Development Guide

1 Environment configuration

1.1 Add framework

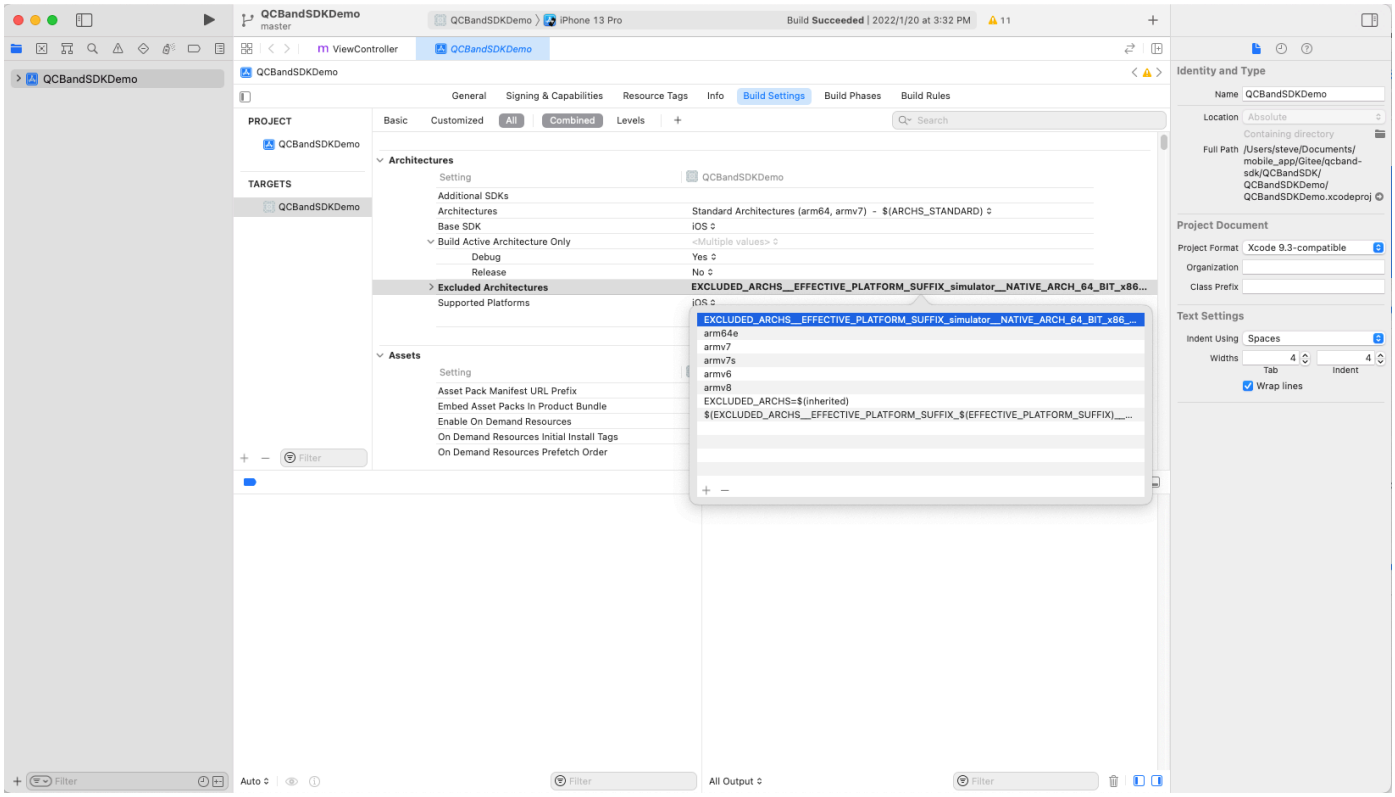
Add `QCBandSDk.framework` to the project, the framework supports iOS 9.0 and above

Note: Because the classification is used in the framework, you need to add settings to the project

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

Add the following code in Target-Build Settings-Excluded Architectures

```
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 Configure Bluetooth permissions

Configure bluetooth permissions in info.plist file

```
<key>NSBluetoothAlwaysUsageDescription</key>
<string>App needs to use your bluetooth device</string>
<key>NSBluetoothPeripheralUsageDescription</key>
<string>App needs to use your bluetooth device</string>
```

2 Usage

2.1 Service UUIDs supported by the device

Defined in `QCSDKManager.h`, the service UUID supported by the device:

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

2.2 import framework library

Introduce the framework library into the code

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

Initialize `[QCSDKManager shareInstance]` with a singleton

> `QCSDKManager`:Peripherals for joining connections
>
> `QCSDKCmdCreator`:Used to send commands to peripherals

2.3 Scan Ring

initialization

Scanning can only be started when permissions are allowed and Bluetooth is turned on.

Import Apple's CoreBluetooth library and follow two protocols `<CBCentralManagerDelegate, CBPeripheralDelegate>`

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

Declare central and peripheral roles

```
/*Central Role,app*/
@property (strong, nonatomic) CBCentralManager *centerManager;

/*Peripheral role, scanned peripherals*/
@property (strong, nonatomic) NSMutableArray<CBPeripheral *> *peripherals;

/*Connected peripheral role*/
@property (strong, nonatomic) CBPeripheral *connectedPeripheral;
```

Instantiate the central role

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

Scan Ring

Using Scan Peripherals

```
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];
```

Note: To obtain the scanned peripheral devices in the agent, you can perform secondary filtering through the device name and other related information.

```
- (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 Cancel the scan of the Ring

Call the interface of the central role to stop scanning

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

2.5 Connect the Ring

start connecting

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

After the connection is successful, pass in the peripheral device to the SDK

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

2.6 Disconnect

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

After disconnecting, remove peripherals

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

3. Instructions supported by the device

3.1 Set Ring time

```
/**
 Set the time of the Ring
 */
+ (void)setTime:(NSDate *)date success:(void (^)(NSDictionary *))suc failed:(void (^)
(void))fail;
```

3.2 Read Ring battery

```
/*!
 *  @func Read Ring battery
 *  @param suc battery:Power level(0~8)
 */
+ (void)readBatterySuccess:(void (^)(int battery))suc failed:(void (^)(void))fail;
```

3.3 Bound Vibration

```
/**
 *  Bound Vibration
 */
+ (void)alertBindingSuccess:(nullable void (^)(void))suc fail:(nullable void (^)
(void))fail;
```

3.4 Set wristband time base/user personal information

```
/**
 Set wristband time base/user personal information

 @param twentyfourHourFormat    : YES 24 hour system; NO 12 hour system
 @param metricSystem            : YES Metric; NO Imperial
 @param gender                  : gender (0=male, 1=female)
 @param age                     : age (years)
 @param height                  : height (cm)
 @param weight                  : weight (kg)
 @param sbpBase                 : systolic blood pressure base (mmhg) (reserved value,
Defaults:0)
 @param dbpBase                 : Diastolic blood pressure base (mmhg) (reserved value,
Defaults:0)
 @param hrAlarmValue            : Heart rate alarm value (bpm) (reserved value,
Defaults: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 Get Ring time base/user personal information

```
/**
 Get Ring time base/user personal information

 @param success isTwentyfour: YES 24 hour system; NO 12 hour system
                isMetricSystem: YES Metric; NO Imperial
                gender: gender (0=male, 1=female)
                age: age (years)
                height: height (cm)
                weight: weight (kg)
                sbpBase: systolic blood pressure base (mmhg) (reserved value, Defaults:0)
                dbpBase: Diastolic blood pressure base (mmhg) (reserved value, Defaults:0)
                hrAlarmValue: Heart rate alarm value (bpm) (reserved value, Defaults: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 Get the version number of the Ring firmware

```
/**
 *  @func Get the version number of the Ring firmware
 *
 *  @param success The format of software and hardware version numbers is
generally"x.x.x"
 */
+ (void)getDeviceSoftAndHardVersionSuccess:(void (^)(NSString *_Nonnull, NSString
*_Nonnull))success fail:(void (^)(void))fail;
```

3.7 Get current steps

```
/*!
 *  @func   Get current steps
 */
+ (void)getCurrentSportSucess:(void (^)(SportModel *sport))suc failed:(void (^)
(void))fail;
```

3.8 Get total statistics for a day(Steps、Calories、Distance、Time)

```
/*!
 *  @func   Get total statistics for a day
 *
 *  @param   index  : 0->Today 1->1 day ago.Maximum 29
 *  @param   suc    :Using this command cannot accurately obtain the statistical data of the day, the device will save the data every 15 minutes, so there will be a 15-minute interval
 */
+ (void)getOneDaySportBy:(NSInteger)index success:(void (^)(SportModel *model))suc fail:(void (^)(void))fail;
```

3.9 Get detailed exercise data for a day

```
/*!
 *  @func   Get detailed exercise data for a day
 *  @discussion There is a tick every 15 minutes, and there will be a maximum of 96 pieces of data per day. For details, please see the returned content
 *  @param items  sports:return all sports models
 */
+ (void)getSportDetailDataByDay:(NSInteger)dayIndex sportDatas:(nullable void (^)(NSArray<SportModel *> *sports))items fail:(nullable void (^)(void))fail;
```

3.10 Get detailed exercise data for a specified time period on a certain day

```
/*!
 *  @func   Get detailed exercise data for a specified time period on a certain day
 *  @param  minuteInterval  minute interval for each index
 *  @param  beginIndex      time period start index
 *  @param  endIndex        time period end index
 *  @param  items           sports:return all sports models
 */
+ (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 Get detailed sleep data for a day

```
//sleep data type
typedef NS_ENUM(NSInteger, SLEEPTYPE) {
    SLEEPTYPENONE,    //no data
    SLEEPTYPESOBER,   //wide awake
    SLEEPTYPELIGHT,   //light sleep
    SLEEPTYPEDEEP,    //Deep sleep
    SLEEPTYPEUNWEARED //not worn
};


@interface QCSleepModel : NSObject
@property (nonatomic, assign) SLEEPTYPE type;       //sleep type
@property (nonatomic, strong) NSString *happenDate; //Start Time yyyy-MM-dd HH:mm:ss
@property (nonatomic, strong) NSString *endTime;    //End Time yyyy-MM-dd HH:mm:ss.
@property (nonatomic, assign) NSInteger total;      //Time interval between start time
and end time (unit: minutes)
@end


/*!
 *  @func  Get detailed sleep data for a day
 *  @discussion The time period corresponding to each sleep type, please see the returned
content for details
 *  @param items    sleeps:return all sleep models
 */
+ (void)getSleepDetailDataByDay:(NSInteger)dayIndex sleepDatas:(nullable void (^)
(NSArray<QCSleepModel *> *sleeps))items fail:(nullable void (^)(void))fail;
```

3.12 Find Ring

```
/**
 * Find Ring
 */
+ (void)lookupDeviceSuccess:(void (^)(void))suc fail:(void (^)(void))fail;
```

3.13 Ring to the camera interface

```
/**
 *  Switch the lower computer to the camera interface
 */
+ (void)switchToPhotoUISuccess:(nullable void (^)(void))success fail:(nullable void (^)
(void))fail;
```

3.14 keep the camera interface

```
/**
 *  Keep the camera interface of the lower computer
 */
+ (void)holdPhotoUISuccess:(nullable void (^)(void))success fail:(nullable void (^)
(void))fail;
```

3.15 Stop taking pictures

```
/**
 *  Stop the lower computer to take pictures
 */
+ (void)stopTakingPhotoSuccess:(nullable void (^)(void))success fail:(nullable void (^)
(void))fail;
```

3.16 Restart the Ring

```
/**
 Restart the Ring
 */
+ (void)resetBandHardlySuccess:(nullable void (^)(void))suc fail:(nullable void (^)
(void))fail;
```

3.17 Get Ring Mac address

```
/**
 *  @func Get Ring Mac address
 *  @param success The Mac address format is "AA:BB:CC:DD:EE:FF"
 */
+ (void)getDeviceMacAddressSuccess:(nullable void (^)(NSString *_Nullable
macAddress))success fail:(nullable void (^)(void))fail;
```

3.18 Get information about the timed blood pressure measurement function

```
/**
 *  Get information about the timed blood pressure measurement function
 *  @param success  featureOn       YES: ON; NO: OFF
 *                  beginTime       Start time, format "HH:mm"
 *                  endTime         End time, the format is "HH:mm"
 *                  minuteInterval  minute interval (minutes)
 */
+ (void)getSchedualBPInfo:(nullable void (^)(BOOL featureOn, NSString *beginTime,
NSString *endTime, NSInteger minuteInterval))success fail:(void (^)(void))fail;
```

3.19 Information on setting the timed blood pressure measurement function

```
/**
 *  Information on setting the timed blood pressure measurement function
 *  @param featureOn        YES: ON; NO: OFF
 *  @param beginTime        Start time, format "HH:mm"
 *  @param endTime          End time, the format is "HH:mm"
 *  @param minuteInterval   minute interval (minutes)
 */
+ (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;
```

3.20 Obtain historical data for timed blood pressure measurements

```
/**
 *  Obtain historical data for timed blood pressure measurements
 *  @param userAge  :User age
 *  @param success  data: Heart rate module data, the current reply is actually unified
as heart rate, which can be processed by itself in the callback
 */
+ (void)getSchedualBPHistoryDataWithUserAge:(NSInteger)userAge success:(nullable void (^)
(NSArray<BloodPressureModel *> *data))success fail:(nullable void (^)(void))fail;
```

3.21 Reset the Ring to factory settings

```
/**
 *  Reset the Ring to factory settings
 */
+ (void)resetBandToFacotrySuccess:(nullable void (^)(void))success fail:(nullable void
(^)(void))fail;
```

3.22 Get historical data of exercise records

```
/**
 * @func Get historical data of exercise records
 * @param lastUnixSeconds The time when the last exercise data occurred (seconds since 1970-01-01 00:00:00)
 * @note success models Motion data array
 */
+ (void)getExerciseDataWithLastUnixSeconds:(NSUInteger)lastUnixSeconds getData:(nullable void (^)(NSArray<ExerciseModel *> *models))getData fail:(nullable void (^)(void))fail;
```

3.23 Get historical data for manual blood pressure measurements

```
/**
 *  Get historical data for manual blood pressure measurements
 *  @param lastUnixSeconds Time when the last manual blood pressure data occurred (seconds since 1970-01-01 00:00:00)
 *  @param success  data blood pressure data array
 */
+ (void)getManualBloodPressureDataWithLastUnixSeconds:(NSUInteger)lastUnixSeconds success:(nullable void (^)(NSArray<BloodPressureModel *> *data))success fail:(nullable void (^)(void))fail;
```

3.24 Get timed heart rate historical data

```
/**
 * @func Get timed heart rate historical data
 * @param dates: List of dates for which historical data needs to be obtained
 * @note success models Timed heart rate data array
 */
+ (void)getSchedualHeartRateDataWithDates:(NSArray<NSDate *> *)dates success:(nullable void (^)(NSArray<SchedualHeartRateModel *> *models))success fail:(nullable void (^)(void))fail;


/**
 * @func Get timed heart rate historical data
 * @param dayIndexs The number of days for which historical data needs to be obtained (0->today, 1->yesterday, 2->the day before yesterday, and so on)
 * @note success models Timed heart rate data array
 */
```

```
+ (void)getSchedualHeartRateDataWithDayIndexs:(NSArray<NSNumber*> *)dayIndexs success:
(void (^)(NSArray<QCSchedualHeartRateModel *> *_Nonnull))success fail:(void (^)
(void))fail;
```

3.25 Get information about the timed heart rate function

```
/**
 *  Get information about the timed heart rate function
 *  @param success  enable  Whether the timed heart rate function is enabled. YES: ON;
NO: OFF
 */
+ (void)getSchedualHeartRateStatusWithCurrentState:(BOOL)enable success:(nullable void
(^)(BOOL enable))success fail:(nullable void (^)(void))fail;
```

3.26 Information on setting the timed heart rate function

```
/**
 *  Information on setting the timed heart rate function
 *  @param enable   Whether the timed heart rate function is enabled. YES: ON; NO: OFF
 */
+ (void)setSchedualHeartRateStatus:(BOOL)enable success:(nullable void (^)(BOOL
enable))success fail:(nullable void (^)(void))fail;
```

3.27 According to the specified time stamp, the new version of Sports+ (V2) data summary information

```
/**
 According to the specified time stamp, the new version of Sports+ (V2) data summary
information
 @param timestamp
 @param finished spSummary - Motion+Summary info array
 */
+ (void)getSportPlusSummaryFromTimestamp:(NSTimeInterval)timestamp finished:(nullable
void (^)(NSArray *_Nullable spSummary, NSError *_Nullable error))finished;
```

3.28 According to the specified new version of the campaign + summary information, get some summary information and detailed data of the campaign

```
/**
 According to the specified new version of the campaign + summary information, get some summary information and detailed data of the campaign
 @param finished spSummary - Motion+Summary info array
 */
+ (void)getSportPlusDetailsWithSummary:(OdmGeneralExerciseSummaryModel *)summary finished:(nullable void (^)(OdmGeneralExerciseSummaryModel *_Nullable summary, OdmGeneralExerciseDetailModel *_Nullable detail, NSError *_Nullable error))finished;
```

3.29 Get/set user target information

```
/**
 Get/set user target information

 * @param suc stepTarget:     Step target
           calorieTarget:     Calorie Goal, Unit: Calories
          distanceTarget:     Distance to target, unit: meters
           sportDuration:     Exercise duration target Unit: minutes (reserved value, default: 0)
            sleepDuration:    Sleep duration target unit: minutes (reserved value, default: 0)
 */
+ (void)getStepTargetInfoWithSuccess:(nullable void (^)(NSInteger stepTarget,NSInteger calorieTarget,NSInteger distanceTarget,NSInteger sportDuration,NSInteger sleepDuration))suc fail:(nullable void (^)(void))fail;

/**
 Set user target information

 * @param stepTarget:       Step target
 * @param calorieTarget:    Calorie Goal, Unit: Calories
 * @param distanceTarget:   Distance to target, unit: meters
 * @param sportDuration:    Exercise duration target Unit: minutes (reserved value, default: 0)
 * @param sleepDuration:    Sleep duration target unit: minutes (reserved value, default: 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;
```

3.30 Obtain historical data of timed body temperature measurement

```
/**
*  Obtain historical data of timed body temperature measurement
*/
+ (void)getSchedualTemperatureDataByDayIndex:(NSInteger)dayIndex finished:(nullable void
(^)(NSArray *_Nullable temperatureList, NSError *_Nullable error))finished;
```

3.31 Get historical data for manual body temperature measurements

```
/**
 *  Get historical data for manual body temperature measurements
 */
+ (void)getManualTemperatureDataByDayIndex:(NSInteger)dayIndex finished:(nullable void
(^)(NSArray *_Nullable temperatureList, NSError *_Nullable error))finished;
```

3.32 Get historical data for blood oxygen measurements

```
/**
 *  Get historical data for blood oxygen measurements
 */
+ (void)getBloodOxygenDataByDayIndex:(NSInteger)dayIndex finished:(void (^)(NSArray *
_Nullable, NSError * _Nullable))finished;
```

3.33 Send firmware file

```
/**
 Send the firmware file and request to use the bin file to upgrade, the result will be
processed in the callback

 @param data            OTA binary character stream
 @param start           start sending callback
 @param percentage      progress callback
 @param success         success callback
 @param failed          failure callback
 */

+ (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;
```

3.34 Receive a Ring message

```
@interface QCSDKManager : NSObject

/*
 *  Receive notifications from Ring, find phone
 */
@property(nonatomic,copy)void(^findPhone)(void);

/*
 *  Receive notifications from Ring, enter camera
 */
@property(nonatomic,copy)void(^switchToPicture)(void);

/*
 *  Receive notification of Ring, take photo
 */
@property(nonatomic,copy)void(^takePicture)(void);

/*
 *  Receive a notification from the Ring to end taking pictures
 */
@property(nonatomic,copy)void(^stopTakePicture)(void);

// singleton class instance
+ (instancetype)shareInstance;

@end
```

3.35 Set/get timed blood oxygen switch status

```
/**
 *  Information on setting the timed oximetry function
 *  @param featureOn        YES: ON; NO: OFF
 */

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

/**
 *  Get information about the timed oximetry function
 *  @param success featureOn YES: 开启; NO: 关闭
 */

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

3.36 Send measurement commands (commands are encapsulated in QCSDKManager)

```
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,
};

//The measurement result is the result in the hanle callback
//When measuring heart rate, result returns NSNumber: @(60)
//When measuring blood pressure, the result returned is
NSDictionary:@{@"sbp":@"120",@"dbp":@"60"}
//When measuring blood oxygen, the result returns 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;
```

3.37 Sleep protocol (get a day to today)

```
/*!
 *  @func   Get all sleep data from a certain day to today
 *  @param  fromDayIndex  The number of days from today, (0: means today, 1: means
yesterday)
 *  @param  items      Returned sleep data (key: days from today, value: corresponding
sleep data)
 *  @param  fail        failed callback
 */
+ (void)getSleepDetailDataFromDay:(NSInteger)fromDayIndex sleepDatas:(nullable void (^)
(NSDictionary <NSString*,NSArray<QCSleepModel*>*>*_Nonnull))items fail:(nullable void (^)
(void))fail;
```

3.38 RealTime HeartRate Measuring

```
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;
```

3.39 Set Sport Mode State（Only Ring support）

```
/// 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;
```

Get callback：

```
    [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);
    };
```

3.40 Get Schedual Stress Datas（Only Ring support）

```
/// 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;
```