応を行う必要があります。

Requesting Permissions at Run Time

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

Runtime Permission の対応を行わない場合、下記の例外が発生して BLE 機能を使用する事ができません。

```
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　Runtime Permission への対応

■　Runtime Permission の対応処理の追加（推奨）

1. AndroidManifext.xml に下記の内容を追記します

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

2. DeviceScanActivity.java に ACCESS_COARSE_LOCATION パーミッション取得処理を追加します

具体的な処理の追加方法は、下記の Web サイトなどを参照してください。

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

■　Runtime Permission の適用除外（非推奨）

Android 6.0 以降で、Runtime Permission 対応処理を追加せずに BluetoothLeGatt サンプルアプリケーションを動作させるには、build.gradle 中の targetSdkVersion を Android 6.0 未満に指定して、Android 6.0 から導入された Runtime Permission の適用を除外します。

```
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.2"

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

上記の設定ファイル中の targetSdkVersion の値を 25 から 21 に変更します。

6.1.2　BLE 仮想 UART サービスを持つ BLE デバイスの検索

アドバタイジングデータから BLE 仮想 UART サービスの UUID をフィルタリングすることで、スキャニング画面で BLE 仮想 UART 機能を持つ BLE デバイスのみを表示させることが可能です。

build.gradle に下記の記述を追加します。

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

ここでは、Google Guava の使用を宣言しています。Google Guava に関しては、下記を参照してください。

https://github.com/google/guava

本サンプルコードでは、Bytes#indexOf(byte[] array, byte[] target)メソッドを使用しています。

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

DeviceScanActivity.java に下記のコードを追加します。

```
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();
                    }
                }
            });
        }
    ....
```

コードの追加

コードの追加

コードの追加

onLeScan メソッドは、Android 端末が BLE デバイスのアドバタイジングを受信すると呼び出されます。上記のサンプルコードでは、アドバタイジングデータの中に Renesas Virtual UART Service の UUID（表 5-2）が含まれていたら、発見した BLE デバイスの情報を一覧表示リストに登録しています。

アドバタイジングデータの詳細に関しては、下記の資料を参照してください。

- Bluetooth® low energy プロトコルスタック BLE 仮想 UART アプリケーション(R01AN3130) 「8.2 アドバタイジング動作」
- BLUETOOTH SPECIFICATION Version 4.2 [Vol 3, Part C], 11 ADVERTISING AND SCAN RESPONSE DATA FORMAT

6.2　サービスとキャラクタリスティックの検索

SampleGattAttributes.java に下記のコードを追加します。

```
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");
}
```

コードの追加

必ずしも必要な処理ではありませんが、サービスとキャラクタリスティックの一覧表示画面での操作が容易になります（図 6-2）。

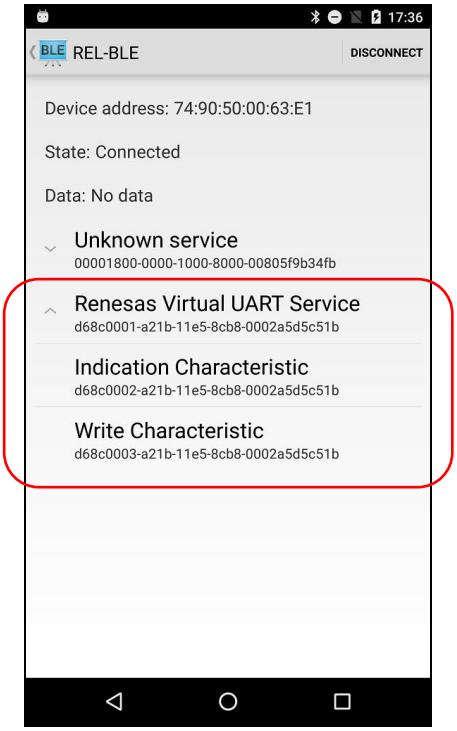

図 6-2　サービスとキャラクタリスティックの一覧画面 (BluetoothLeGatt サンプルアプリ)

6.3　キャラクタリスティックの操作

BluetoothLeService.java に、下記のキャラクタリスティックに値を書き込むメソッドとディスクリプタに値を書き込むメソッドを追加します。これらのメソッドは、DeviceControlActivity.java に追加するソースコードから呼び出されます。

```
/**
 * 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);
}
```

DeviceControlActivity.java に下記のコードを追加します。追加するコードの説明は後述します。

```
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();
                                                                    コードの追加
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));
                                                                    コードの追加
            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) {
                ....
```

```
            //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();
            }
            ....
```

コードの修正

コードの追加

6.3.1　RL78/G1D から Android へのデータ送信

ターミナルソフト上で ESC キーを入力して、「Virtual UART Mode」に遷移します（図 6-3）。

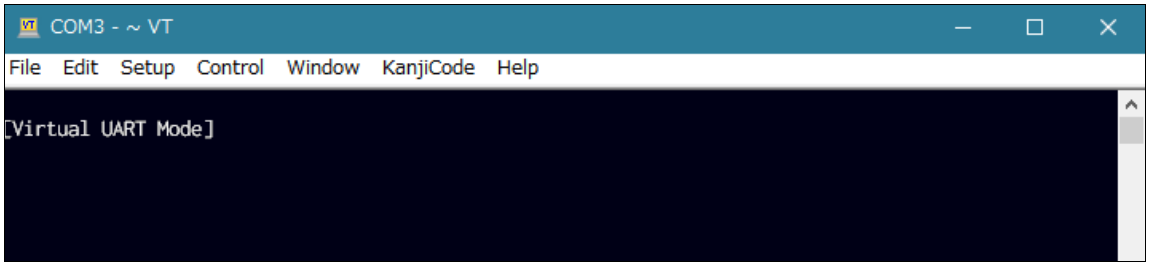

図 6-3 Virtual UART Mode

ターミナルソフトに”[Virtual UART Mode]”の文字列が出力されたのを確認したら、BluetoothLeGatt サンプルアプリで、「Indication Characteristic」の項目をタップします（図 6-4）。

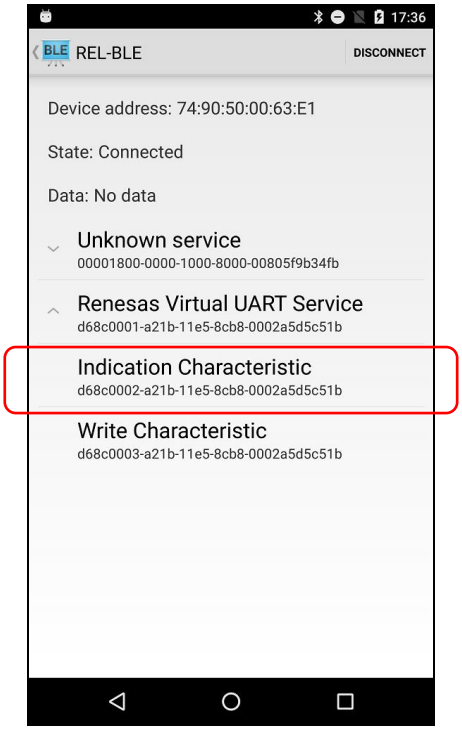

図 6-4　サービスとキャラクタリスティックの一覧画面 (BluetoothLeGatt サンプルアプリ)

「Indication Characteristic」の項目をタップすると、ターミナルソフトに”CONNECT”の文字列が表示されます。”CONNECT”の文字列の表示を確認後、ターミナルソフトに"Renesas-BLE"を入力（最後に Enter キー入力）すると、RL78/G1D から Android アプリに"Renesas-BLE"が送信されます（図 6-5）。

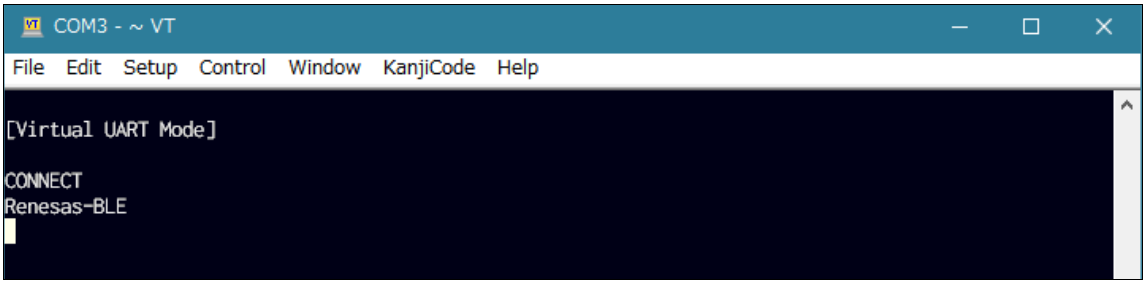

図 6-5　Android にデータを送信 (BluetoothLeGatt サンプルアプリ)

図 6-6 は、RL78/G1D が送信した”Renesas-BLE”を BluetoothLeGatt サンプルアプリ（Android アプリ）が受信した画面です。

ターミナルソフトに文字を入力すると、入力された文字は、一文字もしくは数文字ごとにキャラクタリスティックの Indication を利用して BluetoothLeGatt サンプルアプリに通知されます。BluetoothLeGatt サンプルアプリは、Indication で受信したデータを「Data:」部分に表示します。受信したデータに改行コードが含まれている場合は、それまでに取得したデータの内容を Toast で表示します。

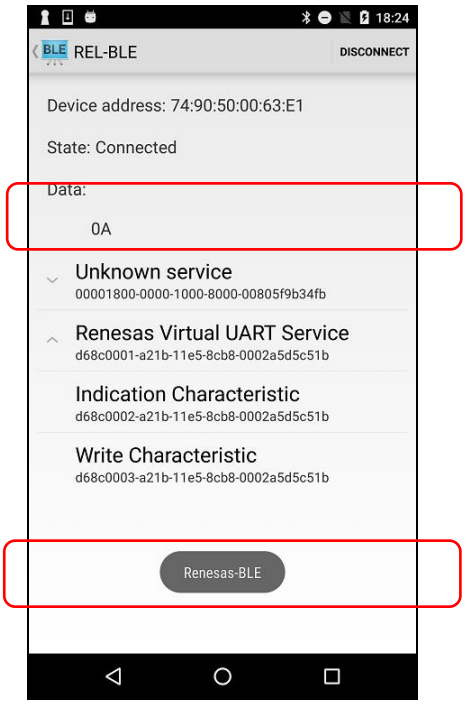

図 6-6　RL78/G1D からデータを受信 (BluetoothLeGatt サンプルアプリ)

以下、ソースコードの説明を記載します。

RL78/G1D から Android アプリへのデータ送信は、「Indication Characteristic」キャラクタリスティックの Indication を利用して行います。BluetoothLeGatt サンプルアプリは、下記のソースコードで Indication を有効にする処理を行っています。

```
//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);
}
```

「Indication Characteristic」の Indication を有効にするには、下記の手順を行います。

1. BluetoothLeService#setCharacteristicNotification()を使用して「Indication Characteristic」の Indication を有効にします

2. 「Indication Characteristic」の«Client Characteristic Configuration»（表 6-1）に対応する Characteristic Descriptor を取得します

3. 取得した Characteristic Descriptor に対して Configuration Value（表 6-2）を書き込みます

※ 2. 3.の処理は、setGattNotification メソッドで行っています。setGattNotification メソッドの実装は、6.3 に記載のソースコードを参照してください。

表 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)

表 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)

これらの処理が完了すると、Android アプリで RL78/G1D の Indication を受信する事が可能になります。

Indication を有効にするための詳細な仕様に関しては、下記の資料を参照してください。

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

Indication で通知されたキャラクタリスティックの値を取得するには、下記の手順を行います。

RL78/G1D が「Indication Characteristic」の値を更新すると（RL78/G1D から Android アプリへのデータ送信）、BluetoothLeService.java で定義されている下記のメソッドが呼び出されます。

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

broadcastUpdate(ACTION_DATA_AVAILABLE, characteristic)の呼び出しによって、DeviceControlActivity.java で定義されている onReceive メソッドが呼び出されます。onReceive メソッドでは、受信したデータの処理を行います。

```
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));
        }
    }
};
```

上記のソースコードでは、onReceive メソッドで受信したデータの値を表示する処理を行っていますが、実際のアプリケーションでは、用途に応じて受信データをバッファリングするなどの処理を行ってください。下記のソースコードは、受信したデータをバッファリングして、改行コードを受信したらこれまでに受信したデータを Toast で表示する例です。

```
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();
            }
        }
    }
};
```

onReceive メソッドで取得するデータは、BluetoothLeService.java の下記のソースコード（※）からブロードキャストされています。

※ データの末尾に「改行コード＋受信データの 16 進数バイナリ文字列」が追加されています。ブロードキャストするデータの形式は、アプリケーションの用途に応じて変更してください。

```
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　Android から RL78/G1D へのデータ送信

BluetoothLeGatt サンプルアプリで、「Write Characteristic」の項目（図 6-7）をタップします。

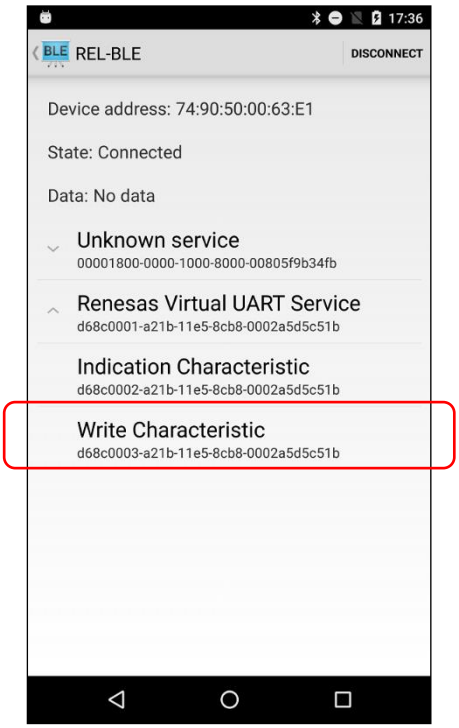

図 6-7　サービスとキャラクタリスティックの一覧画面 (BluetoothLeGatt サンプルアプリ)

「Write Characteristic」の項目をタップすると、送信データ入力ダイアログが表示されます（図 6-8）。送信するデータを入力して「OK」ボタンをタップします。

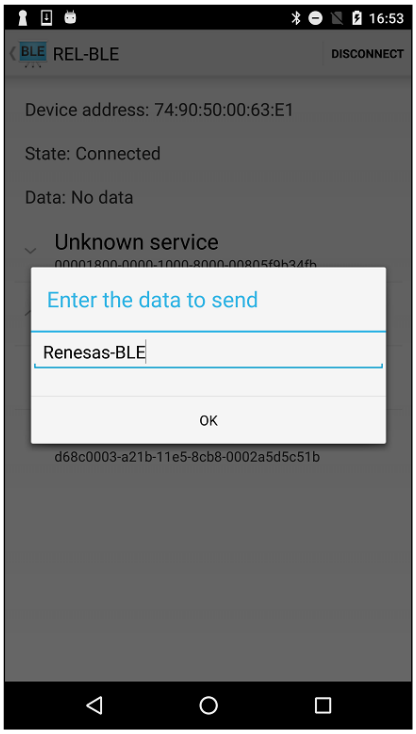

図 6-8　送信データの入力 (BluetoothLeGatt サンプルアプリ)

「OK」ボタンをタップすると、BluetoothLeGatt サンプルアプリに入力した"Renesas BLE"の文字列が RL78/G1D に送信されます。図 6-9 は、RL78/G1D の受信した文字列がターミナルソフトに出力された画面です。

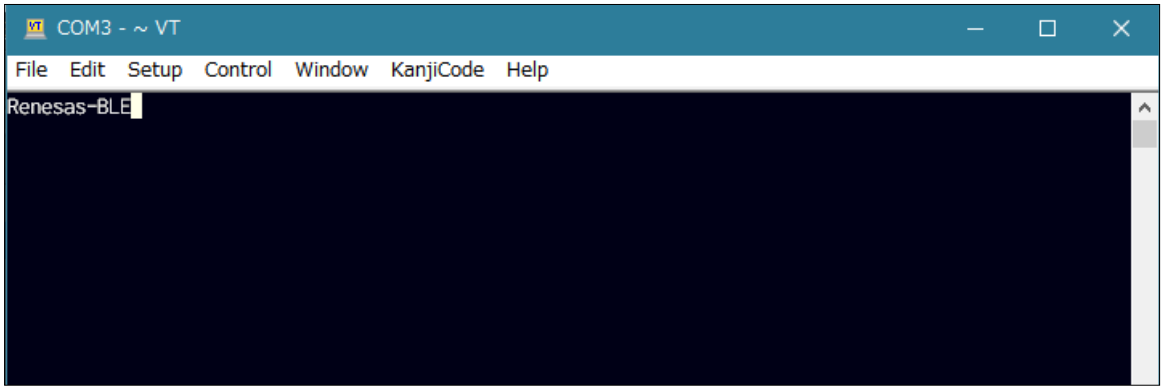

図 6-9　Android からデータを受信 (ターミナルソフト)

下記のソースコードで送信処理を行っています。

```
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();
```

BluetoothLeService#writeCharacteristic()を使用して、送信データを「Write Characteristic」に書き込みます。

上記のソースコードでは、ダイアログ（AlertDialog）を表示して送信データを入力していますが、実際のアプリケーションでは、用途に応じて送信データを用意する必要があります。また、最大データ長は 20 バイトのため、20 バイトを超えるデータを送信するには、送信データを分割して「Write Characteristic」に書き込む必要があります。（表 5-1　仮想 UART プロファイルの仕様　を参照してください。）

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2017/4/27	-	初版発行
1.01	2017/6/30	16	BluetoothLeService.java に追加する必要のあるメソッドのソースコードを追記。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 4.1