WCHBLElib 库接口函数说明

版本：1A

http://wch.cn

一、概述

WCHBLElib是Linux系统下的BLE通用接口库，提供BLE设备的扫描、连接和检测，蓝牙服务和特征枚举、读写、通知，控制器版本查询等操作API函数。

二、库函数介绍

2.1. 查询蓝牙控制器版本

int WCHBLEGetBluetoothVer();

函数正确返回值对应下表：

返回值	版本号
0	V1.0
1	V1.1
2	V1.2
3	V2.0
4	V2.1
5	V3.0
6	V4.0
7	V4.1
8	V4.2
9	V5.0
10	V5.1
11	V5.2

返回值为-1 表示设备查询超时，返回值为-2 表示系统蓝牙未打开。

2.2. 查询控制器是否支持低功耗蓝牙

bool WCHBLEIsLowEnergySupported();

函数返回值为 FALSE 表示不支持；

返回值为 TRUE 表示支持。

2.3. 设置 BLE 第二广播（secondary advertising）

void WCHBle_Set_Secondary_Advertising(uint32_t phy);

该函数可以设置第二广播模式 1M/2M/CODED，BLE5.0 及以上版本支持该函数。

参数 phy 表示选择第二广播的模式：1 表示 1M，2 表示 2M，3 表示 CODED。

2.4.设置 BLE 发送接收（TX/RX）PHY

void WCHBle_Set_PHY_TX_RX(uint32_t phys);

该函数可设置发送接收 PHY，BLE5.0 及以上版本支持该函数。

参数 phys 为设置发送接收 PHY 模式：1537 表示 1M，7681 表示 2M，32257 表示 CODED。

2.5. 扫描 BLE 设备

int WCHBle_BLE_Scan(int ScanTime, FunDiscoverDeviceInfo Ble_AdvertisingDevice_Info);

该函数会在规定时间内扫描附近的 BLE 设备，通过扫描回调函数返回扫描结果（设备地址、设备名称、信号强度 RSSI）。

参数 ScanTime 是设定的扫描时间，以秒为单位；

参数 Ble_AdvertisingDevice_Info 为扫描结果的回调函数（设备地址、设备名称、信号强度 RSSI）；

函数返回值为 1 表示系统蓝牙未打开；

返回值为 0 表示扫描成功。

2.6. 连接设备

WCHBLEHANDLE* WCHBle_Connect(const char *mac, FunConnectionStateCallback connectionstate);

该函数通过 mac 地址连接设备。

函数返回值为 WCHBLEHANDLE，此句柄在后续函数调用时将作为参数传递；

参数 mac 是设备地址，可通过扫描 BLE 设备函数获取；

参数 connectionstate 是连接回调函数，上传连接状态。

2.7. 断开设备连接

void WCHBle_Disconnect(WCHBLEHANDLE *connection);

该函数将断开已连接的设备；

connection 为设备连接句柄。

2.8. 注册连接断开回调

void WCHBle_register_on_disconnect(WCHBLEHANDLE *connection, FunDisconnectionStateCallback disconnection_state);

该函数将注册设备连接断开事件。

connection 为设备连接句柄。

disconnection_state 为连接断开状态回调

2.9. 获取设备服务

int WCHBle_Discover_Primary(WCHBLEHANDLE *connection, GattPrimaryService *Services, int *Services_Count);

该函数将获取连接设备的所有服务。

参数 connection 为设备连接句柄；

参数 Services 获取设备的服务；

参数 Services_Count 获取设备服务的个数；

函数返回值为 0 表示获取服务成功；

函数返回值为-1 表示服务失败。

2.10. 获取设备特征

int WCHBle_Discover_Characteristics(WCHBLEHANDLE *connection,

GattCharacteristic *Characteristics, int *Characteristics_Count);

该函数用于获取连接设备的所有特征。

参数 connection 为设备连接句柄；

参数 Characteristics 为获取设备特征的结构体数组，结构体定义如下：

```
typedef struct {
    uint16_t  handle;
    uint8_t   properties;
    uuid_t    uuid;
} GattCharacteristic;
```

其中 handle 为特征句柄，uuid 为特征标识；

properties 为特征的特性值，具体含义如下：

bit0(0x01)为 1 支持广播（broadcast）操作，为 0 表示不支持广播模式；

bit1(0x02)为 1 表示支持读操作，为 0 则表示不支持读操作；

bit2(0x04)为 1 表示支持无应答写操作（write without response），为 0 则表示不支持无应答写操作；

bit3(0x08)为 1 表示支持有应答写操作（write with response），为 0 则表示不支持有应答写操作；

bit4(0x10)为 1 表示支持通知操作（notification），为 0 则表示不支持通知操作。

参数 Characteristics_Count 获取特征的个数；

函数返回值为 0 表示获取特征成功；

函数返回值为-1 表示获取特征失败。

2.11. 写入特征值

int WCHBle_Write_Characteristic(WCHBLEHANDLE *connection, const char *CharacteristicUUID, bool WriteWithResponse, const char *Buffer, size_t Buffer_Length);

该函数用于写入特征值。

参数 connection 为设备连接句柄；

参数 CharacteristicUUID 为特征标识；

参数 WriteWithResponse 为传输模式，为 0 表示无应答传输，为 1 表示有应答传输；

参数 Buffer 为代写入的字符串；

参数 Buffer_Length 为代写入字符串的长度；

函数返回值为 0 成功；

函数返回值为 1 表示写入失败；

2.12. 读取特征值

int WCHBle_Read_Char_by_UUID(WCHBLEHANDLE *connection, const char *CharacteristicUUID, char *Buffer, size_t *Buffer_Length);

该函数用于读取特征值。

参数 connection 为设备连接句柄；

参数 CharacteristicUUID 为特征标识；

参数 Buffer 为存放读取结果的字符串；

参数 Buffer_Length 为期望读取的字符串长度；

函数返回值 0 表示读取特征成功；

其他返回值表示读取特征失败。

2.13. 注册通知

void WCHBle_register_notification(WCHBLEHANDLE *connection, FunRegisterNotifyCallback notification_handler);

该函数用于注册通知，通过通知回调函数接收数据。

参数 connection 为设备连接句柄；

参数 notification_handler 为通知回调函数。

2.14. 打开通知

int WCHBle_Open_Notification(WCHBLEHANDLE *connection, const char *CharacteristicUUID);

该函数用于打开通知。

参数 connection 为设备连接句柄；

参数 CharacteristicUUID 为特征标识；

函数返回值为 0 表示打开通知成功；

函数返回值-1 表示打开通知失败。

2.15. 关闭通知

int WCHBle_Close_Notification(WCHBLEHANDLE *connection, const char *CharacteristicUUID);

该函数用于关闭通知。

参数 connection 为设备连接句柄；

参数 CharacteristicUUID 为特征标识；

函数返回值为 0 表示关闭通知成功；

函数返回值为-1 表示关闭通知失败。

2.16. 获取设备 MTU

int WCHBle_Get_MTU(WCHBLEHANDLE *connection, const char *CharacteristicUUID uint16_t *mtu);

该函数用于获取当前连接设备的 MTU 值。

参数 connection 为设备连接句柄；

参数 CharacteristicUUID 为特征的 UUID（该特征需支持无应答写 write without response）；

参数 mtu 为存放获取结果变量；

返回值为 0 表示获取 MTU 值成功，其它返回值表示失败。

三、回调函数介绍

回调函数分别有四种：设备扫描回调、连接状态回调、断开连接状态回调和通知回调。

3.1. 设备扫描回调

typedef void (*FunDiscoverDeviceAdvInfo)(const char *addr, const char *name, int8_t rssi);

该函数用于返回扫描到的 BLE 设备信息。
参数 addr，扫描到的设备地址；
参数 name，扫描到的设备名称；
参数 rssi，扫描到的设备信号强度。

3.2. 连接状态回调

typedef void(*FunConnectionStateCallBack)(WCHBLEHANDLE *connection,int state);
该函数用于返回设备连接时的状态，在连接状态发生改变时会上报当前连接状态。
参数 connection 为设备连接句柄；
参数 state，1 表示连接成功，0 表示连接失败。

3.3. 断开连接状态回调

typedef void(*FunDisconnectionStateCallBack)(void *user_data);
该函数用于上报设备连接断开事件。

3.4. 通知回调

typedef void (*FunRegisterNotifyCallBack)(const uuid_t *uuid, const uint8_t *data, size_t data_length);
该函数用于接收串口发来的数据；
参数 uuid 为特征标识；
参数 data，接收的数据；
参数 data_length，接收数据的长度。

四、接口调用顺序介绍

4.1. 调用顺序一

1. 扫描设备 WCHBle_BLE_Scan
2. 连接设备 WCHBle_Connect
3. 注册断开连接回调 WCHBle_register_on_disconnect
4. 获取设备服务 WCHBle_Discover_Primary
5. 获取设备特征 WCHBle_Discover_Characteristics
6. 注册通知 WCHBle_register_notification
7. 打开通知 WCHBle_Open_Notification
8. 获取设备 MTU 值 WCHBle_Get_MTU
9. 写入特征值 WCHBle_Write_Characteristic
10. 读取特征值 WCHBle_Read_Char_by_UUID
11. 关闭通知 WCHBle_Close_Notification
12. 断开 BLE 设备连接 WCHBle_Disconnect