ChipletRing APP SDK manual IOS

Document version number:	1.0.5	Document No. :	
Document secret level:		Department:	Chiplet Research and Development Department
Product Name:	BCL_603M	Attribution Items:	
Writer:	Zheng Yuhu	Date written:	2023.11.2

Wuxi Yongxin technology Co., Ltd.

Revision record:

Version Number	Revision person	Date of revision	Revised Description
1.0.0	Zheng Yuhu	2023.11.3	Update the directory by fixing typos
1.0.2	Zheng Yuhu	2023.11.23	Summary of resolved issues
1.0.3	Zheng Yuhu	2023.11.29	Q&A added
1.0.4	Zheng Yuhu	2023.12.20	Added Troubleshooting
1.0.5	Zheng Yuhu	2024.07.09	Added sleep logic flowchart

Catalogue

I、Document Brief

1、 Document purpose

In order to facilitate the secondary development of communication between IOS APP and ring, the communication protocol is specially encapsulated to achieve simplicity and clarity, so that developers do not need to pay attention to the communication layer with ring, and focus on the development of business logic interaction level.

2、 Scope of application

Supported iOS version: iOS 13.0+
Supported language version: Swift 5.0+
Developer tools: xcode 15+

3、 Introduction Note

This is the Bluetooth communication SDK of smart ring, which mainly provides the API management class RingManager related to ring communication, and the related data storage management class RingDBManager, and also provides the realization of some complex algorithms, such as sleep-related algorithms.

Note: Since the sdk requires Bluetooth, only real machine debugging is supported, and be careful to add Bluetooth permission description in your project info

4、 Function Description

Function module	Instructions	Relevant documentation
Bluetooth basic module	● Switch operation of Bluetooth ● Bluetooth search link operation ● Bluetooth data write listening operation	

Communication protocol module	• Time management • Version number management • Battery Management • Heart rate measurement • Blood oxygen measurement • Temperature measurement • Step management • History management • System Settings • Log management	

II、 Quick use

1、 Use CocoaPods

Start by adding a new Spec source to your project's Podfile file by copying and pasting the following statement to the top of your project's Podfile:

```
1. source 'https://github.com/wcb133/RingSpec.git'
```

2、 Import SDK

Import this SDK in the pod:

```
1. pod 'Rings-SDK'
```

Then install:

```
1. pod install
```

Just introduce the SDK where it is used:

```
1. import RingsSDK
```

III、 API Function Description

1、 Device connection related

1.1 Search for Bluetooth

Interface description: search nearby bluetooth devices, after began to search through the callback to obtain a list of search to equipment, can also be through RingManager. Shared. The devices for current search attribute to the equipment

Interface declaration:

```
1. RingManager.shared.startScan { devices in
2. Print (" search to the equipment list = = = = = = = = = > \ (String
   (describing: devices)) ")
3. }
```

What to watch for: None

1.2 Stop searching

Interface description: stop search nearby bluetooth devices, to stop search not empty has searched the equipment list, is not empty RingManager. Shared. Devices

Interface statement:

```
1. RingManager.shared.stopScan()
```

Note: When invoking this interface, ensure that it is connected to the ring

1.3 Connect the specified Bluetooth

Interface description: Connect the specified device by the device uuid, uuid obtained from the device model DeviceInfo

Interface declaration:

```
1. RingManager.shared.startConnect(deviceUUID: "uuidString", resultBlock: { res
   in
2.       switch res {
3.       case .success(let deviceInfo):
4. Print (" connected equipment = = = = = = = = = \ (String (describing: deviceInfo
   peripheralName)) ")
5.       case .failure(let error):
6. print(" Connection failed ========> \(error)")
7. }
8. })
9.
```

What to watch for: None

1.4 Disconnect Bluetooth

Interface description: Disconnect the current device

Interface declaration:

```
1. RingManager.shared.disconnect()
```

Note: When invoking this interface, ensure that it is connected to the ring

1.5 Currently connected to Bluetooth

Interface Description: Get the currently connected device

Interface declaration:

```
1. let currentDevice = RingManager.shared.currentDevice
```

Note: When invoking this interface, ensure that it is connected to the ring

1.6 Bluetooth connection status

Interface description: Get device connection status

Interface declaration:

```
1. let isDidConnect = RingManager.shared.isDidConnect
```

Note: None

1.7 Listen for connection status

Interface description: Device connection status change monitoring

Interface declaration:

```
1. RingManager.shared.connectStateChangeBlock = { isConnected in
2. print(" Is it connected ========\(isConnected)")
3.
4. }
```

What to watch for: None

2、 Instruction function

2.1 Synchronizing time

Interface description:

Interface declaration:

```
1. RingManager.shared.syncTime(date: Date()) { res in
2.          switch res {
3.          case .success(let isSuccess):
```

```
4. Print (" synchronization time results = = = = = = \ (isSuccess) ")
5.          case .failure(let error):
6. print(" Sync failed ======\(error)")
7. }
8. }
```

Note: When calling this interface, ensure that it is connected to the ring

2.2 Read time

Interface description:

Interface declaration:

```
1. RingManager.shared.readTime { res in
2.         switch res {
3.         case .success(let value):
4. print(" Success =====>\(value) milliseconds ")
5.         case .failure(let error):
6. print(" fail =====>\(error)")
7. }
8. }
```

Note: When calling this interface, ensure that it is connected to the ring

2.3 Read the software version number

Interface description:

Interface declaration:

```
1. RingManager.shared.readAppVersion { res in
2.          switch res {
3.          case .success(let version):
4. Print (" software version number = = = = = > \ (version) ")
5.          case .failure(let failure):
6. print(" fail =====>\(failure)")
7. }
8. }
```

Note: When calling this interface, ensure that it is connected to the ring

2.4 Read the hardware version number

Interface description:

Interface declaration:

```
RingManager.shared.readHardWareVersion { res in
        switch res {
        case .success(let version):
Print (" the hardware version number = = = = = > \ (version) ")
        case .failure(let failure):
print(" fail =====>\(failure)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.5 Read the battery level

Interface description: If the battery level returned is 101, it means that it is being charged

Interface statement:

```
RingManager.shared.readBattery { res in
        switch res {
        case .success(let value):
print(" Electricity =====>\(value)")
        case .failure(let failure):
print(" fail =====>\(failure)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.6 Read the battery charging status

Interface description: Read the battery charging status, return the result as ChargeStatus enumeration type, the value is:

- full: ChargeStatus
- charging: Being charged
- normal: Normal uncharged state

Interface declaration:

```
RingManager.shared.readChargeStatus { res in
        switch res {
        case .success(let state):
print(" status =====>\(state)")
        case .failure(let error):
print(" fail =====>\(error)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.7 Measure the heart rate in real time

Interface description: Output real-time heart rate measurement value, BPM.Only when isOpenWave is set to true, the waveform data will be read, and it will be called back through the tableBlock closure.

Interface declaration:

```
RingManager.shared.readHeartRate(progressBlock: { progress in
print(" progress =====>\(progress)")
      }, isOpenWave: true) { seq, num, datas in
print(" serial number ====>\(seq)")
print(" Number of data ====>\(num)")
print(" Waveform data ====>\(datas)")
      } resultBlock: { res in
        switch res {
        case .success(let success):
print(" success =====>\(success)")
        case .failure(let failure):
print(" fail =====>\(failure)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.8 Heart rate variability

Interface description: Output heart rate variability in milliseconds (ms).The waveform data is read only if isOpenWave is set to true, and it is called back through the tableBlock closure.

Interface declaration:

```
RingManager.shared.readHRV(progressBlock: { progress in
BDLogger. Info (" progress = = = = = > \ (progress) ")
      }, isOpenWave: true) { seq, num, datas in
BDLogger.info(" serial number ====>\(seq)")
BDLogger.info(" Number of data ====>\(num)")
BDLogger. Info (" waveform data = = = = > \ (datas) ")
      } resultBlock: { res in
        switch res {
        case .success(let success):
BDLogger. Info (" success = = = = = > \ (success) ")
        case .failure(let failure):
BDLogger. Info (" = = = = = > failure \ (failure) ")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.9 Measure blood oxygen value in real time

Interface description: Output real-time blood oxygen value.Only when isOpenWave is set to true, the waveform data is read, and it is called back through the tableBlock closure.

Interface declaration:

```
RingManager.shared.readO2(progressBlock: { progress in
BDLogger. Info (" progress = = = = = > \ (progress) ")
      }, isOpenWave: true) { seq, num, datas in
BDLogger.info(" serial number ====>\(seq)")
BDLogger.info(" Number of data ====>\(num)")
BDLogger. Info (" waveform data = = = = > \ (datas) ")
      } resultBlock: { res in
        switch res {
        case .success(let success):
BDLogger. Info (" success = = = = = > \ (success) ")
        case .failure(let failure):
BDLogger. Info (" = = = = = > failure \ (failure) ")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.10 Read the temperature

Interface description: Read the current temperature, return the result in Celsius (° C)

Interface declaration:

```
RingManager.shared.readTemperature { res in
        switch res {
        case .success(let value):
BDLogger. Info (" = = = = = > \ temperature (value) ")
        case .failure(let error):
BDLogger. Info (" = = = = = > failure \ (error) ")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.11 Read the number of real-time steps on the day

Interface description:

Interface declaration:

```
RingManager.shared.readSteps { res in
        switch res {
        case .success(let value):
print(" Number of steps =====>\(value) steps ")
        case .failure(let error):
print(" fail =====>\(error)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.12 Clear the number of live steps

Interface description:

Interface declaration:

```
RingManager.shared.clearSteps { res in
        switch res {
        case .success(let isSuccess):
print(" Result =====>\(isSuccess)")
        case .failure(let error):
print(" fail =====>\(error)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.13 Read local data

Interface description: Read the local historical data, and the internal will be each data stored in the database.For how to get the data from the database, see the RingDBManager section of the database management class

Interface declaration:

```
RingManager.shared.readDatas { progress, dataModel in
Print (" progress = = = = = > \ (progress) = = \ [dataModel) ")
      } resultBlock: { res in
        switch res {
        case .success(let state):
print(" Result =====>\(state)")
        case .failure(let error):
print(" fail =====>\(error)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

This API obtains the device history data. Each history information obtained is called back by the progressBlock closure. After obtaining all the results, the final result is called back by the

resultBlock.dataModel is the device data model, and the related properties are as follows:

```
public final class RingDataModel: NSObject,TableCodable {
// Total number
  public var total:UInt32 = 0
// Serial number, starting from 1
  public var serialNum:UInt32 = 0
// Timestamp, seconds
  public var timestamp:UInt32 = 0
// Number of accumulated steps by the end of the current day
  public var stepsOfTheDay:UInt16 = 0
// Heart rate,0 is invalid
  public var rate = 0
// Blood oxygen,0 does not work
  public var O2 = 0
// Heart rate variability,0 is not valid
  public var hrv = 0
// Mental stress index,0 is not valid
  public var mentalStress = 0
// Temperature, signed
  public var temp:Float = 0
// Motion intensity 0: stationary 0x65: motion
  public var isRunning = false
// Sleep type 0: ineffective 1: awake 2: light sleep 3: deep sleep 4.
   Eye movement Period
  public var sleepType = 0
// RR Number of periods
  public var RRNums = 0
// RR array
  public var rrs:[Int] = []
}
```

Returned value: state is of the ReadDataResult enumeration type with the values:

```
public enum ReadDataResult {
case empty // No data
case complete // Finished
}
```

2.14 Delete data

Interface Description: Delete all local data history

Interface statement:

```
RingManager.shared.clearRingData { res in
        switch res {
        case .success(let isSuccess):
print(" Result =====>\(isSuccess)")
        case .failure(let failure):
print(" fail =====>\(failure)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.15 Set the collection period

Interface description: collection cycle setting, collection cycle setting unit is seconds (s)
Interface declaration:

```
RingManager.shared.setFrequency(time: time) { res in
          switch res {
          case .success(let isSuccess):
Print (" results = = = = = > \ (isSuccess) ")
          case .failure(let failure):
Print (" = = = = = > failure \ (failure) ")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.16 Read acquisition cycle

Interface description: acquisition cycle read, read the acquisition cycle of seconds (s)
Interface declaration:

```
RingManager.shared.readFrequency { res in
        switch res {
        case .success(let value):
print(" success =====>\(value) seconds ")
        case .failure(let error):
print(" fail =====>\(error)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

2.17 factory data reset

Interface description:

Interface declaration:

```
RingManager.shared.reset { res in
        switch res {
        case .success(let isSuccess):
print(" success =====>\(isSuccess)")
        case .failure(let error):
print(" fail =====>\(error)")
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

3、 Firmware Upgrade (OTA)

Interface description: For firmware upgrade, fileUrl indicates the local path where the firmware file resides. The upgrade progress and result are called back through handle

Interface declaration:

```
if let path = Bundle.main.path(forResource: "otafileName", ofType: nil) {
  let url = URL(fileURLWithPath: path)
  RingManager.shared.startOTA(url) { res in
    switch res {
    case .start:
print(" Start upgrading =====>")
    case .progress(let value):
print(" Upgrade progress =====>\(value)")
    case .success:
print(" Upgrade successful =====>")
    case .fail(let errorString):
print(" Upgrade failed =====>\(errorString)")
}
}
}
```

Note: When calling this interface, ensure that it is connected to the ring

4、 Database RingDBManager

Use func readDatas(progressBlock: @escaping (Double, RingDataModel)->Void, resultBlock: escaping (Result<ReadDataResult, ReadError>)->Void) gets the local history from the device and saves it to the local

database, which is accessed using the RingDBManager class. A singleton is provided internally to manipulate the relevant data. The singleton can be obtained by RingDBManager.shared. The following is the related API description.

4.1 Get the specified time data

Interface description: Get all the historical data from the specified time to date from the database with a 10-bit timestamp parameter
Interface declaration:

```
1. public func getObjects(from timestamp:TimeInterval) -> [RingDataModel]
```

Note: When invoking this interface, ensure that it is connected to the ring

4.2 Get the data of a certain day

Interface description: Get the historical data of a certain day from the database (get the data from 00:00 to 24:00 on the date)
Interface declaration:

```
1. public func getObjects(of date:Date) -> [RingDataModel]
```

Note: When invoking this interface, ensure that it is connected to the ring

4.3 Get the last piece of data

Interface Description: Obtains the latest historical data from the database at the current time
Interface declaration:

```
1. public func getLatestObject() -> RingDataModel?
```

Note: When invoking this interface, ensure that it is connected to the ring

4.4 Get the sleep data of a certain day

Interface description: Get the sleep data of a certain day from the database (get the data from 18:00 on the previous day to 18:00 on the date). The data obtained by this part of the interface is mainly used for the calculation of sleep
Interface declaration:

```
1. public func getSleepObjects(of date:Date) -> [RingDataModel]
```

Note: When invoking this interface, ensure that it is connected to the ring

4.5 Delete all data

Interface Description: Delete all historical data from the local database
Interface statement:

```
1. public func deleteAll() -> Bool
```

Note: When invoking this interface, ensure that it is connected to the ring

4.6 Delete the data from the specified time to the present

Interface description: Delete all historical data from the specified time up to the present time with a 10-bit timestamp parameter
Interface declaration:

```
1. public func deleteAllBeforeTimestamp(timestamp:TimeInterval) -> Bool
```

Note: Before invoking this interface, ensure that it is connected to the ring

5、 Logical algorithm

This part of the code is in the RingManager class, use ringmanager.shared to get the singleton, and then call the relevant API

5.1 Calculate walking distance

Interface description: Walking distance calculation, steps parameter is the number of steps, stepSize parameter is the step length, the unit is centimeter (cm), the return result is the distance, the unit is meters (m)
Interface declaration:

```
1. func calculateDistance(steps:Int,stepSize:Int) -> Float
```

Note: When calling this interface, ensure that it is connected to the ring

5.2 Sleep time calculation

Interface description: Sleep time calculation, obtain the specified date of sleep data and sporadic sleep data.The return value is a tuple, the first element of the tuple ($0.0) is the sleep data collection, the second element ($0.1) is a two-dimensional array, is a collection of sporadic sleep segments.The time of the first data point in the $0.0 array is the time of falling asleep, the time of the last data point is the time of waking up, and the time of each data point in the middle is the sleep time.In the same way, the sporadic sleep duration can be calculated by using the data in the $0.1 array.
Interface declaration:

```
1. func caculateSleepData(targetDate: Date) -> ([RingDataModel], [[RingDataModel
```

```
]])
```

Note: When invoking this interface, ensure that it is connected to the ring

5.3 Get sleep duration

Interface description: Get the sleep duration, pass the sleep time data point set, you can get the sleep duration, return the sleep duration unit is minute (min)
Interface declaration:

```
1. func calculateSleepTimes(sleepDatas:[RingDataModel]) -> Int
```

Note: When invoking this interface, ensure that it is connected to the ring

6、 Log configuration

Interface description: Set the log saving path, the default is saved in the sandbox Document.You can modify the default save path through the following API
Interface declaration:

```
1. func configLogPath(directoryPath: String = defaultLogDirectoryPath)
```

Note: When invoking this interface, ensure that it is connected to the ring

IV、 Others

1、 Problems that may be encountered

1.1 Errors using the SDK

Upgrade your development tools to the appropriate range
nullSupported iOS version: iOS 13.0+
nullSupported language versions: Swift 5.0+
nullDeveloper tool: xcode 15+

1.2 When reading the software version, it returns two more Spaces

The version number is 10 bytes, with Spaces filled in.
Supplement: Firmware versions are usually numbers with the letter S at the end

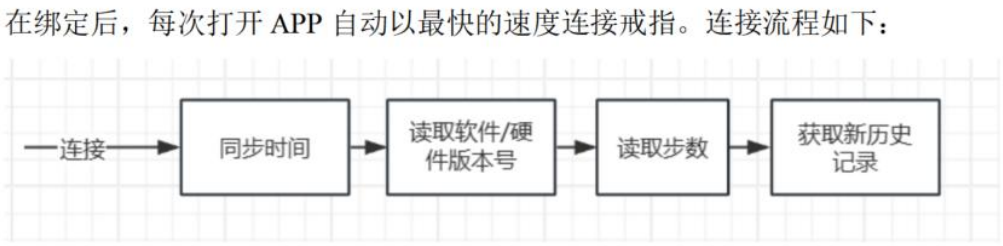
1.3 Call interface, return timeout

Send a command, after 3 seconds, the ring does not respond, will return a timeout
The timeout problem should be analyzed in detail

1.4 How can the sleep function of the ring be measured

Just wear the ring to sleep.If the APP reads the readDatas interface during sleep, it can also get the current sleep data.

1.5 The sleep data read contains the previous days' data



Probably because the time stamps are not synchronized

1.6 Cloud OTA Updates

This SDK provides path update mode. OTA firmware package is placed in the cloud. After downloading, the path of firmware package can be given to SDK
Note: OTA update version will clear the data

1.7 The database has no data or the database only has n data

Problem: I called readData, but I can't get the data in db
Specific: RingManager. Shared. ReadHeartRate callback, after readDatas, getObjects database data
Solution: Delete the APP that accesses the SDK and reinstall it (the reason is that the data read is judged to be not the first read)

2、 Supplementary information

2.1 Sleep data

Provide sleep use case tests (contact us, send as zip pack) including wake up, baseline, fall asleep, wake up

during sleep, and combination tests with.DAT files under each item. The data is analyzed as follows

2.1.1 Follow the protocol

1 Change not 1 is to fall asleep, and change 1 again is to be awake

2.1.2 Data parsing

Chronotypes:

0: Invalid

1: Sober

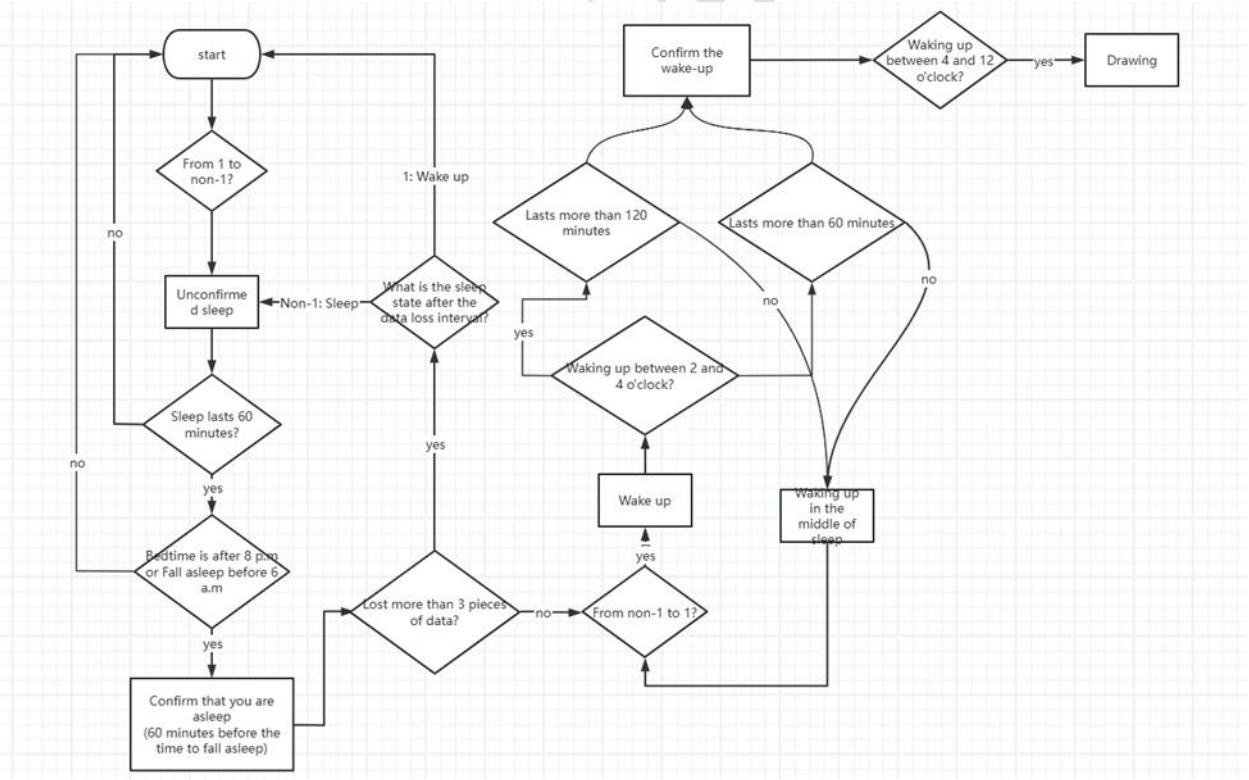
2: Light sleep

3: Deep sleep

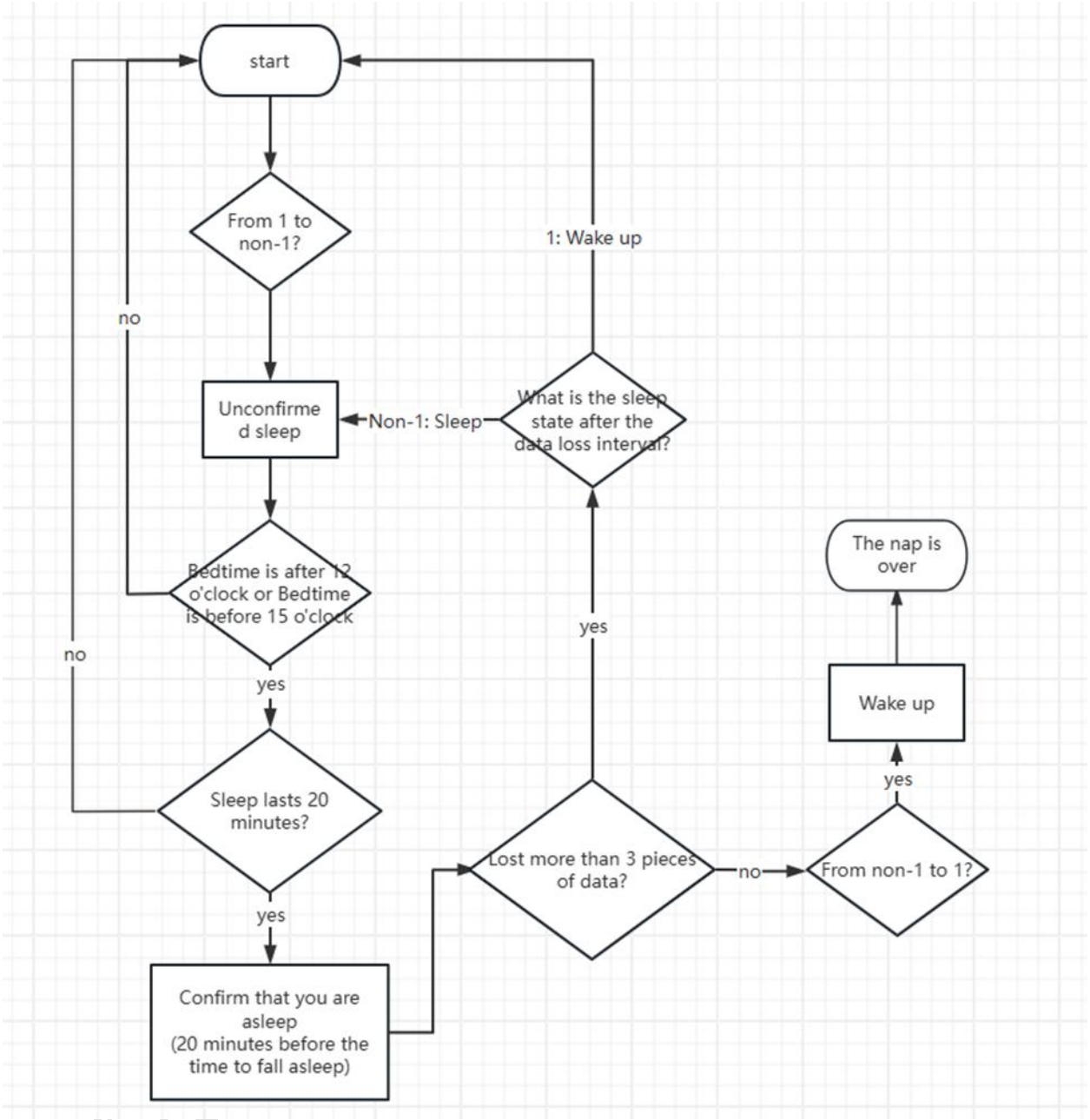
4: Eye movement period

```
[ 180:  12]2023.10.18_21.00.00 step:    0 hr: 70 spo2:  0 hrv:  0 sprit:0 temp:   0 sport:0 sleep:1 sleep_time:    0 rr_num: 0
[ 180:  13]2023.10.18_21.05.00 step:    0 hr: 70 spo2:  0 hrv:  0 sprit:0 temp:   0 sport:0 sleep:2 sleep_time:    0 rr_num: 0
[ 180:  14]2023.10.18_21.10.00 step:    0 hr: 70 spo2:  0 hrv:  0 sprit:0 temp:   0 sport:0 sleep:2 sleep_time:    0 rr_num: 0
[ 180:  15]2023.10.18_21.15.00 step:    0 hr: 70 spo2:  0 hrv:  0 sprit:0 temp:   0 sport:0 sleep:2 sleep_time:    0 rr_num: 0
[ 180:  16]2023.10.18_21.20.00 step:    0 hr: 70 spo2:  0 hrv:  0 sprit:0 temp:   0 sport:0 sleep:2 sleep_time:    0 rr_num: 0
```

2.2 Sleep logic diagrams



2.3 Nap logic diagram



3、 Q&A

3.1 Ring

Q：Will manual measurement data be saved to the database? Heart rate, blood, oxygen?

A：yes

Q：What is the default collection period?

A：1200s/20min

Q：How is the ring data generated and what is the generated data structure?

A：The ring will be automatically tested every certain time (default is 5 minutes), including the content of the test, including heart rate, blood oxygen, heart rate variability, sleep status, etc., so as to generate a sample data point, a data point corresponds to a RingDataModel model object, the sample data will be stored in the ring, over time, More and more data points will be stored in the ring, but only the most recent 7 days of data will be saved.

Q:Where is the log when testing with ChipletRing

A:Find my → Log on the APP page

3.2 SDK

Q：How do I use the SDK to read data from the ring?

A： After connecting the ring, the data in the ring will not be actively reported, you need to call the corresponding SDK method to obtain, you can use the following methods to read the data in the ring

```
1.  func readDatas(progressBlock: @escaping (Double, RingDataModel)->Void, r
    esultBlock: @escaping (Result<ReadDataResult, ReadError>)->Void)
```

Note: When calling this method for the first time, all the historical data points stored in the ring will be read. After successfully obtaining all the historical data, calling this method will only read the new data points that have not been read in the ring. If no new data is generated in the ring or the data in the ring has been read, the data obtained by calling this method will be empty. Internally, this method stores all the data points it gets into a local database built into the SDK.

How does the SDK know if the data has been read?

In the preferences, there is a local save record that has not been read

Q：How do I get data for a given date from the SDK's built-in database?

A：After we read the data from the ring, the RingDBManager management class can be used to get the relevant data, and getObjects(of date:Date) can be used to get the specified date data. Such as getting today's data

```
1.  // Get today's data from the database
2.  let date = Date()
3.  let datasOfToday = RingDBManager.shared.getObjects(of: date)
```

Q：The general process of using the SDK to develop apps and obtain data

A：After connecting the device, use the following method to read the data in the ring

```
1.  func readDatas(progressBlock: @escaping (Double, RingDataModel)->Void, res
    ultBlock: @escaping (Result<ReadDataResult, ReadError>)->Void)
```

After obtaining the data successfully, call the following method to obtain the data of the specified date from the database to get the latest data.

```
1.  func caculateSleepData(targetDate: Date) -> ([RingDataModel], [[RingDataMo
    del]])
```

The return value is a tuple, and the first element of the tuple ($0.0) is the sleep data set, where the first data point is the sleep fall point data, and the last data point is the sleep wake point data. The second element of the tuple ($0.1) is a two-dimensional array that is a collection of sporadic sleep segments. Each of these arrays has the same meaning as described above.

Q： After obtaining the sleep data, how to obtain the sleep time, wake time and sleep time? How to calculate the wake time, light sleep time, deep sleep time and eye movement time?

A：for example

```
        // Get sleep data from last night
        let date = Date()
        let allDatasOfSleeps = RingManager.shared.caculateSleepData(targetDate: date)
        let datas = allDatasOfSleeps.0
        print("sleeping ======>\(String(describing: datas.first?.timestamp))")
        print("waking ======>\(String(describing: datas.last?.timestamp))")

        let sleepTimes = RingManager.shared.calculateSleepTimes(sleepDatas: datas)
        print("睡眠时间 ======>\(sleepTimes)分钟")


        var lastModel:RingDataModel?
        // Wakefulness time, in seconds
        var wakeTimes:UInt32 = 0
        // Light sleep duration, in seconds
        var lightSleepTimes:UInt32 = 0
        // Length of deep sleep, in seconds
        var deepSLeepTimes:UInt32 = 0
        // Eye movement duration, in seconds
        var eyesTimes:UInt32 = 0
        datas.forEach { model in
            if let tempLastModel = lastModel {
                switch model.sleepType {
                case 1:// Wakefulness
                    wakeTimes += model.timestamp - tempLastModel.timestamp
                case 2:// Light sleep
                    lightSleepTimes += model.timestamp - tempLastModel.timestamp
                case 3:// deep sleep
                    deepSLeepTimes += model.timestamp - tempLastModel.timestamp
                case 4:// Eye movement
                    eyesTimes += model.timestamp - tempLastModel.timestamp
                default :
```

```
                    break
                }
                lastModel = model
            }else{
                lastModel = model
            }
        }

        print("wakeTimes ======>\(wakeTimes)")
        print("lightSleepTimes ======>\(lightSleepTimes)")
        print("deepSLeepTimes ======>\(deepSLeepTimes)")
        print("eyesTimes ======>\(eyesTimes)")
```

3.3 Troubleshooting

3.3.1 Failed to Search for Nearby Bluetooth Devices Using the SDK

```
1. iOS demo run RingManager.shared.startScanMethod keeps reporting
   errors Rings-SDK_Example[44209:4371892] [CoreBluetooth] API MISUSE: <CBCent
   ralManager: 0x2816cb680> can only accept this command while in the powered
   on state
```

Situation: in the process of using the SDK, call RingManager. Shared the startScan program for the next step, not stuck here, in the debugging shows that the above message

Troubleshooting method: After testing with ChipletRingAPP, it is found that Bluetooth can be found, but the connection cannot be connected: rssi: -74, the connection fails

Find the reason: the ring is out of power/the charge is too low, the problem is solved after charging.

Note: RingManager. Shared. After startScan call don't care about the message, a search will be automatically connected to the equipment.

3.3.2 Failed to Decompress the Compressed Package

Here provide the git web site: https://github.com/wcb133/Rings-SDK. downloadable