REX-USB60MB

USB Serial Converter (Micro-USB B タイプ)

Android 用サンプルプログラム

USB60BCR について

2013 年 09 月

第 1.0 版

ラトックシステム株式会社

1. 本プログラムの概要

本サンプルプログラムは、当社製 USB Serial Converter REX-USB60MB を使って Android タブレット(Nexus7)と RS-232C 接続のバーコードスキャナーを接続して使用する方法を説明するためのサンプルプログラムです。

Android アプリから USB デバイスを使用するには、アプリ側で AndroidOS が提供する USB Host API を直接呼び出すか、USB デバイス用のドライバを呼び出す必要があります。

本サンプルプログラムでは、USB シリアル変換チップ用に FTDI 社製よりクラスドライバとして公開されている D2xx.jar を利用しております。

※1: D2xx.jar は、FTDI 社のサイト(http://www.ftdichipcom/Android.htm)からダウンロードできます。

本サンプルプログラムの機能は、REX-USB60MB に接続されたバーコードリーダーでスキャンされた情報の内容を表示するだけの単純なものです。
あくまで Android 上で USB Serial Converter へのアクセス方法説明するためのもので、ご利用いただく場合、お客様で十分な評価を行っていただきますようお願いします。

本書は、Eclipse を使った Android アプリ用の統合開発環境でのアプリ作成経験者を対象としています。
以降の作業を進めるにあたっては、Eclipse を使った Android アプリ用の統合開発環境および、パソコンの USB ポートから Android タブレットへアプリをダウンロードして実行できる環境を準備をしておいてください。

サンプルプログラムの作成は、以下のステップで行います。
・USB60BCR のダウンロードとプロジェクトのインポート
・FTDI 社製クラスドライバ(D2xx.jar)のダウンロードと jar ファイルのコピー

以降で具体的な操作手順を示します。

2. サンプルプログラムの作成と操作手順

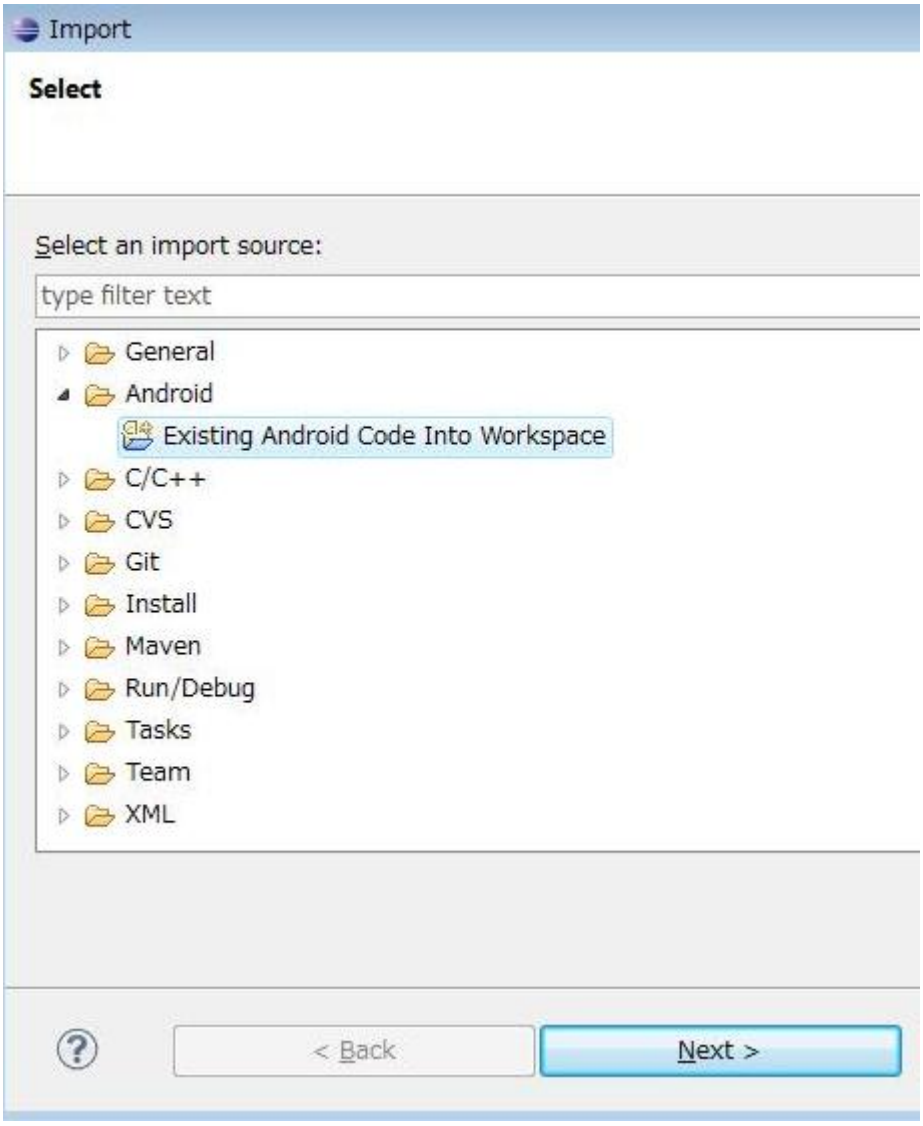
2-1. USB60BCR のダウンロードとプロジェクトのインポート

(1) RATOC e2eStore の「REX-USB60MB」の製品ページを開きます。そして、[ダウンロード]タブをクリックしてダウンロードファイルの一覧画面に切り替えます。一覧リストから「Android OS 対応サンプルプログラム USB60BCR …」をダウンロードします。

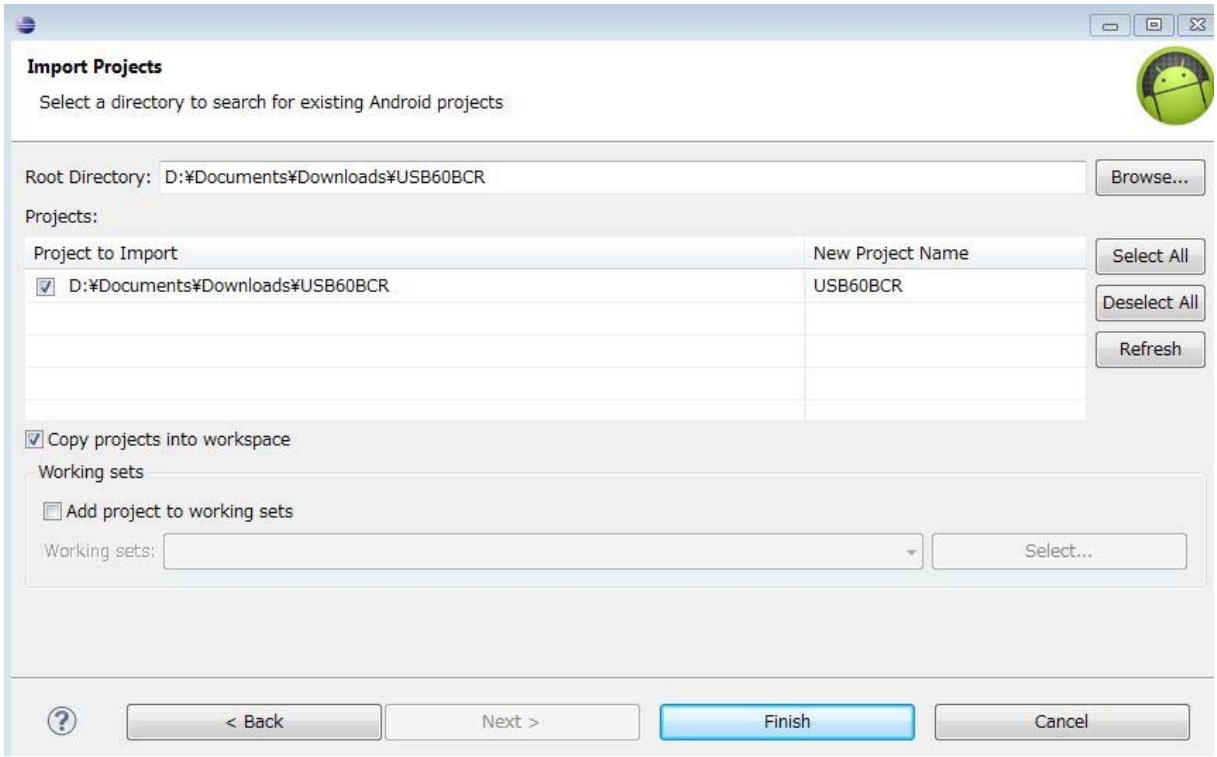
(2) ダウンロードしたファイルを任意のフォルダーで解凍します。

(3) Eclipse で USB60BCR プロジェクトをインポートします。

[File]-[Import]から[Android]の[Existing Android Code Into Workspace]を選択して、[Next]を押します。



「USB60BCR」を選択し、「Copy project to workspace」にチェックをして、右下の[Finish]を押します。

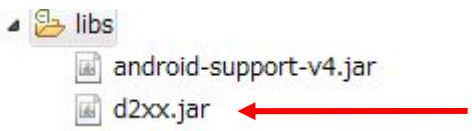


2-2. FTDI 社製クラスドライバ(D2xx.jar)のダウンロードと jar ファイルのコピー

(1)FTDI 社のサイト(http://www.ftdichip.com/Android.htm)
から TN_147_Java_D2xx_for_Android_Demo_Source.zip をダウンロードします。

(2) ダウンロードしたファイルを任意のフォルダーで解凍します。

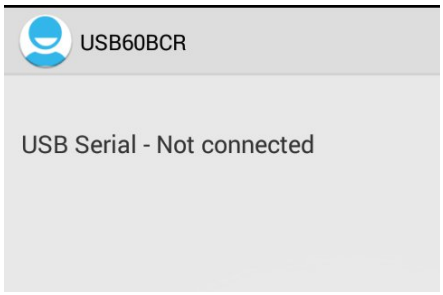
(3) [TN_147_Java_D2xx_for_Android_Demo_Source¥libs]フォルダーにある[D2xx.jar]ファイルを USB60BCR プロジェクトの[libs]フォルダーへコピーします。



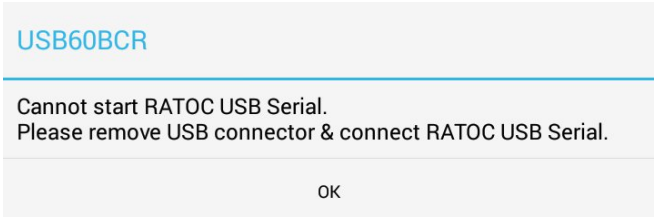
以上でサンプルプログラムを実行する設定はできました。

2-3. Nexus7 への Android アプリのインストールと実行

(1) パソコンからターゲットの Android タブレットを USB ケーブルで接続します。
(2) Eclipse からプログラムがダウンロード可能な状態になったら、Eclipse 上から「USB60BCR」プロジェクトを選択して[Run]をクリックします。
(3) 「Android Device Chooser」の画面から「choose a running Android device」にチェックを入れて、対象の Android タブレットを選択して、「OK」を押します。
(4) Android アプリの実行ファイル(apk)が生成後にダウンロードされてタブレット上でサンプルプログラムが実行されます。
(5) REX-USB60MB が接続されていないので、「USB Serial – Not connected」と表示され



REX-USB60MB が接続されていないことを表わす以下のダイアログが表示されます。

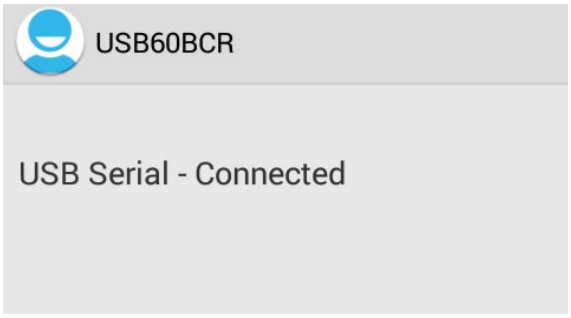


(6) [OK]をクリックした後、Android タブレットの USB ポートに REX-USB60MB を接続します。Nexus7 等で 1 つ USB ポートがデバイス機能とホスト機能共有している端末では、PC と接続している USB ケーブルを取り外した後に、REX-USB60MB を接続してください。

(7) 画面の中央に USB60BCR 実行許可の確認ダイアログが表示されます。

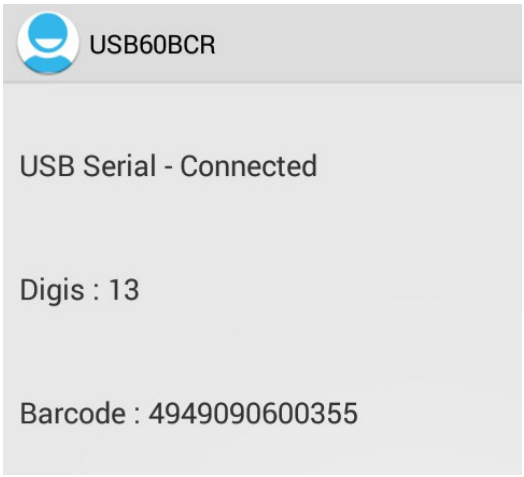
(8) [OK]を押して実行を許可します。

(9) REX-USB60MB を検出すると「USB Serial – Not connected」が「USB Serial - Connected」に変わります。



(10) 次に、バーコードスキャナーを接続し、磁気カードをスキャンします。

(11) バーコードが読み込まれると、読み込んだコードの桁数とその下に読み込んだコードが表示されます。



最後に、REX-USB60MB を Android タブレットの USB ポートから外してください。
「USB Serial – Not connected」に変わります。

以上でサンプルプログラムの動作が確認できました。

3. サンプルプログラムソースの説明

ここから先は、サンプルプログラムのソースを例にポイントとなる箇所を説明していきます。

3-1. マニフェストファイル「AndroidManifest.xml」

AndroidManifest.xml では、対象とする USB デバイスを特定するためのインテントフィルタを設定します。

```
14        <activity
15            android:name="com.ratocsystems.usb60sample.USB60BCR.Main"
16            android:launchMode="singleTask"                              ①
17            android:configChanges="orientation"
18            android:screenOrientation="portrait"
19            android:label="@string/title_activity_main" >
20            <intent-filter>
21                <action android:name="android.intent.action.MAIN" />
22
23                <category android:name="android.intent.category.LAUNCHER" />
24            </intent-filter>
25
26            <intent-filter>
27                <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />   ②
28            </intent-filter>
29
30            <meta-data android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"   ③
31                android:resource="@xml/device_filter" />
32
33        </activity>
```

<Activity>エレメントの中に以下を指定します。

① ***android:launchMode="singleTask"***

Activity 起動方法を singleTask に設定します。

② ***<action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />***

<intent-filter>エレメントで対象とする USB デバイスを特定するためのインテントフィルタを設定します。

③***<meta-data action android:name="android.hardware .... USB_DEVICE_ATTACHED"***
android:resource="@xml/device_filter" />

<meta-data> エレメントには次で説明する XML リソースファイルを指定します。

3-2. XML リソースファイル「device_filter.xml」

[res/xml] フォルダー下の device_filter.xml では、REX-USB60F, REX-USB60MI, REX-USB60MB の 3 製品のベンダーID とプロダクト ID の指定を<resources>エレメントの中に 1 行ずつ記述します。

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3         <usb-device vendor-id ="1412" product-id= "45088" /> <!-- 0x0584,0xB020 REX-USB60F -->
4         <usb-device vendor-id ="1412" product-id= "45103" /> <!-- 0x0584,0xB02F REX-USB60MI -->
5         <usb-device vendor-id ="1412" product-id= "45115" /> <!-- 0x0584,0xB03B REX-USB60MB -->
6 </resources>
```

3-3. Java ソースファイル「Main.java」

D2xxManager クラスと FT_Device クラスをインポートします。

```
37
38 import com.ftdi.j2xx.D2xxManager;
39 import com.ftdi.j2xx.FT_Device;
40
```

REX-USB60MB 用インスタンス名を定義します。

```
51
52     static Context DeviceUARTContext;
53
54     private UsbManager mUsbManager = null;
55
56     public static D2xxManager ftdid2xx = null;
57     FT_Device ftDev = null;
58     D2xxManager.DriverParameters d2xxDrvParameter;
59
```

■REX-USB60MB の開始処理および終了処理を呼び出すタイミングについて

アプリの起動・終了以外に USB Serial Converter の接続・取外しや端末のスリープに対応するための REX-USB60MB の開始処理および終了処理を呼び出すタイミングについて説明します。

REX-USB60MB の開始処理は次の 3 つのタイミングで行います。

1. アプリ起動時
2. USB Serial Converter の接続時
3. 端末のスリープからの復帰時

onResume メソッドがこれらのタイミングに共通で呼び出されるメソッドになります。

したがって、***onResume*** メソッドが呼ばれたときに REX-USB60MB の開始処理を行います。

一方、REX-USB60MB の終了処理は次の 3 つのタイミングで行います。

1. アプリ終了時
2. USB Serial Converter の取り外し時
3. 端末がスリープへ入る時

onStop メソッドがこれらのタイミングに共通で呼び出されるメソッドになります。

したがって、***onStop*** メソッドが呼ばれたときに REX-USB60MB の終了処理を行います。

■*onCreate* メソッド

D2xxManager#getInstance メソッドで REX-USB60MB 用インスタンスを生成します。

```
135
136         try {
137             ftdid2xx = D2xxManager.getInstance(DeviceUARTContext = this);
138         } catch (D2xxManager.D2xxException ex) {
139             ex.printStackTrace();
140         }
141
```

ライブラリが識別可能な USB デバイスの追加するために ***D2xxManager#setVIDPID*** メソッドを呼び出して、REX-USB60 シリーズのベンダーID およびプロダクト ID を追加します。

```
142         if (ftdid2xx != null) {
143             int i;
144             for(i = 0; iPidTb[i] != 0xffff; i++) {
145                 if(!ftdid2xx.setVIDPID(iVid,iPidTb[i]))     // Set VID/PID of
146                     Log.i("ftd2xx-java","setVIDPID Error");
147             }
148         }
149
```

USB 接続状態を監視するブロードキャストレシーバーとして ***mUsbReceiver*** の登録を行い、このブロードキャストレシーバーで受信するインテントを登録します。

```
mIntent = new Intent(ACTION_USB_PERMISSION);
mPermissionIntent = PendingIntent.getBroadcast(this, 0, mIntent,

IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
filter.addAction(UsbManager.ACTION_USB_DEVICE_ATTACHED);
filter.addAction(UsbManager.ACTION_USB_DEVICE_DETACHED);

registerReceiver(mUsbReceiver, filter);     // register for USB60
```

■*onResume* メソッド

onResume メソッドでは、***Search_MyUsbSerial*** メソッドを使って接続された USB デバイスの ***UsbDevice*** オブジェクトを取得し、本プログラム（USB60BCR）での使用を許可するため、***UsbManager#requestPermission*** メソッドを呼び出します。

```
@Override
public void onResume() {
    super.onResume();

    if (D) Log.e(TAG, "+++ onResume +++");

    if (ftDev != null)
        return;

    UsbDevice mUsbDevice = Search_MyUsbSerial();

    if(mUsbDevice != null) {

        mUsbManager.requestPermission(mUsbDevice, mPermissionIntent);

        return;
    }
```

■*Search_MyUsbSerial* メソッド

ポートのオープンで使用するために、USB ポートに接続されているデバイスの *UsbDevice* オブジェクトを検索します。

```
D2xxManager.FtDeviceInfoListNode DevInfoNode = ftdid2xx.getDeviceInfoListDetail(openIndex);
int MyDeviceId = DevInfoNode.location / 16;
HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();
Iterator<UsbDevice> deviceIterator  = deviceList.values().iterator();
// Search UsbDevice object
while(deviceIterator.hasNext()){
     UsbDevice mUsbDev = deviceIterator.next();
     if (MyDeviceId == mUsbDev.getDeviceId()) {
         mUsbDevice = mUsbDev; // get UsbDevice object
         break;
     }
}
```

■*onStop* メソッド

onStop メソッドでは、*End_MyUSBSerial* メソッドを呼び出して、REX-USB60MB の終了処理を行います。

```
268
269     @Override
270     public void onStop() {
271         if (D) Log.e(TAG, "+++ onStop +++");
272         End_MyUsbSerial();
273         super.onStop();
274     }
275
```

■*onReceive* ブロードキャストレシーバー

onCreate メソッドで登録された USB 接続状態を監視するブロードキャストレシーバー処理。REX-USB60MB を装着すると、「この USB デバイスが接続されたときに USB60BCR を開きますか....」といった実行の許可を求める画面が表示されますが、このときに許可をした場合に、REX-USB60MB の初期化処理のために *Start_MyUsbSerial* メソッドを呼び出します。

```
BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        Log.d(TAG, "onReceive action : " + action);

        if (ACTION_USB_PERMISSION.equals(action)) {
            synchronized (this) {
                UsbDevice device = (UsbDevice)intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);

                if (intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                    if (device != null) {
                        if (alertDlg !=null) {
                            alertDlg.dismiss();
                            alertDlg = null;
                        }
                        Log.d(TAG, "Permission granted " + device);
                        Start_MyUsbSerial(device);
                    }
```

REX-USB60MB が取り外された時は、REX-USB60MB を終了するために ***End_MyUsbSerial*** メソッドを呼び出します。

```
650             } else if (UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action)) {
651                 End_MyUsbSerial();
652                 mTvStatus.setText(R.string.Notconnected);
653             }
```

■***Start_MyUsbSerial*** メソッド

Start_MyUsbSerial メソッドでは、REX-USB60MB の初期化のために ***connectFunction*** メソッドを呼び出し、その後、通信パラメーターを設定するため ***SetConfig*** メソッドを呼び出します。

```
213     private boolean Start_MyUsbSerial(UsbDevice MyUsbDevice) {
214
215         if (connectFunction(MyUsbDevice) == false) {
216             mTvStatus.setText(R.string.Notconnected);
217             return false;
218         }
219
220         SetConfig(baudRate, dataBit, stopBit, parity, flowControl);
221         mTvStatus.setText(R.string.Connected);
222
```

■***connectFunction*** メソッド

connectFunction メソッドは、ポートをオープンするために ***D2xxManager#openByUsbDevice*** メソッドを呼び出します。さらに、データ受信処理のためのスレッド ***readThread*** を開始します。

```
296     public boolean connectFunction(UsbDevice MyUsbDevice)
297     {
298
299         if (MyUsbDevice == null)
300             return false;
301
302         if(null == ftDev) {
303             ftDev = ftdid2xx.openByUsbDevice(DeviceUARTContext, MyUsbDevice);
304         } else {
305             synchronized(ftDev)
306             {
307                 ftDev = ftdid2xx.openByUsbDevice(DeviceUARTContext, MyUsbDevice);
308             }
309         }
310
```

さらに、データ受信処理のためのスレッド ***readThread*** を開始します。

```
315
316         if (true == ftDev.isOpen()) {
317             Toast.makeText(DeviceUARTContext, "open device port OK", Toast.LENGTH_SHORT).show();
318
319             if(false == bReadThreadGoing) {
320                 read_thread = new readThread(handler);
321                 read_thread.start();
322                 bReadThreadGoing = true;
323             }
324         } else {
```

■*SetConfig* メソッド

SetConfig メソッドでは、通信パラメータを設定するために ***FT_DEVICE*** の以下の各メソッドを呼び出します。

setBaudRate メソッド：ボーレート

setDataCharacteristics メソッド：データビット、ストップビット、パリティ

setFlowControl メソッド：フロー制御

■*End_MyUsbSerial* メソッド

End_MyUsbSerial メソッドでは、***ReadThread*** を停止させた後にポートをクローズします。

```
227    public void End_MyUsbSerial() {
228
229        bReadThreadGoing = false;
230        try {
231            Thread.sleep(50);
232        }
233        catch (InterruptedException e) {
234            e.printStackTrace();
235        }
236
237        if(ftDev != null) {
238            synchronized(ftDev) {
239                if( true == ftDev.isOpen()) {
240                    ftDev.close();
241                }
242            }
243            ftDev = null;
244        }
245    }
246
```

■*readThread* スレッド

REX-USB60MB からのデータ受信処理はブロックされるため、別スレッド ***readThread*** として処理します。

```
539
540    private class readThread  extends Thread {
541        Handler mHandler;
542
543        readThread(Handler h){
544            mHandler = h;
545            this.setPriority(Thread.MIN_PRIORITY);
546        }
547
```

そして ***readThread*** の ***run*** メソッド内で受信処理を行います。
FT_DEVICE #getQueueStatus メソッドで受信データが存在するかチェックして、
受信データが存在すれば、データ受信は ***FT_DEVICE #read*** メソッド呼び出します。
取得されたバーコードデータは、***readThread*** から MainActivity 内のハンドラーへメッセージで通知し、データ表示処理は、MainActivity 内で処理します。

```
548         @Override
549         public void run()
550         {
551             int i;
552
553             while(true == bReadThreadGoing) {
554                 try {
555                     Thread.sleep(50);
556                 } catch (InterruptedException e) {
557                     e.printStackTrace();
558                 }
559
560                 synchronized(ftDev)
561                 {
562                     iavailable = ftDev.getQueueStatus();
563                     if (iavailable > 0) {
564
565                         if(iavailable > readLength){
566                             iavailable = readLength;
567                         }
568                         if((savePosition + iavailable) > readLength)
569                             iavailable = readLength - savePosition;
570
571                         ftDev.read(readData, iavailable);
572                         for (i = 0; i < iavailable; i++) {
573                             readDataToText[savePosition+i] = (char) readData[i];
574                         }
```

そして ***readThread*** で取得されたバーコードデータは、***readThread*** から MainActivity 内のハンドラーへメッセージで通知し、データ表示処理は、MainActivity 内で処理します。

```
584
585                     } else {
586                         if (savePosition > 0) {
587                             saveLength = savePosition;
588                             Message msg = mHandler.obtainMessage();
589                             mHandler.sendMessage(msg);
590                             savePosition = 0;
591                         }
592                     }
```

上記 sendMessage が以下のハンドラー処理へ通知されます。

```
490
491     final Handler handler =  new Handler()
492     {
493         @Override
494         public void handleMessage(Message msg) {
495
496             int rChar;
497             int rbufp;
```

以上でサンプルプログラムの説明を終わります。

本書では、Android の USB Host API については説明しておりません。これについて知りたい場合は、Google の開発者向けサイトの以下を参照してください。

http://developer.android.com/guide/topics/connectivity/usb/host.html

最後に、このサンプルプログラムは、あくまで Android 上で USB Serial Converter へのアクセス方法を説明するためのもので、データ送信処理や FTDI 製 USB Serial Converter Controller 固有の設定処理、エラー処理などは含まれておりません。

本サンプルプログラムおよび本書に関するお問い合わせは、下記のラトックシステムの Web サイト上の問い合わせフォームからお願いします。

サポートセンター宛メール
http://web1.ratocsystems.com/mail/support.html