开发指南

开发指南

1 环境配置

1.1 添加framework

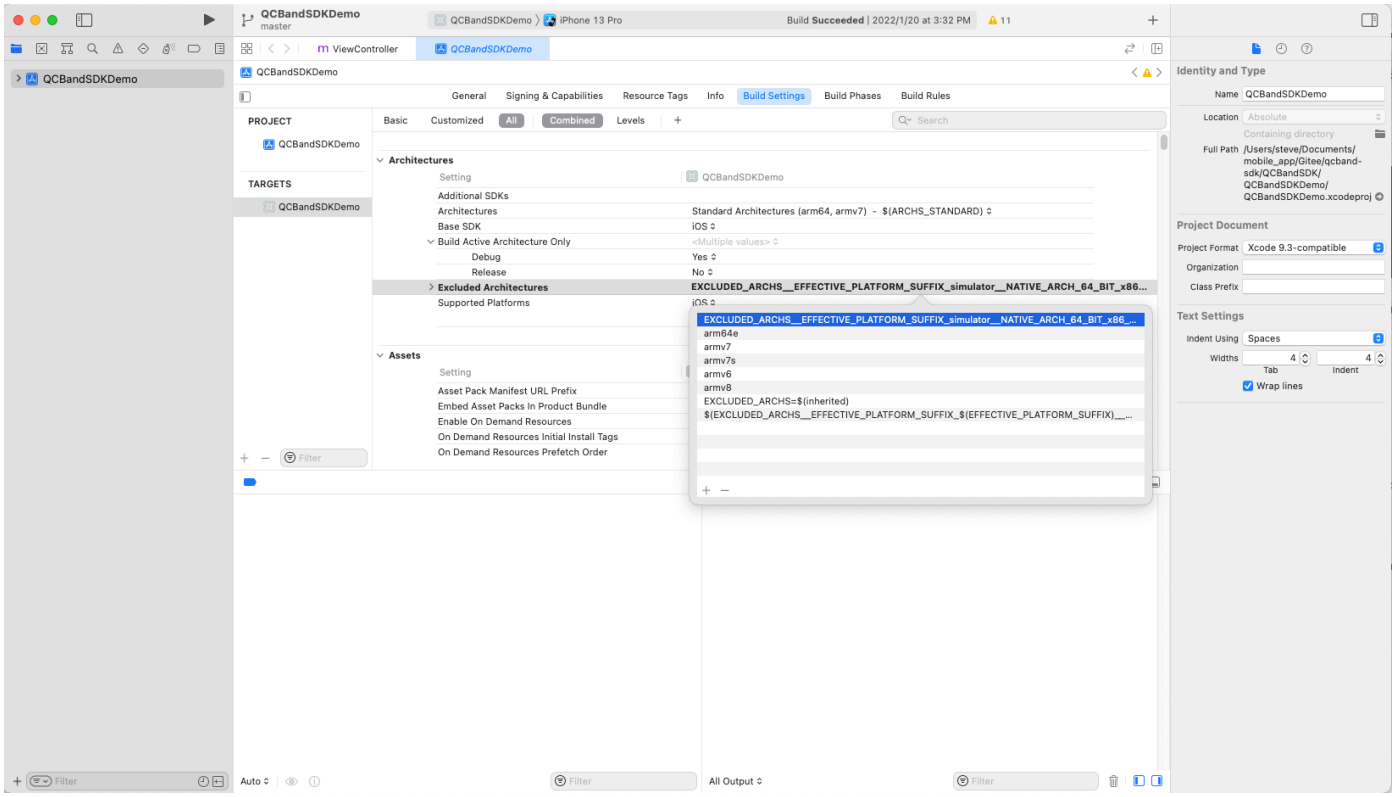
将`QCBandSDk.framework`加入到项目，framework支持iOS 9.0以上版本

注意:因为framework中使用了分类，需要在项目中添加设置

Target->Build Settings -> Other Linker Flags 添加 **-ObjC**

在Target-Build Settings-Excluded Architectures中添加以下代码

```
EXCLUDED_ARCHS__EFFECTIVE_PLATFORM_SUFFIX_simulator__NATIVE_ARCH_64_BIT_x86_64=arm64
arm64e armv7 armv7s armv6 armv8 EXCLUDED_ARCHS=$(inherited)
$(EXCLUDED_ARCHS__EFFECTIVE_PLATFORM_SUFFIX_$(EFFECTIVE_PLATFORM_SUFFIX)__NATIVE_ARCH_64_
BIT_$(NATIVE_ARCH_64_BIT))
```



1.2 配置蓝牙权限

在info.plist文件中配置蓝牙权限

```
<key>NSBluetoothAlwaysUsageDescription</key>
<string>App需要是使用您的蓝牙设备</string>
<key>NSBluetoothPeripheralUsageDescription</key>
<string>App需要是使用您的蓝牙设备</string>
```

2 使用

2.1 设备支持的服务UUID

在`QCSDKManager.h`中定义了，设备支持的服务UUID：

```
extern NSString *const QCBANDSDKSERVERUUID1;
extern NSString *const QCBANDSDKSERVERUUID2;
```

2.2 导入framwrok库

在代码中引入framework库

```
#import <QCBandSDK/QCSDKManager.h>
#import <QCBandSDK/QCSDKCmdCreator.h>
```

使用单例初始化`[QCSDKManager shareInstance]`

> `QCSDKManager`:用于加入连接的外围设备
>
> `QCSDKCmdCreator`:用于向外围设备发送指令

2.3 扫描戒指

初始化

在权限允许和蓝牙开启状态下，才能开始扫描。

导入Apple的CoreBluetooth库, 并遵循两个协议`<CBCentralManagerDelegate, CBPeripheralDelegate>`

```
#import <CoreBluetooth/CoreBluetooth.h>
```

声明中心角色和外围角色

```
/*中心角色,app*/
@property (strong, nonatomic) CBCentralManager *centerManager;

/*外设角色,扫描到的外围设备*/
@property (strong, nonatomic) NSMutableArray<CBPeripheral *> *peripherals;

/*连接的外设角色*/
@property (strong, nonatomic) CBPeripheral *connectedPeripheral;
```

实例化中心角色

```
self.centerManager = [[CBCentralManager alloc] initWithDelegate:self queue:nil];
```

扫描戒指

使用扫描外围设备

```
NSArray *serviceUUIDStrings = @[QCBANDSDKSERVERUUID1,QCBANDSDKSERVERUUID2];

NSMutableArray *uuids = [NSMutableArray array];
for (id obj in serviceUUIDStrings) {
    if ([obj isKindOfClass:[NSString class]]) {
        CBUUID *uuid = [CBUUID UUIDWithString:obj];
        [uuids addObject:uuid];
    }
}

NSDictionary *option = @{CBCentralManagerScanOptionAllowDuplicatesKey : [NSNumber
numberWithBool:NO]};
[self.centerManager scanForPeripheralsWithServices:uuids options:option];
```

注:在代理中获取扫描到的外围设备,可以通过设备名称等相关信息，进行二次过滤。

```
- (void)centralManager:(CBCentralManager *)central didDiscoverPeripheral:(CBPeripheral
*)peripheral advertisementData:(NSDictionary<NSString *,id> *)advertisementData RSSI:
(NSNumber *)RSSI {
    if (peripheral.name.length > 0) {
        [self.peripherals addObject:peripheral];
        [self.deviceList reloadData];
    }
}
```

2.4 取消扫描设备

调用中心角色的接口，停止扫描

```
[self.centerManager stopScan];
```

2.5 连接戒指

开始连接

```
self.connectedPeripheral = self.peripherals[indexPath.row];
[self.centerManager connectPeripheral:self.connectedPeripheral options:nil];
```

连接成功后，传入外围设备到SDK

```
- (void)centralManager:(CBCentralManager *)central didConnectPeripheral:(CBPeripheral
*)peripheral {
    [[QCSDKManager shareInstance] addPeripheral:peripheral];
}
```

2.6 断开连接

```
[self.centerManager cancelPeripheralConnection:self.connectedPeripheral];
```

断开连接后，删除外围设备

```
- (void)centralManager:(CBCentralManager *)central didDisconnectPeripheral:(CBPeripheral
*)peripheral error:(nullable NSError *)error {
    [[QCSDKManager shareInstance] removePeripheral:peripheral];
}
```

3. 戒指支持的指令

3.1 设置手环时间

```
/**
 *  Set the time of the watch
 *  设置戒指的时间
 *
 */
+ (void)setTime:(NSDate *)date success:(void (^)(NSDictionary *featureList))suc failed:
(void (^)(void))fail;
```

3.2 读取戒指电量

```
/*!
 *  @func 读取戒指电量
 *  @param suc battery:电量等级(0~8)
 */
+ (void)readBatterySuccess:(void (^)(int battery))suc failed:(void (^)(void))fail;
```

3.3 绑定戒指亮灯

```
/**
 *  绑定戒指亮灯
 */
+ (void)alertBindingSuccess:(nullable void (^)(void))suc fail:(nullable void (^)
(void))fail;
```

3.4 设置戒指时间进制/用户个人信息

```
/**
 设置戒指时间进制/用户个人信息
 @param twentyfourHourFormat    : YES 24小时制; NO 12小时制
 @param metricSystem            : YES 公制; NO 英制
 @param gender                  : 性别 (0=男, 1=女)
 @param age                     : 年龄 (岁)
```

```
 @param height                  : 身高（厘米）
 @param weight                  : 体重（kg）
 @param sbpBase                 : 收缩压基值（mmhg）(保留值，可设置0)
 @param dbpBase                 : 舒张压基值（mmhg）(保留值，可设置0)
 @param hrAlarmValue            : 心率报警值（bpm）(保留值，可设置0)
 */
+ (void)setTimeFormatTwentyfourHourFormat:(BOOL)twentyfourHourFormat
    metricSystem:(BOOL)metricSystem
    gender:(NSInteger)gender
    age:(NSInteger)age
    height:(NSInteger)height
    weight:(NSInteger)weight
    sbpBase:(NSInteger)sbpBase
    dbpBase:(NSInteger)dbpBase
    hrAlarmValue:(NSInteger)hrAlarmValue
    success:(void (^)(BOOL, BOOL, NSInteger, NSInteger, NSInteger, NSInteger, NSInteger,
NSInteger, NSInteger))success
    fail:(void (^)(void))fail;
```

3.5 获取戒指时间进制/用户个人信息

```
/**
 获取戒指时间进制/用户个人信息
 @param success isTwentyfour: YES 24小时制; NO 12小时制
               isMetricSystem: YES 公制; NO 英制
               gender: 性别 (0=男, 1=女)
               age: 年龄（岁）
               height: 身高（厘米）
               weight: 体重（kg）
               sbpBase: 收缩压基值（mmhg）(保留值，可设置0)
               dbpBase: 舒张压基值（mmhg）(保留值，可设置0)
               hrAlarmValue: 心率报警值（bpm）(保留值，可设置0)
 */
+ (void)getTimeFormatInfo:(nullable void (^)(BOOL isTwentyfour, BOOL isMetricSystem,
NSInteger gender, NSInteger age, NSInteger height, NSInteger weight, NSInteger sbpBase,
NSInteger dbpBase, NSInteger hrAlarmValue))success fail:(nullable void (^)(void))fail;
```

3.6 获取戒指固件的版本号

```
/**
 *  @func 获取戒指固件(Application)的版本号
 *  @param success 软件硬件版本号格式一般为"x.x.x"
 */
+ (void)getDeviceSoftAndHardVersionSuccess:(void (^)(NSString *_Nonnull, NSString
*_Nonnull))success fail:(void (^)(void))fail;
```

3.7 获取当前计步信息

```
/*!
 *  @func    获取当前计步信息(可以同步最新纪录)
 */
+ (void)getCurrentSportSucess:(void (^)(SportModel *sport))suc failed:(void (^)
(void))fail;
```

3.8 获取某天总的统计数据

```
/*!
 *  @func    获取某天总的统计数据
 *  @param    index   : 0->当天 1->1天前.最大29
 *  @param    suc     :使用这个命令不能准确的获取当天的统计数据,设备没15分钟会存一次数据,所以会有15分钟
间隔
 */
+ (void)getOneDaySportBy:(NSInteger)index success:(void (^)(SportModel *model))suc fail:
(void (^)(void))fail;
```

3.9 获取某天的详细运动数据

```
/*!
 *  @func    获取某天的详细运动数据
 *  @discussion 每15分钟一个刻度, 每天最多会有96条数据. 详细请看返回内容
 *  @param items  sports:返回所有的运动model
 */
+ (void)getSportDetailDataByDay:(NSInteger)dayIndex sportDatas:(nullable void (^)
(NSArray<SportModel *> *sports))items fail:(nullable void (^)(void))fail;
```

3.10 获取某天指定时间段详细运动数据

```
/*!
 *  @func   获取某天指定时间段详细运动数据
 *  @param  minuteInterval  每个索引的分钟间隔
 *  @param  beginIndex      时间段开始索引
 *  @param  endIndex        时间段结束索引
 *  @param  items           sports:返回所有的运动model
 */
+ (void)getSportDetailDataByDay:(NSInteger)dayIndex minuteInterval:
(NSInteger)minuteInterval beginIndex:(NSInteger)beginIndex endIndex:(NSInteger)endIndex
sportDatas:(nullable void (^)(NSArray<SportModel *> *sports))items fail:(nullable void
(^)(void))fail;
```

3.11 获取某天的详细睡眠数据

```
//睡眠数据类型
typedef NS_ENUM(NSInteger, SLEEPTYPE) {
    SLEEPTYPENONE,    //无数据
    SLEEPTYPESOBER,   //清醒
    SLEEPTYPELIGHT,   //浅睡
    SLEEPTYPEDEEP,    //深睡
    SLEEPTYPEUNWEARED //未佩戴
};


@interface QCSleepModel : NSObject
@property (nonatomic, assign) SLEEPTYPE type;       //睡眠类型
@property (nonatomic, strong) NSString *happenDate; //发生时间 yyyy-MM-dd HH:mm:ss
@property (nonatomic, strong) NSString *endTime;    //结束时间 yyyy-MM-dd HH:mm:ss.
@property (nonatomic, assign) NSInteger total;      //开始时间与结束时间的时间间隔(单位：分钟)
@end


/*!
 *  @func   获取某天的详细睡眠数据
 *  @discussion 每种睡眠类型对应的时间段，详细请看返回内容
 *  @param items    sleeps:返回所有的睡眠model
 */
+ (void)getSleepDetailDataByDay:(NSInteger)dayIndex sleepDatas:(nullable void (^)
(NSArray<QCSleepModel *> *sleeps))items fail:(nullable void (^)(void))fail;
```

3.12 获取某天指定时间段详细睡眠数据(预留的接口)

```
/*!
 *  @func   获取某天指定时间段详细睡眠数据(预留的接口)
 *  @param  minuteInterval  每个索引的分钟间隔
 *  @param  beginIndex      时间段开始索引
 *  @param  endIndex        时间段结束索引
 *  @param  items           sports:返回所有的睡眠model
 */
+ (void)getSleepDetailDataByDay:(NSInteger)dayIndex minuteInterval:
(NSInteger)minuteInterval beginIndex:(NSInteger)beginIndex endIndex:(NSInteger)endIndex
sleepDatas:(nullable void (^)(NSArray<SleepModel *> *sleeps))items fail:(nullable void
(^)(void))fail;
```

3.13 查找戒指(戒指亮灯)

```
/**
 * 查找戒指
 */
+ (void)lookupDeviceSuccess:(void (^)(void))suc fail:(void (^)(void))fail;
```

3.14 切换到拍照界面

```
/**
 *  切换下位机(手环)到拍照界面
 */
+ (void)switchToPhotoUISuccess:(nullable void (^)(void))success fail:(nullable void (^)
(void))fail;
```

3.15 保持拍照界面

```
/**
 *  保持下位机(手环)拍照界面
 */
+ (void)holdPhotoUISuccess:(nullable void (^)(void))success fail:(nullable void (^)
(void))fail;
```

3.16 停止拍照界面

```
/**
 *  停止下位机(手环)拍照
 */
+ (void)stopTakingPhotoSuccess:(nullable void (^)(void))success fail:(nullable void (^)(void))fail;
```

3.17 硬重启戒指

```
/**
 硬重启戒指
 */
+ (void)resetBandHardlySuccess:(nullable void (^)(void))suc fail:(nullable void (^)(void))fail;
```

3.18 获取戒指Mac地址

```
/**
 *  @func 获取戒指Mac地址
 *  @param success Mac地址格式为"AA:BB:CC:DD:EE:FF"
 */
+ (void)getDeviceMacAddressSuccess:(nullable void (^)(NSString *_Nullable macAddress))success fail:(nullable void (^)(void))fail;
```

3.19 获取定时血压测量功能的信息

```
/**
 *  获取定时血压测量功能的信息
 *  @param success  featureOn       YES: 开启; NO: 关闭
 *                  beginTime       开始时间, 格式为"HH:mm"
 *                  endTime         结束时间, 格式为"HH:mm"
 *                  minuteInterval  分钟间隔
 */
+ (void)getSchedualBPInfo:(nullable void (^)(BOOL featureOn, NSString *beginTime, NSString *endTime, NSInteger minuteInterval))success fail:(void (^)(void))fail;
```

3.20 设置定时血压测量功能的信息

```
/**
 *  设置定时血压测量功能的信息
 *  @param featureOn         YES: 开启; NO: 关闭
 *  @param beginTime         开始时间, 格式为"HH:mm"
 *  @param endTime           结束时间, 格式为"HH:mm"
 *  @param minuteInterval    分钟间隔
 */
+ (void)setSchedualBPInfoOn:(BOOL)featureOn beginTime:(NSString *)beginTime endTime:
(NSString *)endTime minuteInterval:(NSInteger)minuteInterval success:(nullable void (^)
(BOOL featureOn, NSString *beginTime, NSString *endTime, NSInteger
minuteInterval))success fail:(void (^)(void))fail;
```

3.21 获取定时血压测量的历史数据

```
/**
 *  获取定时血压测量的历史数据
 *  @param userAge  :用户年龄
 *  @param success  data 心率模块数据, 目前的回复其实统一为心率, 回调中可自行处理
 */
+ (void)getSchedualBPHistoryDataWithUserAge:(NSInteger)userAge success:(nullable void (^)
(NSArray<BloodPressureModel *> *data))success fail:(nullable void (^)(void))fail;
```

3.22 重置戒指到出厂设置状态

```
/**
 *  重置戒指到出厂设置状态, 慎用
 */
+ (void)resetBandToFacotrySuccess:(nullable void (^)(void))success fail:(nullable void
(^)(void))fail;
```

3.23 获取锻炼历史数据

```
/**
 * @func 获取锻炼历史数据
 * @param lastUnixSeconds 最后一条锻炼数据的发生时间(距1970-01-01 00:00:00的秒数)
 * @note success models 锻炼数据数组
 */
+ (void)getExerciseDataWithLastUnixSeconds:(NSUInteger)lastUnixSeconds getData:(nullable
void (^)(NSArray<ExerciseModel *> *models))getData fail:(nullable void (^)(void))fail;
```

3.24 获取手动测量血压测量的历史数据

```
/**
 *  获取手动测量血压测量的历史数据
 *  @param lastUnixSeconds 最后一条手动血压数据的发生时间(距1970-01-01 00:00:00的秒数)
 *  @param success  data 血压数据数组
 */
+ (void)getManualBloodPressureDataWithLastUnixSeconds:(NSUInteger)lastUnixSeconds
success:(nullable void (^)(NSArray<BloodPressureModel *> *data))success fail:(nullable
void (^)(void))fail;
```

3.25 获取定时心率历史数据

```
/**
 * @func 获取定时心率历史数据
 * @param dates 需要获取历史数据的日期列表
 * @note success models 定时心率数据数组
 */
+ (void)getSchedualHeartRateDataWithDates:(NSArray<NSDate *> *)dates success:(nullable
void (^)(NSArray<SchedualHeartRateModel *> *models))success fail:(nullable void (^)
(void))fail;


/**
 * @func 获取定时心率历史数据
 * @param dayIndexs 需要获取历史数据的天数(0->今天，1->昨天，2->前天，依次类推)
 * @note success models 定时心率数据数组
 */
+ (void)getSchedualHeartRateDataWithDayIndexs:(NSArray<NSNumber*> *)dayIndexs success:
(void (^)(NSArray<QCSchedualHeartRateModel *> *_Nonnull))success fail:(void (^)
(void))fail;
```

3.26 获取定时心率功能的信息

```
/**
 *  获取定时心率功能的信息
 *  @param success  enable  定时心率功能是否开启. YES: 开启; NO: 关闭
 */
+ (void)getSchedualHeartRateStatusWithCurrentState:(BOOL)enable success:(nullable void
(^)(BOOL enable))success fail:(nullable void (^)(void))fail;
```

3.27 设置定时心率功能的信息

```
/**
 *  设置定时心率功能的信息
 *  @param enable   定时心率功能是否开启. YES: 开启; NO: 关闭
 */
+ (void)setSchedualHeartRateStatus:(BOOL)enable success:(nullable void (^)(BOOL
enable))success fail:(nullable void (^)(void))fail;
```

3.28 根据指定时间戳, 获取运动数据概要信息

```
/**
 根据指定时间戳, 获取该时间戳后的新版运动+(V2)数据概要信息
 @param timestamp 时间戳
 @param finished spSummary - 运动+概要信息数组
 */
+ (void)getSportPlusSummaryFromTimestamp:(NSTimeInterval)timestamp finished:(nullable
void (^)(NSArray *_Nullable spSummary, NSError *_Nullable error))finished;
```

3.29 根据指定新版运动+概要信息, 获取该次运动的部分概要信息和详细数据

```
/**
 根据指定新版运动+概要信息, 获取该次运动的部分概要信息和详细数据
 @param finished spSummary - 运动+概要信息数组
 */
+ (void)getSportPlusDetailsWithSummary:(OdmGeneralExerciseSummaryModel *)summary
finished:(nullable void (^)(OdmGeneralExerciseSummaryModel *_Nullable summary,
OdmGeneralExerciseDetailModel *_Nullable detail, NSError *_Nullable error))finished;
```

3.30 获取/设置用户目标信息

```
/**
 获取用户目标信息
 @param suc stepTarget 步数目标
            calorieTarget 卡路里目标, 单位: 卡
            distanceTarget 距离目标, 单位: 米
            sportDuration 运动时长目标 单位: 分钟(保留值, 默认: 0)
            sleepDuration 睡眠时长目标 单位: 分钟(保留值, 默认: 0)
 */
```

```
+ (void)getStepTargetInfoWithSuccess:(nullable void (^)(NSInteger stepTarget,NSInteger calorieTarget,NSInteger distanceTarget,NSInteger sportDuration,NSInteger sleepDuration))suc fail:(nullable void (^)(void))fail;

/**
 设置用户目标信息
 @param stepTarget 步数目标
 @param calorieTarget 卡路里目标, 单位: 卡
 @param distanceTarget 距离目标, 单位: 米
 @param sportDuration 运动时长目标 单位: 分钟(保留值, 默认: 0)
 @param sleepDuration 睡眠时长目标 单位: 分钟(保留值, 默认: 0)
 */
+ (void)setStepTarget:(NSInteger)stepTarget calorieTarget:(NSInteger)calorieTarget distanceTarget:(NSInteger)distanceTarget sportDurationTarget:(NSInteger)sportDuration sleepDurationTarget:(NSInteger)sleepDuration success:(nullable void (^)(void))suc fail:(nullable void (^)(void))fail;
```

3.31 获取定时体温测量的历史数据

```
/**
*  获取定时体温测量的历史数据
*/
+ (void)getSchedualTemperatureDataByDayIndex:(NSInteger)dayIndex finished:(nullable void (^)(NSArray *_Nullable temperatureList, NSError *_Nullable error))finished;
```

3.32 获取手动体温测量的历史数据

```
/**
 *  获取手动体温测量的历史数据
 */
+ (void)getManualTemperatureDataByDayIndex:(NSInteger)dayIndex finished:(nullable void (^)(NSArray *_Nullable temperatureList, NSError *_Nullable error))finished;
```

3.33 获取血氧测量的历史数据

```
/**
 *  获取血氧测量的历史数据
 */
+ (void)getBloodOxygenDataByDayIndex:(NSInteger)dayIndex finished:(void (^)(NSArray *_Nullable, NSError * _Nullable))finished;
```

3.34 发送固件文件

```
/**
 发送固件文件，要求使用bin文件升级，结果会在回调里边处理

 @param data             OTA二进制字符流
 @param start            开始发送回调
 @param percentage       进度回调
 @param success          成功回调
 @param failed           失败回调
 */

+ (void)syncOtaBinData:(NSData *)data
                 start:(nullable void (^)(void))start
            percentage:(nullable void (^)(int percentage))percentage
               success:(nullable void (^)(int seconds))success
                failed:(nullable void (^)(NSError *error))failed;
```

3.35 收到手环消息

```
@interface QCSDKManager : NSObject

/*
 *  收到手表，查找手机的通知
 */
@property(nonatomic,copy)void(^findPhone)(void);

/*
 *  收到手表，进入相机的通知
 */
@property(nonatomic,copy)void(^switchToPicture)(void);

/*
 *  收到手表，拍照的通知的回调
 */
@property(nonatomic,copy)void(^takePicture)(void);

/*
 *  收到手表，结束拍照的通知的回调
 */
@property(nonatomic,copy)void(^stopTakePicture)(void);

// 单例类实例
+ (instancetype)shareInstance;

@end
```

3.36 设置/获取定时血氧开关状态

```
/**
 *  设置定时血氧测量功能的信息
 *  @param featureOn        YES: 开启; NO: 关闭
 */

+ (void)setSchedualBOInfoOn:(BOOL)featureOn success:(nullable void (^)(BOOL
featureOn))success fail:(void (^)(void))fail;

/**
 *  获取定时血氧测量功能的信息
 *  @param success featureOn YES: 开启; NO: 关闭
 */

+ (void)getSchedualBOInfoSuccess:(nullable void (^)(BOOL featureOn))success fail:(void
(^)(void))fail;
```

3.37 发送测量指令 (指令封装在QCSDKManager中)

```
typedef NS_ENUM(NSInteger, QCMeasuringType) {
    QCMeasuringTypeHeartRate = 0,   //Heart rate measurement
    QCMeasuringTypeBloodPressue,    //blood pressure measurement
    QCMeasuringTypeBloodOxygen,     //blood oxygen measurement
    QCMeasuringTypeOneKeyMeasure,   //One-click measurement
    QCMeasuringTypeStress,
    QCMeasuringTypeBloodGlucose,
    QCMeasuringTypeCount,
};

//测量结果为hanle回调中的result
//测量心率的时候,result返回的是NSNumber: @(60)
//测量血压的时候,result返回的是NSDictionary:@{@"sbp":@"120",@"dbp":@"60"}
//测量血氧的时候,result返回的是NSNumber: @(98)


/// Send measurement order
/// @param type                     :Measurement type
/// @param measuring          :Real-Time Measuring Value
/// @param handle                 :Measurement result callback (error code: -1: failed to
send start command, -2: failed to send end command, -3: bracelet is not properly worn)
- (void)startToMeasuringWithOperateType:(QCMeasuringType)type measuringHandle:(void(^)(id
_Nullable result))measuring completedHandle:(void(^)(BOOL isSuccess,id _Nullable
result,NSError * _Nullable error))handle;

/// Send measurement order
/// @param type                     :Measurement type
```

```
/// @param measuring          :Real-Time Measuring Value
/// @param handle                 :Measurement result callback (error code: -1: failed to send start command, -2: failed to send end command, -3: bracelet is not properly worn)
- (void)startToMeasuringWithOperateType:(QCMeasuringType)type timeout:(NSInteger)timeout measuringHandle:(void(^)(id _Nullable result))measuring completedHandle:(void(^)(BOOL isSuccess,id _Nullable result,NSError * _Nullable error))handle;

/// stop measurement command
/// @param type             :Measurement type
/// @param handle        :Measurement result callback (error code:-1: Failed to send end command)
- (void)stopToMeasuringWithOperateType:(QCMeasuringType)type completedHandle:(void(^)(BOOL isSuccess,NSError *error))handle;
```

3.38 睡眠协议(获取某一天到今天)

```
/*!
 *  @func   获取从某天到今天的所有睡眠数据
 *  @param  fromDayIndex  距离今天的天数，(0:表示今天，1：表示昨天)
 *  @param  items      返回的睡眠数据(key：距离今天的天数，value：对应的睡眠数据)
 *  @param  fail        失败的 回调
 */
+ (void)getSleepDetailDataFromDay:(NSInteger)fromDayIndex sleepDatas:(nullable void (^)(NSDictionary <NSString*,NSArray<QCSleepModel*>*>*_Nonnull))items fail:(nullable void (^)(void))fail;
```

3.39 实时心率测量

```
typedef enum {
    QCBandRealTimeHeartRateCmdTypeStart = 0x01,//Start real-time heart rate measurement
    QCBandRealTimeHeartRateCmdTypeEnd,//End real-time heart rate measurement
    QCBandRealTimeHeartRateCmdTypeHold,//Continuous heart rate test (for continuous measurement to keep alive)
} QCBandRealTimeHeartRateCmdType;


/**
 *  RealTime HeartRate Measuring
 *   实时心率测量
 *
 *  @param type         :commond type
 *  @param finished     :finish callback
 */
+ (void)realTimeHeartRateWithCmd:(QCBandRealTimeHeartRateCmdType)type finished:(nullable void (^)(BOOL))finished;
```

3.40 发起运动

```
/// Set Sport Mode State
///
/// - Parameters:
///   - sportType: type
///   - state: state
///   - finished: finished callback
+ (void)operateSportModeWithType:(OdmSportPlusExerciseModelType)sportType state:
(QCSportState)state finish:(void(^)(id _Nullable,NSError * _Nullable))finished;
```

获取回调：

```
    [QCSDKManager shareInstance].currentSportInfo = ^(QCSportInfoModel * _Nonnull
sportInfo) {

 NSLog(@"sportType:%zd,duration:%zd,state:%u,hr:%zd,step:%zd,calorie(unit:calorie):%zd,di
stance(unit:meter):%zd",sportInfo.sportType,sportInfo.duration,sportInfo.state,sportInfo.
hr,sportInfo.step,sportInfo.calorie,sportInfo.distance);
    };
```

3.41 获取压力数据

```
/// Get Schedual Stress Datas (Only Ring Support)
///
/// - Parameters:
/// - dates: 0-6,0:today,1:yesterday....
/// - finished: finished callback
+ (void)getSchedualStressDataWithDates:(NSArray<NSNumber*> *)dates finished:(void (^)
(NSArray * _Nullable, NSError * _Nullable))finished;;


/// Get Schedual Stress Status
///
/// - Parameter finished: finished callback
+ (void)getSchedualStressStatusWithFinshed:(nullable void (^)(BOOL,NSError *_Nullable
error))finished;


/// Set Schedual Stress Status
///
/// - Parameters:
///   - enable:YES:On,NO:Off
///   - finished: finished callback
+ (void)setSchedualStressStatus:(BOOL)enable finshed:(nullable void (^)(NSError
*_Nullable error))finished;
```