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
5. 使用蓝牙的地方添加#import <CoreBluetooth/CoreBluetooth.h>，视情况而定
6. 注释警告解除方法：Build Settings -> Documentations Comments -> 将YES改为NO
7. 由于CPCL、TSPL接口优化后，老用户如果使用该版本的SDK，则修改的内容较多，因此SDK预留了PTOldCommandCPCL、PTOldCommandTSPL两个类，他们分别保留了之前旧的接口

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

```objectivec
/*!
 *  \~chinese
 *  打印机名称
 *
 *  \~english
 *  Printer name
 */
@property(strong,nonatomic,readwrite) NSString *name;

/*!
 *  \~chinese
 *  打印机mac地址
 *
 *  \~english
 *  Printer mac address
 */
@property(strong,nonatomic,readwrite) NSString *mac;

/*!
 *  \~chinese
 *  打印机蓝牙模块
 *
 *  \~english
 *  Printer Bluetooth module
 */
@property(assign,nonatomic,readwrite) PTPrinterModule module;

/*!
 *  \~chinese
 *  发现蓝牙时获取到的广播信息
 *
 *  \~english
 *  The broadcast information obtained when Bluetooth is found
 */
@property(strong,nonatomic,readwrite) NSDictionary *advertisement;

/*!
 *  \~chinese
 *  蓝牙外设UUID
 *
 *  \~english
 *  Bluetooth peripherals UUID
 */
@property(strong,nonatomic,readwrite) NSString *uuid;

/*!
 *  \~chinese
 *  发现外设时获取到的信号强度值，单位分贝
 *
```

```objc
 *   \~english
 *   The signal strength value obtained when peripherals are found, unit is db
 */
@property(strong,nonatomic,readwrite) NSNumber *rssi;

/*!
 *   \~chinese
 *   信号强度等级分0-5级
 *
 *   \~english
 *   Signal strength level is from 0 to 5
 */
@property(strong,nonatomic,readwrite) NSNumber *strength;

/*!
 *   \~chinese
 *   由信号强度计算的距离
 *
 *   \~english
 *   The distance calculated by signal strength
 */
@property(strong,nonatomic,readwrite) NSNumber *distance;

/*!
 *   \~chinese
 *   蓝牙外设
 *
 *   \~english
 *   Bluetooth peripherals
 */
@property(strong,nonatomic,readwrite) CBPeripheral *peripheral;

/*!
 *   \~chinese
 *   外设的ip地址
 *
 *   \~english
 *   IP
 */
@property(strong,nonatomic,readwrite) NSString *ip;

/*!
 *   \~chinese
 *   端口
 *
 *   \~english
 *   port
 */
@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、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

```
/*!
 *  \~chinese
 *   连接模式
 *
 *  \~english
 *   Connect Mode
 */
typedef NS_ENUM(NSInteger, PTDispatchMode) {
    /*! *\~chinese 未知类型 *\~english Unknown */
    PTDispatchModeUnconnect  = 0,
    /*! *\~chinese 蓝牙 *\~english BLE */
    PTDispatchModeBLE        = 1,
    /*! *\~chinese 无线 *\~english WiFi */
    PTDispatchModeWiFi       = 2
};

/*!
 *  \~chinese
 *   打印完成后打印机返回的状态
 *
 *  \~english
 *   Printer Status
 */
typedef NS_ENUM(NSInteger, PTPrintState) {
    /*! *\~chinese 打印成功 *\~english Print success */
    PTPrintStateSuccess             = 0xcc00,
    /*! *\~chinese 打印失败（缺纸） *\~english Print failure (paper out) */
    PTPrintStateFailurePaperEmpty   = 0xcc01,
    /*! *\~chinese 打印失败（开盖） *\~english Print failure (cover open) */
    PTPrintStateFailureLidOpen      = 0xcc02
};

/*!
 *  \~chinese
 *   返回连接的错误类型
 *
 *  \~english
 *   Connect error
```

```
 */
typedef NS_ENUM(NSInteger, PTConnectError) {

    /*! *\~chinese 连接超时 *\~english Connect timeout */
    PTConnectErrorBleTimeout                    = 0,
    /*! *\~chinese 获取服务超时 *\~english Disvocer Service timeout */
    PTConnectErrorBleDisvocerServiceTimeout,
    /*! *\~chinese 验证超时 *\~english Validation timeout */
    PTConnectErrorBleValidateTimeout,
    /*! *\~chinese 未知设备 *\~english Unkown device */
    PTConnectErrorBleUnknownDevice,
    /*! *\~chinese 系统错误,由coreBluetooth框架返回 *\~english System error,
returned by coreBluetooth */
    PTConnectErrorBleSystem,
    /*! *\~chinese 验证失败 *\~english Validation failure */
    PTConnectErrorBleValidateFail,
    /*! *\~chinese 无线连接超时 *\~english Wifi connect time out */
    PTConnectErrorWifiTimeout,
     /*! *\~chinese socket错误 *\~english Socket error */
    PTConnectErrorWifiSocketError
};
```

- 属性

```
/*!
 *  \~chinese
 *  连接成功后的打印机属性类
 *
 *  \~english
 *  Printer property after connect success
 */
@property (strong, nonatomic, readwrite) PTPrinter
 *printerConnected;

/*!
 *  \~chinese
 *  连接方式
 *
 *  \~english
 *  Connect style
 */
@property (assign, nonatomic) PTDispatchMode                          mode;

/*!
 *  \~chinese
 *  数据发送成功
 *
 *  \~english
```

```
 *  Send data success
 */
@property (copy, nonatomic, readwrite) PTEmptyParameterBlock
sendSuccessBlock;

/*!
 *  \~chinese
 *  数据发送失败
 *
 *  \~english
 *  Send data fail
 */
@property (copy, nonatomic, readwrite) PTEmptyParameterBlock
 sendFailureBlock;

/*!
 *  \~chinese
 *  发送数据的进度条
 *
 *  \~english
 *  Send progress
 */
@property (copy, nonatomic, readwrite) PTNumberParameterBlock
sendProgressBlock;

/*!
 *  \~chinese
 *  接收外设返回的数据
 *
 *  \~english
 *  ReceiveDara
 */
@property (copy, nonatomic, readwrite) PTDataParameterBlock
receiveDataBlock;

/*!
 *  \~chinese
 *  打印完成后返回的状态
 *
 *  \~english
 *  PrintStatus
 */
@property (copy, nonatomic, readwrite) PTPrintStateBlock
 printStateBlock;

/*!
 *  \~chinese
 *  发现外设
 *
```

```
 *   \~english
 *   FindDevice
 */
@property (copy, nonatomic, readwrite) PTPrinterParameterBlock
 findBluetoothBlock;

/*!
 *   \~chinese
 *   发现所有的外设
 *
 *   \~english
 *   FindAllDevice
 */
@property (copy, nonatomic, readwrite) PTPrinterMutableArrayBlock
findAllPeripheralBlock;

/*!
 *   \~chinese
 *   连接成功
 *
 *   \~english
 *   Connect success
 */
@property (copy, nonatomic, readwrite) PTEmptyParameterBlock
 connectSuccessBlock;

/*!
 *   \~chinese
 *   连接失败
 *
 *   \~english
 *   Connect fail
 */
@property (copy, nonatomic, readwrite) PTBluetoothConnectFailBlock
 connectFailBlock;

/*!
 *   \~chinese
 *   断开连接
 *
 *   \~english
 *   unconnect
 */
@property (copy, nonatomic, readwrite) PTUnconnectBlock
unconnectBlock;

/*!
 *   \~chinese
 *   外设的信号强度
```

```
 *
 *  \~english
 *  Rssi
 */
@property (copy, nonatomic, readwrite) PTNumberParameterBlock
readRSSIBlock;

/*!
 *  \~chinese
 *  外设过滤器
 *
 *  \~english
 *  Peripheral filter
 */
@property (copy, nonatomic, readwrite) PTPeripheralFilterBlock
 peripheralFilterBlock;
```

- 方法

```
 /*!
 *  \~chinese
 *  发送数据
 *
 *  @param data  发送的数据
 *
 *  \~english
 *  Send data
 *
 *  @param data Send data
 */
- (void)sendData:(NSData *)data;

/*!
 *  \~chinese
 *  开始扫描蓝牙
 *
 *  \~english
 *  Start scanning Bluetooth
 */
- (void)scanBluetooth;

/*!
 *  \~chinese
 *  停止扫描蓝牙，连接成功后SDK会自动停止扫描
 *
 *  \~english
```

```
 *   Stop scanning Bluetooth, The SDK will automatically stop scanning
after the connection is successful.
 */
- (void)stopScanBluetooth;

/*!
 *   \~chinese
 *   扫描Wi-Fi
 *
 *   @param wifiAllBlock   扫描到的外设数组
 *
 *   \~english
 *   Scan Wi-Fi
 *
 *   @param wifiAllBlock   Scanned peripheral array
 */
- (void)scanWiFi:(PTPrinterMutableArrayBlock)wifiAllBlock;

/*!
 *   \~chinese
 *   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
 *
 *   @param bluetoothBlock   外设数组
 *
 *   \~english
 *   Get all the printers found, trigger once when finding new printer or
trigger once every 3 seconds
 *
 *   @param bluetoothBlock   Scanned peripheral array
 */
- (void)whenFindAllBluetooth:(PTPrinterMutableArrayBlock)bluetoothBlock;

/*!
 *   \~chinese
 *   发现蓝牙回调，coreBlueTooth框架每发现一台打印机就会调用
 *
 *   @param bluetoothBlock   参数为发现的打印机对象
 *
 *   \~english
 *   Trigger this method when finding Bluetooth, coreBlueTooth will trigger
it when finding one printer
 *
 *   @param bluetoothBlock   The parameter is the discovered printer object
 */
- (void)whenFindBluetooth:(PTPrinterParameterBlock)bluetoothBlock;

/*!
 *   \~chinese
 *   连接设备更新RSSI回调
```

```
 *
 *  @param readRSSIBlock  参数是型号强度
 *
 *  \~english
 *  Trigger this method when connecting new device to update RSSI
 *
 *  @param readRSSIBlock  Trigger block, parameter is the signal strength
of connecting printer
 */
- (void)whenReadRSSI:(PTNumberParameterBlock)readRSSIBlock;

/*!
 *  \~chinese
 *  连接打印机
 *
 *  @param printer        要连接的打印机
 *
 *  \~english
 *  Connect printer
 *
 *  @param printer       Connected printer
 */
- (void)connectPrinter:(PTPrinter *)printer;

/*!
 *  \~chinese
 *  断开打印机连接
 *
 *  @param printer  要断开的打印机
 *
 *  \~english
 *  Disconnect printer
 *
 *  @param printer  Unconnect printer
 */
- (void)unconnectPrinter:(PTPrinter *)printer;

/*!
 *  \~chinese
 *  连接成功回调
 *
 *  @param connectSuccessBlock  连接成功的回调
 *
 *  \~english
 *  Trigger this method when connecting successfully
 *
 *  @param connectSuccessBlock  Trigger block
 */
- (void)whenConnectSuccess:(PTEmptyParameterBlock)connectSuccessBlock;
```

```
/*!
 *  \~chinese
 *  连接失败的回调
 *
 *  @param connectFailBlock   连接失败返回的错误类型
 *
 *  \~english
 *  When connect error is occurred
 *
 *  @param connectFailBlock  block block with connect error parameter
 */
- (void)whenConnectFailureWithErrorBlock:
(PTBluetoothConnectFailBlock)connectFailBlock;

/*!
 *  \~chinese
 *  断开连接的回调
 *
 *  @param unconnectBlock   回调的Block
 *
 *  \~english
 *  Trigger this method when disconnecting
 *
 *  @param unconnectBlock  Trigger block
 */
- (void)whenUnconnect:(PTUnconnectBlock)unconnectBlock;

/*!
 *  \~chinese
 *  数据发送成功的回调
 *
 *  @param sendSuccessBlock   回调block
 *
 *  \~english
 *  Callback for successful data transmission
 *
 *  @param sendSuccessBlock  Trigger block
 */
- (void)whenSendSuccess:(PTEmptyParameterBlock)sendSuccessBlock;

/*!
 *  \~chinese
 *  数据发送失败的回调
 *
 *  @param sendFailureBlock   回调block
 *
 *  \~english
 *  Data send failure
```

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

/*!
 *  \~chinese
 *  数据发送进度的回调
 *
 *  @param sendProgressBlock  回调block
 *
 *  \~english
 *  Callback of data transmission progress
 *
 *  @param sendProgressBlock  Trigger block
 */
- (void)whenSendProgressUpdate:(PTNumberParameterBlock)sendProgressBlock;

/*!
 *  \~chinese
 *  接收到数据回调
 *
 *  @param receiveDataBlock  回调block
 *
 *  \~english
 *  Received data callback
 *
 *  @param receiveDataBlock  Trigger block
 */
- (void)whenReceiveData:(PTDataParameterBlock)receiveDataBlock;

/*!
 *  \~chinese
 *  接收到打印机打印状态回调，针对CPCL ESC指令
 *
 *  @param printStateBlock  回调block
 *
 *  \~english
 *  Trigger this method when receiving print state ,For CPCL and ESC
instructions
 *
 *  @param printStateBlock  Trigger block
 */
- (void)whenUpdatePrintState:(PTPrintStateBlock)printStateBlock;

/*!
 *  \~chinese
 *  手机的蓝牙是否打开，需要子线程中执行该接口,YES:打开，NO:关闭
 *
```

```
 *   @return 状态
 *
 *   \~english
 *   Whether the Bluetooth of the mobile phone is turned on, Need to
execute the interface in the child thread,YES:open   NO:close
 *
 *   @return status
 */
- (BOOL)getBluetoothStatus;

/*!
 *   \~chinese
 *   设置蓝牙连接超时时间
 *
 *   @param timeout   超时时间，单位秒
 *
 *   \~english
 *   Set the time of Bluetooth timeout
 *
 *   @param timeout   timeout,unit is second
 */
- (void)setupBleConnectTimeout:(double)timeout;

/*!
 *   \~chinese
 *   设置外设过滤block
 *
 *   @param block   回调block
 *
 *   \~english
 *   Set peripheral filter block
 *
 *   @param block   Trigger block
 */
- (void)setupPeripheralFilter:(PTPeripheralFilterBlock)block;

/*!
 *   \~chinese
 *   设置SDK中心
 *
 *
 *   @param manager          中心
 *   @param delegate         接收中心代理消息的对象
 *
 *   \~english
 *   Set the SDK Center
 *
 *   @param manager       Center
```

```
 *  @param delegate          The object that receives the central proxy
message
 *
 */
- (void)registerCentralManager:(CBCentralManager *)manager delegate:
(id<CBCentralManagerDelegate>)delegate;

/*!
 *  \~chinese
 *  注销代理
 *
 *  \~english
 *  unregister Delegate
 *
 */
- (void)unregisterDelegate;

/*!
 *  \~chinese
 *  SDK版本
 *
 *  @return SDK版本
 *
 *  \~english
 *  SDK version
 *
 *  @return SDK version
 */
- (NSString *)SDKVersion;

/*!
 *  \~chinese
 *  SDK打包时间
 *
 *  @return SDK打包时间
 *
 *  \~english
 *  SDK build time
 *
 *  @return SDK build time
 */
- (NSString *)SDKBuildTime;

/*!
 *  \~chinese
 *  SDK描述
 *
 *  @return SDK描述
 *
```

```
 *   \~english
 *   SDK description
 *
 *   @return SDK description
 */
- (NSString *)SDKDescription;
```

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:
(CFStringEncodings)encodeType;
```

- 解码

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

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

2.7 PTBitmap

> 图片处理类,一般在SDK里已经处理

- 枚举

```
*!
 *   \~chinese
 *   压缩模式
 *
 *   \~english
 *   Compress Mode
 */
typedef NS_ENUM(NSInteger,PTBitmapCompressMode) {

    PTBitmapCompressModeNone = 0,    /*! *\~chinese 不压缩 *\~english None */
    PTBitmapCompressModeZPL2 = 16,   /*! *\~chinese ZPL2压缩算法 *\~english
ZPL2 compress */
    PTBitmapCompressModeTIFF = 32,   /*! *\~chinese TIFF压缩算法 *\~english
TIFF compress */
    PTBitmapCompressModeLZO = 48,    /*! *\~chinese LZO压缩算法 *\~english LZO
compress */
};

/*!
 *   \~chinese
 *   位图模式
 *
 *   \~english
 *   Bitmap Mode
 */
typedef NS_ENUM(NSInteger, PTBitmapMode) {

    PTBitmapModeBinary = 0,     /*! *\~chinese 黑白二值图像 *\~english Binary
*/
    PTBitmapModeDithering = 1,  /*! *\~chinese 灰阶抖动图像 *\~english
Dithering */
    PTBitmapModeColumn = 2,     /*! *\~chinese 无效 *\~english not supported
*/
};
```

```
/*!
 *  \~chinese
 *  哪种指令处理图片
 *
 *  \~english
 *  which instruction handles the image
 */
typedef NS_ENUM(NSInteger, PTBitmapCommand) {

    PTBitmapCommandTSPL = 0,    /*! *\~chinese TSPL指令 *\~english TSPL
command */
    PTBitmapCommandOther = 1,   /*! *\~chinese 其他的指令 *\~english Other
command */
};
```

- 数据取反

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

- 生成打印机图片数据

```
/*!
 *  \~chinese
 *  生成打印机打印图片数据
 *
 *  @param image     图片
 *  @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
 *  @param compress  数据压缩类型
 *  @return          提供给打印机使用的图片数据
 *
 *  \~english
 *  Generate data of printing image for the printer
 *
```

```
 *  @param image    image
 *  @param mode     Type of generated bitmap data; simple black and white
 image or dithering
```

- @param compress Type of data compression

  - @return Image data provided to the printer */

- (NSData *)getImageData:(CGImageRef)image mode:(PTBitmapMode)mode compress:(PTBitmapCompressMode)compress;

/*!

- ~chinese
- 生成打印机打印图片数据 *
- @param image 图片
- @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
- @param compress 数据压缩类型
- @param command 指令类型
- @return 提供给打印机使用的图片数据 *
- ~english
- Generate data of printing image for the printer *
- @param image image
- @param mode Type of generated bitmap data; simple black and white image or dithering
- @param compress Type of data compression
- @param command Type of command
- @return Image data provided to the printer */

- (NSData *)getImageData:(CGImageRef)image mode:(PTBitmapMode)mode compress:(PTBitmapCompressMode)compress command:(PTBitmapCommand)command;

/*!

- ~chinese
- 用column算法生成的图片数据 *
- @param sourceBitmap 输入数据
- @return 位图数据 *
- ~english
- Image data generated by the column algorithm *
- @param sourceBitmap input data
- @return bitmap Data */

- (NSData *)generateColumnData:(CGImageRef)sourceBitmap;

/*!

- ~chinese
- 将bitmap数据转成图片 *
- @param image 图片
- @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
- @return 预览的图片 *
- ~english

- Convert bitmap data to image *
- @param image image
- @param mode Type of generated bitmap data; simple black and white image or dithering
- @return Review Image */

- (UIImage *)generateRenderingWithImage:(CGImageRef)image mode:(PTBitmapMode)mode;

2.8 PTBitmap+GrayLevel

> 图片处理扩展类

* 处理接口

```objective-c
/**
 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
 */
```

```objc
+ (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
 */
```
```objc
+ (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
 */
```
```objc
+ (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
 */
```
```objc
+ (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
 */
```
```objc
+ (NSData *)systemGraylevel256Datas:(UIImage *)image;
```

2.9 PTLabel

- 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 PTLabel : 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
```

2.10 PTOldCommandCPCL

该类是保留SDK3.0.0之前的版本的旧CPCL接口

2.11 PTOldCommandTSPL

该类是保留SDK3.0.0之前的版本的旧TSPL接口

三、如何连接外设说明

连接外设用到的几个方法，具体情况参考Demo

- BLE

```swift
//Swift4.2:

//获取蓝牙状态，需要在子线程调用
PTDispatcher.share().getBluetoothStatus()

//开始扫描蓝牙
PTDispatcher.share().scanBluetooth()

//扫描到的蓝牙，以数组的形式返回
PTDispatcher.share()?.whenFindAllBluetooth({ (array) in

 })

//关闭扫描，连接成功后会自动关闭
PTDispatcher.share().stopScanBluetooth()

//连接打印机
PTDispatcher.share().connect(printer)

//断开连接
PTDispatcher.share().unconnectPrinter(PTDispatcher.share().printerConnect
ed)

//连接成功
PTDispatcher.share().whenConnectSuccess {
 }
//连接失败
PTDispatcher.share().whenConnectFailureWithErrorBlock { (error) in
 }
```

- WiFi

```swift
//要先连接路由器
let router = PTRouter.init()
if router.connected {
    SVProgressHUD.show(withStatus: PRTLocalKey("Scanning network segment,
pls. wait"))
    PTDispatcher.share().scanWiFi({ [weak self](ptArray) in
        SVProgressHUD.showSuccess(withStatus: PRTLocalKey("Scan success"))
        self?.printListArray = ptArray as! [PTPrinter]
        self?.tableView.reloadData()
    })
}else {

  UIAlertController.showActionAlert(PRTLocalKey("Prompt"), message:
PRTLocalKey("Connect router in iOS system setting"), confirm:
PRTLocalKey("Setting"), confirmHandle: { (_) in
```

```
                let url = URL.init(string: "APP-Prefs:root=WIFI")
                UIApplication.shared.openURL(url!)
        })
    }

//连接打印机
PTDispatcher.share().connect(printer)

//断开连接

PTDispatcher.share().unconnectPrinter(PTDispatcher.share().printerConnected)

  //连接成功
PTDispatcher.share().whenConnectSuccess {
  }
//连接失败
PTDispatcher.share().whenConnectFailureWithErrorBlock { (error) in
 }
```

- 发送数据

```
PTDispatcher.share().send(data)

 //进度
PTDispatcher.share()?.whenSendProgressUpdate({ (progress) in


})

//发送成功
PTDispatcher.share().whenSendSuccess {
 }

//发送失败
PTDispatcher.share().whenSendFailure { [weak self] in

 }

// 接收蓝牙返回数据
PTDispatcher.share().whenReceiveData { (temp) in

 }
```

四、指令使用案例

4.0 SDK提供的功能

- 打印格式条形码
- 打印二维码
- 打印文本
- 打印图片（黑白、灰阶抖动）
- 打印小票

通过本 Demo，您可以了解到：

- 如何导入、链接和使用 `PrinterSDK.framework` 框架
- 如何通过 Bluetooth 4.0和 便携式 BLE 蓝牙打印机进行通讯
- 如何通过 WiFi 和便携式 WiFi 打印机进行通讯
- 如何使用打印机指令集中的基础指令，并根据自己的需求分装成为功能

4.1 通过模板打印

通过模板打印，电子面单的样式已经预先编辑好，只需要填充相应的数据项，即可打印输出一张面单。这里以申通快递为例，具体代码见 PTTestTSC 类。

1.填充数据

```
// 使用说明：
// 1. 初始化一个 NSMutableDictionary，在相应的键值下塞入对应的的数据，键值必须是下面样例中用到的键值。
// 2．如果一个数据项没有数据  那么也需要设置成空字符串@""，比如 [templateDict setObject:@"" forKey:kCollection];
PTLabelTemplate *template = [[PTLabelTemplate alloc] init];
NSMutableDictionary *templateDict = [[NSMutableDictionary alloc] init];

[templateDict setObject:@"363604310467" forKey:LTBarcode];
[templateDict setObject:@"上海 上海市 长宁区" forKey:LTDistributing];

[templateDict setObject:@"申大通" forKey:LTReceiver];
[templateDict setObject:@"13826514987" forKey:LTReceiverContact];
[templateDict setObject:@"上海市宝山区共和新路4719弄共和小区12号306室"
forKey:LTReceiverAddress];

[templateDict setObject:@"快小宝" forKey:LTSender];
[templateDict setObject:@"13826514987\r\n" forKey:LTSenderContact];
[templateDict setObject:@"上海市长宁区北瞿路1178号（鑫达商务楼）1号楼305室"
forKey:LTSenderAddress];

[templateDict setObject:@"SHENTONG" forKey:LTExpressCompany];

NSData *cmdData = [template getShenTongTemplate:templateDict];
```

2.读入模板

> 使用时需要把模板文件拖入你的工程中。模板文件是 TXT 纯文本文件。

```
NSString *path = [[NSBundle mainBundle] pathForResource:@"ShenTong"
ofType:@"txt"];
NSData *templateData = [template getTemplateDataWithFilePath:path];
```

3.结合模板和数据

```
NSMutableData *finalData = [[NSMutableData alloc] initWithData:templateData];
[finalData appendData:cmdData];
```

4.发送数据

```
[[PTDispatcher share] sendData:finalData];
```

4.2 CPCL案例

```
PTCommandCPCL *cmd = [[PTCommandCPCL alloc] init];
//初始化标签
[cmd cpclLabelWithOffset:0 hRes:200 vRes:200 height:300 quantity:1];

 UIImage *logo = [UIImage imageNamed:@"abcd.jpg"];
 NSData *bmpData = [PTBitmap getImageData:logo.CGImage
mode:PTBitmapModeDithering compress:PTBitmapCompressModeNone];
[cmd cpclCompressedGraphicsWithImageWidth:logo.size.width
imageHeight:logo.size.height x:20 y:0 bitmapData:bmpData];
//打印
[cmd cpclPrint];
[[PTDispatcher share] sendData:cmd.cmdData];
```

4.3 ESC案例

```
PTCommandESC *cmd = [[PTCommandESC alloc] init];
    [cmd initializePrinter];
    [cmd setJustification:0];
    [cmd setLineSpacing:10];
    UIImage *logoImage = [UIImage imageNamed:@"boliji.jpg"];
    //PTBitmapCompressModeLZO压缩算法:  支持汉码机型
    BOOL ret = [cmd appendRasterImage:logoImage.CGImage mode:PTBitmapModeBinary
compress:PTBitmapCompressModeLZO package:YES];
    [cmd printAndReturnStandardMode];
    if (ret) {
        NSData *sendData = [cmd getCommandData];
```

```
        [PrinterPort sendWithData:sendData];
    }else {
        NSLog(@"The data exceeds the cache and cannot be printed.");
        [ProgressHUD showError:@"print fail"];
    }
```

4.4 TSPL案例

```
PTCommandTSPL *tspl = [[PTCommandTSPL alloc] init];
//清除缓存
[tspl setCLS];
//设置打印区域
[tspl setPrintAreaSizeWithWidth:80 Height:80];
[tspl addBitmapWithXPos:0 YPos:0 Mode:1 image:image.CGImage
bitmapMode:PTBitmapModeBinary compress:PTBitmapCompressModeNone];
//设置打印份数
[tspl printWithSets:1 Copies:1];
[[PTDispatcher share] sendData:tspl.cmdData];
```

4.5 ZPL案例

```
PTCommandZPL *zpl = [[PTCommandZPL alloc] init];

//新的标签起始格式
[zpl XA_FormatStart];
[zpl LL_LabelLength:400];
[zpl PW_PrintWidth:700];
[zpl A_SetFontWithOrientation:@"N" height:50 width:60 location:@"B"
fontName:@"CYRI_UB" extension:@"FNT"];
[zpl FO_FieldOriginwithXAxis:100 YAxis:100];
[zpl FD_FieldData:@"Wireless Printer Fonts"];
//字段分隔符
[zpl FS_FieldSeparator];
//结束格式
[zpl XZ_FormatEnd];
//发送数据
[[PTDispatcher share] sendData:zpl.cmdData];
```