

# Android ZPL 编程手册

**v3.4.1**

（注：浏览时请使用 PDF 左侧导航栏）

# 1. 介绍

这个手册介绍怎么通过 ZPL 指令实现标签打印。常量定义再 ZPLConst 类中。

## 2. ZPLPrinter

### 2.1. ZPLPrinter

构造函数，创建打印对象。

ZPLPrinter(IDeviceConnection connection)

#### 【参数】

➤ connection

连接对象，可通过 POSConnect.createDevice(deviceType)获取。

### 2.2. addStart

此方法用于标签的开头

ZPLPrinter addStart()

#### 【返回值】

ZPLPrinter

### 2.3. addEnd

标签格式的结束。调用此方法后，即会打印标签。

ZPLPrinter addEnd()

#### 【返回值】

ZPLPrinter 对象

### 2.4. addText

文本打印

ZPLPrinter addText(int x, int y, String fontName, String rotation, int sizeW, int sizeH, String

```
content)
ZPLPrinter addText(int x, int y, char fontName, int sizeW, int sizeH, String content)
ZPLPrinter addText(int x, int y, char fontName, String content)
ZPLPrinter addText(int x, int y, String content)
```

【参数】

- x  
文本的起始 x 值
- y  
文本的起始 y 值
- font  
文本的字体类型，默认 FNT\_F。

变量	基础字体尺寸，高 x 宽
FNT_A	9 x 5
FNT_B	11x7
FNT_C、FNT_D	18x10
FNT_E	28x15
FNT_F	26x13
FNT_G	60x40
FNT_O	15*12

其他字库字体，请使用自定义名。

- sizeW  
文字有效宽度，默认为基础尺寸。请使用基础尺寸的整数倍。
- sizeH  
文字有效高度。默认为基础尺寸。请使用基础尺寸的整数倍。
- rotation  
顺时针旋转角度，默认 ROTATION\_0

变量	描述
ROTATION_0	不旋转
ROTATION_90	顺时针旋转 90 度
ROTATION_180	顺时针旋转 180 度
ROTATION_270	顺时针旋转 270 度

- content  
文本内容

【返回值】

ZPLPrinter 对象

## 2.5. addTextBlock

该方法用于打印具有定义的宽度和高度的文本块。文本块带有自动换行功能。

如果文本超出文本块高度，文本将被截断。

ZPLPrinter addTextBlock(int x, int y, char fontName, String rotation, int sizeW, int sizeH, int textblockWidth, int textblockHeight, String content)

ZPLPrinter addTextBlock(int x, int y, char fontName, int sizeW, int sizeH, int textblockWidth, int textblockHeight, String content)

ZPLPrinter addTextBlock(int x, int y, char fontName, int textblockWidth, int textblockHeight, String content)

ZPLPrinter addTextBlock(int x, int y, int textblockWidth, int textblockHeight, String content)

### 【参数】

➤ x

文本的起始 x 值

➤ y

文本的起始 y 值

➤ font

文本的字体类型，默认 FNT\_F。

变量	基础字体尺寸，高 x 宽
FNT_A	9 x 5
FNT_B	11x7
FNT_C、FNT_D	18x10
FNT_E	28x15
FNT_F	26x13
FNT_G	60x40
FNT_O	15*12

其他字库字体，请使用自定义名。

➤ sizeW

文字有效宽度，默认为基础尺寸。请使用基础尺寸的整数倍。

➤ sizeH

文字有效高度。默认为基础尺寸。请使用基础尺寸的整数倍。

➤ rotation

顺时针旋转角度，默认 ROTATION\_0

变量	描述
ROTATION_0	不旋转
ROTATION_90	顺时针旋转 90 度
ROTATION_180	顺时针旋转 180 度
ROTATION_270	顺时针旋转 270 度

➤ textblockWidth

文本块宽度 （以点为单位）

- `textblockHeight`  
文本块高度 （以点为单位）
- `content`  
文本内容

#### 【返回值】

ZPLPrinter 对象

## 2.6. setCustomFont

设置自定义字体。机器断电之后，设置会失效。

ZPLPrinter setCustomFont(String font, char alias, int codePage)

#### 【参数】

- `font`  
字库字体名及后缀，例如:LZHONGHEI.TTF
- `alias`  
字体别名，对应 addText 中的 `fontName`。范围:A 至 Z 和 0 至 9。
- `codePage`  
字符编码

变量	描述
CODE_PAGE_UTF8	Unicode （UTF-8 编码） - Unicode 字符集
CODE_PAGE_UTF16	Unicode （UTF-16 Big-Endian 编码） - Unicode 字符集
CODE_PAGE_UTF16_2	Unicode （UTF-16 Little-Endian 编码） - Unicode 字符集
CODE_PAGE_USA1	单字节编码 - 美国 1 字符集
CODE_PAGE_USA2	单字节编码 - 美国 2 字符集
CODE_PAGE_UK	单字节编码 - 英国字符集
CODE_PAGE_NL	单字节编码 - 荷兰字符集
CODE_PAGE_DK	单字节编码 - 丹麦 / 挪威字符集
CODE_PAGE_SWEDE	单字节编码 - 瑞典 / 芬兰字符集
CODE_PAGE_GER	单字节编码 - 德国字符集
CODE_PAGE_FR1	单字节编码 - 法国 1 字符集
CODE_PAGE_FR2	单字节编码 - 法国 2 字符集
CODE_PAGE_ITA	单字节编码 - 意大利字符集
CODE_PAGE_ES	单字节编码 - 西班牙字符集
CODE_PAGE_JA	单字节编码 - 日本 （包含日元符号的 ASCII）字符集

#### 【返回值】

ZPLPrinter 对象

## 2.7. setPrinterWidth

设置打印宽度

ZPLPrinter setPrinterWidth(int width)

### 【参数】

➤ width

纸张的宽度，单位为点。

### 【返回值】

ZPLPrinter 对象

## 2.8. setLabelLength

设置标签长度

ZPLPrinter setLabelLength(int length)

### 【参数】

➤ length

标签的长度，单位为点。

### 【返回值】

ZPLPrinter 对象

## 2.9. addReverse

区域反显

ZPLPrinter addReverse(int x, int y, int width, int height)

ZPLPrinter addReverse(int x, int y, int width, int height, int radius)

### 【参数】

➤ x

区域的起始 x 值

➤ y

区域的起始 y 值

➤ width

区域宽度

➤ height

区域高度。

➤ radius

圆角程度，范围:0~8，默认为 0。

**【返回值】**

ZPLPrinter 对象

## 2.10. addBox

绘制矩形

ZPLPrinter addBox(int x, int y, int width, int height, int thickness)

ZPLPrinter addBox(int x, int y, int width, int height, int thickness, int radius)

**【参数】**

➤ x

矩形的起始 x 值

➤ y

矩形的起始 y 值

➤ width

矩形宽度

➤ height

矩形高度。

➤ thickness

线条宽度。

➤ radius

圆角程度，范围:0~8，默认为 0。

**【返回值】**

ZPLPrinter 对象

## 2.11. addGraphicDiagonalLine

此函数功能为绘制对角线。

ZPLPrinter addGraphicDiagonalLine(int x, int y, char orientation, int width, int height, int thickness)

**【参数】**

➤ x

水平起始位置

➤ y

垂直起始位置

➤ orientation

对角线的方向。

变量	描述
R (或/)	右倾斜的对角线
L (或\)	左倾斜的对角线

➤ width

框的宽度 (范围: 1-32000 , 单位: dot)

➤ height

框的高度 (范围: 1-32000 , 单位: dot)

➤ thickness

边界厚度 (范围: 1-32000 , 单位: dot)

【返回值】

ZPLPrinter 对象

## 2.12. addGraphicEllipse

此函数功能为绘制图形椭圆。

ZPLPrinter addGraphicEllipse(int x, int y, int width, int height, int thickness)

【参数】

➤ x

水平起始位置

➤ y

垂直起始位置

➤ width

椭圆宽度 (范围: 3-4095 , 单位: dot)

➤ height

椭圆高度 (范围: 3-4095 , 单位: dot) 。

➤ thickness

边界厚度 (范围: 2-4095 , 单位: dot) 。

【返回值】

ZPLPrinter 对象

## 2.13. addGraphicCircle

此函数用于打印圆形

ZPLPrinter addGraphicCircle(int x, int y, int diameter, int thickness)

【参数】

➤ x

水平起始位置



- y  
垂直起始位置
- diameter  
圆的直径 (范围: 3-4095 , 单位: dot) 。
- thickness  
边界厚度 (范围: 1-4095 , 单位: dot) 。

【返回值】  
ZPLPrinter 对象

2.14. addBarcode

添加一维条码  
ZPLPrinter addBarcode(int x, int y, String codeType, String ratio, byte textPosition, String data, int width, int height)  
ZPLPrinter addBarcode(int x, int y, String codeType, String data, int height)  
ZPLPrinter addBarcode(int x, int y, String codeType, String data)

- 【参数】
- x  
条码起始 x 值
  - y  
条码起始 y 值
  - codeType  
条码类型

变量	描述
BCS_CODE11	Code 11 条码
BCS_INTERLEAVED2OF5	Interleaved 2 of 5 条码 （交叉二五码）
BCS_CODE39	Code 39 条码
BCS_EAN8	EAN-8 条码
BCS_UPCE	UPC-E 条码
BCS_CODE93	Code 93 条码
BCS_CODE128	Code 128 条码 （子集 A、 B 和 C）
BCS_EAN13	EAN-13 条码
BCS_CODABAR	ANSI Codabar 条码
BCS_MSI	MSI 条码
BCS_PLESSEY	Plessey 条码
BCS_UPC_EAN	UPC/EAN 扩展
BCS_UPCA	UPC-A 条码

- ratio  
条码方向, 默认 ROTATION\_0。

➤ **textPosition**

文本位置，默认 HRI\_TEXT\_BELOW。

变量	描述
HRI_TEXT_NONE	不显示文本注释
HRI_TEXT_ABOVE	文本注释显示在上面
HRI_TEXT_BELOW	文本注释显示在下面

➤ **data**

条码文本内容

➤ **width**

条码模块宽度,默认为 2 点。

➤ **height**

条码高度。默认 50 点。

**【返回值】**

ZPLPrinter 对象

## 2.15. addQRCode

添加二维条码

ZPLPrinter addQRCode(int x, int y, String data)

ZPLPrinter addQRCode(int x, int y, int size, String data)

**【参数】**

➤ **x**

二维码起始 x 值

➤ **y**

二维码起始 y 值

➤ **data**

二维码内容

➤ **size**

放大系数，范围 1~10，默认为 3。

**【返回值】**

ZPLPrinter 对象

## 2.16. printBitmap

打印图片

ZPLPrinter printBitmap(int x, int y, Bitmap bmp, int width, AlgorithmType algorithmType)

ZPLPrinter printBitmap(int x, int y, Bitmap bmp, int width)

通过压缩的方式传输图片，可以节省传输时间。

`ZPLPrinter printBmpCompress(int x, int y, Bitmap bmp, int width, AlgorithmType algorithmType)`

`ZPLPrinter printBmpCompress(int x, int y, Bitmap bmp, int width)`

打印 base64 字符串图片文件,仅支持能打印压缩图片的打印机。

`ZPLPrinter printBase64Img(int x, int y, String base64String, int pageWidth)`

`ZPLPrinter printBase64Img(int x, int y, String base64String, int pageWidth, AlgorithmType algorithmType)`

#### 【参数】

➤ x

图片起始 x 值

➤ y

图片起始 y 值

➤ bmp

图片 Bitmap 对象

➤ base64String

图片文件的 base64 字符串

➤ width

图片的打印宽度

➤ algorithmType

算法类型。默认使用 `AlgorithmType.Threshold`。

Dithering 抖动算法

Threshold 二值法

#### 【返回值】

`ZPLPrinter` 对象

## 2.17. printPdf

打印 PDF 文件,仅支持能打印压缩图片的打印机

`ZPLPrinter printPdf(int x, int y, String path, int pageWidth, AlgorithmType algorithmType) throws IOException`

`ZPLPrinter printPdf(int x, int y, String path, int pageWidth) throws IOException`

打印 base64 格式的 PDF 文件

`ZPLPrinter printBase64Pdf(int x, int y, String base64String, int pageWidth, AlgorithmType algorithmType) throws IOException`

`ZPLPrinter printBase64Pdf(int x, int y, String base64String, int pageWidth) throws IOException`

### 【参数】

- x  
起始 x 坐标
- y  
起始 y 坐标
- path  
pdf 文档路径
- base64String  
pdf 文件的 base64 字符串
- pageWidth  
打印宽度
- algorithmType  
算法类型。默认使用 AlgorithmType.Threshold。  
Dithering 抖动算法  
Threshold 二值法

### 【返回值】

ZPLPrinter 对象

## 2.18. downloadBitmap

下载图片到打印机。打印机断电存储会消失。

ZPLPrinter downloadBitmap(int width, String bmpName, Bitmap bmp)

ZPLPrinter downloadBitmap(int width, String bmpName, Bitmap bmp, AlgorithmType algorithmType)

### 【参数】

- width  
图片的打印宽度
- bmpName  
图片名称及拓展名， 名称为 1 至 8 个字母数字字符。
- bmp  
图片 Bitmap 对象
- algorithmType  
算法类型。默认使用 AlgorithmType.Threshold。  
Dithering 抖动算法  
Threshold 二值法

### 【返回值】

ZPLPrinter 对象

## 2.19. addBitmap

打印图片

ZPLPrinter addBitmap(int x, int y, String bmpName, int mx, int my)

ZPLPrinter addBitmap(int x, int y, String bmpName)

### 【参数】

- x  
图片起始 x 值
- y  
图片起始 y 值
- bmpName  
下载图片时取得名称和拓展名
- mx  
x 轴方向的放大系数，默认值为 1，范围为 1~10。
- my  
y 轴方向的放大系数，默认值为 1，范围为 1~10。

### 【返回值】

ZPLPrinter 对象

## 2.20. addPrintCount

设置打印数量

ZPLPrinter addPrintCount(int count)

### 【参数】

- count  
标签数量

### 【返回值】

ZPLPrinter 对象

## 2.21. setPrintSpeed

设置打印速度

ZPLPrinter setPrintSpeed(int speed)

### 【参数】

- speed  
打印速度。单位为 inches/sec

【返回值】

ZPLPrinter 对象

## 2.22. setPrintOrientation

该方法用于将标签格式反转 180 度

ZPLPrinter setPrintOrientation(String orientation)

【参数】

➤ orientation

打印方向

变量	描述
ROTATION_0	正常
ROTATION_180	反转

【返回值】

ZPLPrinter 对象

## 2.23. setPrintDensity

设置打印浓度

ZPLPrinter setPrintDensity(int density)

【参数】

➤ density

打印浓度(范围: 0-30)

【返回值】

ZPLPrinter 对象

## 2.24. setCloseHeadModel

设置打印头关闭模式

ZPLPrinter setCloseHeadModel(char model)

【参数】

➤ model

打印头关闭模式

变量	描述
CLOSE_HEAD_FEED	进纸
CLOSE_HEAD_CALIBRATE	传感器校正
CLOSE_HEAD_DETECTING	侦测标签长度
CLOSE_HEAD_NONE	无
CLOSE_HEAD_C_FORM	校正并回纸
CLOSE_HEAD_BACK	回到纸始
CLOSE_HEAD_SMART	智能进纸

【返回值】

ZPLPrinter 对象

## 2.25. rfidCalibration

此函数功能为校正 RFID 应答器位置。

ZPLPrinter rfidCalibration()

【返回值】

ZPLPrinter 对象

## 2.26. rfidWrite

此函数功能为写入（编码）RFID 标签。

ZPLPrinter rfidWrite(char format, int begin, int size, char memoryBlock, String text)

【参数】

➤ format

格式：A = ASCII、H = 十六进制、E = EPC（可先使用 rfidDefineEPC 定义 EPC 数据结构）

➤ begin

起始块编号

➤ size

写入的字节数

➤ memoryBlock

内存分段：E = EPC 96 位、0 = 保留、1 = EPC、2 = TID（标签 ID）、3 = 用户

➤ text

写入的文本数据

【返回值】

ZPLPrinter 对象

## 2.27. rfidDefineFont

此函数功能为定义打印 rfid 读取到的文本格式，结合 ZPL\_RfidRead 一起使用。  
ZPLPrinter rfidDefineFont(int x, int y, char fontName, String rotation, int sizeW, int sizeH)

【参数】

- x  
文本的起始 x 值
- y  
文本的起始 y 值
- fontName  
文本的字体类型。

变量	基础字体尺寸，高 x 宽
FNT_A	9 x 5
FNT_B	11x7
FNT_C、FNT_D	18x10
FNT_E	28x15
FNT_F	26x13
FNT_G	60x40
FNT_O	15*12

其他字库字体，请使用自定义名。

- sizeW  
文字有效宽度，默认为基础尺寸。请使用基础尺寸的整数倍。
- sizeH  
文字有效高度。默认为基础尺寸。请使用基础尺寸的整数倍。
- rotation  
顺时针旋转角度，默认 ROTATION\_0

变量	描述
ROTATION_0	不旋转
ROTATION_90	顺时针旋转 90 度
ROTATION_180	顺时针旋转 180 度
ROTATION_270	顺时针旋转 270 度

【返回值】

ZPLPrinter 对象

## 2.28. rfidRead

此函数功能为读取 （编码）RFID 标签。请结合 readData 一起使用。  
ZPLPrinter rfidRead(char format, int begin, int size, char memoryBlock, boolean isPrint)



#### 【参数】

➤ **format**

格式：A = ASCII、H = 十六进制、E = EPC（可先使用 `rfidDefineEPC` 定义 EPC 数据结构）

➤ **begin**

起始块编号

➤ **size**

写入的字节数

➤ **memoryBlock**

内存分段：E = EPC 96 位、0=保留、1=EPC、2=TID（标签 ID）、3 =用户

➤ **isPrint**

读到的内容是否打印出来。`true` 为打印结合 `rfidDefineFont` 一起使用，`false` 为不打印。

#### 【返回值】

ZPLPrinter 对象

## 2.29. rfidSetPower

此函数功能为设置 RFID 读取和写入功率级别

ZPLPrinter rfidSetPower(String read, String write)

#### 【参数】

➤ **read**

读取功率，值：0 至 30

➤ **write**

写入功率，值：0 至 30

#### 【返回值】

ZPLPrinter 对象

## 2.30. rfidDefineEPC

此函数功能为定义 EPC 数据结构

ZPLPrinter rfidDefineEPC(byte... bit)

#### 【参数】

➤ **bit**

分区大小集合。最大单个分区为 64 位。

#### 【返回值】

ZPLPrinter 对象

## 2.31. rfidSetParam

此函数功能为设置 RFID 参数

ZPLPrinter rfidSetParam(char labelType, int pos, int len, int number, char err)

**【参数】**

➤ labelType

标签类型。接受值:8 = EPC Class 1, 第 2 代(Gen 2)

➤ pos

应答器的读取 / 写入位置 (编程位置)

➤ len

无效打印输出的长度

➤ number

标签数量,读取 / 编码失败时尝试的标签数量。接受的值: 1 至 10

➤ err

错误处理,接受的值如下:

N = 不执行任何操作 (打印机放弃导致错误的标签格式, 并移至下一个队列标签)

P = 将打印机置于暂停模式 (标签格式将一直保留在队列中, 直到用户取消)

E = 将打印机置于出错模式 (标签格式将一直保留在队列中, 直到用户取消)

**【返回值】**

ZPLPrinter 对象

## 2.32. printerStatus

查询打印机状态

void printerStatus(IStatusCallback callback)

void printerStatus(int timeout, IStatusCallback callback)

**【参数】**

➤ callback

状态回调

```
public interface IStatusCallback {  
    void receive(int status);  
}
```

status(HEX)	描述
00	正常

01	前盖开
02	卡纸
03	卡纸且前盖开
04	缺纸
05	缺纸且前盖开
08	无色带
09	无色带且前盖开
0A	无色带且卡纸
0B	无色带、卡纸且前盖开
0C	无色带、缺纸
0D	无色带、缺纸且前盖开
10	暂停
20	打印中
80	其他错误
-1	读取超时或数据异常

➤ **timeout**

超时时间，单位为毫秒。默认 5000ms。

**【返回值】**

ZPLPrinter 对象

## 2.33. setCharSet

设置将打印内容传输给打印机所采用的字符编码，默认编码为“UTF-8”

`void setCharSet(String charSet)`

**【参数】**

➤ **charSet**

打印机所能识别的字符编码类型

## 2.34. sendData

该方法用于发送数据到打印机。

`ZPLPrinter sendData(byte[] data);`

`ZPLPrinter sendData(List<byte[]> datas);`

**【参数】**

➤ **data**

需发送的字节数组

➤ `datas`

需发送的字节数组集合

【返回值】

ZPLPrinter 对象

## 2.35. waitSendResultSync

等待数据发送完成.

`void waitSendResultSync(int timeout)`

【参数】

➤ `timeout`

最多等待时间, 单位为毫秒

【返回值】

`void`

## 3. ImageUtils

### 3.1. handleImageEffect

这个方法用于调整图片的对比度和亮度。

`static Bitmap handleImageEffect(Bitmap bmp, float contrast, float brightness)`

【参数】

➤ `bmp`

原图片

➤ `contrast`

对比度, 范围 0~2

➤ `brightness`

亮度, 范围-255~255

【返回值】

调整后的图片对象