目　录

第一部分:概述

Android SDK 为开发者提供了一套在 Android 平台简易地与映美的打印机通讯的开发包。通讯方式支持 WIFI、Bluetooth。该开发包功能简易，可靠，可扩充，封装了连接打印机，发送、接收打印机指令，图形文件转换为打印机指令等过程，也就是说，用户不必关注一些协议相关的细节和复杂的通讯过程，就可以开发出功能强大的与打印机通讯的软件。打印机指令请参考《映美打印机编程手册》。

Android SDK 支持 Android 3.1 以上。

提示：无线打印 V3.0 新增防丢单功能，通过执行严密的双向传输协议，当出现打印机出现故障、离开无线信号有效范围、通讯网络故障等异常时，能以重打或续打方式实现每份单据的完整性。

第二部分:程序范例

1、导入 RemotePrinter 库文件

下面使用 Eclipse 开发工具来作为示例。

(1) 将 RemotePrinter.jar 拷贝到开发项目的目录下，该演示是放到项目的根目录下。

(2) 选中项目，右键菜单，Build Path -> Configure Build Path…，如下图 2-1 所示：

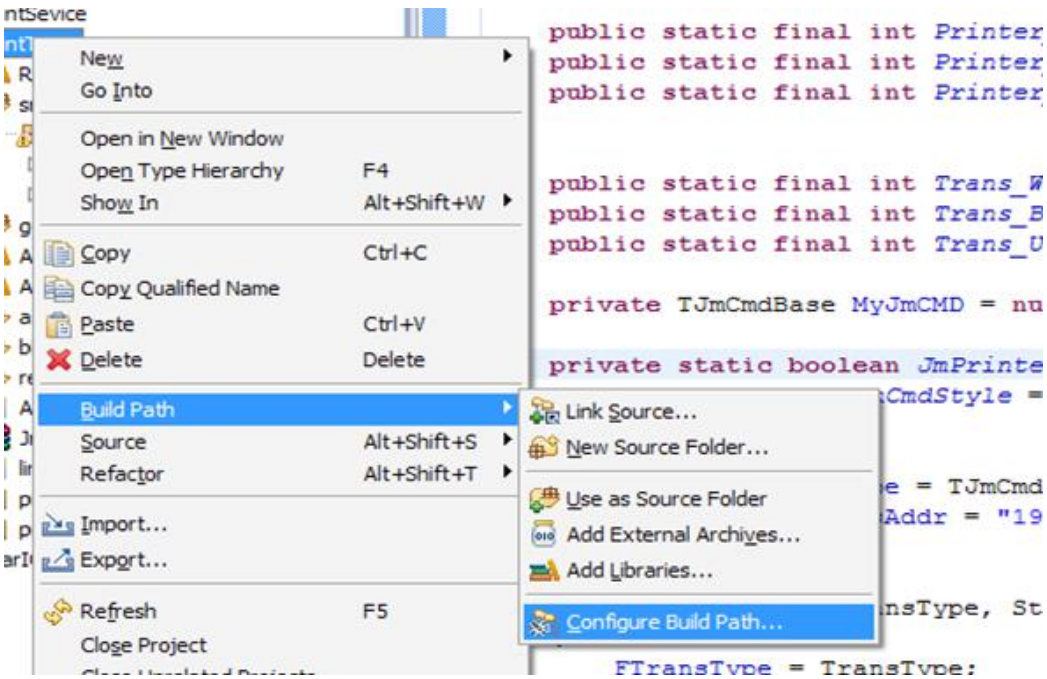

图 2-1

(3) 添加 RemotePrinter.jar 到项目中 (Add JARS…)，如下图 2-2 所示：

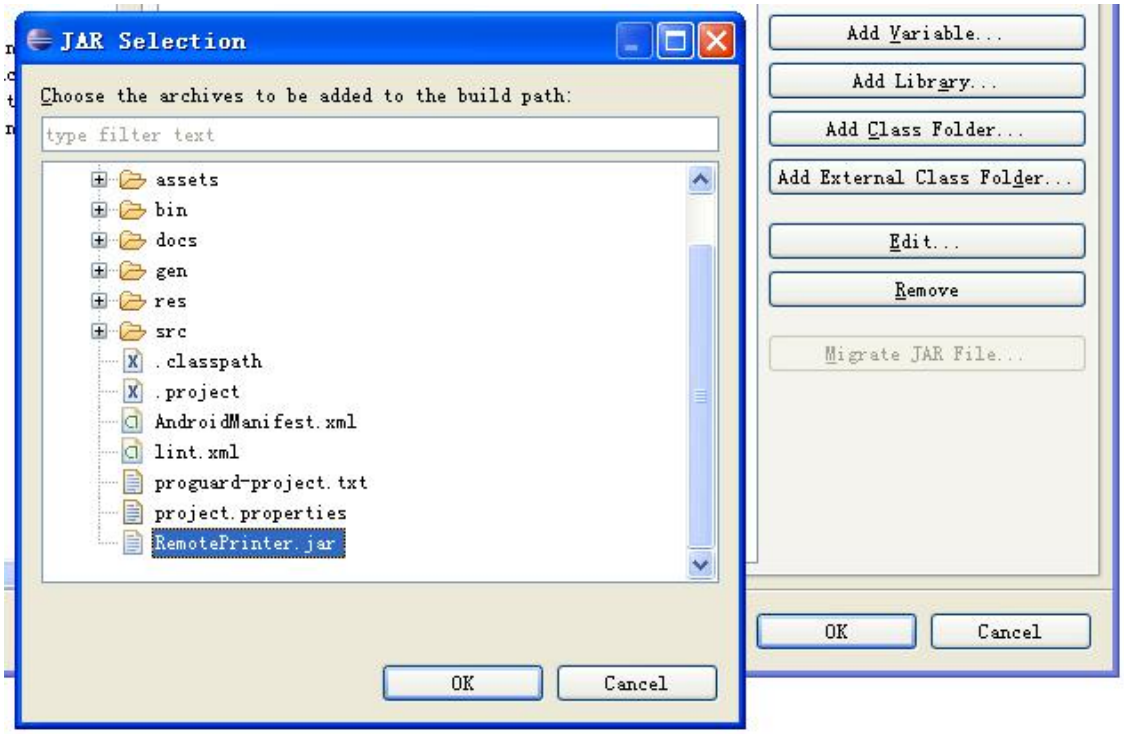

图 2-2

(4) 将 RemotePrinter 移到最前端。如下图 2-3 所示：

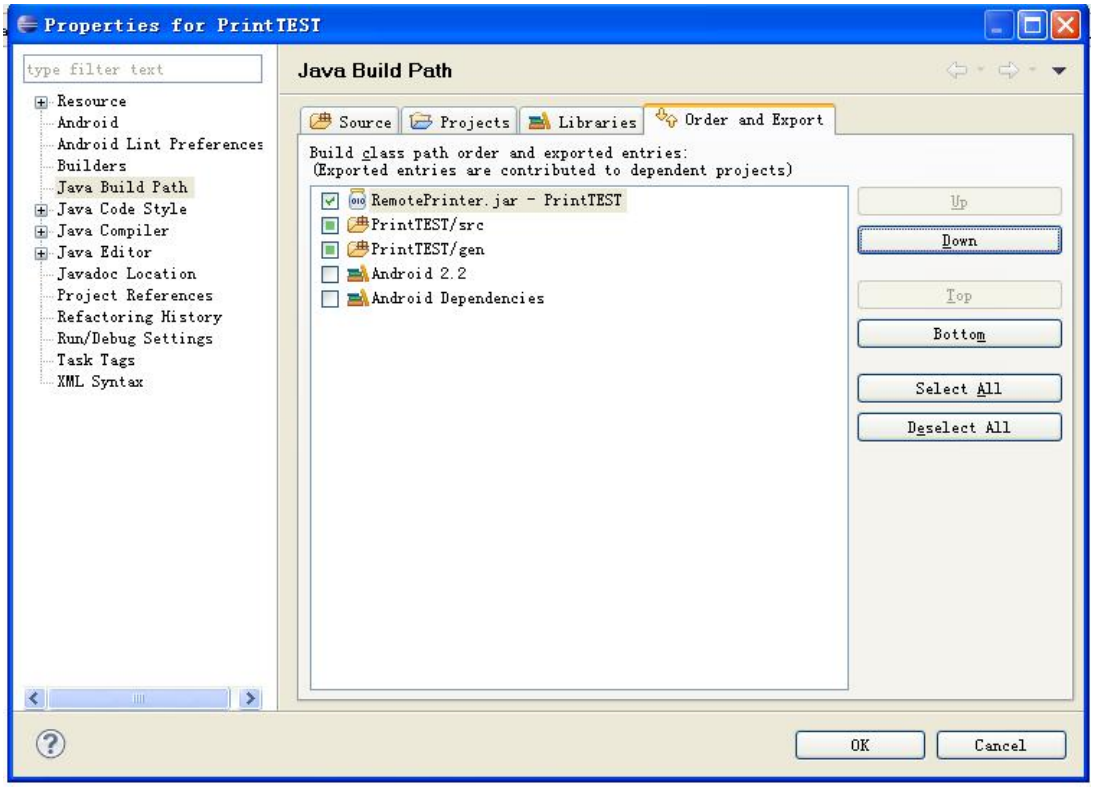

图 2-3

2、简单例子

学习这个 SDK 最简单的方法就是看一个简单的例子。假设我们已经把 JAR 加载到我们的工程中。

2.1 文本打印（两种通讯方式）

1.使用 WiFi 通讯方式

首先获取 wifi 打印机的 IP 地址等信息，使用 RemotePrinter 类的静态方法 getWifiPrinterIP，在回调方法参数中的 DeviceInfo 对象封装了打印机的 IP 地址、打印机类型和 Mac 地址。代码如下：

```
private int searchTime = 24; //查找设备时间 24 秒

RemotePrinter.getWifiPrinterIP(searchTime,new RefleshHandler(){

                @Override
                public void deviceFound(DeviceInfo deviceInfo){

                    String mIP = deviceInfo.ip;//IP地址
                    String mPrinterType = deviceInfo.type;//打印机类型
                    String mMac = deviceInfo.mac;//Mac地址
                }

}
```

假设：IP 为 192.168.43.250,端口号为 9100 。

现在我们要使用打印机打印“Hello World”信息。

（1）传统方式代码如下：

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

String tmpSendData = "Hello Word\r\n";
// 第一步：创建RemotePrinter类实例，传入参数，包含传输数据的方式、打印机的IP地址和端口号。
RemotePrinter MyPrinter = new RemotePrinter(TransType.TRANS_WIFI, "192.168.43.250:9100");
// 第二步：调用 open(false)方法打开单向通讯通道
if (MyPrinter.open(false)) {
 //第三步：对数据进行拆分，每包1K（最大不超过2K）
    List<byte[]> list = ByteArrayUtils.fen(tempSendData.getBytes(),1024);
    for(int i=0;i<list.size();i++){
// 第四步：调用sendData()方法发送数据
      MyPrinter.sendData(list.get(i));
// 第五步：适当延迟(自己根据打印机情况设置)，因为数据发送太快，有些打印机缓冲区小，导致溢出乱码
      Thread.sleep(500);
    }
```

```
} else {
        ……      //打开失败
}
MyPrinter.close();          //关闭通讯通道
MyPrinter = null;              //释放 RemotePrinter实例
}
```

（2）采用防丢单方式打印，不需要对数据进行分包和延迟，但它主要多出两个步骤，在发送数据之前，要调用开始打印任务函数，结束的时候调用打印结束函数，两者之间可以使用一次或多次发送数据函数，以便数据量很大的时候能够分批传输。代码如下（以下代码只是为了说明流程，具体实现参看示例程序）：

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

String tmpSendData = "Hello Word\r\n";
// 第一步：创建RemotePrinter类实例，传入参数，包含传输数据的方式、打印机的IP地址和端口号。
RemotePrinter MyPrinter = new RemotePrinter(TransType.TRANS_WIFI, "192.168.43.250:9100");
// 第二步：调用 open(true)方法打开双向通讯通道
if (MyPrinter.open(true)) {
// 第三步：调用startSendData()开始打印任务
//每一个阶段逻辑上都可以用一个循环来处理，具体实现时不一定用while来实现
    boolean startFlag = myPrinter.startSendData();
    while(!startFlag){
      //实际上是根据错误信息弹出提示框，提示用户是否继续打印？
      Thread.sleep(3000); //模拟对话框时间
    //如果用户选择继续打印，则执行：
      startFlag = myPrinter.continueSendData();
      //如果用户选择取消打印，则执行：
      //return;
    }
//第四步：调用sendData()发送数据(可以多次调用)
    int sendDataLength = myPrinter.sendData(tempData.getBytes());
    boolean sendFlag = false;
    if(sendDataLength == tempData.getBytes().length){
      sendFlag = true;
    }
    while(!sendFlag){
      //实际上是根据错误信息弹出提示框，提示用户是否继续打印？
      Thread.sleep(3000); //模拟对话框时间
    //如果用户选择继续打印，则执行：
      sendFlag = myPrinter.continueSendData();
      //如果用户选择取消打印,由于任务已经开始，要取消打印任务，则执行下面流程：
      //1.先清除打印机缓存
```

```
      //  boolean cleanFlag = myPrinter.cleanPrinterCache();
      //  if(cleanFlag){
      //2.结束打印任务
      //    boolean finishFlag = myPrinter.finishSendData();
      //      if(finishFlag){
     //         //取消任务成功..
     //       }else{
     //         //取消任务失败(这里如果出现的错误是打印中途出错，也算是取消任务成功,返回true)..
    //        }
    //    }else{
    //      //取消任务失败..
    //    }
    //  return; //重新开始
    }
//第五步：调用finishSendData()结束打印任务
    boolean finishFlag = myPrinter.finishSendData();
    while(!finishFlag){
      //实际上是根据错误信息弹出提示框，如果出现的错误是打印中途出错，应该提示用户整单重新打印，而不是续发）否则，续发
      Thread.sleep(3000);
      finishFlag = myPrinter.continueSendData();
      //如果用户选择的是”取消”,结束发送，重新开始
      //return;
    }
    //发送成功
    MyPrinter.close();          //关闭通讯通道，方便别人或自己再次连接
    MyPrinter = null;           //释放 RemotePrinter实例
}
```

2.使用蓝牙通讯方式

使用蓝牙通讯方式跟使用 WIFI 通讯方式是类似的，只是创建 RemotePrinter 实例时传入的参数有点区别，假设打印机的蓝牙 Mac 地址为：74:51:BA:52:07:B4，现在我们要使用打印机打印“Hello World”信息。以传统代码举例：

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

String tmpSendData = "Hello Word\r\n";
// 第一步：创建RemotePrinter类实例，传入参数，包含传输数据的方式、打印机蓝牙Mac地址。
RemotePrinter MyPrinter = new RemotePrinter(TransType.TRANS_BT, "74:51:BA:52:07:B4");
// 第二步：调用 open(false)方法打开单向通讯通道
if (MyPrinter.open(false)) {
```

```
// 第三步：调用sendData()方法发送数据
    MyPrinter.sendData(tmpSendData.getBytes())
} else {
        ……      //打开失败
}
MyPrinter.close();          //关闭通讯通道
MyPrinter = null;              //释放 RemotePrinter实例
}
```

3.使用 USB 通讯方式

使用 USB 方式的时候要创建 UsbPrinter 对象，而不是 RemotePrinter 对象。

注意：使用 USB 通讯方式，Android 系统必须支持 usb printer 驱动。

现在我们要使用打印机打印“Hello World”信息。代码如下（防丢单流程与之前一致或者参看示例代码）：

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

String tmpSendData = "Hello Word\r\n";
// 第一步：创建UsbPrinter类实例，传入Context对象参数
UsbPrinter MyPrinter = new UsbPrinter(context);
// 第二步：调用 openUsbPrinter()方法打开通讯通道,并传递一个超时参数(ms),
// 以及是否开启防丢单标识（true代表开启）
// 在打开过程中，如果打印机设备没有USB权限会请求，当超过设定的时间，即使
// 按确定获取得到权限也属于打开失败，这时候重试即可，或者设定一个比较合理的时间
if (MyPrinter.openUsbPrinter(timeout, flag)) {
// 第三步：调用sendData()方法发送数据
    MyPrinter.sendData(tmpSendData.getBytes())
} else {
        ……      //打开失败
}
MyPrinter.close();          //关闭通讯通道
MyPrinter = null;              //释放 UsbPrinter 实例
}
```

2.2 图形打印

要打印图形，需要把图形转换为图形指令数据，然后将该图形指令数据发送到打印机。由于各种打印机的图形指令数据格式有所不同，所以首先需要确定打印机的类型。

调用 RemotePrinter 类的静态函数 ConvertImage，显式地指定打印机的类型。WiFi 通讯方式代码如下所示：

```
import com.printer.RemotePrinter;
import com.printer.VAR.PrinterType;
import com.printer.VAR.TransType;

// surFileName 为原始图片路径（含名称）。 图片格式为 JPG, PNG,  BMP 等
//指定打印机类型为热敏打印机(注意，这里的打印机类型只是参考。具体是什么类型，在连接成功后通
//过 getPrinterType 函数获取，参看实例程序)
byte[] tmpBuf = null;
tmpBuf = RemotePrinter.ConvertImage(PrinterType.PT_THERMAL, surFileName);
if (tmpBuf != null) { // 如果生成图形指令数据成功
    RemotePrinter myJmPrinter = new RemotePrinter(TransType.TRANS_WIFI,"192.168.43.250:9100");
    //按照2.1节介绍的方式进行打印
    ......
    myJmPrinter.close(); // 关闭通讯通道
    myJmPrinter = null; // 释放RemotePrinter实例
}
```

第三部分:类及函数详细信息

1、类继承关系

```
java.lang.Object
        ↳com.print.printerlib.RemotePrinter

        ↳com.print.printerlib.UsbPrinter

public class RemotePrinter extends java.lang.Object

public class UsbPrinter extends java.lang.Object
```

2、构造函数

2.1 RemotePrinter 构造函数

函数原型

public **RemotePrinter** (TransType transType, String transAddr)

描述

该函数创建 RemotePrinter 的实例，同时设置传输参数。

参数

transType：传输类型参数。见 **表 3-2-1**

transAddr：当选择通讯方式为 WIFI 时，该值为 IP:Port，例如：打印机的 IP 为 192.168.43.250，Port 为 9100，则这里应该传入的参数值为 “192.168.43.250:9100”。当选择通讯方式为 Blue Tooth 时，该值为 Blue Tooth 的地址。

传输参数	值	描　述
TRANS_WIFI	0	通讯方式为 WIFI。
TRANS_BT	1	通讯方式为蓝牙。

表 3-2-1 TransType 值

返回值

如果函数调用成功，返回该类的实例，否则返回 null。

2.2 UsbPrinter 构造函数

函数原型

public **UsbPrinter**(Context context)

描述

该函数创建 UsbPrinter 的实例，同时设置参数。

参数

context：Android 环境上下文对象

回值

如果函数调用成功，返回该类的实例，否则返回 null。

3、通讯函数

3.1 open 函数

函数原型（两个）

1.boolean **open** (boolean flag);

2.boolean **open**(int timeout,boolean flag);

描述

打开通讯通道。第 1 个 open 函数用于无线类型打印机，第二个函数用于 USB 类型打印机。当所有的通讯操作已完成，则调用 close 关闭通讯通道。open 与 close 是一对出现的，在它们之间，可包含一个或多个通讯操作。

参数

timeout：设定获取 usb 权限的等待时间，由于 Android 系统申请 USB 权限时不能及时获得，需要设定一个时间来等待，具体时间视情况而定，有可能因为设置时间过短，导致函数返回 false，但是有可能已经获得权限，再次调用即可。

flag： 开始防防丢单功能的标识。true 代表开启，false 代表不开启。注意，当开启之后，使用 false 并不能切换成原来的方式，只有打印机关机才能复原。

返回值

如果函数调用成功，返回 **true**，否则返回 **false**，要获得更多的错误信息，请调用 getLastErrorCode 和 getMessage。

相关函数

close

3.2 close 函数

函数原型

```
boolean close ();
```

描述

关闭通讯通道。通讯开始时，调用 open 打开通讯通道。当所有的通讯操作已完成，则调用 close 关闭通讯通道。open 与 close 是一对出现的，在它们之间，可包含一个或多个通讯操作。注意需要及时关闭通讯通道，防止过度占用通讯通道让其他打印任务无法进行。

返回值

如果函数调用成功，返回 **true**，否则返回 **false**，要获得更多的错误信息，请调用 getLastErrorCode 和 getMessage。

相关函数

open

3.3 startSendData 函数（在防丢单模式下使用）

函数原型

```
public boolean startSendData();
```

描述

该函数的作用是查看打印机状态是否正常，比如是否有缺纸、开盖、上次打印任务未完成等以及初始化打印机。这是打印任务的**开始阶段**。

返回值

如果函数调用成功，返回 **true**，否则返回 **false**，要获得更多的错误信息，请调用 getLastErrorCode 和 getMessage。当返回 **false** 的时候，可以进行续发操作，详情参看续发函数。

相关函数

continueSendData

3.4 sendData 函数

函数原型

```
public int sendData(byte[] data);
```

描述

该函数的作用是发送数据到打印机。这是打印任务的**发送阶段**。

参数

data： 待发送的数据。byte 类型的数组。传统方式下，数组大小不超过 2K，由开发人员进行分包。具体参看示例。防丢单模式下，不需要进行分包和延时。

返回值

如果函数调用成功，返回发送的字节数，否则返回-1。要获得更多的错误信息，请调用 getLastErrorCode 和 getMessage。发送数据失败时，可以进行续发操作，详情参看续发函数。

相关函数

continueSendData

3.5 finishSendData 函数（在防丢单模式下使用）

函数原型

```
public boolean finishSendData();
```

描述

该函数的作用是结束这次打印任务，并对打印途中是否出错进行判断等。这是打印任务的**结束阶段**。

返回值

如果函数调用成功，返回 true，否则返回 false，要获得更多的错误信息，请调用 getLastErrorCode 和 getMessage。当返回 false 的时候，可以进行续发操作，详情参看续发函数。

相关函数

continueSendData

3.6 continueSendData 函数（在防丢单模式下使用）

函数原型

```
public boolean continueSendData();
```

描述

该函数用来续发开始、发送、结束其中一个阶段。注意，它只能续发相应的一个阶段。开始任务失败时调用这个函数就会续发开发阶段，发送失败时调用这个函数就会续发发送阶段，以此类推。

返回值

如果续发成功，返回 true，否则返回 false，要获得更多的错误信息，请调用 getLastErrorCode 和 getMessage。当返回 false 的时候，可以继续进行续发操作。

相关函数

startSendData

sendData

finishSendData

3.7 cleanPrinterCache 函数（在防丢单模式下使用）

函数原型

```
public boolean cleanPrinterCache();
```

描述

该函数的作用是清除打印机缓冲区中的数据。当打印一半时，打印机出现问题（纸尽等）导致发送阶段失败，可以选择不续发，想要重新开始打印任务，这时候就必须先结束打印任务，即调用 finishSendData 函数。但由于打印机缓冲区还有残留数据，任务是无法结束的，所以得在结束任务之前调用此函数，当清除成功后，结束任务才能进行。

返回值

如果函数调用成功，返回 true，否则返回 false，要获得更多的错误信息，请调用 getLastErrorCode 和 getMessage。

相关函数

finishSendData

4、辅助函数

4.1 图片转换函数

	convertImage (String srcFileName)	convertImage(Bitmap srcImage)
参数	源图片路径（含名称）。该图片格式可为 JPG, PNG, BMP 等常用图片格式。	Bitmap 类型的对象
返回值	byte[]	byte[]

描述

该函数用于转换图片为图形指令数据，调用 sendData 将图形指令数据发送到打印机，从而实现了打印图片的功能。

程序范例：参阅 RemotePrinter 中第二部分的图片打印。

相关函数

RemotePrinter，sendData

4.2 getWifiPrinterIP 函数

函数原型

public static void **getWifiPrinterIP**(int timeout,RefleshHandler handler);

描述

该函数的作用是通过 UDP 广播获取 wifi 打印机 IP、类型、mac 地址信息。

参数

timeout：超时时间，0 为使用默认值,单位为秒。

handler：回调接口，对搜索结果进行处理。

相关函数

cancelGetWifiPrinterIP

4.3 cancelGetWifiPrinterIP 函数

函数原型

public static void **cancelGetWifiPrinterIP**();

描述

该函数的作用是取消搜索 wifi 打印机，在关闭页面之前调用，否则连接不会立即断开。

相关函数

getWifiPrinterIP

4.4 isConnected 函数

函数原型

public boolean **isConnected**();

描述

该函数的作用是查看连接是否正常。

返回值

如果处保持连接状态，返回 **true**，否则返回 **false**，要获得更多的错误信息，请调用 getLastErrorCode 和 getMessage。

相关函数

open

4.5 getUsbPrinterStatus 函数

获取 USB 打印机状态（1 个字节），具体每一位代表的信息如下：

Bit(s)	说明
7..6	保留字段
5	1 = 纸尽，0 = 正常
4	1 = 联机，0 = 脱机
3	1 = 没有错误，0 = 出错
2..0	保留字段

想要知道是否出错判断 bit3 就行，具体哪一种类型错误，这里只能查到两种（bit4 和 bit5）。

函数原型

public byte[] getUsbPrinterStatus(int timeout);

描述

用于查询 Usb 打印机的状态。

参数

timeout：超时时间(ms)

返回值

如果函数调用成功，返回一个字节，否则返回 **null**。

4.6 获取错误信息函数

函数原型

int **getLastErrorCode** ();

描述

getLastErrorCode 函数获取了通讯过程中最后一个错误代码。错误代码是一个 16 位值。开发人员应该在函数返回标识错误代码的值后立即调用 getLastErrorCode。错误代码的具体内容请参阅错误码意义对照表。

返回值

如果函数调用成功，返回通讯过程中最后一个错误代码和错误信息。

程序范例：

```
//创建 RemotePrinter 类的实例，并设置通讯参数(通讯方式:WIFI,  IP: 192.168.43.250,  端口: 9100)
RemotePrinter myPrinter = new RemotePrinter(TransType.TRANS_WIFI, "192.168.43.250:9100");
if (myPrinter.Open()) { // 打开通讯通道,自动获取打印机类型
    ……  //打开之后的操作
} else {
    // 打开通道失败
    int ErrCode  = myPrinter.getLastErrorCode();
    System.out.print("错误码: "+ ErrCode);
}
……
```

4.7 html2PrintData 函数

函数原型

public static byte[] **html2PrintData**(String htmlString);

描述

将 HTML 按照映美所制定的转换协议转换成相应的打印数据。(协议内容参看相应手册)

返回值

HTML 对应的打印数据

4.8 JSON2PrintData 函数

函数原型

public static byte[] **JSON2PrintData**(String jsonString);

描述

将 JSON 按照映美所制定的转换协议转换成相应的打印数据。(协议内容参看相应手册)

返回值

JSON 对应的打印数据

4.9 getPrinterModel 函数

函数原型

public String getPrinterModel()

描述

返回打印机型号。

返回值

打印机型号字符串

附录

附录一 函数列表

函　　数	描　　述
构造函数	
RemotePrinter	创建 RemotePrinter 实例
通讯函数	
open	打开通讯通道。
close	关闭通讯通道。
startSendData	查询打印机状态以及初始化。
sendData	发送数据到打印机。
finishSendData	结束打印任务以及判断中途是否出错。
continueSendData	续发开始、发送、结束其中一个阶段。
cleanPrinterCache	清除打印机缓存。
辅助函数	
convertImage	转换图片为图形打印指令。
getWifiPrinterIP	获取 wifi 打印机类型、IP 和 MAC 地址。
cancelGetWifiPrinterIP	取消搜索 Wifi 打印机。
getLastErrorCode	获取了最后一个错误代码。
isConnected	判断是否与打印机连接。
getLastErrorCode	获取错误码

附录二 错误码意义对照表

错误码	含义	备注
0x0001	Socket 连接失败	
0x0002	获取打印机类型失败	
0x1001	WIFI 连接错误	
0x1002	WIFI 断开连接时出错	
0x1003	WIFI 对象为空	WIFI 连接还没有建立或已断开
0x1004	WIFI 发送数据时出错	
0x1005	WIFI 接收数据时出错	可能超时或打印机无数据返回
0x2001	蓝牙连接错误	
0x2002	蓝牙断开连接时出错	
0x2003	蓝牙对象为空	蓝牙连接还没有建立或已断开
0x2004	蓝牙发送数据时出错	
0x2005	蓝牙接收数据时出错	可能超时或打印机无数据返回
0x4001	图片格式错误	图片格式不支持

0x4002	该文件不存在	
0x4003	转换图片为点阵数据失败	
0x4041	发送错误	当前数据包发送不出去
0x405	发送数据错误	获取打印机类型时错误
0x4051	数据包头错误	包头封装错误
0x4052	数据校验错误	CRC 不一致
0x4053	打印机接收缓冲区满	
0x4054	打印机接收超时	
0x406	包类型错误	打印机返回包类型不对
0x407	重置阶段错误	
0x4071	再次重置错误	
0x408	开始任务阶段错误	
0x409	结束任务阶段错误	
0x501	接收不到数据	可能信号较弱或者打印机断电
0x502	接收不到数据包	收到数据但不是包类型
0x5021	5 秒内没收到数据(1)	数据计时
0x5022	5 秒内没收到数据(2)	收到数据，没找到包头计时
0x5023	5 秒内没收到数据(3)	接收到完整数据包后计时
0x503	2 秒内没收到数据(1)	包序号出错计时
0x5031	2 秒内没收到数据(2)	
0x504	1 秒内没收到数据(1)	残缺数据包计时
0x5041	1 秒内没收到数据(2)	包头不够计时
0x505	接收时 socket 错误	
0x6001	脱机	
0x6002	钱箱打开	
0x6003	上盖打开	
0x6004	纸尽	
0x6005	切刀故障	
0x6006	其他故障	
0x701	切换协议错误	
0x702	清除缓存错误	
0x703	再次清除失败	
0x704	查询状态失败	
0x7041	再次查询状态失败	
0x705	有未完成的打印任务	可能取消任务不成功或者结束错误
0x801	打印途中出现错误	纸尽、上盖打开等

0xFFFE	不支持该函数使用	非法使用防丢单专用函数
0xFFFF	未知错误	