REX-USB60MI

USB シリアルコンバータ（Micro-USB タイプ）

Android 用サンプルプログラム
USB60Samp01 について

2012 年 8 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、弊社製 USB シリアルコンバータ REX-USB60MI を AndroidOS 上で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、FTDI 社製の USB シリアル変換チップに対応した USB ドライバとして公開されている FTDriver (※1)を利用しております。

※1: FTDriver は、github のサイト(http://github.com/ksksue/FTDriver)からソースコードを zip でダウンロードできます。
FTDriver は、あくまで個人的な活動で作成されています。
作成者のご好意により紹介の許可を得ておりますので、FTDriver の作成者へメール等で直接のお問い合わせされないようお願いいたします。

本サンプルプログラムの機能は、REX-USB60MI に接続されたバーコードリーダでスキャンされたバーコード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 シリアルコンバータ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。
・USB60Samp01 のダウンロードとプロジェクトのインポート
・FTDriver のダウンロードとプロジェクトのインポート
・サンプルプログラムのプロジェクトと FTDriver プロジェクトと関連付け
・FTDriver への REX-USB60MI の VID/PID の追加

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

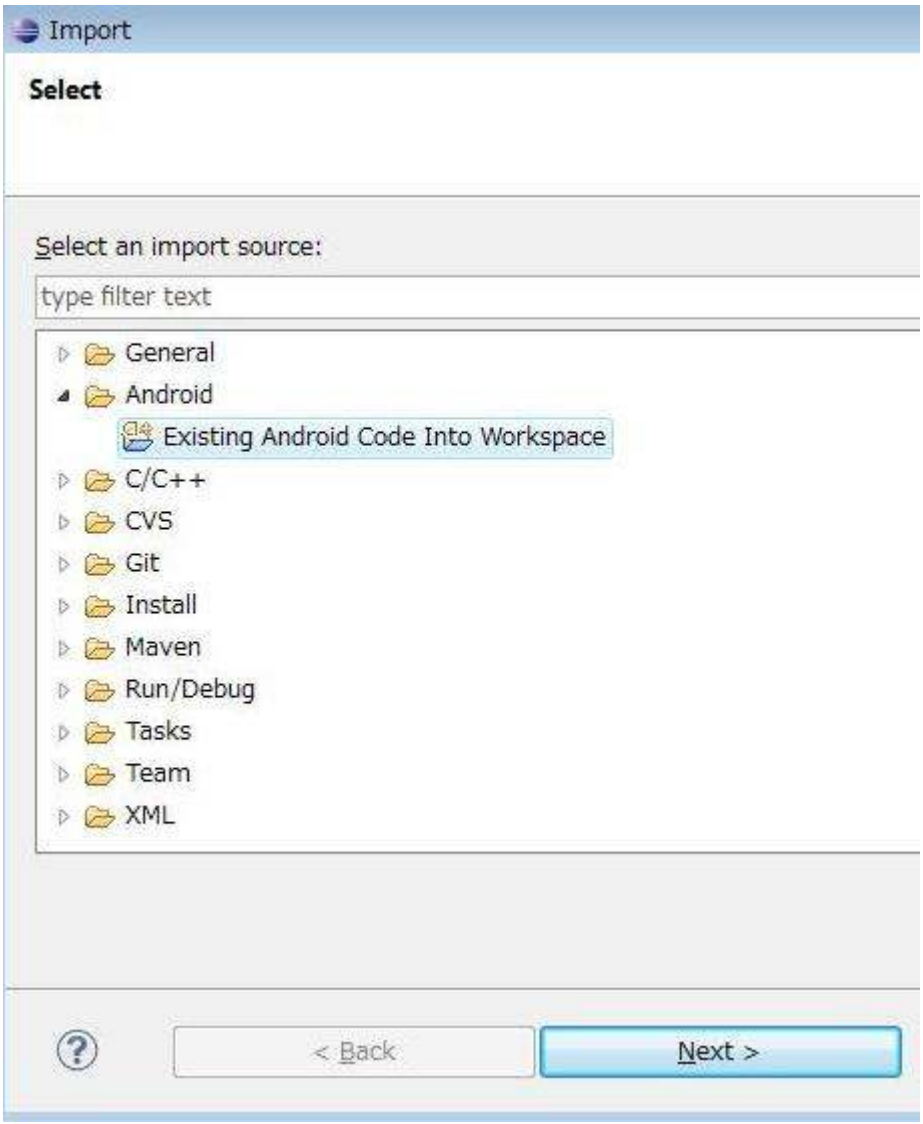
2-1. USB60Samp01 のダウンロードとプロジェクトのインポート

(1) ラトックシステムのダウンロードのサイトから USB60Samp01 のソースコード一式をダウンロードします。

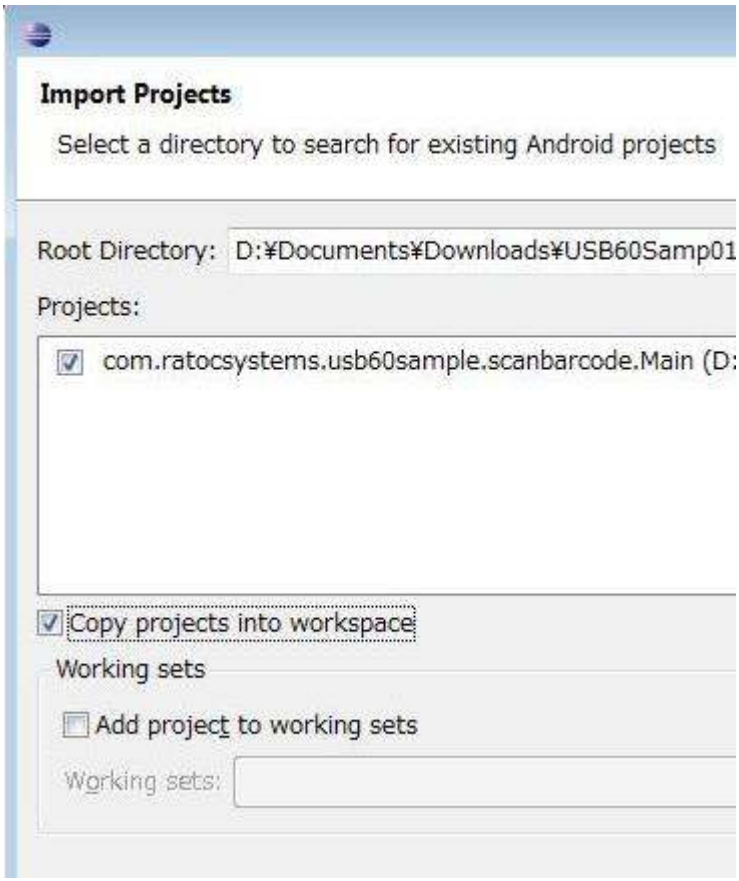
(2) ダウンロードしたファイルを任意のフォルダで解凍します。

(3) Eclipse で USB60Samp01 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。



「com.ratocsystems.usb60sample.scanbarcode.Main」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. FTDriver のダウンロードとプロジェクトのインポート

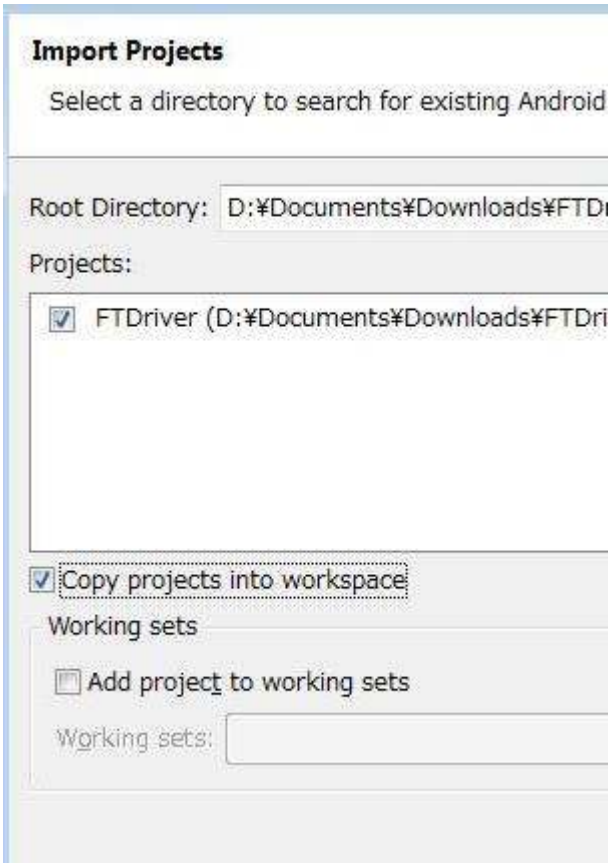
(1) github のサイト(http://github.com/ksksue/FTDriver)から FTDriver のソースコードをダウンロードします。

(2) ダウンロードしたファイルを任意のフォルダで解凍します。

(3) Eclipse で FTDriver プロジェクトのインポートします。

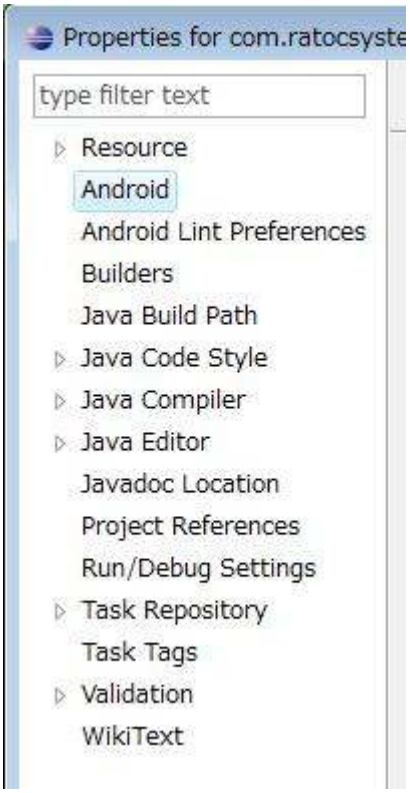
2-1.と同じく[File]-[Import]から[Android]の[Existing Android Code Into Workspace]を選択して、[Next]を押します。

そして、[Import Pojects]の画面に表示された「FTDriver」を選択し、「Copy project to workspace」にチェックをして、[Finish]を押します。

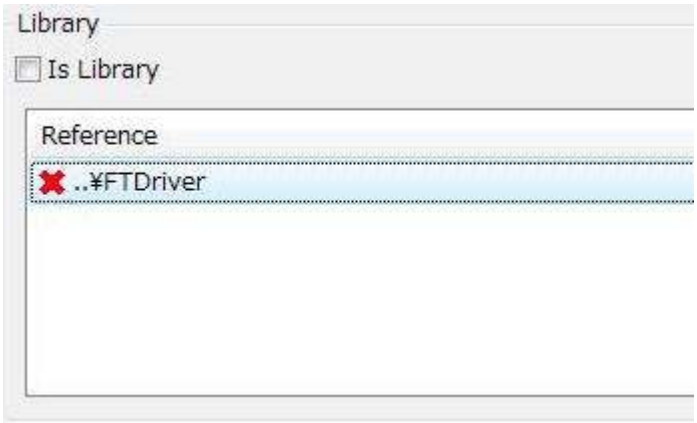


2-3. サンプルプログラムのプロジェクトと FTDriver プロジェクトと関連付け

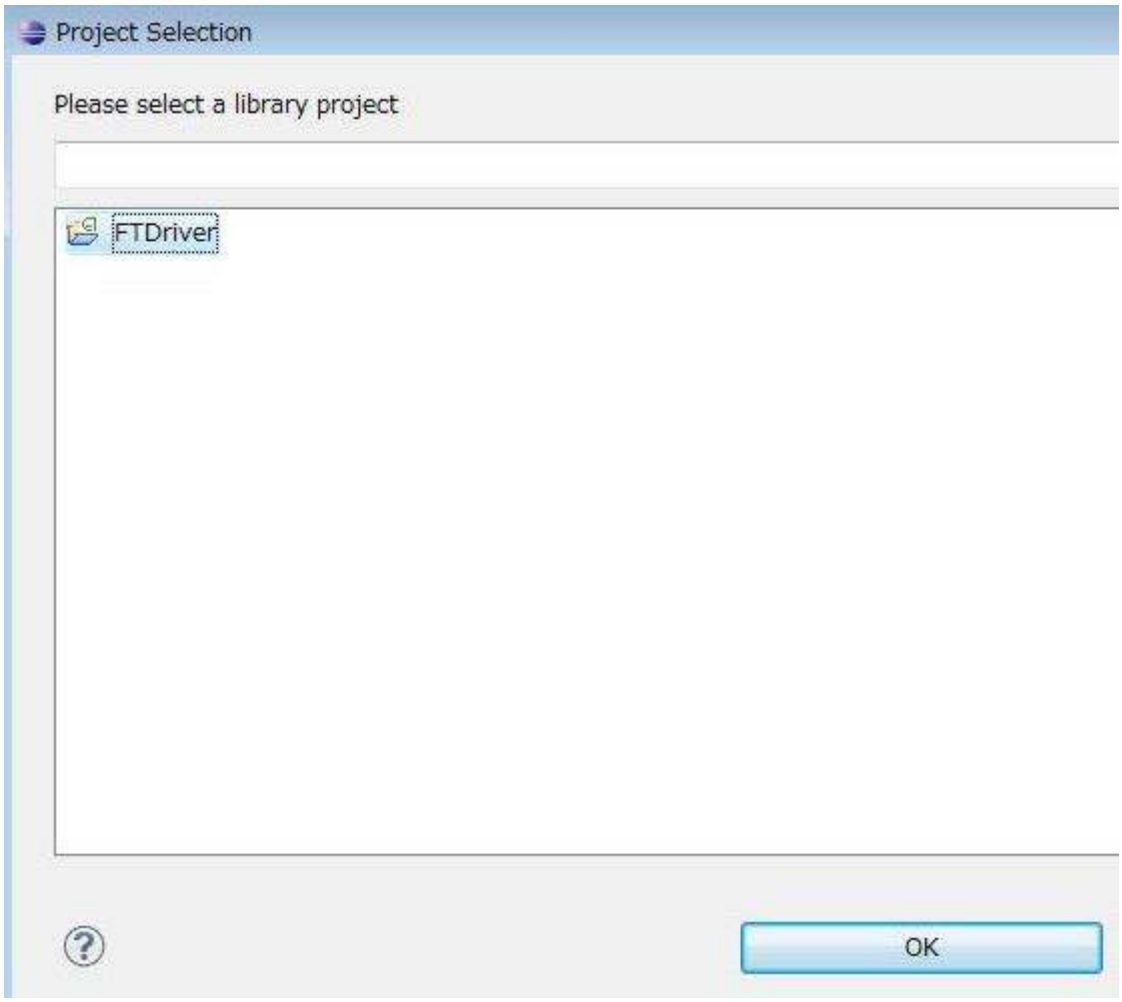
(1) 「com.ratocsystems.usb60sample.scanbarcode.Main」プロジェクトを選択して、右クリックメニューから「Properties」を選択します。
[type filter text]の[Android]を選択します。



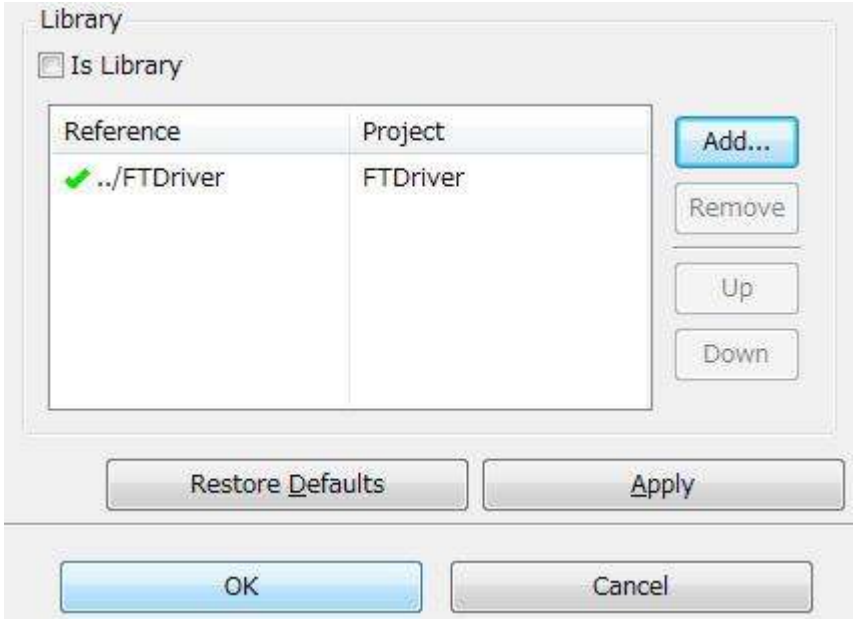
Library 項目にある FTDriver に緑のチェックマークが表示されていれば、そのまま「OK」をクリックしてプロパティ画面は閉じて、2-4. へ進んでください。
下のように FTDriver に赤の×マークが付いている場合は、右側の「Remove」を押して、一旦削除します。



(2)「Add」ボタンを押して、FTDriver を選択し、「OK」を押します。



(3) Library 項目に表示された FTDriver に緑のチェックマークが表示されたら、「OK」をクリックしてプロパティ画面は閉じます。



2-4. FTDriver へ REX-USB60MI の VID/PID を追加

(1) FTDriver プロジェクトの[src]下にある FTDriver.java を開きます。

(2) ファイルの先頭部分にある、以下の USB デバイスの VID/PID のテーブルを探します。

```
44
45    private static final UsbId[] IDS = {
46        new UsbId(0x0403, 0x6001, 6, 1, FTDICHIPTYPE.FT232RL),  // FT232RL
47        new UsbId(0x0403, 0x6014, 9, 1, FTDICHIPTYPE.FT232H),   // FT232H
48        new UsbId(0x0403, 0x6010, 5, 2, FTDICHIPTYPE.FT2232C),  // FT2232C
49        new UsbId(0x0403, 0x6010, 5, 2, FTDICHIPTYPE.FT2232D),  // FT2232D
50        new UsbId(0x0403, 0x6010, 7, 2, FTDICHIPTYPE.FT2232HL), // FT2232HL
51        new UsbId(0x0403, 0x6011, 8, 4, FTDICHIPTYPE.FT4232HL), // FT4232HL
52        new UsbId(0x0403, 0x6015, 10, 1, FTDICHIPTYPE.FT230X),  // FT230X
53        new UsbId(0x0584, 0xB020, 4, 1, FTDICHIPTYPE.FT232RL),  // REX-USB60F thanks to @hyokota555
54        new UsbId(0x0000, 0x0000, 0, 2, FTDICHIPTYPE.CDC),      // CDC
55    };
56    private UsbId mSelectedDeviceInfo;
57
```

このテーブルの REX-USB60F の次行に USB の PID を「0xB02F」に変更した REX-USB60MI の行を追加します。

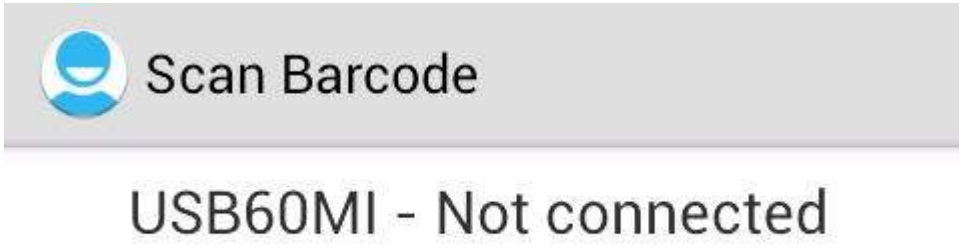
```
45    private static final UsbId[] IDS = {
46        new UsbId(0x0403, 0x6001, 6, 1, FTDICHIPTYPE.FT232RL),  // FT232RL
47        new UsbId(0x0403, 0x6014, 9, 1, FTDICHIPTYPE.FT232H),   // FT232H
48        new UsbId(0x0403, 0x6010, 5, 2, FTDICHIPTYPE.FT2232C),  // FT2232C
49        new UsbId(0x0403, 0x6010, 5, 2, FTDICHIPTYPE.FT2232D),  // FT2232D
50        new UsbId(0x0403, 0x6010, 7, 2, FTDICHIPTYPE.FT2232HL), // FT2232HL
51        new UsbId(0x0403, 0x6011, 8, 4, FTDICHIPTYPE.FT4232HL), // FT4232HL
52        new UsbId(0x0403, 0x6015, 10, 1, FTDICHIPTYPE.FT230X),  // FT230X
53        new UsbId(0x0584, 0xB020, 4, 1, FTDICHIPTYPE.FT232RL),  // REX-USB60F thanks to @hyokota555
54        new UsbId(0x0584, 0xB02F, 4, 1, FTDICHIPTYPE.FT232RL),  // REX-USB60MI
55        new UsbId(0x0000, 0x0000, 0, 2, FTDICHIPTYPE.CDC),      // CDC
56    };
57    private UsbId mSelectedDeviceInfo;
```

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。

2-5. サンプルプログラムの動作

パソコンからターゲットの Android タブレットを USB ケーブルで接続して、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「com.ratocsystems.usb60sample.scanbarcode.Main」プロジェクトを選択して[Run]をクリックします。
「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。
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。

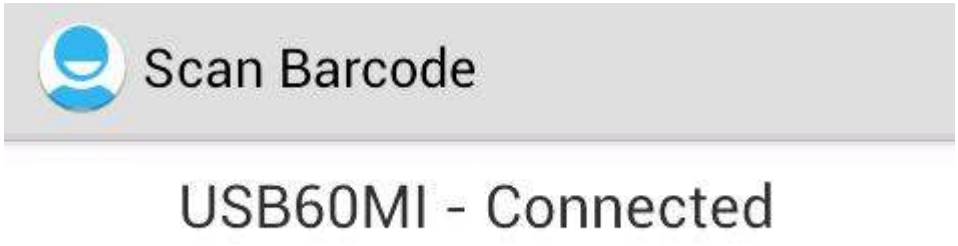
REX-USB60MI が接続されていないので、「USB60MI – Not connected」と表示されています。



それでは、Android タブレットの USB ホストポートに REX-USB60MI を接続してみてください。

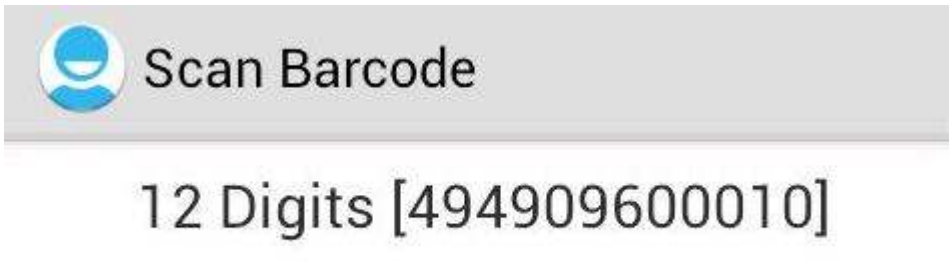
画面の中央に「この USB デバイスが接続されたときに Scan Barcode を開きますか？....」
といった実行許可の確認ダイアログが表示されます。
[OK]を押して実行を許可してください。

「USB60MI – Not connected」が「USB60MI - Connected」に変わっています。



次に、バーコードスキャナを接続し、スキャンしてください。

バーコードが読み込まれると、左に読み込んだコードの桁数、読み込んだコードがその右に表示されます。



最後に、REX-USB60MI を Android タブレットの USB ポートから外してください。
「USB60MI – 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=".Main"
16            android:label="@string/title_activity_main" >
17            <intent-filter>
18                <action android:name="android.intent.action.MAIN" />
19
20                <category android:name="android.intent.category.LAUNCHER" />
21            </intent-filter>
22
23            <intent-filter>
24                <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
25            </intent-filter>
26
27            <meta-data android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
28                android:resource="@xml/device_filter" />
29
30        </activity>
```

<Activity>エレメントの中に android.hardware.usb.action.USB_DEVICE_ATTACHED インテント用の <intent-filter> と <meta-data> エレメントのペアを記述します。

<meta-data>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「deive_filter.xml」

deive_filter.xml では、REX-USB60MI のベンダーID とプロダクト ID の指定を<resources> エレメントの中に記述します。

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

3-3. Java ソースファイル「Main.java」

```
11
12 import jp.ksksue.driver.serial.FTDriver;
13 import android.hardware.usb.UsbManager;
14 import android.os.Bundle;
```

FTDriver クラスと UsbManager クラスをインポートします。

REX-USB60MI 用インスタンス名を定義します。

```
38
39     // REX-USB60MI
40     FTDriver mUSB60;
41
42     // Baudrate is 9600bps
43     private int mBaudrate = FTDriver.BAUD9600;
44
```

上記例では、ボーレートは 9600bps 固定としています。

このアプリを認識するためのインテントアクション名を指定します。

```
60
61     private static final String ACTION_USB_PERMISSION =
62             "sanbarcode.action.USB_PERMISSION";
63
```

「OnCreate」メソッドは、REX-USB60MI 用インスタンスを生成して、USB 接続状態監視のブロードキャストレシーバを登録したら、REX-USB60MI を初期化して処理を開始します。
REX-USB60MI 用のインスタンスを生成します。

```
74
75         // Get Class Driver (FTDriver) for FTDI USB Serial Convertor
76         mUSB60 = new FTDriver((UsbManager)getSystemService(Context.USB_SERVICE));
77
```

USB 接続状態を監視するブロードキャストレシーバとして「mUsbReceiver」を登録します。
このブロードキャストレシーバで受信するインテントを登録します。

```
77
78         IntentFilter filter = new IntentFilter();
79         filter.addAction(UsbManager.ACTION_USB_DEVICE_ATTACHED);
80         filter.addAction(UsbManager.ACTION_USB_DEVICE_DETACHED);
81         registerReceiver(mUsbReceiver, filter);
82
```

REX-USB60MI の初期化処理のために FTDriver の「begin」メソッドを呼び出します。
「mainloop」はデータ受信のための処理です。

```
86
87         if (mUSB60.begin(mBaudrate)) {
88             mainloop();
89         } else {
90             if (D) Log.e(TAG, "+++ Not Connected +++");
91             mTvStatus.setText(R.string.Notconnected);
92         }
93
```

「onDestory」メソッドは、アプリ終了時の処理です。
REX-USB60MI を終了するために FTDriver の「end」メソッドを呼び出します。

```
101
102     @Override
103     protected void onDestroy() {
104         mUSB60.end();
105         mStop = true;
106         unregisterReceiver(mUsbReceiver);
107
108         super.onDestroy();
109     }
110
```

REX-USB60MI からのデータ受信処理「mScanBcrLoop」を別スレッドとして処理します。

```
110
111     private void mainloop() {
112         mStop = false;
113         mTvStatus.setText(R.string.Connected);
114         // Start thread for scannig Barcode
115         new Thread(mScanBcrLoop).start();
116     }
117
```

データ受信処理は別スレッドとして処理されるため、Runnable インターフェースを実装して、run() メソッド内で受信処理を行います。

```
121
122     private Runnable mScanBcrLoop = new Runnable() {
123         @Override
124         public void run() {
125
```

データ受信は FTDriver の「read」メソッド呼び出します。

```
134
135             while (true) {
136                 // get receive data
137                 len = mUSB60.read(rbuf);
138
139                 if (len > 0) {
140
```

受信データが存在すれば、データを取り出して表示するための処理を行います。

最後は、最初に「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の処理です。

```
194     //
195     // Broardcast Receiver for USB60 Hotplug insert/remove, detect
196     //
197     BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
198         @Override
199         public void onReceive(Context context, Intent intent) {
200             String action = intent.getAction();
201
```

初めて、REX-USB60MI を装着すると、「この USB デバイスが接続されたときに Scan Barcode 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と 2 回目以降に REX-USB60MI を装着して検出された場合に、FTDriver の「usbAttached」と「begin」メソッドを呼び出します。

【2 回目以降に REX-USB60MI が装着された時】

```
205             if (UsbManager.ACTION_USB_DEVICE_ATTACHED.equals(action)) {
206                 if (!mUSB60.isConnected()) {
207                     if (D) Log.e(TAG, "+++ Insert +++");
208                     mUSB60.usbAttached(intent);
209                     mUSB60.begin(mBaudrate);
210                     mainloop();
211                 }
212
```

【初めて、REX-USB60MI を装着して、アプリが許可された時】

```
226             } else if (ACTION_USB_PERMISSION.equals(action)) {
227                 synchronized(this) {
228                     if (!mUSB60.isConnected()) {
229                         mUSB60.usbAttached(intent);
230                         mUSB60.begin(mBaudrate);
231                     }
232                 }
233                 mainloop();
234             }
```

REX-USB60MI が取り外された時は、FTDriver の「usbDetached」と「begin」メソッドを呼び出します。

```
216             } else if (UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action)) {
217                 if (D) Log.e(TAG, "+++ Removed +++");
218                 mTvStatus.setText(R.string.Notconnected);
219                 mUSB60.usbDetached(intent);
220                 mUSB60.end();
221                 mStop = true;
222
```

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 シリアルコンバータ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 シリアルコンバータ・コントローラ固有の設定処理、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