TTCIOTSDK

Level: public information

iOS BLE API user Guide

September 19, 2017

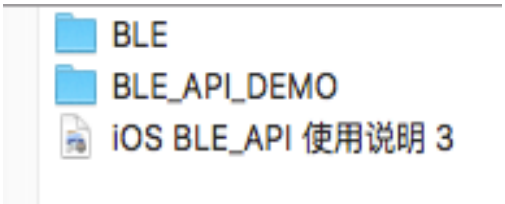
version 1.3

Copyright © ShenZhen ShengRun Technology Co.,Ltd.
All Rights Reserved

Version	Revision date	Revisionist	Reviewers	Modify the content
1.0	2016-04-15	Li Jiong	Zhang Yan	1. iOS BLE API instructions first release
1.1	2017-01-16	Li Jiong	Zhang Yan	1. Add the CC2640 data transceiver related instructions
1.2	2017-03-09	Li Jiong	Zhang Yan	1. Modify the description of the 2541 send data length
1.3	2017-09-01	Li Jiong	Zhang Yan	1. Add the access to the module parameters and set the description; 2. Add the read and write instructions for the data of the custom service

Contents

1.iOS BLE SDK Introduction



1)BLE directory file is the library file need to add.

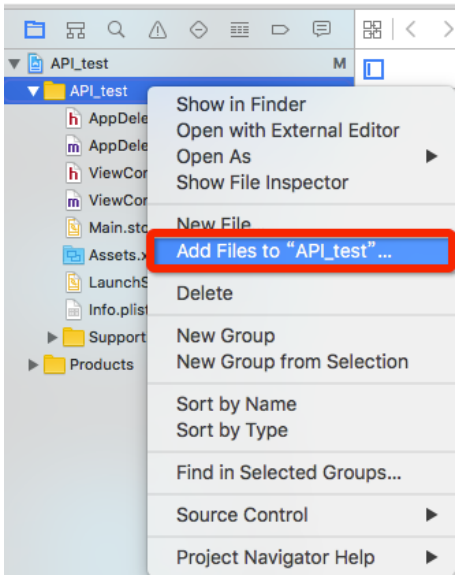
2)BLE_API_DEMO is the sample code.

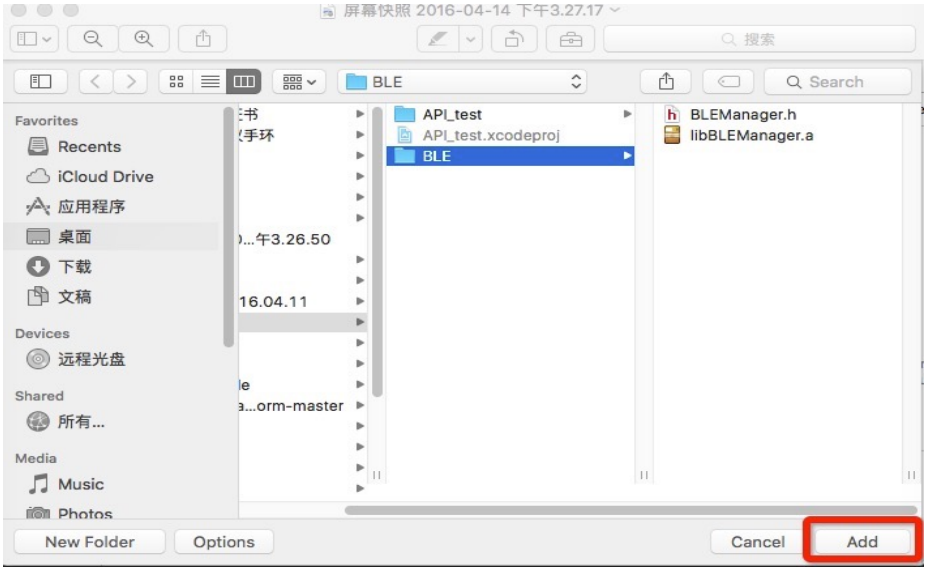
2.Engineering configuration

2.1 Import Bluetooth static files

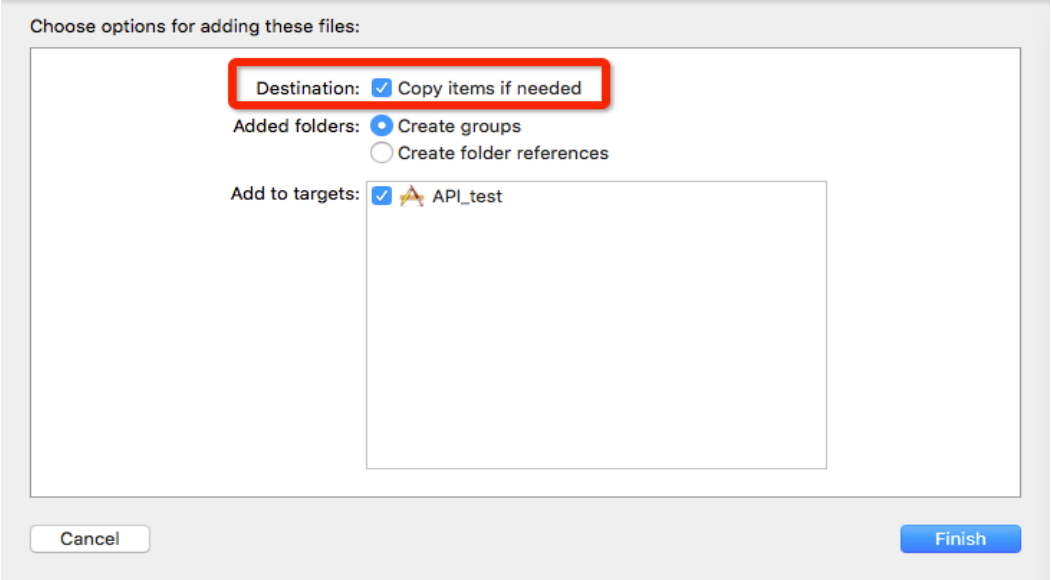
Here are two ways:

Method 1: Add the static file to the project, and then right click on the project, add to the project, as follows:

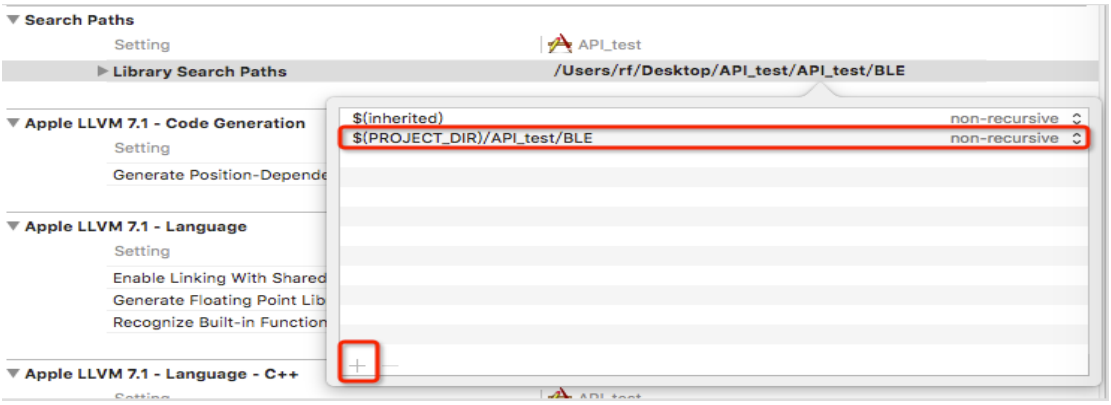


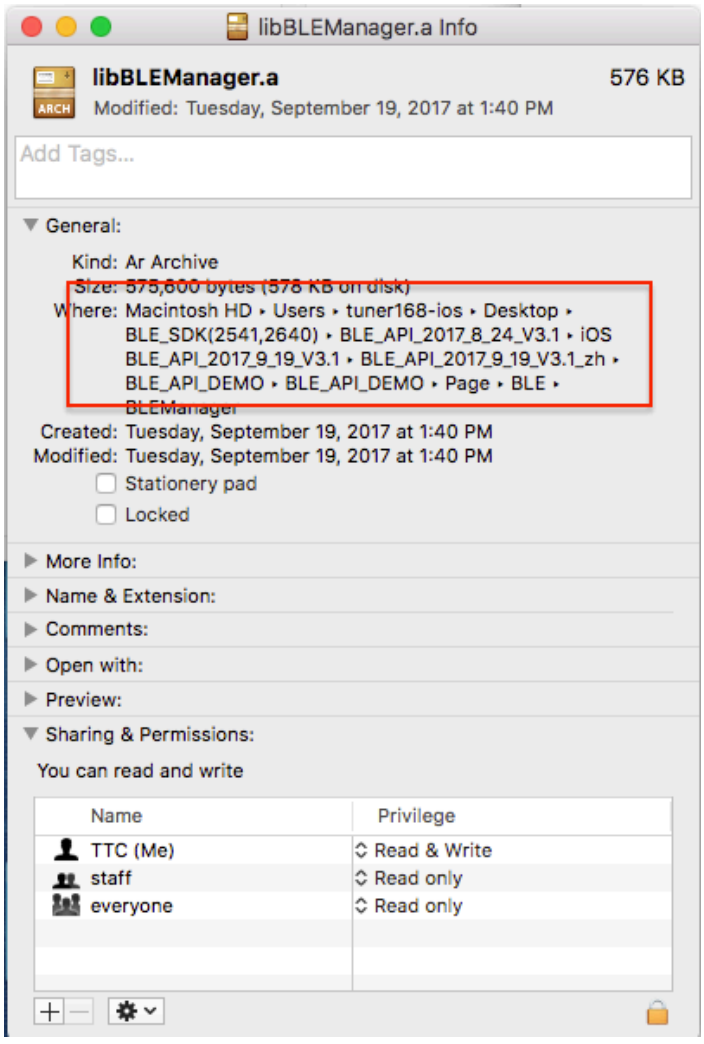


Method 2: directly to the static file into the project, remember to check copy items if needed.



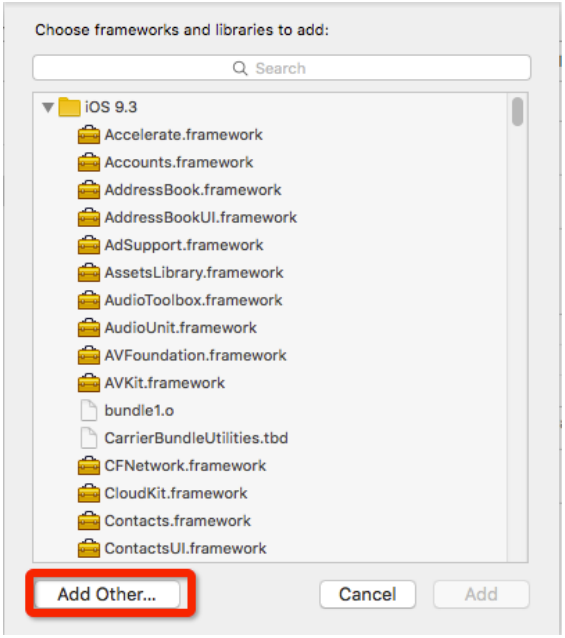
(1) After the success of the import, see the project TARGETS -> Build Search -> Search Paths -> Library Search Paths, as shown in Figure, if there is no static file path, please manually add, show static file introduction, click on the figure '+' Add the file path copy to the following figure.





(2) The static file does not support bitcode, in the project TARGETS -> Build Settings -> Build Options -> Enable Bitcode is set to NO.

(3) Add a Bluetooth framework in the project TARGETS -> Link Binary With Libraries Add CoreBluetooch.framework, until Link Binary With Libraries appears CoreBluetooch.framework; if there is no libBLEManager.a, click “Add Other…” until Link Binary With Libraries appears libBLEManager.a.



(4) If the Xcode prompts you not to find the <DeviceInfo.h> and <oad.h > files during the debugging project, add the two files from the API provided by the company to the project because the contents of the above two files are used in the libBLEManager.a file.

(5) If you need to debug, please debug on the real machine, libBLEManager does not support debugging on the simulator.

3.Sample Code

3.1 Create a BLEManager instance

```
BLEManager *manager = [BLEManager defaultManager];
```

Need to set manager's delegate.

3.2 Scanning Device

```
[manager scanDeviceTime:(NSInteger)];
```

This parameter passes a value of the NSInteger type that indicates the duration of the search. After searching for the device, it calls back the method,

```
- (void)scanDeviceRefrash:(NSMutableArray *)array.
```

The array array holds the DeviceInfo object, which contains the attributes that can be found in the <DeviceInfo.h> file View.

3.3 Connecting Device

- (void)connectToDevice:(CBPeripheral *)device;

Parameters device: the slave object to be connected;

You can get the slave object by - (CBPeripheral *)getDeviceByUUID:(NSString *)uuid; this uuid parameter can get from the DeviceInfo object in the array inside the callback method of the search device;

After the connection, the connection is successful or failed, you can in the following two callbacks which get the answer;

```
/**
 *  Connect device successfully callback method
 *
 *  @param device: Device Object
 *  @param error : Error Message
 */
- (void)connectDeviceSuccess:(CBPeripheral *)device error:(NSError *)error;

/**
 *  Disconnect the device success callback
 *
 *  @param device: vDevice Object
 *  @param error:  Error Message
 */
- (void)didDisconnectDevice:(CBPeripheral *)device error:(NSError *)error;
```

3.4 To send data from the machine

There are two ways to send data, one is encrypted, the other is not encrypted; the default is to send data encryption method, if you need is not encrypted data, please manager's isEncryption attribute is set to NO.

The default data transmission channel is the 1001 channel under the 100-0 service

- (void)sendDataToDevice1:(NSString *)dataStr device:(CBPeripheral *)device;

Parameter 1: the data sent, the data is hexadecimal string (Note: If it is encrypted, the length of the string is 34, if it is not encrypted, the maximum length of the string is 40).

Parameter 2: Slave object.

3.5 Receives data from the slave

(1)If it is active to read the slave data can be

- (void)receiveDeviceDataSuccess_3:(NSData *)data device:(CBPeripheral *)device;

this callback method to get the data.

Parameter 1: Get the data;

Parameter 2: Slave object.

(2) If the slave is active broadcast data, you can

- (void)receiveDeviceDataSuccess_1:(NSData *)data device:(CBPeripheral *) device;

this callback method to the data.

3.6 Read the module parameters

3.6.1 How to Read Module Parameters (Custom Get Data)

(1) Send the data of the corresponding parameter to the 1005 channel according to the underlying specification: for example, to obtain the current device name and send "0e" to the 1005 channel, use the following method:

- (void)sendDataToDevice5:(NSString *)dataStr device:(CBPeripheral *) device;

(2) The next step to read 1004 channel data, using the following method

- (void)readDataWithDevice4:(CBPeripheral *)device;

Then, you can parse the data in the following callback method by using the underlying description:

- (void)bleManagerPeripheral:(CBPeripheral*)peripheral didUpdateValueForCharacteristic:(CBCharacteristic *)characteristic error:(NSError *)error;

3.6.2 How to read module parameters (acquiescence fetch data)

In our API, we have encapsulated the process for obtaining the corresponding parameters, such as the method for obtaining the name of the device as

follows:

- (void)readDeviceSettingName:(CBPeripheral *)device;

Then, we can get the required data in the corresponding callback method, which is parsed as the final data. The callback method after obtaining the device name is as follows:

- (void)receiveDeviceSettingName:(NSString *)name device:(CBPeripheral *)device;

We also encapsulate the methods for obtaining other parameters, which can be found in the API. As long as you follow the process above you can get the data you want.

3.7 Set the module parameters

3.7.1 How to set module parameters (custom setting data)

(1) Send the data of the corresponding parameter to the 1005 channel according to the underlying specification: for example, to obtain the current device name and send "0e" to the 1005 channel, use the following method:

- (void)sendDataToDevice5:(NSString *)dataStr device:(CBPeripheral *)device;

(2) The next step to write data to the 1003 channel, using the following method

- (void)sendDataToDevice3:(NSString *)dataStr device:(CBPeripheral *)device;

3.7.2 How to set module parameters (default setting data)

As with the data, we also encapsulate all the processes that set the data, such as the customer wants to set the module name and send the name data to the device using the following method:

- (void)setDeviceName:(NSString *)name device:(CBPeripheral *)device;

If you want to set the name of the device to abcd, you only need to pass the name parameter of the above method to @ "abcd".

3.8 How to operate the device corresponding to the channel notify

Use the following method to operate the notify state of the device's corresponding channel:

- (void)notification:(UInt16)serviceUUID characteristicUUID:(UInt16)cUUI-

D peripheral:(CBPeripheral *)device enableState:(BOOL)isEnable;

Parameter 1 is the service UUID of the device, parameter 2 is the UUID of the corresponding channel, parameter 3 is the device object, parameter 4 is the value of a BOOL type, the value is YES, and the channel is opened.

For example, to open the 10002 service under the 1002 channel notify,we should follow the following example code to operate:

[BLEManager defaultManager] notification:0x1000 characteristicUUID: 0x1002 peripheral:device enableState:YES];

Whether the set is successful, we can make a judgment in the following callback method:

- (void) bleManagerPeripheral:(CBPeripheral *)peripheral didUpdateNotificationStateForCharacteristic:(CBCharacteristic *)characteristic error:(NSError *)error;

3.9 How to read and write data for channels under non 1000 service

3.9.1 How to write data for channels under non 1000 service

Use the following method:

- (void)bleManagerPeripheral:(CBPeripheral *)peripheral writeValue:(NSString *)string serviceUUID:(UInt16)serviceUUID characteruisticUUID:(UInt16)characteruisticUUID encryption:(BOOL)encryption response:(BOOL)response

Parameter 1 is the device object, parameter 2 is the written data, parameter 3 is the UUID of the device service, parameter 4 is the UUID of the corresponding channel, parameter 5 is the encryption, parameter 6 is the way to write data, YES is the response write (CBCharacteristicWriteResponse), for NO stands for no response to write(CBCharacteristicWriteWithoutResponse).

For example, to the 2001 channel under the 2000 service to write 123456 this data, the data for the encryption (this is based on whether the module is encrypted), response parameter is set to NO, the sample code is as follows:

[BLEManager DefaultManager] bleManagerPeripheral:device writeValue: @"123456" serviceUUID:0x2000 characteruisticUUID:0x2001 encryption: YES response:NO];

3.9.2 How to read data for channels under non 1000 service

As described in Section 3.8, open the corresponding channel notify, and t-

hen when the device has data sent to the APP, we can in the following callback method to get the data inside:

- (void)bleManagerPeripheral:(CBPeripheral *)peripheral didUpdateValueForCharacteristic:(CBCharacteristic *)characteristic error:(NSError *)error;

The obtained data is the value attribute value of the characteristic object.

4. Contact us

深圳市昇润科技有限公司
ShenZhen ShengRun Technology Co.,Ltd.
Tel: 0755-86233846 Fax: 0755-82970906
Official website：www.tuner168.com
Alibaba website：http://shop1439435278127.1688.com
E-mail: marketing@tuner168.com
Address: Room 602, B Block of Jingu Pioneer Park, Longzhu 4th Road,Xili Town,Nanshan District, Shenzhen