Description of the WCHBLElib Library Interface Functions

Version: 1A
http://wch.cn

I. Overview

WCHBLElib is a BLE common interface library for Linux system, which provides BLE device scanning, connection and detection, Bluetooth service and feature enumeration, read/write, notification, controller version query and other operation API functions.

II. Introduction to Library Functions

2.1. Query Bluetooth controller version

int WCHBLEGetBluetoothVer(); int WCHBLEGetBluetoothVer().

The correct return value of the function corresponds to the following table:

return value	version number
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

A return value of -1 means the device query timeout, and a return value of -2 means the system Bluetooth is not turned on.

2.2. Queries whether the controller supports low-power Bluetooth

bool WCHBLEIsLowEnergySupported();
A return value of FALSE indicates that the function is not supported; a return value of TRUE indicates that it is supported.

2.3. Setting up BLE Second Broadcast (secondary advertising)

void WCHBle_Set_Secondary_Advertising(uint32_t phy).
This function can set the second broadcast mode 1M/2M/CODED, which is supported by BLE5.0 and above. The parameter phy means select the mode of second broadcast: 1 means 1M, 2 means 2M, 3 means CODED.

2.4. Setting the BLE transmit-receive (TX/RX) PHY

void WCHBle_Set_PHY_TX_RX(uint32_t phys).

This function can set the transmit-receive PHY, which is supported by BLE5.0 and above.

The parameter phys sets the transmit-receive PHY mode: 1537 for 1M, 7681 for 2M, 32257 for CODED.

2.5. Scanning for BLE devices

int WCHBle_BLE_Scan(int ScanTime, FunDiscoverDeviceInfo Ble_AdvertisingDevice_Info);

This function scans for nearby BLE devices within a specified time period and returns the scanning results (device address, device name, signal strength RSSI) via the scan callback function.

The parameter ScanTime is the set scan time in seconds;

The parameter Ble_AdvertisingDevice_Info is the callback function for the scanning result (device address, device name, signal strength RSSI)

A return value of 1 indicates that the system Bluetooth is not turned on; a return value of 0 indicates a successful scan.

2.6. connected device

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

This function connects to the device by its mac address.

The return value of the function is WCHBLEHANDLE, which will be passed as a parameter in subsequent function calls; the parameter mac is the device address, which can be obtained by scanning the BLE device function;

The parameter connectionstate is the connection callback function to upload the connection state.

2.7. Disconnecting the device

void WCHBle_Disconnect(WCHBLEHANDLE *connection).

This function will disconnect the connected device;

connection is the device connection handle.

2.8. Registering connection disconnection callbacks

void WCHBle_register_on_disconnect(WCHBLEHANDLE *connection, FunDisconnectionStateCallback disconnection_state);

This function registers for a device disconnection event. connection is the device connection handle.

disconnection_state is the disconnection state callback.

2.9. Obtaining Device Services

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

this function will get all the services of the connected device.

The parameter connection is the device connection handle; the parameter Services gets the services of the device;

Parameter Services_Count Get the number of services of the device; the return value of

the function is 0, which means that the service has been successfully acquired;

A function return value of -1 indicates a service failure.

2.10. Getting Device Characteristics

int WCHBle_Discover_Characteristics(WCHBLEHANDLE *connection.

GattCharacteristic *Characteristics, int *Characteristics_Count).

This function is used to get all the characteristics of the connected device. The parameter connection is the device connection handle;

The parameter Characteristics is an array of structures to obtain the characteristics of the device, defined as follows: typedef struct {

handle; uint8_t

handle; uint8_t

properties; uuid_t

uuid; uuid_t

} GattCharacteristic.

where handle is the feature handle and uuid is the feature identifier;

properties is the characteristic value of the feature, the meaning is as follows:

bit0(0x01) is 1 to support broadcast operation, 0 means not support broadcast mode; bit1(0x02) is 1 to support read operation, 0 means not support read operation;

A bit2(0x04) value of 1 indicates that write without response is supported, while a value of 0 indicates that write without response is not supported;

A value of 1 for bit3(0x08) indicates that write with response is supported, while a value of 0 indicates that write with response is not supported.

Answer the write operation;

A bit4(0x10) value of 1 indicates that the notification operation is supported, while a value of 0 indicates that the notification operation is not supported.

Parameter Characteristics_Count Get the number of characteristics;

A return value of 0 means the feature was successfully acquired;

a return value of -1 means the feature was not acquired.

2.11. Write eigenvalues

WCHBle_Write_Characteristic(WCHBleHANDLE)

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

This function is used to write the feature values.

The parameter connection is a device connection handle;

The parameter CharacteristicUUID is the characteristic identification;

The parameter WriteWithResponse is the transmission mode, 0 means no response transmission, 1 means response transmission; the parameter Buffer is the string to be written;

The parameter Buffer_Length is the length of the string to be written; the function returns 0 Success;

A return value of 1 indicates a write failure;

2.12. Read eigenvalues

WCHBle_Read_Char_by_UUID

WCHBle_Read_Char_by_UUID(WCHBleHANDLE *connection, int const int

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

This function is used to read the feature values.
The parameter connection is a device connection handle;
The parameter CharacteristicUUID is the characteristic identifier; the parameter Buffer is a string to store the reading result;
The parameter Buffer_Length is the length of the string to be read; the return value of the function is 0, which means that the reading of the feature is successful;

Other return values indicate a failure to read the feature.

2.13. Registration Notice

void WCHBle_register_notification(WCHBLEHANDLE *connection, FunRegisterNotifyCallback notification_handler);

This function is used to register for notifications and receive data through the notification callback function. The parameter connection is the device connection handle;

The parameter notification_handler is the notification callback function.

2.14. Turn on notifications

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

This function is used to turn on notifications.
The parameter connection is a device connection handle;
The parameter CharacteristicUUID is the characteristic identifier; the return value of the function is 0, which means that the notification was turned on successfully;

The function returns -1 indicating that opening the notification failed.

2.15. Notification of closure

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

This function is used to turn off notifications.
The parameter connection is a device connection handle;
The parameter CharacteristicUUID is the characteristic identifier; the return value of the function is 0, which means that the notification has been turned off successfully;

A function return value of -1 indicates a failure to close the notification.

2.16. Get device MTU

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

This function is used to get the MTU value of the currently connected device. The parameter connection is the device connection handle;
Parameter CharacteristicUUID is the UUID of the characteristic (the characteristic needs to support write without response);
The parameter mtu is the variable that holds the result;

A return value of 0 indicates success in getting the MTU value, while other return values indicate failure.

Third, the introduction of callback functions

There are four types of callback functions: device scan callbacks, connection status callbacks, disconnection status callbacks, and notification callbacks.

3.1. Device Scan Callbacks

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

This function is used to return the scanned BLE device information.

Parameter addr, the address of the scanned device;

Parameter name, name of the scanned device;

Parameter rssi, the signal strength of the scanned device.

3.2. connection state callback

typedef void(*FunConnectionStateCallBack)(WCHBLEHANDLE *connection,int state);

This function is used to return the status of the device connection, and report the current connection status when the connection status changes. The parameter connection is the device connection handle;

Parameter state, 1 means the connection was successful, 0 means the connection failed.

3.3. Disconnect Status Callback

typedef void(*FunDisconnectionStateCallBack)(void *user_data);

This function is used to report a device connection disconnection event.

3.4. notification callback

typedef void (*FunRegisterNotifyCallBack)(const uuid_t *uuid, const uint8_t *data, size_t data_length).

This function is used to receive data from the serial port; the parameter uuid is the feature identification;

Parameter data, received data;

Parameter data_length, the length of the received data.

IV. Introduction to the interface calling order

4.1 Order of invocation I

1. Scanning device WCHBle_BLE_Scan
2. Connect device WCHBle_Connect
3. Register Disconnect Callback WCHBle_register_on_disconnect
4. Get device service WCHBle_Discover_Primary
5. Get device characteristics WCHBle_Discover_Characteristics
6. Register for notification WCHBle_register_notification
7. Open Notification WCHBle_Open_Notification
8. Get device MTU value WCHBle_Get_MTU
9. Write Characteristic WCHBle_Write_Characteristic
10. Read feature value WCHBle_Read_Char_by_UUID
11. Close Notification WCHBle_Close_Notification
12. Disconnect the BLE device WCHBle_Disconnect