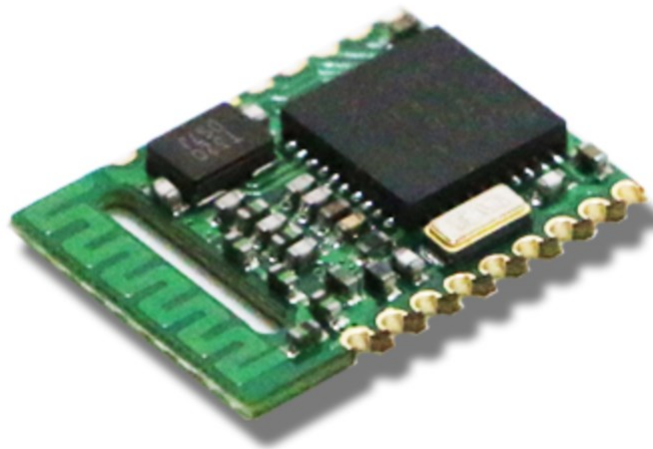


CC2541 Bluetooth Low Energy (BLE) Module and Protocol

(Transparent Transmission + Direct Driving)

Version: V2.3



Contents

Contents	2
1. Summary	3
2. Schematic diagram of working mode	6
3. Package and Pin Assignment.....	6
4. UART Transparent Transmission Protocol.	8
5. UART AT Command.....	11
Connection Interval Configuration	11
Module Rename	11
Baud Rate Configuration.....	12
Acquire MAC Address	12
Module Reset	12
Broadcast Cycle Configuration	12
Add Customized Broadcast Packet.....	13
Product ID Definition.....	13
Transmit Power Configuration.....	13
Data Delay Configuration.....	14
6. AT Command List.....	15
7. Broadcast Data Configuration	17
8. System Reset and Recovery.....	18
9. IOS APP Programming Reference	19
10. BLE Protocol (APP Interface).....	21
Bluetooth Data Channel [Service UUID:0xFFE5].....	21
Serial Port Data Channel [Service UUID:0xFFE0].....	21
PWM Output (4 Channel) [Service UUID:0xFFB0].....	22
ADC Input (2 Channel) [Service UUID:0xFFD0]	24
Programmable IO (8- Channel) [Service UUID:0xFFFF0].....	25
Timed Rollover Output (2 channels) [Service UUID: 0xFFFF0]	27
Level pulse width count (2 channels) [Service UUID:0xFFFF0].....	29
Anti-Hijacking Password [Service UUID:0xFFC0]	30
Battery power report [Service UUID:0x180F]	31
RSSI report [Service UUID:0xFFA0]	32
Module Parameter Configuration [Service UUID:0xFF90].....	33
Device Information [Service UUID:0x180A]	38
Port timing event configuration [Service UUID:0Xfe00]	39
11. Test the Transparent Transmission function with APP.....	45
12. Test with USB Dongle and Btool	46
Connect BLE module.....	46
Test the direct-drive function	47
Test the Transparent Transmission function	52
13. Host reference code (Transparent Transmission).....	55
14. Application and Implementation.	56
15. Trouble shooting.	57
16. Contact us.....	59

1. Summary

DL-CC2541 is a low-power Bluetooth 4.2 module. This BLE module can work in bridge mode (transparent transmission mode) and direct drive mode, it features an universal serial port design, full duplex bidirectional communication. After the module is started, it will automatically broadcast, and the mobile phone that has opened a specific APP will scan and dock it. After success, it can be monitored through the BLE protocol.

In the bridge mode, the user's CPU can communicate with the mobile device through the UART port of the module, and the user can also manage and control certain communication parameters through the specific serial port AT command. The specific meaning of user data is defined by the upper-layer application program. The mobile device can write to the module through the APP, and the written data will be sent to the user's CPU through the serial port. After the module receives the data packet from the user's CPU serial port, it will automatically forward it to the mobile device. In this mode of development, the user must be responsible for the code design of the main CPU and the intelligent mobile APP code design for mobile devices.

In the direct drive mode, the user performs simple peripheral expansion of the module. The APP directly drives the module through the BLE protocol to complete the supervision and control of the module by the intelligent mobile device. In this mode of software development, the user is only responsible for the design of APP codes on the smart mobile device.

Features

1. Simple to use, without any Bluetooth protocol stack application experience;
2. The user interface uses a universal serial port design, full duplex bidirectional communication, and the minimum baud rate supports 4800bps;
3. Support bridge mode (serial port transparent transmission) or direct drive mode (no additional CPU required);
4. Default 20ms connection interval, fast connection;
5. Support AT command software reset module to obtain MAC address;
6. Support AT commands to adjust the Bluetooth connection interval and control different forwarding rates. (Dynamic power adjustment);
7. Support AT command to adjust the transmission power, modify the broadcast interval, customize the broadcast data, customize the device identification code, set the data delay (user CPU serial port reception preparation time), modify the serial port baud rate, modify the module name, all will be saved after power off;
8. The length of the serial data packet can be any length below 200 bytes (including 200). (Automatic distribution of large packages);
9. High-speed transparent transmission and forwarding, the fastest up to 4K/S, can work stably in 2.5K-2.8K (IO5, IO6);

10. Support mobile device APP to modify the module name, save after power off, modify the serial port baud rate, product identification code, custom broadcast content, broadcast cycle, all will be saved after power off;
11. Support mobile device APP to reset the module remotely and set the transmission power;
12. Support mobile device APP to adjust the Bluetooth connection interval. (Dynamic power adjustment);
13. Full IO expansion including debug port;
14. Support connection status, broadcast status prompt pin / general IO flexible configuration;
15. Six bidirectional programmable IO, external interrupt trigger input detection, full low power operation. (Trigger alarm, lighting control, remote control toys, etc. various input and output switching applications);
16. Two programmable timed single/cycle inversion output ports. (Smart appointment timing application);
17. Two ADC inputs (14bit), enable/disable, free configuration of sampling period. (Temperature and humidity, photometric and other applications);
18. Four programmable PWM (120Hz) output. (Applications such as dimming and speed control);
19. The RSSI of the module end is continuously collected, readable and can automatically notify the APP, enable/disable, and the collection frequency can be set freely. (Object tracking, Anti-lost alarm application);
20. Support module power prompt, power reading, automatic report. (Device power reminder);
21. Support anti-hijacking password setting, modification and recovery to prevent malicious connections from third parties. It can also not be used. Independent password operation result notification to facilitate APP programming;
22. Support single foot position (long press) for 5 seconds to restore factory settings, APP remote restore factory settings;
23. Support customizing the PWM output initialization state (full high, full low, PWM output state value before power down);
24. Support custom PWM frequency ($61.036 \text{ Hz} \leq f \leq 8 \text{ kHz}$, default 120Hz);
25. Real-time system status of broadcast content prompt module, including battery power, custom device identification code, four-way PWM current output value or acquisition value of two ADCs, current IO status, etc.; (suitable for broadcast application solutions);
26. Two-level pulse width counting, $0 \sim 0\text{xFFFFFFFF ms}$ (about 49.7 days);
27. Support internal RTC real-time clock, APP terminal can be synchronized at any time;
28. Support 6-channel IO and 4-channel PWM timing control, this feature is disabled by default;
29. Four-way PWM supports gradient mode (suitable for dimming effect control);
30. Support IO configuration and output state saving function, can customize the default initialization state;
31. Support light recovery and deep recovery modes, flexible recovery of user data, while the reserved product must be configured;

32. Support to get the Bluetooth connection status (connection, normal disconnection and timeout disconnection) string prompt from the TX serial port;
33. Support low level enable mode and pulse width enable mode, support remote shutdown;
34. In pulse enable mode, it supports 30 seconds of automatic shutdown without connection;
35. Under pulse enable mode, support square wave alarm to prompt connection timeout (disconnection)
36. Very low power consumption under standby mode, sleep current from official datasheet (TI-CC2540 chip) is 0.4uA, the measured power consumption of the module is as follows:

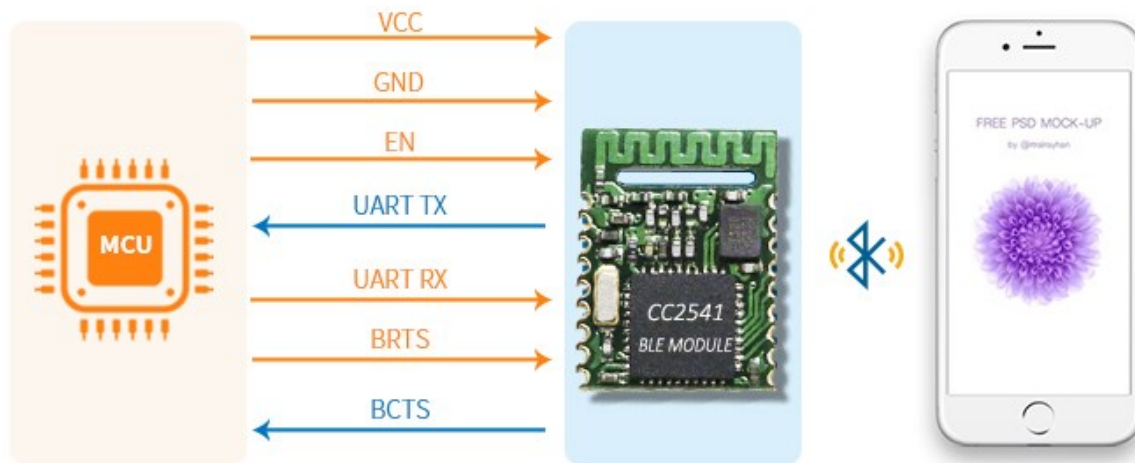
Event	Average Current (Integral Measured) *1	Average Current (Meter Measured) *2	Duration	Test Conditions/Remarks
Sleeping	0.36uA	0.3-0.41uA	—	EN disconnected
Broadcasting	205uA	0.15~0.55mA	3.86ms	Broadcast cycle 250ms
Connecting	245uA	0.41mA	2.24ms	Connection cycle 100ms
Single BLE Data Receiving	333uA	0.63mA	3.0ms	(20bytes,10 Times/Sec.)
Receives data and transmits via UART serial port	498uA	2.69mA	5.2ms	(20bytes,10 Times/Sec.)
Single BLE Data Transmitting	345uA	0.68mA	3.2ms	(20bytes,10 Times/Sec.)

*1 Note: Test method: Connect a 10Ω resistor in series with the power circuit, use an oscilloscope to check its voltage drop waveform, and perform integral calculation.

*2 Note: Multi-meter test method: connect a multi-meter (set at μA or mA level) in series between the battery and the module to check the value shown, with the test voltage of 3.07 V.

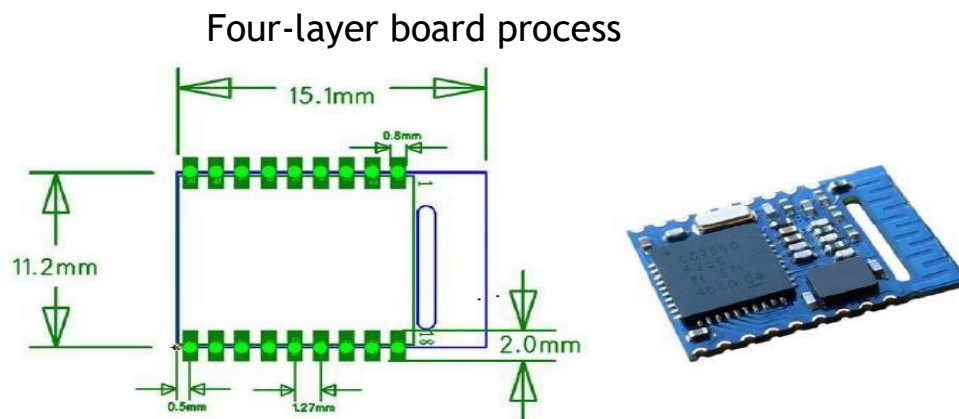
The above data is sampled and measured data, for reference only. If the lower power consumption is expected, connection interval or broadcast cycle can be appropriately increased, as shown in the chapters of "Module Parameter Settings" and "UART AT Commands"

2. Schematic diagram of working mode



Note: to avoid the large current caused by the output level difference between the IO of the user's CPU and the IO of the module, it is recommended to connect a small isolation resistor in series on the output signal line TX and BCTS of the module.

3. Package and Pin Assignment



Pin No.	Name	Chip Pin	I/O	Description
Pin1	GND	GND	-	Ground
Pin2	VCC	VCC	-	Module power supply positive 2V-3.6V
Pin3	IO7	P2.2	O	Output port (can be flipped regularly) / sleep status indicator
Pin4	IO6	P2.1	O	Output port (can be flipped regularly)/connection status indication (low level, or square wave prompt, See the chapter "Module Parameter Setting" for details)
Pin5	RES	RST	I	Module reset, active low

Pin6	EN	P2.0	I	<p>Module enable control line, the default is level trigger mode</p> <p>Level trigger mode, active low, with internal pull-up.</p> <p>0: The module starts broadcasting until it connects to the mobile device</p> <p>1: Regardless of the current state of the module, immediately enter a full sleep state (0.4uA)</p> <p>Pulse trigger mode, every time a pulse is received (W>200ms), the module will switch cyclically between power-on (broadcasting, allowing discovery and connection) and power-off (full sleep state)</p> <p>(For mode switching, please refer to the relevant chapters of "Module Parameter Settings")</p>
Pin7	IO5	P1.7	I/O	<p>Programmable bidirectional IO, which can be set as input or output through BLE protocol</p> <p>When used as input, it can be used as level pulse width count input terminal</p>
Pin8	U+	USB+	I/O	CC2541 pin USB+, not used
Pin9	U-	USB-	I/O	CC2541 pin USB+, not used
Pin10	REST ORE /IOO	P1.2	I/O	<p>Factory reset trigger or programmable bidirectional IO</p> <p>Within 30 seconds after power-on, keep this pin low for 5s, the system will restore some parameters (light recovery), if it is maintained for more than 20s, all parameters will be restored</p> <p>(Deep Recovery) (see the chapter of "System Reset and Recovery")</p> <p>30 seconds after power-on, it can be used as normal IO and can be used through BLE protocol</p> <p>(See "Programmable IO (8 channels) [Service UUID: 0xFFFF0]") Set to input or output use</p>
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 a data sending request (used to wake up the module)</p> <p>0: The host sends data, and the module will wait to receive data from the host. At this time, the module will not sleep.</p> <p>1: The host has no data to send, or after the host sends data, this signal line should be set to 1</p>
Pin15	BCTS	P0.4	O	<p>Data input signal (used to wake up the host, optional)</p> <p>0: The module has data sent to the host, and the host receives the module data</p> <p>1: The module has no data sent to the host, or after the module data has been sent, this signal will be set to 1</p>
Pin16	TX	P0.3	O	Module serial port TX
Pin17	RX	P0.2	I	Module serial RX
Pin18	ADC1	P0.1	I	Analog quantity acquisition, channel 1

Note: Because it is a simplified version with compact size, some IOs are not exported, and the corresponding functions cannot be used.

4. UART Transparent Transmission Protocol

The bridge mode means to set up a bi-directional communication between user CPU and mobile devices by connecting the module with user CPU through serial port. Users can re-set serial port baud rate and BLE connection interval by the specified AT commands (see details in “AT Command”). The module will have different data TX & RX capability, as per different serial port 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 serial port baud rate at the high speed of 115200 bps. Configuration can be saved after power-off

When the BLE connection interval is 20ms and the serial port baud rate is at 115200 bps, the module has the highest transmit ability in theory (4 K/s). Take the configuration in the level-enabled mode as an example, UART transparent transmission protocol will be introduced in detail as below.

The module can transmit a packet of the maximum 200-Byte through serial port 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 automatically (into 1 ~ 20 Bytes/packet) before sending. The module will transmit them to the master RXD in turn, when received the packets.

1. Hardware protocol of serial port: 115200 bps, 8, no parity, 1 stop bit.
2. When EN is set at high level, the Bluetooth module is in full sleep mode. When EN is set low, the module will start broadcast at the interval of 200 ms, until it pairs with mobile devices. When EN jumps from low to high, the module will enter sleep mode immediately, regardless of the current status.
3. After the module is connected, BRTS needs to be pulled low if the master (MCU) has data to send to the BLE module, and the data transmission can be started around 100ms afterwards. BRTS should be pulled high by the master after transmission finished and make the module exit the serial RX mode. Pay attention to confirming that the data transmission has been completely finished before BRTS pulled-high. 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 500μs afterwards. And this delay can be configured through the AT command (see details in "UART AT command"). BCTS will be set high by the module when data transmission is finished.

5. If the master BRTS is being kept at a low level, the Bluetooth module will always be in RX mode and the power consumption will be high.
6. After the module is connected, a string of "TTM:OK\r\n0" will be printed from TX. The string could be used to confirm whether the normal transmit operation is done. Of course, the connection status prompt pin can be used instead. Also, the connection can be checked by sending a specific confirmation string to the module from mobile devices. When APP automatically disconnects the module, there will be a string "TTM:DISCONNET\r\n0" from TX. If the disconnection is abnormal, the string will be "TTM:DISCONNETFORTIMEOUT\r\n0".
7. The default Bluetooth connection interval is 20 ms. If low-speed TX mode is needed for saving power, connection interval must be adjusted by AT command (the maximum connection interval to be 2000 ms). 80-Bytes is maximum transmission length for each interval. Set the connection interval as T (unit: ms), and the highest transmit rate per second V (unit: Byte/s) is as follows:

$$V = 80 * 1000 / T$$
(V is only relevant with T)
 If the Bluetooth connection interval of the module is 20 ms, and 80-Bytes is maximum transmission length for each interval, the theoretical maximum transmission capacity (transmit rate) will be $80 * 50 = 4K$ Byte/s. Tests have shown that the packet loss is very little when transmit rate under 2 K/s. For safety's sake, it is suggested to do check-sum and re-transmission processing in the upper layer, no matter for high or low speed transmit applications.
8. Here is an example of the communication with 20ms connection intervals in below. Configuration can be set by yourself. But the lower the transmit rate V0, the less packet leakage is.

Communication Mode	BLE Connection Interval T (ms)	Highest Theoretical Transmit Rate V (Byte/s) $V=80*1000/T$	Serial Data Packet Length (Byte)	Serial Port Transmission Interval TS (ms) When $L<80$, $TS \geq T$ When $80<L<160$, $TS \geq T*2$ When $160<L<200$, $TS \geq T*3$	Actual Transmit Rate V0(byte/s) $V0=L*1000/TS$	Remarks
1	20	4K	80	$TS \geq T$, If $TS=20ms$	$80*1000/20=4K$	Too low TS, not recommended
2	20	4K	200	$TS \geq T*3$, If $TS=70ms$	$200*1000/70=2.8K$	
3	20	4K	200	$TS \geq T*3$, If $TS=80ms$	$200*1000/80=2.5K$	
4	20	4K	80	$TS \geq T$, If $TS=35ms$	$80*1000/30=2.6K$	
5	20	4K	70	$TS \geq T$, If $TS=30ms$	$70*1000/30=2.3K$	
6	20	4K	60	$TS \geq T$, If $TS=30ms$	$60*1000/30=2K$	
7	20	4K	40	$TS \geq T$, If $TS=30ms$	$40*1000/30=1.3K$	
8	20	4K	20	$TS \geq T$, If $TS=30ms$	$20*1000/30=666byte$	

Note: Specific communication mode can be designed according to the practical application. Packet length of serial port can be designed between 80 Bytes and 200 Bytes (large packet transmission). According to BLE protocol, the formulas are as follows:

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

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

When $160 < L < 500$, $TS \geq T * 3$,

Transmission modes that comply with the above-said conditions are generally safe in operation. However, among them, when $TS = T$, $TS = T * 2$ or $TS = T * 3$, it is workable, but high package loss will be caused. It is recommended to add check-sum re-transmission mechanism. In other words, when a serial port data packet is as big as $80 \text{ Byte} < L < 200 \text{ Bytes}$, the data can be sent to the module for one time, but certain time needs to be spared for module data transmission by Bluetooth. Otherwise there will be a rear-end data collision. For example, when the connection interval $T = 20\text{ms}$, if the data packet length $L = 200$, the TS must larger than $T * 3 = 60\text{ms}$. So, setting $TS = 70\text{ms}$ is a reasonable choice.

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

Note: Test shows that in iOS, calling the writing function to Characteristic with the parameter “CBCharacteristicWriteWithResponse” (writing mode with response) will reduce partially the transmit efficiency, but the correctness of a single packet will be ensured. While with the parameter “CBCharacteristicWriteWithoutResponse” (writing mode without response), the transmit efficiency will be increased, but the correctness of data packet needs to be checked by APP in upper layer.

5. UART AT Command

The string beginning with "TTM" will be parsed and executed as an AT command, and returned from the serial port as it is, after which the output execution result will be added, "TTM:OK\r\n\0" or "TTM:ERP\r\n \0" etc. Serial data packets that do not start with "TTM" will be regarded as transparent data.

● Connection Interval Setting

Input the following string to the serial port RX to set the BLE connection interval:

"TTM:CIT-Xms"

Where X="20", "50", "100", "200", "300", "400", "500", "1000", "1500", "2000", and the unit is ms. After executing this command, you will get the following confirmation from the serial port TX:

"TTM:TIMEOUT\r\n\0" means that the change has timed out and the modification failed;
"TTM:OK\r\n\0" means that the change is successful, and it is running at a new connection interval;

The success of this connection interval setting depends on the mobile device's restriction on the connection interval, and the maximum connection interval is different for different IOS versions. Using iPhone4s (IOS5.1.1) test, the fastest support 20ms, the slowest support 2s, in addition, due to the internal mechanism of the BLE protocol, this command will have different execution efficiency under different connection intervals. In IOS5.1.1, when the current connection interval is 2000ms (the longest 2000ms), it may take about 100s to wait for the longest connection interval (such as 100ms) when changing to other connection intervals. Executing this AT command will have a quick execution efficiency.

Note: This connection interval will not be saved when the power is off, and the change command is only valid after the connection is successful.

● Module Rename

Input the following string to the serial port RX, with module name behind "-", and the length shall be within 16 bytes,

"TTM:REN-"+Name

It will also receive "TTM:OK\r\n\0" confirmation from TX. If the command format is incorrect, it will return:

"TTM:ERP\r\n\0"

The test shows that due to the IOS version, the device name modification can be changed immediately in IOS6 and above. This name is saved when power off.

● Baud Rate Configuration

Input the following string to the serial port RX, the parameter after-is the new baud rate, see AT command list, such as:

"TTM:BPS-115200"

Afterwards, you will receive "TTM:OK\r\n\0" confirmation from TX. If the set value is not in the option or the command format is incorrect, it will return:

"TTM:ERP\r\n\0"

Tests show that in IOS5, the device name modification cannot be successful, but it can be changed immediately in IOS6. The user can set it through the PC and use it, or set it through the BLEAPP interface of the mobile device. See "Module parameter setting [Service UUID: 0xFF90]".

● Acquire MAC Address

Input the following string to the serial port RX: "TTM:MAC-?"

Will receive from TX: "TTM:MAC-xxxxxxxxxxx\r\n\0"

"Xxxxxxxxxxxx" after the string is the 6-byte Bluetooth address of the module.

● Module Reset

Input the following string to the serial port RX: "TTM:RST- SYSTEMRESET"

Will force the module to soft reset once.

● Broadcast Cycle Configuration

Input the following string to the serial port RX to set the broadcast period of the module, $T=X * 100ms$

"TTM:ADP-(X)"

X= One of "2", "5", "10", "15", "20", "25", "30", "40", "50". It will receive "TTM:OK\r\n\0" confirmation from the TX pin.

If the command format is incorrect, it will return: "TTM:ERP\r\n\0"

The broadcast cycle setting is saved when power off. After restarting the module, the module will broadcast according to the new broadcast cycle.

● Add Customized Broadcast Packet

Enter the following string to the serial port RX to customize the broadcast content
"TTM:ADD-"+Data

Among them, "Data" is the data to be added to the broadcast, and the length is $0 < L \leq 16$.

It will receive "TTM:OK\r\n\0" confirmation from the TX pin.

If the command format is incorrect, it will return: "TTM:ERP\r\n\0"

This command will take effect immediately after it is set. You can broadcast some customized content through this function, and the data will be saved after power-off.

If it is set to 16 all 0 data, it is considered that the custom broadcast data is not used, but the default broadcast content is used.

● Product ID Definition

Enter the following string to the serial port RX to customize the broadcast content:

"TTM:PID-"+Data

Among them, Data is a two-byte product identification code, ranging from 0x0000 to 0xFFFF ($L=2$). Will receive from TX pin confirm with "TTM:OK\r\n\0".

If the format of the command is incorrect, it will return: "TTM:ERP\r\n\0"

This identification code will be saved after power failure and will appear in the broadcast. You can use this Filter equipment or determine whether it is a specific product.

● Transmit Power Configuration

Input the following string to the serial port RX to set the corresponding transmit power in dBm.

"TTM:TPL-(X)"

Where $X = "+4", "0", "-6", "-23"$; will receive "TTM:OK\r\n\0" confirmation from the TX pin,

and the module will immediately use the new transmit power to perform communication,

If the instruction format is incorrect, it will return: "TTM:ERP\r\n\0"

Note: This parameter will not be saved after power off.

● Data Delay Configuration

Input the following string to the serial port RX to set the delay between low level output of BCTS and TX data output (in ms). Where X = "0", "2", "5", "10", "15", "20", "25" (all data format is in ASCII code):

"TTM:CDL-Xms"

For example: "TTM:CDL-2ms" means the delay time is 2 ms.

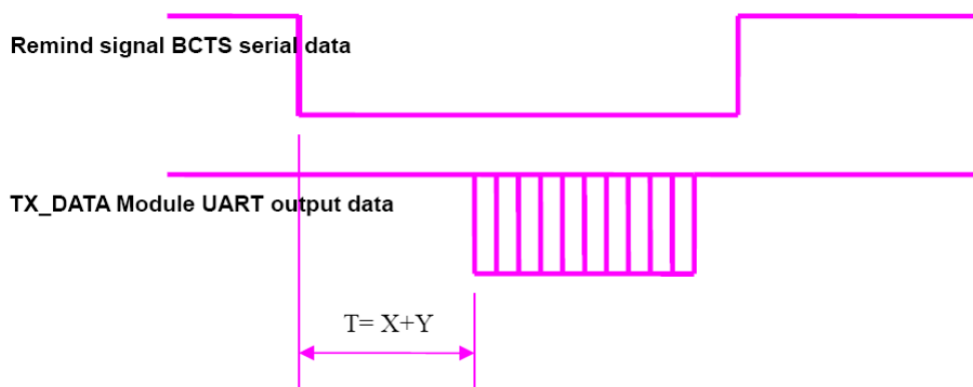
After the command is executed, the following confirmation will be got from TX:

"TTM:OK\r\n\0": It means the successful operation.

"TTM:ERP\r\n\0": It means the incorrect command format.

To make the user CPU have enough time to wake-up from sleep mode and ready to receive data, the module is provided this delay (X) configuration. The BRTS will be set low before there is data to be sent through the serial port, while the delay time between low level output of BCTS and TX data output will be set by this parameter. The actual delay (T) will be $T = (X + Y)$ ms, if the minimum delay is no less than X, while $500 \mu s < Y < 1 \text{ ms}$.

This configuration can be saved after power-off. The scheme of data delay configuration is as follows:



Scheme of Data Delay Configuration

6. AT Command List

AT Command	Saved after Power-off	Parameter Description	Possible Response	Remarks
"TTM:CIT-Xms" (Only effective after successful connection)	No	X="20", "50", "100", "200", "300", "400", "500", "1000", "1500", "2000" Set the corresponding BLE connection interval in ms	"TTM:TIMEOUT\r\n0" " "TTM:OK\r\n0" "TTM:ERP\r\n0" "	Timeout configuration. Successful operation. Incorrect command format.
"TTM:REN- "+Name	Yes	Name, New module name, any string within 15 bytes	"TTM:OK\r\n0" "TTM:ERP\r\n0" "	Successful operation. Incorrect command format
"TTM:BPS-X"	Yes	X="4800", "9600", "19200", "38400", "57600", "115200" Set the corresponding baud rate	"TTM:BPSSET AFTER2S...\r\n0" "TTM:ERP\r\n0"	Successful operation. The new baud rate will be used in two seconds Incorrect command format
"TTM:MAC-?"	—	Acquire MAC Address	"TTM:MAC-xxxxxxx xxxx" xxxxxxxxxxxx is the MAC Address of the module	MAC Address is returned.
"TTM:RST-SYSTEMRESET"	—	Reset the module system	None	Module Reset
"TTM:ADP-(X) "	Yes	X= "2", "5", "10", "15", "20", "25", "30", "40", "50" Set the corresponding Broadcast Cycle, T=X*100ms	"TTM:OK\r\n0" "TTM:ERP\r\n0"	Successful operation. Incorrect command format. If set to "5", it is 500ms

"TTM:ADD-" +Data	Yes	Data is custom broadcast data, data length $L \leq 16$;	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	Successful operation. Incorrect command format.
"TTM:PID-" +Data	Yes	Data is a custom product identification code, the data length is $L=2$, and the default is 0000 ;	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	Successful operation. Incorrect command format.
"TTM:TPL-(X) "	No	$X="+4", "0", "-6", "-23"$ Set the corresponding transmit power, in dBm	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	Successful operation. Incorrect command format.
"TTM:CDL-X ms"	Yes	Set the delay time between low level output of BCTS and TX data output (in ms). $X = "0", "2", "5", "10", "15", "20", \text{ or } "25"$. The actual delay (T) will be $T = (X + Y)$ ms, if the minimum delay is no less than X, while $500 \mu s < Y < 1$ ms.	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	Successful operation. Incorrect command format.
* Note: Word in bold blue is by default. Row in grey means not saved after power-off.				

7. Broadcast Data Configuration

1) Default Broadcast Data

When the module EN pin is set low, the module will broadcast at an interval of 200ms. In the domain of the broadcast data GAP_ADTYPE_MANUFACTURER_SPECIFIC (iOS officially defined programming macro), the following packets are included (default of 9 Bytes):

0x00,0x00,	Customized device type code, 00 00 is by default, and can be set by AT command.
0x00, 0x00, 0x00, 0x00,	Current status of 4 PWM (by default), or 2 ADC acquisition values
0x00,	Percentage of module power supply (2.0 V = 0%).
0x00, 0x00,	I/O configuration, I/O output / input status (real-time changing with I/O current status)

Broadcast data will load automatically the current PWM status, or broadcast data is defined to be the acquisition value of 2 ADC, and data will be shown in the same position of 4-Byte. The module will always load automatically the data of the last-time operated channel. The current status of 4 PWM will be loaded when any value is written in PWM (FFB1). Or acquisition values of 2 ADC will be loaded when any nonzero value is written in ADC (FFD2).

2) Customized Broadcast Data

Customizing the broadcast packet can be realized by AT command, and the maximum length is 16 Bytes (in blue). In the broadcast data GAP_ADTYPE_MANUFACTURER_SPECIFIC domain will contain the following packet, and the length is 2 + n Bytes:

{ 0x00,0x00,	Customized device type code, 00 00 is by default, can be set by AT command;
Data [n],	Customized broadcast data, n < = 16; }

Note: Customized broadcast data can be modified by AT command and saved after power-off. After re-power on, last-time customized broadcast data will be shown. If customized broadcast data is all 0 (16 Byte), the customized broadcast will not be used but the system default broadcast packets. To avoid the extra power consumption caused by too long broadcast data, customized broadcast data can be set to be any value in 1 Byte.

8. System Reset and Recovery

There are three methods of module reset, among which the third one can recovery system parameters:

1. Reset module by AT command (See details in “AT Command”).
2. Remote reset module by the service channel interface of APP [See details in “BLE Protocol (APP Interface)”].
3. Reset module by RESET pin of the module (See details in “Module Parameter Configuration”). 30 seconds after power-on, set the pin low and hold for 5 s, the module will recover the parameters before user modified (light recovery, reset the module immediately after release press). 30 seconds after power-on, set the pin low and hold for 20 s, the module will be factory reset (deep recovery) immediately. This pin is with an internal pull-up, and the module will not enter recovery mode by default.

- **System parameters reset in light recovery including:**

- A. Anti-hijack password recovers to "000000". No password will be used by default.
- B. Four-channel PWM initialization mode, restored to 0x01, all four channels output 100% high pulse width;
- C. output status is 0, if IO is configured as output, default output low level

- **In addition to the above system parameters, the following parameters are included in in deep recovery**

- A. Serial port baud rate recovers to 9600bps.
- B. Device name recovers to "TAv22u-XXXXXXX" and X is the last four Bytes of MAC address.
- C. Data delay recovers to 0 (500 μ s < Delay < 1 ms).
- D) Output frequency of four-channel PWM, restoring to 0x8235 (120Hz);
- E. Broadcast cycle recovers to 2 (200 ms).
- F. Product ID, recovers to 0x00,0x00;
- G. IO configuration bytes are 0x00, default IO7, IO6 as signal footer, IO5-IO0 as input;
- H. Custom broadcast length, recovers to 0;
- I. All customized broadcast data recovers to 0. Default broadcast data is used but NOT customized broadcast data).
- J. EN mode recovers to 0. Level-enabled mode is by default.

Note: Due to the special use of RESTORE pin (IO0) in circuit design, continuous low level in 30 s before power-on should be avoided, otherwise the module will enter recovery mode.

9. IOS APP Programming Reference

The module is always to broadcast as slave, waiting for mobile phone to scan and connect as master. The scanning and connection are usually completed by APP. Due to the particularity of BLE protocol, there is no need to scan and connect Bluetooth LE devices in the system settings of the Smart phone. Smart devices are responsible for BLE connection, communication, disconnection, etc. And usually it is implemented by the APP.

Regarding BLE programming in iOS, the key point is the read, write and enable notify switch to Characteristic (or called channel) to. To read and write in the channel can realize the direct control on the direct-drive mode functions of the module and no extra MCU is needed. Typical functions that are involved are as follows:

```

/*!
 * @methodwriteValue:forCharacteristic:withResponse:
 * @paramdataThevalueto write.
 * @paramcharacteristicThecharacteristicon which to perform the write operation.
 * @paramtype Thetype of write to be executed.
 * @discussionWrite 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:(C
BCharacteristicWriteType)type;

```

Note: To write a characteristic value.

```

NSData*d = [[NSData
alloc] initWithBytes:&data length:mdata.length];
[pwriteValue:d
forCharacteristic:c
type:CBCharacteristicWriteWithoutResponse];

```

```

/*!
 * @methodreadValueForCharacteristic:
 * @paramcharacteristicThecharacteristicfor which the value needs to be read.
 * @discussionFetch 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: to read a characteristic value.

[preadValueForCharacteristic:c];

```

/!*
 * @methodsetNotifyValue:forCharacteristic:
 * @paramnotifyValueThevaluetoasettheclientconfigurationdescriptorto.
 * @paramcharacteristicThecharacteristiccontainingtheclientconfiguration.
 * @discussionAsk tostart/stopreceivingnotificationsforacharacteristic.
 * Therelevantdelegatecallbackwillthenbeinvokedwiththestatusoftherequest.
 * @seeperipheral:didUpdateNotificationStateForCharacteristic:error:
 */
-
(void)setNotifyValue:(BOOL)notifyValueforCharacteristic:(CBCharacteristic*)characteristic;

```

Note: to open a characteristic notify enable switch.

```

[self setNotifyValue:YES forCharacteristic:c]; //open notify enable switch.
[self setNotifyValue:NO forCharacteristic:c]; //close notify enable switch.
/!*
 * @methoddidUpdateValueForCharacteristic
 * @paramperipheralPeripheralthatgotupdated
 * @paramcharacteristicCharacteristicthatgotupdated
 * @error errorError messageifsomethingwent wrong
 * @discussiondidUpdateValueForCharacteristiciscalledwhen CoreBluetoothhasupdateda
 * characteristicforaperipheral.Allreadsandnotificationscomeheretobeprocessed.
 *
 */
-
(void)peripheral:(CBPeripheral*)peripheraldidUpdateValueForCharacteristic:(CBCharacteristic*)characteristicerror:(NSError*)error

```

Note: after each reading operation, this callback function will be performed. The application layer saves the data that is read in this function.

10. BLE Protocol (APP Interface)

- Bluetooth Data Channel [Service UUID:0xFFE5]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFE9 (handle:	Write	20	NONE	Written data will output from TX.

Remark: Bluetooth input data will be transmitted to serial output. APP operates write in this channel by BLE API, and the data will be output from TX. See details in “UART Transparent Transmission Protocol”.

- Serial Port Data Channel [Service UUID:0xFFE0]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFE4 (handle: 0x000E)	notify	20	NONE	Notification will be generated from the input data of RX in this channel

Remark: Serial input data will be transmitted to BLE output. If notification switch of FFE4 channel is enabled, (01 00 is needed to be written in 0x000E+1 = 0x000F by BTool), a notify event will be generated in this channel when the master CPU transmits legal data to the module RX through serial port, and APP can directly process and use notify information in the callback function. See details in “UART Transparent Transmission Protocol”.

- PWM output (4 channels) [Service UUID: 0xFFB0]

Characteristic UUID	Operation	Bytes	Default Value	Example	Remark	Corresponding Pin 2540
FFB1 (handle: 0x004D)	Read/write	1	0x01	0x00	Initialize four PWM channels with full low pulse width	--
				0x01	Initialize four PWM channels with full-height pulse width	
				0x02	Initialize the corresponding PWM channel with the current output pulse width	
FFB2 (handle: 0x0050)	Read/write	4	0xFFFF FF FF	0xFF000000	PWM1 Channel output full-height pulse width	P11
				0x00FF0000	PWM2 Channel output full-height pulse width	P10
				0x0000FF00	PWM3 Channel output full-height pulse width	P07
				0x000000FF	PWM4 Channel output full-height pulse width	P06
				0x20202020	PWM1-PWM4Channel output 32/256 pulse width	--
FFB3 (handle: 0x0053)	Read/write	2	0x8235	500 <=w <=65535	PWM Output signal frequency setting, four channels are the same, default as 0x8235 (120Hz)	--
FFB4 (handle: 0x0056)	Read/write	2	0x0000	0 <=t <= 65535	PWM Transition time width, the same for four channels, default as 0x0000 (abrupt change)	--

Remark:

FFB1 is the 4-channel PWM initialization mode setting channel. Write to the FFB1 channel (1 bytes) to configure the initialization mode of the four-channel PWM. The factory setting defaults to 0x01, full-height pulse width output. This set value is saved after power-off.

0x00, output 0% pulse width, full low pulse width, the module allows sleep in this mode;
0x01, output 100% pulse width, full height pulse width, the module allows sleep in this mode;

0x02, use the current PWM value output. After setting, the current PWM output value will be saved immediately as the initial value of the four-channel PWM after the next power-on. In this mode, the module does not enter sleep.

FFB2 is a 4-channel PWM output duty setting channel. Write to the FFB2 channel (4bytes) to adjust the output duty cycle of the four-channel PWM, each byte corresponds to a channel, 0xFF outputs the full high pulse width (100% high pulse width), 0x00 outputs the full low pulse width (0% high pulse width). If set to X, the duty cycle is approximately $X/0xFF$. You can also read this channel, and you will get the final set value. After power-on, the defaulted value is 0xFFFFFFFF, full-height pulse width output. After enabling this function, the module does not enter sleep until it is set to 0xFFFFFFFF (full high), that is, the PWM output is turned off. This channel is to set the duty cycle of four-channel PWM, the setting range is 0x00~0xFF, and the signal frequency defaults to 120Hz (see FFB3 frequency control channel).

For example: 0xFF000000

1. A total of four PWM output channels;
2. 0xFF000000, four bytes correspond to four channels respectively;
3. 0xFF outputs full height pulse width 100%, 0x00 corresponds to full low pulse width 0%;
4. The default pulse width frequency is 120Hz.

FFB3 is a 4-channel PWM output frequency control. Write to the FFB3 channel (2bytes) to adjust the frequency of the four-channel PWM output square wave. The width (w) of the signal period must meet: $500 \leq w \leq 65535$, one unit corresponds to 0.00000025s, and the corresponding square wave period: $0.000125 \text{ s} \leq T \leq 0.01638375 \text{ s}$, so the adjustable range of square wave signal frequency: $61.036 \text{ Hz} \leq f \leq 8 \text{ kHz}$, the four PWM output square wave frequencies are the same. The channel can also be read, and the final set value will be obtained. The set value will be saved after power-off. The factory default w is 0x8235, and the corresponding default pulse width frequency is 120Hz.

Example 1: Output a square wave with a frequency of 120Hz. Write 0x8235 (33333) to FFB3 channel, set the square wave period to 0x8235

* $0.00000025 \text{ s} \times 0.00833325 \text{ s}$, that is, the frequency is about 120 Hz;

Example 2: Output a square wave with a frequency of 1kHz. Write 0x0FA0 (4000) to the

FFB3 channel, and set the square wave period to 0x0FA0

* 0.00000025s = 0.001 s, that is, the frequency is about 1 kHz;

FFB4 is the 4-channel PWM output transition time length control. Write to the FFB4 channel (2bytes) to adjust the frequency change speed of the four-channel PWM output square wave. This is a time t, t must meet: $0 \leq t \leq 65535$, one unit corresponds to 100 ms, The longer the t, the slower the PWM transition from the current value to the target value; the smaller the t, the faster the rotation becomes. When t is 0, it will immediately jump to the target value. The four-channel PWM transition time shares this value. You can also read this channel, and you will get the final set value, which will be saved after power-off. The factory default t is 0x0000, and the corresponding conversion mode is immediate mutation.

-ADC input (2 channels) [Service UUID: 0xFFD0]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFD1 (handle: 0x0036)	Read/write	1	0x00	Enable control 0x00: Close two ADC channels 0x01: open ADC0 channel 0x02: open ADC1 channel 0x03: Open two ADC channels
FFD2 (handle: 0x0039)	Read/write	2	0x01F4	Acquisition period, unit ms For example, 0x01F4 corresponds to 500 ms
FFD3 (handle: 0x003C)	Read/notify	2	0x0000	ADC0 acquisition result, the maximum value is 0x01FFF
FFD4 (handle: 0x0040)	Read/notify	2	0x0000	ADC1 acquisition result, maximum value is 0x01FFF

Remark:

2-channel ADC input control. APP writes to FFD1 channel through BLEAPI interface to enable two 13bitADC channels. Write to FFD2 channel to control the sampling period t of two ADC channels, the unit is ms, $t \geq 100\text{ms}$.

If the notification of channels FFD3 and FFD4 are enabled (if using BTool operation, you need to write 01 00 to 0x003C+1=0x003D and 0x0040+1=0x0041), after each acquisition result, a notify event will be generated in this channel, with the result of this collection (range: 0 ~ 0x1FFF, low byte first), APP can directly process and use in the callback function. The ADC's reference source is the internal reference source of 1.25V, so the floating of the power supply voltage will not cause new measurement errors, and the measured sample voltage must be controlled between 0 ~ +1.25V.

- Programmable IO (8 channels) [Service UUID: 0xFFF0]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFF1 (handle: 0x0017)	Read/write	1	0b00000000	Configuration words of IO7~IO0. When the corresponding bit is set to 0: Bit7, bit6 indicate that IO7, IO6 are used as signal prompt pins, and low level is valid Bit5~bit0 represent IO5~IO0, as input ports When the corresponding bit is set to 1: Bit7, bit6 indicate that IO7, IO6 are used as normal output ports; bit5~bit0 indicate that IO5~IO0 are used as output ports
FFF2 (handle: 0x001A)	write	1	--	The output status of IO7~IO0. It indicates the output level of IO7~IO0 respectively. Bit7 and Bit6 are only valid when IO7 and IO6 are used as ordinary output ports. Bit7 and Bit6 are invalid, when used as signal prompt pins.
FFF3(handle: 0x001D)	Read/notify	1	0x3F	The input status of IO5~IO0. Can read or receive notifications. Under the premise of turning on the notification enablement, a certain input level change will be notified to the APP. IO7 and IO6 can only be used as output or signal prompt pins, and the corresponding pins are invalid.

Remark: IO configuration and control channel.

FFF1 is the configuration channel of 8 IOs, 8 bits correspond to the configuration control of IO7~IO08 respectively.

When BIT7 and BIT6 are set to 0, IO7 and IO6 are used as signal prompt pins, IO7 prompts sleep state, and 0 is wakeup state, 1 is the sleep state; IO6 prompts the connection state, 0 is the connection state, 1 is the disconnection state.

When the upper two bits of BIT7 and BIT6 are set to 1, IO7 and IO6 are used as ordinary output ports, and these two ports cannot be used as normal output ports, but not input port.

When the lower six bits BIT5~BIT0 are set to 1, IO5~IO0 are used as output ports, when they are set to 0, IO5~IO0 are used as input ports.

FFF2 is the output setting channel of 8 IOs. 8 bits correspond to the control of IO7~IO08 IOs respectively. It is valid only when the corresponding bit is set to output. When some IOs are set to output, you can write to the corresponding bits of this channel to realize the output control of these IOs, and the corresponding bits of the input ports are set to be invalid.

Note: IO configuration (FFF1) and output status (FFF2) are not saved by default after power off, but you can save the current configuration and output status of IO1~IO7 except IO0 by writing 0x01 to channel FF99 of the remote control extension channel. After power on, the last saved state will be used to initialize 7 IOs. In other words, the configuration and output state of IO0 cannot be saved after power-off. After IO0 is powered on, it always defaults to the input state, which is used to detect the function of restoring factory settings. (See the relevant chapters of "Module Parameter Setting" for details).

FFF3 is the input status channel of IO5~IO0, the lower 6 bits correspond to the input status of IO5~IO0 respectively. It is valid only when the corresponding bit is set as input. If the notification enable of the FFF3 channel is turned on (if you use BTool operation, you need to write 0100 to 0x001D+1=0x001E), when the level on these pins changed, the APP will generate a notify notification event in this channel, A byte is attached to indicate the status of the 6 IOs. Only the IO corresponding bits of the input port are configured to be valid. The APP terminal can directly process and use this status data in the notified callback function. IO7 and IO6 can only be used as output or signal prompt pins, so the corresponding bits are invalid.

-Timed Rollover Output (2 channels) [Service UUID: 0xFFF0]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFF4 (handle: 0x0021)	Read/write	4	0x00000000	IO6 first rollover delay setting 0: Do not start IO6 rollover X (Non-0): rollover after (X)ms delay
FFF5 (handle: 0x0024)	Read/write	4	0x00000000	IO6 second rollover delay setting 0: No second rollover X (Non-0): rollover after (X)ms delay
FFF6 (handle: 0x0027)	Read/write	4	0x00000000	IO7 first rollover delay setting 0: Do not start IO7 rollover X (Non-0): rollover after (X)ms delay
FFF7 (handle: 0x002A)	Read/write	4	0x00000000	IO7 second rollover delay setting 0: No second rollover X (Non-0): rollover after (X)ms delay

Remark: Schedule timing rollover configuration channel

When the IO6 and IO7 of the module are set as normal output, they can be respectively configured as the timing rollover output mode. You can set the next rollover time of IO6 and IO7 by writing to this channel. By setting the current output IO status, you can achieve a jump from 1 to 0, or a jump from 0 to 1. If it is set to 0, no rollover will be started. This function is only valid when the upper two bits (BIT7 and BIT6) of FFF1 are set to 1 (as output ports).

Channel FFF4 sets the delay time for the first rollover of IO6, and channel FFF5 sets the delay time for the second rollover of IO6. If FFF4 is set to 0, the rollover of IO6 is not started. If FFF4 is set to non-zero, and FFF5 channel is set to zero, only one IO6 rollover will be initiated.

The FFF5 channel must be set first, and the rollover is not started at this time, and then the FFF4 channel is set to a non-zero value to start the IO6 timing rollover. Similarly, you can turn off the timing rollover of IO6 by writing 0 to the FFF4 channel. At this time, any value previously written to the FFF5 channel will be cleared. The unit is ms, and the range is 0 to 0xFFFFFFFFms (4294967295ms, about 1193 hours, about 49.7 days), converted into hexadecimal:

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

Take IO6 as an example, to set a periodical and repeated rollover, the steps are as follows:

1. Set IO6 as normal output, write 0bx1xxxxxx to channel FFF1;
2. Set IO6 currently high (1), by writing 0bx1xxxxxx to FFF2;
3. Set the FFF5 channel to 0x05DC (1.5s), first set the second rollover delay, if it is 0, only rollover once
4. Set the FFF4 channel to 0x01F4 (0.5s), then set the first rollover delay, and the rollover will start at the same time.

Note: Step 3 and Step 4 cannot be reversed. You must set FFF5 first, and then write a non-zero value to FFF4 to start the rollover.

Write a value of 0 to FFF5, which means only rollover once. After the above operations, a square wave with a period of $1.5+0.5=2s$ will be obtained, in which the high level (1) will be maintained for 0.5s and the low level (0) will be maintained for 1.5s. You can write 0 to the FFF4 channel to immediately stop the current flipping behavior of IO6, and IO6 will maintain the current level state.

FFF6 and FFF7 are the timing rollover delay setting channels of IO7, and the method is the same as that of IO6.

Note: If IO6 and IO7 are in the period of timed reversal, the output writing of these two IOs and the reconfiguration of signal reminder operation are invalid. The current timed reversal must be stopped before operation.

- Level Pulse Width Count (2 Channels) [Service UUID: 0xFFFF0]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFF8 (handle:0x002D)	Read/notify	4	0x00000000	IO4 hold time of last level Unit: ms
FFF9 (handle: 0x0031)	Read/notify	4	0x00000000	IO5 hold time of last level Unit: ms

Description: Count the duration of IO level notification channel.

When IO4 and IO5 of the module are set as normal inputs, the level pulse width counting mode can be turned on. This function is only valid when the upper two bits BIT5 and BIT4 of FFF1 are set to 0 (as input ports).

The FFF8 channel is the IO4 (P1.6) level pulse width count notification channel. The APP opens the notification enable of this channel through the BLEAPI interface (if you use BTool to operate, you need to write 01 00 to 0x2D+1=0x2E). After the second rollover, a notify event will be generated in this channel, with the time width of the last level hold, the maximum value: 0xFFFFFFFF(ms), the unit is ms, the range is 0~0xFFFFFFFFms (4294967295ms, about 1193 hours, about 49.7 days), APP can be processed and used directly in the callback function.

The FFF9 channel is the IO5 (P1.7) level pulse width count notification channel. The APP opens the notification enable of this channel through the BLEAPI interface (if you use BTool to operate, you need to write 01 00 to 0x31+1=0x32). After the second flip, a notify notification event will be generated in this channel, with the time width of the last level hold, the range is 0 ~ 0xFFFFFFFFms (4294967295ms, about 1193 hours, about 49.7 days), APP can be directly in the callback function Handling and use.

Note: The counted level is the previous one, not the current level. The current level can be obtained by reading the FFF3 channel. Due to the BLE protocol limitation, the submission delay of the collected results will not be greater than the connection interval time.

- Anti-Hijacking Password [Service UUID:0xFFC0]

The module supports anti-hijacking password. Unauthorized mobile devices (or mobile phones) is prevent from being connected to the module effectively by this service. The initial password is 000000 (ASCII). In this case, APP does not need pairing with the module during connecting, so it is regarded as no use of password and any mobile device with specified APP can connect to the module.

The new password (not all zero) is set and saved by APP. If a new password (not all zero) is set, anti-hijacking is enabled. A password once configured requests will be submitted within 2s after APP connects to the module. Otherwise the connection is broken up. Any write operation except for password submission cannot be executed before APP submits the correct password.

If the password needs to be recovered, the module must be reset first by pull-low RESTORE (IO0) pin for 5 s and the operation must be done within 30 s after connection set-up. For safety, password read is not supported, and all passwords are kept by APP. A password channel is provided to realize the submission, modification and cancellation of the password by protocol. Meanwhile, event notify service of password is also provided to inform APP of the results of password operations, including 4 events: right password, error password, successful password update and cancel password.

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFC1 (handle:0x0045)	Write (Saved after power-off)	12	"123456123456"(ASCII)	Submit current password of 123456, and the new password must be same as the previous one.
			"123456888888"(ASCII)	Change the previous password of 123456 into the new one of 888888, and the previous password must be correct.
			"888888000000"(ASCII)	Cancel password (by changing the password into the default value 000000, and the previous password must be correct.
FFC2 (handle:0x0048)	Notify	1	0(PWD_RIGHT_EVENT)	Right password.
			1(PWD_ERROR_EVENT)	Error password
			2(PWD_UPDATED_EVENT)	Successful password update.
			3(PWD_CANCEL_EVENT)	Cancel password.

Remark:

1. Password is all in 12-Byte ASCII, wherein 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.
3. The execution result of related password operations can be generated in this channel by enabling notification of FFC2(01 00 is needed to be written into 0x0048+1 = 0x0049 by BTool).
4. When APP submits "123456123456", it means the new password is the same with the current one. And APP will be notified in channel FFC2 of "notify:0(PWD_RIGHT_EVENT)". It shows the password submission is correct.
5. When APP submits the password (red part) is different from the current one, such as: "123455xxxxxx", regardless of the value of "xxxxxx" part, APP will be notified in channel FFC2 of "notify: 1(PWD_ERROR_EVENT)". It shows the password submission is wrong.
6. When APP submits "123456888888", it means the new password is "888888" and the current password is "123456". APP will be notified in channel FFC2 of "notify: 2(PWD_UPDATED_EVENT)". It shows the password update is successful.
7. When APP submits "888888000000", it means the new password will be changed to an all-zero value. APP will be notified in channel FFC2 of "notify: 3(PWD_CANCEL_EVENT)". It shows the password is cancelled.

-Battery level report [Service UUID: 0x180F]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
2A19 (handle: 0x000A)	Read/notify	1	Percentage of power supply	Read the percentage of the current battery, or automatically generate a notification

Remark: Battery level reading or notification channel.

APP reads the 2A19 channel through the BLE API interface to obtain the current percentage of the power supply of the module. If the notification enable of this channel is turned on (if you use BTool operation, you need to write 01 00 to 0x000A+1=0x000B), after every power read, a notify notification event will be generated in this channel with power Percentage, maximum value: 100% (3V), minimum value: 0% (2V), APP can be directly processed and used in the callback function

-RSSI report [Service UUID: 0xFFA0]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFA1 (handle: 0x005D)	Read/notify	1	0x00	RSSI value, can be read/notified automatically
FFA2 (handle: 0x005A)	Read/write	2	0x0000	RSSI automatically reads the cycle setting, 0x0000 is to close automatic reading.

Remark: RSSI read or return channel

APP reads the FFA1 channel through the BLEAPI interface to obtain the RSSI received by the current module from the mobile device. If the notification enable of this channel is turned on (if you use BTool operation, you need to write 0100 to 0x005D+1=0x005E), after every RSSI read, a notify event will be generated in this channel with the RSSI value attached, APP can be processed and used directly in the callback function.

APP reads and writes to the FFA2 channel through the BLEAPI interface to set the RSSI reading cycle in ms. When this period is set to 0x0000, it is considered that the automatic periodic reading of RSSI is disabled. But it can still be read actively at any time. The read value of RSSI is signedchar type.

Similarly, to stop using the RSSI return function, you need to turn off the RSSI notification enable of the FFA1 channel and write 0x0000 to the FFA2 channel to disable the module's reading of RSSI, otherwise it will cause excess power consumption.

- Module Parameter Configuration [Service UUID:0xFF90]

Characteristic UUID	Operation	Saved or Not	Bytes	Default Value	Remarks
FF91 (handle:0x0062)	Read / Write	Yes	16	TA _v 22u-xxxxxxx x (ASCII string with terminator)	Device name, XXXXXXXX for the last four Bytes of the MAC address.
FF92 (handle: 0x0065)	Read / Write	No	1	0	Set Bluetooth connection interval: 0: 20 ms 1: 50 ms 2: 100 ms 3: 200ms 4: 300 ms 5: 400 ms 6: 500 ms 7: 1000 ms 8: 2000 ms
FF93 (handle:0x0068)	Read / Write	Yes	1	1	Set baud rate of serial ports: 0: 4800 bps 1: 9600 bps 2: 19200 bps 3: 38400 bps 4: 57600 bps 5: 115200 bps
FF94 (handle:0x006B)	Write	-	1	None	Channel to control remote reset and recovery: - Remote reset control by writing 0x55. - Remote light recovery control and reset by writing 0x35 (recover user data only). - Remote deep recovery control by writing 0x36 (factory reset) and reset
FF95 (handle:0x006E)	Read / Write	Yes	1	0	Set 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 ID
FF97 (handle:0x0074)	Read / Write	No	1	1	Set transmit power: 0: +4dBm 1: 0dBm 2: -6dBm 3: -23dBm

FF98 (handle:0x0077)	Read / Write	Yes	16	Default broadcast packet.	Set customized broadcast data Customizing broadcast data: $0 < n \leq 16$. See details in “Broadcast Data Configuration”.
FF99 (handle:0x007A)	Write	-	1	None	Remote control extension channel: 0x01: Saving-trigger control of I/O configuration output. Writing 0x01 will trigger the saving of current I/O configuration and output status. IO7 ~ IO1 will be initialized to use the saved configuration and output status when re-power on. But IO0 is always set as input by default when power on, which works as the test port of factory reset. 0x02: Remote shutdown control. In pulse-enable mode, writing 0x02 to this channel can shut down the module remotely.
FF9A (handle:0x007D)	Read / Write	Yes	1	0b00000000	System function EN switch: BIT0: EN mode configuration. 0 is by default and is corresponding to the low-level enabled. 1 means pulse-enabled. The module will switch between boot-up (starting broadcast) and shutdown (stopping broadcast) in turn, once EN pin receives a pulse every time. Effective pulse width T must meet: $W > 200 \text{ ms}$. When the broadcast time exceeds 30 s and the module is still not connected, it will shut down automatically. BIT1~BIT7: Unused by now.

Note: Command in Row in blue means this function cannot be saved after power-off.

FF91 is the channel for setting device name. Device name can be acquired and set by read and write to this channel accordingly. The length of device 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-xxxxxxx\0"(16byte), where in vvvv is the current firmware version number and XXXXXXXX is the last four Bytes of MAC address.

FF92 is the channel to set the connection interval. Connection interval between mobile devices and the module can be set by write 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 (20 ms) after power on. Test shows that it takes around 30 s to wait when the connection interval is changed from 500 ms to another interval by

iPhone 4S (iOS 5.1.1). But it will be effectively very quickly if the connection interval is changed from a high frequency one (such as: 20 ms) which is affected by BLE protocols.

FF93 is the channel to set baud rate. Baud rate can be set by read and write to the channel. The new baud rate will be effective in two seconds after set and can be saved after power-off. While 1 (9600 bps) is by default.

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

1. Write 0x55 to this channel will software-reset the module.
2. Write 0x35 to the channel will light-recovery the module. All user settings will be recovered to the factory defaults, including I/O output status, PWM initialization mode and user password. Afterwards, the module will be reset.
3. Write 0x36 to the channel will deep-recovery 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 broadcast cycle. Broadcast cycle can be set by read and write to this channel. The setting can be saved after power-off. While 0 (200 ms) is by default.

FF96 is the channel to set product ID by read and write to the channel. APP can filter and connect to the specific product through this code. The setting can be saved after power-off. And 0x0000 is by default.

FF97 is the channel to set transmit power by write to this channel. The setting cannot be saved after power-off. And 1 (0 dBm) is by default.

FF98 is the channel to set broadcast packets. Broadcast data can be customized by write 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 details in "Broadcast Data Configuration").

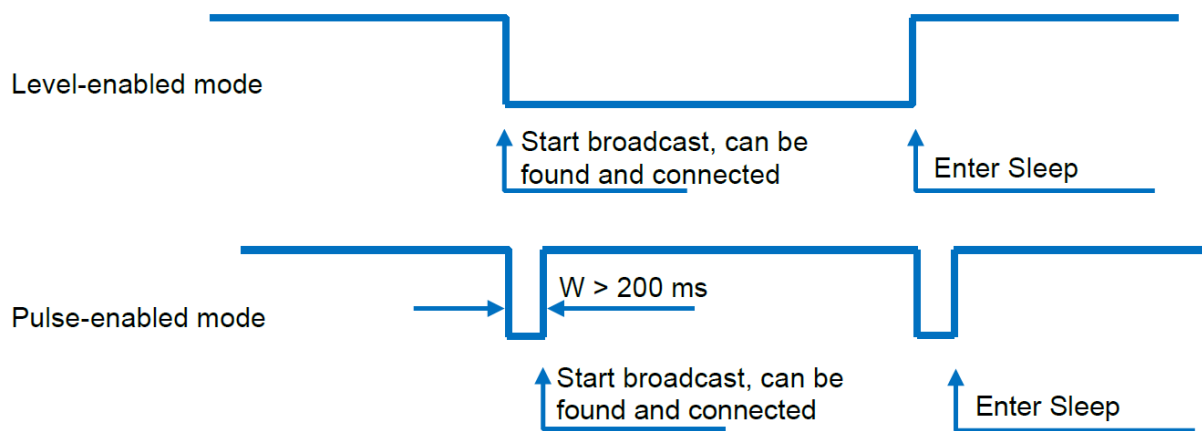
FF99 is the channel for remote control extension. By writing different values, the specific control functions can be realized. Writing 0x01 to this channel will trigger the module to save the current configuration and the output status of all I/Os (except IO0). When re-power on, the module will always initialize IO7 ~ IO1 with the saved settings and output status. While IO0 is always set to default input status to work as the triggering I/O of factory reset. 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 down the module 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. All 0b00000000 is by default. This setting can be saved after power-off.

BIT0: 0 is default to level-enabled mode. In this mode, low-level is enabled to start broadcast and high-level is enabled to sleep (0.4 μ A). 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 μ A) in turn, after get a legal pulse width ($W > 200$ ms). 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 & Pulse Enabled Model

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

1. If EN pin is enabled (set low), the module will keep broadcasting, until it is connected or EN is set high.
2. Regularly 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 (can be found and connected) has the following features:

1. If broadcasting last for 30s after enabled, but still not connected, the module will stop broadcasting and shut down.
2. If regularly disconnected, the module will continue to broadcast for 30s. If it is still

not connected, the module will stop broadcasting and shut down.

3. If disconnected due to timeout, 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) and is connected at low level, it will output high level, if Bluetooth 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, it will output low level pulse (1s) for once.
2. When Bluetooth is regularly disconnected (active disconnecting by APP), it will output low level pulse (0.5s) for once.
3. When Bluetooth is disconnected due to timeout, it will output the square wave of 2 Hz and last for 2 minutes. During this period, it will keep broadcasting and cannot be shut down, until the module re-connects with master device. Broadcast status & IO6 prompt modes in different EN modes are summarized as follows:

Broadcast status & IO6 prompt modes in different EN modes are summarized as follows:

Module	Enabled but Not Connected		Connected		Actively Disconnected		Timeout Disconnected	
	IO6 Prompt	Broadcast Status	IO6 Prompt	Broadcast Status	IO6 Prompt	Broadcast Status	IO6 Prompt	Broadcast Status
Level Enabled Mode	High Level	Keep Broadcast	Low Level	Stop Broadcast	High Level	Keep Broadcast	High Level	Keep Broadcast
Pulse Enabled Mode	High Level	Broadcast for 30 s	Low Level Pulse W = 1 s	Stop Broadcast	Low Level Pulse W = 0.5 s	Broadcast for 30 s	2 Hz Square Wave for 2 min.	Keep Broadcast

- Device Information [Service UUID:0x180A]

Characteristic UUID	Operation	Bytes	Default Value	Remarks
2A23 (handle:0x0003)	Read	8	xxxxxx0000xxxx xx (Hex)	System ID, where xxxxxxxxxxxx is the MAC address of module, with low Byte in front.
2A26 (handle:0x0005)	Read	5	V2.2u (ASCII)	Firmware version number of the module.

Remark: This channel is for module information read.

Acquire module information by read this channel 2A23. For example: xxxxxx0000xxxxxx, wherein xx part is the MAC address of the module in six Bytes (low Byte in the front).

Acquire module version number by read this channel 2A26. For example: Vx.xx, wherein xx is firmware version number.

- Port Timing Event (EVT) Configuration [Service UUID:0Xfe00]

Port timing event configuration service, used to set timing events of IO or PWM port. This service provides the function of setting timed tasks, that is: a certain execution subject performs a certain action at a certain time. The execution body can be one of 10 ports, including 6 sudden change output ports, and 4 gradient PWM output channels. The type of action performed can be sudden change or gradual change.

1. Timing event (EVT) settings:

This service provides 32 timing events that can be set. An event refers to the execution of a specific action at a certain moment.

Timed event (EVT) = execution time + action type; can be set through the event read and write channel (UUID: 0xFE03), which contains the following parameters:

- Event index number, 1 byte, used to indicate the modified or set event index number;
- Execution time (timing time), 7 bytes, used to indicate the event trigger time;
- Action type, 1 byte, used to indicate the action to be executed when the timing overflows, including output high level, output low level, level inversion, PWM mutation, and PWM gradient;
- Operation parameter, 3 bytes, used to set the target duty cycle of PWM and the overhead time of gradual change, this parameter has nothing to do with the 6 sudden change output ports, it is specially used to define the parameters of the gradual change behavior of the four PWM channels;

2. Scheduled task settings

Timed task = a certain execution subject + a certain timed event

The ports (executors) that can be configured to execute timing events include 6 IO ports and 4 PWM output ports. After the port opens the timed event, a timed task is formed. When a timing event is triggered, the port will perform actions according to the definition of the event. Each port can be configured with a maximum of 32 timing events, and there is a separate response switch, the settings between the ports do not conflict with each other, multiple ports can be configured as the same timing event at the same time, but if the action type of the event is invalid for the port, The port will ignore this timing operation. For example, IO0 port (without PWM output function) enables a certain PWM gradual change event. When the timing event is triggered, IO0 will ignore this event. It can be set through the port event read and write channel (UUID: 0xFE05), which contains the following parameters:

- Port index number, 1 byte, used to indicate the modified or set port;
- Event open bit, 4 bytes, a total of 32 bits, respectively control the response switch of

32 timing events, set whether to respond to an event;

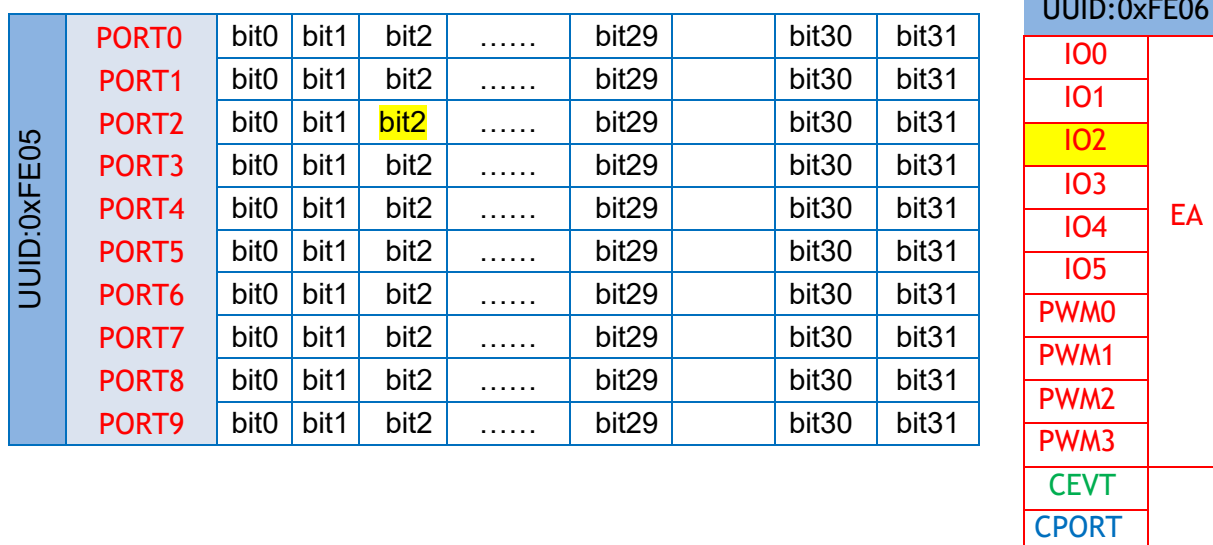
The event read-write channel (UUID: 0xFE03) and the port event read-write channel (UUID: 0xFE05) are multiplexed write interfaces. Each time it is written, the first byte of "event index number" or "port index" "No." points to the event or port that needs to be set (equivalent to a pointer), and the other bytes behind are the specific details of the setting. If you want to get the definition of an event or the setting information of a port, you need to write the index number you want to read through the read event pointer channel (UUID: 0xFE02) or read port event channel (UUID: 0xFE04), and then Read the event read-write channel (UUID: 0xFE03) and the port event read-write channel (UUID: 0xFE05) to obtain information about the specified index.

3. Timing task enable setting

The event port configuration channel (UUID: 0xFE06) is used to control the enable switch of timing tasks (port timing events), including the total enable bit (EA) of all timing tasks, six IO ports and 4 The timing events of each PWM port are individually enabled, including the timing event clear control bit (CEVT) and the port timing event clear control bit (CPORT). After the timed event clear control bit and the timed task clear control bit are set, all 32 timed events and 10 port timed event configuration information will be cleared.

4. Timing event (EVT) response conditions:

- Timing event EVT timing time overflow trigger;
- The response port opens the response switch BIT that points to the EVT;
- Individual enable bit IO/PWM enable for timing events pointing to the response port;
- Timing event total enable bit EA enable;



Timing Event, Timing Task, Timing Task Enable Configuration Diagram

As shown in the above table, EVT2 is triggered at regular time. If PORT2 turns on the corresponding bit2 and the IO2 bit and EA bit are enabled at the same time, IO2 will trigger the action type operation of EVT2.

5. Regarding priority:

Low-index events or port actions will be executed first. For example, the timed time of timed event 1 and timed event 2 are the same,

Two timed events are enabled on port 0 and port 1 at the same time. If the two timed events are triggered at the same time, the priority of the timed event of the execution port is as follows:

1. Timed event 1 of port 0;
2. Timed event 1 of port 1;
3. Timed event 2 of port 0;
4. Timed event 2 of port 1;

Note:

- If you want to control the IO port regularly, you need to configure the corresponding IO port as an output port first. Please refer to the chapter "Programmable IO (8 channels)" in the BLE protocol description in this article.
- The timing function is valid when the module is connected or disconnected.
- The timing function configuration information is not saved after the module is powered off. If it is lost, it can be refreshed synchronously with the APP.
- In order to avoid errors in the RTC clock, it is recommended to update the RTC clock synchronously before connecting or disconnecting the APP.

Characteristic UUID	Operation	Bytes	BIT	Default Value	Definition	Remarks
FE01 (handle: 0x0086)	R/W	7	BYTE7~BYTE0	0x07D00101000000	Second, Minute, Hour, Day, Month, Year (L), Year (H)	<p>RTC clock operation channel.</p> <p>Through this channel, you can easily read and modify the current system clock.</p> <p>Value range: Second: 0-59; Minute: 0-59; Hour: 0-23; Day: 1-31; Month: 1-12; Year: over 2000</p> <p>The default time is January 1, 2000, 0:00:00:00.</p>
FE02 (handle: 0x0089)	R/W	1	BYTE0	0x00	Event index number	<p>Read event pointer.</p> <p>Before reading an event, you must set this pointer to point to the event that needs to be read, and then read the FE03 channel.</p> <p>Value range: 0-31, respectively representing 32 timing events.</p>

FE03 (handle: 0x008C)	R/W	12	BYTE0	0x00	Event index number	Event read and write channel. Reading and writing to this channel can get and set timing events.
			BYTE7-BYTE1	0x000000 00000000	Second, Minute, Hour, Day, Month, Year (L), Year (H)	Event index number: Value range: 0-31: respectively indicate 32 timed events; Timing time (seconds, minutes, hours, day, month and year): The value range is the same as that of the RTC clock. If one of the bytes is invalid (FF means invalid), the lower valid byte time will be used as the cycle timing. Such as the following timing (Hex): 00 FF 01 01 01 D0 07 This timing event will be triggered at 0 seconds in any minute;
			BYTE8	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 mutation; 5: PWM gradient;
			BYTE9	0x00	PWM New duty value	·PWM new duty value: It is valid when the action type is 4 or 5, shows the target duty value after change. Value range: 0-255; 0 means the duty ratio is 0%, namely all low level, 255 means the duty ratio is 100%, namely all high level
			BYTE10	0x00	PWM Low byte of the transition duration	·PWM gradual change time: It is valid when the operating value is 5. The time spent changing from the current duty value to the new duty value, the larger the value, the slower the change; the smaller the value, the faster the change. Value range: 0-65535, unit is 100ms;
			BYTE11	0x00	PWM High byte of the transition duration	

FE04 (handle: 0x008F)	R/W	1	BYTE0	0x00	Port index number	<p>Port event read pointer.</p> <p>Port index: Before reading all applicable events of a certain port, this pointer must be set to point to the port that needs to be read, and then read the FF05 channel.</p> <p>Value range: 0-9: respectively indicate the timing ports of IO0-IO5 and PWM0-PWM3.</p>
FE05 (handle: 0x0092)	R/W	5	BYTE0	0x00	Port index number	<p>Port events read and write channels.</p> <p>Port index number Value range: 0-9: respectively indicate the timing ports of IO0-IO5 and PWM0-PWM3.</p>
			BYTE4-BYTE1	0b000000 00000000 00000000 00000000 00	Timed event enable 0-31	<p>Applicable event enable switch, different bits correspond to 0-31 timing events:</p> <p>Ranges: 1: open; 0: close;</p>

FE06 (handle: 0x0095)	R/W	2	BYTE0	0b000000 00	Bit0: Timing Master Enable	Event port configuration Ranges: 1: enable; 0: close; -BYTE0: Enable all the timing event ports (EA); -BYTE0: BIT1~BIT7, BYTE1: BIT0~BIT2, Enable the single port timing events individually; -BYTE1: BIT3, clear the control bit for timing events, clear all set timing events; (CEVT) -BYTE1: BIT4, clear the control bit for the timing task, clear 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	
			BYTE1	0b000000 00	Bit7:PWM0 Enable	
					Bit0:PWM1 Enable	
					Bit1:PWM2 Enable	
					Bit2:PWM3 Enable	
					Bit3: Timed event clear control	
					Bit4: Timed task clear control	
					-	
					-	
					-	

Tips: here are four steps to implement a timed task:

1. Design one or more timed events (specify what action to perform at what time) (write 0xFE03);
2. Specify which IO port to execute this timing event (establish the relationship between the timing event and the execution subject) (write 0xFE05);
3. Turn on the response enable switch of this IO port to the timing task (allowing response) (write 0xFE06);
4. Turn on the master enable switch of the timed task (write 0xFE06).

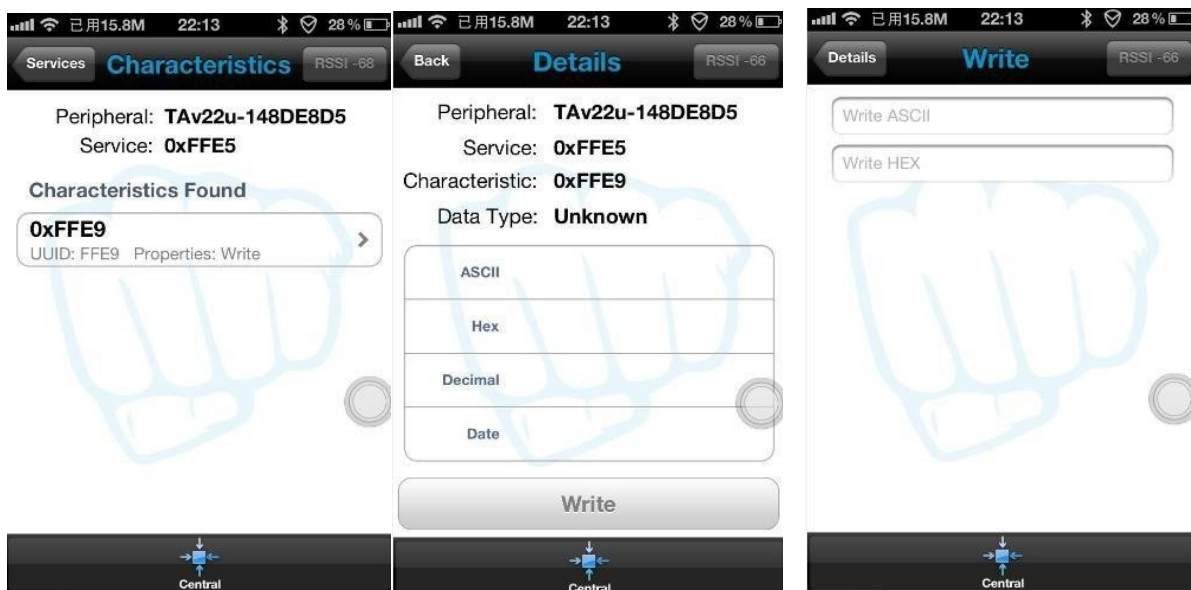
11. Test the Transparent Transmission Function with APP

Use iPhone 4S, iPhone5, iPhone5S, iPad Mini or iPad4 to install the test software LightBule.

After the APP is installed, it will scan automatically. The scanned devices will appear in the list (maybe you need to turn on Bluetooth). Click on a device to connect. After the connection is successful, it will jump to the main service control interface.

Receiving process: select 0xFFE0->0xFFE4, and then click StartNotify to receive the characters sent from the host.

Sending process: select 0xFFE5->0xFFE9, and then click Write to send characters to the host.



12. Test with USB Dongle and Btool

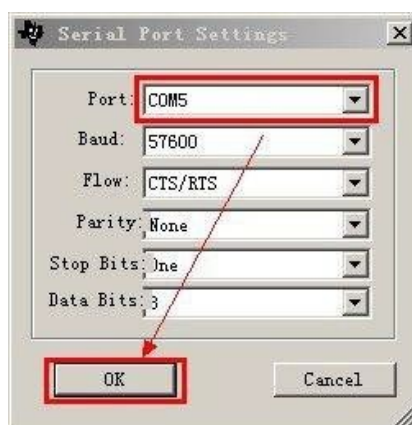
You can use the USB Dongle in TI's official CC2540MinDK development kit to test the BLE module, simulate mobile phone to work with the C:\TexasInstruments\BLE-CC254x-1.2.1\Projects\Btool\Btool.exe in the installation directory for Bluetooth communication testing.

This USB Dongle needs to use the project in the installation directory of C:\TexasInstruments\BLE-CC254x-1.2.1\Projects\ble\HostTestApp\CC2540. Compile and download to USB Dongle. For specific BTOOL usage details, please refer to the official documentation from TI

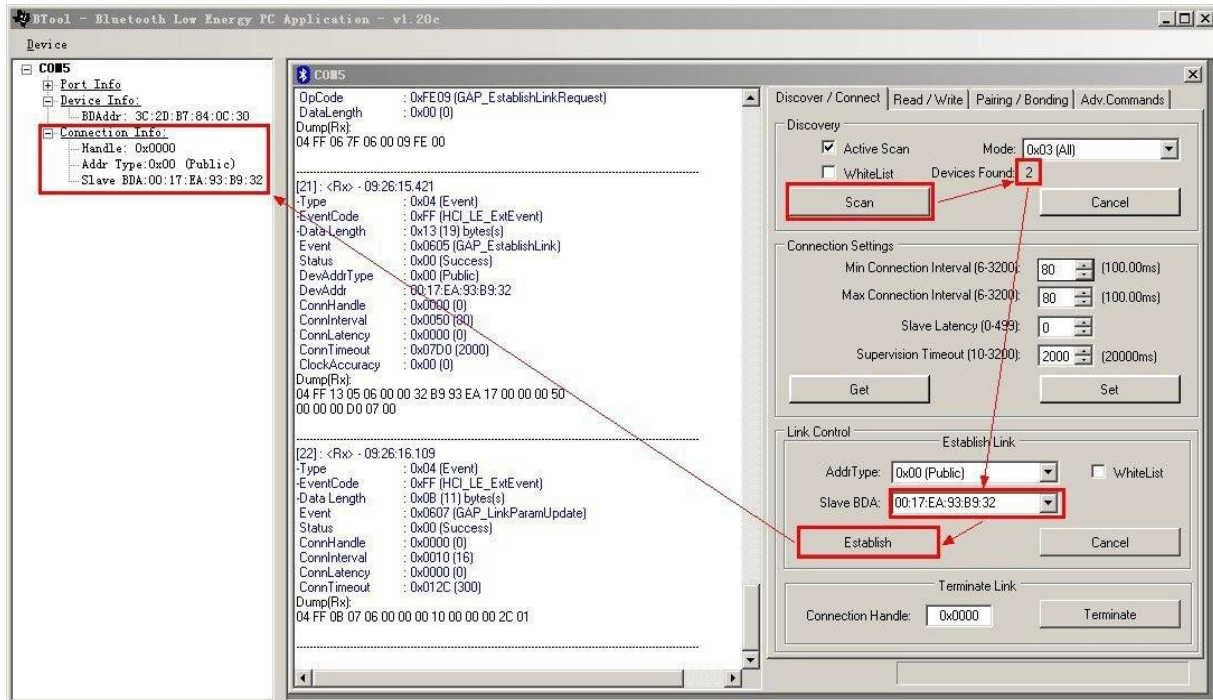
- Connect BLE module

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

1. Open the project file in the C:\TexasInstruments\BLE-CC254x-1.2.1\Projects\ble\HostTestApp directory, compile and download it to USB Dongle;
2. Power on the module (3~3.3v);
3. Lower the module enable pin EN to the ground, and the module will start broadcasting;
4. Insert the USB Dongle into the PCUSB port, a serial device (such as COM5) will appear in the hardware management;
5. Open C:\TexasInstruments\BLE-CC254x-1.2.1\Projects\Btool\Btool.exe;
6. Menu Device->NewDevice, select the serial port found in 4, select the default setting, OK;



7. Scan the connection, and follow the direction of the arrow to scan and connect, where 00:17:ea:93:b9:32 is the physical address of the module. Please make sure it is the target module before connecting.
8. After the connection is successful, the connected module information Connection Info will appear on the left.



Now the module is successfully connected, you can start to test the direct drive function and the Bluetooth serial port forwarding function (transparent transmission).

- Test the direct drive function

Use Btool and Dongle to access any channel defined in the protocol. The classic steps are as follows:

- 1) Find the handle of the channel by UUID
- 2) Remember the handle corresponding to the channel
- 3) Use handle+1 to turn on the channel notification switch
- 4) Use UUID or handle to read the channel
- 5) Use handle to write to the channel

Please note, if you directly use the handle provided in the characteristic value information table in the "BLE Protocol Description (APP Interface)", you can skip steps 1 and 2. The notification switch of the channel must be operated by the handle+1 pointer. The channel mentioned here refers to the characteristic value (Characteristic). When writing to the channel, the number of bytes must be consistent with the length defined

by the protocol, otherwise it will be considered illegal and fail.

Important note: The core operation of BTool is to connect first, then read and write the handle of a channel, and turn on the notification switch. The first few steps (1,2) are to find the corresponding handle. In IOS programming, there is no need to find the handle, but to read and write directly through the channel UUID.

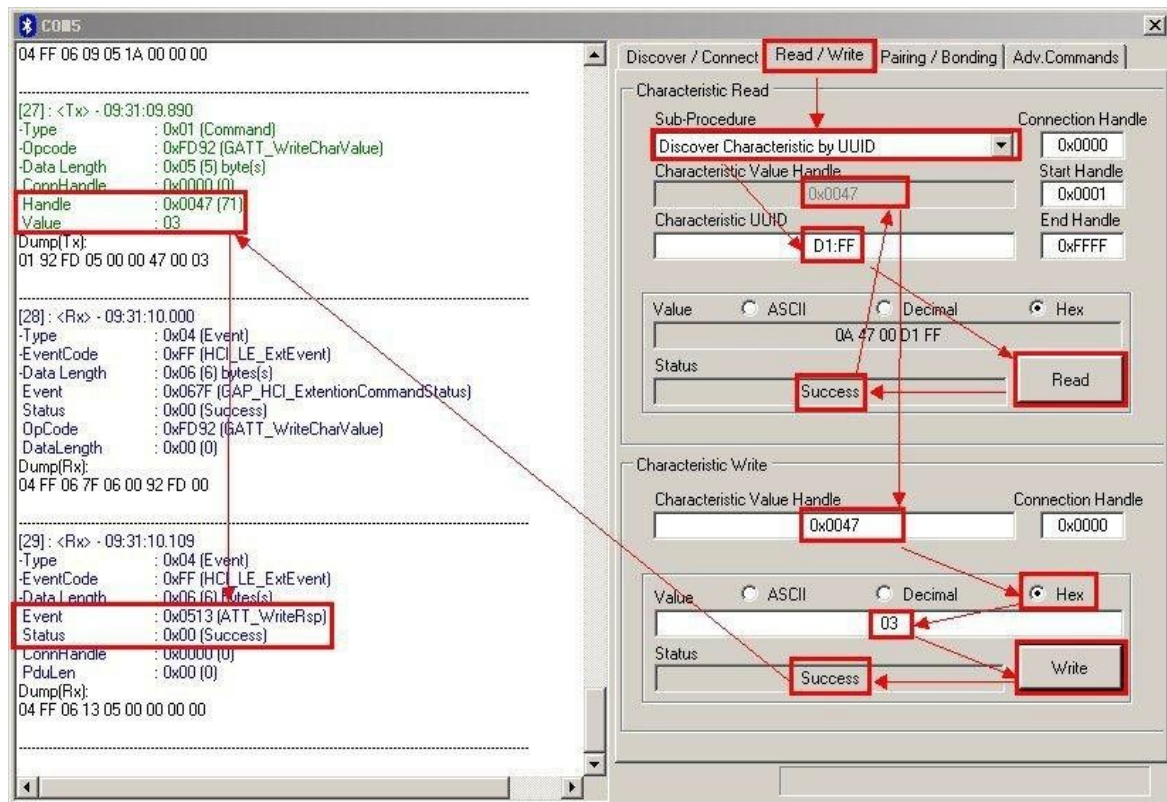
Now take ADC as an example to show you how to use USB Dongle and Btool software tools to directly drive the Bluetooth module. The following example takes the DL-CC2541 BLE transparent transmission module V1.0 as an example. The corresponding Handle may be slightly different from other versions, but the operation flow is completely the same. Other functional test methods are similar, except that the corresponding channel UUID is different, and the read and write methods are the same.

- ADC input (2 channels) 【Service UUID: 0xFFD0】

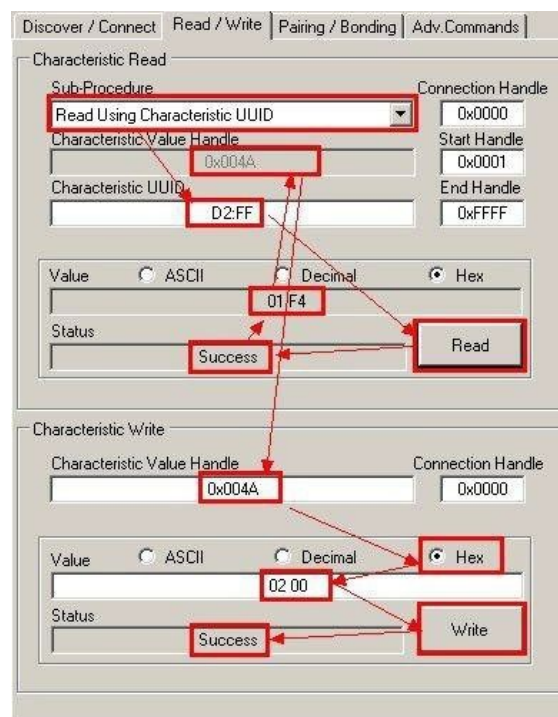
Characteristic UUID	Operation	Bytes	Default Value	Remarks
FFD1(handle: 0x0047)	Read/write	1	0x00	Enable control. 0x00: Close two ADC channels 0x01: open ADC0 channel 0x02: open ADC1 channel 0x03: Open two ADC channels
FFD2(handle: 0x004A)	Read/write	2	0x01F4	Acquisition period, unit ms e.g.: 0x01F4 corresponds to 500 ms
FFD3(handle: 0x004D)	Read/notify	2	0x0000	ADC0 acquisition result, the maximum value is 0x01FFF
FFD4(handle: 0x0051)	Read/notify	2	0x0000	ADC1 acquisition result, maximum value 0x01FFF

1. Turn on ADC0 and ADC1. As shown in the figure, subsequent operations can refer to the direction of the red arrow.

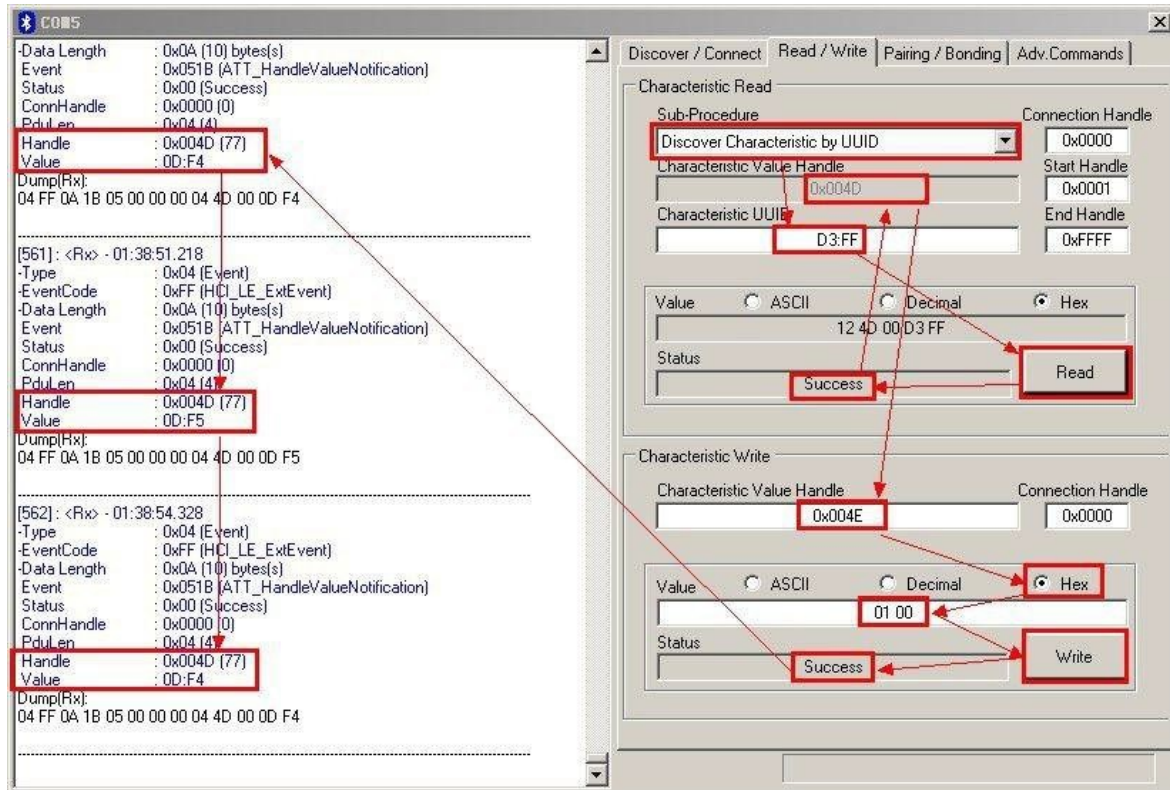
First look for the channel to be controlled (UUID), enter the UUID that needs to be searched, the low bit is in the front D1: FF, after the read is successful, you get handle=0x0047, write 03 to 0x0047, thus open ADC0 and ADC1, see the table above.



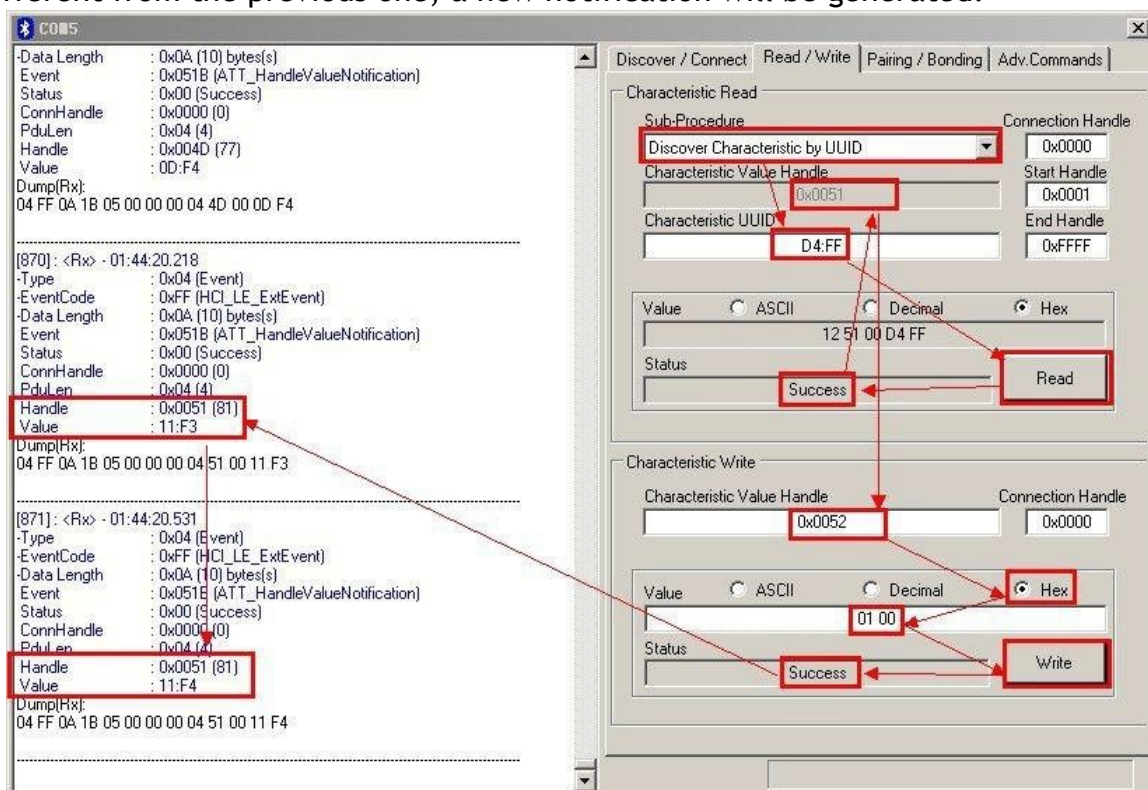
2. Read and reset the acquisition cycle. First read the FFD2 channel, and you can get 01F4, which means the default is 500ms. By obtaining handle=0x004A, write 02 00 high bit first, (0x0200 =512 ms), set to collect once every 512ms



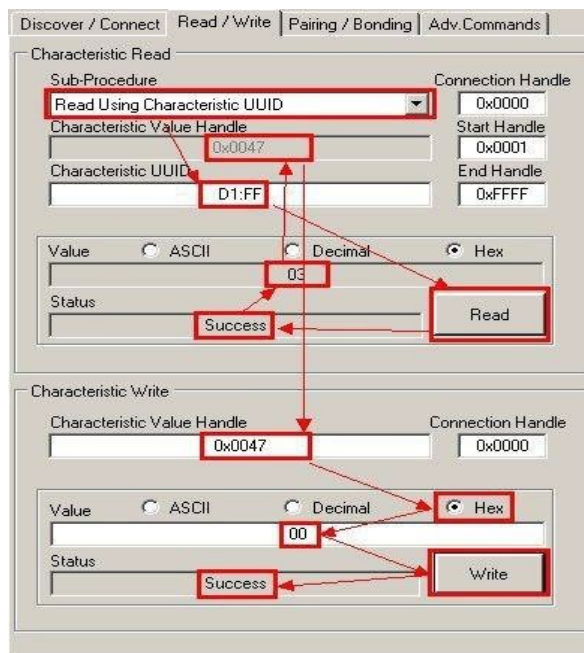
3. Turn on the ADC0 channel notification enable switch, the address of the notification switch is handle+1, $0x004D+1=0x004E$, after that, every time the ADC0 acquisition value is different from the previous one, a new notification will be generated



4. Turn on the ADC1 channel notification enable switch. The address of the notification switch is handle+1, $0x0051+1=0x0052$. After that, every time ADC1 collects a value different from the previous one, a new notification will be generated.



5. Stop ADC0 and ADC1. It is similar to turning on ADC0 and ADC1. Write 00 to 0x0047, which turns off ADC0 and ADC1.



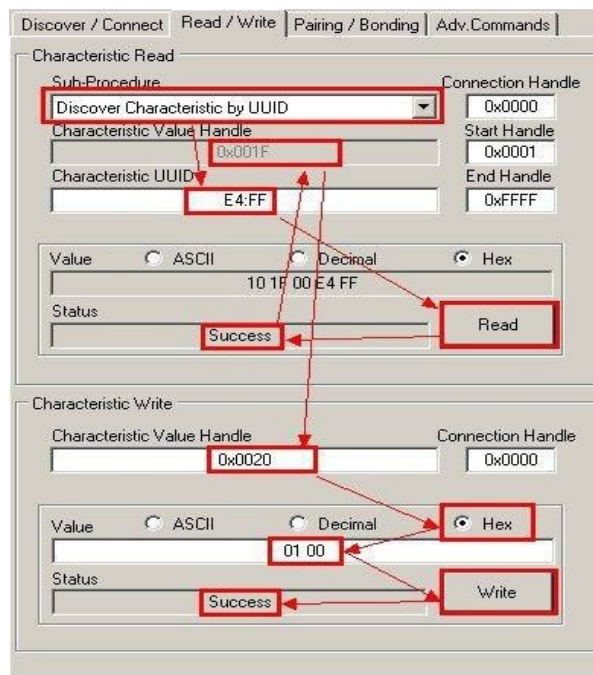
The screenshot shows the 'Read / Write' tab of the BLE module software. The 'Characteristic Read' section is active, with the 'Sub-Procedure' dropdown set to 'Read Using Characteristic UUID'. The 'Characteristic Value Handle' is set to '0x0047' and the 'Characteristic UUID' is 'D1:FF'. The 'Value' field is set to '03' in 'Hex' mode. The 'Status' field shows 'Success'. The 'Read' button is highlighted. The 'Characteristic Write' section is also visible, with the 'Characteristic Value Handle' set to '0x0047' and the 'Value' field set to '00' in 'Hex' mode. The 'Write' button is highlighted. Red arrows indicate the flow of data and the sequence of operations.

- Test Transparent Transmission Function

1. Connect the module like the bridge mode in the system diagram to a serial port terminal or a single-chip microcomputer to perform Bluetooth serial port forwarding test. Note: BRTS must be set low, otherwise the serial port data cannot be received by the module RX.

After using BTool to establish a connection between the BLE module and the USB Dongle (refer to the description in the previous section for the connection process), open the automatic notification switch of the serial data channel by writing 01:00 to Handle: 0x0020, as shown in the figure below.

If the host sends a legal data packet to the RX end of the BLE module, the module will automatically send it to BTool in the form of a notification, and the display bar on the left will display the specific data. The serial port data sent by MCU to the module can be any length within 200 bytes.

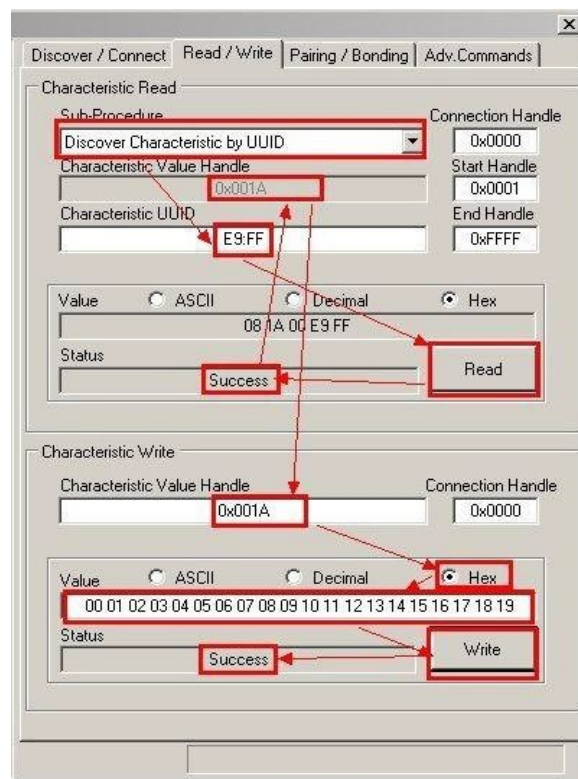


The module sends to the mobile device using the serial data channel, and the UUID corresponding to the characteristic value (channel) is as follows:

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

- Write 1-20 bytes of data to the module through BTool. When the module receives a write operation from the mobile phone, the module will send it to the MCU through the serial port. The user can check whether the data is correct by reading the MCU, or display the data written by the mobile phone to the module through the serial port assistant.

For example: to write 7 bytes of data to the module, it is written through Handle0x001A, as shown in the figure below.

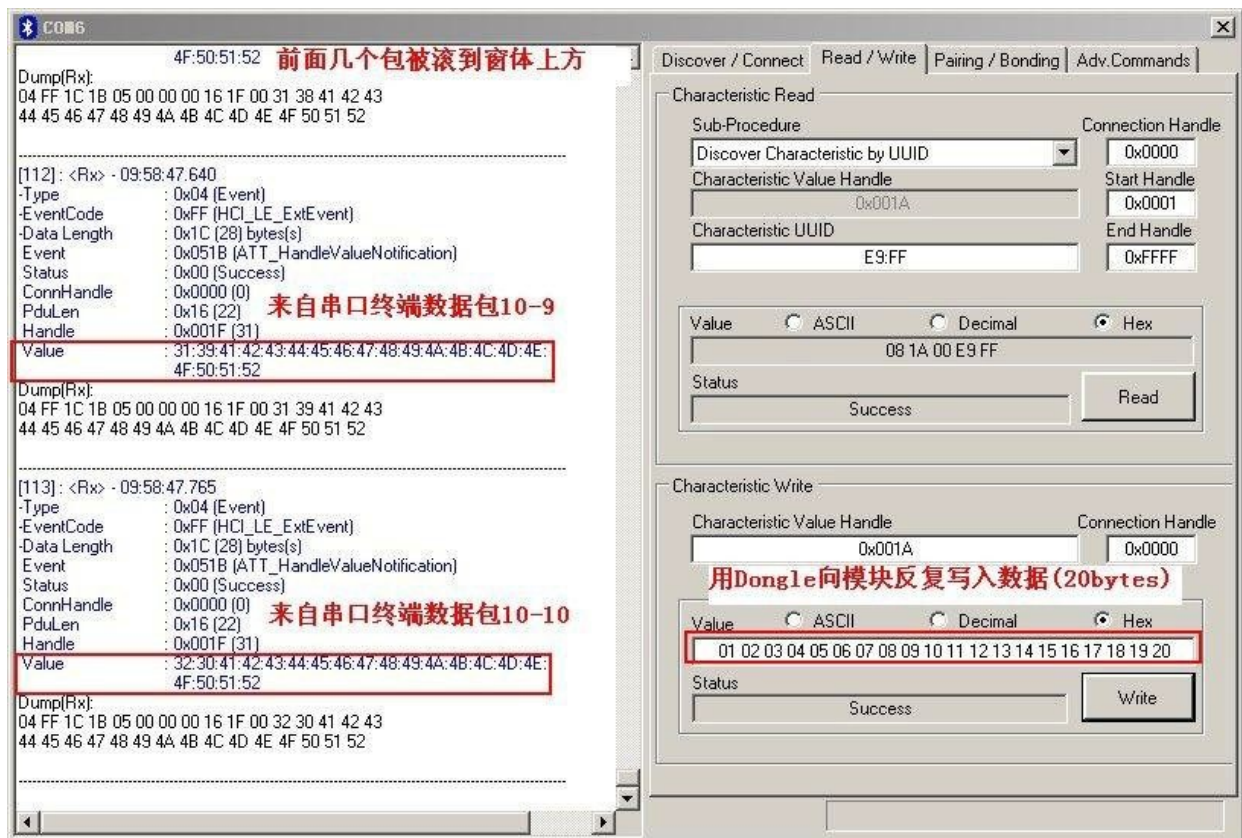


Note: 1-20 bytes can be written to the module, but cannot exceed 20 bytes. Therefore, when programming on the mobile phone, you must send it in packets by yourself, and the length of each packet must not exceed 20 bytes. The mobile device sends to the module through the Bluetooth data channel, and the UUID corresponding to the characteristic value (channel) is as follows:

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

The test of the transparent transmission function can be directly connected to the PC serial port through the level conversion module, and the test can be performed through the serial port terminal. The reference screenshot is as follows:

1. Screenshot of BTool sending and receiving data.



2. A screenshot of the PC terminal connected to the transparent transmission module. Please note that BRTS must be set low, otherwise the serial port data cannot be received by the module.



13. Host Reference Code (transparent transmission)

Logical relationship: Two IO ports of BCTS, are used for notification and control, for sending and receiving between modules. These two IOs are normally high and triggered when they are set low.

If the module has data to send, set BCTS low to notify the microcontroller to receive.

If the microcontroller has data to send, set BRTS low to notify the module to receive.

The schematic code is as follows:

```
void main(void)
{
    EN=0; // Enable EN, start broadcasting
    while(!BLEModuleAck("TTM:OK\r\n0")); // Wait for the mobile phone to scan and connect
    // Wait for the connection until succeed, you can also
    // join the time limit waiting
    // It can also determine the level of the connection
    // prompt signal line
    BRTS=0; // BRTS set low, module is ready to receive
    halMcuWaitMs(2); // 2ms delay
    UARTWrite(HAL_UART_PORT_0,"TTM:CIT-100ms",14);
    // Modify the connection interval and get
    // confirmation from the serial port:
    halMcuWaitMs(5); // Delay 5ms to ensure data has been sent
    BRTS=1; // RTS is set high, sending is complete
    while(!BLEModuleAck("TTM:OK\r\n0")); // Wait for the connection until succeed, you can also
    // join the time limit waiting
    while(1){ // Cyclic sending and receiving test
        while(1){
            if(BCTS== 0){ // Detect, prepare to receive if BCTS is set low
                while(BCTS==0); // Wait for the connection until succeed, you can also
                // join the time limit waiting
                if(UARTRead(uartBuffer)==SUCCESS) // Serial port read data
                {.....} // Use data
            }
            BRTS=0; // RTS set low, module is ready to receive
            halMcuWaitMs(2); // 2ms delay
            send_TX("1234567890"); // Send any data (within 200byte)
            halMcuWaitMs(5); // Delay 5ms to ensure data has been sent
            BRTS=1; // RTS is set high, sending is complete
            halMcuWaitMs(20); // Delay to send the next packet, time depends
        }
    }
}
```

14. Application and Implementation

Basic Operation of Hardware Design

- 1). It is recommended to offer the module with a DC stabilized power supply, a tiny power supply ripple coefficient and the reliable ground. Please pay attention to the correct connection between the positive and negative poles of the power supply. Otherwise, the reverse connection may cause permanent damage to the module;
- 2). Please ensure the supply voltage is between the recommended values. The module will be permanently damaged if the voltage exceeds the maximum value. Please ensure the stable power supply and no frequently fluctuated voltage.
- 3). When designing the power supply circuit for the module, it is recommended to reserve more than 30% of the margin, which is beneficial to the long-term stable operation of the whole machine. The module should be far away from the power electromagnetic, transformer, high-frequency wiring and other parts with large electromagnetic interference.
- 4). The bottom of module should avoid high-frequency digital routing, high-frequency analog routing and power routing. If it has to route the wire on the bottom of module, for example, it is assumed that the module is soldered to the Top Layer, the copper must be spread on the connection part of the top layer and the module, and be close to the digital part of module and routed in the Bottom Layer (all copper is well grounded).
- 5). Assuming that the module is soldered or placed in the Top Layer, it is also wrong to randomly route the Bottom Layer or other layers, which will affect the spurs and receiving sensitivity of the module to some degrees;
- 6). Assuming that there are devices with large electromagnetic interference around the module, which will greatly affect the module performance. It is recommended to stay away from the module according to the strength of the interference. If circumstances permit, appropriate isolation and shielding can be done.
- 7). Assuming that there are routings of large electromagnetic interference around the module (high-frequency digital, high-frequency analog, power routings), which will also greatly affect the module performance. It is recommended to stay away from the module according to the strength of the interference. If circumstances permit, appropriate isolation and shielding can be done.
- 8). It is recommended to stay away from the devices whose TTL protocol is the same 2.4 GHz physical layer, for example: USB 3.0.
- 9). The antenna installation structure has a great influence on the module performance. It is necessary to ensure the antenna is exposed and preferably vertically upward. When the module is installed inside of the case, a high-quality antenna extension wire can be

used to extend the antenna to the outside of the case.

10). The antenna must not be installed inside the metal case, which will cause the transmission distance to be greatly weakened.

15. Trouble Shooting

Unsatisfactory Transmission Distance

- 1). When there is a linear communication obstacle, the communication distance will be correspondingly weakened. Temperature, humidity, and co-channel interference will lead to an increase in communication packet loss rate. The performances of ground absorption and reflection of radio waves will be poor, when the module is tested close to the ground.
- 2). Seawater has a strong ability to absorb radio waves, so the test results by seaside are poor.
- 3). The signal attenuation will be very obvious, if there is a metal near the antenna or the module is placed inside of the metal shell.
- 4). The incorrect power register set or the high data rate in an open air may shorten the communication distance. The higher the data rate, the closer the distance.
- 5). The low voltage of the power supply is lower than the recommended value at ambient temperature, and the lower the voltage, the smaller the power is.
- 6). The unmatchable antennas and module or the poor quality of antenna will affect the communication distance.

Vulnerable Module

- 1). Please ensure the supply voltage is between the recommended values. The module will be permanently damaged if the voltage exceeds the maximum value. Please ensure the stable power supply and no frequently fluctuated voltage.
- 2). Please ensure the anti-static installation and the electrostatic sensitivity of high-frequency devices.
- 3). Due to some humidity sensitive components, please ensure the suitable humidity during installation and application. If there is no special demand, it is not recommended to use at too high or too low temperature.

High Bit Error Rate

- 1). There are co-channel signal interferences nearby. It is recommended to be away from the interference sources or modify the frequency and channel to avoid interferences.
- 2). The unsatisfactory power supply may also cause garbled. It is necessary to ensure the

power supply reliability.

3). If the extension wire or feeder wire is of poor quality or too long, the bit error rate will be high.

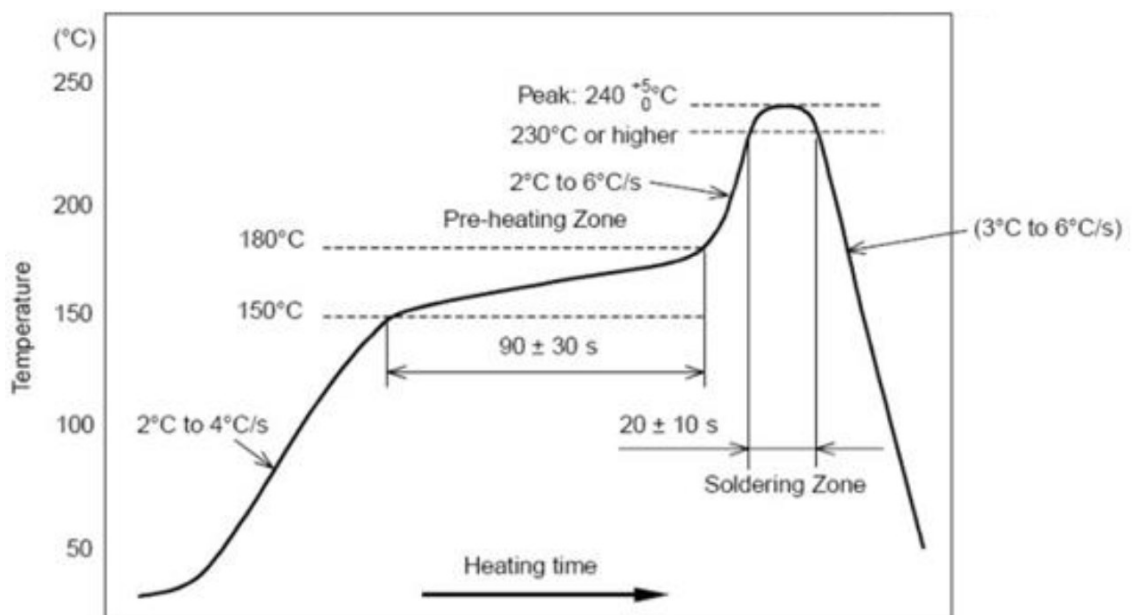
Electrostatics Discharge Warnings

The module will be damaged for the discharge of static. RF-star suggest that all modules should follow the 3 precautions below:

- 1). According to the anti-static measures, bare hands are not allowed to touch modules.
- 2). Modules must be placed in anti- static areas.
- 3). Take the anti-static circuitry (when inputting HV or VHF) into consideration in product design. Static may result in the degradation in performance of module, even causing the failure.

Soldering and Reflow Condition

- 1). Heating method: Conventional Convection or IR/convection.
- 2). Temperature measurement: Thermocouple $d = 0.1 \text{ mm}$ to 0.2 mm CA (K) or CC (T) at soldering portion or equivalent methods.
- 3). Solder paste composition: Sn/3.0 Ag/0.5 Cu
- 4). Allowable reflow soldering times: 2 times based on the following reflow soldering profile.
- 5). Temperature profile: Reflow soldering shall be done according to the following temperature profile.
- 6). Peak temperature: 245°C .



Recommended Reflow for Lead Free Solder

16. Contact us

Shenzhen DreamLnk Technology Co., Ltd

★ Data collection, Smart home, Internet of Things applications, Wireless remote control technology, Remote active RFID, Antennas ★

Office Add.: Room 603, Unit C, Zone A, Huameiju Business Center, Xinhua Rd., Bao'an District, Shenzhen, Guangdong Province, China

Factory Add.: 5th Floor, Building B, Huazhi Innovation Valley, No. 7 Yuhua Street, 138 Industrial Zone, Tangxia Town, Dongguan, Guangdong Province, China