

## Prefix

### How to quickly develop new peripherals for smart phones with lower cost?

- About the application of Bluetooth Low Energy technology in smart mobile devices



The birth of USB protocol has made peripherals of PC spring out. It is the same case with the latest opened Bluetooth Low Energy (BLE) technology. BLE technology makes it possible that electronic devices bridge smart phones. Compared with those wireless transmission technologies such as WiFi, Bluetooth 2.0, BLE technology can get lower energy consumption, faster connection, longer distance of communication range, and other advantages. So it brings wider development scopes for smart phone peripherals.

As a member of BT-SIG, Texas Instruments (TI) introduced CC254x series of SOC low-power Bluetooth transceiver. With classical 51 kernel inside, its most prominent features are rich periphery (including 21 IO, UART, SPI, USB2.0, PWM, ADC, analog comparator, op-amp), super wide working voltage (2v – 3.6v), extreme low energy consumption (<0.4 $\mu$ A), and very short wake-up delay (4 $\mu$ s)

In order to facilitate the transplant and use of BLE application technology in every industry, RF-Star, the strategic partner of TI China in the field of wireless, has introduced low-power Bluetooth transparent transmission modules, 2 among which has been certified with BQB (EPL), FCC, CE and RoHS by BT-SIG.

### **RF-BM-S01** (full pin):

Please refer to: [https://www.bluetooth.org/tpg/EPL\\_Detail.cfm?ProductID=27655](https://www.bluetooth.org/tpg/EPL_Detail.cfm?ProductID=27655),

**RF-BM-S02** (compact size, non full pin):

Please refer to: [https://www.bluetooth.org/tpg/EPL\\_Detail.cfm?ProductID=34109](https://www.bluetooth.org/tpg/EPL_Detail.cfm?ProductID=34109)

(Other modules are under certification process).

Modules, working as a bridge between smart phones and peripherals, make it simple to develop host applications. In the bridge mode (with USART, users' products, embedded with the modules, can communicate with mobile devices (Bluetooth 4.0 supported) with ease. It realizes the super smart control and management. While in the Direct-Drive mode, users can design quickly solutions and even products, in the way of direct expansion of module's periphery, which enables the introduction of unique and personalized peripherals for mobile devices with lower cost and high efficiency.

**RF-BM-S01** BLE module applies CC2540 from TI as core processor. The module runs at 2.4GHz ISM band, with GFSK (Gaussian Frequency Shift Keying) modulation scheme. It has 40 channels (channel space at 2MHz), among which there are 3 fixed radio channels and 37 data channels of adaptive automatic frequency hopping. Physical layers can be combined with classic Bluetooth RF to form a dual mode device. And 2 MHz's channel space can prevent interference from adjacent channels better. Besides, it has a wide output power adjustment (-23 dBm ~ 4dBm) and a high gain receiving sensitivity of -93 dBm.

The module is designed to quickly connect electronic products with smart mobile devices, and can be widely used in various electronic devices, such as instruments, logistics tracking, healthcare, smart home, sports metering, automotive electronics, toys, and etc. With Android 4.3 smart devices integrated with BLE technology, it will be a trend that BLE will be the standard configuration of smart phones. And the market demand on smart phone peripherals will increase geometrically. With this module, users can integrate their existing solutions or products in the shortest development cycle, to occupy the market in the fastest speed, and to empower their company's growth with the strength of new technology.

## Overview

The module can work in bridge mode (transparent transmission mode) and direct-drive mode.

After being started, the module can broadcast automatically. Smart phones with specific application running will scan and pair with it. When connection success, the smart phone can monitor and control the module through Bluetooth protocol.

In bridge mode, user's MCU can communicate with the mobile device in two-way through module's UART. Users can also manage and control certain communication parameters through specific UART AT commands.

The detailed meaning of the user data is defined by the up-application. Mobile devices can write to the module through the APP. And the data written will be sent to the user's MCU through UART. Then the module will transmit the data package from user MCU to the mobile devices automatically.

In the development under this mode, the user need to undertake the code design for MCU code, and the code design of APP for smart mobile terminals.

**In Direct-Drive mode**, users take simple periphery expansion to the module. And APP drives the module directly through BLE protocol, to implement the monitoring and control of the module by smart mobile devices. In this mode, users only need to do the code design for smart mobile terminals.

#### **Features:**

1. Easy to use, no need of any experience of Bluetooth protocol stack application
2. UART design for user interface, full-duplex bi-directional communication, and supporting the minimum baud rate of 4800 bps;
3. Supporting bridge mode (USART transparent transmission), and direct-drive mode (no additional MCU needed)
4. Default connection interval of 20ms, which makes quick connection;
5. Supporting software reset module by AT command, and access to the MAC address;
6. Supporting the adjustment of Bluetooth connection interval by AT command, and the control of different forwarding rates (dynamic power adjustment);
7. Supporting the adjustment of the transmission power by AT command, the change of broadcasting interval, the customization of broadcasting data, the customization of equipment UDID, the setting of data delay (preparation time of user MCU USART receiving), the change of the USART baud rate, and the change of the module names (all settings can be saved after power-off);
8. The length of the UART packets can be any below or equal to the arbitrary length of 200 byte (large packet automatic distribution);
9. High-speed transparent transmission forward rate maximum up to 4 K/S and the stable rate to be at 2.5 K to 2.5 K (IO5, IO6);
10. Supporting the change of module name by APP in mobile devices, the change of UART baud rate and product UDID, the customization of broadcasting contents and cycle (all settings can be saved after power-off);
11. Supporting the remote reset of module by APP in mobile devices, and the setting of transmission power;
12. Supporting the adjustment of Bluetooth connection interval by APP in mobile devices but the setting cannot be saved after power-off (dynamic power adjustment);
13. All IO port expansion, including debug ports
14. Supporting the connection status and the flexible configuration of broadcasting status prompt pin / general IO;
15. 6 two-way programmable IO port, input check triggered by external interrupt, and low power operation (applied in trigger alarm, lighting control, remote control toys, and various i/o switch);
16. 2 programmable single timing / cycling timing output port (applied in smart timing schedule);
17. 2 ADC inputs (14 bit), EN/BAN, free configuration of sampling cycle (applied in temperature/humidity metering, photometry, & etc.);
18. 4 programmable PWM outputs (120 hz) (applied in dimming control);
19. Module-side RSSI continuous acquisition, APP readable and auto-notifying, EN/BAN, free setting of acquisition frequency (applied in finder, anti-loss and alarm);

20. Supporting battery reading and prompt, able to auto upload (notification of remaining battery);
21. Supporting the password setting, modifying and restoring for anti-hijacking, preventing from malicious connection from a third party. Also the notification of independent crypto-operation result to ease the APP programming;
22. Supporting restoring factory settings by 5-second long press and the APP remote recovery;
23. Supporting the custom of PWM output initialization status (low, full, PWM output status value before power-off);
24. Supporting the custom of PWM frequency ( $61.036 \text{ Hz} \leq f \leq 8 \text{ kHz}$ , default 120 Hz);
25. Real-time system status prompt in broadcasting contents, including battery charge, custom UDID, current output value of 4 PWMs or collection value of 2 ADCs , the current state of IO, and etc.;(suitable for broadcast applications);
26. 2 level pulse-width counting,  $0 \sim 0 \text{ XFFFFFFF ms}$  (about 49.7 days);
27. Supporting internal RTC, which can be synchronized any time from APP side;
28. Supporting 6 IO and 4 PWM timing control (default setting OFF);
29. 4 PWMs to support gradient mode (suitable for dimming control);
30. Supporting the saving of IO port configuration and output status, and the customization of the Default initialization status;
31. Supporting the shallow recovery and depth recovery modes, which can recover user data flexibly while reserve the essential configuration of the product;
32. Supporting string prompts of Bluetooth connection status from TX UART (connection, normal disconnection and timeout disconnection) ;
33. Supporting low-level-enabled mode and pulse-width-enabled mode, and the remote shutdown;
34. Supporting auto shutdown after 30 seconds without connection in the pulse-enabled mode;
35. Supporting timeout (break) prompt by square wave alarm in the pulse-enabled mode;
36. Extremely low power in standby mode (0.4 uA of current as per TI Official for CC2540), and the measured power consumption data as follows:

Event	Average current (integral computed*1)	Average current (ammeter measured*2)	Duration	Testing Conditions/Remarks
Sleeping	0.35uA	0.3-0.4uA	—	EN dangling
Broadcasting	202uA	0.14~0.54mA	3.85ms	broadcasting cycle is 250 ms
Connection Event	243uA	0.41 mA	2.25ms	Connection cycle is 100 ms
Single BLE Data Receive Event	332uA	0.65 mA	3.0ms	(20bytes,10 times per second)
Module receiving data and send through	497uA	2.68mA	5.1ms	(20bytes,10 times per second)

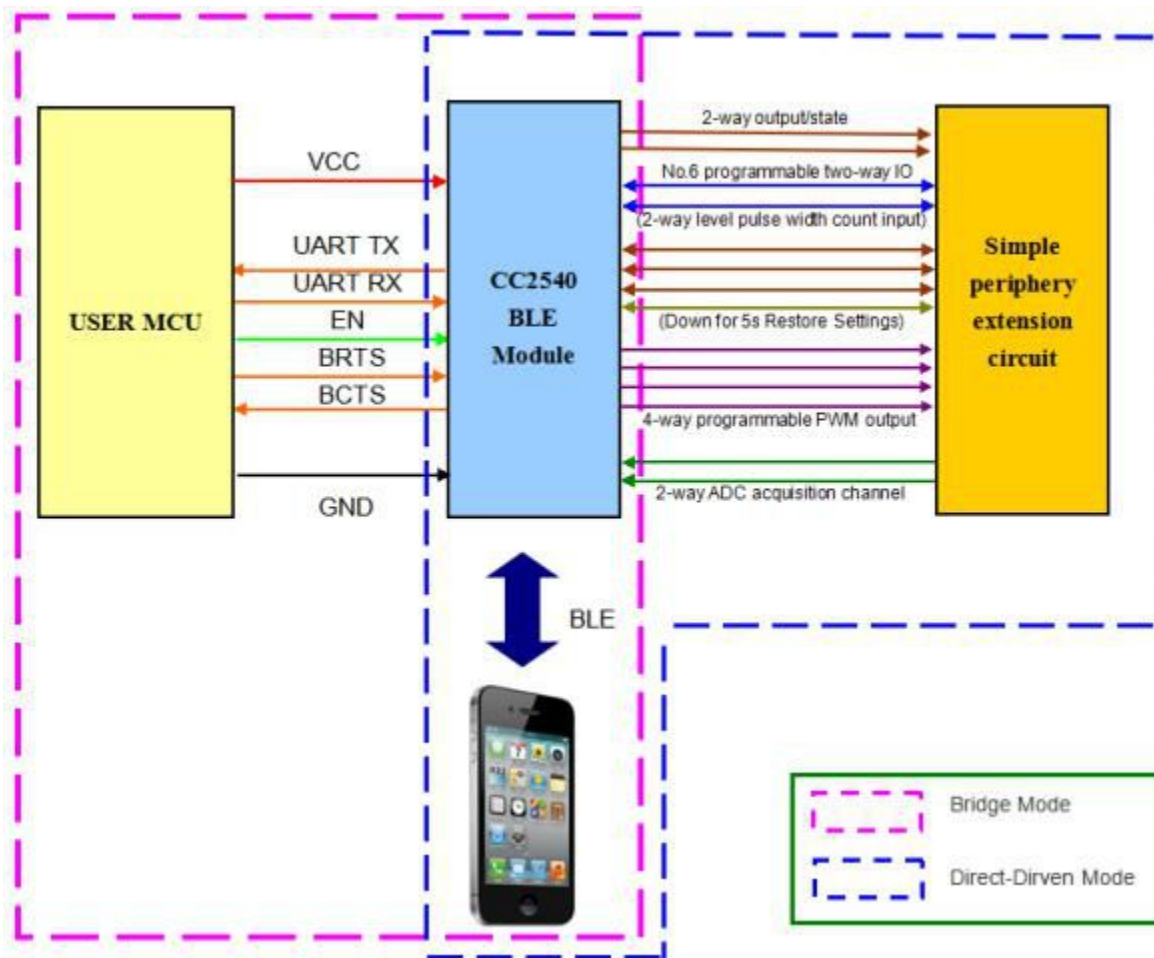
USART				
Single BLE data transmission event	342uA	0.69mA	3.2ms	(20bytes,10 times per second)

\*1 Note: The official test method: connect in series a 10Ω resistor in the circuit with power supply, intercept voltage waveform with oscilloscope and perform integration

\*2 Note: Multimeter test method: connect multimeter (set at uA or mA level) in series between the battery and the module to check the value displayed, with the test voltage of 3.07 V

Above is the measured sampling data of module RF-CC2540A1 and for reference only. If lower power consumption is expected, connection interval or broadcast cycle can be appropriately increased, as shown in the module parameter settings and the USART AT demands in related chapters.

## Schematic Diagram of Working Mode



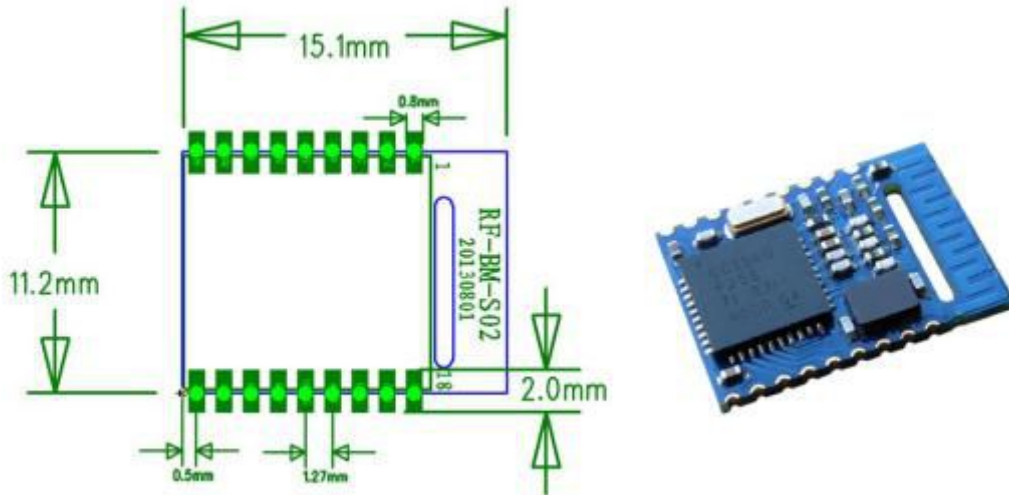
Module Bridge mode and Direct-Driven mode schematic diagram

Note: In order to avoid the output level difference of user MCU's IO and module IO, which will result

to high current, a small isolation resistor is suggested to be connected in series in the output signal line TX, BCTS.

## Packaging size and Pins

### RF-CC2540A1(Double-sided board process)



RF-BM-S02 (BQB Certification)

Module Pin No.	Module Pin Name	Chip Pin Name	I/O	Description
Pin1	GND	GND	-	GND
Pin2	VCC	VCC	-	power supply 2V-3.6V
Pin3	IO7	P2.2	O	Output Port (Timing to Flip) / Sleep status indication
Pin4	IO6	P2.1	O	Output Port Timing Flip/Connect status indication( <i>Prompt when low level or square wave, see the section "Module Parameter Settings"</i> )
Pin5	RES	RST	I	Reset, effect at low level
Pin6	EN	P2.0	I	Module-enabled control line ( <i>level trigger mode as default</i> ) <ul style="list-style-type: none"> <li>➤ level trigger mode, Active when low level, with internal pull-up.</li> <li>0: Module starting to broadcast, until connected to the mobile device</li> <li>1: Entering sleep mode immediately, regardless the current status (0.4uA)</li> <li>➤ Pulse triggering mode - Each time when receiving a pulse, module will shift between boot-up (broadcasting, allowing to be found and connected) and shutdown (complete sleep mode) .</li> </ul>

				<i>(About the mode switching, please refer to the related sections of <b>Module Parameter Setting</b>.)</i>
Pin7	IO5	P1.7	I/O	<ul style="list-style-type: none"> <li>➤ Programmable two-way IO port, which could be used for input or output through the BLE protocol</li> <li>➤ Could be a level PWM count input terminal when used as input</li> </ul>
Pin8	U+	USB+	I/O	Out-pin of CC2540 to USB+, not used
Pin9	U-	USB-	I/O	Out-pin of CC2540 to USB-, not used
Pin10	REST ORE / IO0	P1.2	I/O	<p>Factory setting restore trigger or programmable two-way IO</p> <ul style="list-style-type: none"> <li>➤ <b>Within 30 seconds after power-on</b>, keep this pin at low level for <b>5 s</b>, the system can restore partially (shallow recovery); if keeping more than <b>20 s</b> the system will restore all (deep recovery).</li> </ul> <p>(see section "System Reset and Recovery")</p> <ul style="list-style-type: none"> <li>➤ 30 seconds after power on, could be used as ordinary IO and be set through BLE protocol (see the programmable IO (8) "service UUID: 0 xfff0"</li> </ul>
Pin11	PWM1	P1.1	O	PWM output channel 1
Pin12	PWM3	P0.7	O	PWM output channel 3
Pin13	PWM4	P0.6	O	PWM output channel 4
Pin14	BRTS	P0.5	I	<p>As the data sending requests (for module wake-up)</p> <p>0: Host has data to send, and module will wait for data transmission from the host so will not sleep</p> <p>1. Host has no data to send, or data has been sent. So the value of the signal should be set at "1".</p>
Pin15	BCTS	P0.4	O	<p>Data input signal (for host wake-up, optional)</p> <p>Module has data to send, and the host will receive the data.</p> <p>1: Module has no data to send, or data has been sent, and the value of the signal will be set at "1".</p>
Pin16	TX	P0.3	O	Module UART TXD end
Pin17	RX	P0.2	I	Module UART RXD end
Pin18	ADC1	P0.1	I	Analog acquisition, channel 1

Note: BM - S02 is a compact size edition, so part of the IO are not pinned, the corresponding function cannot be used.

## UART transparent transmission(Bridge mode)

The bridge mode means to set up two-way communication between user CPU and mobile devices by connecting the module with user CPU through UART. Users can reset UART baud rate and BLE connection interval, using the specified AT commands (see behind the section "UART AT command"). The module will have different data TX & RX capability, as per different UART baud rates and BLE connection intervals. Considering the use of low-speed CPU, the default baud rate is set at 9600 BPS. In the application where there is a large amount of data transmission, or there is high real-time demand, it is suggested to set the UART baud rate at the high speed of 115200 BPS.

Settings can be saved and kept after power-off..

**When the BLE connection interval is 20 ms and the UART baud rate is at 115200 Bps**, the module has the highest transmit ability in theory (4 k/s). Given this configuration in the level enabled mode as an example, UART transparent transmission agreement will be detailed introduced as below.

The module can transmit through UART maximum 200 byte packets at one time. According to the packet size, the packet will be sub-packed automatically and sent, with a maximum load of 20 bytes for each wireless sub-packet. Data packets from mobile devices to the module must be sub-packed by their own (into 1 – 20 bytes/packet) before sending. The module will forward them to the host serial RXD end in turn, when receiving the packets.

1. The UART hardware protocol: 115200 BPS, 8, no parity, 1 stop bit.
2. When EN for high level, the bluetooth module is in full sleep mode. When EN set low, the module will start broadcasting at the interval of 200 ms, until it pairs with mobile devices. When EN jumps from low to high, the module will enter into sleep mode immediately, regardless of current status.
3. After the module is connected, BRTS needs to be pulled low if the host (MCU) has data to send to the BLE module, and the data transmission can be started around 100 us afterwards. BRTS should be raised high by the host after transmission finishes and make the module exit the serial RX mode. Pay attention to confirm that UART data transmission has been completely finished before raising BRTS. Otherwise there will be data truncation.
4. When there is data upload request, the module will set BCTS low, until data transmission finishes. The transmission can start at least 500 us afterwards. And this delay can be configured through the AT command (see in section "serial AT command"). BCTS will be set high by the module when data transmission is finished.
5. If the host BRTS being kept a low level, the Bluetooth module will always be in UART RX mode and the power consumption will be high.
6. After the module is connected, a string of "TTM: OK \r \n \0" will be prompted from TX. And it can be certain by the string if the normal forwarding operation is available. Of course the connection status prompt pin can be used instead. Also the connection can be confirmed by sending a specific confirmation string to the module from mobile devices. When APP actively disconnect the module, there will be a prompt of string "TTM: DISCONNECT \r \n \0" from TX. If the disconnection is abnormal, the string prompt will be "TTM: DISCONNECT FOR a TIMEOUT \r \n \0".
7. **The default Bluetooth connection interval is 20 ms.** If low-speed forwarding mode is needed for saving power, connection interval must be adjusted by AT command (longest connection interval to be 2000 ms). Each interval transmit maximum 80 bytes. Set the connection interval as V (unit: ms), and highest forwarding rate per second as V (byte/s), then their relation is as follows:



$$V = 80 \cdot 1000 / T \quad (V \text{ is only relevant with } T)$$

If the Bluetooth connection interval of the module is 20 ms, and each interval can transmit maximum 80 bytes, the theoretical maximum transmission capacity (forwarding rate) will be  $80 \cdot 50 = 4 \text{ k}$  byte/s. Tests have shown that the packet loss is very little when forwarding rate under 2 K/s. For safety's sake, **it is suggested to do checksum and retransmission processing in the up-layer, no matter for high or low speed forwarding applications.**

8. Below is an example of the communication with 20 ms connection interval. Configuration can be set as your own. But the lower the forwarding rate  $V_0$ , the less packet leakage.

Communication reference model	BLE Connection interval	Highest theoretical forwarding capacity $V$ (byte/s) $V = 80 \cdot 1000 / T$	Serial packet length	UART contract interval $TS$ (ms) When $L < 80$ , $TS \geq T$ When $80 < L < 160$ , $TS \geq T \cdot 2$ When $160 < L < 200$ , $TS \geq T \cdot 3$	Actual forward rate $V_0$ (byte/s) $V_0 = L \cdot 1000 / TS$	Remarks
1	20	4K	80	$TS \geq T$ , if $TS = 20\text{ms}$	$80 \cdot 1000 / 20 = 4\text{K}$	TS small, not recommended
2	20	4K	200	$TS \geq T \cdot 3$ , if $TS = 70\text{ms}$	$200 \cdot 1000 / 70 = 2.8\text{K}$	
3	20	4K	200	$TS \geq T \cdot 3$ , if $TS = 80\text{ms}$	$200 \cdot 1000 / 80 = 2.5\text{K}$	
4	20	4K	80	$TS \geq T$ , if $TS = 35\text{ms}$	$80 \cdot 1000 / 30 = 2.6\text{K}$	
5	20	4K	70	$TS \geq T$ , if $TS = 30\text{ms}$	$70 \cdot 1000 / 30 = 2.3\text{K}$	
6	20	4K	60	$TS \geq T$ , if $TS = 30\text{ms}$	$60 \cdot 1000 / 30 = 2\text{K}$	
7	20	4K	40	$TS \geq T$ , if $TS = 30\text{ms}$	$40 \cdot 1000 / 30 = 1.3\text{K}$	
8	20	4K	20	$TS \geq T$ , if $TS = 30\text{ms}$	$20 \cdot 1000 / 30 = 666\text{byte}$	

Note: Specific communication mode can be designed according to the practical application. Serial packet length can be designed in between 80 and 200 bytes (large packet transmission). As per the BLE protocol there is the following relations:

**When  $L < 80$ ,  $TS \geq T$ ;**

**When  $80 < L < 160$ ,  $TS \geq T \cdot 2$ ;**

**When  $160 < L < 200$ ,  $TS \geq T \cdot 3$ ;**

Forwarding modes that complies with abovesaid conditions is safe generally. But among them, when setting  $TS = T$ ,  $T * TS = 2$  or  $TS = T * 3$ , it is workable but not recommended, as the packet loss is relatively high and checksum retransmission mechanism must be added. In other words, when a UART packet is as big as  $80 \text{ byte} < L < 200 \text{ byte}$ , serial data can be sent to the module for one time, but some time needs to be spared for data transmission. Otherwise there will be a rear-end data collision. For example, when the connection interval  $T = 20 \text{ ms}$ ,  $3 = T * TS$  must be greater than  $60 \text{ ms}$ , if the serial packet length  $L = 200$ . So setting  $TS = 70 \text{ ms}$  is a logical choice.

9. The size of the serial data packets can be various and the length can be any value less than 200 bytes, as long as the abovesaid conditions are met. meet the above conditions. But in order to utilize the communications payload in maximum efficiency, while to avoid communication running in full capacity, it is recommended to use serial data packets of **20,40, or 60** bytes in length, and interval between packets is made more than 20 ms.

Note: Test show that in IOS, calling the writing function to Characteristic with the parameter **CBCharacteristicWriteWithResponse** (writing mode with response) will reduce partially the forwarding efficiency, but the correctness of a single packet will be ensured. While with the parameter **CBCharacteristicWriteWithoutResponse** (writing mode without response), the forwarding efficiency will be increased, but the correctness of data packet needs to be checked by APP in up layer.

## UART AT Command

Strings starting with "TTM" will be regarded as AT commands to be parsed and executed. **and will return exactly the same from the UART** Afterwards the execution result will be output (ie. TTM: OK \r \n \ "0" or "TTM: ERP \r \n \ 0", etc. **Serial data packets which do not start with "TTM" will be regarded as transparent transmission data.**

### Connection interval setting

Input the following string to the UART RX to set the BLE connection interval:

```
"TTM:CIT-Xms"
```

Where X = "20", "50", "100", "200", "300", "400", "500", "1000", "1500", or "2000" (ms). After the command is executed, the following confirmations will be got from UART TX:

```
"TTM: TIMEOUT \r \n \ 0" (means timeout and the change failed);
```

```
"TTM: OK \r \n \ 0" (means the change is successful and the new connection interval is applied);
```

**The success of the connection interval setting depends on the constraints of connection intervals by mobile devices.** The maximum connection intervals also vary in different version of iOS.

Tests with iPhone 4 s (iOS 5.1.1) shows the fastest is 20 ms and the slowest is 2 s. On the other hand, due to the BLE protocol internal mechanism, execution efficiency of this command will be different with different connection intervals.

In iOS5.1.1, it takes maximum around 100 s, changing from the current connection interval of 2000 ms (max. 2000 ms) to other connection intervals. While the execution will be fast when executing this AT command in other high-frequency connection intervals.

Note: the connection interval setting cannot be saved after power-off. And command of change is only effective when the connection is successful.

## Module Renaming

Input the following string to the UART RX to rename the module (length of name should not exceed 16 bytes).

"TTM:REN-" + Name

Also confirmation of "TTM: OK \r \n \0" will be received from TX. And if the command format incorrect, the string as follows will be returned:

"TTM:ERP\r\n\0"

Test shows that device name can be changed immediately in iOS6 and above versions and can immediately, but not in iOS5. The name can be saved after power-off.

## Baud rate setting

Input the following string to the UART RX (parameter after "-" being the new baud rate):

"TTM:BPS-115200"

Afterwards confirmation string of "TTM: OK \r \n \0" will be received from TX. If the value set is not in the options, or the command format is incorrect, the string as follows will be returned:

"TTM:ERP\r\n\0"

Test shows that in iOS 5 the baud rate cannot be changed, but can be changed immediately in iOS6 and above versions. Users can set through PC, or through the BLE APP interface of mobile devices. (See the "module parameter Settings " 【service UUID: 0 xff90】 ")

## Acquiring physical address MAC

Input the following string to a UART RX:

"TTM:MAC-?"

And the following string will be received from TX:

"TTM:MAC-xxxxxxxxxxxx\r\n\0"

"xxxxxxxxxxxx" after "-" is the bluetooth address in 6 bytes.

### ➤ Module resetting

Input the following string to UART RX will force the module to be soft-reset once:

"TTM:RST-SYSTEMRESET"

### ➤ Broadcast cycle setting

Input the following string to UART RX, to set the broadcast cycle of the module, T = X \* 100 ms

"TTM:ADP-(X)"

Where X = "2", "5", "10", "15", "20", "25", "30", "40" or "50". Confirmation string of "TTM: OK \r \n \0" will be received from TX. If the command format incorrect, the following string will be returned:

"TTM: ERP\r\n\0"

Broadcast cycle setting can be saved after power-off. After the module is rebooted, the module will

broadcast as per the new broadcast cycle.

### **Additional customized contents of broadcast**

Input the following string to the UART RX to customize broadcast contents

"**TTM:ADD-**" + Data

Where "Data" is the additional data ready to be broadcasted ( $0 < \text{Length} \leq 16$  bytes). The confirmation string of "**TTM: OK \r \n \0**" will be received from TX. If the command format is incorrect, the following string will be returned :

"**TTM:ERP\r\n\0**"

It takes immediate effect when command is executed. Certain customized contents can be broadcasted in this way. And the data can be saved after power-off. If setting all 16 bit data as 0, customized broadcast data will not be used. Instead, the default broadcast contents are applied.

### **Defining product identification code**

Input the following string to the UART RX to define product identification code:

"**TTM:PID-**" + Data

where "Data" is for a two-byte product identification code (ranging from 0x0000 range to 0xFFFF and length =2). The confirmation string of "**TTM: OK \r \n \0**" will be received from TX. If the command format is incorrect, the following string will be returned:

"**TTM:ERP\r\n\0**"

This identification code can be saved after power-off. It will show in the broadcasting, and can be used to filter devices or to determine if it is a specific product.

### **Transmission power setting**

Input the following string to the UART RX to set the corresponding transmission power (in dBm).

"**TTM:TPL-(X)**"

Where X = "+ 4", "0", "6", or "- 23". The confirmation string of "**TTM: OK \r \n \0**" will be received from TX. And the module will immediately communicate with the new transmission power. If the command format is incorrect, the following string will be returned:

"**TTM:ERP\r\n\0**"

**Note: this setting cannot be saved when power-off.**

### **Data delay setting**

Input the following string to the UART RX to set the delay from when BCTS outputs low to when UART TX outputs data (in ms)

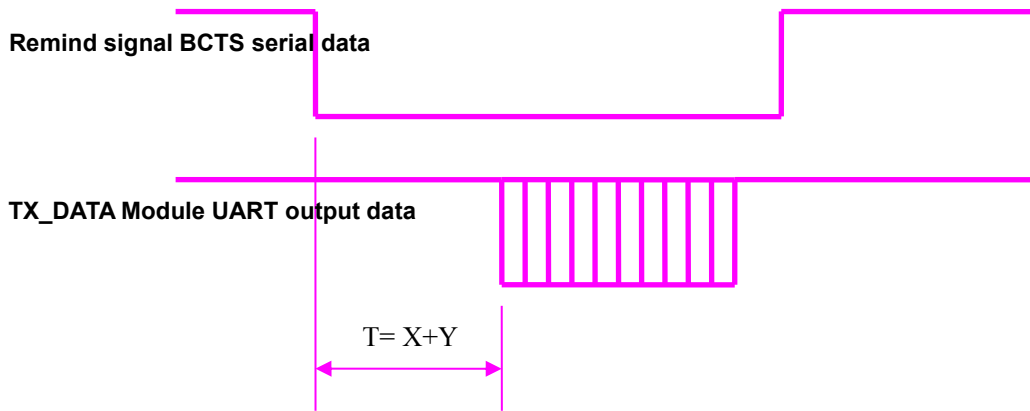
"**TTM:CDL-Xms**"

Where X = "0", "2", "5", "10", "15", "20", or "25". The confirmation string of "**TTM: OK \r \n \0**" will be received from TX. If the command format is incorrect, the following string will be returned:

"**TTM:ERP\r\n\0**"

To make the user CPU have enough time to be waken up from sleep mode and ready to receive data, the module provides this delay (X) setting. The BRTS will be set low before there is data to be

sent through the module's UART. While the delay from when BRTS is set low till when the module TX outputs data will be set by this parameter. The actual delay (T) will be  $T = (X + Y)$  ms, if minimum delay is not less than X, while  $500 \text{ us} < Y < 1 \text{ ms}$ . This setting can be saved after power-off.



**Module uart output data delay setting schemes**

**AT command list**

AT command format	Saved after Power-off	Parameter Description	Possible response	Meaning
"TTM:CIT-Xms" ( Valid only when connection is successful)	no	X="20", "50", "100", "200", "300", "400", "500", "1000", "1500", "2000" Set the BLE connection interval, (in ms)	"TTM:TIMEOUT\r\n\r\n0" " "TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	TIMEOUT setting Setting successful Parameter Error
"TTM:REN-"+ Name	yes	Name,New module name, with length not exceeding 15 bytes.	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Setting successful parameter error
"TTM:BPS-X"	YES	X="4800", "9600", "19200", "38400", "57600", "115200" Set the baud rate	"TTM:BPS SET AFTER 2S ... \r\n\r\n0" "TTM:ERP\r\n\r\n0"	Setting successful and new baud rate will be applied in two seconds Parameter Error
"TTM:MAC-?"	—	acquire MAC address	"TTM:MAC-xxxxxxx xxxx" xxxxxxxxxxxx for module MAC address	Return with MAC address
"TTM:RST-SYSTEMRESET"	—	Reset the module	NO	Resetting the module

"TTM:ADP-(X)"	YES	X = "2", "5", "10", "15", "20", "25", "30", "40", "50"  Set the appropriate broadcast cycle T = X * 100ms  X = "2", "5", "10", "15", "20", "25", "30", "40", "50" set the broadcast cycle T = X * 100 ms	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Setting up the broadcast cycle.(ie.If the parameter set to "5", the cycle will be 500 ms)
"TTM:ADD-Data"	YES	Data for customized broadcast Data, and length L <= 16;	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Setting the customized broadcast contents
"TTM:PID-Data"	YES	Data for customized product identification code, Length L = 2, and default value is <b>0000</b> ;	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Setting the customized product identification code
"TTM:TPL-(X)"	NO	X = "+ 4", "0", and "6", "- 23" set up the transmission power (in dBm)	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Transmission power setting
"TTM:CDL-Xms"	YES	X = "0", "2", "5", "10", "15", "20", "25" set the delay from when BCTS output is set low till when UART outputs data (in ms)	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	If minimum delay not less than X, the actual delay will be X + Y ms (500 us < Y < 1 ms).

Note: Default settings are in blue bold font. Gray highlighted commands cannot be saved after power-off.

## Broadcast Data Setting

**Default broadcast data:** When the module EN pin is set low, or into the TEST mode (plugged after set low), the module will broadcast at an interval of 200 ms. In the domain of the broadcast data GAP\_ADTYPE\_MANUFACTURER\_SPECIFIC (iOS officially defined programming macro) the following contents are included (default of 9 bytes):

{

0x00,0x00,	Customerizing equipment type code, default setting 00 00, and can be set by the AT command;
0x00,0x00,0x00,0x00,	Current output status of four PWM (default), or two ADC values;
0x00,	Percentage of module power suppl (2.0 v = 0%);
0x00,0x00,	IO configuration, IO output / input status (real-time changing with IO current status);

}

Broadcast data will load automatically the status of current PWM output, or it is defined by the user to be the acquisition value of 2 ADC. And the 4 bytes will be in the same position. The module will always load automatically the the data of the last-time operated channel. Any writing to PWM(FFB1) will result in the loading of current status of 4 PWM output. Or, writing any but zero to ADC(FFD2) will result in the loading of 2 ADC acquisition values.

**Custom broadcast data:** if you use the AT command custom the broadcast content, a maximum length of 16 bytes (Blue font part), in the broadcast data GAP\_ADTYPE\_MANUFACTURER\_SPECIFIC domain will contain the following content, length is 2 + n bytes:

```
{
  0x00,0x00,          Custom coding equipment type, the default is 00 00, can be set by the AT
  command;
  Data [n],          Custom broadcast data, n <= 16;
}
```

Note: since the broadcast data can be defined by the AT command to modify, and electricity saving.After power on again, will use the last custom broadcast data.If custom broadcast data to all 0 (16 byte), argues that do not use the custom radio, and use the system default broadcast content.To avoid broadcast data is too long to bring extra power consumption, can also set the custom through broadcast data for any value of 1 byte.

## System Reset and Recovery

There are three kinds of module reset method, including the third method can restore system parameters:

1. Use the AT command reset module (see the serial the AT command section);
2. The use of the service channel interface, use the APP to remote reset module.Interface (see the BLE protocol descriptions (APP) - module parameter Settings "section).
3. Using hardware RESTORE foot foot (see definition table), **electricity for 30 seconds**, the foot down after **5 seconds**, module of the system parameters can RESTORE the user level (shallow recovery, release the foot immediately after reset), if continue to lower after **20**

**seconds** of module, system parameters RESTORE to factory Settings (depth recovery), and immediately reset. The foot with the internal pull-up, not into this mode by default.

**Being restored in shallow recovery system parameters include:**

- A) hijack password, return to "000000", do not use the password by default;
- B) initialization four-channel PWM mode, return to 0 x01, four-channel output 100% high pulse width;
- C) I/o output state of 0, if the IO configured to output, the default output low level;

**Depth in recovery in addition to the above system parameters, include the following parameters:**

- A) a UART baud rate, recovery to 9600 BPS.
- B) device name, return to "TAv22u - XXXXXXXX", X is the MAC after four bytes;
- C) serial data Delay, return to zero (500 us Delay < 1 ms);
- D) four channel PWM output frequency and restore x8235 0 (120 hz);
- E) broadcast cycle, restore to 2 (200 ms);
- F) product key, restore to 0 x00, 0 x00;
- G) IO configuration byte 0 x00, default IO7, IO6 signal hint, IO5 - IO0 do input;
- H) the custom length, back to 0;
- I) custom broadcast data, back to 0, all without the use of custom broadcast data, use the default radio data;
- J) enabled mode back to 0, the default level can make pattern;

**Note:** RESTORE (IO0) feet of particularity, in the circuit design, 30 seconds before they need to avoid the electric fields continuously, otherwise it will enter the recovery mode.

## IOS APP programming reference

Module is always to broadcast from mode, waiting for the intelligent mobile devices do give priority to scan, and connection. The scan and connection is usually done by APP, due to the particularity of BLE protocol, system Settings in the scan bluetooth connection no practical significance. Intelligent equipment must be responsible for BLE connection from the device, communication, disconnect the management issues, such as, it is usually implemented in the APP.

About BLE under the IOS programming, the key is to **Characteristic value**, (**Characteristic, this article called channel**) to **read**, **write**, and **open switch notice**. To read and write channel through the direct control of the straight drive module function can be realized, no additional CPU. Typical functions that extract is as follows:

```
/*!
 * @method writeValue:forCharacteristic:withResponse:
 * @param data The value to write.
 * @param characteristic The characteristic on which to perform the write operation.
```



\* @param type The type of write to be executed.  
\* @discussion Write the value of a characteristic.  
\* The passed data is copied and can be disposed of after the call finishes.  
\* The relevant delegate callback will then be invoked with the status of the request.  
\* @see peripheral:didWriteValueForCharacteristic:error:  
\*/

- (void)writeValue:(NSData \*)data forCharacteristic:(CBCharacteristic \*)characteristic type:(CBCharacteristicWriteType)type;

**Note: to write a characteristic value.**

***NSData \*d = [[NSData alloc] initWithBytes:&data length:mdata.length];***

***[p writeValue:d***

***forCharacteristic:c***

***type:CBCharacteristicWriteWithoutResponse];***

/\*!

\* @method readValueForCharacteristic:

\* @param characteristic The characteristic for which the value needs to be read.

\* @discussion Fetch the value of a characteristic.

\* The relevant delegate callback will then be invoked with the status of the request.

\* @see peripheral:didUpdateValueForCharacteristic:error:

\*/

- (void)readValueForCharacteristic:(CBCharacteristic \*)characteristic;

**Note: read some characteristic value.**

***[p readValueForCharacteristic:c];***

/\*!

\* @method setNotifyValue:forCharacteristic:

\* @param notifyValue The value to set the client configuration descriptor to.

\* @param characteristic The characteristic containing the client configuration.

\* @discussion Ask to start/stop receiving notifications for a characteristic.

\* The relevant delegate callback will then be invoked with the status of the request.

\* @see peripheral:didUpdateNotificationStateForCharacteristic:error:

\*/

- (void)setNotifyValue:(BOOL)notifyValue forCharacteristic:(CBCharacteristic \*)characteristic;

**Note: open the Characteristic value notice enabled switch.**

***[self setNotifyValue:YES forCharacteristic:c]; //open notice enabled switch.***

***[self setNotifyValue:NO forCharacteristic:c]; //close notice enabled switch.***

/\*

\* @method didUpdateValueForCharacteristic

\* @param peripheral Peripheral that got updated

\* @param characteristic Characteristic that got updated

- \* @error error Error message if something went wrong
- \* @discussion didUpdateValueForCharacteristic is called when CoreBluetooth has updated a
- \* characteristic for a peripheral. All reads and notifications come here to be processed.
- \*
- \*/

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

*Note: after each read operations, will perform to the callback function. The application layer save read data in this function.*

Scan on equipment, connection, and other communication details, you can refer to the letter chi da technology provided by the passthrough module test based on the IOS APP source code (ble Transmit Moudel v1.29). FFE9 inside realized with forward and FFE4 bluetooth data to the UART, forwarding serial data to the bluetooth two channel (Characteristic values) operation (notification and write operations), other direct driving function control method, is based on a channel (Characteristic values), speaking, reading and writing. But at different channel UUID and reading and writing bytes.

## BLE protocol description (APP interface )

### Bluetooth data channel 【service UUID: 0 xffe5 】

Characteristic value UUID	Executable operation	Bytes	Default	Remarks
FFE9 (handle: 0x0013)	Write	20	NO	Write data from UART TX output

Note: bluetooth input forwarded to a UART output. APP by BLE API interface to the channel after the write operation, the data will be output from a UART TX. Detailed operation rules can be seen in the UART passthrough protocol description (bridge mode) "section.

### Serial data channel 【service UUID: 0xFFE0】

UUID	Executable operation	Bytes	Default	Remarks
FFE4 (handle: 0x000E)	notify	20	NO	From the UART of the input data was informed in the channel to mobile devices

Note: forwarded to bluetooth serial input output. If you open the FFE4 channel notice made to switch (if using BTool operation, the need to write 01 00 0 + 1 = 0 x000f x000e), the CPU module via the UART sent RX legal data, in this channel will be to create a notify notification events, the APP can be directly processed in the callback function and use. Detailed operation rules can be seen in the UART passthrough protocol description (bridge mode) "section.

**PWM output (4 Channel) 【Service UUID: 0xFFB0】**

Characteristic value UUID	Operation Available	Bytes	Default Value	Example	Remarks	Pin corresponding to the channel
FFB1 (handle: 0x004D)	read /write	1	0x01	0x00	Initialize 4 PWM channels with all-low pulse-width	--
				0x01	Initialize 4 PWM channels with all-high pulse-width	
				0x02	Initialize corresponding PWM channels with current pulse-width	
FFB2 (handle: 0x0050)	read /write	4	0xFFFFFFFF	0xFF000000	PWM1 outputs all-high pulse width	P11
				0x00FF0000	PWM2 outputs all-high pulse width	P10
				0x0000FF00	PWM3 outputs all-high pulse width	P07
				0x000000FF	PWM4 outputs all-high pulse width	P06
				0x20202020	PWM1-PWM4 output pulse width of 32/256	--
FFB3 (handle: 0x0053)	read /write	2	0x8235	500 ≤ w ≤ 65535	PWM output signal frequency setting, the same for 4, 0x8235 (120 hz) by default	--
FFB4 (handle: 0x0056)	read /write	2	0x0000	0 ≤ t ≤ 65535	PWM changing time width, the same for 4, 0x0000 by default (sudden change)	--

**Directions:**

FFB1 is the setting channel for the initialization of 4 PWM. Writing to FFB1 (1 bytes) can initialize the 4 PWM. The factory setting is 0x01 by default with all-high pulse width output. The setting can be saved after power-off.

**0x00**, outputting 0% pulse width (all-low pulse width). Sleep mode allowed under this setting;

**0x01**, outputting 100% pulse width (all-high pulse width). Sleep mode allowed under this

setting;

**0x02**, Outputting current PWM value. This value will be saved immediately after being set, and will be used as the initial value of 4 PWM next time when plugged. Module will not sleep under this setting.

FFB2 is the setting channel for 4 PWM output duty ratio. To write FFB2 (4 bytes) can adjust the output duty ratio of 4 PWM. Each byte corresponds to a channel. 0xFF outputs all-high pulse width (100%) and 0x00 outputs all-low pulse width (0%).

If set as X, the duty ratio will be about  $X/0xFF$ . Also the channel can be read and written, and the final setting will be got. After power on, the default value will be 0xFFFFFFFF (all-high pulse width output).

When this function is enabled, the module will not enter into sleep mode, until it is set as 0xFFFFFFFF (all-high). It means the PWM output is shut down. This channel is used for setting PWM duty ratio, in the range from 0x00 to 0xFF, with default signal frequency of 120Hz. (see "FFB3 frequency control channel").

For example: 0xFF000000

1. Total of four PWM output channels;
2. 0xFF000000, four bytes, corresponding to four channels;
3. 0xFF outputs all-high pulse width (100%), and 0x00 outputs all-low pulse width (0%);
4. The default frequency of the pulse width is 120 Hz.

FFB3 is the channel for 4-channel PWM output frequency control. To write to FFB3 (2 bytes) can adjust the frequency of PWM output square wave. The width of the signal cycle  $w$  must meet:  $500 < w \leq 65535$  (one unit equivalent to 0.00000025 s), and the corresponding square wave cycle:  $0.000125 \text{ s} \leq T \leq 0.01638375 \text{ s}$ . Therefore the adjustable range of square wave signal frequency is:  $61.036 \text{ Hz} \leq f \leq 8 \text{ kHz}$ , and 4 PWM output square waves of the same frequency. Also to read the channel will get the last-time setting, and the setting can be saved after power-off. Factory setting of  $w$  is 0x8235 by default, and the corresponding default pulse-width frequency of is 120 Hz.

Example 1: Output the square wave of 120 Hz. Writing 0x8235 (33333) to FFB3 will set square wave cycle at  $0x8235 * 0.00000025 = 0.00833325 \text{ s}$ . So the frequency is around 120 Hz;

Example 2: Output the square wave of 1 KHZ. Writing 0x0FA0 (4000) to FFB3 will set the square wave cycle at  $0x0FA0 * 0.00000025 = 0.001 \text{ s}$ . So the frequency is around 1 kHz;

FFB4 is the channel for length control of 4 PWM output changing time. To write to FFB (2 bytes) can adjust the speed of frequency changes of 4 PWM output square waves. This is a value of time  $t$ , and  $t$  must meet:  $0 < t \leq 65535$ , (one unit equivalent to 100 ms). The longer the  $t$ , the slower the PWM changes from the current value to the target. While the smaller the  $t$ , the faster the change takes place. When  $t$  is zero, the target will be reached immediately. The changing time of 4 PWM share the same value. Also reading this channel will get the last-time setting, and the setting can be

saved after power-off. Factory setting is 0x0000 by default, corresponding conversion mode for immediate mutations.

**ADC input (2 Channel) 【service UUID: 0xFFD0】**

Characteristic value UUID	Executable operation	Bytes	Default	Remarks
FFD1 (handle: 0x0036)	Read/write	1	0x00	Enable control 0 x00: close two ADC channels 0 x01: open ADC0 channels 0 x02: open the ADC1 channels 0 x03: open two ADC channels
FFD2 (handle: 0x0039)	Read/write	2	0x01F4	Collection cycle (ms) 0x01f4 corresponding to 500 ms
FFD3 (handle: 0x003C)	Read/notify	2	0x0000	ADC0 sampling results, maximum to be 0x01fff
FFD4 (handle: 0x0040)	Read/notify	2	0x0000	ADC1 sampling results, maximum to be 0x01fff

Directions: 2 channel ADC input control. APP writes to FFD1 through BLE API interface to enable two 13 bit ADC channels. Writing to FFD2 can control the sampling cycle of two ADC channels (t, in ms), and t > = 100 ms.If the notify-enable function of FFD3 & FFD4 is enabled (Need to write 01 00 to 0x003C+1=0x003D and 0x0040+1 = 0x0041, if using BTool), a notify event will occur in the channel, each time the collection result comes out. The notify event is attached with the collection result, with the range 0 ~ 0x1FFF (low byte in front), which can be processed and used by the APP in the callback function. ADC reference power supply is the chip internal reference power source of 1.25 V. So the floatation of the voltage of power supply will not lead to any new measurement errors, but the sampling voltage being measured must be controlled between 0 ~ + 1.25 V.

**Programmable IO (8 Channel) 【Service UUID: 0xFFFF】**

Characteristic value UUID	Executive Operation	Bytes	Default Value	Remarks
FFF1 (handle: 0x0017)	Read/write	1	0b00000000	IO7 - IO0 configuration bytes.  When the corresponding bit is set to 0, bit7,bit6 indicate

				<p>that IO7 and IO6 are signal prompt pin and valid at low level.</p> <p>bit5 – bit0 indicate that IO5 - IO0 work as input ports</p> <p>When the corresponding bit is set to 1:</p> <p>bit7, bit6 indicate that IO7 and IO6 work as normal output port;</p> <p>bit5 – bit0 indicate that IO5 - IO0 work as output ports</p>
FFF2 (handle: 0x001A)	write	1	--	<p>IO7 ~ IO0 output status. It indicates the output level in IO7 ~ IO0. Bit 7 and bit6 are only valid when IO7 and IO6 work as normal output ports. When IO7,IO6 works as signal prompt pin, bit7 and bit6 are invalid.</p>
FFF3 (handle: 0x001D)	Read/notify	1	0x3F	<p>IO5 ~ IO0 input status. Notifications can be read and received. When notify-enabled switch on, the change of certain input level will be notified to the APP. IO7 and IO6 can only work as output or signal prompt pin, and the corresponding pins are invalid.</p>

Directions: IO configuration and control channel.

FFF1 is the configuration channel for 8 IO. The 8 bits control IO7 ~ IO0 (8 IO) correspondingly. When the two high bits - BIT7, BIT6 are 0, IO7 and IO6 works as signal prompt pin. IO7 prompts sleep status, where 0 stands for awake status and 1 for **sleep status**. IO6 prompts **connection status**, where 0 stands for connection status and 1 for disconnection status. When BIT7, BIT6 are 1, IO7 and IO6 work as normal output. They cannot work as input ports.

When the 6 low bits, BIT5 - BIT0, are set to 1, IO5 - IO0 work as output ports. When they are set to 0, IO5 - IO0 are used as an input ports.

FFF2 is the configuration channel for 8 IO. The 8 bits control IO7 ~ IO0 (8 IO) correspondingly. And

they are only valid when the corresponding bits are set to work as output. When certain IO is set to output, corresponding bit in this channel can be written, so the output control of the IO is realized. The bit corresponding to the IO that is set as input will be invalid.

Notes: IO configuration (FFF1) and output status (FFF2) are not saved after power-off by default. But the configuration and output status of IO1- IO7 (not including IO0) can be saved by writing 0x01 to FF99, the remote control extension channel. The module will use the last-time saved settings to initialize the 7 IO. That means the configuration and output status of IO0 cannot be saved after power-off. When power on, IO0 is always in default input status. This is used to detect the function of recovering to factory settings.(see the section “module parameter Settings”).

FFF3 is the input status channel of IO5 ~ IO0. The low 6 bit correspond to the input status of IO5 ~ IO0. But they are only valid when the corresponding bits are set as input. If the notify-enabled function of FFF3 is switched on (need to write 01 00 to 0x001D + 1 = 0x001E), a notify event will be created in the channel by APP when the level of the pins are changed, together with a byte to indicate the 6 IO’s status. The status data can be processed and utilized in the callback function by the APP. IO7 and IO6 can only work as output or signal prompt pin, so their corresponding bits are invalid.

Timed Rollover Output (2 Channel) 【Service UUID: 0xFFFF0】

Channel UUID	Executable operation	Bytes	Default value	Remarks
FFF4 (handle: 0x0021)	Read/write	4	0x00000000	Setting of the first time rollover delay of IO6 0: IO6 rollover not started non 0: in ms, delay before rollover
FFF5 (handle: 0x0024)	Read/write	4	0x00000000	Setting of the second time rollover delay of IO6 0: no rollover Non 0: in ms, delay before rollover
FFF6 (handle: 0x0027)	Read/write	4	0x00000000	Setting of first time rollover delay of IO7 0: IO7 rollover not started Non 0: in ms, delay before rollover
FFF7 (handle: 0x002A)	Read/write	4	0x00000000	Setting of second time rollover of IO7 0: No rollover Non 0: in ms, delay before rollover

Directions: Channel to configure timed rollover.

The module can be configured in timed rollover output mode, when IO6 and IO7 are set to normal output. The next rollover time of IO6 and IO7 can be set by writing to this channel. By setting the current status of IO output, 1 can jump to 0, or vice versa. Rollover will not be started if it is set to 0.

This function effects only when the two high bits of FFF1, BIT7 and BIT6, are set to 1 (as output).

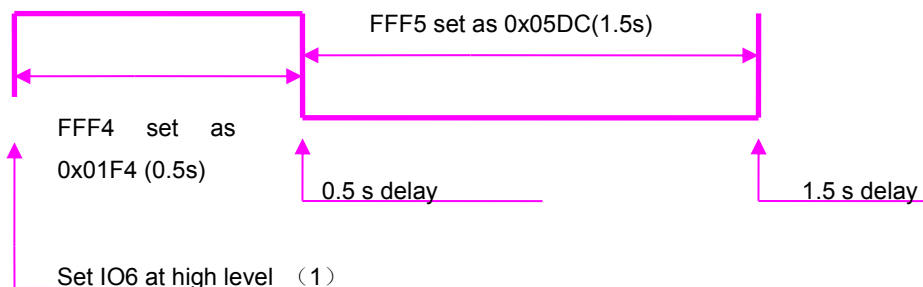
FFF4 is used to set the first time delay of rollover of IO6, and FFF5 is used to set the second time delay of rollover of IO6. If FFF4 is set to 0, rollover of IO6 will not be started. If FFF4 is set to a non-zero value, while FFF5 is set to 0, IO6 rollover is only started for once. FFF5 must be configured first and rollover is not started at this time. Then FFF4 can be set to non-zero values to start the timed rollover of IO6. Similarly, the timed rollover of IO6 can be stopped by writing 0 to FFF4, and any value written before in FFF5 will be cleared. The timing range is from 0 to 0xFFFFFFFF ms (4294967295 ms, or around 1193 hours, or around 49.7 days). Time conversion to hexadecimal is as follows:

0.5s	1s	1.5s	2s	3s	4s	5s
500ms	1000ms	1500ms	2000ms	3000ms	4000ms	5000ms
0x01F4	0x03E8	0x05DC	0x07D0	0x0BB8	0x0FA0	0x1388

Taking IO6 as an example, the steps to set a periodic repeated rollover are as follows:

1. Set IO6 as normal output by writing 0bx1xxxxxx to FFF1;
2. Set IO6 at high level (1) by writing 0bx1xxxxxx to FFF2;
3. Set FFF5 to 0x05DC (1.5 s) to set the second-time rollover delay first (0 for rollover only once)
4. Set FFF4 to 0x01F4 (0.5 s) to set the first-time rollover delay, and the rollover will start immediately

The order of point 3 and 4 cannot be reversed. FFF5 must be configured, before writing non-zero value to FFF4 to start rollover. Writing 0 to FFF5 means rollover for only once. A square wave with the cycle of 1.5 + 0.5 = 2 s will be there, after the operations mentioned above. During the period, high level (1) will remain for 0.5 and low level (0) will remain 1.5 s. Rollover can be stopped immediately by writing 0 to FFF4. IO6 will keep the current level unchanged.



**Rollover Cycle (2 s) Diagram**

FFF6 and FFF7 are the channels to configure the timed rollover delay for IO7, in the same way as for IO6.



Notes: If IO6 and IO7 are in the timed rollover cycle, writing to the IO output or re-configuration to signal prompt pin are all invalid. Current timed rollover has to be stopped before the above operations are performed.

**Level pulse width counting (2 channel) 【ServiceUUID: 0xFF0】**

Channel UUID	Executable Operation	Bytes	Default value	Remarks
FFF8 (handle: 0x002D)	Read/notify	4	0x00000000	IO4 last-time level duration (in ms)
FFF9 (handle: 0x0031)	Read/notify	4	0x00000000	IO5 last-time level duration (in ms)

Directions: counting and notifying channel of IO level duration.

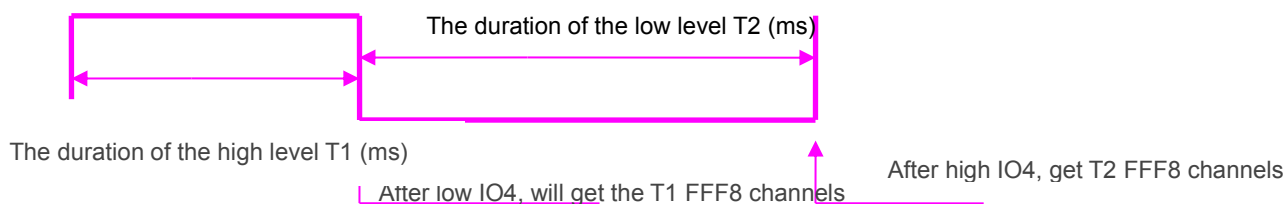
Level pulse width counting mode can be enabled when IO4 and IO5 of the module are set as normal input.

It functions only when the two high bits of FFF1 - BIT5 and BIT4 - are set to zero (as an input port).

FFF8 is the channel for IO4 (P1.6) level pulse width counting and notifying. Notify is enabled through BLE API interface by APP (need to write 01 00 to 0x2D + 1 = 0x2E, if using BTool). A notify event will be created in this channel, after each rollover of IO4, together with the last-time level duration (in ms, range from 0 to 0xFFFFFFFF). Maximum value is 0Xffffff (ms), which equals to 4294967295 ms, or about 1193 hours, or about 49.7 days). It can be processed and used by APP in the callback function.

FFF9 is the channel for IO5 (P1.7) level counting pulse width counting and notifying. Notify is enabled through BLE API interface by APP (need to write 01 00 to 0x31 + 1 = 0x32, if using BTool). A notify event will be created in this channel, after each rollover of IO4, together with the last-time level duration (in ms, range from 0 to 0xFFFFFFFF). Maximum value is 0Xffffff (ms), which equals to 4294967295 ms, or about 1193 hours, or about 49.7 days). It can be processed and used by APP in the callback function.

Notes: The level counted is not the current level but the last-time one). Current level can be got by reading FFF3. Due to the limitation of BLE protocol, delay of submission of collected results will not be longer than the connection interval.



Level pulse width counting diagram (IO4 as an example and IO5 being the same)

### Anti-hijacking key **【Service UUID: 0xFFC0】**

The module supports encryption for anti-hijacking. This service can prevent effectively unauthorized mobile devices (or mobile phones) from being connected to the module. The initial password is 000000 (ASCII). In this case APP does not need to submit a password, so it is regarded as no use of password and any mobile device that has installed the specified APP can connect to the module.

Setting and backup of new password (not all-zero) can be done by APP. If a new password is set (not all-zero value), anti-hijacking is enabled. The APP must submit once the set password to the module within 2 seconds after the Bluetooth connection is done. Otherwise the module will break the connection up. Before the APP submits the correct password to the module, it is no way to do any writing to the channel, except password submission.

If the password needs to be restored, **the module has be reset first. The module will restore the default factory-set password, if RESTORE (IO0) pin is pulled low within 30 s and kept for 5 s.** For safety's sake, the module does not allow password reading, and the APP shall be responsible for password memorizing.

The protocol provides password channels to realize the submission, modification and cancellation of the password. Meanwhile, event notify service of password is also provided, to inform the APP of the password operation results, including 4 events of password correct, password error, password update success, and password use cancelled.

Channel UUID	Executable Operation	Bytes	Example	Remarks
FFC1 (handle: 0x0045)	write (Saved after power-off)	12	"123456123456"(ASCII)	Submit 123456 as the new password , and the new password must be same as the previous one
			"123456888888"(ASCII)	Change the previous password 123456 into the new one of

				888888, and the previous password must be correct
			"888888000000"(ASCII)	Cancel the use of password by changing the new password to 000000, and the previous password must be correct
FFC2 (handle: 0x0048)	notify	1	0 (PWD_RIGHT_EVENT)	Password submission correct.
			1 (PWD_ERROR_EVENT)	Password submission error
			2 (PWD_UPDATED_EVENT)	Password update success
			3 (PWD_CANCEL_EVENT)	Cancelling the use of password

Description:

1.Password is structured with 12-byte ASCII, where the red part is the current password and the blue part is the new password;

2.Current password is "000000" by default, before modified by APP;

The notification of execution results about the password operations will be created in the channel, when the notify-enable function of FFC2 is switched on (need to write 01 00 to 0x0048 + 1 = 0x0049, if using BTool).

When APP submits "123456123456", it means the new password is same as the current one. And the APP will be notified in FFC2 of "notify: 0 (PWD\_ RIGHT\_EVENT)". It shows the password submission is correct;

When the password submitted by the AP (red part) is different from the current one, such as: "123455xxxxxx", regardless of the value of "xxxxxx" part, the APP will be notified in FFC2 of "notify: 1 (PWD\_ ERROR \_EVENT)". It shows the password submission error.

When the APP submits "123456888888", it means the new password is "888888" and the current password is "123456". The APP will be notified in FFC2 of "notify: 2 (PWD\_ UPDATED \_EVENT)". It shows the password update successful;

When the APP submits "888888000000", it means the new password will be changed to all zeros. The APP will be notified in FFC2 of "notify: 3 (PWD\_ CANCEL \_EVENT)". It shows the use of password is cancelled.

**Battery Power Report 【Service UUID: 0x180F】**

Channel UUID	Executable Operation	Bytes	Default Value	Remarks
--------------	----------------------	-------	---------------	---------

2A19 (handle: 0x000A)	Read/notify	1	Percentage of power charge	Reading the current battery power percentage, or automatically creating notification
-----------------------------	-------------	---	----------------------------------	---

Directions: the channel for battery power reading or notifying.

The APP reads 2A19 through the BLE API interface, to obtain the percentage of the current power supply to the module. If the notify-enabled function of the channel is switched on (Needs to write 01 00 to 0x000A + 1 = 0x000B, if using BTool), a notify event will be created in this channel, every time the batter power is read, together with the power percentage (maximum of 100% (3 v), and minimum of 0% (2 v)). The APP can directly process and use the data in the callback function

### RSSI Report 【ServiceUUID: 0xFFA0】

Characteristic value UUID	Executable Operation	Bytes	Default Value	Remarks
FFA1 (handle: 0x005D)	Read/Notify	1	0x00	RSSI 值, 可以读取/自动通知 RSSI values, can be read/automatic notification
FFA2 (handle: 0x005A)	Read/write	2	0x0000	RSSI 自动读取周期设置, 0x0000 为关闭自动读取。 RSSI read automatically cycle set, 0 x0000 for close read automatically.

Directions: RSSI reading and return channel.

The APP reads FFA1 to obtain RSSI that the module receives from the mobile device, through BLE API interface. If the notify-enable function is switched on (Need to write 01 00 to 0x005d + 1 = 0x005e, if using BTool), a notify event will be created in the channel, every time RSSI is read, together with RSSI, which can be processed and utilized in the callback function by APP.

The cycle of RSSI reading (in ms) is set by APP reading and writing to FFA2 through BLE API interface. When this cycle is set to 0x0000, it is considered that automatic periodic reading of RSSI is closed. But active reading is still available at any time. The RSSI value that is read is of signed char type.

RSSI notify-enable function of the channel has to be switched off, if RSSI data return needs to be stopped. Meantime reading RSSI has to be stopped by writing 0x0000 in the channel. Otherwise there will be unnecessary power consumption.

**Module Parameter Settings 【Service UUID: 0xFF90】**

Channel UUID	Executable Operation	Whether or not to save	Bytes	Default Value	Remarks
FF91 (handle: 0x0062)	Read/write	YES	16	TA <sub>v</sub> 22u-xxxxxxx (ASCII string with terminator)	Device name, XXXXXXXX for the last four bytes of the physical address
FF92 (handle: 0x0065)	Read/write	NO	1	0	Bluetooth connection interval: 0: 20ms 1: 50ms 2: 100ms 3: 200ms 4: 300ms 5: 400ms 6: 500ms 7: 1000ms 8: 2000ms
FF93 (handle: 0x0068)	Read/write	YES	1	1	Set the baud rate of UARTs: 0: 4800 bps 1: 9600 bps 2: 19200 bps 3: 38400 bps 4: 57600 bps 5: 115200 bps
FF94 (handle: 0x006B)	write	—	1	no	Channel to control remote reset and recovery: ➤ Remote reset control, by writing 0x55 to reset the module ➤ Remote shallow recovery control, by writing 0x35 to shallow-recover the module (restoring user data only) and reset

					Remote deep recovery control, by writing 0x36 to deep-recover the module (all back to factory settings) and reset
FF95 (handle: 0x006E)	Read/write	YES	1	0	Set the broadcast cycle: 0: 200 ms, 1: 500 ms, 2: 1000 ms, 3: 1500 ms, 4: 2000 ms, 5: 2500 ms, 6: 3000 ms, 7: 4000 ms, 8: 5000 ms,
FF96 (handle: 0x0071)	Read/write	YES	2	0x0000	Set product identification number
FF97 (handle: 0x0074)	Read/write	No	1	1	Set the transmission power: 0: +4 dBm 1: 0 dBm 2: -6 dBm 3: -23 dBm
FF98 (handle: 0x0077)	Read/write	YES	16	The default broadcast content, See the "broadcast data set" section	Set customized broadcast data ,Customizing broadcast data, 0 < n <= 16
FF99 (handle: 0x007A)	write	—	1	无	Remote control extension channel: ➤ 0x01: saving-trigger control of IO configuration output. Writing 0x01 will trigger the saving of current IO configuration and output status. IO7 – IO1 will be initialized to use the saved

					<p>configuration and output status when power on again. But IO0 is always set as input by default when power on, working as the test port of factory setting recovery.</p> <p>➤ <b>0x02</b>: remote shutdown control. In the pulse-enable mode, writing 0x02 to the channel can shut down the module remotely.</p>
FF9A (handle: 0x007D)	Read/write	YES	1	0b <b>00000</b> <b>000</b>	<p>System function enabled switch:</p> <p><b>BIT0</b>: Enabled mode setting, 0 by default, corresponding to the level-enabled at low. 1 means pulse-enabled. Every time the EN pin receives a pulse, the module will switch between boot-up (starting broadcast) and shutdown (stopping broadcast) in turn. Effective pulse width T must meet: <math>W &gt; 200</math> ms. When the broadcast time exceeds 30 s but the module is still not connected, it will shut down automatically</p> <p><b>BIT1~BIT7</b>: Temporary unused。</p>

Note: gray highlighted commands will not be saved when power-off.

Directions: [module information configuration channel.](#)

FF91 is the channel for setting device names. Reading and writing to this channel can obtain and set the module name. The length of the name set must meet the condition:  $0 < L < 17$ . **And the name is suggested to end with the terminator (' \ 0 ')**. The default name is "TAvvvv - XXXXXXXX \ 0" (16 byte), where VVVV is the firmware version number and XXXXXXXX is the last four bytes of the MAC address.

FF92 is the channel to set the connection interval. The interval of connection between mobile devices and the module can be set by writing to this channel. Thus the device power consumption and the data throughput can be controlled in a flexible way.

In order to raise the connection speed, the setting of connection interval will not be saved. It will always work at the default value (20ms) after power on.

Test shows that it takes around 30s to wait when the connection interval is changed from 500 ms to another by iPhone 4S (iOS 5.1.1). But the execution efficiency will be very high if the connection interval is changed from a high frequency one (ie. 20ms), resulting from BLE protocols.

FF93 is the channel to set the baud rate of module UARTs. Baud rate of the module's universal UARTs can be set by reading and writing to the channel. The new baud rate will take effect in two seconds after setting and can be saved after power-off. The factory default setting is 1 (9600 bps).

FF94 is the channel to control the remote reset and recovery. Various controlling functions can be realized by writing different values to the channel.

1. Write **0x55** to this channel will soft-reset the module.
2. Writing **0x35** to the channel will shallow-recover the module. All user settings will be recovered to the factory defaults, including IO output status, PWM initialization mode and user password. Afterwards, the module will be reset.
3. Writing **0x36** to the channel will deeply recover the module. All system settings will be recovered to the factory defaults and the module will be reset afterwards.

FF95 is the channel to set the broadcast cycle of the module. Broadcast cycle can be set by reading and writing to this channel. The setting can be saved after power-off. And the factory default setting is 0 (200 ms).

FF96 is the channel to set the product identification code of the module, by reading and writing to the channel. The APP can be filter and connect to specific product type through this code. The setting can be saved after power-off. And the factory default setting is 0 x0000.

FF97 is the channel to set the transmission power of the module, by writing to this channel. The setting **cannot be saved** after power-off. And the factory default setting is 1 (0 dBm).

FF98 is the channel to set the broadcast contents of the module. Broadcast data can be customized by writing to this channel. The setting can be saved after power-off.

When the data is all 0 (16 byte), it is regarded that default broadcast data is used, instead of customized data. (see the section "broadcast data set").

FF99 is the channel for remote control extension. By writing different values, specific controlling functions can be realized.

Writing **0x01** to this channel will trigger the module to save the current configuration and the output status of all I/Os (except IO0). When power on again, the module will always initialize IO7 – IO1 with the saved settings and output status. While IO0 is always set to default input status, to work as the triggering IO of factory setting recovery. But afterwards, IO0 can also be set as output, just as other I/Os.

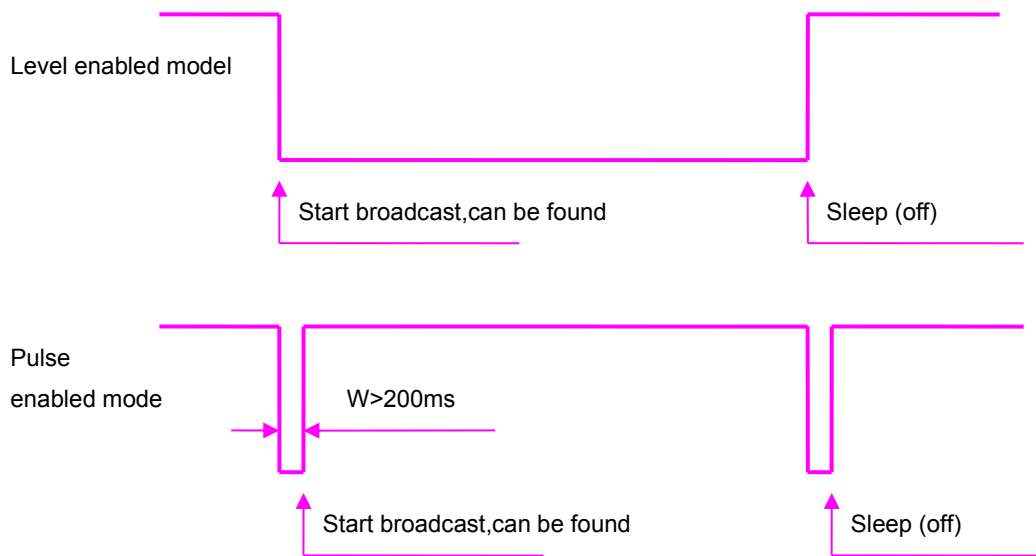
In the pulse-enabled mode, writing **0x02** to this channel will shut the module down remotely. But the function is invalid in level-enabled mode.



FF9A is the channel of system function enabled switch. Writing through BIT0 ~ BIT7 can be turned on or turn off specific functions of the system. 1 means on and 0 means off. Default setting is 0b00000000 for all. [This setting can be saved after power-off.](#)

BIT0: The default setting is 0, in enabled mode, corresponding to low level enabled (starting broadcast) and high level sleeping(0.4 uA). When the bit is set to 1, the module will be in pulse-enabled mode. The module will be switched between on (starting broadcast) and off (deep sleeping, 0.4 uA) in turn. If the module is in the connection status, “off”takes no effect. While the module is in the broadcast status, “off”takes effect.

BIT1~BIT7:Reserved.



**Level enabled model and pulse enabled model diagram**

In level-enabled mode, broadcasting (so can be found and connected) has the following features:

1. If EN pint is enabled (set low), the module will keep broadcasting, until it is connected or EN is set high.
2. Regular disconnected or timeout disconnected, as long as EN is set low, the module will always keep broadcasting, until it is connected again.

In pulse-enabled mode, broadcasting (so can be found and connected) has the following features:

1. If broadcasting for 30 s after enabled, but still not connected, the module will stop broadcasting and shut down.
2. In regular disconnection status, the module will continue to broadcast for 30 s. If it is still not connected, the module will stop broadcasting and shut down.

If in timeout disconnection status, the module will keep broadcasting until it is connected again. And in this case, EN shutdown takes no effect.

In level-enabled mode when IO6 works as signal prompt pin (prompt of bluetooth connection status by default), low level is output when bluetooth is connected. While high level is output, when Bluetooth is not connected, or is disconnected (either timeout or active disconnecting) and not re-connected.

In pulse-enabled mode when IO6 works as signal prompt pin (prompt of bluetooth connection status by default), the output signal has the following features:

1. When connected, low level pulse (1 s) will be output for once.
2. When bluetooth is regularly disconnected (by APP active disconnecting), low level pulse (0.5 s) will be output for once.
3. When Bluetooth is disconnected for timeout, 2 Hz square-wave will be output. The prompt will last for 2 minutes, keeping broadcasting in the duration, and cannot be shut down. The broadcasting will stop until the module re-connect with the main device.

Broadcasting status and IO6 prompt methods in different enabled modes:

Module status	enabled but not connected		connected		Active disconnected		Timeout disconnected	
	IO6 prompt method	Broadca sting status	IO6 prompt method	Broadca sting status	IO6 prompt method	Broadca sting state	IO6 prompt method	Broadca sting status
Level enabled model	High level	Keep broadca sting	Low level	Stop broadca sting	High level	Keep broadca sting	High level	Keep broadca sting
Pulse enabled model	High level	Broadca sting for 30 seconds	A low level pulse w = 1 s	Stop broadca sting	A low level pulse w = 0.5 s	Broadca sting for 30 seconds	2 Hz square wave for 2 minutes	Keep broadca sting

#### Device information 【ServiceUUID: 0x180A】

Channel UUID	Executable Operation	Bytes	Default Value	Remarks
2A23 (handle: 0x0003)	Read	8	xxxxxx0000xxxxxx (Hex)	System ID, where xxxxxxxxxxxx is the physical address of module chip, with low byte in front
2A26 (handle: 0x0005)	Read	5	V2.2u (ASCII)	Software version number of module

Directions: module information reading channel.

2A23 is the channel to obtain the module information. Reading this channel can get the module ID, in the format of xxxxxx0000xxxxxx, where xx part is the physical address MAC of the module chip in six byte (low byte in the front).

2A26 is the channel to read the software version number of the module. Reading this channel will get the module software version, in the format of Vx. Xx where x.xx stands for the firmware version number.

## **Port timed events configuration 【ServiceUUID: 0xFE00】**

Port timed event configuration service is used to set the timed events of IO or PWM ports. This service offers the function to set the timed task. That is, a certain host does certain actions at a certain time. The host of execution can be any one among the 10 ports, including 6 sudden-change outputs and 4 gradual-changeable PWM output channels. Action types can be sudden change, or gradual change.

### **1. Timed event setting**

This service provides up to 32 timed events to be settable. Event refers to the specific action at a specific time.

## **Timed event (EVT) = Execution Time + Action Types**

Settings can be done through event reading and writing channel (UUID: 0xFE03). The following parameters are included:

- Event index number, 1 byte, used to indicate the index number of the event modified or set;
- Execution time (time), 7 bytes, used to indicate the time of triggering an event;
- Action type, 1 byte, used to indicate the action executed when the timing overflow, including outputting high level, outputting low level, level rollover, PWM sudden changes, and PWM gradual changes;
- Operating parameters, 3 bytes, used to set target duty ratio and gradual-change overhead time of PWM. The parameter has nothing to do with the 6 sudden-change outputs, and is used specifically to define the gradual changes of the 4 PWM channels.

### **2. Timed task setting**

## **Timed Task = a host of execution + a Timed event**

The ports which can be set to execute timed events, or we say the hosts, include 6 IO and 4 PWM outputs.

When the timed events are started, timed tasks are formed. When the timed events are triggered, the host will execute as per the definition of events.

Each port can be configured with up to 32 timed events, and has separate responding switches.

There is no conflict among the settings of the ports. And the same timed events can be configured in different port at the same time. But if the action type of the event is invalid for certain ports, the action will be neglected. For example, IO0 port (no PWM output function) is configured to start certain PWM gradual-change event, so IO0 will ignore it when the timed event is triggered. Settings can be done through the reading and writing channel(**UUID:0xFE05**), including the following parameters:

- Port index number, 1 byte, used to indicate the port that is modified or configured;
- Events start bit, four bytes, total of 32 bit, respectively controlling the responding switches of the 32 timed events.

Event reading and writing channel (UUID: 0 xfe03) and port event reading and writing channel (UUID: 0 xfe05) are multiplexing writing interfaces. When writing each time, “event index number” or “port index number” of the first byte is directed to the event or port that needs to be set (as an indicator). And other bytes following are for details of settings.

If it is wanted to get the definition of certain event or the setting information of certain port, it is needed to write the index number that is wished to be read, through event reading indicator channel (UUID: 0xfe02) or port event reading channel port (UUID: 0xfe04), before reading the event reading and writing channel (UUID: 0xfe03) and port event reading and writing channel (UUID: 0xfe05) to get the related information of specified indice.

### **3. Timed task enabled Settings**

Event port configuration channel (UUID: 0 xfe06) is used to control the enable switch of timed tasks (port timed events), including general enable bit (EA) of all the timed taks, individual enable bit of timed events of 6 IO and 4 PWM, control bit of timed event clearance (CEVT), and control bit of port timed event clearance (CPORT).when control **bits of timed event clearance** and **timed task clearance** are set, the configuration information of all the 32 timed events and 10 port timed events will be cleared.

UUID:0xFE03						
EVT 0	EVT 1	EVT 2	EVT5 ...EVT 28	EVT2 9	EVT3 0	EVT3 1
Timing time	Timing time	Timing time		Timing time	Timing time	Timing time
Action type	Action type	Action type	.....	Action type	Action type	Action type
Operating parameters	Operating parameters	Operating parameters		Operating parameters	Operating parameters	Operating parameters

UUID:0xFE05								UUID:0xFE06	
POR T0 POR T1 POR T2 POR T3 POR T4 POR T5 POR T6 POR T7 POR T8 POR T9	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO0	EA
	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO1	
	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO2	
	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO3	
	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO4	
	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO5	
	bit0	bit1	bit2	.....	bit29	bit30	bit31	PWM0	
	bit0	bit1	bit2	.....	bit29	bit30	bit31	PWM1	
	bit0	bit1	bit2	.....	bit29	bit30	bit31	PWM2	
	bit0	bit1	bit2	.....	bit29	bit30	bit31	PWM3	
								CEVT	
								CPOR	

Timed events, timed tasks, and Timed task enabled configuration diagram

#### 4. Condition to respond to Timed event EVT:

- i. Timed event of EVT time overflow triggering;
- ii. Responded port opening up the resonating switch BIT which indicates to the timed event EVT;
- iii. Individual enable bit which indicates to the timed event of responding port making IO/PWM enabled;
- iv. General enable bit of timed events EA enabled;

As the table showed above, EVT2 is timing triggered. If PORT2 turns on the corresponding bit2, while IO2 and EA bit are enabled at the same time, IO2 will trigger EVT2 to execute as per the action type.

#### 5. About the priority:

Low index event or action of port will be executed as priority. If the timings of timed event 1 and timed event 2 are same, and port 0 and port 1 both start these two timed events, and two timed events are triggered simultaneously, the priority of execution ports are like this:

1. timed event 1 at port 0;
2. timed event 1 at port 1;
3. timed event 2 at port 0;
4. timed event 2 at port1;

#### Notes:

If it is wanted to timely control the IO, corresponding IO needs to be configured as output first. See the section "programmable IO (8 channels)".

- ✓ timing function is valid either when the module is connected or disconnected.
- ✓ The configuration information of timing cannot be save after power-off. If the information lost, it can be got back by synchronous refresh through APP.
- ✓ In order to avoid the error of the RTC clock, it is suggested that RTC clock be synchronously refreshed before APP connecting or breaking up.

Characteristic value	Executable operation	Bytes	Byte NO.	Default Value	Definition	Remarks

FE01 (handle: 0x0086)	R/W	7	BYTE 7~ BYTE 0	0x07D001010 00000	Second/minute/hour/day/ month/year (L)/year(H)	RTC clock operation channel Current system clock can be read and modified through the channel value range: Sec.: 0~59; Min.: 0~59; Hour: 0~23; Day: 1~31; Month: 1~12; Year : More than 2000 2000 以上  The default time is 0 hours 0 minutes 0 seconds. January 1, 2000
FE02 (handle: 0x0089)	R/W	1	BYTE 0	0x00	Event index number	Indicator of event reading. This indicator must be set before reading certain event, to make it direct to the event that needs to be read and read FE03 afterwards. Value range: 0 ~ 31, respectively standing for 32 timed events.
FE03 (handle: 0x008C)	R/W	12	BYTE 0	0x00	Event index number	Event reading and writing channel. Reading and writing to this channel will obtain and set timed events. ·Event index number: Value range: 0 ~ 31: respectively

			BYTE 7~ BYTE 1	0x000000 00000000	Second/minute/hour/day/month/year (L) /year(H)	standing for 32 timed events; ·Timing (in second / minute / hour / day / month / year): Value range same as that for RTC clock. If one of the bytes is invalid (FF), the effective byte time at low level will be treated as loop timing. For example, the following timing (Hex) : 00 FF 01 01 D0 07 01 means the timed event will be triggered in the second of 0 at any minute.
			B Y T E 8	0x00	Action Type	·Action type: Value range: 0: no action; 1: IO output low level; 2: IO output high level; 3: IO level rollover; 4: PWM sudden change; 5: PWM gradual change;
			B Y T E 9	0x00	PWM New Duty value	·PWM New Duty Value: Action types are effective for 4 or 5, after the change of Duty values, values range: 0 ~ 255; When is 0, the duty ratio is 0%, that is the low level, When is 255 , the duty ratio is 100%, that is the high level;
			B Y T E 10	0x00	Low Byte of PWM Gradual Change	·PWM sudden change times: Means the time that it takes from the current duty ratio changing to a new duty ratio. It is valid when the operation value is 5. Greater value means slower change, while
			B Y T E 11	0x00	High Byte of PWM Gradual Change	



						smaller value means faster changes. Value range: 0 ~ 65535 (in 100 ms);
FE04 (handle: 0x008F)	R/W	1	B Y T E 0	0x00	Port index number	Indicator of port event reading Port index: The indicator must be set before reading all the applicable events of certain port, to be directed to the port that will be read, followed by reading to FF05. Value range: 0 ~ 9: respectively IO0 ~ IO5 and PWM0 ~ PWM3 timing ports.
FE05 (handle: 0x0092)	R/W	5	B Y T E 0	0x00	Port index number	Port events reading and writing channel.  ·Port index number Value Range: 0 ~ 9: respectively standing for IO0 ~ IO5 and the timing ports of PWM0 ~ PWM3. ·Enable switch of applicable events, with different bit corresponding to the 31 timed events respectively. Value range: 1: on; 0: off;
			B Y T E 4 ~ B Y T E 1	0b0000000 000000000 000000000 00000000	Timed events enabled 0 ~ 31	

FE06 (handle: 0x0095)	R/W	2	B Y T E 0	0b <u>0</u> 000000 0	Bit0: General Enable of Timing	Event port configuration.  Value range: 1: enabled; 0: off;  ·BYTE0: BIT0, for general enabling of port events (EA). ·BYTE0: BIT1 ~ BIT7, BYTE1: BIT0 ~ BIT2, both for separate enabling of timed port events. · BYTE1: BIT3, for control of all timed event clearance;(CEVT) ·BYTE1: BIT4, Empty for timing task control bit, remove the response configuration of all ports to any timing event. (CPORT)
					Bit1:IO0 Enable	
					Bit2:IO1 Enable	
					Bit3:IO2 Enable	
	Bit4:IO3 Enable					
	Bit5:IO4 Enable					
	Bit6:IO5 Enable					
	Bit7:PWM0 Enable					
	Bit0:PWM1 Enable					
	Bit1:PWM2 Enable					
	Bit2:PWM3 Enable					
	Bit3:Control of timed event clearance					
	Bit4: Control of timed task clearance					
	-					
-						
-						

Tips: How to fulfil a timed task in four steps:

1. Design one or more timed events (specifying what to do at what time) (writing 0xfe03);
2. Specify the IO to carry on this event (setting up the relation between the timed task and execution host) (writing 0xfe05);
3. Turn on the response-enabled switch of the IO to timed task (response allowed) (writing 0xfe06);
4. Turn on the general enable switch to timed task (writing 0xfe06).

Examples:

Ex 1: as the diagram below, to set IO2 turnover for once, on 5 seconds 4 minutes 4 hours, January 2, 22013 set IO2 flip again.

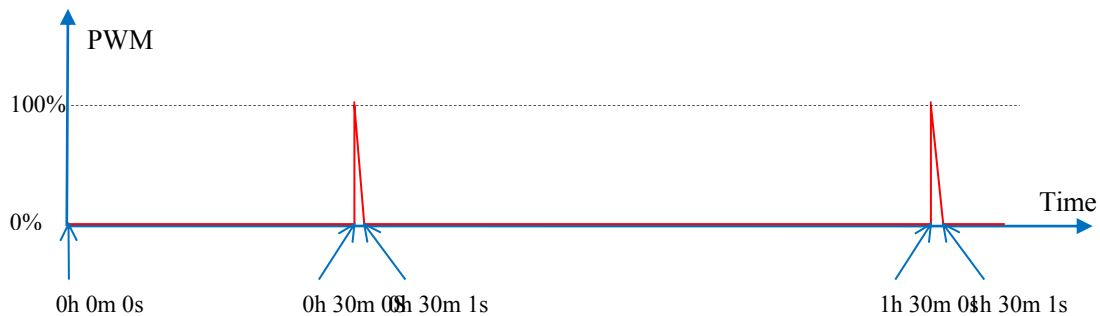


BLE master device operation steps to slave module:

- ✓ Write **0x04** to the IO configuration character (UUID: 0 xfff1) and set IO2 as output;
- ✓ Write the following data in the reading and writing channel (UUID: 0xfe03) to the event, to set a timed task which will trigger IO rollover at 4:04:05, Jan.2, 2013:  
Hex (low byte in the front) : **00 05 04 04 02 01 DD 07 03 00 00 00.**
- ✓ Write the following data to the port event reading and writing channel to turn on the timed event which is at the timed port of IO2.  
Hex (low byte in the front) : **02 01 00 00 00.**
- ✓ Write the following data in the event port configuration characteristic (UUID: 0 xfe06) to enable the general timed events and to enable IO2.  
HEX (low byte in the front): **09 00.**
- ✓ Write the following data to the RTC clock operation channel (UUID: 0xfe01) to update the RTC clock of the module (ie. 3:04:05, Jan. 2, 2013)  
Hex (low bytes in the front) : **05 04 03 02 01 DD 07.**

Till now the timing configuration is completed. The timed event is waiting to be triggered.

Ex 2: as the diagram below, set the duty ratio of PWM0 to change to 100% in a sudden at minute 30 of every hour, and then the duty ratio gradually changes to 0%. The overhead time is 1 s.



Steps of operation to the slave module by BLE master device are as follows:

1. Writing the following data to the event reading and writing channel (UUID: 0xfe03) to set timed event to trigger PWM sudden change at minute 30 of every hour, with duty ratio of 100% :

Hex (low to high byte) : **01 00 1E FF FF FF FF FF 04 FF 00 00.**

2. Writing the following data to the event reading and writing channel (UUID: 0xfe03), to set timed event 2 to trigger PWM gradual change at minute 30 of every hour, with duty ratio of 0%. And the overhead time is 1s:

Hex (low to high byte) : **02 00 1E FF FF FF FF FF 05 00 E8 03.**

3. Write the following data to port event read/write channel (UUID: 0xfe05), to start timed event 1 and timed event 2 at PWM0 timing port:

Hex (low to high byte) : **06 06 00 00 00.**

4. Write the following data to the configuration characteristic of event port (UUID: 0xfe06), to enable the general enable bit of timed port events and the PWM0 enabled bit:

Hex (low to high bytes) : **81 00.**

5. Write the following data to the RTC clock operation channel (UUID: 0 xfe01), to update the RTC clock of the module (ie. 3:04:05, Jan. 2<sup>nd</sup>, 2013):

Hex (low to high byte) : **05 04 03 02 DD 07 01.**

Till now, the timing functional setting has been completed. And the timed event is waiting to be triggered.

## Transmission test by APP

Module test tool (APP) for the IOS platform can be downloaded in AppStore. Running AppStore in iPhone 4S, iPhone 5(S), or iPad 4 and searching BLE Transmit\_Moudel, the APP can be found, downloaded and installed. (source code is available if needed). There are three ways to install this application:

1. Searching and downloading from APP STORE. And AppleID is necessary and can be applied for free.
2. Using the source code to compile and download to your iOS device. And apple developer account is a must.
3. Jailbreaking the iOS device and downloading the IPA file (as the exe file in Windows system) in RF-Star's official website. IPA file can be installed with a third-party program such as PP assistant.

When the APP is installed and run, there will be automatic scanning and the devices scanned and found will be shown in a list (probably will be a prompt to ask for Bluetooth enabling). Clicking the name of a device in the list, the APP will try to connect it. And when the connection is successful, the APP will jump to main UI of control.



Then automatically scan service data, after the prompt to complete.



At this time if the UART of the module is ready (which means the main CPU or the serial terminal is connected), work can start to perform manual and automatic transceiving test. IP stands for the data packets from iPhone, and PC stands for the packets from the main CPU or a serial terminal.



Notes: if a serial terminal is used for test, the data from the terminal must be sent to the mobile phone, **and BRTS must be set low**, in case the module will enter into sleep mode.

Regarding IOS programming, as per the bluetooth protocol, Mobile device and send data by writing to the corresponding service (UUID) of channel B (transmission).

Transmission of module data to mobile device is done in the way of notification. So the notification of the corresponding service (UUID) of Channel A (receive) needs to be enabled, after the APP is started. Then the data packet will be sent automatically from module's UART to the mobile device.

For detailed operations, please refer to source code of the APP for transparent transmission test on iOS platform (BLE Transmit\_Moudel V1.29), provided by RF-Star.

The software may be updated at any time. Please maintain your attention to the official website of RF-Star (<http://www.szrfstar.com/en>).

## Test by USB Dongle and Btool

The BLE module can be tested of bluetooth communication by the USB dongle in the TI official CC2540 MinDK simulating mobile phones and the Btool installed in the directory of C: \ Texas Instruments \ BLE - CC254x - 1.2.1 \ Projects \ Btool \ Btool.exe.

The USB Dongle needs to use the engineering program installed in the directory of:

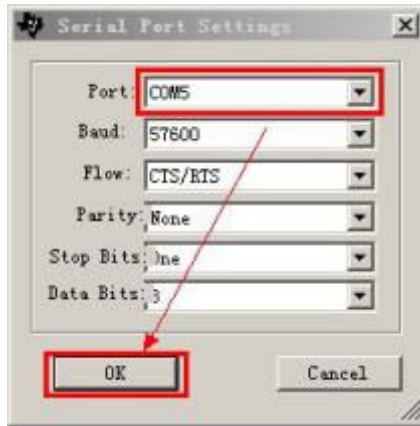
C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\ble\HostTestApp\CC2540 to compile and download the tool into a USB dongle. The details of BTOOL can be found in the official documentation

CC2540 Mini Development Kit User's Guide (Rev. B).pdf

### Connect BLE module

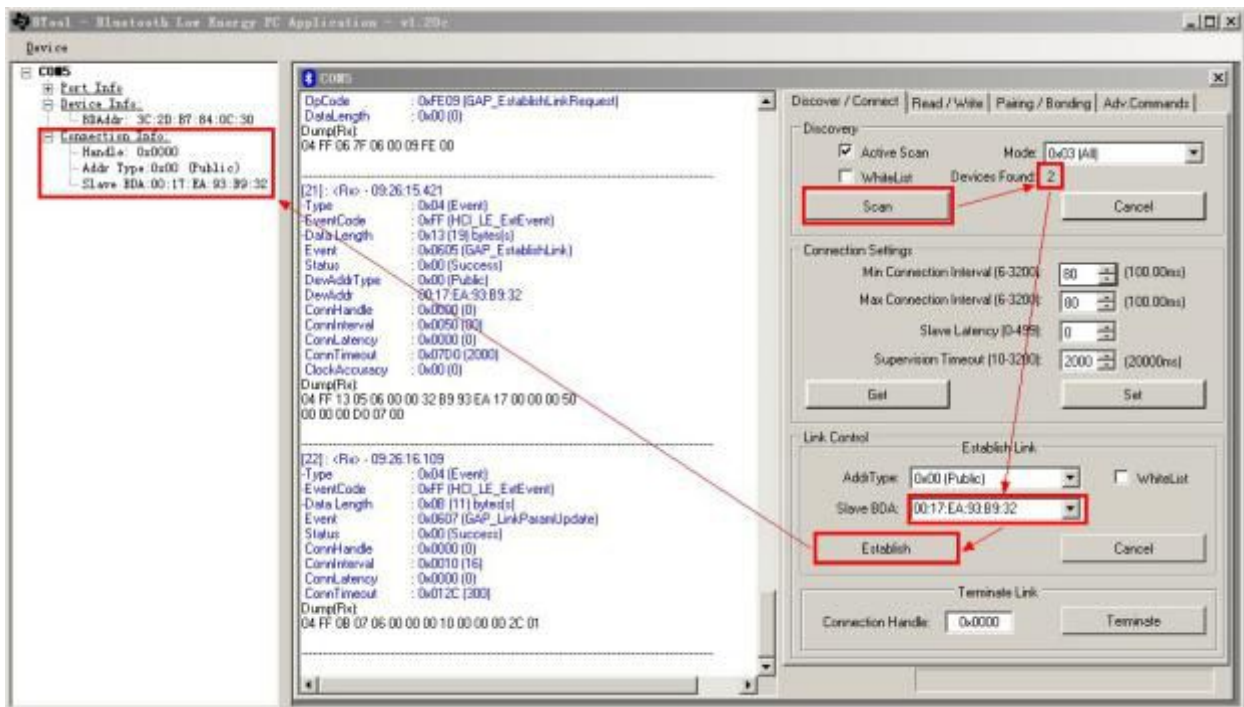
The connection of USB Dongle and the module is basis of communication. The steps of scanning and connecting are as follows:

- 1.Open the engineering file under the directory C: \ Texas Instruments \ BLE - CC254x - 1.2.1 \ Projects \ BLE \ HostTestApp and compile it. Then download it to the USB Dongle;
- 2.Plug the module (3 ~ 3.3 v);
- 3.Ground the module enable pin, and the module starts broadcasting.
- 4.Insert the USB Dongle to a PC USB port, and a UART device will appear in the hardware management (such as: COM5);
- 5.Click Device menu - > New Device, and select the UART which appeared in Step 4. Then choose the default Settings, and click OK;



6. Scan and connect in the order of arrow directions, in which 00:17: ea: 93: b9:32 si the MAC address of the module. Confirm that it is the target module before connecting to it.

7. Afterwards, the information of the module will show in the left column.



That means the connection is successful. Next, it starts testing the direct drive and bluetooth UART forwarding (transparent transmission) functions of the module.

### Testing Direct-Drive function

Any channel within the definition of the protocol can be visited by Btool and Dongle. The typical steps are as follows:

- 1) Find the handle of channel through UUID
- 2) Remember the handle which a channel corresponds to.
- 3) Turn on the channel notify switch by **handle+1**

- 4) Read the channel, using UUID or handle.
- 5) Write the channel, using handle.

If using directly the handles provided in the Characteristic List of BLE Protocol Manual (APP Interface), Step 1 and 2 can be skipped. **Notify switch of channels must be manipulated by handle+1 indicator.**the so-called “channel”here refers to the (**Characteristic**) value. When writing a channel, the byte number must be consistent with the length defined by the Protocol. Or it will be regarded illegal and fail.

**Important Notice:** The key operation with Btool is the 3 steps: first to connect, then reading and writing the handle of certain channel, and turning on the notify switch afterwards. Step 1 & 2 work to find the corresponding handle. But in iOS programming, there is no need to look for the handle. Reading and writing can be done through the channel UUID directly.

Now take the example of ADC to illustrate hwo to use the USB Dongle and Btool to drive directly the bluetooth module.

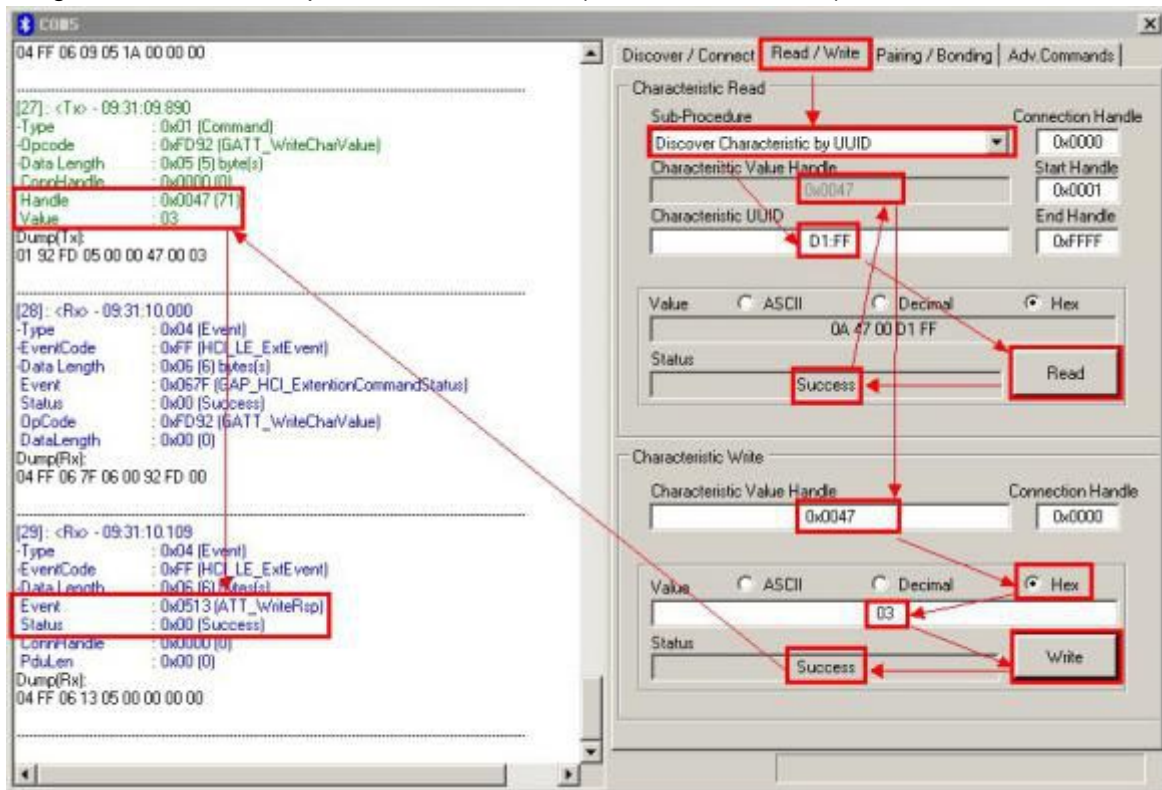
The module type in the following example is RF-Star’s BLE transparent transmission module V2.0. Its handle may be of a little difference from other versions. But the operation is exactly the same. Other testing methods are similar. They just have different channel UUID, but have the same way of reading and writing.

**ADC input (2 Channel) 【service UUID: 0 xffd0 】**

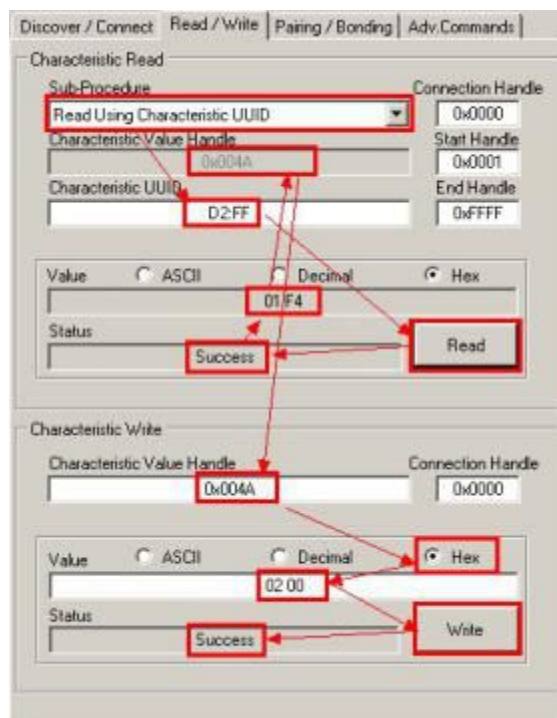
Channel UUID	Executable operation	Bytes	Default value	Remarks
FFD1 (handle: 0x0047)	Read/write	1	0x00	Enable control 0x00:Close the two ADC channels 0x01:Open ADC channels 0x02:Open ADC1 channel  0x03:Open the two ADC channels
FFD2 (handle: 0x004A)	Read/write	2	0x01F4	Sampling cycle ( ms ) ie. 0x01f4 corresponding to 500 ms
FFD3 (handle: 0x004D)	Read/notify	2	0x0000	ADC0 sampling results, maximum value of 0x01fff
FFD4 (handle: 0x0051)	Read/notify	2	0x0000	ADC1 sampling result, maximum value of 0x01fff



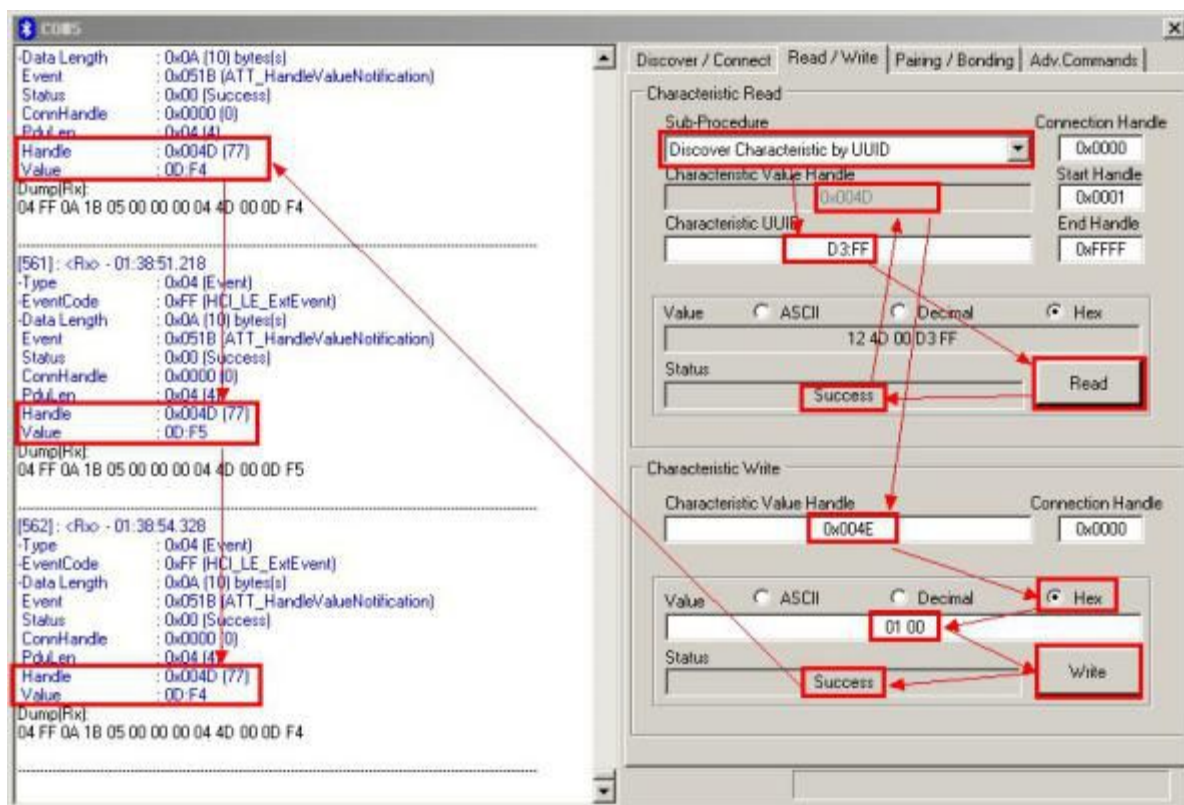
1. Open ADC0 and ADC1. And the subsequent operations can be done in the order of what the red arrow directs (as shown below). First is to find the UUID that needs to be controlled. Input the UUID with low byte in the front (D1:FF) and read. When reading is successful, the handle is got (= 0x0047). Writing 03 to 0x0047 will open ADC0 and ADC1 (see the chart above).



2. Read and reset the sampling cycle. First to read FFD2 channel and get 01 F4, which means the default value is 500 ms. Get the handle = 0x004A and write 0200 (high byte in the front) to the handle. And the the sampling cycle will be set at 512 ms ( 0x0200 = 512 ms).

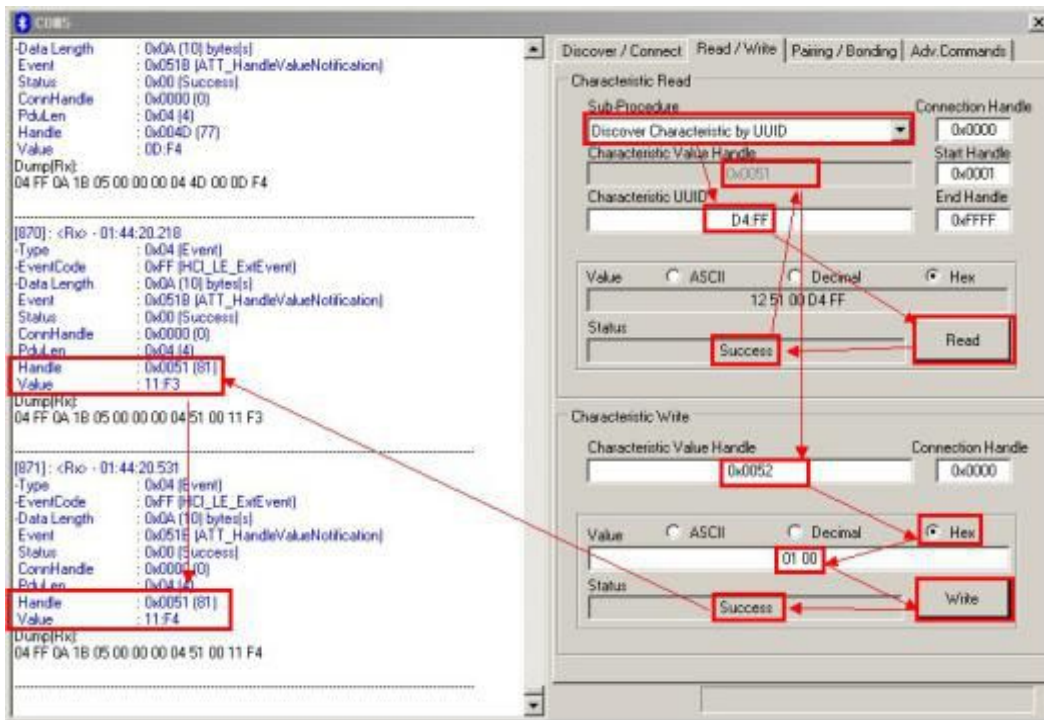


3. Open notify enable switch of ADC0. The address of notify switch is  $handle+1$ ,  $0x004D + 1 = 0x004e$ . Since then, every time when the sample cycle of ADC0 is different from the last time one, there will a new notification.

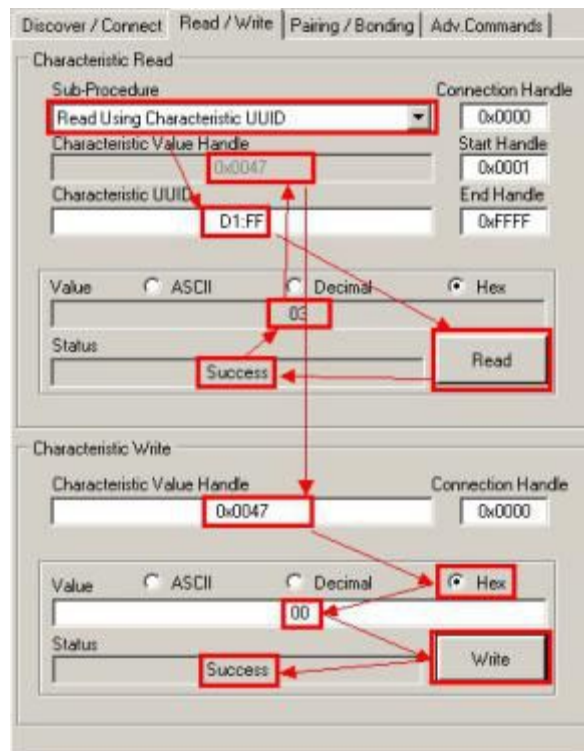


4. Open the notify enable switch of ADC1 channel. The address of notify switch is  $handle+1$ ,

0x0051 + 1 = 0x0052. Since then, every time when the sample cycle of ADC1 is different from the last time one, there will a new notification.



5. Similarly, writing 00 to 0x0047 will stop ADC0 and ADC1.

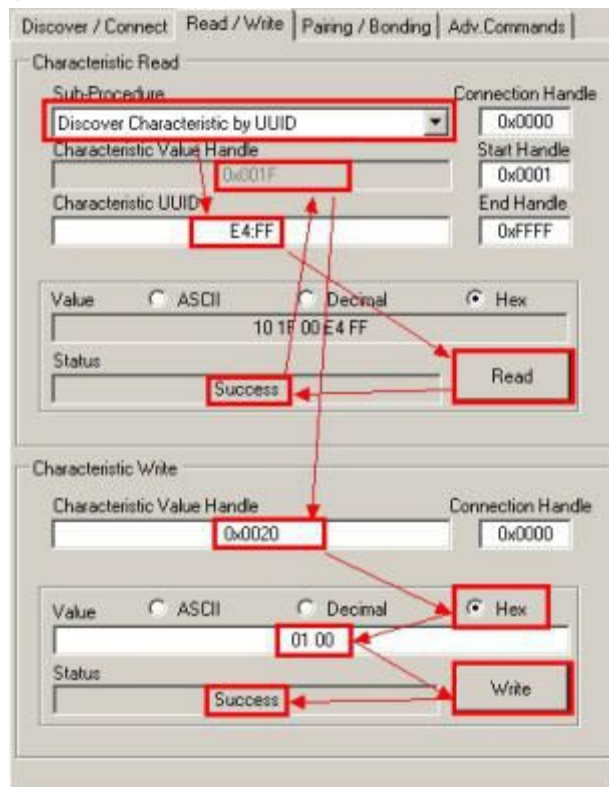


### Testing transparent transmission function

Connect the module to the serial terminal or SOC, in the bridge mode as shown in the diagram, and the bluetooth serial forwarding test can be done.

Note: please the BRTS must be set low, or the serial data cannot be received by module RX.

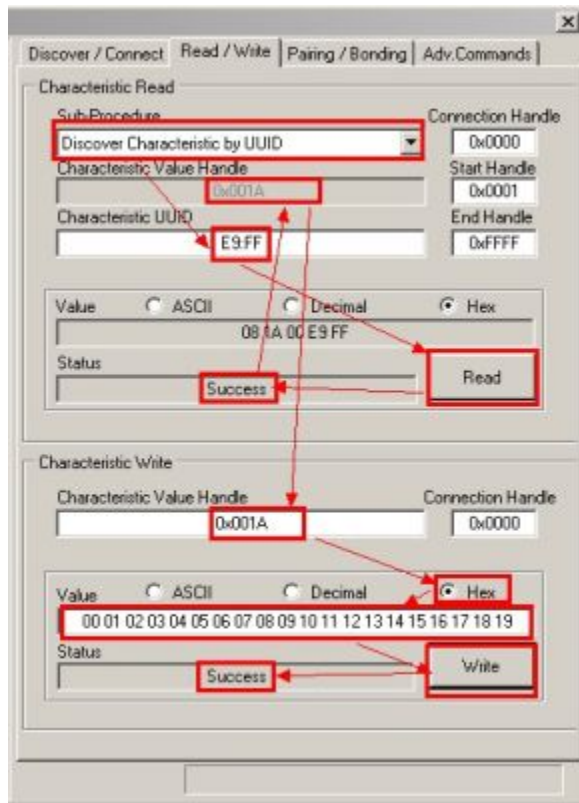
1. The auto-notify switch of serial data channel will be turned on by writing 01:00 to handle: 0x0020, after the BLE module is connected to USB Dongle with Btools (as shown in the diagram below). If the host sends legal data packets to BLE RX, the module will send a notification automatically to Btool, and the detailed data will be shown in the left column. The serial data sent from MCU to the module can be in any length not exceeding 200 bytes.



Serial data channel is used for the module to send data to mobile phones. The UUID of the corresponding characteristic is as follows:

Name	Wireless packet data length	UUID	Handle	Notify Enable Handle
Serial data channel	20 Bytes	0xFFE4	0x001F	0x0020

2. Write data of 1-20 bytes to the module through BTool. When the module receives the data written from mobile phones, it will send the data to MCU through UART. Users can check if the data is correct by reading MCU, or can see the data written through UART assistant. For example, write the data of 7 to the modul through handle 0x001a, as shown in the figure below.



Notes: The data written to the module can be of 1-20 bytes, but no more than 20 bytes. So when programming at mobile phone end, data must be sent in several packets, and the length of each packet can not exceed 20 bytes.

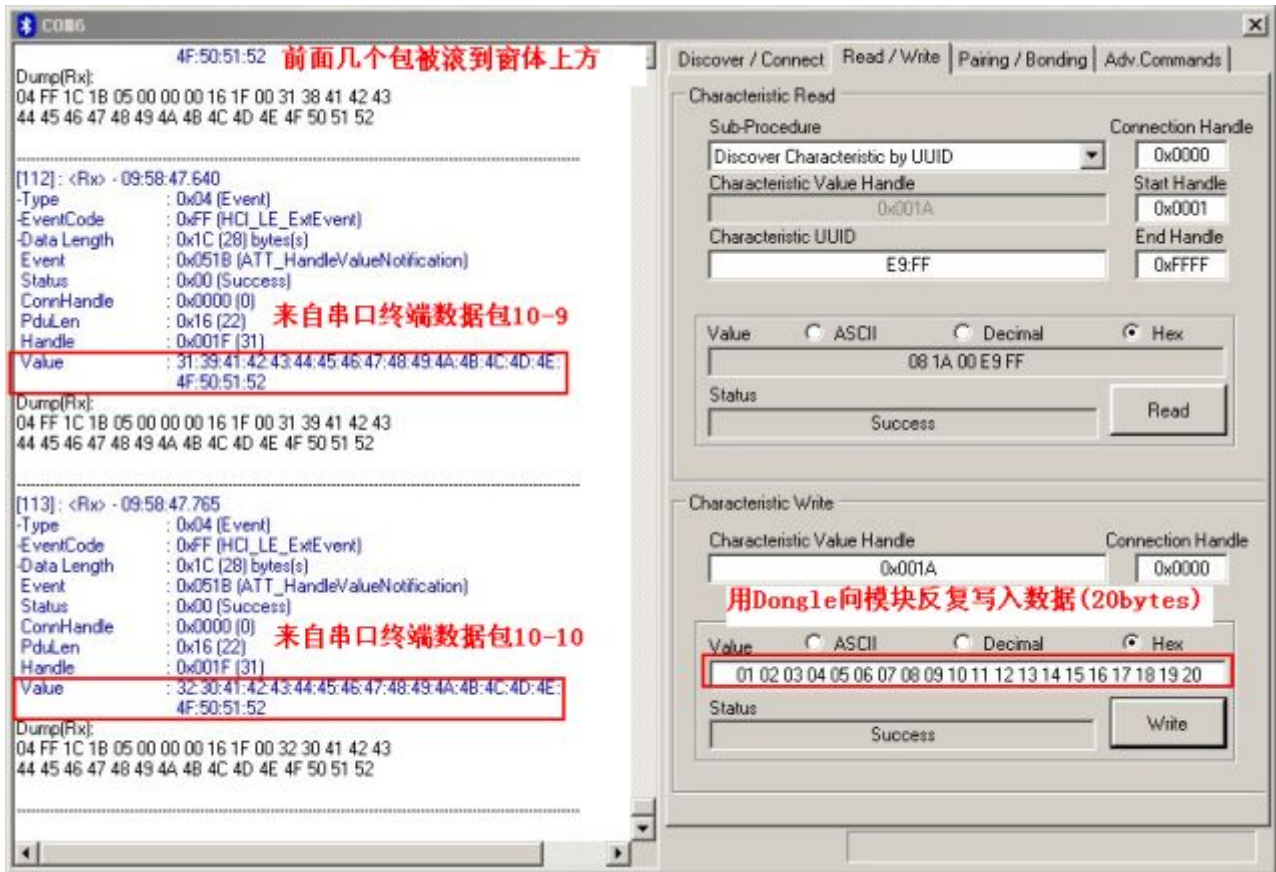
Mobile devices send data to the module through bluetooth data channel, UUID of which is as follows:

Name	Wireless packet data length	UUID	Handle
Bluetooth data channel	20 Bytes	0xFFE9	0x001A

Transparent transmission function test can also be done through UART terminal, by directly connecting to PC UART via level shifting module.

Please refer the screenshots below:

1. Transceiving data with BTool



2.PC terminal connecting to transparent transmission module. Note: BRTS must be set low, or the serial data cannot be received by the module.



## Master reference code (transparent transmission)

Logical relation: The two IOs – BCTS and BRTS – are used for notifying and controlling of transceiving.

These two IOs are normally set high, and triggered when setting low. If the module needs to send data, set BCTS low to inform MCU to receive. If MCU needs to send data, set BRTS low to inform the module to receive. Schematic codes are as follows:

```
void main(void)

    EN = 0 ; //Enable,start broadcast
    while(!BLEMoudleAck("TTM:OK\r\n\r\n0")); //Waiting for Phone scan,Connecting
                                                //Waiting for the connection is
                                                //successful,Also can add wait time limit
                                                //Also can judge connection prompt signal
                                                //level

    BRTS = 0; //Low BRTS, Notice CC2540 module is ready
                                                //to receive

    halMcuWaitMs(2); //Delay for 2ms
    UARTWrite( HAL_UART_PORT_0, "TTM:CIT-100ms", 14);
                                                //Modify connect interval, from the UART
                                                //to be confirmed:

    halMcuWaitMs(5); //Delay for 5ms,and ensure the data has
                                                //been issued

    BRTS = 1; //RTS high and sent
    while(!BLEMoudleAck("TTM:OK\r\n\r\n0")); //Waiting to set success, also can add wait
                                                //time limit

    while(1){ //Loop transceiver test
        while(1){
            if(BCTS == 0){ //Testing,if BCTS low is ready to receive
                while(BCTS==0); //Waiting to be sent, but also timed wait
                if(UARTRead(uartBuffer) == SUCCESS) //USART to read data
                    {... ...} //Service data
            }
            BRTS = 0; //RTS low,and notice CC2540 module is
                    //ready to receive

            halMcuWaitMs(2); //Delay for 2ms
            send_TX("1234567890"); //Send any data (within 200byte)
            halMcuWaitMs(5); //Delay for 5ms,and ensure the data has
                    //been issued

            BRTS = 1; //RTS high, sent
            halMcuWaitMs(20); //Delay again send next packet, and delay
                    //depending on the packet size
        }
    }
}
```

# Contact Us

**SHENZHEN RF-STAR TECHNOLOGY CO.,LTD.**

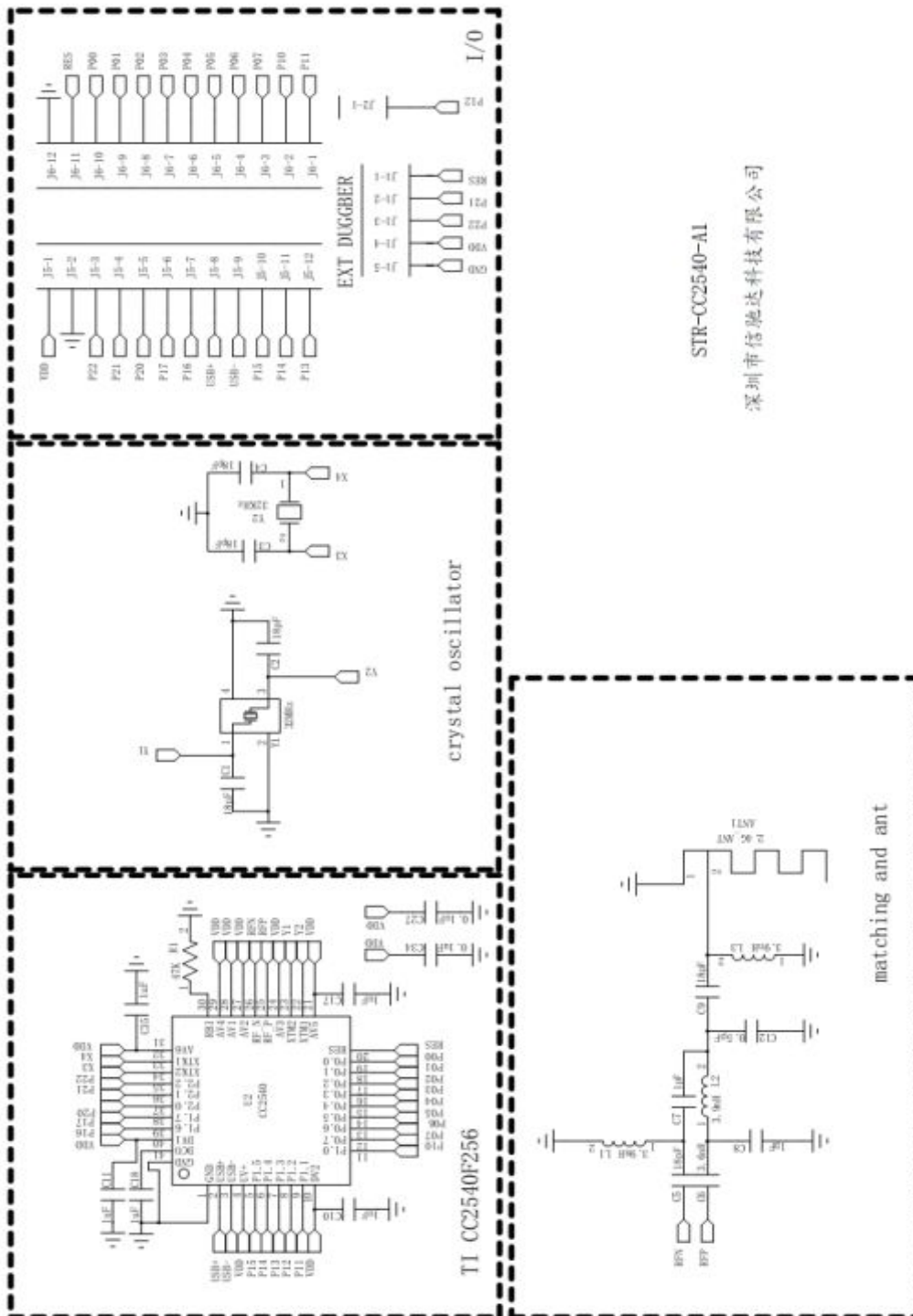
Tel: 0755-8632 9829 Web: [www.szrfstar.com](http://www.szrfstar.com)

Fax: 0755-8632 9413 E-mail: [venus@szrfstar.com](mailto:venus@szrfstar.com)

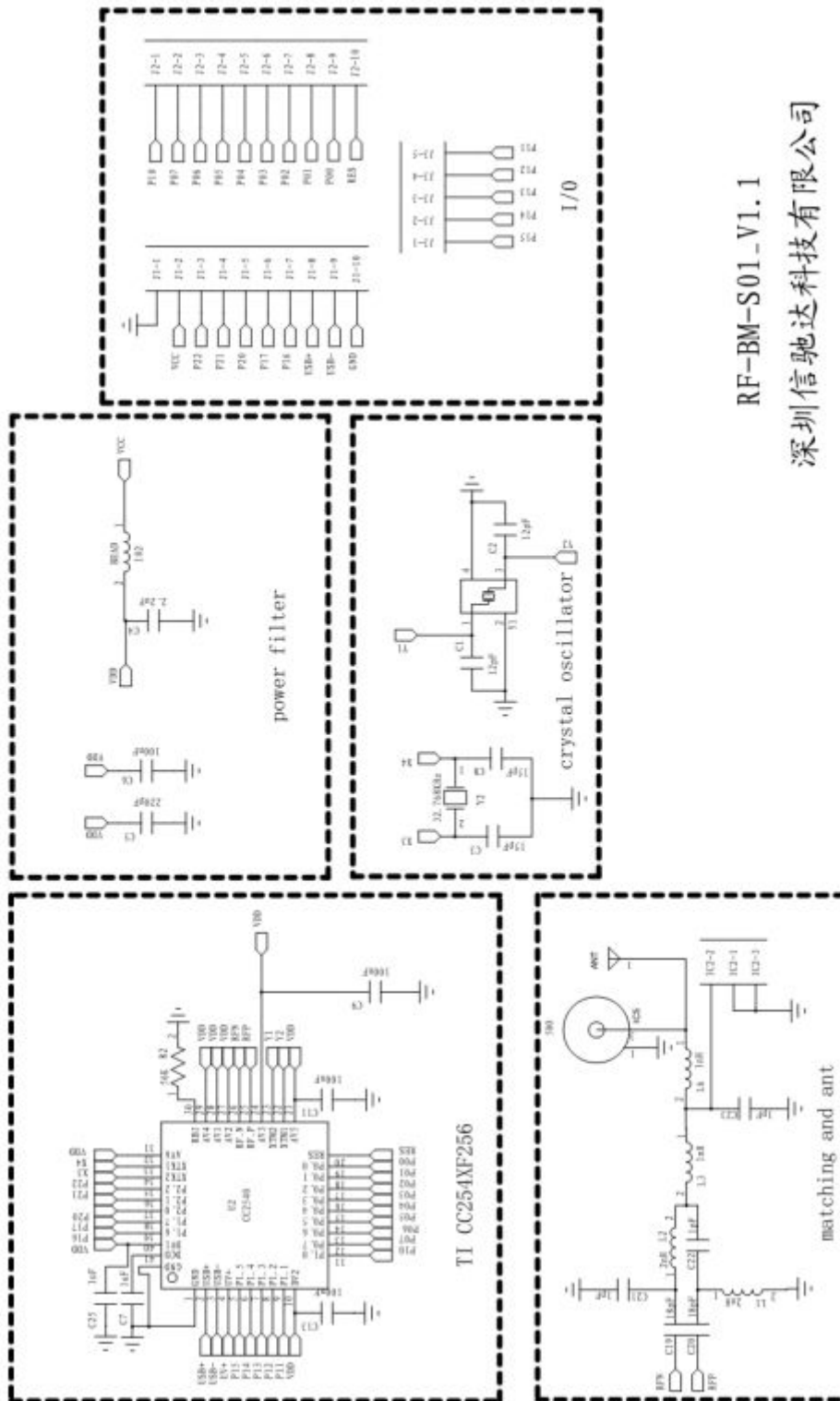
Add: 2F,Block8,Dist.A,Internet Industry Base,Baoyuan Road ,Baoan Dist,Shenzhen

附录 A: 模块原理图

## Appendix B: Schematic Diagram



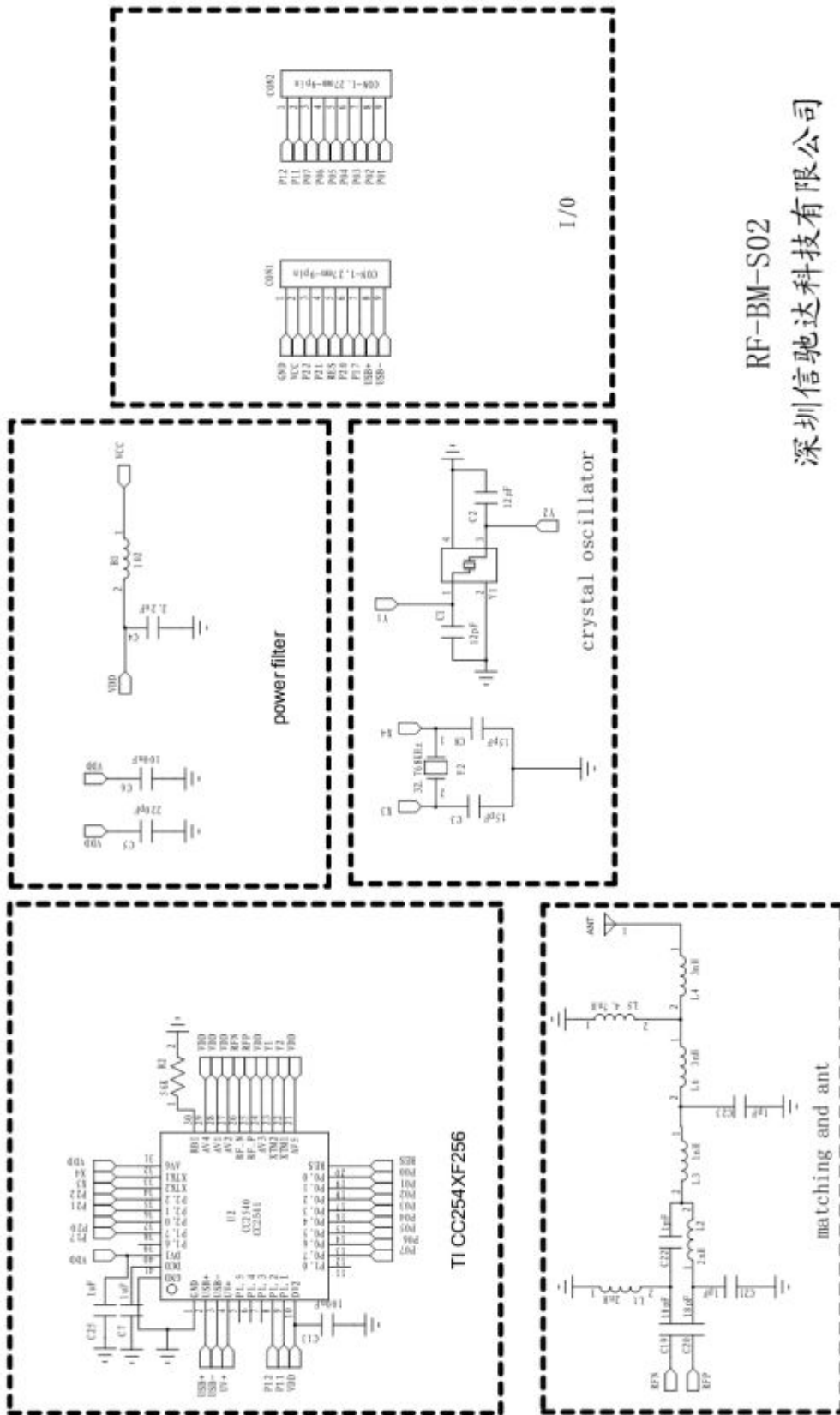




RF-BM-S01\_V1.1

深圳信驰达科技有限公司


RF-BM-S01



RF-BM-S02  
深圳信驰达科技有限公司




RF-BM-S02

**附录 B: FCC 认证证书**  
**Appendix B: FCC Certification**

<b>TCB</b>	<b>GRANT OF EQUIPMENT          AUTHORIZATION</b> Certification Issued Under the Authority of the Federal Communications Commission By:	<b>TCB</b>								
	Nemko Canada Inc. 303 River Road Ottawa, Ontario, K1V 1H2 Canada	Date of Grant: 01/17/2014 Application Dated: 01/17/2014								
ShenZhen RF-STAR Technology CO.,LTD 2F,BLDG.8,Zone A,BaoAn Internet Industry Base, BaoYuan Road,XiXiang, BaoAn DIST, ShenZhen, China										
Attention: Aroo woo										
<b>NOT TRANSFERABLE</b> EQUIPMENT AUTHORIZATION is hereby issued to the named GRANTEE, and is VALID ONLY for the equipment identified hereon for use under the Commission's Rules and Regulations listed below.										
<b>FCC IDENTIFIER:</b> 2ABN2-RFBMS01 <b>Name of Grantee:</b> ShenZhen RF-STAR Technology CO.,LTD <b>Equipment Class:</b> Digital Transmission System <b>Notes:</b> Bluetooth 4.0 (BLE) Module <b>Modular Type:</b> Limited Single Modular										
<u>Grant Notes</u>	<u>FCC Rule Parts</u>	<table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; border-bottom: 1px solid black;">Frequency Range (MHZ)</th> <th style="text-align: center; border-bottom: 1px solid black;">Output Watts</th> <th style="text-align: center; border-bottom: 1px solid black;">Frequency Tolerance</th> <th style="text-align: center; border-bottom: 1px solid black;">Emission Designator</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">15C</td> <td style="text-align: center;">2402.0 - 2480.0</td> <td style="text-align: center;">0.00216</td> <td></td> </tr> </tbody> </table>	Frequency Range (MHZ)	Output Watts	Frequency Tolerance	Emission Designator	15C	2402.0 - 2480.0	0.00216	
Frequency Range (MHZ)	Output Watts	Frequency Tolerance	Emission Designator							
15C	2402.0 - 2480.0	0.00216								
Limited modular approval. Power output listed is conducted.										
										

# 附录 C : RoSH 认证证书

## Appendix C: RoHS Certification

		
<h3>TEST REPORT</h3>		
REPORT No.: <b>BTS201312302663</b>	Date: 2014-01-09	Page 1 of 8
Client Company: Shenzhen RF-STAR Technology Co., LTD		
Client Address: 2F, BLDG.8, Zone A, BaoAn Internet Industry Base, BaoYuan Road, XiXiang, BaoAn DIST, ShenZhen		
Report on the submitted samples said to be:		
Sample Name	Bluetooth 4.0 (BLE) Module	
Style/ Item No.	RF-BM-S01; RF-BM-S01_V1.1	
Product Rating	DC3.3V, Max40mA; Max0.132w, 2483.5MHz	
Operating Frequency Range	2402-2483.5MHz	
RF Output Power	-23dBm-4dBm	
Sample Receiving Date	December 30, 2013	
Testing Period	December 30, 2013 to January 09, 2014	
Results	See next pages	
-----		
Summary of Test Results:		
<u>TEST REQUEST</u>		<u>CONCLUSION</u>
A EU RoHS Directive 2011/65/EU and its amendment directives		Pass
-----		
Signed for and on behalf of BTS		
 Liping_Xu		
<small>This document is issued by the Company subject to its General Conditions of service of service printed overleaf, available on request for electronic format documents, subject to terms and conditions for electronic documents. It is drawn to the attention of liability indemnification and jurisdiction issues defined therein, any holder of the document is advised that information contained herein reflects the company's findings at the time of its intervention only and within the limits of client's instructions. If any the company's sole responsibility is to its client and this document does not incorporate parties to a transaction from exercising all their rights and obligations under the transaction documents this document cannot be reproduced except in full without prior written approval of the company, any unauthorized alteration, forgery or falsification of the content or appearance of this document is unlawful and offenders may be prosecuted to the fullest extent of the law unless otherwise stated the results shown in this test report refer only to the samples tested.</small>		
<small>Shenzhen Technology Service Co., LTD (BTS) 深圳市艾思特技术服务有限公司</small>		<small>21st of General Building, BaoAn Hi-Tech Industry Park, Linggang District, Shenzhen, City, Guangdong Province, P.R.China 中国广东省深圳市宝安区西乡街道西乡高新技术产业园综合楼2118#</small>
<small>传真 / fax: 0755-89589101    chazun@bestow.cn</small>		<small>邮编 / post code: 518129    电话 / telephone: 0755-89500120-8015</small>

附录 D : 蓝牙 BQB-EPL 证书

Appendix D: End Product Listing



# EPL Bluetooth® End Product Listing

## The Bluetooth SIG Hereby Recognizes

**ShenZhen RF STAR Technology CO.,LTD.**

Member Company

**RF-BM-S01\_v1.1**

Qualified Design Name

Qualified Design ID(s): **B016552**

Contact Person: **Aroo Wong**

Series: **V1.1**

Publish Date: **13 November 2013**

EPL Type: **Other**

This certificate acknowledges the Bluetooth® Specifications declared by the member were achieved in accordance with the Bluetooth Qualification Process as specified within the Bluetooth Specifications and as required within the current PRD

