

目录

1. SDK 导入说明.....	1
2. 初始化.....	1
3. 连接设备.....	2
1. 设定蓝牙参数.....	2
2. 设置蓝牙发现规则.....	2
3. 开启扫描.....	2
4. 连接设备.....	2
5. 关闭扫描.....	3
4. 数据发送.....	3
5. 数据接收.....	3

1. SDK 导入说明

1.把压缩包里面的 `PrinterSDK.framework` 拖进你的项目中

2.TARGETS -> Build Phases -> Link Binary With Libraries 添加 `libc++.tbd`

3.TARGETS -> Build Settings -> Enable Bitcode 设置为 NO

4.TARGETS -> Build Settings -> Other Linker Flags 添加 `-ObjC`

5.PCH 文件导入 `#import <PrinterSDK/PrinterSDK.h>`

2. 初始化

建议使用一个单例蓝牙管理类，`BleManager` 为管理类的属性。具体使用请看 Demo

初始化代码

```
Manager.bleManager = [BleManager new];
```

3. 连接设备

1.设定蓝牙参数

初始化 `BleOptions` 类，把 `BleOptions` 对象设置到蓝牙管理类，具体使用请看 Demo

```
BleOptions *options = [BleOptions new];{
    options.scanForPeripheralsWithOptions = @[CBCentralManagerScanOptionAllowDuplicatesKey:YES];
}
Manager.bleManager.bleOptions = options;
```

2.设置蓝牙发现规则

```
[Manager.bleManager setFilterOnDiscoverPeripherals:^(BOOL(NSString * _Nonnull peripheralName, NSDictionary *
_Nonnull advertisementData, NSNumber * _Nonnull RSSI) {

    // 一般设置名字是否包含某些前缀

    if (peripheralName.length) {
        return YES;
    }
    return NO;
}];
```

一般来说这里根据 BLE 前缀去设定规则，如果不设定则默认发现所有设备

3.开启扫描

开启扫描，返回的设备都是根据设定规则返回的

```
[Manager.bleManager scanPeripheralsWithBlock:^(CBCentralManager * _Nonnull central, CBPeripheral *  
_Nonnull peripheral, NSDictionary * _Nonnull advertisementData, NSNumber * _Nonnull RSSI) {  
  
    // 扫描到的设备  
  
}];
```

4.连接设备

```
[Manager.bleManager connectToPeripheral:peripheral withConnected:^(CBCentralManager * _Nonnull central,  
CBPeripheral * _Nonnull peripheral) {  
  
    // 连接成功  
  
} withFail:^(CBCentralManager * _Nonnull central, CBPeripheral * _Nonnull peripheral, NSError * _Nonnull error)  
  
    // 连接失败  
  
} withState:^(ConnectState state) {  
  
    // 当前的状态  
  
}];
```

5.关闭扫描

```
[Manager.bleManager cancelScan];
```

4. 数据发送

```
// 不带回调  
  
[Manager.bleManager write:data];  
  
[Manager.bleManager write:data progress:^(NSUInteger total, NSUInteger progress) {  
  
    // 写入数据进度  
  
} receCallBack:^(NSData * _Nullable data) {  
  
    // 数据返回  
  
}];
```

```
};
```

5. 数据接收

```
// 被动接收数据

Manager.bleManager.readTotalCallback = ^(NSData * _Nullable data) {

    // 所有数据都会调用这个回调，可用于接收打印机主动返回的数据

};

// 主动发送数据后返回数据

[Manager.bleManager write:data progress:nil receCallBack:^(NSData * _Nullable data) {

    // 数据返回

}];
```