# Android USB Host 使用详解

## 零 USB背景知识

USB是一种数据通信方式，也是一种数据总线，而且是最复杂的总线之一。   
硬件上，它是用插头连接。一边是公头（plug），一边是母头（receptacle）。例如，PC上的插座就是母头，USB设备使用公头与PC连接。   
目前USB硬件接口分三种，普通PC上使用的叫Type；原来诺基亚功能机时代的接口为Mini USB；目前[Android](http://lib.csdn.net/base/android" \o "Android知识库" \t "_blank)手机使用的Micro USB。

Host   
USB是由Host端控制整个总线的数据传输的。单个USB总线上，只能有一个Host。   
OTG   
On The [Go](http://lib.csdn.net/base/go)，这是在USB2.0引入的一种mode，提出了一个新的概念叫主机协商协议（Host Negotiation Protocol），允许两个设备间商量谁去当Host。

了解更多USB知识，请参考USB官网以及下面这篇文章：   
<http://www.crifan.com/files/doc/docbook/usb_basic/release/html/usb_basic.html>

## 一、Android中的USB

Android对Usb的支持是从3.1开始的，显然是加强Android平板的对外扩展能力。而对Usb使用更多的，是Android在工业中的使用。Android工业板子一般都会提供多个U口和多个串口，它们是连接外设的手段与桥梁。Android USB有两种模式Host  Mode和Accessory Mode,下面就来介绍一下Android Usb使用模式之一的USB Host。

android.hardware.usb包下提供了USB开发的相关类。   
我们需要了解UsbManager、UsbDevice、UsbInterface、UsbEndpoint、UsbDeviceConnection、UsbRequest、UsbConstants。   
1、UsbManager:获得Usb的状态，与连接的Usb设备通信。   
2、UsbDevice：Usb设备的抽象，它包含一个或多个UsbInterface，而每个UsbInterface包含多个UsbEndpoint。Host与其通信，先打开UsbDeviceConnection，使用UsbRequest在一个端点（endpoint）发送和接收数据。   
3、UsbInterface：定义了设备的功能集，一个UsbDevice包含多个UsbInterface，每个Interface都是独立的。   
4、UsbEndpoint：endpoint是interface的通信通道。   
5、UsbDeviceConnection：host与device建立的连接，并在endpoint传输数据。   
6、UsbRequest：usb 请求包。可以在UsbDeviceConnection上异步传输数据。注意是只在异步通信时才会用到它。   
7、UsbConstants：usb常量的定义，对应[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "_blank)/usb/ch9.h

## 二、USB插入事件

Usb的插入和拔出是以系统广播的形式发送的，只要我们注册这个广播即可。

@Override

protected void onResume() {

super.onResume();

IntentFilter usbFilter = new IntentFilter();

usbFilter.addAction(UsbManager.ACTION\_USB\_DEVICE\_ATTACHED);

usbFilter.addAction(UsbManager.ACTION\_USB\_DEVICE\_DETACHED);

registerReceiver(mUsbReceiver, usbFilter);

}

@Override

protected void onPause() {

super.onPause();

unregisterReceiver(mUsbReceiver);

}

private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

tvInfo.append("BroadcastReceiver in\n");

if(UsbManager.ACTION\_USB\_DEVICE\_ATTACHED.equals(action)) {

tvInfo.append("ACTION\_USB\_DEVICE\_ATTACHED\n");

} else if(UsbManager.ACTION\_USB\_DEVICE\_DETACHED.equals(action)) {

tvInfo.append("ACTION\_USB\_DEVICE\_DETACHED\n");

}

}

};

## 三、Usb插入时启动程序

有些应用场景是，Usb插入后启动特定程序处理特定问题。   
我们的做法就是在Manifest中某个Activity加入Usb插入的action。

<intent-filter>

<action android:name="android.hardware.usb.action.USB\_DEVICE\_ATTACHED"/>

</intent-filter>

<meta-data android:name="android.hardware.usb.action.USB\_DEVICE\_ATTACHED"

android:resource="@xml/usbfilter" />

在usbfilter中加入厂商id和产品id的过滤，如下：

<?xml version="1.0" encoding="utf-8"?>

<resources>

<usb-device vendor-id="1234" product-id="5678" />

</resources>

结果就是，当此型号设备通过Usb连接到系统时，对应的Activity就会启动。

## 四、UsbManager的初始化

mUsbManager = (UsbManager)getSystemService(Context.USB\_SERVICE);

## 五、列出Usb设备

HashMap<String,UsbDevice> deviceHashMap = mUsbManager.getDeviceList();

Iterator<UsbDevice> iterator = deviceHashMap.values().iterator();

while (iterator.hasNext()) {

UsbDevice device = iterator.next();

tvInfo.append("\ndevice name: "+device.getDeviceName()+"\ndevice product name:"

+device.getProductName()+"\nvendor id:"+device.getVendorId()+

"\ndevice serial: "+device.getSerialNumber());

}

## 六、USB使用权限

[安卓](http://lib.csdn.net/base/android)系统对USB口的使用需要得到相应的权限，而这个权限要用户亲自给才行。   
首先我们会确认一下上一节中的device是否已经获得权限，如果没有就要主动申请权限：

//先判断是否为自己的设备

//注意：支持十进制和十六进制

//比如：device.getProductId() == 0x04D2

if(device.getProductId() == 1234 && device.getVendorId() == 5678) {

if(mUsbManager.hasPermission(device)) {

//do your work

} else {

mUsbManager.requestPermission(device,mPermissionIntent);

}

}

我们仍然使用广播来获得权限赋予情况。

public static final String ACTION\_DEVICE\_PERMISSION = "com.linc.USB\_PERMISSION";

注册广播

mPermissionIntent = PendingIntent.getBroadcast(this,0,new Intent(ACTION\_DEVICE\_PERMISSION),0);

IntentFilter permissionFilter = new IntentFilter(ACTION\_DEVICE\_PERMISSION);

registerReceiver(mUsbReceiver,permissionFilter);

接收器的代码：

private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

tvInfo.append("BroadcastReceiver in\n");

if (ACTION\_DEVICE\_PERMISSION.equals(action)) {

synchronized (this) {

UsbDevice device = intent.getParcelableExtra(UsbManager.EXTRA\_DEVICE);

if (intent.getBooleanExtra(UsbManager.EXTRA\_PERMISSION\_GRANTED, false)) {

if (device != null) {

tvInfo.append("usb EXTRA\_PERMISSION\_GRANTED");

}

} else {

tvInfo.append("usb EXTRA\_PERMISSION\_GRANTED null!!!");

}

}

}

}

};

## 七、通信

UsbDevice有了权限，下面就可以进行通信了。   
这里要用到：UsbInterface、UsbEndpoint（一进一出两个endpoint，双向通信）、UsbDeviceConnection。   
注意：通信的过程不能在UI线程中进行。   
得到授权后，将做一些通信前的准备工作，如下：

private void initCommunication(UsbDevice device) {

tvInfo.append("initCommunication in\n");

if(1234 == device.getVendorId() && 5678 == device.getProductId()) {

tvInfo.append("initCommunication in right device\n");

int interfaceCount = device.getInterfaceCount();

for (int interfaceIndex = 0; interfaceIndex < interfaceCount; interfaceIndex++) {

UsbInterface usbInterface = device.getInterface(interfaceIndex);

if ((UsbConstants.USB\_CLASS\_CDC\_DATA != usbInterface.getInterfaceClass())

&& (UsbConstants.USB\_CLASS\_COMM != usbInterface.getInterfaceClass())) {

continue;

}

for (int i = 0; i < usbInterface.getEndpointCount(); i++) {

UsbEndpoint ep = usbInterface.getEndpoint(i);

if (ep.getType() == UsbConstants.USB\_ENDPOINT\_XFER\_BULK) {

if (ep.getDirection() == UsbConstants.USB\_DIR\_OUT) {

mUsbEndpointIn = ep;

} else {

mUsbEndpointOut = ep;

}

}

}

if ((null == mUsbEndpointIn) || (null == mUsbEndpointOut)) {

tvInfo.append("endpoint is null\n");

mUsbEndpointIn = null;

mUsbEndpointOut = null;

mUsbInterface = null;

} else {

tvInfo.append("\nendpoint out: " + mUsbEndpointOut + ",endpoint in: " +

mUsbEndpointIn.getAddress()+"\n");

mUsbInterface = usbInterface;

mUsbDeviceConnection = mUsbManager.openDevice(device);

break;

}

}

}

}

发送数据如下：

result = mUsbDeviceConnection.bulkTransfer(mUsbEndpointOut, mData, (int)buffSize, 1500);//需要在另一个线程中进行

## 八、一个Android程序：

### 1）在AndroidManifest.xml文件中添加

<uses-feature android:name="android.hardware.usb.host" />

<uses-sdk android:minSdkVersion="12" />

有可能你在其它地方看见这样的写法

<uses-feature android:name="android.hardware.usb.host" android:required="true"/>

本人建议不要这样写，因为这样写的话可能在/system/etc/permissions目录下不能自动创建android.hardware.usb.host.xml文件，而需要手动创建。

### 2）在<activity ...>添加

<intent-filter>

     <action android:name="android.hardware.usb.action.USB\_DEVICE\_ATTACHED" />

</intent-filter>

<meta-data android:name="android.hardware.usb.action.USB\_DEVICE\_ATTACHED"

     android:resource="@xml/device\_filter" />

在res/xml文件夹下新建device\_filter.xml

<resources>

    <usb-device vendor-id="3544" product-id="8199" />

    <usb-device vendor-id="5251" product-id="4608" />

</resources>

 其中vendor-id和product-id为插入USB设备的生产厂家号和产品号，但插入（attached）上面列出的设备之一时就会弹出选择打开应用程序的对话框。

注：上面的id为10进制的，而通过电脑上查看的id为16进制的。

### 3）获取所有已连接上的USD设备的信息

public class MainActivity extends ActionBarActivity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        UsbManager mManager = (UsbManager) getSystemService(Context.USB\_SERVICE);

        HashMap<String, UsbDevice> deviceList = mManager.getDeviceList();

        Iterator<UsbDevice> deviceIterator = deviceList.values().iterator();

        String info = new String();

        while (deviceIterator.hasNext())

        {

            UsbDevice device = deviceIterator.next();

            info += "Device Name:" + device.getDeviceName() + "\n";

            info += "Device ID:" + device.getDeviceId() + "\n";

            info += "Device Class:" + device.getDeviceClass() + "\n";

            info += "Device Protocl:" + device.getDeviceProtocol() + "\n";

            info += "Device Vendor ID:" + device.getVendorId() + "\n";

            info += "Device Product ID:" + device.getProductId() + "\n";

            info += "Device Interface Count:" + device.getInterfaceCount() + "\n\n";

            // Get interface details

            for (int index = 0; index < device.getInterfaceCount(); index++)

            {

                UsbInterface mUsbInterface = device.getInterface(index);

                info += "Interface " + index + ":\n";

                info += "Interface ID:" + mUsbInterface.getId() + "\n";

                info += "Inteface class:" + mUsbInterface.getInterfaceClass() + "\n";

                info += "Interface protocol:" + mUsbInterface.getInterfaceProtocol() + "\n";

                info += "Endpoint count:" + mUsbInterface.getEndpointCount() + "\n\n";

                // Get endpoint details

                for (int epi = 0; epi < mUsbInterface.getEndpointCount(); epi++)

                {

                    UsbEndpoint mEndpoint = mUsbInterface.getEndpoint(epi);

                    info += "Endpoint " + epi + ":\n";

                    info += "Attributes: " + mEndpoint.getAttributes() + "\n";

                    info += "Direction: " + mEndpoint.getDirection() + "\n";

                    info += "Number: " + mEndpoint.getEndpointNumber() + "\n";

                    info += "Interval: " + mEndpoint.getInterval() + "\n";

                    info += "Packet size: " + mEndpoint.getMaxPacketSize() + "\n";

                    info += "Type: " + mEndpoint.getType() + "\n\n";

                }

            }

        }

        TextView textView = (TextView)findViewById(R.id.info);

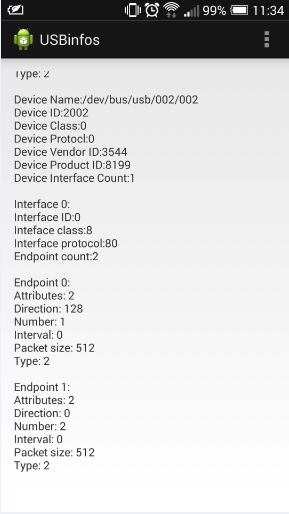
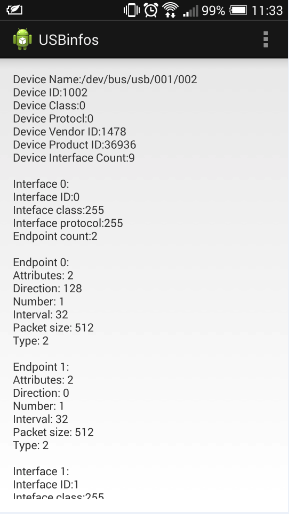
        textView.setText(info);

    }

}

通过上面这段代码可以获得USB设备的所有信息，包括接口（interface）和端点（endpoint）的信息，只有了解设备的类型才能在后面进行通信。注：本人在测试时即使没有连接设备也会有显示一个设备，这个设备即使删除它后系统也会自动生成，这里不用管它，之后把它过滤掉就可以了。

[测试](http://lib.csdn.net/base/softwaretest)结果如下：



上面左图为第一个设备（无对应设备，懂的大神可以解释一下），右图为连接的U盘。Device Name 的值是一个文件路径，在路径下可以看到相应的文件，插入设备时自动生成，拔出时自动删除，懂[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "_blank)的应该明白原理，我是不懂，所以就不解释了。

源码下载链接：[源码链接](http://download.csdn.net/detail/glouds/8055743)

## 九、Android USB HID设备通信controlTransfer 接口参数

关于public int controlTransfer (int requestType, int request, int value, int index, byte[] buffer, int length, int timeout)方法的参数说明

其实此方法对应了Linux的一个函数就是：[usb\_control\_msg()](http://hi.baidu.com/zengzhaonong/blog/item/6bb44682452a4da00df4d245.html" \t "_blank)，大家可以查下看下有如下一个结构体，其中的参数对应相信你看到后就不会陌生了，usb\_ctrlrequest

|-------------------------------|  
| \_\_u8    bRequestType--|  
| \_\_u8    bRequest     ---|  
| \_\_le16 wValue      ----|  
| \_\_le16 wIndex      ----|  
| \_\_le16 wLength    ----|  
|-------------------------------|

bRequestType  
    D7     数据的传输方向：0表示从主机到设备； 1表示从设备到主机；  
    D6~5   命令的类型：   0表示标准命令；    1表示类命令；      2表示厂商提供的命令； 3保留；  
    D4~0   接收对象；     0表示设备；       1表示接口；       2表示端点；         3表示其他；  
bRequest  
    命令的序号(其实就是命令)；所有的命令都是以不同编码值的方式传递给设备的，bRequest就表示USB命令的编码值

    我对这个参数的理解也就是请求号  
wValue

    这个字段对于不同的命令有不同的含义，下面有介绍

 wIndex

    传递索引或者诸如接口或者端点号的偏移量，这个参数要查找你要通信的HID设备，我这里的HID的接口的索引是0。   我对此参数也不很理解，从别人那看来的。我的应用中使用的默认值0，通信无问题。

wLength  
    数据的长度，为0或者buffer的长度

USB标准命令  
---------------------------------------------------------------   
命令            bReuestType bRequest         wValue      wIndex     wLength     数据           
Get\_Descriptor 1000 0000b -GET\_DESCRIPTOR   描述符类型   -0或语言ID --描述符长度 --描述符           
                                           -描述符的索引号   
  
wValue   高8位表示描述符类型； 低8位表示描述符索引号   
  
GET\_DESCRIPTOR = 6   
GetHubdescriptor = 0XA0   
Get\_Descriptor = 0X80   
  
---------------------------------------------------------------   
bRequestType    bRequest   wValue     wIndex   wLength   
80                          6              100          0        12   
80                          6              200          0         9   
80                          6              300          0         FF   
80                          6              301          409      FF   
80                          6              302          409      FF   
80                          6              303          409      FF   
  
wValue:

100        表示获取设备描述符usb\_device\_descriptor   
200        表示获取配置描述符usb\_config\_descriptor   
300        表示获取字符描述符   
301        iSerialNumber   
302        iProduct   
303        iManufacturer

到这里相信都能把这几个参数与controltransfer方法的参数对应起来了

requestType = bRequestType

request= bRequest

value=wValue

index = wIndex

buffer 发送的数据

length发送数据长度

timeout超时时间

希望能都够帮助一些人