iMinPrinter 开发者文档

文档更新说明

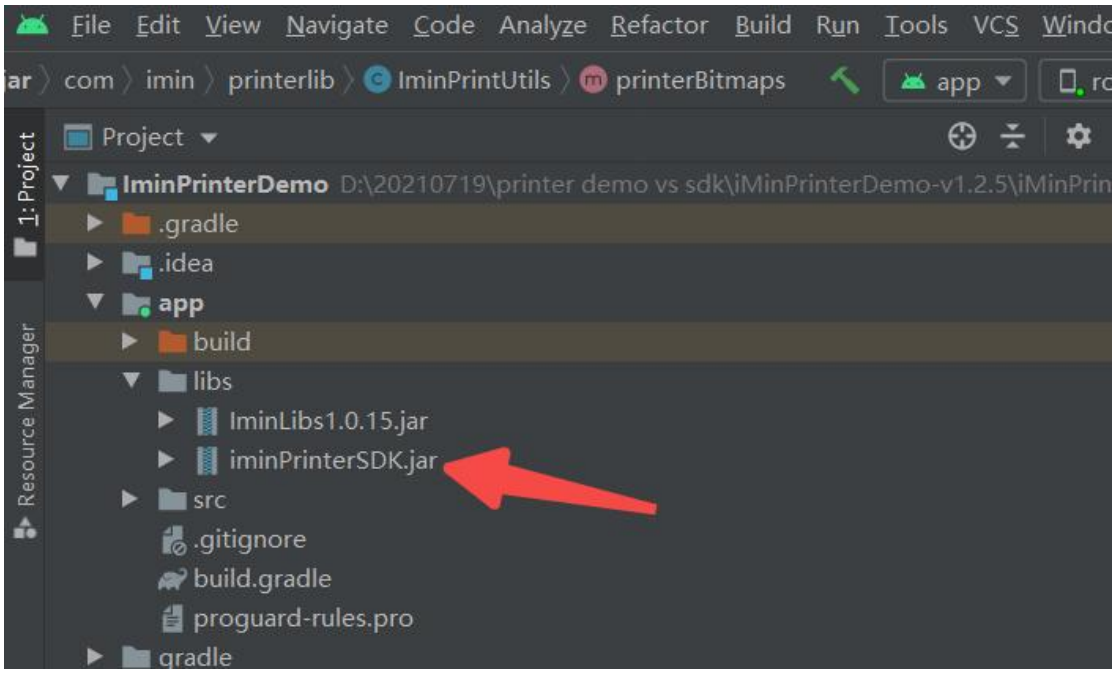
版本号	更新日期	更新内容	撰写人
v1.0.0	2024/1/31	初始版本	谢华炎
v1.0.1	2024/4/18	1. 针对 2.2 获取打印机状态添加说明 2. 针对 3.5t 添加注意说明	谢华炎

目录

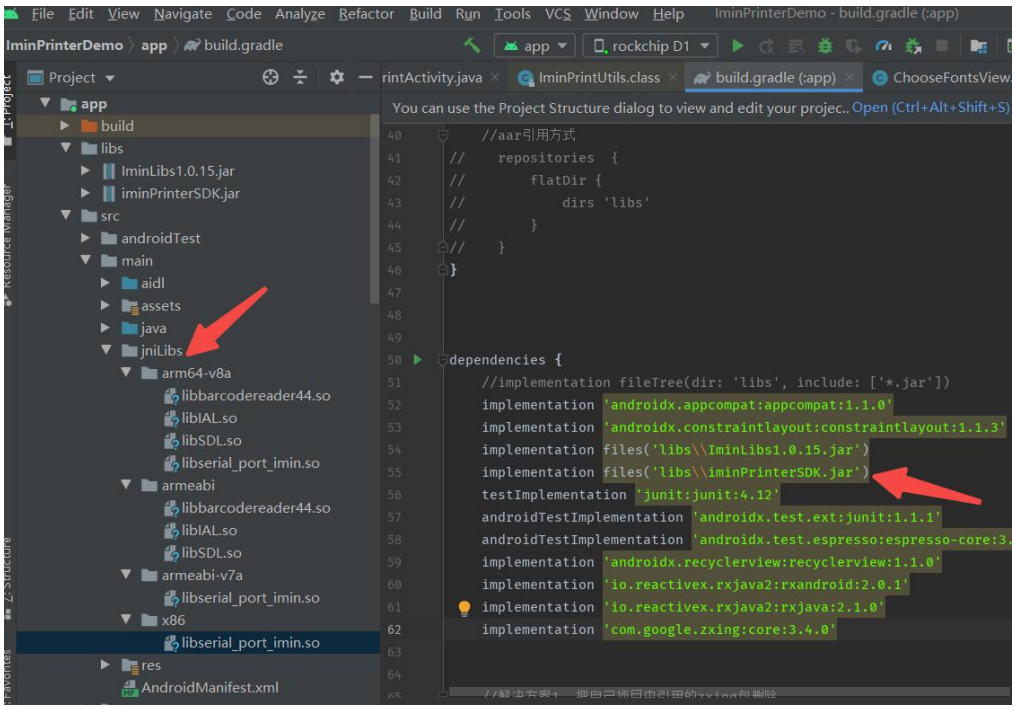
1.通过 SDK.jar 方式接入调用打印机

1.1 集成方式

从 https://oss-sg.imin.sg/docs/en/Printer.html 开发者文档中下载 iminPrintSDK.zip 包，解压缩取出里面的 iminPrintSDK.jar 包之后把 jar 包放置 app-libs 目录下
如图所示：



另需注意：如果是 SPI 打印需要对接 so 库，so 库文件通过 zip 包中的 demo 源码里面 jniLibs 文件夹目录下，对接示意图：



初始化打印机工具类

IminPrintUtils mIminPrintUtils = IminPrintUtils.getInstance(TestPrintActivity.this);

类说明

IminPrintUtils ： iMin 打印接口管理类，包含初始化打印机，执行各个打印方法

2. 接口定义说明

通过上述方法获得 IminPrintUtils 对象后调用如下接口实现自己的打印

2.1 打印机初始化及设置

方法：initPrinter(int printType)

打印机初始化，

int printType （1-3）-> 1.IminPrintUtils.PrintConnectType.USB –> USB

2.IminPrintUtils.PrintConnectType.SPI –> SPI

3.IminPrintUtils.PrintConnectType.Bluetooth –> Bluetooth

示例：

2.1.1.初始化 spi 打印机：

（M2 203,M2 202,M2 Pro）

```
IminPrintUtils.getInstance(TestPrintActivity.this).initPrinter(IminPrintUtils.PrintConnectType.SPI);
```

2.1.2.初始化 USB 打印机：

（D4 系列，D1w，D1,D1 Pro,Falcon 1,Swift 1,M2 Max,S1 系列）

```
IminPrintUtils.getInstance(TestPrintActivity.this).initPrinter(IminPrintUtils.PrintConnectType.USB );
```

2.1.3.初始化蓝牙打印（以上机型均支持）：

```
IminPrintUtils.getInstance(TestPrintActivity.this).initPrinter(IminPrintUtils.PrintConnectType.BLUETOOTH, BluetoothDevice device);
```

备注：BluetoothDevice 通过开启蓝牙之后获取列表中的 BluetoothPrinter 对应的 device 即可

2.2 获取打印机的最新状态

注意：初始化打印机之后如果需要获取打印机状态建议稍微延迟 1s 时间再获取。因为连接打印机的时候中间需要初始化打印机，如果直接获取有可能没有及时响应，导致获取到错误的值

2.2.1 SPI 类型的打印机方法 ：

```
mIminPrintUtils.getPrinterStatus(IminPrintUtils.PrintConnectType.SPI, new Callback() {
                                                    @Override
                                                    public void callback(int status) {
                                                        Log.d("TAG", " print SPI status:" +
status +
                                                        "   PrintUtils.getPrintStatus==   " +
PrintUtils.getPrintStatus());
                                                    }
                                                });
```

iminPrinterSDK-12_V1.2.0_2401251422.jar 版本以及之后的支持，旧版本不支持

```
int status = mIminPrintUtils.getPrinterStatus(IminPrintUtils.PrintConnectType.SPI );
```

返回值说明：

-1：打印机未连接或者打印机未上电

0：打印机工作正常

99：其他错误（开盖缺纸，过热等其他错误）

2.2.2.USB 类型的打印机方法：

```
int status = mIminPrintUtils.getPrinterStatus(IminPrintUtils.PrintConnectType.USB );
```

返回值说明：

D4 系列：

0 打印机正常、1 打印机未连接或未上电、2 打印机和调用库不匹配、3 打印头打开开盖、4 切刀未复位 、5 打印头过热、6 黑标错误、7 缺纸、-1 打印机初始化失败

S1 系列：

0 打印机正常、1 打印机未连接或未上电、-1 打印机初始化失败、8 纸将尽、7 缺纸/开盖

D1 /D1 Pro /M2 Max ：

0 打印机正常、-1 打印机初始化失败、1 打印机未连接或未上电、7 缺纸/开盖

2.3 走纸相关

2.3.1.打印走纸一行

函数：void printAndLineFeed()

说明：默认走纸 10 个垂直点 10*0.125

示例：

```
mIminPrintUtils.printAndLineFeed();
```

2.3.2.打印走纸 n 个 0.125mm

函数：void printAndFeedPaper(int value)

参数：范围 0≤value≤255

将打印缓冲区中的数据打印并进纸 n 个垂直点距。打印结束后，将下一行的开始设定为打印起始位置（ 一个垂直点距为 0.125mm,以下同）

示例：

```
mIminPrintUtils.printAndFeedPaper(100);
```

2.4 切刀（切纸）相关 （仅支持台式机 D4 系列，S1 系列，Falcon 1）

2.4.1 切纸

函数：void partialCut()（半切）

参数：

备注：使用该方法，发送指令打印机即切纸不会有等待打印完成或者停留，容易造成小票内容被切割，或者切纸的位置不对等问题

示例：

```
mIminPrintUtils.partialCut();
```

iminPrinterSDK-12_V1.2.0_2401251422.jar 版本以及之后的支持，旧版本不支持

2.4.2 走纸并切纸（推荐使用）

函数：void partialCutPaper()（半切）

参数：

备注：该切刀方法使用之后会等待小票打印完成之后走一行空行再进行切纸，不会出现内容被切割等问题

示例：

```
mIminPrintUtils.partialCutPaper();
```

2.5 文字打印相关

2.5.1.设置对齐模式

函数：void setAlignment(int alignment)

参数：alignment-> 对齐方式：0->居左 ， 1->居中，2->居右

备注：全局方法，对之后执行的文本打印有影响，使用完之后需要取消相关设置。

示例：

```
mIminPrintUtils.setAlignment(0);
```

2.5.2.设置字体大小

函数：void setTextSize(int size)

参数：0<size

初始化默认字体大小为:28px

备注：全局方法，对之后的文本打印有影响，使用完之后需要取消设置，调整字体大小会影响字符宽度，每行字符数量也会随之改变

示例：

```
mIminPrintUtils.setTextSize(28);
```

2.5.3 设置字体样式

函数：void setTextTypeface(Typeface typeface)

参数：typeface->常规字体类型 Typeface.DEFAULT-> 等宽字体类型 Typeface.MONOSPACE，->黑体字体类型 Typeface.DEFAULT_BOLD -> sans serif 字体类型 Typeface.SANS_SERIF -> serif 字体类型 Typeface.SERIF

备注：全局方法，对之后的文本打印有影响，使用完之后需要取消设置

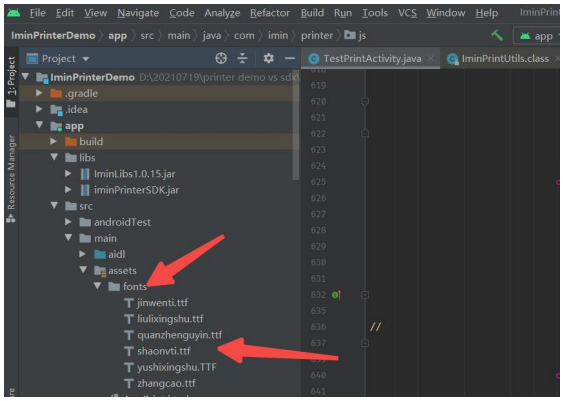
示例：

```
mIminPrintUtils.setTextTypeface(Typeface.DEFAULT);
```

支持自定义字体，调用放需要把字体库放置 assets 目录下

通过 Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/zhangcao.ttf");获取到 typeface 对象即可

设置字体图如图：



示例：

```
Typeface typeface = Typeface.createFromAsset(getAssets(), "fonts/zhangcao.ttf");
mIminPrintUtils.setTextTypeface(typeface );
mIminPrintUtils.setTextTypeface(Typeface.DEFAULT);
```

2.5.4.设置字体风格

函数：`void setTextStyle(int style)`

参数：style->粗体 Typeface.BOLD->粗斜体 Typeface.BOLD_ITALIC->斜体 Typeface.ITALIC->常规 Typeface.NORMAL

默认初始化设置 Typeface.NORMAL

备注：全局方法，对之后的文本打印有影响，使用完之后需要取消设置

示例：

```
mIminPrintUtils.setTextStyle(Typeface.NORMAL);
```

2.5.5.设置打印文字之间的行间距

函数：`void setTextLineSpacing(float space)`

参数：0≤space≤255

初始化默认 1.0f

备注：全局方法，对之后的文本打印有影响，使用完之后需要取消设置

示例：

```
mIminPrintUtils.setTextLineSpacing(1.0f);
```

2.5.6.设置打印内容的宽度

函数：`void setTextWidth(int width)`

参数：0≤width≤576

58mm 的纸张最适宜的宽度 384,80mm 的纸张最优宽度值 576

默认：58mm 的纸张 384,80mm 的纸张 576

备注：全局方法，对之后的文本/图片打印有影响，使用完之后回复默认设置，如果输入其他数值可能会影响打印效果

示例：

```
mIminPrintUtils.setTextWidth(576);
```

2.5.7 打印文本图片

函数：`void printText(String text)`

参数：text 需要打印的内容

备注：如果 text 后面添加“\n”结尾表示即时打印，如果文本后面不加“\n”标识，则会进入缓存，待缓存满后或者下一个打印的内容开始前才打印

示例：

```
mIminPrintUtils.printText("PrinterTestContent \n");
```

2.5.8 打印文本图片添加单个文本图片的对齐方式

函数：void printText(String text, int alignment)

参数：text 打印的内容，需要加“\n”结尾

Alignment->对齐方式：0->居左 ， 1->居中，2->居右 只针对单条打印文本图片内容

备注：如果 text 后面添加“\n”结尾表示即时打印，如果文本后面不加“\n”标识，则会进入缓存，待缓存满后或者下一个打印的内容开始前才打印；Alignment 只针对单条打印文本图片内容

示例：

```
mIminPrintUtils.printText("PrinterTestContent \n",0);
```

2.5.9 打印文本图片反白效果

函数：void printAntiWhiteText(String text)

参数：text 打印的内容，需要加“\n”结尾

备注：如果 text 后面添加“\n”结尾表示即时打印，如果文本后面不加“\n”标识，否则不会进行反白打印，内容会进入缓存，做普通打印

示例：

```
mIminPrintUtils.printAntiWhiteText(“ORDER NOTE:no\n”);
```

2.6 表格打印

函数：mIminPrintUtils.printColumnsText(String[] colTextArr, int[] colWidthArr, int[] colAlign, int[] size);

参数：

colTextArr ->各列文本字符串数组

colWidthArr->各列宽度权重即各列所占比

colAlign->各列对齐方式：0 居左, 1 居中, 2 居右

Size->各列文本字符串的字体大小

示例：

```
mIminPrintUtils.printColumnsText(new String[]{"1","iMin","iMin"},new int[]{1,2,1},new int[]{1,0,2} , new int[]{26,26,26} );
```

2.7 一维码的打印

2.7.1. 设置一维码的宽度

函数：void setBarCodeWidth(int width)

参数：width 条码宽度级别 2<=width<=6，如果不设置则默认条码宽度级别为 3

示例：

```
mIminPrintUtils.setBarCodeWidth(3);
```

2.7.2.设置一维码的高度

函数：void setBarCodeHeight(int height)

参数：height－>条形码高度 1<=height<=255，每 8 点为 1mm，如果不设置默认条形码的高度为 100

示例：

```
mIminPrintUtils.setBarCodeHeight(100);
```

2.7.3.设置打印一维码的文字的位置

函数：void setBarCodeContentPrintPos(int position)

参数：position->文字位置（0 - 3）：//如果不设置，默认 0

0 -> 不打印文字

1 -> 文字在条码上方

2 -> 文字在条码下方

3 -> 条码上下方均打印

示例：

```
mIminPrintUtils.setBarCodeContentPrintPos(2);
```

2.7.4.打印一维码

函数：void printBarCode(int barCodeType, String barCodeContent) throws UnsupportedEncodingException

参数：barCodeType -> 条码类型 ；barCodeContent->条码内容

条形码类型（0-6,73,8）	支持的条形码内容长度	支持的 ASCII 代码范围
0 --> UPC-A	条形码内容长度 = 11,12	48 ≤range≤ 57
1 --> UPC-E	条形码内容长度 = 11,12	48 ≤range≤ 57
2 --> JAN13 / EAN13	条形码内容长度= 12,13	48 ≤range≤ 57
3 --> JAN8 / EAN8	条形码内容长度 = 7	48 ≤range≤ 57
4 --> CODE39	条形码内容长度 >=1	48≤range≤57,65≤range≤90, range= 32, 36, 37, 42, 43, 45, 46, 47
5 --> ITF	条形码内容长度 >=2	48 ≤range≤ 57
6 --> CODABAR	条形码内容长度 >=2	48≤range≤57, 65≤range≤68, 97≤range≤100, range = 36, 43, 45, 46, 47, 58
73 -->CODE128，8-->CODE128	条形码内容长度 >=2	0≤range≤127

示例：

```
mIminPrintUtils.printBarCode(4, "123456");
mIminPrintUtils.printBarCode(73, "{A1456AAA");//CODE128 A
mIminPrintUtils.printBarCode(73, "{B12CAa--");//CODE128 B
mIminPrintUtils.printBarCode(73, "{C009999789101");//CODE128 C
```

2.7.5.打印一维码并设置对齐方式

函数：void printBarCode(int barCodeType, String barCodeContent, int alignmentMode) throws UnsupportedEncodingException

参数：barCodeType 同上，barCodeContent 同上，

alignmentMode –>（0-2）

0 -> 居左 ， 1 -> 居中 ， 2 -> 居右

示例：

```
mIminPrintUtils.printBarCode(4, "123456",1);
mIminPrintUtils.printBarCode(73, "{A1456AAA",2);//CODE128 A
mIminPrintUtils.printBarCode(73, "{B12CAa--", 1);//CODE128 B
mIminPrintUtils.printBarCode(73, "{C009999789101",1);//CODE128 C
```

备注：条码种类不同有如下区别

编码	说明
code39	最长打印 13 个数字
UPC-E	最长打印 12 个数字
UPC-A	最长打印 12 个数字
JAN13 / EAN13	最长打印 13 个数字
JAN8 / EAN8	要求 8 位数字（最后一位校验位），有效长度 8 个数字
ITF	要求输入数字，且有效小于 14 位，必须是偶数位
CODABAR	要求 0-9 及 6 个特殊字符，最长打印 18 个数字
CODE128	Code128 分三类： A 类：包含大写字母、数字、标点等； B 类：大小写字母，数字； C 类：纯数字，复数字符，若为单数位，最后一个将忽略； 接口默认使用B 类编码，若要使用A 类、C 类编码需在内容前面加“{A”、“{C”，例 如：“{A2344A”，”{C123123”，”{A1A{B13B{C12”。

2.8 二维码的打印

2.8.1. 设置二维码的大小

函数：void setQrCodeSize(int level)

参数：level -> 二维码块大小，单位：点，1 <= level <= 13，默认设置 9

示例：

```
mIminPrintUtils.setQrCodeSize(2);
```

2.8.2. 设置二维码纠错

函数：void setQrCodeErrorCorrectionLev(int level)

参数：level -> 48 <= level <= 51，默认设置 51

示例：

```
mIminPrintUtils.setQrCodeErrorCorrectionLev(48);
```

2.8.3. 设置条码和二维码的左边距

函数：void setLeftMargin(int marginValue)

参数：marginValue -> 0< marginValue <576，默认设置 0

示例：

```
mIminPrintUtils.setLeftMargin(0);
```

备注：全局方法，设置之后对于后面的内容均有效

2.8.4.打印二维码

函数：void printQrCode(String qrStr)

参数：qrStr->二维码的内容

示例：

```
mIminPrintUtils.printQrCode("123456");
```

2.8.5.打印二维码添加对齐方式

函数：void printQrCode(String qrStr, int alignmentMode)

参数：qrStr->二维码的内容；alignmentMode –>（0-2）

0 -> 居左 ， 1 -> 居中 ， 2 -> 居右

示例：

```
mIminPrintUtils.printQrCode("123456", 0);
```

2.9 设置纸张规格

2.9.1.设置纸张规格

函数：void setPageFormat(int style)

参数：style->（0-1）

0->80mm，（D4 系列机型，S1，Falcon 1（设置 80mm）,D1w）

1->58mm （M2 系列，Swift 1，D1，D1 Pro）

示例：

```
mIminPrintUtils.setPageFormat(type);
```

备注：全局方法，初始化打印机设置一次即可

3.0 打印图片相关

3.0.1.打印图片

函数：void printSingleBitmap(Bitmap bitmap)

参数：bitmap->位图对象

示例：

```
mIminPrintUtils.printSingleBitmap(bitmap);
```

备注：该方法添加对大图片进行质量压缩的功能，如果 bitmap 需要重复使用，建议重新生成 bitmap 对象

3.0.2.打印图片设置对齐方式

函数：void printSingleBitmap(Bitmap bitmap, int alignmentMode)

参数：bitmap->图片对象；alignmentMode->（0-2）

0 -> 居左 ， 1 -> 居中 ， 2 -> 居右

示例：

```
mIminPrintUtils.printSingleBitmap(bitmap,1);
```

备注：该方法添加对大图片进行质量压缩的功能，如果 bitmap 需要重复使用，建议重新生成 bitmap 对象

3.0.3.打印多个位图

函数：void printMultiBitmap(List<Bitmap> bitmaps)

参数：bitmaps -> 位图列表

示例：

```
mIminPrintUtils.printMultiBitmap(bitmaps);
```

3.0.4.打印多个位图，并设置对齐方式

函数：void printMultiBitmap(List<Bitmap> bitmaps, int alignmentMode)

参数：bitmaps -> 位图列表；alignmentMode->（0-2）

0 -> 居左 ， 1 -> 居中 ， 2 -> 居右

示例：

```
mIminPrintUtils.printMultiBitmap(bitmaps, 1);
```

3.0.5.打印位图自带处理图片功能

函数：void printSingleBitmapBlackWhite(Bitmap bitmap)

参数：bitmap->位图对象

示例：

```
mIminPrintUtils.printSingleBitmapBlackWhite(bitmap);
```

备注：该方法添加对大图片进行质量压缩的功能，如果 bitmap 需要重复使用，建议重新生成 bitmap 对象

3.0.6.位图方式打印 19 位一维码

函数：void printBarCodeToBitmapFormat(String barCodeContent,int width,int height,int codeFormat)

参数：barCodeContent->一维码内容

width -> 一维码宽度，80mm 的纸张建议 1300/1200 最大不超过 1300

height-> 一维码高度，默认 120，条形码高度 1<=height<=255，每 8 点为 1mm

codeFormat->条码类型

CodeFormat.AZTEC -> BarcodeFormat.AZTEC

CodeFormat.CODABAR -> BarcodeFormat.CODABAR

CodeFormat.CODE_39 -> BarcodeFormat.CODE_39

CodeFormat.CODE_93 -> BarcodeFormat.CODE_93

CodeFormat.CODE_128 -> BarcodeFormat.CODE_128

CodeFormat.DATA_MATRIX -> BarcodeFormat.DATA_MATRIX

CodeFormat.EAN_13 -> BarcodeFormat.EAN_13

CodeFormat.ITF -> BarcodeFormat.ITF

CodeFormat.MAXICODE -> BarcodeFormat.MAXICODE

CodeFormat.PDF_417 -> BarcodeFormat.PDF_417

CodeFormat.QR_CODE -> BarcodeFormat.QR_CODE

CodeFormat.RSS_14 -> BarcodeFormat.RSS_14

CodeFormat.RSS_EXPANDED -> BarcodeFormat.RSS_EXPANDED

CodeFormat.UPC_A -> BarcodeFormat.UPC_A

CodeFormat.UPC_E -> BarcodeFormat.UPC_E

CodeFormat.UPC_EAN_EXTENSION -> BarcodeFormat.UPC_EAN_EXTENSION

示例：

```
mIminPrintUtils.printBarCodeToBitmapFormat("11110AQ899015859344",1300,120,
CodeFormat.CODE_128);
```

3.1 打印双二维码相关

备注：目前支持双二维码的机型（ M2-203， M2 Pro,，M2 Max， D1）

3.1.1.设置双二维码的大小

函数：void setDoubleQRSize(int size)

参数：size -> 1<= size <= 8

示例：

```
mIminPrintUtils.setDoubleQRSize(5);
```

3.1.2.设置双二维码偏移值

函数：第一个二维码： void setDoubleQR2Level(int level)，

第二个二维码：void setDoubleQR2Level(int level)

参数：level -> 0-3，默认值 2

示例：

```
mIminPrintUtils.setDoubleQR1Level(1);
mIminPrintUtils.setDoubleQR2Level(2);
```

3.1.3.设置双二维码中二维码一的左边距

函数：void setDoubleQR2MarginLeft(int marginValue)

参数：marginValue -> 0<= marginValue <= 200

示例：

```
mIminPrintUtils.setDoubleQR1MarginLeft(-80);
```

3.1.4.设置双二维码中二维码二的左边距

函数：void setDoubleQR2MarginLeft(int marginValue)

参数：marginValue -> 0<= marginValue <= 200

示例：

```
mIminPrintUtils.setDoubleQR2MarginLeft(-80);
```

3.1.5.设置双二维码中二维码一的版本

函数：void setDoubleQR1Version(int version)

参数：version（0-40）-> 0<= version <=40 默认设置值 0

示例：

```
mIminPrintUtils.setDoubleQR1Version(0);
```

3.1.6.设置双二维码中二维码二的版本

函数：void setDoubleQR2Version(int version)

参数：version（0-40）-> 0<= version <=40 默认设置值 6

示例：

```
mIminPrintUtils.setDoubleQR2Version(6);
```

3.1.7.打印双二维码

函数：void printDoubleQR(String qrCode1, String qrCode2)

参数：colTextArr-> 双二维码的内容

示例：

```
mIminPrintUtils.printDoubleQR("www.iMin.sg", "www.google.com");
```

3.2 仅 TF1/Falcon 1 支持的功能

3.2.1.获取切刀次数

函数：int getPrintCutterNumber()

参数：

示例：

```
mIminPrintUtils.getPrintCutterNumber();
```

3.2.2.获取打印机的打印长度

函数：int getPrinterPaperDistance()

参数：

示例：

```
mIminPrintUtils.getPrinterPaperDistance();
```

3.2.3.获取序列号

函数：int getPrinterSerialNumber()

参数：

示例：

```
mIminPrintUtils.getPrinterSerialNumber();
```

3.3 内外置打印机切换

函数：void setInitIminPrinter(boolean initIminPrinter)

参数：initIminPrinter->true 默认 imin 内部打印设备；

false 获取 usb 设备列表的默认连接第一个

示例：

```
mIminPrintUtils.setInitIminPrinter(true);
```

iminPrinterSDK-12_V1.2.0_2401251422.jar 版本以及之后的支持，旧版本不支持

3.4 重置数据

备注：该方法在首次初始化打印机之前调用，或者关闭程序的时候调用，打印过程中不需要调用，如果打印过程中调用该方法则可能导致打印中断，出现打印内容不完整或者出现乱码的情况

函数：void resetDevice()

参数：

示例：

```
mIminPrintUtils.resetDevice();
```

3.5 断开 SDK 连接

备注：该方法在关闭程序的时候调用，打印过程中不需要调用，如果打印过程中调用该方法则整个打印中断并且连接关闭

注意：断开 sdk 连接的时候建议把当前获取到的 mIminPrintUtils 置空即 mIminPrintUtils=null;便于下一次使用 sdk 的时候初始化

函数：void disConnectDevices()

参数：

示例：

```
mIminPrintUtils.disConnectDevices();
```

3.6 打印日志开关

备注：该方法在关闭程序的时候调用，打印过程中不需要调用，如果打印过程中调用该方法则整个打印中断并且连接关闭

函数：void setIsOpenLog(int open)　//开启日志打印 1 开启　0 关闭　，SDK 默认 0

参数：

示例：

```
mIminPrintUtils.setIsOpenLog(0);
```

3.7 注意

iminPrinterSDK-12_V1.2.0_2401251422.jar 版本以及之后版本的 SDK，只针对 imin/一敏 2023 年 12 月 21 日之后的 ROM 进行完全适配如果客户对接的 ROM 版本不是最新的，麻烦申请升级更新

3.8 事务打印

事务打印模式适用于需要控制打印内容并得到打印结果反馈(是否打印出小票)的需求，此模式相当于建立一个事务队列缓冲区，
当开发者进入事务打印模式，将开启一个事务队列，可以向其中增加打印方法，此时打印机不会立刻打印内容，当提交事务后，
打印机才会依次执行队列中的任务，执行结束将获得此次事务的结果反馈。
事务打印注意事项：
1.当进入缓冲（事务）打印后，提交打印成功将返回成功结果，但遇到打印机异常如缺纸、过热等，将会丢掉本次提交事
务中所有指令任务，同时反馈异常，即当一单任务执行前或执行中打印机异常，则此单不会打出；
2.当指令打印和缓冲（事务）打印交替使用时，如果打印机异常，不会清除指令打印的内容！
3.进入事务打印模式后，但不会立即打印输出，会将输出内容缓存到缓存区，当调用exitPrinterBuffer()或commitPrinterBuffer()等方法才会进行打印输出。
4.事务打印结果回调在 IPrinterCallback方法中的 onPrintResult(intcode,Stringmsg)方法（会有一定的耗时，要等物理打印出纸，不推荐单行频繁使用事务打印，将会影响打印速度，推荐整张小票使用事务打印），对应返回 code 如下：
a)0!打印成功，msg 为“Print successful”；
b)1/2 /3!打印失败；

3.8.1 进入事务模式
函数：void enterPrinterBuffer(boolean clean);
参数：
clean->是否清除事务队列的数据　默认 false
true->清除事务队列的未打印的数据，
false->不清除事务队列未打印的数据，下次提交事务一起打印出来
示例：

```
IminPrintUtils.getInstance(BufferActivity.this).enterPrinterBuffer(false);
```

3.8.2 提交事务
函数：void　commitPrinterBuffer();
参数：
示例：

```
IminPrintUtils.getInstance(BufferActivity.this).commitPrinterBuffer();
```

3.8.2 提交事务有返回值
函数：void　commitPrinterBuffer(PrintResultCallback callback);
参数：PrintResultCallback　事务打印结果返回回调
返回值：result1 -> 0　　正常

1 过热
2 开盖或者缺纸
3 其他错误

示例：

```
IminPrintUtils.getInstance(BufferActivity.this).commitPrinterBuffer(new PrintResultCallback() {
                        @Override
                        public void printResult(int result1) {
                            Log.d("IminPrintUtils_Buffer", "commitPrinterBuffer printResult: " +
result1);
                            //showToast(result+ "");
                            if (stringBuilder != null){
                                stringBuilder.append(result1+"\t");
                                resultTV.setText(stringBuilder.toString());
                            }

                        }
                    });
```

3.8.4 退出事务打印

函数：void exitPrinterBuffer(boolean commit);

参数：

commit->是否清除事务队列的数据 true 清除事务队列的缓存数据，false 不清楚队列

示例：

```
IminPrintUtils.getInstance(BufferActivity.this).exitPrinterBuffer(false);
```

3.8.5 退出事务打印有返回值

函数：void exitPrinterBuffer(boolean commit,PrintResultCallback callback);

参数：

commit->是否清除事务队列的数据 true 清除事务队列的缓存数据，false 不清楚队列

参数：PrintResultCallback 事务打印结果返回回调

返回值：result1 -> 0 正常
1 过热
2 开盖或者缺纸
3 其他错误

示例：

```
IminPrintUtils.getInstance(BufferActivity.this).exitPrinterBuffer(true, new PrintResultCallback()
{
                        @Override
                        public void printResult(int result1) {
                            if (stringBuilder != null){
                                stringBuilder.append(result1+"\t");
                                resultTV.setText(stringBuilder.toString());
                            }
```

整个事务打印示例：

```
IminPrintUtils.getInstance(BufferActivity.this).enterPrinterBuffer(false);//进入事务模式，此后所有命令不会立刻输出
printText(/*something*/)
printBitmap(/*bitmapresource*/)
//……其它打印相关方法——打印一些内容
commitPrinterBuffer()/commitPrinterBufferWithCallback(callback)//提交一次事务，此时打印机将开始打印，当打印成功或失败将在 callback 中返回
……等待上一次事务的返回
printText(/*something*/)
printBitmap(/*bitmapresource*/)
//……其它打印相关方法——可以选择等待或不等待上一次事务的返回继续打印内容
commitPrinterBuffer()/commitPrinterBufferWithCallback(callback)//继续提交下一次事务，此时打印机将继续打印
exitPrinterBuffer(true)/exitPrinterBufferWithCallback(true,callback)//退出事物模式时调用，如果在上一次提交后又输入新的数据则会继续打印否则不打印
```