

- 一、SDK导入说明
- 二、SDK开放类说明
 - 2.0 PTPrinter
 - 2.1 PTDispatcher
 - 2.2 PTCommandCPCL
 - 2.3 PTCommandESC
 - 2.4 PTCommandTSPL
 - 2.5 PTCommandZPL
 - 2.6 PTEncode
 - 2.7 PTBitmap
 - 2.8 PTBitmap+GrayLevel
 - 2.9 PTLabel

一、SDK导入说明

- 1.把压缩包里面的PrinterSDK.framework拖进你的项目中
- 2.需要另外添加SystemConfiguration.framework这个系统库
- 3.app打包时需要将Enable Bitcode的选项设置成No
- 4.在Build Settings -> Linking -> Other Linker Flags 选项中添加-ObjC

二、SDK开放类说明

2.0 PTPrinter

外设的属性类，比如说外设的名称name、mac地址、蓝牙的广播包advertisement、蓝牙的uuid、信号强度、WiFi相关的router和ip等等，当扫描到外设或者连接外设时，所传的参数就是属性类

- 属性

```
@interface PTPrinter : NSObject

// Printer's distance calculate by RSSI
@property(strong, nonatomic, readwrite) NSString *name;
//打印机Mac地址          @"34:81:F4:24:D7:AB"
//Printer Mac address
@property(strong, nonatomic, readwrite) NSString *mac;
//打印机Mac地址字符串前6位 @"3481F4"
//Former 6 numbers of printer Mac address character string
@property(strong, nonatomic, readwrite) NSString *macKey;
@property(strong, nonatomic, readwrite) NSString *lastTime;
//打印机模型 暂时无效
//Printer model ineffective temporarily
```

```

@property(strong, nonatomic, readwrite) NSString *model;
@property(assign, nonatomic, readwrite) PTPrinterModule module;
//打印机支持的指令类型
//Command type supported by the printer
@property(assign, nonatomic, readwrite) PTCommandMode commandMode;
//打印机厂商
//Printer supplier
@property(assign, nonatomic, readwrite) PTPrinterVender vender;
//蓝牙供应商, 由macKey决定
//Bluetooth supplier, decided by macKey
@property(assign, nonatomic, readwrite) PTBluetoothVender bluetoothVender;
//发现蓝牙时获取到的广播信息
//The broadcast information obtained when Bluetooth is found
@property(strong, nonatomic, readwrite) NSDictionary *advertisement;
// BLE
// 蓝牙外设UUID : Bluetooth peripherals UUID
@property(strong, nonatomic, readwrite) NSString *uuid;
//发现外设时获取到的信号强度值, 单位分贝
//The signal strength value obtained when peripherals are found, unit is db
@property(strong, nonatomic, readwrite) NSNumber *rssi;
//信号强度等级分0-5级,
//Signal strength level is from 0 to 5
@property(strong, nonatomic, readwrite) NSNumber *strength;
//由信号强度计算的距离
//The distance calculated by signal strength
@property(strong, nonatomic, readwrite) NSNumber *distance;
// when value is YES, SDK will get printer's message
@property(assign, nonatomic, readwrite) BOOL notify;
//蓝牙外设 : Bluetooth peripherals
@property(strong, nonatomic, readwrite) CBPeripheral *peripheral;
@property(strong, nonatomic, readwrite) NSDate *time;
// wiFi
@property(strong, nonatomic, readwrite) PTRouter *router;
@property(strong, nonatomic, readwrite) NSString *ip;
@property(strong, nonatomic, readwrite) NSString *port;

```

2.1 PTDispatcher

数据的回调类

- 1.三个枚举分别表示蓝牙的通讯方式、指令CPCL的打印状态回调和连接失败的类型
- 2.定义数据回调的block
- 3.将block当做方法的参数

- 枚举

```
/** 当前通讯方式 : Current communication method */
```

```

typedef NS_ENUM(NSInteger, PTDispatchMode) {

    PTDispatchModeUnconnect    = 0,
    PTDispatchModeBLE          = 1,
    PTDispatchModeWiFi         = 2,
};

typedef NS_ENUM(NSInteger, PTPrintState) {

    // 打印成功 Print success
    PTPrintStateSuccess        = 0xcc00,
    // 打印失败 (缺纸) Print failure (paper out)
    PTPrintStateFailurePaperEmpty = 0xcc01,
    // 打印失败 (开盖) Print failure (cover open)
    PTPrintStateFailureLidOpen   = 0xcc02,
};

typedef NS_ENUM(NSInteger, PTBleConnectError) {

    //连接超时 Connect timeout
    PTBleConnectErrorTimeout    = 0,
    //获取服务超时 Disvocer Service timeout
    PTBleConnectErrorDisvocerServiceTimeout = 1,
    //验证超时 validation timeout
    PTBleConnectErrorValidateTimeout = 2,
    //未知设备 unkown device
    PTBleConnectErrorUnknownDevice = 3,
    //系统错误, 由coreBluetooth框架返回
    //System error, returned by coreBluetooth
    PTBleConnectErrorSystem      = 4,
    //验证失败 validation failure
    PTBleConnectErrorValidateFail = 5
};

```

- 属性

```

/** connected device */
@property (strong, nonatomic, readwrite) PTPrinter
*printerConnected;

@property (assign, nonatomic) PTDispatchMode
mode;

@property (weak, nonatomic, readonly) CBCentralManager*
centralManager;
/** send data success */

```

```

@property (copy, nonatomic, readwrite) PTNumberParameterBlock
sendSuccessBlock;
/** send data fail */
@property (copy, nonatomic, readwrite) PTEmptyParameterBlock
sendFailureBlock;
/** send progress */
@property (copy, nonatomic, readwrite) PTNumberParameterBlock
sendProgressBlock;
/** receiveData */
@property (copy, nonatomic, readwrite) PTDataParameterBlock
receiveDataBlock;
/** printStatus */
@property (copy, nonatomic, readwrite) PTPrintStateBlock
printStatsBlock;
/** findDevice */
@property (copy, nonatomic, readwrite) PTPrinterParameterBlock
findBluetoothBlock;
/** findAllDevice */
@property (copy, nonatomic, readwrite) PTPrinterDictionaryBlock
findBluetoothAllBlock;
/** findAllDevice */
@property (copy, nonatomic, readwrite) PTPrinterDictionaryBlock
findWiFiAllBlock;
/** connecte success */
@property (copy, nonatomic, readwrite) PTEmptyParameterBlock
connectSuccessBlock;
/** connecte fail*/
@property (copy, nonatomic, readwrite) PTBluetoothConnectFailBlock
connectFailBlock;
/** unconnect */
@property (copy, nonatomic, readwrite) PTNumberParameterBlock
disconnectBlock;
/** rssi */
@property (copy, nonatomic, readwrite) PTNumberParameterBlock
readRSSIBlock;

```

- 开始扫描蓝牙

```

/**
    Start scanning Bluetooth
    开始扫描蓝牙
    */
- (void)scanBluetooth;

```

- 停止扫描蓝牙

```
/**
  Stop scanning Bluetooth
  停止扫描蓝牙
 */
- (void)stopScanBluetooth;
```

- 扫描WiFi

```
/**
  扫描Wi-Fi
  Scan Wi-Fi

  @param wifiAllBlock 扫描Wi-Fi成功回调Block, 参数为打印机字典 Block is
  triggered when successfully scanning wi-Fi, parameter is printer
  dictionary
  */
- (void)scanWiFi:(PTPrinterDictionaryBlock)wifiAllBlock;
```

- 连接外设

```
/**
  连接打印机
  Connect printer

  @param printer 连接的打印机 connected printer
  */
- (void)connectPrinter:(PTPrinter *)printer;
```

- 断开外设连接

```
/**
  断开打印机连接
  Disconnect printer

  @param printer 要连接的打印机 the printer to connect
  */
- (void)unconnectPrinter:(PTPrinter *)printer;
```

- 发现蓝牙

```

/**
 发现蓝牙回调，coreBluetooth框架每发现一台打印机就会调用（会重复发现打印机）
  Trigger this method when finding Bluetooth, coreBluetooth will
  trigger it when finding one printer (will find printer repeatably)
  @param bluetoothBlock 回调，参数为发现的打印机对象 trigger, parameter is
  the printer object found
  */
- (void)whenFindBluetooth:(PTPrinterParameterBlock)bluetoothBlock;

```

- 获取已发现的外设

```

/**
 获取已发现的所有打印机，每新发现新的打印机或隔三秒调用一次 不建议使用
  Get all the printers found, trigger once when finding new printer or
  trigger once every 3 seconds

  @param bluetoothAllBlock 回调Block，参数为打印机字典 trigger Block,
  parameter is printer dictionary
  */
- (void)whenFindBluetoothAll:
  (PTPrinterDictionaryBlock)bluetoothAllBlock;

/**
 获取已发现的所有打印机，每新发现新的打印机或隔三秒调用一次 建议使用
  Get all the printers found, trigger once when finding new printer or
  trigger once every 3 seconds

  @param bluetoothBlock 回调Block，参数为打印机数组 trigger Block,
  parameter is printer dictionary
  */
- (void)whenFindAllBluetooth:
  (PTPrinterMutableArrayBlock)bluetoothBlock;

```

- 连接设备更新RSSI

```

/**
 连接设备更新RSSI回调
  Trigger this method when connecting new device to update RSSI

  @param readRSSIBlock 回调block 参数为连接打印机信号强度 trigger block,
  parameter is the signal strength of connecting printer
  */
- (void)whenReadRSSI:(PTNumberParameterBlock)readRSSIBlock;

```

- 连接成功

```
/**
 连接成功回调
  Trigger this method when connecting successfully

  @param connectSuccessBlock 回调block : trigger block
  */
- (void)whenConnectSuccess:(PTEmptyParameterBlock)connectSuccessBlock;
```

- 连接失败

```
/**
 出现连接错误时
  when connect error is occurred

  @param connectFailBlock 带连接错误参数的block block with connect error
  parameter
  */
- (void)whenConnectFailurewithErrorBlock:
(PtBluetoothConnectFailBlock)connectFailBlock;
```

- 断开连接

```
/**
 断开连接回调
  Trigger this method when disconnecting

  @param unconnectBlock 回调block : trigger block
  */
- (void)whenUnconnect:(PTNumberParameterBlock)unconnectBlock;
```

- 发送数据

```
/**
 发送数据
  Send data

  @param data 要发送的数据 the data to send
  */
- (void)sendData:(NSData *)data;
```

- 数据发送成功

```
/**
 数据发送成功
  Data send success

  @param sendSuccessBlock 回调block : trigger block
  */
- (void)whenSendSuccess:(PTNumberParameterBlock)sendSuccessBlock;
```

- 数据发送失败

```
/**
 数据发送失败
  Data send failure

  @param sendFailureBlock 回调block
  */
- (void)whenSendFailure:(PTEmptyParameterBlock)sendFailureBlock;
```

- 数据发送进度

```
/**
 数据发送进度
  Data sending progress

  @param sendProgressBlock 回调block, 参数是发送进度0~1
  */
- (void)whenSendProgressUpdate:
(PTNumberParameterBlock)sendProgressBlock;
```

- 蓝牙接收到数据后返回数据

```
/**
 蓝牙接收到数据回调, 只有蓝牙接收到数据就会有回调
  Trigger this method when Bluetooth receives the data, only when
  Bluetooth receives data can it trigger

  @param receiveDataBlock 回调block, 参数时接收到的data数据
  */
- (void)whenReceiveData:(PTDataParameterBlock)receiveDataBlock;
```


- 接收到打印机状态

```
/**
 接收到打印机打印状态回调
  Trigger this method when receiving print state

  @param printStateBlock 回调block, 参数为打印状态枚举
  */
- (void)whenUpdatePrintState:(PTPrintStateBlock)printStateBlock;
```

- 获取蓝牙中心蓝牙是否打开

```
/**
 获取蓝牙中心蓝牙是否打开, 蓝牙中心刚初始化时, 状态值默认时未开启的
  Get the information of whether Bluetooth of center device is ON, when
  the center device is initializing, its status defaults to OFF.

  @return 蓝牙开启状态
  */
- (BOOL)isBluetoothStatePowerOn;
```

- 设置蓝牙连接超时时间

```
/**
 设置蓝牙连接超时时间
  Set the time of Bluetooth timeout

  @param timeout 时间, 单位为秒, 大于0 : time, unit is second, > 0
  */
- (void)setupBleConnectTimeout:(double)timeout;
```

- 获取SDK版本信息

```
- (NSString *)SDKVersion;
- (NSString *)SDKBuildTime;
- (NSString *)SDKDescription;
```

- 设置过滤一些扫描到的外设

```
/**
 设置外设过滤block, 比如将扫描到没有蓝牙名称的外设和不可连接的外设过滤
```

```

    @param block 过滤block
    */
    - (void)setupPeripheralFilter:(PTPeripheralFilterBlock)block;

eg:
PTDispatcher.share().setupPeripheralFilter {
    guard let _ = $0.0?.name else { return false }
    if let connectable = $0.1?[CBAdvertisementDataIsConnectable] as?
NSNumber {
        if !connectable.boolValue {
            return false
        }
    }
    return true
}

```

- 设置SDK中心

```

/**
    设置SDK中心，当需要SDK兼容你自己创建的蓝牙代理方法，又不想再创建中心时可使用此接口

    @param manager 中心
    @param delegate 接收中心代理消息的对象
    */
    - (void)registerCentralManager:(CBCentralManager *)manager delegate:
(id<CBCentralManagerDelegate>)delegate;

```

- 注销代理

```

    - (void)unregisterDelegate;

```

2.2 PTCommandCPCL

CPCL指令接口类，详情在cpcl指令接口文档

2.3 PTCommandESC

ESC指令接口类，详情在ESC指令接口文档

2.4 PTCommandTSPL

TSPL指令接口类，详情在TSPL指令接口文档

2.5 PTCommandZPL

ZPL指令接口类，详情在ZPL指令接口文档

2.6 PTEncode

发送Text的编码类，默认是kCFStringEncodingGB_18030_2000的编码

- 编码

```
/** encoding */
+ (NSData *)encodeDataWithString:(NSString *)string;

/** support various encoding */
+ (NSData *)encodeDataWithString:(NSString *)string encodingType:
(CFStringEncoding)encodingType;
```

- 解码

```
/** decoding */
+ (NSString *)decodeStringWithData:(NSData *)data;

/** support various decoding */
+ (NSString *)decodeDataWithString:(NSData *)data encodingType:
(CFStringEncoding)encodingType;
```

2.7 PTBitmap

图片处理类，一般在SDK里已经处理

- 枚举

```
/**
 * Photo compression algorithm supported by MPT-8: ZPL2 compression
algorithm, TIFF compression algorithm, other models please choose
NULL.
 * MPT-8 支持的图片压缩算法:ZPL2压缩算法, TIFF 压缩算法,其他机型选择 NULL
 */
```

```
typedef NS_ENUM(NSInteger, PTBitmapCompressMode) {

    PTBitmapCompressModeNone = 0,
    PTBitmapCompressModeZPL2 = 16,
    PTBitmapCompressModeTIFF = 32,
    PTBitmapCompressModeLZO = 48
};

typedef NS_ENUM(NSInteger, PTBitmapMode) {
    //转为黑白二值图像，适用于黑白图片
    PTBitmapModeBinary = 0,
    PTBitmapModeDithering = 1,
    PTBitmapModeColumn = 2
};
```

- 数据取反

```
/**
 * Data negation processing
 * 数据取反处理 eg: TSPL的位图处理需要取反
 *
 * @param data 输入数据 input data
 * @return 取反输出数据 data negation
 */
+ (NSData *)negativeData:(NSData *)data;
```

- 生成打印机图片数据

```
/**
 * Generate data of printing image for the printer
 * 生成打印机打印图片数据
 *
 * @param image 图片/image
 * @param mode 生成的位图数据类型 简单的黑白二值化或者抖动处理/type of generated
 * bitmap data; simple black and white image or dithering
 * @param compress 数据压缩类型/type of data compression
 * @return 提供给打印机使用的图片数据/image data provided to the printer
 */
+ (NSData *)getImageData:(CGImageRef)image mode:(PTBitmapMode)mode
compress:(PTBitmapCompressMode)compress;

//和上面的接口对比，增加指令类型的参数，目的是当是TSPL的指令时需要先取反
+ (NSData *)getImageData:(CGImageRef)image mode:(PTBitmapMode)mode
compress:(PTBitmapCompressMode)compress command:
(PTBitmapCommand)command;
```

2.8 PTBitmap+GrayLevel

图片处理扩展类

- 处理接口

```
/**
    Get grayscale data of image , gray = red*0.29891 + green*0.58661 +
    blue*0.11448
    获取图像的灰度数据, gray = red*0.29891 + green*0.58661 + blue*0.11448

    @param image 图片 image
    @return 灰度数据, 每个字节代表每个像素的256阶灰度值/grayscale data, each
    byte indicates 256-level grayscale value of each pixel
    */
+ (NSData *)graylevel256Datas:(UIImage *)image;

/**
    Get grayscale data of image , gray = red*0.29891 + green*0.58661 +
    blue*0.11448
    获取图像的灰度数据, gray = red*0.29891 + green*0.58661 + blue*0.11448

    @param image 图片 image
    @param needReverse 是否反转灰度值 0->255 254->1
    @return 灰度数据, 每个字节代表每个像素的256阶灰度值/grayscale data, each
    byte indicates 256-level grayscale value of each pixel
    */
+ (NSData *)graylevel256Datas:(UIImage *)image reverse:
    (BOOL)needReverse;

/**
    Get image from grayscale photo
    从灰度图片数据获取图像

    @param data 灰度数据, 每个字节代表每个像素的256阶灰度值/grayscale data, each
    byte indicates 256-level grayscale value of each pixel
    @param width 图片宽度 image width
    @return 图片 image
    */
+ (UIImage *)imageWithGraylevel256Data:(NSData *)data width:
    (NSUInteger)width;

/**
    grayscale data sharpen
    灰度数据锐化

    @param data 灰度数据, 每个字节代表每个像素的256阶灰度值/grayscale data, each
    byte indicates 256-level grayscale value of each pixel
    @return 经过锐化的256阶灰度数据 sharpening 256-level grayscale data
```

```

*/
+ (NSData *)sharpenGraylevel256Data:(NSData *)data width:
(NSUInteger)width;

/**
    Get the data with thermal compensation processing
    获取经过热补偿处理的数据

    @param data 灰度数据, 每个字节代表每个像素的256阶灰度值/grayscale data, each
byte indicates 256-level grayscale value of each pixel
    @param width 图片宽度 image width
    @return 处理结果数据 data of processing result
*/
+ (NSData *)historyCompensateGraylevel256Data:(NSData *)data width:
(NSUInteger)width;

/**
    return to the data of black and white image
    返回黑白图像数据

    @param data 灰度数据, 每个字节代表每个像素的256阶灰度值 grayscale data, each
byte indicates 256-level grayscale value of each pixel
    @param width 图片宽度 image width
    @return 黑白图像数据, 每个字节代表一个像素, 每个字节的值为0或255, 0黑色 255白
色/data of black and white image, each byte indicates one pixel, the
value of each byte is 0 or 255, 0 is black, 255 is white
*/
+ (NSData *)binaryDataOneBytePerPixelGraylevel256Data:(NSData *)data
width:(NSUInteger)width;

/**
    grayscale data obtained by importing image into the device grayscale
space
    将图片导入设备灰度空间获取的灰度数据

    @param image 图片
    @return 灰度数据, 每个字节代表每个像素的256阶灰度值/grayscale data, each
byte indicates 256-level grayscale value of each pixel
*/
+ (NSData *)systemGraylevel256Datas:(UIImage *)image;

```

2.9 PLabel

★★

- 使用电子面单模板, 只需要填充相应的表单数据, 即可发送打印出一张面单。
- 注意: 1. 当使用模板打印时, 您必须填充我们提供的模板使用范例中所填充的所有表单

项。

2. 如果有空数据项，比如申明价值为空，则传入@""空字符串。
3. 不同的模板，所要填充的数据项是不同的，具体以我们的范例为准。

*/

/**

- By using electronic waybill template, only filling in it accordingly can send and print it out.
- Note: 1. When using template to print, you should fill in all the blanks as the template sample showed

2.If there is null data, e.g. claiming value is null, please input null character string @"".

3.The data to fill in differs depending on the template, please subject to the sample showed.

*/

• 属性和方法

```
@interface PTLLabel : NSObject
```

```
@property(strong, nonatomic, readwrite) NSString *express_company;    //  
快递公司
```

```
@property(strong, nonatomic, readwrite) NSString *delivery_number;    //  
运单号
```

```
@property(strong, nonatomic, readwrite) NSString *order_number;    //  
订单号
```

```
@property(strong, nonatomic, readwrite) NSString *distributing;    //  
集散地
```

```
@property(strong, nonatomic, readwrite) NSString *barcode;    //  
条形码
```

```
@property(strong, nonatomic, readwrite) NSString *barcode_text;    //  
条形码下方的字符
```

```
@property(strong, nonatomic, readwrite) NSString *qrcode;    //  
二维码
```

```
@property(strong, nonatomic, readwrite) NSString *qrcode_text;    //  
二维码下方的字符
```

```
@property(strong, nonatomic, readwrite) NSString *receiver_name;    //  
收件人 名字
```

```
@property(strong, nonatomic, readwrite) NSString *receiver_phone;    //  
收件人 电话
```

```
@property(strong, nonatomic, readwrite) NSString *receiver_address;    //  
收件人 地址
```

```
@property(strong, nonatomic, readwrite) NSString *receiver_message;    //  
收件人 信息
```

```
@property(strong, nonatomic, readwrite) NSString *sender_name;        //
发件人 名字
@property(strong, nonatomic, readwrite) NSString *sender_phone;        //
发件人 电话
@property(strong, nonatomic, readwrite) NSString *sender_address;      //
发件人 地址
@property(strong, nonatomic, readwrite) NSString *sender_message;      //
发件人 信息


@property(strong, nonatomic, readwrite) NSString *article_name;        //
物品名
@property(strong, nonatomic, readwrite) NSString *article_weight;      //
物品重量


@property(strong, nonatomic, readwrite) NSString *amount_declare;      //
申明价值
@property(strong, nonatomic, readwrite) NSString *amount_paid;         //
到付金额
@property(strong, nonatomic, readwrite) NSString *amount_paid_advance; //
预付金额


- (NSData *)dataWithSourceFile:(NSString *)filePath;
- (NSData *)dataWithTSPL;
- (NSData *)getTemplateData:(NSString *)source labelDict:(NSDictionary *)labelDict orderDetails:(NSArray *)orderDetails;
- (NSData *)getTemplateData:(NSString *)source labelDict:(NSDictionary *)labelDict;
+ (NSData *)getPaperStauts; // 获取纸张状态


@end
```


