Demo APP for MicroLifeDeviceSDK (iOS)

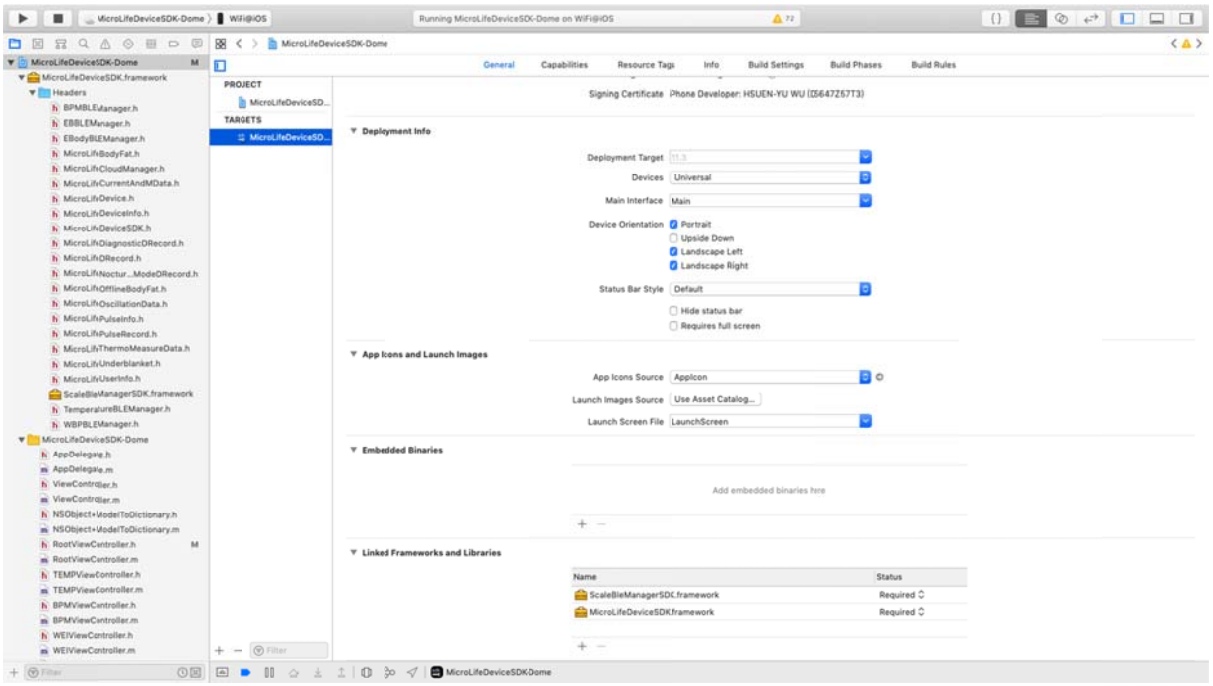
Table of Contents

Chapter 1 Development Environment

This user manual serves as a quick introduction to MicroLifeDeviceSDK / APIs and shows how to integrate into an iOS Demo App.

1.1. First of all, add MicroLifeDeviceSDK.framework into a development project. The minimum supported version is iOS 10. The compatible version of Xcode IDE is 10.2.1.

1.2. Under TARGETS / General / Linked Frameworks and Libraries, add MicroLifeDeviceSDK.framework.



1.3. Under TARGETS / Build Settings / Enable, set Bitcode to NO.

1.4. Under TARGETS / Build Settings / Other Linkeder Flags, use the linker flag “-all_load”.

1.5. Import header file as bellows.

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

Chapter 2 Operation Sequence of BLEManager

The BLEManager is responsible for managing the Bluetooth communication.

2.1. Initiate a BLEManage and register the delegation of BLEManager. The DataResponseDelegate is dedicated for response messages of BLEManager.

2.2. Once the BLEManager is initiated, the BLEManager checks the status of Bluetooth by BLEManagerCellPhoneBluetoothDidUpdateState.

2.3. While turning on Bluetooth, BLEManager runs the scan/ discovery procedure automatically by BLEManagerDidDiscoverBluetoothDeviceMacAddress to get devices' information in the vicinity.

2.4. Call the bindingDevice:(NSData *)macAddress to bind an activated device. The status of binding can be found by the following section 2.5. The BLEManager will automatically connect to a bound device while it is activated.

2.5. Connection status：

2.5.1. Connection / binding：

- (void) * BLEManagerDidConnectDevice;

2.5.2. Fail of connection：

- (void) * BLEManagerDidFailToConnectDevice;

2.5.3. Disconnection：

- (void) * BLEManagerDidDisconnectDevice;

Chapter 3 APIs of BLEManager

By utilizing the APIs of BLEManager, can transfer data between device and App via Bluetooth. The APIs includes two parts: one is for fundamental Bluetooth and another is dedicated for communication with a designated device / BPM (Blood Pressure Monitor) with specific commands and protocol.

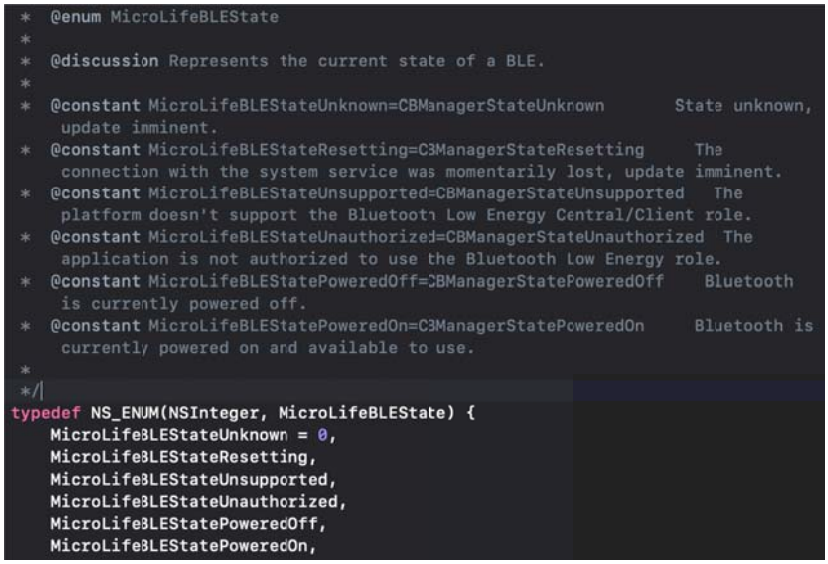
3.1. Device searching：

3.1.1.

	+(instancetype)shareInstanceWhithAuthorizationkey:(NSString *)key Target * Names:(NSArray *)target * Names
Definition	Searching for device with default / customized name
Parameter	Key: Authorization code target * Names: device name which can be included single or multiple
	`self.aBPMBLEManager = [BPMBLEManager shareInstanceWhithAuthorizationkey:$DKkey TargetBPMNames:@[@"A6 BT",@"A6 BASIS PLUS BT",@"A7 TOUCH BT",@"B3 BT",@"B6 Connect",@"A6",@"Progress"]];` `self.aBPMBLEManager.dataResponseDelegate = self;`

3.1.2.

	- (void) *BLEManagerCellPhoneBluetoothDidUpdateState:(MicroLifeBLEState)state;
Definition	Response for the shareInstanceWhithAuthorizationkey (). This is to manage Bluetooth status of cellphone.
Parameter	status: Bluetooth status of cellphone

```
 *  @enum MicroLifeBLEState
 *
 *  @discussion Represents the current state of a BLE.
 *
 *  @constant MicroLifeBLEStateUnknown=CBManagerStateUnknown       State unknown,
     update imminent.
 *  @constant MicroLifeBLEStateResetting=CBManagerStateResetting     The
     connection with the system service was momentarily lost, update imminent.
 *  @constant MicroLifeBLEStateUnsupported=CBManagerStateUnsupported   The
     platform doesn't support the Bluetooth Low Energy Central/Client role.
 *  @constant MicroLifeBLEStateUnauthorized=CBManagerStateUnauthorized  The
     application is not authorized to use the Bluetooth Low Energy role.
 *  @constant MicroLifeBLEStatePoweredOff=CBManagerStatePoweredOff    Bluetooth
     is currently powered off.
 *  @constant MicroLifeBLEStatePoweredOn=CBManagerStatePoweredOn     Bluetooth is
     currently powered on and available to use.
 *
 */
typedef NS_ENUM(NSInteger, MicroLifeBLEState) {
    MicroLifeBLEStateUnknown = 0,
    MicroLifeBLEStateResetting,
    MicroLifeBLEStateUnsupported,
    MicroLifeBLEStateUnauthorized,
    MicroLifeBLEStatePoweredOff,
    MicroLifeBLEStatePoweredOn,
```

	- (void) * BLEManagerDidDiscoverBluetoothDeviceMacAddress:(NSData *)macAddress Name:(NSString *)name RSSI:(NSNumber *)RSSI;
Definition	Response for the shareInstanceWhithAuthorizationkey (). This is to manage information of devices that are discovered in the vicinity.
Parameter	macAddress: MAC address of device name: device name RSSI: RSSI

3.2. Binding / Pairing：

3.2.1.

	- (void)bindingDevice:(NSData *)macAddress
Definition	Binding a specified device by MAC
Parameter	macAddress: MAC address of device

3.2.2.

	- (void) * BLEManagerDidConnectDevice;
Definition	Response for the bindingDevice() with the status Connection

	- (void) * BLEManagerDidFailToConnectDevice
Definition	Response for the bindingDevice() with the status Fail of Connection

	- (void) * BLEManagerDidDisconnectDevice;
Definition	Response for the bindingDevice() with the status Disconection

3.3. MAC address of binding device：

3.3.1.

	- (NSData *)getBindingDevice;
Definition	Get MAC address of a binding device or nil.

3.4. Unbinding：

3.4.1.

	- (void)unBindingDevice;
Definition	Unbinding device

3.4.2.

	- (void) * BLEManagerDidFailToConnectDevice;
Definition	Response for the unBindingDevice()

3.5. Disconnection：

3.5.1.

	- (void)disconnectDevice;
Definition	Disconnect with device

3.5.2.

	- (void) * BLEManagerDidFailToConnectDevice;
Definition	Response for the disconnectDevice ()

3.6. Searching for devices ：

3.6.1.

	- (void)reScan;
Definition	Search for devices repeatedly

3.6.2.

	- (void) * BLEManagerDidDiscoverBluetoothDeviceMacAddress:(NSData *)macAddress Name:(NSString *)name RSSI:(NSNumber *)RSSI;
Definition	This is to get information of devices which are discovered in the vicinity.
Parameter	macAddress: MAC of device name: device name RSSI: RSSI

3.7. Stop searching for devices：

3.7.1.

	- (void)stopScan;
Definition	Stop searching

3.7.2. Null.

3.8. Read usual mode history data from BPM：

3.8.1.

	- (void)readUsualModeHistoryData;
Definition	Read the history data of usual mode from BPM

3.8.2.

	- (void)WBPBLEManagerResponseReadUsualModeHistoryData:(MicroLifeDRecord *)data

	IsNoData:(BOOL)isNoData;
Definition	Response for the readUsualModeHistoryData()
Parameter	data：history data of usual mode isNoData：the status True or False

3.9. Read diagnostic mode history data from BPM：

3.9.1.

	- (void)readDiagnosticModeHistoryData;
Definition	Read diagnostic mode history data from BPM

3.9.2.

	- (void)WBPBLEManagerResponseReadDiagnosticModeHistoryData:(MicroLifeDiagnosticDRecord *)data IsNoData:(BOOL)isNoData;
Definition	Response for the readDiagnosticModeHistoryData()
Parameter	data：history data of diagnostic mode isNoData：the status True or False

3.10. Clear selected mode history data of the BPM：

3.10.1.

	- (void)clearHistoryDataWithSelectedUsualMode:(BOOL)clearUsualMode DiagnosticMode:(BOOL)clearDiagnosticlMode NocturnalMode:(BOOL)clearNocturnalMode;
Definition	Clear history data by selected mode
Parameter	clearUsualMode：clear data for Usual Mode clearDiagnosticlMode：clear data for Diagnosticl Mode clearNocturnalMode：clear data for Nocturnal Mode

3.10.2.

	- (void)WBPBLEManagerResponseClearSelectedModeHistoryData:(BOOL)isSuccess;
Definition	Response for the clearHistoryDataWithSelectedUsualMode()
Parameter	isSuccess：the status True or False

3.11. Clear current mode history data of the BPM：

3.11.1.

	- (void)clearCurrentModeHistoryData;
Definition	Clear data for current mode

3.11.2.

	- (void)WBPBLEManagerResponseClearCurrentModeHistoryData:(BOOL)isSuccess;
Definition	Response for the clearCurrentModeHistoryData()
Parameter	isSuccess：the status True or False

3.12. Disconnect the Bluetooth with BPM：

3.12.1.

	- (void)disconnect;
Definition	Disconnect the Bluetooth with BPM

3.12.2.

	- (void)WBPBLEManagerDidDisconnectDevice;
Definition	Response for the disconnect ()

3.13. Write device Time to BPM：

3.13.1.

	- (void)writeDeviceTime;

Definition	Synchronization between device and cellphone

3.13.2.

	- (void)WBPBLEManagerResponseWriteDeviceTime:(BOOL)isSuccess;
Definition	Response for the writeDeviceTime()
Parameter	isSuccess：the status True or False

3.14. Write a new user ID to BPM：

3.14.1.

	- (void)writeUserID:(NSString *)ID
Definition	Write a new user ID to BPM
Parameter	ID：User ID

3.14.2.

	- (void)WBPBLEManagerResponseWriteUserID:(BOOL)isSuccess;
Definition	Response for the writeUserID ()
Parameter	isSuccess：the status True or False

3.15. Read nocturnal mode setting：

3.15.1.

	- (void)readNocturnalModeSetting;
Definition	Read the setting of Nocturnal mode

3.15.2.

	- (void)WBPBLEManagerResponseReadNocturnalModeSetting:(MicroLifeDeviceInfo *)deviceInfo;

Definition	Response for the readNocturnalModeSetting()
Parameter	deviceInfo：device information (setting) of nocturnal mode

3.16. Change nocturnal mode setting ：

3.16.1.

	- (void)changeNocturnalModeSettingOn:(BOOL)open StartYear:(NSInteger)year StartMonth:(NSInteger)month StartDay:(NSInteger)day StartHour:(NSInteger)hour;
Definition	Change the setting of nocturnal mode Note: This command has to match with specific device. It can be used the following: "readDeviceIDAndInfo(MicroLifeDeviceInfo.openNocturnalMode)" to check if the device supports it or not.
Parameter	open： ON/OFF for Nocturnal mode year：year month：month day：day hour：hour

3.16.2.

	- (void)WBPBLEManagerResponseChangeNocturnalModeSetting:(BOOL)isSuccess;
Definition	Response for the changeNocturnalModeSettingOn()
Parameter	isSuccess：the status True or False

3.17. Erase all of device measure & error times：

3.17.1.

	- (void)eraseAllOfDeviceMeasureAndErrorTimes;
Definition	Erase all of device measure & error times

3.17.2.

	- (void)WBPBLEManagerResponseEraseAllOfDeviceMeasureAndErrorTimes:(BOOL)isSuccess;;
Definition	Response for the eraseAllOfDeviceMeasureAndErrorTimes ()
Parameter	isSuccess：the status True or False

3.18. Read device ID and info from BPM：

3.18.1.

	- (void)readDeviceIDAndInfo;
Definition	Read device ID and info from BPM

3.18.2.

	- (void)WBPBLEManagerResponseReadDeviceIDAndInfo:(MicroLifeDeviceInfo *)deviceInfo;
Definition	Response for the readDeviceIDAndInfo ()
Parameter	deviceInfo：device ID and info

3.19. Read device Time from BPM：

3.19.1.

	- (void)readDeviceTime;
Definition	Read device Time from BPM

3.19.2.

	- (void)WBPBLEManagerResponseReadDeviceTime:(MicroLifeDeviceInfo *)deviceInfo;
Definition	Response for the readDeviceTime ()
Parameter	deviceInfo：Time of device

3.20. Read user ID and version data from BPM：

3.20.1.

	- (void)readUserAndVersionData;
Definition	Read user ID and version data from BPM

3.20.2.

	- (void)WBPBLEManagerResponseReadUserAndVersionData:(MicroLifeUserInfo *)user VersionData:(MicroLifeDeviceInfo *)verData;
Definition	Response for the readUserAndVersionData ()
Parameter	user：user ID verData：version data

3.21. Read Nocturnal mode history data from BPM：

3.21.1.

	- (void)readNocturnalModeHistoryData;
Definition	Read the history data of Nocturnal mode from BPM

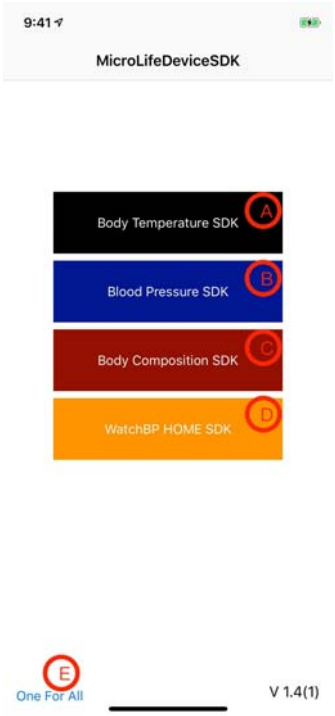
3.21.2.

	- (void)WBPBLEManagerResponseReadNocturnalPatternHistoryData:(MicroLifeNocturnalModeDRecord *)data IsNoData:(BOOL)isNoData;
Definition	Response for the readNocturnalModeHistoryData ()
Parameter	data：Nocturnal pattern history data isNoData：the status True or False

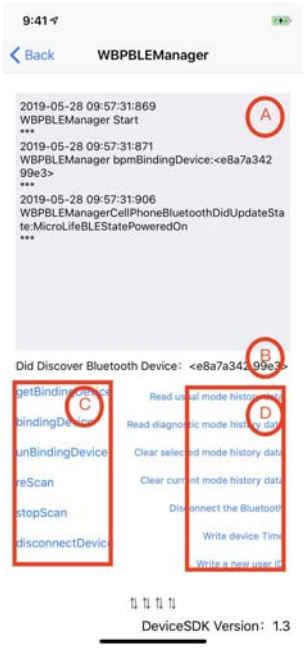
Chapter 4 Instruction of Demo App

4.1. Getting Started：

Start the app and then select the button “WatchBP HOME SDK” / “D” to communicate with the designate device.



4.2. Operating Interface and Sequence：



4.2.1. Region A：The log window is used to display information about communication handshake between App and device.

4.2.2. Region B：The display is for current device with MAC address.

4.2.3. Region C：This part is included fundamental Bluetooth tasks such as Connection, Disconnection and Bonding/ Pairing.

4.2.4. Region D：This part is to communicate with the device WatchBP HOME by different functions / commands such as data transferring, synchronization and so on.

4.2.5. Refer to WBPViewController from the demo code (sample code) to get more detailed.

4.2.6. Operation Sequence：

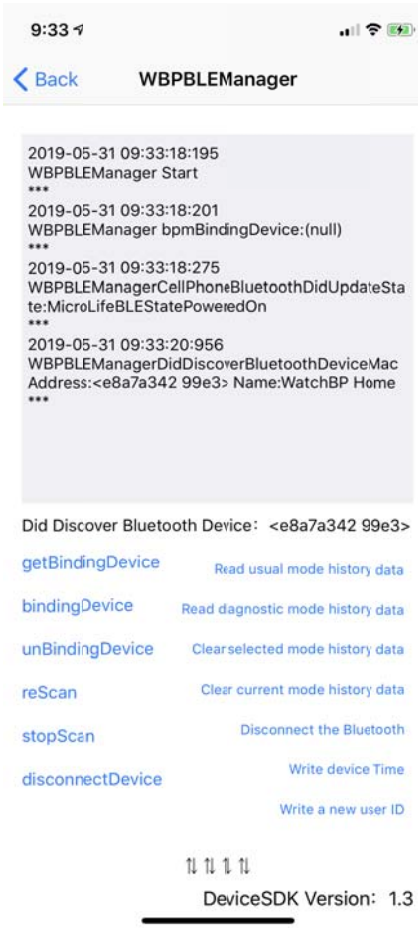
4.2.6.1. The scanning (discovery) is automatically run to discover devices in the vicinity.

4.2.6.2. If a device is bonded, it will be connected accordingly. If not, the “bindingDevice” can be used to run bonding process.

4.2.6.3. Once the device is connected, select each function / command from Region D to manage Read and Write.

Chapter 5 Functionality of Demo App

5.1. Scanning / Discovery：



1. First of all, the WBPBLEManager starts Scanning / Discovery procedure. The detailed information will be displayed in Region A.

2. The WBPBLEManager bpmBindingDevice: * : (null) indicates that the remote device wasn't bonded yet.

3. The WBPBLEManagerCellPhoneBluetoothDidUpdateState:MicroLifeBLEStatePoweredOn indicates that the power status of Bluetooth from cell phone is On.

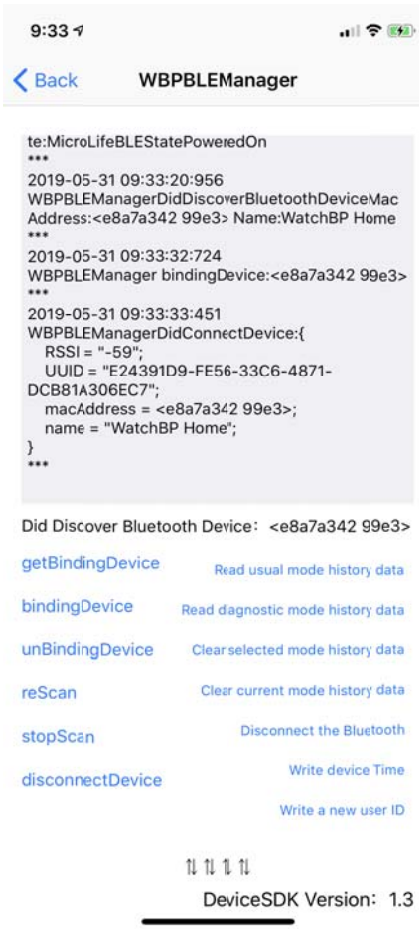
4. The WBPBLEManagerDidDiscoverBluetoothDeviceMacAddress: <e8a7a342 99e3> Name:WatchBP Home indicates that the device with MAC address <e8a7a342 99e3> is discovered nearby.

5. In Region B, the "Did Discover Bluetooth Device : <e8a7a342 99e3>" stands for that the discovered/ selected device can be used in next steps such as Connection or Bonding.

6. In Region C, click the

	function “bindingDevice” to run bonding process.

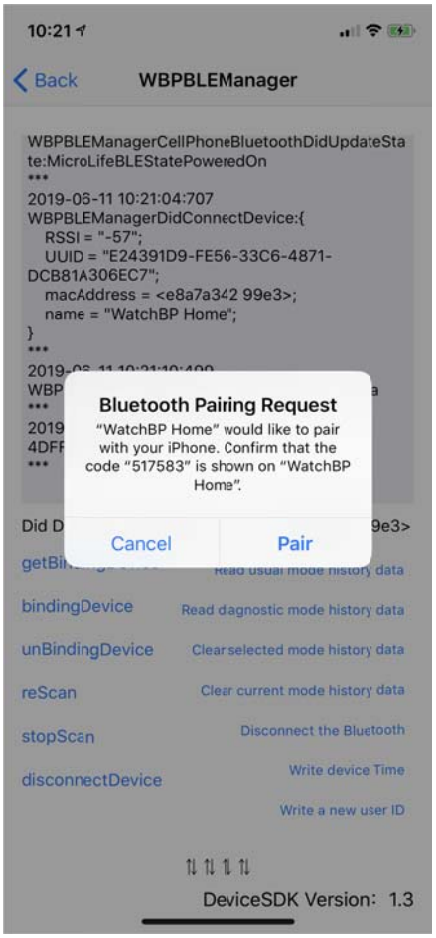
5.2. Connection：



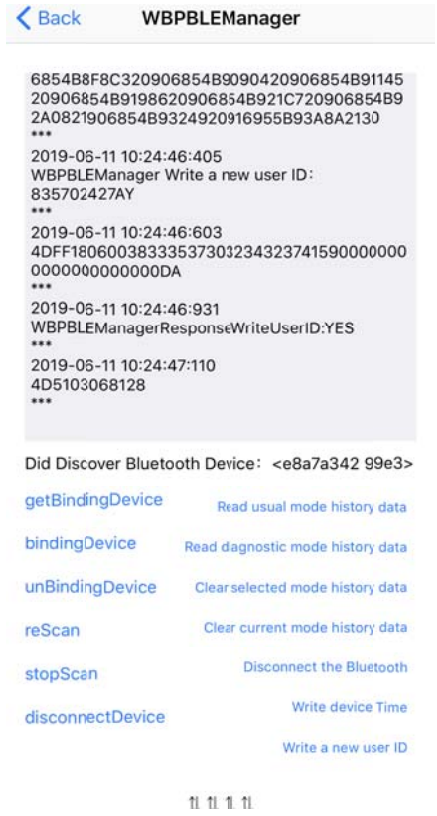
1 The information about the device bonding will be shown “WBPBLEManager bindingDevice:<e8a7a342 99e3>” in Region A.

2.The following indicates that the device is connected.
WBPBLEManagerDidConnectDevice:{
RSSI = "-68";
UUID = "E24391D9-FE56-33C6-4871-DCB81A306EC7";
macAddress = <e8a7a342 99e3>;
name = "WatchBP Home";
}

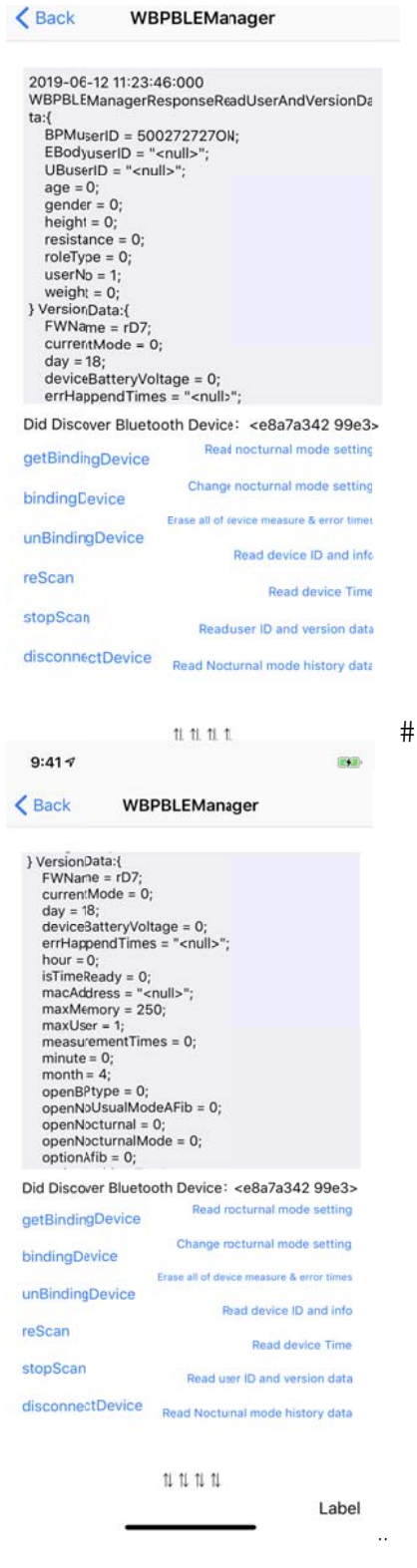
5.3. Bonding / Pairing：

	1. There is a message to confirm the bonding between device and cellphone if they haven't bonded yet. 2. Click "Pair" to run the bonding process.

5.4. Testing Command：Write a new user ID to BPM

	1. The command “Write a new user ID” is to write a new user ID to device. 2. The following “WBPBLEManager Write a new user ID：835702427AY”indicates that the writing value is “835702427AY” made up of ASCII code. 3. The response is as below: WBPBLEManagerResponse WriteUserID:YES. It means that the process is successful. 4. It can be checked by the command “Read user ID and version data”.

5.5. Testing Command：Read user ID and version data from BPM



#

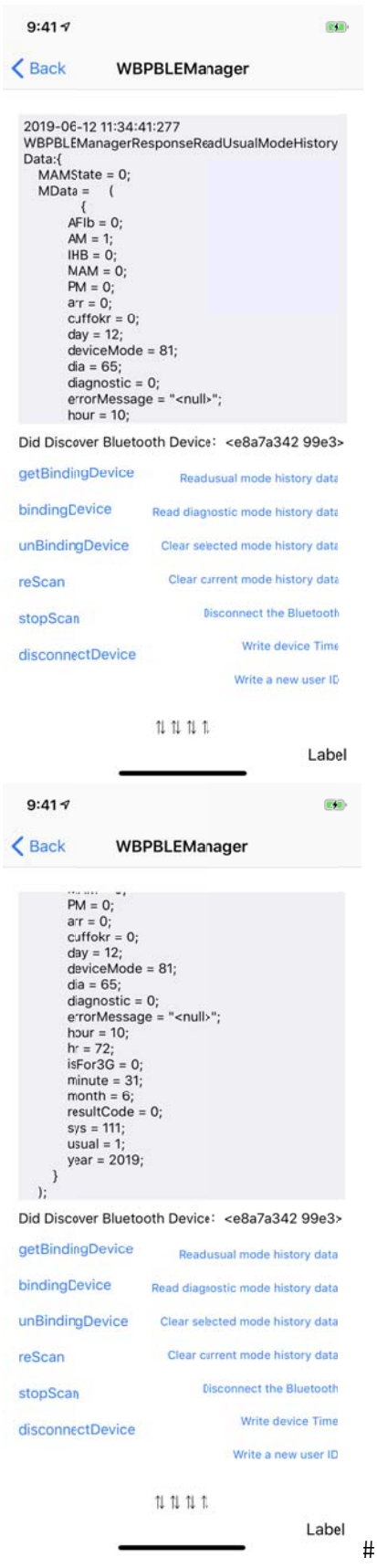
1. The command “Read user ID and version data” is to receive the information of user ID and device version.

2. The response is as below:

```
WBPBLEManagerResponseReadUserAndVersionData:{
    BPMuserID = 500272727ON;
    EBodyuserID = "<null>";
    UBuserID = "<null>";
    age = 0;
    gender = 0;
    height = 0;
    resistance = 0;
    roleType = 0;
    userNo = 1;
    weight = 0;
} VersionData:{
    FWName = rD7;
    currentMode = 0;
    day = 18;
    deviceBatteryVoltage = 0;
    errHappendTimes = "<null>";
    hour = 0;
    isTimeReady = 0;
    macAddress = "<null>";
    maxMemory = 250;
    maxUser = 1;
    measurementTimes = 0;
    minute = 0;
    month = 4;
    openBPtype = 0;
    openNoUsualModeAFib = 0;
    openNocturnal = 0;
    openNocturnalMode = 0;
    optionAfib = 0;
    optionAmbientT = 1;
    optionDeviceID = 0;
    optionDiagnosticModeAFib = 0;
    optionIHB = 1;
    optionMAM = 1;
```

	optionTubeless = 0;
	protocolVersion = 0;
	second = 0;
	year = 2019;
	}
	
	3. The “BPMuserID” stands for user ID and the value is “500272727ON”.
	
	4. The “VersionData” stands for device version included name of firmware “FWName” and its building date “day”, “month” and “year”.

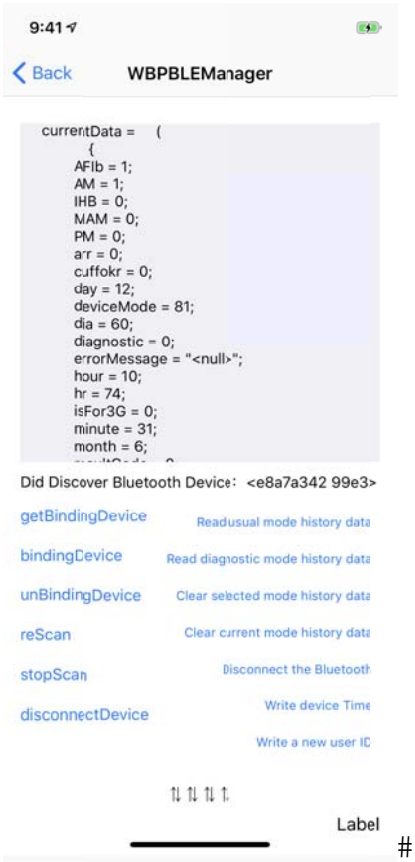
5.6. Testing Command：Read usual mode history data from BPM

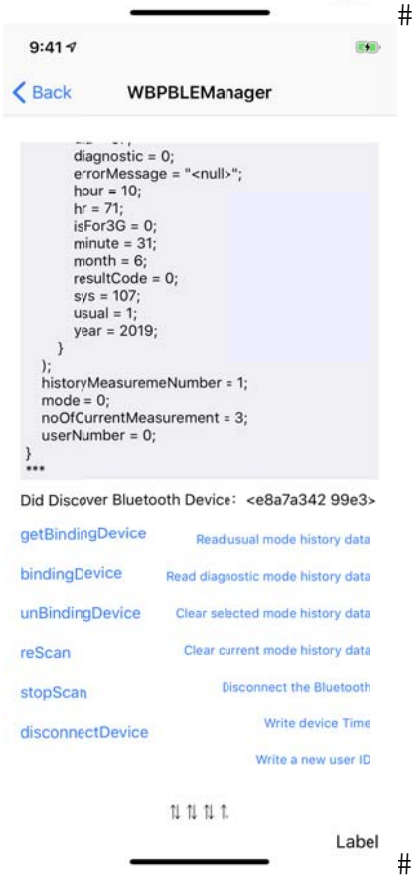


1. The command “Read usual mode history data” is to read the data of usual mode.

2. The response is as below:

```
WBPBLEManagerResponseReadUsualModeHistoryData:{
    MAMState = 0;
    MData =     (
                {
            AFIb = 0;
            AM = 1;
            IHB = 0;
            MAM = 0;
            PM = 0;
            arr = 0;
            cuffokr = 0;
            day = 12;
            deviceMode = 81;
            dia = 65;
            diagnostic = 0;
            errorMessage = "<null>";
            hour = 10;
            hr = 72;
            isFor3G = 0;
            minute = 31;
            month = 6;
            resultCode = 0;
            sys = 111;
            usual = 1;
            year = 2019;
        }
    );
    currentData =     (
                {
            AFIb = 1;
            AM = 1;
            IHB = 0;
            MAM = 0;
            PM = 0;
            arr = 0;
            cuffokr = 0;
            day = 12;
            deviceMode = 81;
            dia = 60;
```



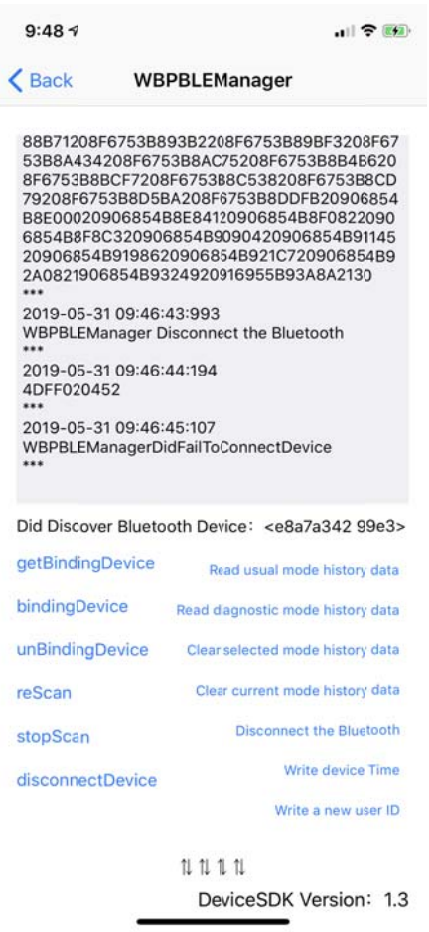



```
            diagnostic = 0;
            errorMessage =
"<null>";
            hour = 10;
            hr = 74;
            isFor3G = 0;
            minute = 31;
            month = 6;
            resultCode = 0;
            sys = 115;
            usual = 1;
            year = 2019;
        },
                {...},
                {...}
    );
    historyMeasuremeNumber
= 1;
    mode = 0;
    noOfCurrentMeasurement
= 3;
    userNumber = 0;
}
```

3. Both “MData” & “currentData” are made up of blood pressure reading systolic (sys), diastolic (dia), pulse (hr), and date & time (year, month, day, hour, minute) respectively. The “MData” stands for the “history” data. The “currentData” stands for the “current” measurement data.

4. For instance, the above-mentioned MData stands for “2019/6/12 10:31 sys=111 dia=62 hr=72…”.

5.7. Disconnection



1. In order to disconnect the device completely, either “disconnectDevice” (Region C) or “Disconnect the Bluetooth” (Region D) is available.

2. The response is the following “WBPBLEManagerDidFailToConnectDevice” once the disconnection command is executed.