|  |  |  |
| --- | --- | --- |
| 文档状态： | 文档编号： | 2019002 |
| [ ] Draft  [√] Released  [ ]Modifying | 编 撰： | 郭海岛 |
| 编撰日期： | 20190814 |
| 保密级别： | 公开 |
| 文档版本： | 1.0.10 |

# 开发手册

**修订表**

| 编号 | 生成版本 | 修订人 | 修订章节与内容 | 修订日期 |
| --- | --- | --- | --- | --- |
| 1 | V1.0.0 | 郭海岛 | 创建 | 2019-08-14 |
| 2 | V1.0.1 | 郭海岛 | 补充蓝牙描述 | 2019-08-15 |
| 3 | V1.0.2 | 郭海岛 | ESC二维码尺寸计算 | 2019-08-22 |
| 4 | V1.0.3 | 郭海岛 | ESC一维码，小程序补充 | 2019-09-02 |
| 5 | V1.0.4 | 郭海岛 | 图片打印补充 | 2019-09-26 |
| 6 | V1.0.5 | 郭海岛 | BLE的UUID补充 | 2019-10-16 |
| 7 | V1.0.6 | 郭海岛 | 补充老机型ESC二维码打印指令 | 2019-10-18 |
| 8 | V1.0.7 | 郭海岛 | 2.2章节补充iOS获取BLE蓝牙MAC地址方法 | 2020-01-17 |
| 9 | V1.0.8 | 郭海岛 | 增加第三章打印机状态和设置调整 | 2020-02-20 |
| 10 | V1.0.9 | 郭海岛 | 补充Andriod反色打印，水印打印接口 | 2020-05-29 |
| 11 | V1.0.10 | 郭海岛 | 补充电池电量查询指令描述 | 2020-06-19 |

**目 录**

[开发手册 1](#_Toc33112778)

[一、基础知识 4](#_Toc33112779)

[1.1打印机工作模式 4](#_Toc33112780)

[1.2打印机指令集 5](#_Toc33112781)

[1.3蓝牙 6](#_Toc33112782)

[二、开发系统 12](#_Toc33112783)

[2.1、Android 12](#_Toc33112784)

[2.2、iOS 17](#_Toc33112785)

[2.3、WinCE系统 23](#_Toc33112786)

[2.4、微信小程序 28](#_Toc33112787)

[2.5、Web应用 32](#_Toc33112788)

[2.6、云打印应用 33](#_Toc33112789)

[2.7、Windows系统 35](#_Toc33112790)

[三、打印机状态和设置调整 36](#_Toc33112791)

[3.1打印机状态显示 36](#_Toc33112792)

[3.2打印机状态查询 37](#_Toc33112793)

[3.3打印机固件和序列号查询 39](#_Toc33112794)

[3.4打印机OEM厂商查询 40](#_Toc33112795)

[3.5打印机设置调整 41](#_Toc33112796)

[四、字符编码 42](#_Toc33112797)

[4.0 ANSI编码 42](#_Toc33112798)

[4.1 Unicode编码 44](#_Toc33112799)

[4.2 UTF8编码 46](#_Toc33112800)

[4.3 组合字符 47](#_Toc33112801)

[五、常见开发问题 48](#_Toc33112802)

[5.1 二维码尺寸计算（ESC指令） 48](#_Toc33112803)

[5.2 二维码尺寸计算（CPCL指令） 55](#_Toc33112804)

[5.3一维码（ESC指令） 57](#_Toc33112805)

[5.4 定位走纸问题 59](#_Toc33112806)

[5.5 图片打印问题 60](#_Toc33112807)

## 一、基础知识

打印指令，又称打印控制命令。计算机通过打印控制语言，以软件命令的方法来控制打印机操作，解释执行打印数据，获得打印结果的。

### 1.1打印机工作模式

打印机主要工作模式有两种：驱动打印，指令打印。

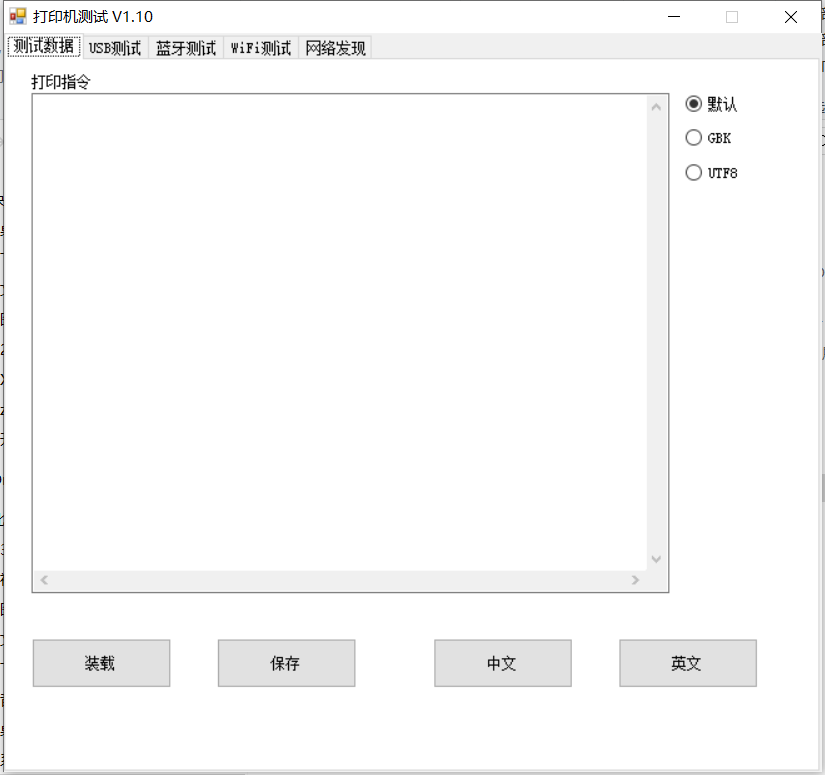
1、驱动打印需要在操作系统中安装打印机驱动，然后再根据蓝牙或是WiFi接口进行打印机的手动添加。然后有系统中的软件调用打印机驱动实现打印。

打印机安装参考：“手动添加打印机说明手册”

2、指令打印则直接通过发送和组织打印指令，让打印机打印相应内容。

可以通过“Windows\_打印机测试程序.rar”来安装并测试打印机功能。选择装载测试指令，然后选择USB/蓝牙/WiFi的不同通信通道进行打印指令测试。

注：网络发现只有CS系列打印机支持。



Windows打印机测试程序

### 1.2打印机指令集

#### 1.2.1 ESC指令

由Epson公司最早提出并实现，在针式打印机和票据打印领域的事实上的行业标准，主要用于单据打印，其特点在于是一行行输出打印内容，对每一条命令打印机立即给予响应执行。在打印机收到回车符号（“0x0A”）就会打印输出内容。

其指令以不可见字符为主，因此需要熟悉16进制的数据编码。

具体指令集参考：“ZICOX ESC 中文指令手册\_V2.2.pdf”

#### 1.2.2 CPCL指令

CPCL指令则常见于移动打印机上，主要以一个面单为基本单位，对数条命令组成的完整程序进行解释和执行、驱动打印通过指令绘制不同图案和文字。最后通过“PRINT”关键词一次性打印输出一个面单。

其指令是以ASCII可见字符为主，可读性强，易于上手。

具体指令集参考：“ZICOX\_CPCL打印指令集1.5.pdf”

#### 1.2.3 ZPL指令

ZPL命令来斑马打印机指令集，只有CS系列做了兼容性设计。

ZPL指令在线仿真：

<https://zplprinter.azurewebsites.net/>

### 1.3蓝牙

蓝牙分为传统蓝牙和BLE蓝牙两种连接方式。

协议规范：<https://www.bluetooth.com/specifications/archived-specifications/>

对于苹果的iOS系统来讲传统蓝牙需要经过苹果的安全认证，因此都是走BLE蓝牙通道通道进行通信。

BLE是Bluetooth Low Energy的缩写，又叫蓝牙4.0，区别于蓝牙3.0和之前的技术。

* 蓝牙双模简称BR（统蓝牙和BLE蓝牙两种都有），商标为Bluetooth Smart Ready；
* 单模简称BLE或者LE,商标为Bluetooth Smart

Android是在4.3后才支持BLE，这说明不是所有蓝牙手机都支持BLE，而且支持BLE的蓝牙手机一般是双模的。双模兼容传统蓝牙，可以和传统蓝牙通信，也可以和BLE通信，常用在手机上，Android4.3和IOS4.0之后版本都支持BLE，也就是双模设备。

蓝牙传输的最大数据包MTU在不同手机和设备上也是不同，一般是通过设备和手机之间进行通信约定的。

iOS可以通过maximumWriteValueLengthForType获得设备的最大数据包（iOS9以后）；

官方参考：

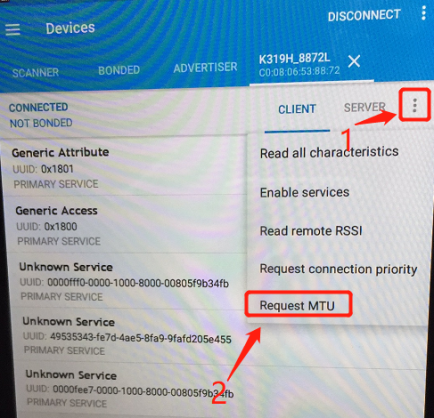
<https://developer.apple.com/documentation/corebluetooth/cbperipheral/1620312-maximumwritevaluelengthfortype?language=objc>

Android则是在API 21(Android 5.1)有一个 BluetoothGatt 新增一个requestMtu()的方法来调整MTU。

官方参考：

<https://developer.android.com/reference/android/bluetooth/BluetoothGatt>

可以通过“nRFConnect-4.6.1.apk”的第三方应用，可以查看支持的MTU大小，如下图：



MTU大小调整

#### 1.3.1 传统蓝牙

传统蓝牙3.0根据802.11适配层协议应用了Wi-Fi技术，极大提高了传输速度。蓝牙3.0的数据传输率提高到了大约24Mbps

1、传统蓝牙采用的是SPP（Serial Port Profile）协议进行数据传输。

2、SPP的UUID：00001101-0000-1000-8000-00805F9B34FB

3、手机一般以客户端的角色主动连接SPP协议设备

#### 1.3.2 BLE蓝牙

BLE蓝牙的最大特点就是低功耗，而低速率和简单的交互协议是降低功耗的重要组成部分，因此它的的传输速率只有2Mbps。

BLE的UUID：

serv\_id = '0000FFF0-0000-1000-8000-00805F9B34FB'

write\_id='0000FFF2-0000-1000-8000-00805F9B34FB'

BLE关键术语和概念的摘要：

一个ble蓝牙设备有多个或一个Profile。

一个Profile中有多个服务Service（通过服务的uuid找到对应的Service）。

一个Service中有多个特征Characteristic（通过特征的uuid找到对应的Characteristic）

一个Characteristic中包括一个value和多个Descriptor（通过Descriptor的uuid找到对应的Descriptor）

其次,要知道一些名词：

1、profile

可以理解为一种规范，一个标准的通信协议，它存在于从机中。

蓝牙组织规定了一些标准的profile，例如 HID OVER GATT，防丢器，心率计等。

每个profile中会包含多个service，每个service代表从机的一种能力。

2、service

可以理解为一个服务，在ble从机中，通过有多个服务，例如电量信息服务、系统信息服务等，每个service中又包含多个characteristic特征值。

每个具体的characteristic特征值才是ble通信的主题。比如当前的电量是80%，所以会通过电量的characteristic特征值存在从机的profile里，这样主机就可以通过这个characteristic来读取80%这个数据

官方Service定义：

<https://www.bluetooth.com/specifications/gatt/services/>

3、characteristic

特征值，ble主从机的通信均是通过characteristic来实现，可以理解为一个标签，通过这个标签可以获取或者写入想要的内容。

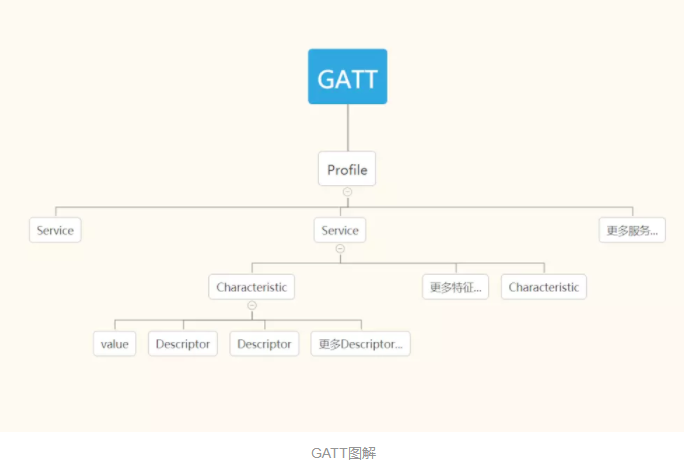
官方特征值定义：

<https://www.bluetooth.com/specifications/gatt/characteristics/>

4、UUID

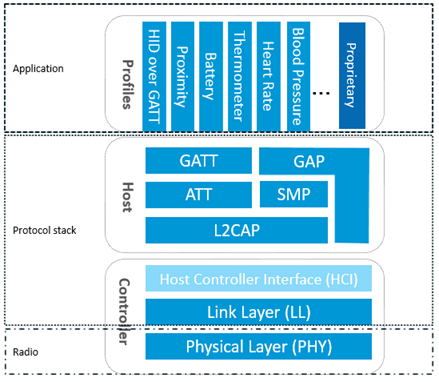
统一识别码，我们刚才提到的service和characteristic，都需要一个唯一的uuid来标识UUID的格式：00001101-0000-1000-8000-00805F9B34FB

综上，每个从机都会有一个叫做profile的东西存在，不管是上面的自定义的simpleprofile，还是标准的防丢器profile，他们都是由一系列service组成，然后每个service又包含了多个characteristic，主机和从机之间的通信，均是通过characteristic来实现。





Profile构成图



BLE协议栈架构图

UUID有16bit的，或者128bit的。16bit的UUID是官方通过认证的，需要花钱购买，128bit是自定义的，这个就可以自己随便设置。

其中有一个UUID非常特殊，它被蓝牙联盟采用为官方UUID，这个UUID如下所示：

https://images2018.cnblogs.com/blog/1366713/201807/1366713-20180707202931611-302677347.png

蓝牙联盟将自己定义的attribute或者数据只用16bit UUID来表示上图xxxx位置，其实它也是128bit。

例如设备的UUID扫描结果如下图所示：



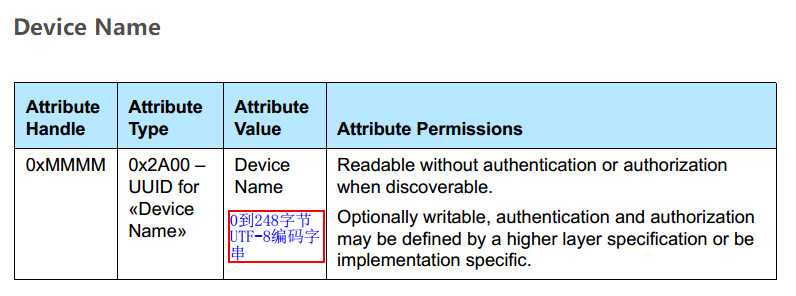
UUID实例

* 第1个UUID 0x2A05标识：

Service Changed org.bluetooth.characteristic.gatt.service\_changed 0x2A05

* 第2个UUID 0x2A00标识：

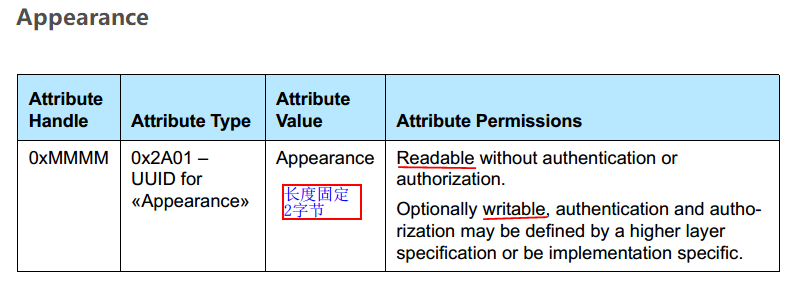
Device Name org.bluetooth.characteristic.gap.device\_name 0x2A00



Device Name长度可以到248字节，但一般不会需要那么长的名字，一般会建议最长到达40字节就好了。

* 第3个UUID 0x2A01标识：

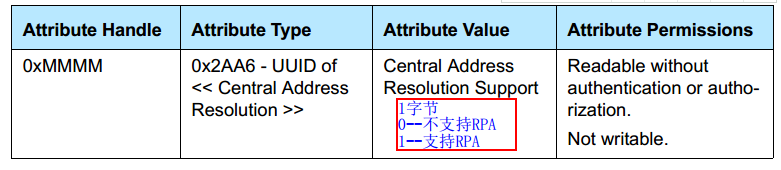
Appearance org.bluetooth.characteristic.gap.appearance 0x2A01



Appearance即表明这是个什么设备，如鼠标？键盘？等等

* 第4个UUID 0x2AA6标识：

Central Address Resolution org.bluetooth.characteristic.gap.central\_address\_resolution 0x2AA6



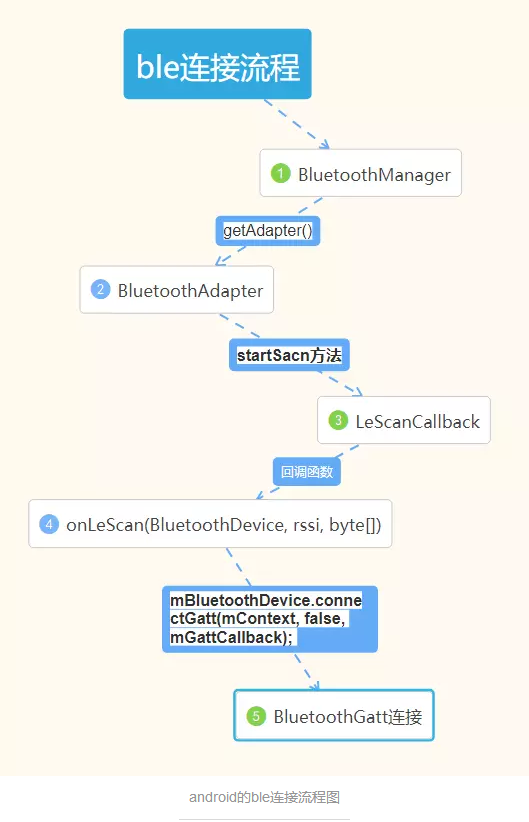
BT 4.2添加的新特性。因为在BT 4.2中，为了提高BT的安全性（原来SMP的方式是可以破解的），加入了Random Address的内容。这里的Central Address Resolution就是用来表明暴露GAP Service的这个Central设备，是否支持Resolvable Private Address(RPA)的。

Peripheral设备通过读对方Central设备的GAP Service的Central Address Resolution，就能知道其是否支持RPA了，如果其支持，那Peripheral就可以在自己的Directed Advertisement中使用RPA了。如果不支持，那还是只能使用public地址了。

关于蓝牙的public地址，private地址，请参考：

https://blog.csdn.net/suxiang198/article/details/47730649

* 第5-6个UUID 0xFFF1和0xFFF2标识分别是读取和写入；
* 第7-8个UUID标识也是读取和写入；
* 第9-11个UUID标识是支持微信的接口；



一般读，写和通知的UUID 就是 characteristic UUID

另外注意，连接设备前，请先关闭扫描蓝牙，否则连接成功后，再次扫描会发生阻塞，扫描不到设备。

官方Declarations 及Service、Characteristic、Descriptor的links：

<https://www.bluetooth.com/specifications/gatt/declarations/>

<https://www.bluetooth.com/specifications/gatt/services/>

<https://www.bluetooth.com/specifications/gatt/characteristics/>

<https://www.bluetooth.com/specifications/gatt/descriptors/>

官方协议规范links：

<https://www.bluetooth.com/zh-cn/specifications/protocol-specifications/>

## 二、开发系统

对于Android、iOS、WinCE系统都有配套的SDK提供，微信小程序只是提供一个参考例子。

### 2.1、Android

参考代码：demo\_ad\_sdk(cpcl通用版).zip

接口说明：

默认的坐标位置，线宽等都是以像素点为单位，像素点到毫米换算请除以8即可。

#### 2.1.1页面打印（print）

/\*\*

\* 页模式下打印页面

\*

\* @param horizontal

\* 0:正常打印，不旋转；1：整个页面顺时针旋转180°后，打印

\* @param skip

\* 具体数值如下：

\* 0：打印技术后不定位，直接停止；

\* 1：打印结束后定位到标签分割线，如果无缝隙，最大进纸30mm后停止；

\* 2：打印完走纸到左黑标；

\* 3：打印完走纸到右黑标。

\* @return 操作结果 0：发送成功；非0：请查询错误列表

\*/

public abstract void print(int horizontal,int skip);

#### 2.1.2页面创建（pageSetup）

/\*\*

\* 设置打印纸张大小(打印区域)的大小

\* @param pageWidth 打印区域宽度

\* @param pageHeight 打印区域高度

\*/

public abstract void pageSetup(int pageWidth, int pageHeight);

#### 2.1.3画框（drawBox）

/\*\*

\* 边框

\* @param lineWidth 边框线条宽度

\* @param top\_left\_x 矩形框左上角x坐标

\* @param top\_left\_y 矩形框左上角y坐标

\* @param bottom\_right\_x 矩形框右下角x坐标

\* @param bottom\_right\_y 矩形框右下角y坐标

\*/

public abstract void drawBox(int lineWidth,

int top\_left\_x, int top\_left\_y,

int bottom\_right\_x, int bottom\_right\_y);

#### 2.1.4画线（drawLine）

/\*\*

\* 线条

\* @param lineWidth 线条宽度

\* @param start\_x 线条起始点x坐标

\* @param start\_y 线条起始点y坐标

\* @param end\_x 线条结束点x坐标

\* @param end\_y 线条结束点y坐标

\* @param fullline true:实线 false:虚线

\*/

public abstract void drawLine(int lineWidth,

int start\_x, int start\_y,

int end\_x, int end\_y,boolean fullline);

#### 2.1.5文本（drawText）

/\*\* 打印文字

\* @param x 起始横坐标

\* @param y 起始纵坐标

\* @param text 字符串

\* @param fontsize

\* 字体大小 1：16点阵；2：24点阵；3：32点阵；

\* 4：24点阵放大一倍；5：32点阵放大一倍

\* 6：24点阵放大两倍；7：32点阵放大两倍；其他：24点阵

\* @param rotate

\* 旋转角度 0：不旋转；1：90度；2：180°；3:270°

\* @param bold

\* 是否粗体 0：取消；1：设置

\* @param underline

\* 是有有下划线 false:没有；true：有

\* @param reverse

\* 是否反白 false：不反白；true：反白

\*/

public abstract void drawText(int text\_x, int text\_y, String text,

int fontSize, int rotate,int bold, boolean reverse, boolean underline);

#### 2.1.6文本框（drawText）

/\*\*

\* \* 页模式下添加文本框（支持自动换行）

\* 如果需要设置对齐方式，计算坐标时就需要知道文字区域的大小

\* 以下四个参数就是文字区域的坐标

\* @param text\_x 文字起始x坐标

\* @param text\_y 文字起始y坐标

\* @param width 文本框宽度

\* @param height 文本框高度

\* @param text 文本内容

\* @param fontSize字体大小

\* @param rotate 旋转度数

\* @param bold 加粗

\* @param reverse 反显

\* @param underline 下划线

\*/

public abstract void drawText(int text\_x, int text\_y, int width, int height,

String str, int fontsize,int rotate ,int bold, boolean underline,

boolean reverse) ;

#### 2.1.7条码（drawBarCode）

/\*\*

\* 页模式下绘制一维条码

\* @param x 打印的起始横坐标

\* @param y 打印的起始纵坐标

\* @param str 字符串

\* @param barcodetype 条码类型

\* 0：CODE39；1：CODE128；2：CODE93；3：CODEBAR；

\* 4：EAN8；5：EAN13；6：UPCA ;7:UPC-E;8:ITF

\* @param rotate 旋转角度 false：不旋转

\* true：90度；

\* @param barWidth 条码宽度设置参数（非条码像素宽度，而是窄条码的宽度点数）

\* @param barHeight 条码高度

\* @throws UnsupportedEncodingException

\*/

public abstract void drawBarCode(int start\_x, int start\_y, String text,

int type, int rotate,int linewidth, int height);

#### 2.1.8条码（drawQrCode）

/\*\*

\* 二维码

\* @param start\_x 二维码起始位置

\* @param start\_y 二维码结束位置

\* @param text 二维码内容

\* @param rotate 旋转角度

\* @param ver : QrCode宽度(2-6)

\* @param lel : QrCode纠错等级(0-20)

\*/

public abstract void drawQrCode(int start\_x, int start\_y, String text,

int rotate, int ver, int lel);

#### 2.1.9条码（drawGraphic）

/\*\*

\* 图片

\* @param start\_x 图片起始点x坐标

\* @param start\_y 图片起始点y坐标

\* @param bmp\_size\_x 图片的宽度 固定设为0

\* @param bmp\_size\_y 图片的高度 固定设为0

\* @param bmp 图片

\*/

public abstract void drawGraphic(int start\_x, int start\_y, int bmp\_size\_x,

int bmp\_size\_y, Bitmap bmp);

#### 2.1.10反色画线（drawINVERSE）

/\*\*

\* 图片

\* @param start\_x 线条起始点x坐标

\* @param start\_y 线条起始点y坐标

\* @param end\_x 线条结束点x坐标

\* @param end\_y 线条结束点y坐标

\* @param width 线条宽度

\*/

public void drawINVERSE(int x0, int y0, int x1, int y1, int width)

#### 2.1.11水印打印（drawBackText）

/\*\*

\* 图片

\* @param text\_x 水印文字起始点x坐标

\* @param text\_y水印文字起始点y坐标

\* @param text 水印文字字符内容

\* @param fontName水印文字字符名 -- 无效

\* @param fontName水印文字大小

有效值1-10

\* @param rotate 旋转度数

1-90度 2-180度 3-270度

\* @param bold 加粗 1-加粗

\* @param backgound 灰度等级 0-255

\*/

public void drawBackText(int text\_x, int text\_y,

String text, String fontName, int fontSize,

int rotate, boolean bold, int backgound)

#### 2.1.11状态检测（printerStatus）

/ \*\*\*\*\* 检测打印机状态，在print（，）方法之后 \*\*\*\*\*\*\*/

zpSDK.printerStatus();

int a=zpSDK.GetStatus();

if(a==-1)

{ //"获取状态异常------";

Toast.makeText(this,"获取状态异常-----", Toast.LENGTH\_LONG).show();

}

if(a==1)

{//"缺纸----------";

Toast.makeText(this,"缺纸----------", Toast.LENGTH\_LONG).show();

}

if(a==2)

{ //"开盖----------";

Toast.makeText(this,"开盖----------", Toast.LENGTH\_LONG).show();

}

if(a==0)

{ //"打印机正常-------";

Toast.makeText(this,"打印机正常-------", Toast.LENGTH\_LONG).show();

}

#### 2.1.12蓝牙写入数据（Write）

void Write(byte[] Data)

单独发送数据给打印机。

#### 2.1.13蓝牙读取数据（Read）

void Read(byte[] Data, int len, int timeout)

接收打印机数据

参数：

Data：要接收的数据

len：接收的长度

timeout：超时，单位毫秒

### 2.2、iOS

参考代码：BTdemo (通用，包括打图片，库).rar

#### 2.2.0蓝牙MAC地址获取

由于iOS系统只能通过BLE蓝牙进行通信，而BLE蓝牙的MAC地址是无法从iOS提供的接口中获得（每一个iOS设备扫描到的蓝牙设备编号是不一样的）。因此在广播中加入了蓝牙MAC地址。

苹果手机获得广播信息接口参考代码：

<https://www.jianshu.com/p/bf83b6757aed>

- (void)peripheral:(CBPeripheral \*)peripheral didDiscoverServices:(NSError \*)error {

for (CBService \*service in peripheral.services) {

[peripheral discoverCharacteristics:nil forService:service];

}

}

- (void)peripheral:(CBPeripheral \*)peripheral didDiscoverCharacteristicsForService:(CBService \*)service error:(NSError \*)error {

for (CBCharacteristic \*characteristic in service.characteristics) {

if ([characteristic.UUID.UUIDString isEqualToString:@"585A"]) {

[peripheral setNotifyValue:YES forCharacteristic:characteristic];

[peripheral readValueForCharacteristic:characteristic];

}

}

}

-(void)peripheral:(CBPeripheral\*)peripheral didUpdateValueForCharacteristic:(CBCharacteristic\*)characteristic error:(NSError\*)error

{

NSString \*value = [[NSString alloc] initWithData:characteristic.value encoding:NSUTF8StringEncoding];

NSLog(@"MAC地址是macString:%@",value);

}

将[characteristic.UUID.UUIDString isEqualToString:@"2A25"]替换为：“585A”来搜到相应的MAC地址即可。

设备广播的MAC地址在LightBlue和nRF Connect下的广播如下图所示：

#### 2.2.1 搜索（scanStart）

／\*

\* scanStart 搜索

\* callback 服务的回调，具体使用见demo

\*／

- (void)scanStart:(BLOCK\_CALLBACK\_SCAN\_FIND)callback;

#### 2.2.2 停止搜索（scanStop）

／\*

\* 停止搜索

\*／

- (void)scanStop;

#### 2.2.3 连接设备（open）

／\*连接设备\*／

- (bool)open:(CBPeripheral\*)peripheral;

#### 2.2.4 关闭设备（close）

／\*关闭设备\*/

- (void)close;

#### 2.2.5 发送字符串数据（write）

/\*发送数据 字符串类型\*/

- (bool)write:(NSString\*)strData;

#### 2.2.6 发送字节数据（writeData）

/\*发送数据 字节类型\*/

- (bool)writeData:(NSData\*)data;

#### 2.2.7 清空缓存（flushRead）

/\*清空缓存\*/

- (void)flushRead;

#### 2.2.8 读数据（readBytes）

/\*读数据

\*data：要接收的数据

\*len：接收的数据长度

\*timeout：超时

\*/

- (bool)readBytes:(BytePtr)data len:(int)len timeout:(int)timeout;

#### 2.2.9开始创建一个打印页面（StartPage）

／\* 开始创建一个打印页面

\*pageHeight：页面的高

\*pageWidth：页面的宽

\*skip：定位情况：0--打印后直接停止 1-打印完走到标签缝隙

\*rotate：页面旋转

\*0--不旋转、90--旋转90度、180--旋转180度、270---旋转270度

\*／

-(void)StartPage:(int) pageWidth pageHeight:(int)pageHeight skip:(bool)skip rotate:(int)rotate;

#### 2.2.10结束打印页面（end）

/\*结束打印页面\*/

-(void)end;

#### 2.2.11打印字符串（zp\_drawText）

／\*

\*打印字符串

\*x：x坐标

\*y：y坐标

\*text：内容

\*font；字体 55--16点阵 24--24点阵 56--32点阵

\*fontsize：放大倍数 如果写1或者0表明不放大，2是放大一倍（字体\*2）

\*bold： 1-加粗 0-不加粗

\*rotate：旋转 0-不旋转 90-90度 180-180度 270-270度 （可能某些版本无此参数）

\*／

-(void)zp\_drawText:(int)x y:(int)y text:(NSString\*)text

font:(int)font fontsize:(int)fontsize bold:(int)bold rotate:(int)rotate;

#### 2.2.12打印直线（zp\_drawLine）

／\*打印直线

\*startPointX：开始的x坐标

\*startPointY：开始的y坐标

\*endPointX：结束的x坐标

\*endPointY：结束的y坐标

\*width：线条宽度 0-1-2-3

\*／

-(void)zp\_drawLine:(int)startPointX startPiontY:(int)startPointY

endPointX:(int)endPointX endPointY:(int)endPointY width:(int)width;

#### 2.2.13打印条形码（zp\_darw1D\_barcode）

／\*

\*打印条形码

\*x：x坐标

\*y：y坐标

\*height：条码高度

\*text：内容

\*／

-(void)zp\_darw1D\_barcode:(int)x y:(int)y height:(int)height text:(NSString\*)text;

#### 2.2.14打印二维码（zp\_ darwQRCode）

／\*

\*打印二维码

\*x：x坐标

\*y：y坐标

\*unit\_width： 二维码宽点数（大小）

\*text：内容

\*／

-(void) zp\_darwQRCode:(int)x y:(int)y unit\_width:(int)unit\_width text:(NSString\*)text;

#### 2.2.14打印矩形（zp\_ darwRect）

／\*打印矩形\*／

-(void)zp\_darwRect:(int)left top:(int)top right:(int)right bottom:(int)bottom width:(int)width;

#### 2.2.15打印图片（drawBitmap）

／\*

\*打印图片

\*image：图片资源

\*x：x坐标

\*y：y坐标

\*／

-(void) drawBitmap:(UIImage\*) image x:(int) x y:(int) y ;

#### 2.2.16获取需要发送的指令数据（getData）

／\*获取需要发送的指令数据，用writeData或者write发送。\*／

-(NSData\*) getData:(int)sendLength;

#### 2.2.17清空数据（reset）

／\*清空数据buff\*／

-(void) reset;

#### 2.2.18获取打印机状态指令（print\_status\_detect）

／\*给打印机发送获取打印机状态指令\*/

-(void) print\_status\_detect;

/\*获取到状态

\*返回0正常

\*返回1--打印机缺纸，返回2，打印机开盖，返回-1，打印机异常。

\*/

-(int)print\_status\_get:(int)timeout;

#### 2.2.19蓝牙过滤参考

1、CBUUID的服务名过滤

@[[CBUUID UUIDWithString:@"FFF0"]] 是为了过滤掉其他设备，可以搜索特定标示的设备。

[\_manager scanForPeripheralsWithServices:@[[CBUUID UUIDWithString:@"FF15"]] options:@{CBCentralManagerScanOptionAllowDuplicatesKey : @YES }];

2、名字、广播内容、 信号强度过滤

//设置查找设备的过滤器

[weakSelf.baby setFilterOnDiscoverPeripherals:^BOOL(NSString \*peripheralName, NSDictionary \*advertisementData, NSNumber \*RSSI) {

if (peripheralName.length >1) {

return YES;

}

return NO;

}];

### 2.3、WinCE系统

参考代码：cpcl\_wince应用例子.zip

动态库：zp\_cpcl\_sdk.dll

接口参考：zp\_sdk.cs

条码类型：

public enum BARCODE\_TYPE

{

BARCODE\_CODE128,

BARCODE\_CODE39,

BARCODE\_CODE93,

BARCODE\_CODABAR,

BARCODE\_EAN8,

BARCODE\_EAN13,

BARCODE\_UPC,

};

#### 2.3.1打开蓝牙（bt\_open）

/\*

\*参数1：BTAddr

\* 蓝牙设备的MAC地址

\*参数2：password

\* 蓝牙设备PIN码，默认“0000”

\*/

public bool bt\_open(UInt64 BTAddr,String password)

#### 2.3.2关闭蓝牙（bt\_close）

public void bt\_close()

#### 2.3.3搜索蓝牙（bt\_search）

/\*

\*参数1：Name\_Array

\* 蓝牙设备的名字字符串

\*参数2：Addr\_Array

\* 蓝牙设备的MAC地址

\*参数3：DeviceNum

\* 总共扫描到的设备数量

\*/

public bool bt\_search(Char[,] Name\_Array, UInt64[] Addr\_Array, ref int DeviceNum)

#### 2.3.4打印状态检测（printer\_status\_get/printer\_status\_detect）

先执行printer\_status\_detect函数，再通过printer\_status\_get函数获得当前打印机状态

public void printer\_status\_detect()

/\*

\* 返回值 0-正常 1--打印机缺纸，返回2，打印机开盖，返回-1，打印机异常。

\*/

public int printer\_status\_get(int timeout)

#### 2.3.5创建打印页面（page\_create）

/\*\*

\* 设置打印纸张大小(打印区域)的大小

\* int PageWidthMM 打印区域宽度

\* int PageHeightMM 打印区域高度

\* bool Rotate 无效

\*/

public bool page\_create(int PageWidthMM, int PageHeightMM, bool Rotate)

#### 2.3.6字体选择（page\_set\_font）

/\*

\* FontName；字体 55--16点阵 24--24点阵 56--32点阵

\* FontHeight：大小

\* Angle：旋转 0-不旋转 90-90度 180-180度 270-270度 （可能某些版本无此参数）

\* bold： 1-加粗 0-不加粗  
\* Italic： 无效

\* Underline 1-下划线 0-无下划线

\*／

public void page\_set\_font(int FontName, int FontHeight, int Angle,

bool Bold, bool Italic, bool Underline)

#### 2.3.7打印字符串（draw\_text）

／\*

\*打印字符串

\*x：x坐标

\*y：y坐标

\*text：内容

\*／

public void draw\_text(int x, int y, String text)

#### 2.3.8反色区域（draw\_inverse\_line）

／\*

\*画出反色区域

\*x0：起始点x坐标

\*y0：起始点y坐标

\*x1：结束点x坐标

\*y1：结束点y坐标

\* PenWidth：画笔宽度

\*／

public void draw\_inverse\_line(int x0, int y0, int x1, int y1, int PenWidth)

#### 2.3.9打印直线（draw\_line）

／\*画线

\*x0：起始点x坐标

\*y0：起始点y坐标

\*x1：结束点x坐标

\*y1：结束点y坐标

\* PenWidth：画笔宽度

\*／

public void draw\_line(int x0, int y0, int x1, int y1, int PenWidth)

#### 2.3.10打印矩形（draw\_rect）

／\*画线

\* left：起始点x坐标

\*top：起始点y坐标

\*right：结束点x坐标

\*bottom：结束点y坐标

\* PenWidth：画笔宽度

\*／

public void draw\_rect(double left, double top, double right, double bottom, int PenWidth)

#### 2.3.11打印条形码（draw\_barcode）

／\*

\*打印条形码

\*x：x坐标

\*y：y坐标

\* pData：内容

\*TypeCode：条码类型，见BARCODE\_TYPE

\*Height：条码高度，像素点

\*LineWidth：窄条码的宽度点数（宽条码和窄条码的比率1 = 2.0 : 1）

\*Rotate：0-水平 1-垂直

\*／

public bool draw\_barcode(int x, int y, String pData, BARCODE\_TYPE TypeCode, int Height, int LineWidth, int Rotate)

#### 2.3.12打印QR二维码（draw\_barcode2d\_qrcode）

／\*

\*打印二维码

\*x：x坐标

\*y：y坐标

\*pData：内容

\*Ver：无效

\*Size：二维码大小

\*Rotate：无效

\*／

public bool draw\_barcode2d\_qrcode(int x, int y, String pData, int Ver, int Size, int Rotate)

#### 2.3.13打印（page\_print）

public bool page\_print()

#### 2.3.14走纸到标签（goto\_mark\_lable）

／\*

\*走纸到标签缝隙处

\*MaxFeedMM-无效

\*/

public bool goto\_mark\_lable(int MaxFeedMM)

#### 2.3.15走纸到黑标（goto\_mark\_right /goto\_mark\_left）

／\*

\*走纸到右黑标处

\*MaxFeedMM-无效

\*/

public bool goto\_mark\_right(int MaxFeedMM)

／\*

\*走纸到左黑标处

\*MaxFeedMM-无效

\*/

public bool goto\_mark\_left(int MaxFeedMM)

### 2.4、微信小程序

参考代码：小程序\_BLESerialPort.rar

微信小程序论坛：

官方接口：<https://developers.weixin.qq.com/miniprogram/dev/api/>

微信小程序的发送给蓝牙设备需要转换成byte数据，其字符串转换byte处理需要借助base64的编解码函数：

wx.base64ToArrayBuffer(string base64)

#### 2.4.1 Base64编码

Base64是网络上最常见的用于传输8Bit字节码的编码方式之一，Base64就是一种基于64个可打印字符来表示二进制数据的方法。

* Java版本：

import java.util.Base64;

* 对于标准的Base64：

加密为字符串使用Base64.getEncoder().encodeToString();

加密为字节数组使用Base64.getEncoder().encode();

解密使用Base64.getDecoder().decode();

* 对于URL安全或MIME的Base64：

只需将上述getEncoder()getDecoder()更换为getUrlEncoder()getUrlDecoder()

或getMimeEncoder()和getMimeDecoder()即可。

* 在线编解码

<https://the-x.cn/base64/>

注意：请采用中文编码格式GB2312

中文编码格式GB2312

中文编码格式GB2312



* 例子
  + 源码文本指令数据

! 0 200 200 350 1

CENTER

SETMAG 2 2

TEXT 24 0 0 10 物料领用单

B PDF-417 0 80 XD 3 YD 12 C 3 S 2

PDF Data

ABC123

ENDPDF

LEFT

SETMAG 0 0

TEXT 24 0 100 170 需求单号：MR190704000033

TEXT 24 0 100 200 SR单号：SR120180406172018294

TEXT 24 0 100 230 领用编码：02220175

TEXT 24 0 100 260 领用人：张三

BOX 0 0 556 300 3

PRINT

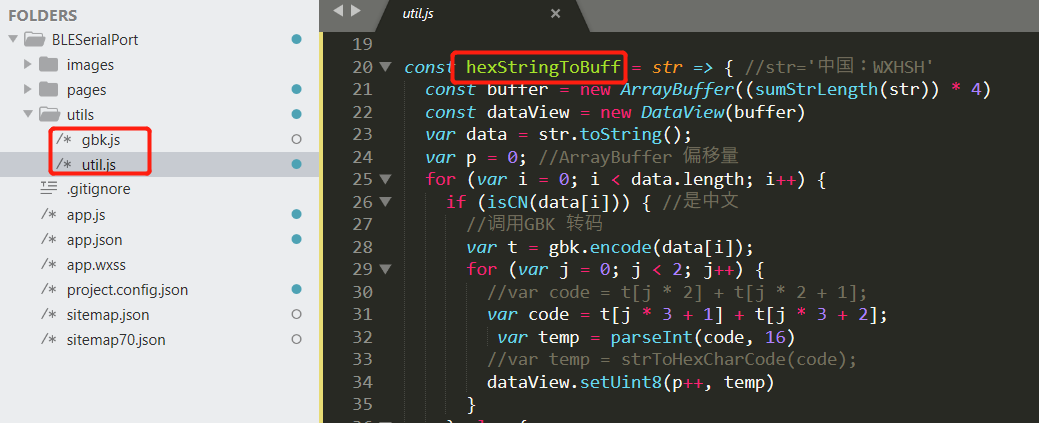
* + Base编码后数据

ISAwIDIwMCAyMDAgMzUwIDEKQ0VOVEVSClNFVE1BRyAyIDIKVEVYVCAyNCAwIDAgMTAgzu/Bz8Hs08O1pQpCIFBERi00MTcgMCA4MCBYRCAzIFlEIDEyIEMgMyBTIDIKUERGIERhdGEKQUJDMTIzCkVORFBERgpMRUZUClNFVE1BRyAwIDAKVEVYVCAyNCAwIDEwMCAxNzAg0OjH87WlusWjuk1SMTkwNzA0MDAwMDMzClRFWFQgMjQgMCAxMDAgMjAwIFNStaW6xaO6U1IxMjAxODA0MDYxNzIwMTgyOTQKVEVYVCAyNCAwIDEwMCAyMzAgwezTw7HgwuujujAyMjIwMTc1ClRFWFQgMjQgMCAxMDAgMjYwIMHs08PIy6O61cXI/QpCT1ggMCAwIDU1NiAzMDAgMwpQUklOVAo=

#### 2.4.2 字符直接转GBK编码

通过更新两个js文件（gbk.js和util.js），实现hexStringToBuff的函数，将字符串直接转换成中文编码数据（注意：GB2312编码收入的汉字不全容易出现乱码问题）。

具体参考共享路径“公开资料\开发手册和SDK”里面的“小程序\_BLESerialPort”。



实现方法



调用方法

#### 2.4.3 蓝牙通信

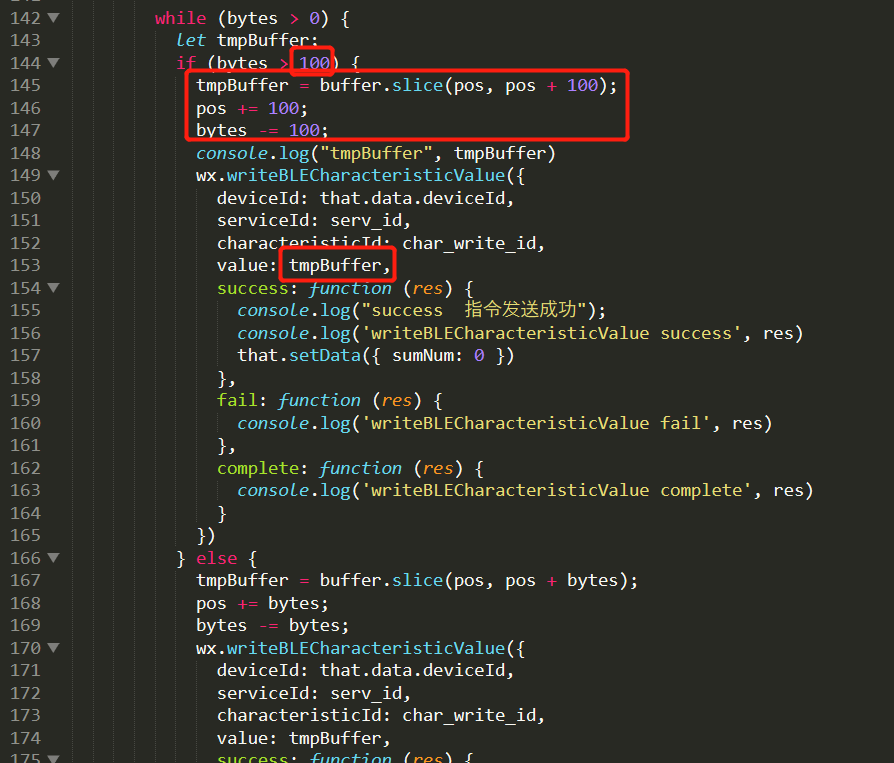
微信小程序采用BLE蓝牙通信模式（具体介绍见1.3章节介绍），由于不同手机对于BLE传输的支持不同因此需要进行拆包发送，每包的最大字节数是蓝牙设备与手机系统一起决定的。对于一些老的手机的话每包发送20字节比较稳妥（具体原因可以参考1.3章节介绍）。

请务必拆包发送数据，

请务必拆包发送数据，

请务必拆包发送数据。

代码如下：



拆包方法

另外微信小程序只是将蓝牙设备的接口进行回调，因此如果数据包超过最大字节后，如果设备特征值是write 属性的话，系统应该会吐回 10008 错误的；如果设备特征值是 writeWithoutResponse 属性的情况，系统是不会向小程序报错。这两个属性的差异只是在有无应答。

notification，indicate，read则是三种不同的从设备读取数据的方法。

1、notification和indication都是基于GATT的，取决于你在应用层如何对这些数据处理，如果你对notification的数据处理不及时，是会被后续数据冲击丢失的。indication要等回复才能发下一个包，传输更可靠。

2、Read的方式速度比Notification要低，用write和read来发送和获取特征值的方式方式只适用于手机，只手机主动发起的。而设备若想主动发送数据给手机那么只能用notification或indication。

#### 2.4.4关于getBLEDeviceServices失败

错误信息：

getBLEDeviceServices fail {errCode: 10004, errMsg: "getBLEDeviceServices:fail:no service"}

微信官方论坛：

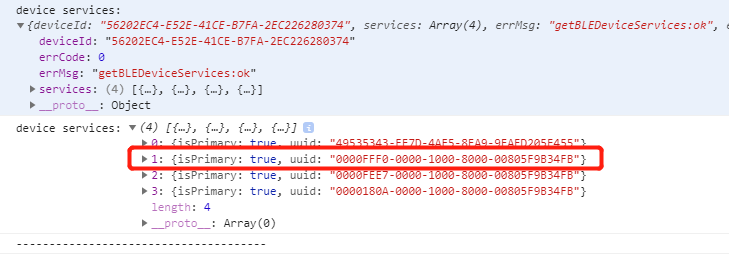
<https://developers.weixin.qq.com/community/develop/doc/000a40cbef4090979288052e356800?highLine=getBLEDeviceServices%2520>

不能在手机蓝牙里配对，配对过的，会出现10004。取消手机里面的配对。

<https://developers.weixin.qq.com/community/develop/doc/000a6a6c5d4c00a7c4d88e3e756c00>

解决方法：

针对一些手机可能就会有这样的问题，因此建议失败后采用默认的ServicesUUID。



### 2.5、Web应用

#### 2.5.1安卓Native.js蓝牙连接

实现参考：

https://blog.csdn.net/mss359681091/article/details/82111872

#### 2.5.2网络控件

AO打印是英文Active-Online Print的简称，也称主动在线打印。打印前支持AO通讯协议的AO打印机首先通过普通网络与C-Lodop服务保持在线链接，网页程序利用JavaScript语句向打印机输出复杂内容。

相比于传统网络打印，打印机由被动链接变为主动链接，这种部署的优势很明显，既可以外网异地打印，也可以内外网本地打印，对打印两端的网络位置无要求，从而轻松实现广域网远程打印，手机、平板打印问题也一举搞定。鉴于当前宽带普及，加之云服务器租金越来越便宜，这都为AO打印的流行创造了条件。

<http://blog.sina.com.cn/s/blog_721e77e50102xbci.html>

#### 2.5.3 标准Web访问蓝牙

官方介绍：

<https://developer.mozilla.org/en-US/docs/Web/API/Bluetooth>

现阶段支持的浏览器：



代码参考：

<https://github.com/WebBluetoothCG/demos/tree/gh-pages/bluetooth-printer>

### 2.6、云打印应用

#### 2.6.1 Http

#### 2.6.2 MQTT

#### 2.6.3 WebSocket

## 2.7、Windows系统

Windows系统上多数采用驱动方式，由本地服务器作为打印服务器，然后给远端的程序调用实现打印。

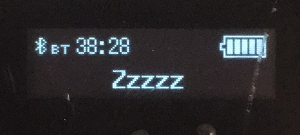
具体参考《手动添加打印机说明手册》。

## 三、打印机状态和设置调整

### 3.1打印机状态显示

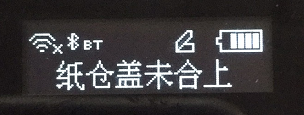
打印机主要状态：开盖、缺纸、定位失败、低电、打印中、打印头过热、休眠。

休眠，即打印机在没有操作情况下进入省电休眠模式，如需唤醒，则按电源按键即可。蓝牙连接，接通电源后会自动退出休眠。屏幕显示如下图所示：



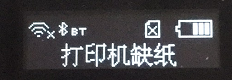
休眠状态

开盖，即提示纸仓盖没有关好。屏幕提示如下图所示：



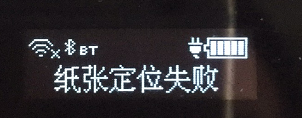
开盖报警

缺纸，即提示打印机内没有可打印纸张。屏幕提示如下图所示：



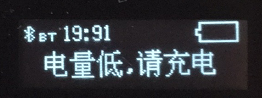
打印机缺纸

定位失败，即提示打印机收到定位指令但是无法定位到黑标或是标签缝隙处。屏幕提示如下图所示：



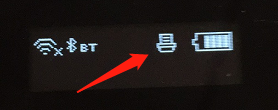
定位失败

低电，即提示打印机电池电量不足。屏幕提示如下图所示：



低电提示

打印中，即提示打印机现在正在打印机。屏幕提示如下图所示：



打印中

### 3.2打印机状态查询

通过私有指令：0x1D 0x99(BASE64编码：HZk=)实现状态查询。

例1：自己组织指令

void Write(byte[] Data)： 单独发送数据给打印机。

int Read(byte[] Data, int len, int timeout)：接收打印机数据

byte[] wtData=new byte[2];

byte[] rdData=new byte[4]

wtData[0]=0x1D;

wtData[1]=0x99;

Write(wtData);

memset(rdData,0,sizeof(rdData));

int rdLen=Read(rdData,4,1000);

If(rdLen == 4){

If(rdData[2]==0){

//正常

}

if(rdData[2]&0x01){

//缺纸

}

if(rdData[2]&0x02){

//开盖

}

if(rdData[2]&0x04){

//打印头过热

}

if(rdData[2]&0x08){

//定位失败

}

if(rdData[2]&0x10){

//低电

}

if(rdData[2]&0x20){

//正在打印

}

}

例2：采用SDK接口

/ \*\*\*\*\*

检测打印机状态，用在print（，）方法之后

\*\*\*\*\*\*\*/

zpSDK.printerStatus();

int a=zpSDK.GetStatus();

if(a==-1)

{ //"获取状态异常------";

Toast.makeText(this,"获取状态异常-----", Toast.LENGTH\_LONG).show();

}

if(a==1)

{//"缺纸----------";

Toast.makeText(this,"缺纸----------", Toast.LENGTH\_LONG).show();

}

if(a==2)

{ //"开盖----------";

Toast.makeText(this,"开盖----------", Toast.LENGTH\_LONG).show();

}

if(a==0)

{//"打印机正常-------";

Toast.makeText(this,"打印机正常-------", Toast.LENGTH\_LONG).show();

}

### 3.3打印机固件和序列号查询

不同打印机型号，其固件版本查询指令有所差异。

#### 3.3.1 XT423等

* 版本获取指令

在任意通信端口发送如下报文数据：

0x1D 0x49 0x41

Base64编码：HUlB

打印应答数据：

0x32 0x2e 0x34 0x38 0x00(2.48)

* 打印机序列号获取指令

在任意通信端口发送如下报文数据：

0x1D 0x49 0x44

Base64编码：HUlE

打印应答数据：

0x32 0x32 0x36 0x35 0x32 0x34 0x30 0x38 0x33 0x36 0x00 (2265240836)

#### 3.3.2 CS系列

CS系列由2个CPU构成，因此其版本分为2个。

* 第一个CPU版本获取指令

在任意通信端口发送如下报文数据：

0x1f 0x52 0x0a 0x00 0x20 0x01 0x56 0x45 0x52 0x53 0x49 0x4f 0x4e 0x00

Base64编码：H1IKACABVkVSU0lPTgA=

打印应答数据：

0x20 0x06 0x00 0x34 0x2e 0x31 0x30 0x36 0x00

长度0x06，有效数据字符串：0x34 0x2e 0x31 0x30 0x36 0x00

* 第二个CPU版本获取指令

在任意通信端口发送如下报文数据：

0x1f 0x52 0x0e 0x00 0x20 0x01 0x56 0x45 0x52 0x53 0x49 0x4f 0x4e 0x5f 0x43 0x4d 0x44 0x00

Base64编码：H1IOACABVkVSU0lPTl9DTUQA

打印应答数据：

0x 20 0x04 0x00 0x31 0x32 0x30 0x00

长度0x04，有效数据字符串：0x31 0x32 0x30 0x00

* 打印机序列号获取指令

在任意通信端口发送如下报文数据：

0x1f 0x52 0x0b 0x00 0x20 0x00 0x43 0x50 0x55 0x5f 0x49 0x44 0x33 0x32 0x00

Base64编码：H1ILACAAQ1BVX0lEMzIA

打印应答数据：

0x20 0x04 0x00 0x60 0xf9 0xc7 0xa9

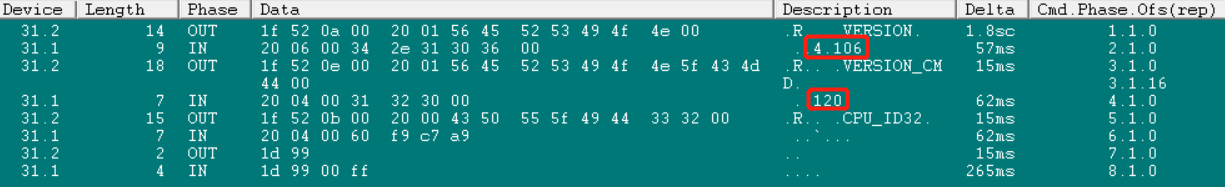
长度0x06，有效数据Int：0x60 0xf9 0xc7 0xa9，实际序列号：2848455008，一般会以最后4位数（5008）作为蓝牙广播的名字后缀。



序列号换算

版本信息USB数据抓包如下图所示：

此设备版本信息：4.106.120 设备序列号



抓包数据

### 3.4打印机OEM厂商和型号查询

OEM查询，在任意通信端口发送如下报文数据：

0x 1D 0x4A 0x01

Base64编码：HUoB

打印应答数据：

0x 7A 0x6B (zk)

型号查询，在任意通信端口发送如下报文数据：

1F 52 10 00 20 01 4F 45 4D 5F 4D 4F 44 45 4C 4E 41 4D 45 00

Base64编码：H1IQACABT0VNX01PREVMTkFNRQA=

打印应答数据：

20 04 00 43 53 33 00 (CS3)

### 3.5电量查询

电量查询，在任意通信端口发送如下报文数据：

1F 52 0E 00 20 00 42 41 54 54 45 52 59 5F 43 41 50 00

Base64编码：H1IOACAAQkFUVEVSWV9DQVAA

打印应答数据：

20 04 00 5C 00 00 00

int 值 5C=92 92%

### 3.6打印机设置调整

不同型号、不同程序版本的打印机其设置内容不同，以下列举几种常见设置。

长按设置键，打印机进入设置界面，短按设置键切换选项，显示是要设置的项时，短按走纸键，对项的参数值进行调整。

注意：如果打印机在休眠状态，请先按一下“电源按键”，唤醒打印机后再按设置按键才有效。

#### 3.6.1 打印模式

“打印模式”，分别有“快速”、“正常”、“高质量”，会直接影响打印效果，打印清晰度是“高质量”最好，然后依次是“正常”，“快速”，打印速度就正好相反，高质量打印速度最慢，快速则打印速度最快。

#### 3.6.2 打印黑度

“打印黑度”，打印黑度由低到高有“-1”“0”“1”三个可调整值。因为纸张型号或批次不同，有些情况会出现太黑或太淡，可以通过调节黑度来做打印效果的优化。

#### 3.6.3 合盖走纸

“盒盖走纸”可以设置成“是”或“否”。主要针对标签纸（黑标纸）的应用，每次更换纸卷后可以自动走出一张纸起到准确定位作用。

#### 3.6.4 自动补打

“自动补打”可以设置成“是”或“否”。主要提供在打印过程中发生缺纸或是定位失败的情况下，更换纸卷后自动将刚才补打的功能。

#### 3.6.5 语言

“LANG”语言设置，可以设置成“中文”或“English”。可以将设置菜单切换成英文，方便海外用户使用。

## 四、字符编码

参考：

<https://www.jianshu.com/p/64ec0f6b6245>

<https://www.unicode.org/reports/tr29/>

### 4.0 ANSI编码

在全世界所有国家和民族的文字符号统一编码的Unicode编码方案问世之前，各个国家、民族为了用计算机记录并显示自己的字符，都在ASCII编码方案的基础上，设计了各自的编码方案。

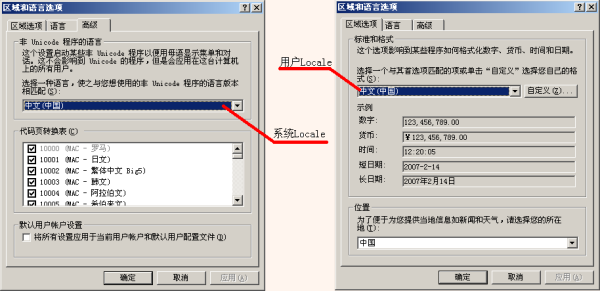
比如欧洲先后设计了EASCII和ISO/IEC 8859系列字符编码方案；为了显示中文及相关字符，中国设计了GB系列编码(“GB”为“国标”的汉语拼音首字母缩写，即“国家标准”之意)。同样，日文、韩文、世界各国文字都有它们各自的编码。所有这些各个国家和地区所独立制定的既兼容ASCII又互相不兼容的字符编码，微软统称为ANSI编码。

所以，即使知道是ANSI编码，还需要知道这是哪一个国家的才能解码；另外，也无法用同一种ANSI编码表示既有汉字、又有韩文的文本。

严格来说，ANSI的字面意思并非字符编码，而是美国的一个非营利组织——美国国家标准学会(American National Standards Institute)的缩写。ANSI这个组织做了很多标准制定工作，包括C语言规范ANSI C，还有各国字符编码对应的“代码页(code page)”标准。

微软为了适应世界上不同地区用户的文化背景和生活习惯，在Windows中设计了区域（Locale）设置的功能。Local是指特定于某个国家或地区的一组设定，包括代码页，数字、货币、时间和日期的格式等。在Windows内部，其实有两个Locale设置：系统Locale和用户Locale。系统Locale决定代码页，用户Locale决定数字、货币、时间和日期的格式。我们可以在控制面板的“区域和语言选项”中设置系统Locale和用户Locale。

每个Locale都有一个对应的代码页。系统Locale对应的代码页被作为Windows的默认代码页。在没有文本编码信息时，Windows按照默认代码页的编码方案解释文本数据。这个默认代码页通常被称作ANSI代码页（ACP）。



Windows的区域选择

ANSI代码页还有一层意思，就是微软自己定义的代码页。在历史上，IBM的个人计算机和微软公司的操作系统曾经是PC的标准配置。微软公司将IBM公司定义的代码页称作OEM代码页，在IBM公司的代码页基础上作了些增补后，作为自己的代码页，并冠以ANSI的字样。我们在“区域和语言选项”高级页面的代码页转换表中看到的包含ANSI字样的代码页都是微软自己定义的代码页。例如：

874 （ANSI/OEM - 泰文）

932 （ANSI/OEM - 日文 Shift-JIS）

936 （ANSI/OEM - 简体中文 GBK）

949 （ANSI/OEM - 韩文）

950 （ANSI/OEM - 繁体中文 Big5）

1250 （ANSI - 中欧）

1251 （ANSI - 西里尔文）

1252 （ANSI - 拉丁文 I）

1253 （ANSI - 希腊文）

1254 （ANSI - 土耳其文）

1255 （ANSI - 希伯来文）

1256 （ANSI - 阿拉伯文）

1257 （ANSI - 波罗的海文）

1258 （ANSI/OEM - 越南）

### 4.1 Unicode编码

最初，Unicode 编码是被设计为 16 位的，提供了 65,536 个字符的空间。当时人们认为这已经大到足够编码世界上现代文本里所有的文字和字符了。后来考虑到要编码历史上的文字以及一些很少使用的日本汉字和中国汉字[^2]，Unicode 编码扩展到了 21 位（从 U+0000 到 U+10FFFF）。

Unicode 编码空间包含 1,114,112 个编码点。然而，其中只有128,237 个编码点，编码空间的 12% 被赋值。

下面是整个编码空间的布局，一个像素代表一个编码点。使用小方块来表示以保证视觉的一致性；每个小方块是 16×16 = 256 个编码点，每个大方块是一个面有 65536 个 编码点。总共加起来有 17 个面板（见下面Unicode编码面板）。

* 白色表示未用空间；
* 蓝色表示已用空间；
* 绿色表示自用区域；
* 小的红色区域是代理区。

第一个面板被称作『基本多语言面板（Basic Multilingual Plane，简称 BMP）。BMP包含现代文本所需的基本所有字符，包括拉丁文、斯拉夫文、希腊文、汉字（中国），日文、朝鲜文、阿拉伯文、希伯来文、梵文（印度）等等。这个面板就是最初Unicode设计所占用的空间(16位，65536个字符)。后来扩展到现在这个规模，然而，大部分现代字符都在BMP的范围内。

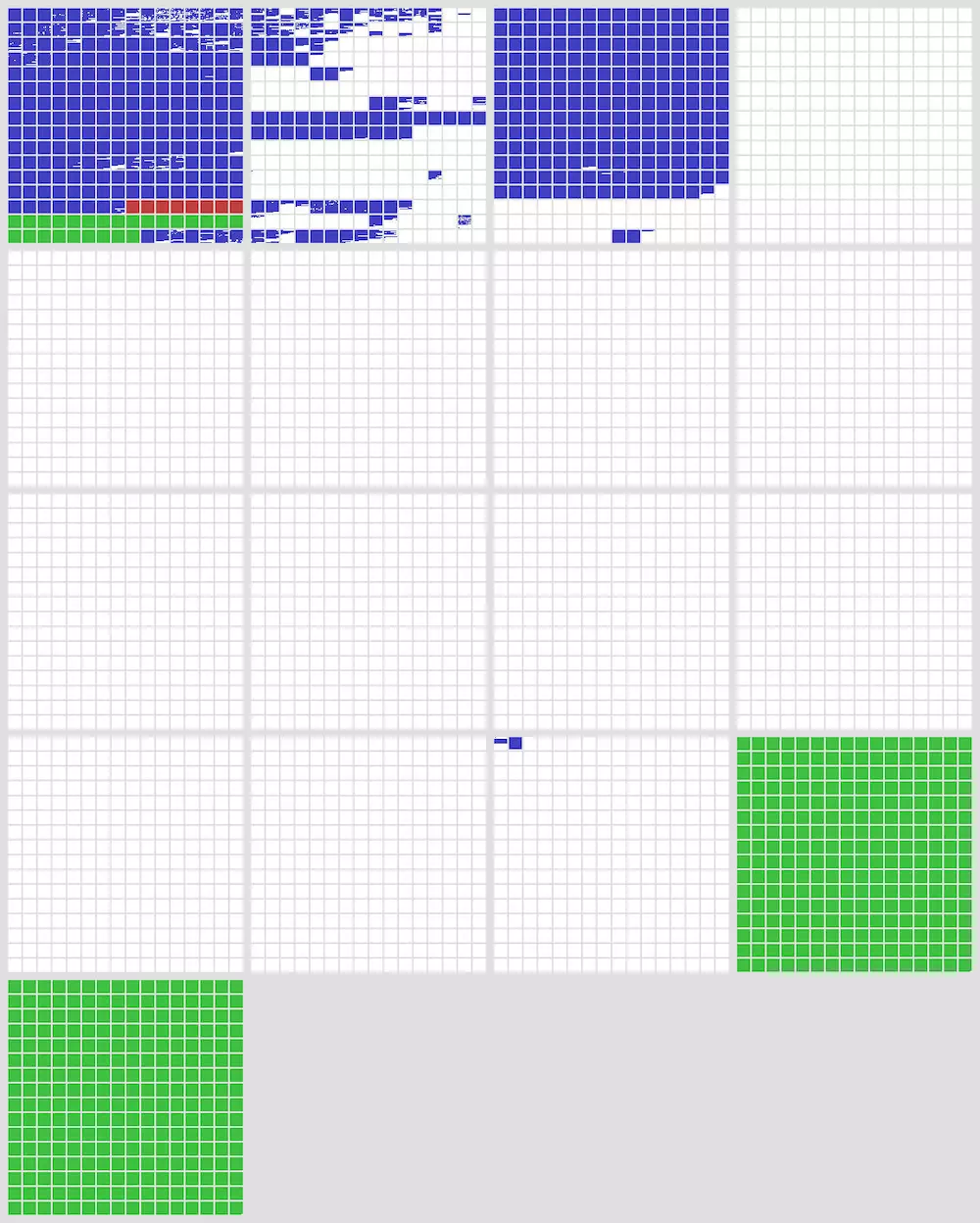
第二个面板则是包括历史上的文字，比如苏美尔楔形文字和埃及象形文字还有emoji表情。

第三个面板包含一大块不常用的和历史上的汉字字符。

剩下的是空的，除了倒数第三个面板中有一小部分被用作格式化字符；

倒数两个面板全部保留自用。

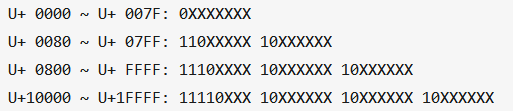
为了和以前的ASCII编码兼容，Unicode的128个字符就是ASCII的拷贝。这样很容易从小编码转向Unicode。由于Unicode被设计为以抽象的方式戴表一个字符，而不规定这个字符如何呈现。如此一来，Unicode 对中文、日文和韩文（CJK）里使用的汉字（也就是所谓的统一汉字）都使用完全相同的码点（这一决定颇具争议），尽管在这些书写系统里，每个汉字都发展出了独特的字形变体。



Unicode编码面板

### 4.2 UTF8编码

Unicode 是「字符集」，UTF-8 是「编码规则」。Unicode 是一个标准，定义了一个字符集以及一系列的编码规则，即 Unicode 字符集和 UTF-8、UTF-16等。UTF-8以 8 位为一个编码单位的可变长编码。会将一个码位编码为 1 到 4 个字节，如下图所示：

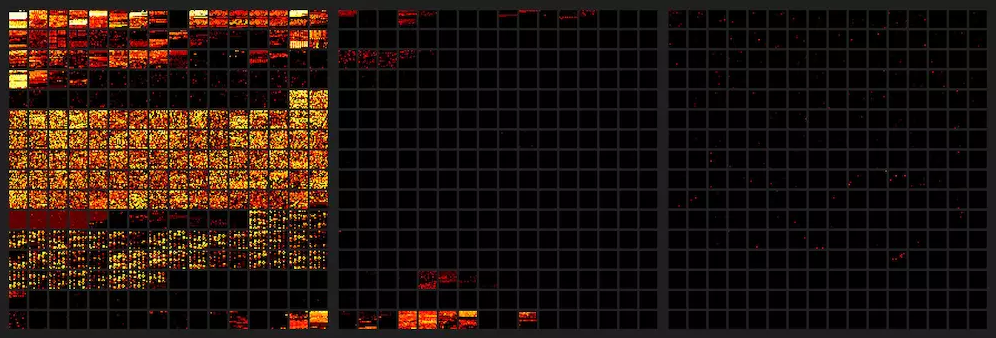


UTF8编码

如果存储在内存或硬盘上Unicode编码最省事的办法就是用32位来存储编码点下标，但是这样的话，每个字符都占四个字节，当你处理大量文本的时候，这样就太浪费内存或带宽了。

编码面板中的前三个面板的使用频率图(编码点使用频率图)。频率增长的方向是黑（没出现）、红、黄、白。可以看到，绝大多数文本分布在BMP内，有些零散的使用来自第二三个面板。第二个面板下高频率使用的字符则是部分emoji表情。

为了解决Unicode编码占据的内存问题，unicode就有了几个紧凑的编码。32位整数编码被称作 UTF-32（UTF=”Unicode Transformation Format”），但是很少被用来存储。最常见的是UTF-8。从上面的热力图可知两个编码涵盖的是最常见的文本，内存能最大程度的利用。这些都是可变长度编码，分别由 8-bit 或 16-bit 为一个单元组成。这些方案中，下标值较小的编码点占用的字节数也少，会节省不少内存。这样做的代价是处理UTF-8需要以编程的方式来处理。



编码点使用频率图

### 4.3 组合字符

Unicode 包含一个系统，可以合并多个编码点，动态组合字符。此系统用各种方式增加灵活性，而不引起编码点的巨大组合膨胀。Unicode 支持各种各样的变音符号，包括尖音符号的和重音符号、元音变音符号、变音符号等等。所有这些变音符可以被使用在任何字母表的字母中。事实上，多个变音符号可以被使用在一个字母上。

如果 Unicode 试图为每个字母组合或变音符组合分配一个独立的编码点，事情会变得无法控制。相反，动态组合系统可以让你构造你想要的任何字符，通过以一个基础编码点（字母）开始然后附加额外的编码点，被称作“组合标识”，来指定变音符。当一个文字渲染器看到字符串中有这样的序列时，它会自动堆叠变音符到基础字母的上面或下面来造出一个组合字符。

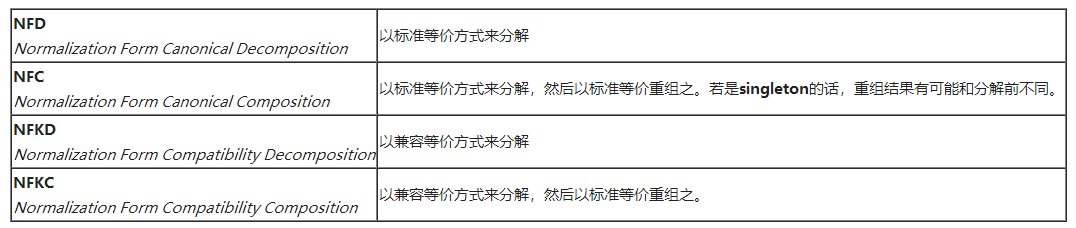
例如，带重音的字符“Á”会被表示成由两个编码点组成的字符串：U+0041 “A” 拉丁大写字母A加上 U+0301 “◌́”组合尖音符号。这个字符串自动被渲染成单个字符：“Á”。

有时候我们会看到某些人的签名中有很奇怪的字符，其实他们就是利用了组合字符。比如Á́́ 就是多添加了几个尖音符号:U+0041U+0301U+0301U+0301。

Unicode 中，预设字符和动态组合系统并存。后果就是有多种方法表示同一个字符串——不同编码点序列产生相同用户可感知的字符。例如，我们之前看到的，表示字符 “Á”，我们可以用一个编码点 U+00C1 ，也可以用两个编码点 U+0041 和U+0301。要解决这个等值字符串的问题，Unicode 定义了几种形式正规化方法，比如NFD和NFC。

正规形式

统一码定义了四种正规形式。这些形式或其（转换）算法表列如下。所有的形式都会使用到标准顺序，好让结果序列能保证是等价类中的唯一形式。这些算法都是幂等转换，但因为singletons和标准顺序的关系，都不是单射。 此外，也没有一个形式在字符串串接下会是封闭的，意味着在同一个正规形式下的两个字符串串接，并不保证产生的是正规形式。 这会发生在当一个字符串并不是以基础字符为开头或以其它中间或结尾字符为开头的字符串附加到另一个字符串时。



具体字符参考：“U0300.pdf-Combining Diacritical Marks Range: 0300–036F”

## 五、常见开发问题

### 5.1 二维码尺寸计算（ESC指令）

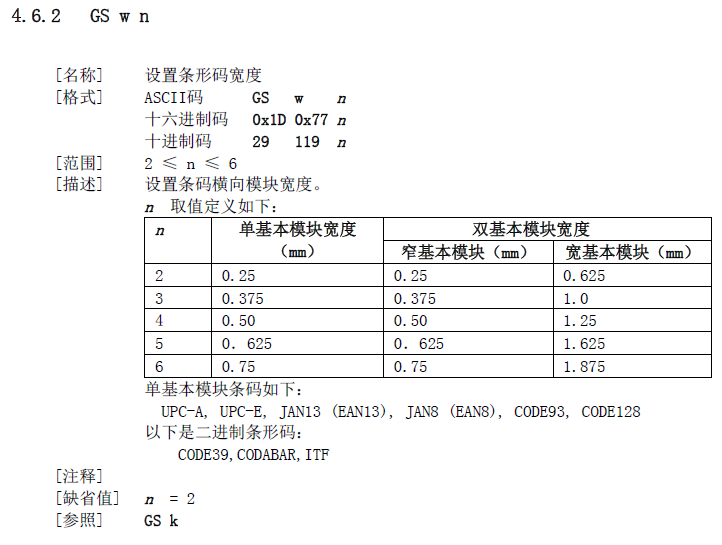
在ESC指令中，有需要打印二维码，并且希望一行中的二维码尺寸一样（二维码内容有长短）。

指令文档请参考：“ZICOX ESC 中文指令手册\_V2.2”

二维码打印相关指令：

#### 5.1.1、设置放大倍数（1D 77）

虽然是一个一维码设置指令，但是同样可以作用于二维码，相当于放大系数设置。



例如：1D 77 04 代表设置 scale=4 放大4倍

#### 5.1.2、设置版本（1D 6F）

设计二维码的尺寸，0标识跟进字符串自动调节。非0则根据如下计算规格计算：

qr\_size=21-4+Ver\*4

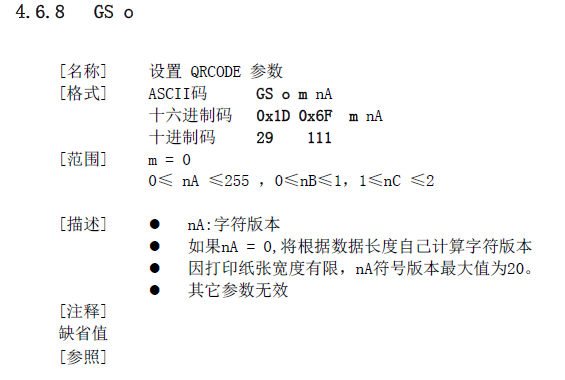
如果字符生成的二维码尺寸超过，则以字符尺寸为主。

因此如果需要固定大小的二维码尺寸可以据此生成符合要求的尺寸。

例如：

1D 6F 04 代表设置Ver=4，则qr\_size=21-4+4\*4=33（像素点）

指令格式如下：

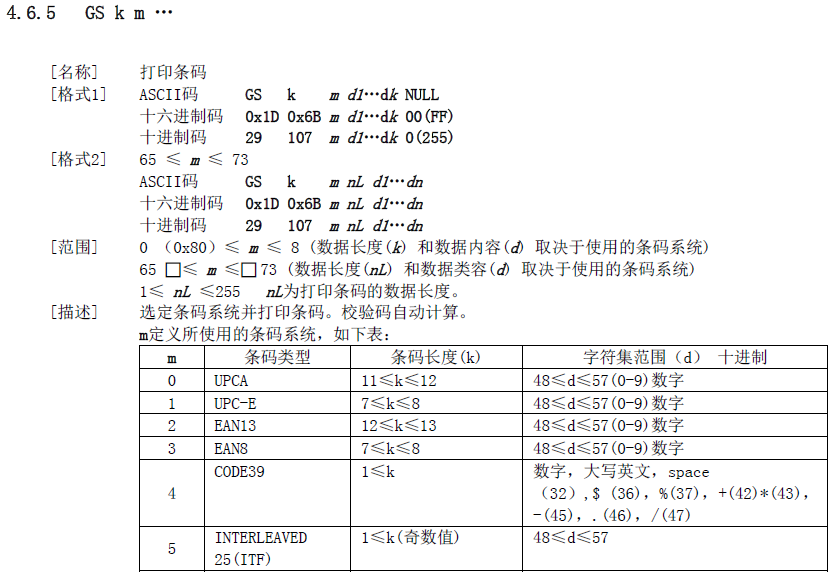


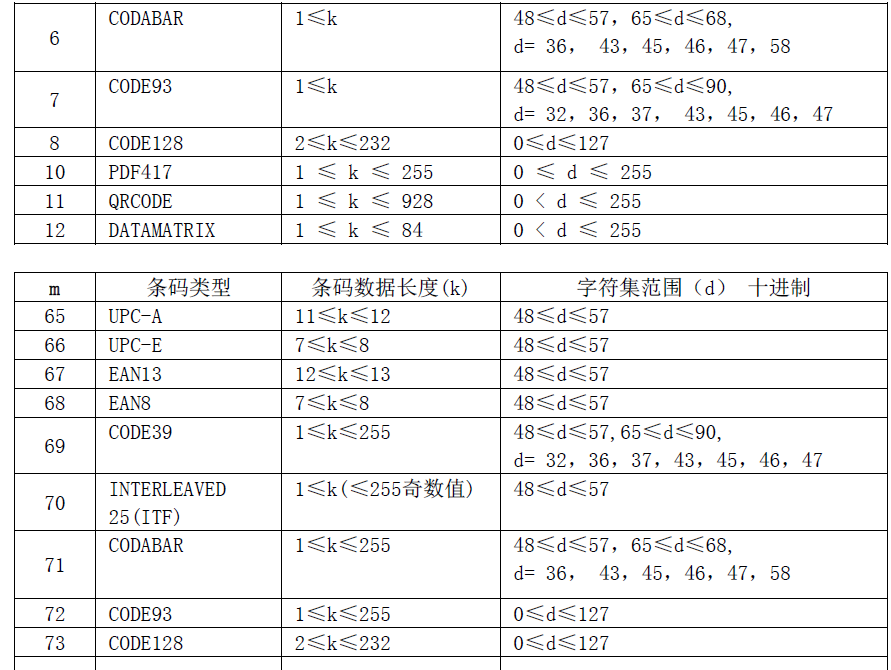
#### 5.1.3、设置二维码数据（1D 6B）

打印二维码数据指令，第三个字节0B标识QR二维码。

例如：

1D 6B 0B … -> QR code

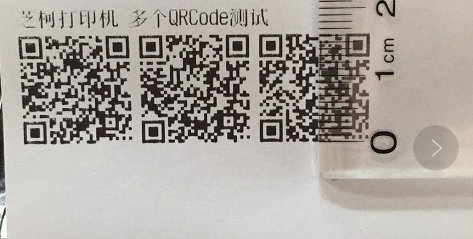




综上，16进制数据：

d6 a5 bf c2 b4 f2 d3 a1 bb fa 20 b6 e0 b8 f6 51 52 43 6f 64 65 b2 e2 ca d4 0a 1b 40 1d 6f 04 1d 77 04 1d 6b 0b 31 32 33 00 20 1d 77 04 1d 6b 0b 68 74 74 70 73 3a 2f 2f 77 65 63 68 61 74 2e 71 71 2e 63 6f 6d 00 20 1d 77 04 1d 6b 0b 68 74 74 70 3a 2f 2f 77 77 77 2e 7a 69 63 6f 78 2e 63 6f 6d 3f 6d 6f 64 65 6c 3d d6 a5 bf c2 00 20 1b 64 04 0a 0a

打印效果：



这里1d 77 04设置放大4倍，Ver=4，因此qr\_size=（21-4+Ver\*4）\*4=33\*4=132（像素点），即132/8=16mm。

#### 5.1.4、一体化指令（1D 28 6B）

此指令为新增指令（但是此指令无法固定二维码大小），具体参考：

* GS ( k <Function 165> QR Code: Select the model

GS ( k pL pH cn fn n1 n2

pL = 4

pH = 0

cn = 49（0x31）

fn = 65（0x41）

n1 = 50 (選擇 Mode 2 0x32)

n2 = 0

HEX： 1D 28 6B 04 00 31 41 32 00

* GS ( k <Function 167> QR Code: Set the size of module

GS ( k pL pH cn fn n

pL = 3

pH = 0

cn = 49（0x31）

fn = 67（0x43）

n = 3 (建議)

HEX： 1D 28 6B 03 00 31 43 03

* GS ( k <Function 169> QR Code: Select the error correction level

GS ( k pL pH cn fn n

pL = 3

pH = 0

cn = 49（0x31）

fn = 69（0x45）

n = 49(建議)

HEX 1D 28 6B 03 00 31 45 31

* GS ( k <Function 180> QR Code: Store the data in the symbol storage area

GS ( k pL pH cn fn m d1...dk

pL = (字串長度+3 ) mod 256

pH = ((字串長度+3 ) / 256 )

cn = 49（0x31）

fn = 80（0x50）

m = 48（0x30）

example:

HEX：

1D 28 6B 83 00 31 50 30 41 42 31 31 32 32 33 33 34 34 31 30 32 30 39 32 36 39 39 39 39 30 30 30 30 30 31 34 35 30 30 30 30 30 31 35 34 30 39 38 37 36 35 34 33 31 32 33 34 35 36 37 38 79 64 58 5A 74 34 4C 41 4E 31 55 48 4E 2F 6A 31 6A 75 56 63 52 41 3D 3D 3A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 3A 34 3A 34 3A 31 3A B0 AE B9 71 A6 C0 3A 31 3A 31 30 35 3A A4 66 B8 6E 3A 31 3A 32 31 30 3A A4 FB A5 A4 3A 31 3A 32 35

* GS ( k <Function 181> QR Code: Print the symbol data in the symbol storage area

GS ( k pL pH cn fn m

pL = 3

pH = 0

cn = 49（0x31）

fn = 81（0x51）

n = 48（0x30）

HEX: 1D 28 6B 03 00 31 51 30

综上：

1B 40 1D 28 6B 04 00 31 41 32 1D 28 6B 03 00 31 43 03

1D 28 6B 83 00 31 50 30 41 42 31 31 32 32 33 33 34 34 31 30 32 30 39 32 36 39 39 39 39 30 30 30 30 30 31 34 35 30 30 30 30 30 31 35 34 30 39 38 37 36 35 34 33 31 32 33 34 35 36 37 38 79 64 58 5A 74 34 4C 41 4E 31 55 48 4E 2F 6A 31 6A 75 56 63 52 41 3D 3D 3A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 3A 34 3A 34 3A 31 3A B0 AE B9 71 A6 C0 3A 31 3A 31 30 35 3A A4 66 B8 6E 3A 31 3A 32 31 30 3A A4 FB A5 A4 3A 31 3A 32 35

1D 28 6B 03 00 31 51 30 0a 0a 0a

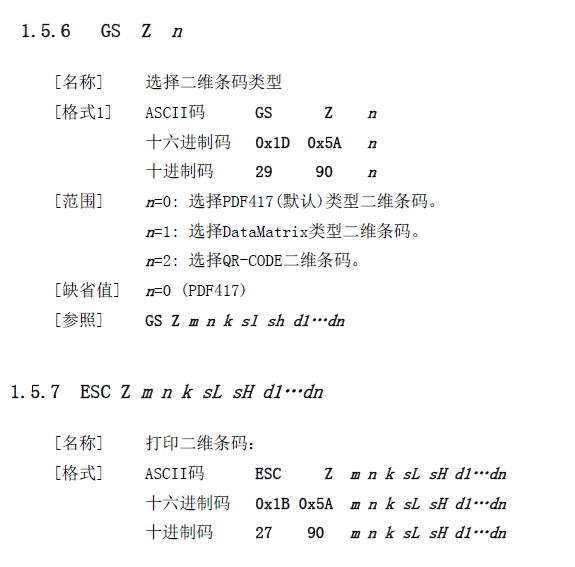
打印效果：

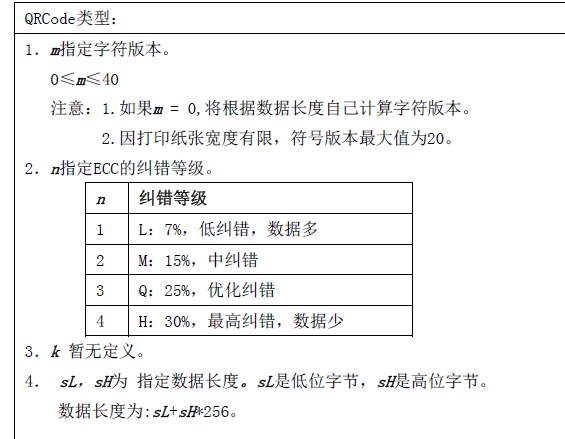


打印效果图

#### 5.1.5、老机型指令（1D 5A）

具体指令参考：“ESC\_POS指令集V1.2.pdf”

****

****

举例：

1B 40 0D 0A 1B 61 01

1D 77 04

1D 5A 02

1B 5A 00 01 00 27 00 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35 36 37 38 39

1B 64 04

打印效果：



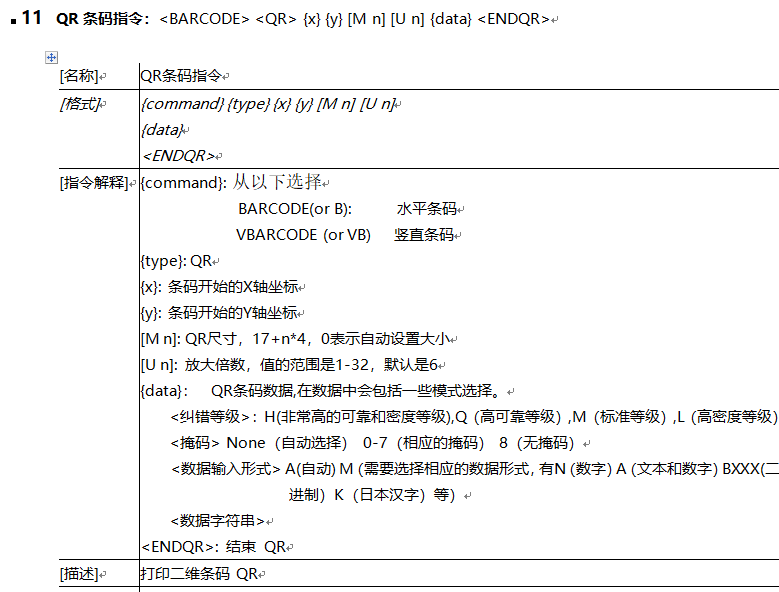
### 5.2 二维码尺寸计算（CPCL指令）

参考：

1、QRCode-编码解码标准.pdf；

2、ZICOX\_CPCL打印指令集1.6.pdf

QR二维码指令格式如下：



[M n]: 代码QR二维码尺寸（二维码版本），为0则自动计算，其他值的计算公式如下：

qr\_size=21-4+M\*4

[U n]:放大倍数

例如CPCL指令如下：

**! 0 200 200 300 1**

**B QR 10 10 M 6 U 3**

**M0,http://bing.com/ki45678jgjk3X\_5gDEB114rUSF9yDZ-12345678941654654**

**ENDQR**

**B QR 300 10 M 6 U 3**

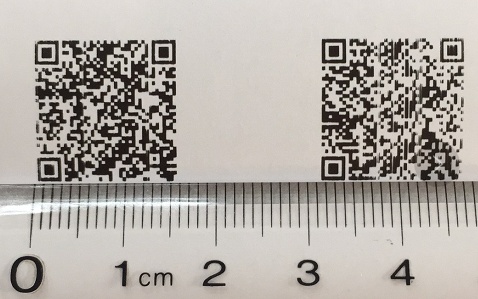
**M0,1234**

**ENDQR**

**PRINT**

根据上述参数，二维码尺寸为41，放大倍数3，最终尺寸123个像素点即15.375mm。

打印效果如下图所示：



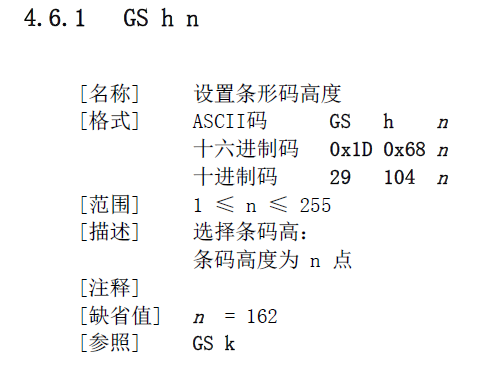
打印效果图



### 5.3一维码（ESC指令）

具体指令参考“ZICOX ESC 中文指令手册\_V2.2”

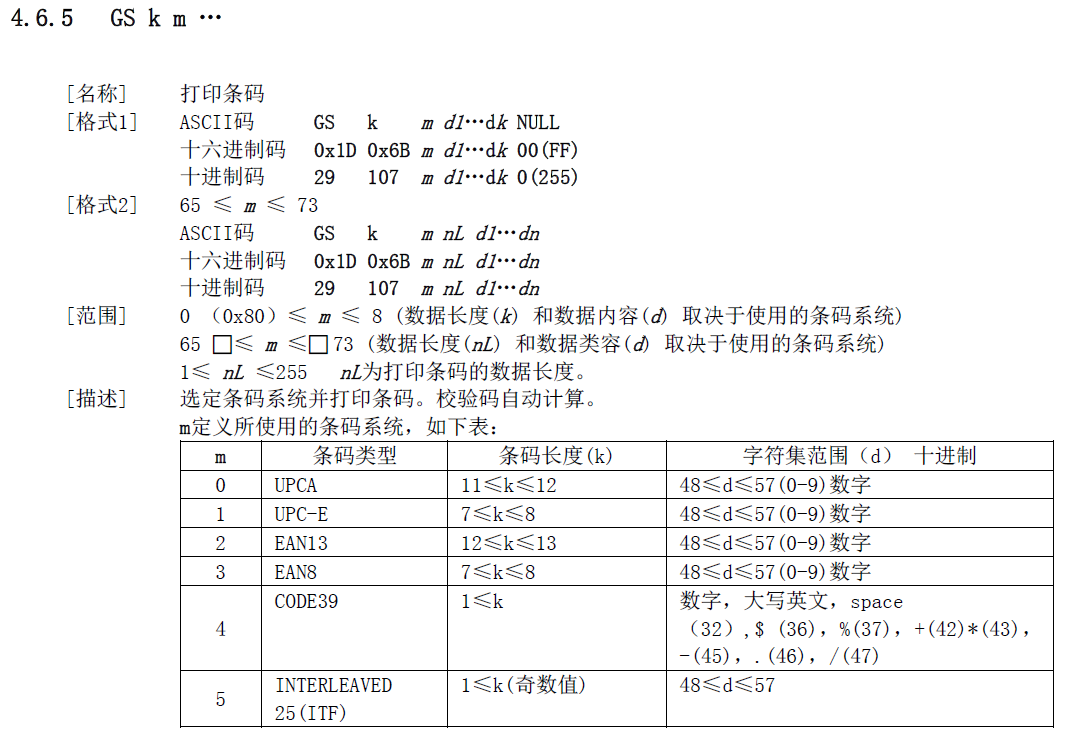
#### 5.3.1 条码高度



#### 5.3.2 条码宽度



#### 5.3.3 条码数据



注意：输出打印需要加回车符号（0x0A）

综上，16进制数据：

1D 68 20 1D 77 02 1D 6B 08 4d 52 34 33 32 34 32 33 35 34 32 33 35 00 0a 0a 0a

打印效果：



一维码打印效果

### 5.4 定位走纸问题

打印机可以走纸到标签缝隙、左黑标、右黑标，可以通过三条指令分别实现。

* 走纸到缝隙：1D 0C
* 走纸到左黑标：0C
* 走纸到右黑标：0E

代码示范：

if(!zp.connect(SelectedBDAddress))

{

Msg(this,"connect fail------", Toast.LENGTH\_LONG).show();

return;

}

…

zp.Write(new byte[]{0x1d,0x0c});

zp.disconnect();

### 5.5 图片打印问题

根据不同指令体系，有不同的图片数据封装方式。

可以借助“图片打印测试工具-VXX.exe”实现各类指令的封装。

程序运行界面如下：



运行界面

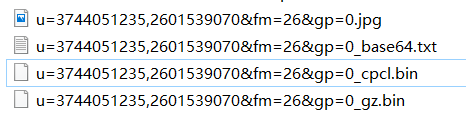
点击原始图片位置即可弹出文件对话框，选择需要处理的图片即可。通过抖动算法可以将彩色图片进行模拟灰度处理，并且可以通过设定灰度阀值来增强对比效果。

效果如下图所示：

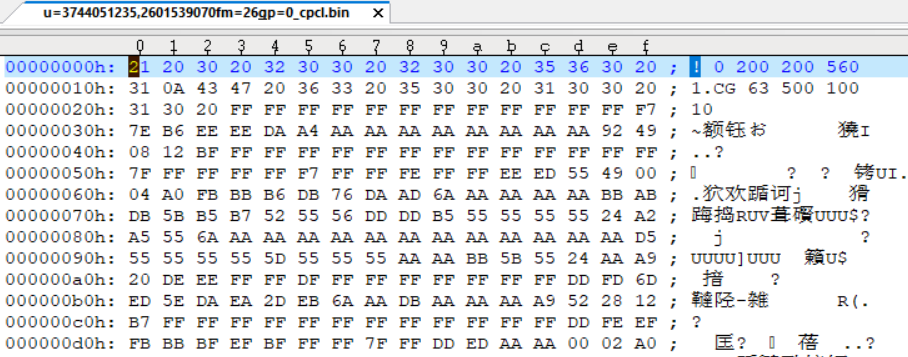


灰度处理效果图

根据不同的指令，可以生成不同指令文件，例如选择CPCL指令，则会图片目录下面生成相应指令文件



如下选择压缩模式则会生成不可见字符，否则可以直接过记事本打开。



指令数据文件内容