HPRTPrinter

V1.1.2变化（2016.3.21）

API 变动：

- 重写了 ESC-POS 指令集：HPRTCommandESC，所有接口都发生改变。
  - 重新实现了ESC-POS大约200个指令函数，每一个函数都标注了页数，都可以在官方文档《esc-pos product v10.5.pdf》中找到原型。
  - 为什么 SDK 看起来没有功能性说明注释?
    > 由于函数名已经可以充分说明函数的功能，过多注释影响代码可读性。
    >
    > 注释中只说明了参数的取值范围。
  - 如何封装 ESC-POS 基础指令实现功能？见 HPRTFunctionESC。
  - 新的 ESC-POS 指令部分增强了生成指令的可靠性和可维护性。
- TSC-TSPL 指令集改名为：HPRTCommandTSPL。（只改了类名）
- SDK 的通讯部分和指令部分耦合度如何?
  > 调整后的 SDK 结构实现了指令部分和通讯部分的解耦合。客户可以根据自己的需求决定是否自己实现和打印机的通讯部分。

V1.1.1变化（2016.3.10）

API 改动：

- HPRTDispatcher 新增一个协议 `HPRTDispatchProtocol`
  - 同时继续支持原来的用通知获取连接状态
  - 使用协议比使用通知在代码结构上更优雅。

```
@protocol HPRTDispatchProtocol <NSObject>

@optional
/**
 *  每发现一个蓝牙设备调用一次 或者观察通知：
PRTBluetoothUpdatePeripheralList（携带数据）
 */
- (void)didDiscoverPeripheral:(CBPeripheral *)peripheral RSSI:
(NSNumber *)RSSI;

/**
 *  成功连接 BLE 或者观察通知：HPRTDispatchModeChangeToBLE
 */
- (void)didConnectPeripheral:(CBPeripheral *)peripheral;

/**
 *  成功连接 WiFi 或者观察通知：HPRTDispatchModeChangeToWIFI
 */
- (void)didConnectWiFi:(NSString *)ip;
```

- 获取是否和打印机断开连接，依旧是注册通知观察者：`HPRTDispatchModeChangeToOffline`

V1.1.0变化(2016.3.8)：

- 新增 TSC-TSPL 指令集，支持标签打印机。

API 改动：

> - HPRTPrinter改为 HPRTPrinterESC，功能：ESC-POS 指令集
> - 新增 HPRTPrinterTSC，功能：TSC-TSPL 指令集
> - HPRTDispatcher 中的 currentInstructionMode保存当前指令集模式
> - TSC-TSPL 测试案例：HPRTTestTSC
> - ESC-POS 测试案例：HPRTTestESC

一个典型的 TSC-TSPL 指令集使用：

```
HPRTPrinterTSC *tsc = [[HPRTPrinterTSC alloc] init];

[tsc initCMDQueue]; //初始化命令队列(必要)

[tsc setPrintAreaSizeWithWidth:@"100" Height:@"80"];     //设置打印区域
[tsc setCLS];    //清除缓冲区
[tsc printTexTWithXPos:.....]; //打印文本
[tsc printBarCodeWith........]; //打印条形码

[tsc sendCMDQueue];//发送队列中的指令到打印机(必要)

/*
必须在设置指令前初始化命令队列(initCMDQueue)，设置所有指令后发送指令队列
(sendCMDQueue)。
*/
```

- 提示：提供的 Demo 会根据打印机型号自动设定指令集。也可以手动设定指令集。根据不同的指令集更改测试页面。逻辑比较繁杂。但是所有的指令集 api测试案例都在 HPRTTestTSC 和 HPRTTestESC 这两个类，可以直接查看。

V1.0.3变化（2016.2.29）：

- 新增WiFi 连接功能。
- 通讯功能统一为 HPRTDispatcher 接口

Wifi 连接接口：

```
#pragma mark HPRTNetWork

/**
 *  获取当前 Wifi 的路由信息
 *
 *  @return 路由信息字典
 */
- (NSDictionary *)getRouterInfo;

/**
 *  通过 ip 地址连接 wifi
 *
 */
- (void)connectWifi:(NSString *)ip;
```

蓝牙4.0 BLE接口：

```
#pragma mark HPRTBluetooth

/**
 *  扫描蓝牙
 */
- (void)scanBluetooth;

/**
 *  停止扫描蓝牙
 */
- (void)stopScanBluetooth;

/**
 *  连接蓝牙
 *
 */
- (void)connectBluetooth:(CBPeripheral *)peripheral;

/**
 *  断开连接蓝牙
 */
- (void)disConnectBluetooth;
```

发送数据给打印机打印：

```
/**
 *  发送一条数据打印出来
 */
- (void)sendData:(NSData *)data;

/**
 *  打印队列，将需要的一组数据(NSData)加入 NSArray,将会依次打印出来。
 *
 *  @param dataQueue like @[data1,data2,data3]
 */
- (void)sendDataQueue:(NSArray *)dataQueue;
```

断开当前连接：

```
#pragma mark CurrentConnect

/**
 *  取消当前连接（Wifi 或者 BLE）
 */
- (void)disConnectCurrentLine;
```

打印机的连接状态：

```
/*
 *  @enum PRTTransmitDispatchMode
 *
 *  @discussion Represents the current transmit dispatch mode of a PRTTransmitDispatchMode.
 *
 *  @constant PRTPrinterModeOffline     Offline Mode    离线模式
 *  @constant PRTPrinterModeBle         BLE     Mode    蓝牙连接模式
 *  @constant PRTPrinterModeWifi        Wifi    Mode    Wifi连接模式
 *
 */
typedef NS_ENUM(NSInteger, DispatchMode) {

    DispatchModeOffline = 0,
    DispatchModeBLE,
    DispatchModeWIFI,
};

/**
 *  当前通讯调度模式
 */
@property(nonatomic,assign) NSInteger currentDispatchMode;

/**
 *  当前连接的蓝牙
 */
@property(strong,nonatomic,readwrite) CBPeripheral *bluetoothConnected;
```