低功耗蓝牙(BLE)模块及协议

协议版本：V2.3（透传+直驱）

目录

●概述

模块可以工作在桥接模式(透传模式)和直驱模式。模块启动后会自动进行广播，已打开特定 APP 的手机会对其进行扫描和对接，成功之后便可以通过 BLE 协议对其进行监控。

桥接模式下，用户 CPU 可以通过模块的通用串口和移动设备进行双向通讯，用户也可以通过特定的串口 AT 指令，对某些通讯参数进行管理控制。用户数据的具体含义由上层应用程序自行定义。移动设备可以通过 APP 对模块进行写操作，写入的数据将通过串口发送给用户的 CPU。模块收到来自用户 CPU 串口的数据包后，将自动转发给移动设备。此模式下的开发，用户必须负责主 CPU 的代码设计，以及智能移动设备端 APP 代码设计。

直驱模式下，用户对模块进行简单外围扩展，APP 通过 BLE 协议直接对模块进行驱动，完成智能移动设备对模块的监管和控制。此模式下的软件开发，用户只须负责智能移动设备端 APP 代码设计。

主要特点：

1. 使用简单，无需任何蓝牙协议栈应用经验；
2. 用户接口使用通用串口设计，全双工双向通讯，最低波特率支持 4800bps；
3. 同时支持桥接模式(串口透传)，或者直接驱动模式(无需额外 CPU)；
4. 默认 20ms 连接间隔，连接快速；
5. 支持 AT 指令软件复位模块，获取 MAC 地址；
6. 支持 AT 指令调整蓝牙连接间隔，控制不同的转发速率。（动态功耗调整）；
7. 支持 AT 指令调整发射功率，修改广播间隔，自定义广播数据，自定义设备识别码，设定数据延时(用户 CPU 串口接收准备时间)，修改串口波特率，修改模块名，均会掉电保存；
8. 串口数据包长度，可以是 200byte 以下(含 200)的任意长度。（大包自动分发）；
9. 高速透传转发，最快可达 4K/S，可稳定工作在 2.5K-2.8K（IO5,IO6）；
10. 支持移动设备 APP 修改模块名称，掉电保存，修改串口波特率，产品识别码，自定义广播内容，广播周期，均掉电保存；
11. 支持移动设备 APP 对模块进行远程复位，设置发射功率；
12. 支持移动设备 APP 调节蓝牙连接间隔，掉电不保存。（动态功耗调整）；
13. 包括调试口在内的全 IO 外扩；

14. 支持连接状态，广播状态提示脚/普通 IO 灵活配置；
15. 6 个双向可编程 IO，外部中断引发输入检测，全低功耗运行。（触发报警，照明控制，遥控玩具，等各种输入输出开关量应用）；
16. 2 个可编程定时单次/循环翻转输出口。（智能预约定时应用）；
17. 两路 ADC 输入(14bit),使能/禁止，采样周期自由配置。（测温湿度，光度等应用）；
18. 四路可编程 PWM(120Hz)输出。（调光，调速等应用）；
19. 模块端 RSSI 连续采集，可读可自动通知 APP，使能/禁止,采集频度自由设定。（寻物防丢报警应用）；
20. 支持模块电量提示，电量读取，可自动上报。（设备电量提醒）；
21. 支持防劫持密码设置，修改和恢复，防止第三方恶意连接。也可不使用。独立的密码操作结果通知，方便 APP 编程；
22. 支持单脚位下地(长按)5s 恢复出厂设置，APP 远程恢复出厂设置；
23. 支持 PWM 输出初始化状态自定义(全高，全低，掉电前 PWM 输出状态值)；
24. 支持 PWM 频率自定义（61.036 Hz<=f<=8 kHz，默认 120Hz）；
25. 广播内容提示模块实时系统状态，包括电池电量，自定义设备识别码，四路 PWM 当前输出值或两路 ADC 的采集值，当前 IO 状态等；（适合广播应用方案）；
26. 两路电平脉宽计数，0～0xFFFFFFFFms(约 49.7 天)；
27. 支持内部 RTC 实时时钟，APP 端可随时同步校准；
28. 支持 6 路 IO 和四路 PWM 的定时控制，默认不开启此功能；
29. 四路 PWM 支持渐变模式（适合调光效果控制）；
30. 支持 IO 配置和输出状态保存功能，可自定义默认的初始化状态；
31. 支持浅恢复和深度恢复模式，灵活恢复用户数据，而保留产品必须配置；
32. 支持从 TX 串口获取蓝牙连接状态（连接，正常断线和超时断线）字串提示；
33. 支持低电平使能模式和脉宽使能模式，支持远程关机；
34. 脉冲使能模式下支持 30 秒无连接自动关机；
35. 脉冲使能模式下支持方波报警提示连接超时(断线)；
36. 极低功耗的待机模式，CC2540 芯片官方数据睡眠电流 0.4uA，模块实测功耗如下：

事件	平均电流 (积分计算*1)	平均电流 (电表测量*2)	持续时间	测试条件/备注
模块睡眠功耗	0.36uA	0.3-0.41uA	—	EN 悬空
广播	205uA	0.15～0.55mA	3.86ms	广播周期 250ms
连接事件	245uA	0.41mA	2.24ms	连接周期 100ms
单次 BLE 数据接收事件	333uA	0.63mA	3.0ms	(20bytes,10 次/秒)
模块接收数据 并串口发送	498uA	2.69mA	5.2ms	(20bytes,10 次/秒)
单次 BLE 数据发送事件	345uA	0.68mA	3.2ms	(20bytes,10 次/秒)

*1 注：测试方式：在电源回路上串一个 10R 的电阻，使用示波器截取压降波形，进行积分计算。

*2 注：万用表测试方式：用万用表 uA 或 mA 档串在电池与模块之间查看显示值。测试电压为 3.07V。

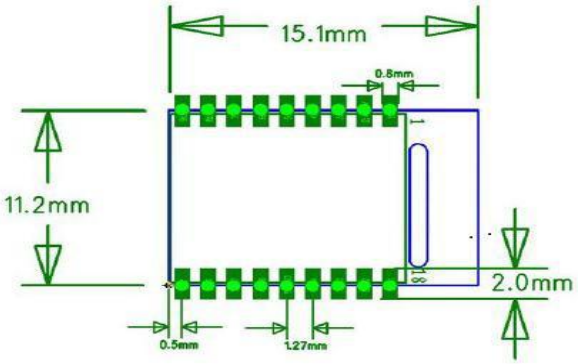
以上数据为 抽样实测数据，仅供参考。如果希望得到更低功耗，可适当增大连接间隔或者广播周期，详见《模块参数设置》和《串口 AT 指令》相 关章节。

●工作模式示意图

注:为避免用户 CPU 的 IO 和模块 IO 的输出电平差异导致大电流,建议在模块的输出信号线 TX,BCTS 上串入一小额隔离电阻。

●封装尺寸脚位定义

➢四层板工艺



模块脚位序号	模块脚位名称	芯片脚位名称	输入/输出	说明
Pin1	GND	GND	-	模块地 GND
Pin2	VCC	VCC	-	模块电源正极 2V-3.6V
Pin3	IO7	P2.2	O	输出口(可定时翻转)/睡眠状态指示
Pin4	IO6	P2.1	O	输出口(可定时翻转)/连接状态指示（低电平，或方波提示，详见《模块参数设置》章节）
Pin5	RES	RST	I	模块复位，低有效
Pin6	EN	P2.0	I	模块使能控制线，默认为电平触发模式 ➢ 电平触发模式，低电平有效，带内部上拉。 0：模块开始广播，直到连接到移动设备 1：无论模块当前状态，立即进入完全睡眠状态(0.4uA) ➢ 脉冲触发模式，每收到一次脉冲（W>200ms）,模块会在开机（进行广播，允许被发现和连接）以及关机（完全睡眠状态）之间循环切换 （关于模式的切换请参考《模块参数设置》相关章节）
Pin7	IO5	P1.7	I/O	➢ 可编程双向 IO，可通过 BLE 协议设置成输入或输出使用

				➢ 当做为输入时，可做为电平脉宽计数输入端
Pin8	U+	USB+	I/O	CC2540 引出脚 USB+，没使用
Pin9	U-	USB-	I/O	CC2540 引出脚 USB-，没使用
Pin10	REST ORE /IO0	P1.2	I/O	恢复出厂设置触发或可编程双向 IO ➢ 上电后 **30** 秒内，保持此引脚低电平 **5s**，系统会恢复部分参数（浅恢复），若保持 **20s** 以上则将会恢复全部参数（深度恢复）（见《系统复位与恢复》章节） ➢ 上电后 30 秒后，做为普通 IO 使用，可以通过 BLE 协议（见《可编程 IO(8 路)【服务 UUID：0xFFF0】》）设置成输入或输出使用
Pin11	PWM1	P1.1	O	PWM 输出通道 1
Pin12	PWM3	P0.7	O	PWM 输出通道 3
Pin13	PWM4	P0.6	O	PWM 输出通道 4
Pin14	BRTS	P0.5	I	作为数据发送请求（用来唤醒模块） 0：主机有数据发送，模块将等待接收来自主机的数据,此时模块不睡眠 1：主机无数据发送，或主机数据发送完毕之后，应该将此信 号线置 1
Pin15	BCTS	P0.4	O	数据输入信号（用来唤醒主机，可选） 0：模块有数据发送到主机，主机接收模块数据 1：模块无数据发送到主机，或模块数据发送完毕之后，会将此信号置 1
Pin16	TX	P0.3	O	模块串口发送端
Pin17	RX	P0.2	I	模块串口接收端
Pin18	ADC1	P0.1	I	模拟量采集，通道 1

注：由于是追求小尺寸的精简版，部分 IO 没有引出，对应功能无法使用。

●串口透传协议说明(桥接模式)

模块的桥接模式是指，通过通用串口和用户 CPU 相连，建立用户CPU 和移动设备之间的双向通讯。用户可以通过串口，使用指定的 AT 指令对串口波特率，BLE 连接间隔进行重设置(详见后面《串口 AT 指令》章节)。针对不同的串口波特率以及 BLE 连接间隔，以及不同的发包间隔，模块将会有不同的数据吞吐能力。为协调低速 CPU 的使用，默认波特率为 9600bps，在有大数据量传输，或者高实时性需求的应用中，建议设定为高速串口波特率 115200bps，支持掉电保存。

模块 **BLE** 连接间隔为 **20ms**，串口波特率为 **115200 bps** 时，模块具有最高理论转发能力(4K/S)。这里就在电平使能模式下，这种配置为例，对透传协议做详细介绍。

模块可以从串口一次性最多传输 200 字节数据包，模块会根据数据包大小自动分包发送，每个无线包最大载荷为 20 个字节。移动设备方发往模块的数据包，必须自行分包(1—20 字节/包)发送。模块收到无线包后，会依次转发到主机串口接收端。

1. 串口硬件协议：115200 bps , 8, 无校验位，1 停止位。
2. EN 为高电平，蓝牙模块处于完全睡眠状态。EN 置低时，模块会以 **200ms** 的间隔开始广播，直到和手机对接成功。当 EN 从低到高跳变，不论模块状态，会立即进入睡眠。
3. 连接成功之后，主机（MCU）如有数据发送至 BLE 模块，需将 BRTS 拉低，主机可在约 100**us** 后开始发送数据。发送完毕之后主机应主动抬高BRTS，让模块退出串口接收模式。要注意的是，抬高 BRTS 之前请确认串口数据完全发送完毕，否则会出现数据截尾现象。
4. 当模块有数据上传请求时，模块会置低 BCTS，最快会在 500**us** 之后开始发送，直到数据发送完毕。这个延时可以通过 AT 指令进行配置，见《串口 AT 指令》章节。数据发送完毕，模块会将 BCTS 置高。
5. 如若主机的 BRTS 一直保持低电平，则蓝牙模块会一直处于串口接收模式，会有较高的功耗。
6. 在模块连接成功后，会从 TX 给出**"TTM:OK\r\n\0"**字串，可以根据此字串来确定是否可以进行正常转发操作。当然也可以使用连接状态提示脚，也可以通过手机发送一个特定的确认字串到模块，主机收到后即可确认已经连接。当连接被 APP 端主动断开后，会从 TX 给出“**TTM:DISCONNET\r\n\0**”字串提示，如果是非正常断开，会从 TX 给出“**TTM:DISCONNETFORTIMEOUT\r\n\0**”字串提示。
7. 模块的蓝牙默认连接间隔为 **20 ms**，如果需要节省功耗采用低速转发模式，需通过AT 指令调整连接间隔（最长连接间隔 2000ms），每个连接间隔最多传输 80 个字节，连接间

隔为T(单位:ms),那么每秒最高转发速率**V**（单位**byte/s**）为：

V = 80*1000/T (V 只和 T 有关)

如果模块的蓝牙连接间隔为**20ms**，而每个间隔最多传输**80 byte**，因此理论最高传输能力(转发速率)为**80*50 =4Kbyte/s**。测试表明，转发速率在**2 K/s**以下，漏包机率很低。安全起见，无论是低速或者高速转发应用，都建议在上层做校验重传处理。

8. 以下是就20ms连接间隔的通讯模式举例，也可以自行配置。转发速率V0越低，丢包率越低：

通讯参考模式	BLE连接间隔T(ms)	理论最高转发能力V(byte/s) V=80*1000/T	串口包长度L(byte)	串口发包间隔TS(ms) 当L<80时，TS>=T 当 80<L<160时，TS>=T*2 当160<L<200时，TS>=T*3	实际转发速率V0(byte/s) V0=L*1000/TS	备注
1	20	4K	80	TS>=T即可，若取TS=20ms	80*1000/20=4K	TS偏小,不推荐
2	20	4K	200	TS>=T*3即可，若取TS=70ms	200*1000/70=2.8K	
3	20	4K	200	TS>=T*3即可，若取TS=80ms	200*1000/80=2.5K	
4	20	4K	80	TS>=T即可，若取TS=35ms	80*1000/30=2.6K	
5	20	4K	70	TS>=T即可，若取TS=30ms	70*1000/30=2.3K	
6	20	4K	60	TS>=T即可，若取TS=30ms	60*1000/30=2K	
7	20	4K	40	TS>=T即可，若取TS=30ms	40*1000/30=1.3K	
8	20	4K	20	TS>=T即可，若取TS=30ms	20*1000/30=666byte	

注：可以根据实际应用设计特定的通讯模式，串口包的长度可以设计在 80byte <L< 200byte 之间(大包传输)，根据 BLE 协议有以下关系：

当取 **L<80** 时，**TS>=T**；

当取 **80<L<160** 时，**TS>=T*2**；当

取 **160<L<200** 时，**TS>=T*3**；

满足以上条件的转发模式都是相对安全的，其中取 TS=T，TS=T*2，TS=T*3，可用但不推荐，丢包率相对较高，必须加入校验重发机制。也就是说，当串口包采用 80byte <L< 200byte 的大包时，串口数据可以一次性传递给模块，但需要预留模块通过蓝牙发送数据的时间，否则会出现追尾现象。如：在连接间隔设置为 T=20ms 时，如串口数据包长度选择 L=200，则 TS 必须大于 T*3=60ms，取 TS=70ms 是比较合理的选择。

9. 串口数据包的大小可以不定长，长度可以是 200 字节以下的任意值，同样满足以上条件即可。但为最大效率地使用通讯的有效载荷，同时又避免通讯满负荷运行，推荐使用 20，40，60 字节长度的串口数据包，包间间隔取大于 20ms。

注：经测试，在 **IOS** 中，调用对 **Characteristic** 的写函数使用 **CBCharacteristicWriteWithResponse** 参数，使用带回应写模式，这种模式会降低部分转发效率，但可保证单个数据包的正确性，而使用 **CBCharacteristicWriteWithoutResponse** 参数，使用不带回应写模式，这种模式会有利于提高转发效率，但数据包的正确性需要 **APP** 上层去校验。

●串口 AT 指令：

以“TTM”开头的字串会当成AT指令进行解析并执行，并从串口原样返回，之后会追加输出执行结果，"TTM:OK\r\n\0"或"TTM:ERP\r\n\0"等。不以**“TTM”**开头的串口数据包，将被视为透传数据。

➢连接间隔设定

向串口 RX 输入以下字串，设定 BLE 连接间隔：

"TTM:CIT-Xms"

其中 X="20"，"50"，"100"，"200"，"300"，"400"，"500"，"1000"，"1500"，"2000"，单位 ms。在执行完此指令之后，会从串口 TX 得到以下确认：

"TTM:TIMEOUT\r\n\0"表示更改超时，修改失败； "TTM:OK\r\n\0"表示更改成功，正以新的连接间隔在运行；这个连接间隔设定的成功与否取决于移动设备对连接间隔的限制，不同的 IOS 版本最大连接间隔也有不同。使用 iPhone4s（IOS5.1.1）中测试，最快支持 20ms，最慢支持 2s，另外，由于 BLE 协议内部机制，不同的连接间隔下此指令会有不同的执行效率。在 IOS5.1.1 中，从当前连接间隔为 2000ms 的情况下(最长 2000ms)，改变到其他连接间隔，可能最长需要等待约 100s 左右，而在其他高频度连接间隔（如：100ms）下执行此 AT 指令，会有很快的执行效率。

注：此连接间隔掉电不保存，并且更改指令只有在连接成功后有效。

➢模块重命名

向串口 RX 输入以下字串，- 以后为模块名，长度为 16 个字节以内，

"TTM:REN-"+Name

同样会从 TX 收到"TTM:OK\r\n\0"确认，如果指令格式不对，则会返回：

"TTM:ERP\r\n\0"

测试表明，由于 IOS 版本关系，设备名称修改在 IOS6 以上版本中可立即变更，在 IO5 中无法立即变更。此名称掉电保存。

➢波特率设定

向串口 RX 输入以下字串，- 后参数为新波特率，见 AT 指令表，如：

"TTM:BPS-115200"

之后会从 TX 收到"TTM:OK\r\n\0"确认，如果设置值不在选项中，或者指令格式不对，则返回：

"TTM:ERP\r\n\0"

测试表明，在 IOS5 中，设备名称修改无法成功，但在 IOS6 中可立即变更。用户可以通过 PC 进行设置后使用，也可以通过移动设备的 BLEAPP 接口进行设置。见《模块参

数设置【服务 UUID：0xFF90】》。

➢获取物理地址 MAC

向串口 RX 输入以下字串:

"TTM:MAC-?"

会从 TX 收到:

"TTM:MAC-xxxxxxxxxxxx\r\n\0"
字串后面"xxxxxxxxxxxx"为 6 字节模块蓝牙地址。

➢模块复位

向串口 RX 输入以下字串:

"TTM:RST-
SYSTEMRESET"

会迫使模块软复位一次。

➢广播周期设定

向串口 RX 输入以下字串，设置模块的广播周期，T=X * 100ms

"TTM:ADP-(X)"

其中 X="2","5","10","15","20","25","30","40","50"之一。会从 TX 脚收到"TTM:OK\r\n\0"确认，如果指令格式不对，则会返回：

"TTM:ERP\r\n\0"

广播周期设定掉电保存，重启模块后，模块将按照新的广播周期进行广播。

➢附加自定义广播内容

向串口 RX 输入以下字串，自定义广播内容

"TTM:ADD-"+Data

其中 Data 为准备附加的广播的数据，长度 0<L<=16 。会从 TX 脚收到"TTM:OK\r\n\0"确认，如果指令格式不对，则会返回：

"TTM:ERP\r\n\0"

此指令设置后立即生效，可以通过此功能广播一些自定义内容，数据掉电保存。如

果设置为 16 个全 0 数据，则认为不使用自定义广播数据，而是使用默认广播内容。

➢定义产品识别码

向串口 RX 输入以下字串，自定义广播内容

"TTM:PID-"+Data

其中 Data 为两个字节的产品识别码，范围 0x0000~0xFFFF(L=2) 。会从 TX 脚收到"TTM:OK\r\n\0"确认，如果指令格式不对，则会返回："TTM:ERP\r\n\0"此识别码掉电保存，会出现在广播中，可以以此来过滤设备或判断是否是特定的产品。

➢发射功率设定

向串口 RX 输入以下字串，设置相应的发射功率，单位 dBm。

"TTM:TPL-(X)"

其中 X="+4","0","-6","-23"，会从 TX 脚收到"TTM:OK\r\n\0"确认，并且模块立即使用新的发射功率进行通讯，如果指令格式不对，则会返回：

"TTM:ERP\r\n\0"

注：此参数掉电不保存。

➢数据延时设定

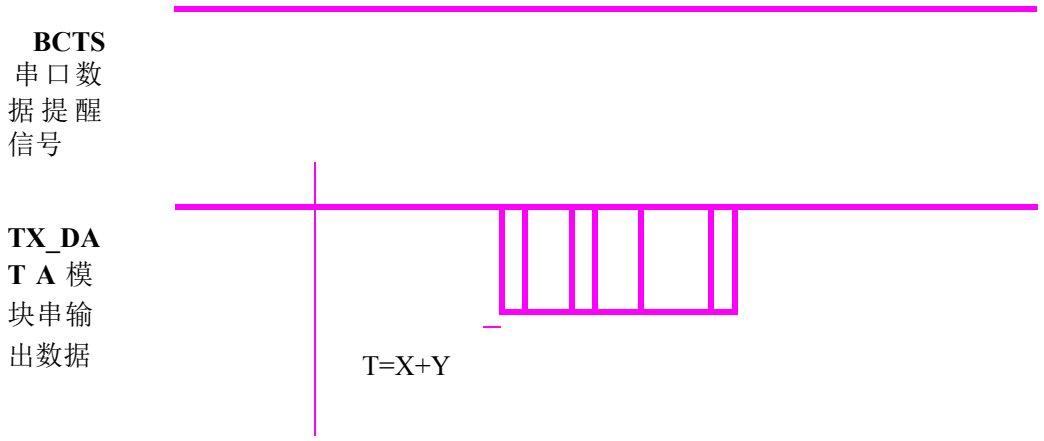
向串口 RX 输入以下字串，设置 BCTS 输出低到串口 TX 输出数据之间的延时，单位 ms

"**TTM:CDL-Xms**"

其中 X="0","2","5","10","15","20","25"之一，如果指令无误，会从 TX 收到"TTM:OK\r\n\0"确认，如果指令格式不对，则会返回：

"TTM:ERP\r\n\0"

为让用户 CPU 有足够的时间从睡眠中唤醒，到准备接收，模块提供了这个延时(X)设定，在模块串口有数据发出之前会置低 BRTS，而 BRTS 输出低到模块 TX 输出数据之间的延时由此参数设定。可以保证最小延时不小于 X，实际延时会是 T=(X+Y)ms，其中 500us<Y<1ms。此参数掉电保存。



模块串口输出数据延时设定示意图

AT 指令表

AT 指令格式	掉电保存	参数说明	可能的回应	含义
"TTM:CIT-Xms"(连接成功后才有效)	否	X="20"，"50"，"100"，"200"，"300"，"400"，"500"，"1000"，"1500"，"2000"设置相应的 BLE 连接间隔，单位 ms	"TTM:TIMEOUT\r\n\0" "TTM:OK\r\n\0"" TTM:ERP\r\n\0"	设置超时 设置成功 错误参数
"TTM:REN-"+Name	是	Name，新模块名，长度为 15 字节以内的任意字串。	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	设置成功错误参数
"TTM:BPS-X"	是	X="4800"，"9600"，"19200"，"38400"，"57600"，"115200"设置相应的波特率	"TTM:BPSSET AFTER2S...\r\n\0" "TTM:ERP\r\n\0"	设置成功,会在两秒后使用新的波特率 错误参数
"TTM:MAC-?"	—	获取 MAC 地址	"TTM:MAC-xxxxxxxxxxxx" xxxxxxxxxxxx 为模块 MAC 地址	返回 MAC 地址
"TTM:RST-SYSTEMRESET"	—	让模块系统复位	无	复位模块
"TTM:ADP-(X)"	是	X = "2","5","10","15","20","25","30","40","50" 设置相应的广播周期，T=X* 100ms	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	设置广播周期,如设置为"5"，则为 500ms

"TTM:ADD-" +Data	是	Data 为自定义广播数据，数据长度 L<=16；	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	设置自定义广播内容
"TTM:PID-" +Data	是	Data 为自定义产品识别码，数据长度 L=2，默认为 **0000**；	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	设置自定义产品识别码
"TTM:TPL-(X)"	否	X="+4","**0**","-6","-23"设置相应的发射功率，单位 dBm	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	发射功率设定
"TTM:CDL-X ms"	是	X="**0**","2","5","10","15", "20","25" 设置 BCTS 输出低到串口输出数据之间的延时，单位 ms	"TTM:OK\r\n\0"" TTM:ERP\r\n\0"	最小延时不于 X，实际延时会是 X+Y ms, 500us<Y<1ms.

* 注：蓝色粗体为默认设置。灰色提示指令，掉电不保存。

● 广播数据设置

默认广播数据：当模块的 EN 脚被置低后，或者进入 TEST 模式（拉低 TEST 后上电），模块将会进行间隔为 200ms 的广播，在广播数据中的

GAP_ADTYPE_MANUFACTURER_SPECIFIC（IOS编程中官方定义宏）

域中包含了以下内容，默认广播内容为 9 个字节：

{

0x00,0x00,	自定义设备类型编码，默认为 00 00 ,可由 AT 指令进行设定；
0x00,0x00,0x00,0x00,	四路 PWM 当前输出状态(默认)，或者两路 ADC 的采集值；
0x00,	模块供电电量百分比,2.0v =0%；
0x00,0x00,	IO 配置，IO 输出输入状态，此字节随 IO 当前状态实时变化；

}

广播中的数据会默认自动加载 PWM 当前输出状态，或者用户定义成两路 ADC 采集结果，为相同位置的四个字节。模块广播数据中总是自动加载最后操作过的通道对应的数据，对 PWM(FFB1)写任意值会导致加载四路 PWM 当前输出状态，或者对 ADC(FFD2)写非零值会导致加载两路 ADC 的采集值。

自定义广播数据：如果使用 AT 指令自定义了广播内容，最大长度为 16 字节(蓝色部分)，在广播数据中的 GAP_ADTYPE_MANUFACTURER_SPECIFIC 域中将包含了以下内容，长度为 2+n 个字节：

{

0x00,0x00,	自定义设备类型编码，默认为 00 00 ,可由 AT 指令进行设定；
Data[n],	自定义广播数据，n <=16 ;

}

注：自定义广播数据可通过 AT 指令修改，并且掉电保存。重新上电后，将会使用最后自定义的广播数据。如果自定义广播数据为全 0 (16byte)，则认为不使用自定义广播，而使用系统默认的广播内容。为避免广播数据过长带来多余的功耗，也可以通过设置自定义广播数据为 1 字节的任意值。

● 系统复位与恢复

让模块复位有三种方法，其中第三种方法可以恢复系统参数：

1. 使用 AT 指令复位模块(详见《串口 AT 指令》章节)；
2. 使用服务通道接口，用 APP 对模块进行远程复位。(详见《BLE 协议说明(APP 接口)一模块参数设置》章节)；
3. 使用硬件 RESTORE 脚位（见脚位定义表），上电 30 秒内，将此脚位拉低 5 秒后，模块的系统参数会恢复用户级参数（浅恢复，释放此脚位后立即复位），如果持续拉低 20 秒后会将模块的所有系统参数恢复到出厂设置（深度恢复），并立即复位。此脚位带内部上拉，默认不会进入此模式。浅恢复中被恢复的系统参数包括：

   a) 防劫持密码，恢复到“000000”，默认不使用密码；

   b) 四通道 PWM 初始化模式，恢复到 0x01，四通道都输出 100%高脉宽；c) IO 输出状态为 0，如果将 IO 配置成输出，则默认输出低电平；深度恢复中除了上述系统参数外，还包括以下参数：

   a) 串口波特率，恢复到 9600bps；

   b) 设备名称，恢复到"TAv22u-XXXXXXXX"，X 是 MAC 的后四个字节；

   c) 串口数据延时，恢复到 0 (500us<Delay<1ms)；

   d) 四通道 PWM 的输出频率，恢复到 0x8235 (120Hz)；

   e) 广播周期，恢复到 2 (200ms)；

   f) 产品识别码，恢复到 0x00,0x00；

   g) IO 配置字节为 0x00，默认 IO7,IO6 做信号提示脚，IO5-IO0 做输入；

   h) 自定义广播长度，恢复到 0；

   i) 自定义广播数据，恢复到全 0，不使用自定义广播数据，使用默认广播数据；

   j) 使能模式恢复到 0，默认电平使能模式；注：RESTORE（IO0）脚位的特殊性，在电路设计中，需避免上电前 30 秒持续下地，否则会进入恢复模式。

●IOSAPP 编程参考

模块总是以从模式进行广播，等待智能移动设备做为主设备进行扫描，以及连接。这个扫描以及连接通常是由 APP 来完成，由于 BLE 协议的特殊性，在系统设置中的扫描蓝牙连接没有现实意义。智能设备必须负责对 BLE 从设备的连接，通讯，断开等管理事宜，而这一切通常是在 APP 中实现。

有关 BLE 在 IOS 下的编程，最关键的就是对特征值(Characteristic，本文叫通道)的读，写，以及开启通知开关。通过对通道的读写即可实现对模块直驱功能的直接控制，无需额外的 CPU。典型函数说明摘抄如下：

```
/*!
 *  @methodwriteValue:forCharacteristic:withResponse:
 *  @paramdataThevaluetowrite.
 *  @paramcharacteristicThecharacteristicon whichtoperformthewrite operation.
 *  @paramtype Thetype ofwritetobe executed.
 *  @discussionWritethevalueofacharacteristic.
 *      Thepasseddatais copiedandcan bedisposedof afterthecall finishes.
 *      Therelevantdelegatecallbackwillthenbeinvokedwiththestatusoftherequest.
 *  @seeperipheral:didWriteValueForCharacteristic:error:
 */
-(void)writeValue:(NSData*)dataforCharacteristic:(CBCharacteristic*)characteristictype:(C
BCharacteristicWriteType)type;
```

说明：对某个特征值进行写操作。

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

```
/*!
 *  @methodreadValueForCharacteristic:
 *  @paramcharacteristicThecharacteristicforwhichthevalueneeds toberead.
 *  @discussionFetchthevalueofacharacteristic.
 *      Therelevantdelegatecallbackwillthenbeinvokedwiththestatusoftherequest.
 *  @seeperipheral:didUpdateValueForCharacteristic:error:
 */
-(void)readValueForCharacteristic:(CBCharacteristic*)characteristic;
```

说明：读取某个特征值。

[preadValueForCharacteristic:c];

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

说明：打开特征值通知使能开关。

[self setNotifyValue:YESforCharacteristic:c];//打开通知使能开关

[self setNotifyValue:NOforCharacteristic:c];//关闭通知使能开关

```
/*
 * @methoddidUpdateValueForCharacteristic
 * @paramperipheralPheripheralthatgotupdated
 * @paramcharacteristicCharacteristicthatgotupdated
 * @error errorError messageifsomethingwent wrong
 * @discussiondidUpdateValueForCharacteristiciscalledwhen CoreBluetoothhasupdateda
 * characteristicforaperipheral.Allreadsandnotificationscomeheretobeprocessed.
 *
 */
-(void)peripheral:(CBPeripheral*)peripheraldidUpdateValueForCharacteristic:(CBCharacteri
stic*)characteristicerror:(NSError*)error
```

说明：每次执行完读取操作后，会执行到这个回调函数应用层在此函数内保存读取到的数据。

●BLE 协议说明(APP 接口)

➢ 蓝牙数据通道【服务 UUID：0xFFE5】

特征值 UUID	可执行的操作	字节数	默认值	备注
FFE9 (handle: 0x0013)	Write	20	无	写入的数据将会从串口 TX 输出

说明：蓝牙输入转发到串口输出。APP 通过 BLEAPI 接口向此通道写操作后，数据将会从串口 TX 输出。详细操作规则见《串口透传协议说明(桥接模式)》章节。

➢串口数据通道【服务 UUID：0xFFE0】

特征值 UUID	可执行的操作	字节数	默认值	备注
FFE4 (handle: 0x000E)	notify	20	无	从串口 RX 输入的数据将会在此通道产生通知发给移动设备

说明：串口输入转发到蓝牙输出。如果打开了 FFE4 通道的通知使能开关（如果使用 BTool 操作，需向 0x000E+1=**0x000F** 写入 **01 00**），主 CPU 通过串口向模块 RX 发送的合法数据后，将会在此通道产生一个 notify 通知事件，APP 可以直接在回调函数中进行处理和使用。详细操作规则见《串口透传协议说明(桥接模式)》章节。

➢PWM 输出(4 路)【服务 UUID：0xFFB0】

特征值 UUID	可执行的操作	字节数	默认值	举例	备注	通道对应 2540 脚位
FFB1 (handle: 0x004D)	read /write	1	0x01	0x00	用全低脉宽初始化四路 PWM 通道	--
				0x01	用全高脉宽初始化四路 PWM 通道	
				0x02	用当前输出脉宽初始化对应的 PWM 通道	
FFB2 (handle: 0x0050)	read /write	4	0xFFFFFFFF	0xFF000000	PWM1 通道输出全高脉宽	P11
				0x00FF0000	PWM2 通道输出全高脉宽	P10
				0x0000FF00	PWM3 通道输出全高脉宽	P07
				0x000000FF	PWM4 通道输出全高脉宽	P06
				0x20202020	PWM1-PWM4 通道输出 32/256 脉宽	--
FFB3 (handle: 0x0053)	read /write	2	0x8235	500 <=w <=65535	PWM 输出信号频率设置，四路相同，默认为 0x8235 (120Hz)	--
FFB4 (handle: 0x0056)	read /write	2	0x0000	0 <=t <= 65535	PWM 转变时间宽度，四路相同，默认为 0x0000 (突变)	--

说明：

FFB1 为 4 通道 PWM 初始化模式设置通道。对 **FFB1** 通道进行写操作（1 bytes）即可配置四路 PWM 的初始化模式，出厂设置默认为0x01，全高脉宽输出，此设定值掉电

保存。

0x00，输出 0%脉宽, 全低脉宽，此模式下模块允许睡眠；

0x01，输出 100%脉宽, 全高脉宽，此模式下模块允许睡眠；

0x02，使用当前 PWM 值输出，设定后会立即保存当前的 PWM 输出值，做为下次上电后四路 PWM 的初始化值，此模式下模块不进入睡眠。

FFB2 为 4 通道 PWM 输出占空比设置通道。对 **FFB2** 通道进行写操作（4bytes）即可调节四路 PWM 的输出占空比，每个字节分别对应一个通道，0xFF 输出全高脉宽(100%高脉宽)，0x00 输出全低脉宽(0%高脉宽)。如设置为 X，则占空比约为 X/0xFF。同样可以对此通道进行读操作，将会得到最后设置值。上电后，默认为 0xFFFFFFFF，全高脉宽输出。开启此功能后，模块不进入睡眠，直到设置为 0xFFFFFFFF(全高)，即关闭PWM 输出。此通道是对四路 PWM 做占空比设置，设置范围为 0x00~0xFF，信号频率默认为 120Hz(见 FFB3 频率控制通道)。

例如：0xFF000000

1. 一共四个 PWM 输出通道；
2. 0xFF000000，四个字节分别对应四个通道；
3. 0xFF 输出全高脉宽 100%，0x00 对应全低脉宽 0%；
4. 默认脉宽频率为 120Hz。

FFB3 为 4 通道 PWM 输出频率控制。对 **FFB3** 通道进行写操作（2bytes）即可调节四路 PWM 的输出方波的频率，信号周期的宽度 w 必须满足：500<=w<=65535，一个单位对应 0.00000025s，对应方波周期：0.000125s<=T<=0.01638375s，因此方波信号频率的可调范围：61.036Hz<=f<=8 kHz，四路 PWM 输出方波频率相同。同样可以对此通道进行读操作，将会得到最后设置值，此设定值掉电保存。出厂设置默认 w 为 0x8235，对应默认脉宽频率为 120Hz。

例 1：输出频率为 120Hz 的方波。对 FFB3 通道写 0x8235 (33333)，设定方波周期为 0x8235 * 0.00000025s＝0.00833325 s ,即频率约为 120 Hz；

例 2：输出频率为 1kHz 的方波。对 FFB3 通道写 0x0FA0 (4000)，设定方波周期为 0x0FA0 * 0.00000025s＝0.001 s ,即频率约为 1 kHz；

FFB4 为 4 通道 PWM 输出转变时间长度控制。对 **FFB4** 通道进行写操作（2bytes）

即可调节四路 PWM 的输出方波的频率的变化速度，这是一个时间量 t，t 必须满足：0 <= t<=65535，一个单位对应 100 ms，t 越长，PWM 从当前值转变到目标值越慢，t 越小，转变得越快，当 t 为 0 时，就会立即突变到目标值。四路 PWM 转变时长共用此值。同样可以对此通道进行读操作，将会得到最后设置值，此设定值掉电保存。出厂设置默认 t 为 0x0000，对应转换模式为立即突变。

➢ADC 输入(2 路)【服务 UUID：0xFFD0】

特征值 UUID	可执行的操作	字节数	默认值	备注
FFD1 (handle: 0x0036)	Read/write	1	0x00	使能控制。 0x00:关闭两个 ADC 通道 0x01:打开 ADC0 通道 0x02:打开 ADC1 通道 0x03:打开两个 ADC 通道
FFD2 (handle: 0x0039)	Read/write	2	0x01F4	采集周期，单位 ms 如 0x01F4 对应 500 ms
FFD3 (handle: 0x003C)	Read/notify	2	0x0000	ADC0 采集结果,最大值 0x01FFF
FFD4 (handle: 0x0040)	Read/notify	2	0x0000	ADC1 采集结果,最大值 0x01FFF

说明：2 通道 ADC 输入控制。APP 通过 BLEAPI 接口向 FFD1 通道写操作，来使能两个 13bitADC 通道。向 FFD2 通道写操作，来控 制两个 ADC 通道采样周期 t,单位为 ms,t>=100ms。如果打开了通道 FFD3,FFD4 的通知使能（如果使用 BTool 操作，需向 0x003C+1=**0x003D** 和 0x0040+1=**0x0041** 写入 **01 00**），每产生一次采集结果后，将会在此通道产生一个 notify 通知事件，附带了本次采集结果,范围：0～0x1FFF，低字节在前，APP 可以直接在回调函数中进行处理和使用。ADC 的参考源为芯片内部参考源 1.25V，因此电源电压的浮动，不会导致新的测量误差，而被测量采样电压必须控制在 0～+1.25V 之间。

➢可编程 IO(8 路)【服务 UUID：0xFFF0】

特征值 UUID	可执行的操作	字节数	默认值	备注
FFF1 (handle: 0x0017)	Read/write	1	0b00000000	IO7~ IO0 的配置字。 当相应位被设置为 0 时：bit7,bit6 表 示 IO7,IO6 做为信号提示脚位，低电平有效 bit5~bit0 表示 IO5~IO0 做为输入口 当相应位被设置为 1 时：bit7,bit6 表 示 IO7,IO6 做为普通输出口 bit5~bit0 表示 IO5~IO0 做为输出口
FFF2 (handle: 0x001A)	write	1	--	IO7~ IO0 的输出状态。表示在 IO7~IO0 分别输出的电平,bit7 和 bit6 仅在 IO7,IO6 做为普通输出口时有效,做为信号提示脚位时 bit7 和 bit6 无效。
FFF3 (handle: 0x001D)	Read/notify	1	0x3F	IO5~ IO0 的输入状态。可以读取或接收通知。在打开通知使能的前提下，某个输入电平的变化都会通知到 APP。IO7,IO6 只能做为输出或者信号提示脚，对应位无效。

说明：IO 配置和控制通道。

FFF1 为 8 个 IO 的配置通道，8 bit 分别对应 IO7～IO08 个 IO 的配置控制，高两位 BIT7，BIT6 为 0 时，IO7 和 IO6 做为信号提示脚，IO7 提示睡眠状态，0 为唤醒态，1 为睡眠态；IO6 提示连接状态，0 为连接状态，1 为断开状态；高两位 BIT7，BIT6 为 1

时，IO7 和 IO6 则做为普通输出口使用，这两个口无法做为输入口使用。

低六位 BIT5～BIT0 为 1 时，IO5～IO0 做为输出口使用，为 0 时，IO5～IO0 做为输入口使用。

FFF2 为 8 个 IO 的输出设置通道，8 bit 分别对应 IO7～IO08 个 IO 的控制，仅当相应位被设置成输出时才有效。当某些 IO 被设置成输出时，可以向此通道的相应位进行写操作，便可实现对这些 IO 的输出控制，被设置成输入口的对应位无效。

注：IO 的配置(FFF1)以及输出状态(FFF2)默认掉电不保存，但可以通过向远程控制扩展通道 **FF99** 道写入 **0x01** 来保存当前除 IO0 以外的 IO1～IO7 的配置以及输出状态，模块上电后会使用最后保存的状态来初始化 7 个 IO。也就是说，IO0 的配置和输出状态无法掉电保存，IO0 上电后总是默认为输入状态，用来检测恢复出厂设置的功能。(详见《模块参数设置》有关章节)。

FFF3 为 IO5~IO0 的输入状态通道，低 6 位分别对应 IO5～IO0 的输入状态。仅当相应位被设置成输入时有效。如果 FFF3 通道的通知使能被打开（如果使用 BTool 操作，需向 0x001D+1=**0x001E** 写入 **0100**），当这些脚位上的电平发生改变，APP 端将会在此通道产生一个 notify 通知事件，附带了一个字节表示 6 个 IO 的状态，仅被配置成输入口的 IO 对应位有效，APP 端可以直接在通知的回调函数中，进行处理和使用此状态数据。IO7,IO6 只能做为输出或者信号提示脚，因此对应位无效。

➢定时翻转输出(2 路)【服务 UUID：0xFFF0】

特征值 UUID	可执行的操作	字节数	默认值	备注
FFF4 (handle: 0x0021)	Read/write	4	0x00000000	IO6 第一次翻转延时设置 0：不启动 IO6 翻转 非 0：ms,延时后翻转
FFF5 (handle: 0x0024)	Read/write	4	0x00000000	IO6 第二次翻转延时设置 0：不进行二次翻转 非 0：ms,延时后翻转
FFF6 (handle: 0x0027)	Read/write	4	0x00000000	IO7 第一次翻转延时设置 0：不启动 IO7 翻转 非 0：ms,延时后翻转
FFF7 (handle: 0x002A)	Read/write	4	0x00000000	IO7 第二次翻转延时设置 0：不进行二次翻转 非 0：ms,延时后翻转

说明：预约定时翻转配置通道。

模块的 IO6,IO7 当被设置成普通输出时，可以分别配置成定时翻转输出模式。可以通过对此通道进行写操作，来设定 IO6,IO7 的下次翻转时间，通过设置当前输出 IO 的状态，可以实现 1 到 0 的跳变，或者 0 到 1 的跳变。如果设置为 0,则不启动翻转。

此功能仅在 FFF1 高两位 BIT7,BIT6 被设为 1 时有效(做为输出口)。

FFF4 通道设定 IO6 第一次翻转的延时时间，FFF5 通道设定 IO6 第二次翻转的延时时间。如果 FFF4 设置为 0，则不启动 IO6 的翻转。如果 FFF4 设置为非 0，而 FFF5 通道设置为 0，则仅启动一次 IO6 的翻转。必须先设置 FFF5 通道，此时翻转未被启动，再设置 FFF4 通道为非 0 值来启动 IO6 的定时翻转。同样，可以通过对 FFF4 通道写入 0 来关闭 IO6 的定时翻转，此时以前写入 FFF5 通道的任意值将被清零。单位为 ms，范围为

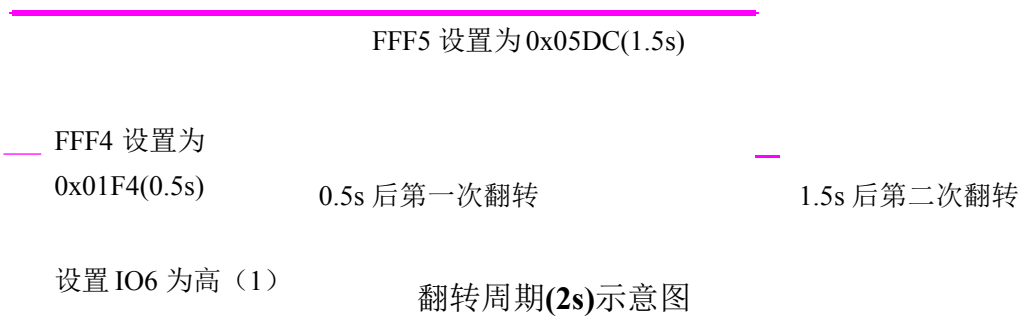
0 ～0xFFFFFFFFms(4294967295ms,约 1193 小时，约 49.7 天)，换算成十六进制为：

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

以 IO6 为例，设置一个周期性反复翻转，步骤如下：

1. 设置 IO6 为普通输出，向通道 FFF1 写入 0bx1xxxxxx;
2. 设置 IO6 当前为高(1)，通过向 FFF2 写入 0bx1xxxxxx;
3. 设置 FFF5 通道为 **0x05DC**(1.5s)，先设置第二次翻转延时，为 0 则只翻转一次
4. 设置 FFF4 通道为 **0x01F4**(0.5s)，再设置第一次翻转延时，同时会启动翻转

3，4 不能颠倒，必须先设置 FFF5，再通过对 FFF4 通道写非 0 值来启动翻转。向 FFF5 写入 0 值，表示只翻转一次。经过以上操作会得到一个周期为 1.5+0.5＝2s 的方波，其中高电平(1)将维持 0.5s,低电平(0)将维持 1.5s。可以向 FFF4 通道写入 0 来立即中止 IO6 当前的翻转行为，IO6 将会保持当前电平状态。



翻转周期**(2s)**示意图

FFF6，FFF7 为 IO7 的定时翻转延时设置通道，方法和 IO6 的设置一致。 注：

如果 IO6,IO7 处于定时翻转期间，对这两个 IO 的输出写以及重新配置成信号提醒操作均无效，操作前必须先停止当前定时翻转。

➢电平脉宽计数(2 路)【服务 UUID：0xFFF0】

特征值 UUID	可执行的操作	字节数	默认值	备注
FFF8 (handle: 0x002D)	Read/notify	4	0x00000000	IO4 之前电平保持的时间，单位 ms
FFF9 (handle: 0x0031)	Read/notify	4	0x00000000	IO5 上次电平保持的时间单位：ms

说明：计数 I0 电平持续时间通知通道。

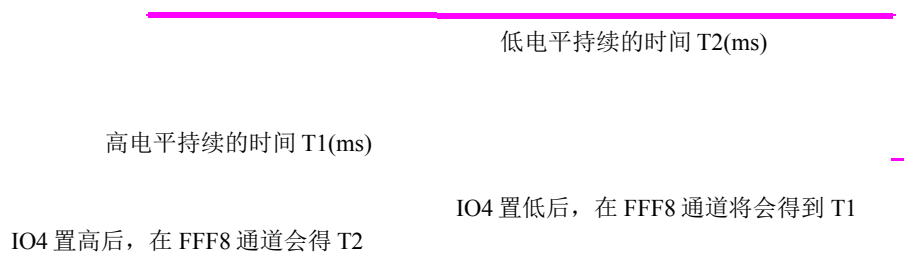
模块的 IO4,IO5 当被设置成普通输入时，可以开启电平脉宽计数模式。

此功能仅在 FFF1 的高两位 BIT5,BIT4 被设为 0 时有效(做为输入口)。

FFF8 通道为 IO4（P1.6）电平脉宽计数通知通道，APP 通过 BLEAPI 接口打开了此通道的通知使能（如果使用 BTool 操作，需向 0x2D+1=**0x2E** 写入 **01 00**），IO4 每次翻转后，会在此通道产生一个 notify 通知事件，附带了上个电平保持的时间宽度，最大值：0xFFFFFFFF(ms)，单位为 ms，范围为 0 ～0xFFFFFFFFms(4294967295ms,约 1193 小时，约 49.7 天),APP 可以直接在回调函数中进行处理和使用。

FFF9 通道为 IO5（P1.7）电平脉宽计数通知通道，APP 通过 BLEAPI 接口打开了此通道的通知使能（如果使用 BTool 操作，需向 0x31+1=**0x32** 写入 **01 00**），IO5 每次翻转后，会在此通道产生一个 notify 通知事件，附带了上个电平保持的时间宽度，范围为 0 ～0xFFFFFFFFms(4294967295ms,约 1193 小时，约 49.7 天), APP 可以直接在回调函数中进行处理和使用。

注：被计数的是上一个电平，不是当前电平。当前电平可以通过读取 FFF3 通道来获得。由于 BLE 的协议限制，采集结果的提交延时不会大于连接间隔时间。



电平脉宽计数示意图（以 **IO4** 为例，**IO5** 雷同）

➢防劫持密钥【服务 UUID：0xFFC0】

模块支持防劫持加密，此服务可以有效防止被非授权移动设备(手机)连接到此模块。模块的初始密码为 000000 (ASCII)，此情况下 APP 无需提交密码，视为不使用密码，任何安装指定 APP 的移动设备可以对其发起连接。

新密码（非全 0）的设置和备份保存由 APP 完成，如果设置了新密码（非全 0），开始启用防劫持密码。在 APP 对此模块进行连接后，必须在蓝牙连接后的 2 秒内向模块提交一次曾经设置的连接密码，否则模块会断开连接。在 APP 提交正确密码到模块之前，无法对服务通道进行任何除提交密码之外的写操作。

如果想恢复密码，需先重置模块，在 **30** 秒钟之内拉低 **RESTORE**（**IO0**）脚位（见脚位定义表），并保持 **5** 秒，模块密码会被恢复出厂设置。为了安全起见，模块不提供密码读操作，密码的记忆由 APP 来负责。

协议提供了密码通道来实现密码的提交，修改，和取消密码服务。同样也提供了密码事件通知服务来通知 APP 对密码操作的结果，其中包括密码正确，密码错误，密码修改成功，取消使用密码四个事件。

特征值 UUID	可执行的操作	字节数	举例	备注
FFC1 (handle: 0x0045)	write (掉电保存)	12	“123456123456”(ASCII)	提交当前密码 **123456**，新密码和旧密码必须一致
			“123456888888”(ASCII)	把旧密码 123456 修改为新密码 888888，旧密码必须正确
			“888888000000”(ASCII)	取消密码，新密码修改为 000000，旧密码必须正确
			0（PWD_RIGHT_EVENT）	提交密码正确

FFC2 (handle: 0x0048)	notify	1	1（PWD_ERROR_EVENT）	提交密码错误
			2（PWD_UPDATED_EVENT）	密码修改成功
			3（PWD_CANCEL_EVENT）	取消密码

说明：

1. 密码结构为 12 字节 ASCII 码，红色部分为当前密码，蓝色部分为新密码；
2. 当前密码在被 APP 修改之前，默认为“000000”；
3. 通过打开通道 **FFC2** 的通知使能（如果使用 BTool 操作，需向 0x0048+1= **0x0049** 写入 **01 00**），将会在此通道产生有关密码操作的执行结果通知。
4. 当 APP 提交密码“123456123456”，新密码和当前密码相同，APP 会在 FFC2 通道得到通知 notify:0(PWD_ RIGHT_EVENT)，表示提交密码正确；
5. 当 APP 提交密码（红色部分）和当前密码不一致，如：“123455xxxxxx”，x 部分不论是何值，APP 会在 FFC2 通道得到通知 notify:1(PWD_ ERROR_EVENT)，表示密码提交错误；
6. 当 APP 提交密码“123456888888”，新密码为“888888”，当前密码为“123456”，APP 会在 FFC2 通道得到通知 notify:2(PWD_ UPDATED_EVENT)，表示密码修改成功；
7. 当 APP 提交密码“888888000000”，新密码被修改为全 0，则表示取消使用密码，APP 会在 FFC2 通道得到通知 notify:3(PWD_ CANCEL_EVENT)。

➢电池电量报告【服务 UUID：0x180F】

特征值 UUID	可执行的操作	字节数	默认值	备注
2A19 (handle: 0x000A)	Read/notify	1	供电电量的百分比	读取当前电量的百分比，或者自动产生通知

说明：电池电量读取或通知通道。

APP 通过 BLE API 接口向 2A19 通道读操作，来获取当前模块的供电电量的百分比。如果打开了此通道的通知使能（如果使用 BTool 操作，需向 0x000A+1=**0x000B** 写入 **01 00**），每读取到一次电量后，将会在此通道产生一个 notify 通知事件，附带了电量百分比，最大值：100%(3V)，最小值：0%(2V),APP 可以直接在回调函数中进行处理和使用。

➢RSSI 报告【服务UUID：0xFFA0】

特征值 UUID	可执行的操作	字节数	默认值	备注
FFA1 (handle: 0x005D)	Read/Notify	1	0x00	RSSI 值，可以读取/自动通知
FFA2 (handle: 0x005A)	Read/write	2	0x0000	RSSI 自动读取周期设置，0x0000 为关闭自动读取。

说明：RSSI 读取或回传通道。

APP 通过 BLEAPI 接口向 FFA1 通道读操作，来获取当前模块收到移动设备的 RSSI。如果打开了此通道的通知使能（如果使用 BTool 操作，需向 0x005D+1=**0x005E** 写入 **0100**），每读取到一次 RSSI 后，将会在此通道产生一个 notify 通知事件，附带了 RSSI 值,APP 可以直接在回调函数中进行处理和使用。

APP 通过 BLEAPI 接口向 FFA2 通道读写操作，来设定 RSSI 的读取周期，单位为 ms。当此周期被设置为 0x0000 时，被认为关闭 RSSI 自动周期性读取。但仍然可以随时主动读取。RSSI 的读取值为signedchar 类型。

同样，停止使用 RSSI 回传功能，需关闭 FFA1 通道的 RSSI 通知使能，并向 FFA2 通道写入 0x0000,来关闭模块对 RSSI 的读取，否则会造成多余的功耗。

➢模块参数设置【服务 UUID：0xFF90】

特征值 UUID	可执行的操作	是否保存	字节数	默认值	备注
FF91 (handle: 0x0062)	Read/write	是	16	TAv22u-xxxxxxxx（带结束符的 ASCII 字串）	设备名称,xxxxxxxx 为物理地址的后四个字节
FF92 (handle: 0x0065)	Read/write	否	1	0	蓝牙通讯连接间隔： 0： 20ms 1： 50ms 2：100ms 3：200ms 4：300ms 5：400ms 6：500ms 7：1000ms 8：2000ms
FF93 (handle: 0x0068)	Read/write	是	1	1	设定串口波特率： 0：4800 bps 1： 9600 bps 2：19200 bps 3： 38400 bps 4： 57600 bps 5：115200 bps
FF94 (handle: 0x006B)	write	—	1	无	远程复位恢复控制通道： ➢ 远程复位控制，写入 **0x55** 对模块进行复位 ➢ 远程浅恢复控制，写入 **0x35** 对模块进行浅恢复（仅仅恢复用户数据），并复位 ➢ 远程深度恢复控制，写入 **0x36** 对模块进行深度恢复

					（让模块所有参数回到出厂设置），并复位
FF95 (handle: 0x006E)	Read/write	是	1	0	设定广播周期： 0：200ms, 1：500ms, 2：1000ms, 3：1500ms, 4：2000ms, 5：2500ms, 6：3000ms, 7：4000ms, 8：5000ms,
FF96 (handle: 0x0071)	Read/write	是	2	0x0000	设定产品识别码
FF97 (handle: 0x0074)	Read/write	否	1	1	设定发射功率： 0：+4dBm 1： 0 dBm 2：-6dBm 3：-23dBm
FF98 (handle: 0x0077)	Read/write	是	16	默认广播内容(详见《广播数据设置》章节)	设定自定义广播数据 自定义广播数据，0 <n <=16
FF99 (handle: 0x007A)	write	—	1	无	远程控制扩展通道： ➢ **0x01**：IO 配置输出保存触发控制，写入 0x01 可触发保存当前的 IO 配置以及输出状态，重新上电之后都会使用当前 IO 配置以及输出状态初始化 IO7~IO1，IO0 上电后总默认为输入，做为恢复出厂设置检测口 ➢ **0x02**：远程关机控制，当在脉冲使能模式下，向此通道

					写入 0x02，可对模块进行远程关机
FF9A (handle: 0x007D)	Read/write	是	1	0b**00000000**	系统功能使能开关： **BIT0**：使能模式设置，默认为0，对应低电平电平使能，1 表示脉冲使能，当 EN 脚每收到一个脉冲，模块将会在开机（开始广播）和关机（停止广播）之间轮流切换。有效脉宽 T，必须满足 W>200ms。当广播时间超过 30s，仍未被连接，则会自动进入关机状态。 **BIT1～BIT7**：暂未使用。

*注：灰色提示指令，掉电不保存

说明：模块信息配置通道。

FF91 为设备名称设置通道。可以通过对此通道进行读写操作，来获取和设定模块名称。设置的名称长度 L，必须满足 0<L<17，建议以结束符结尾（**'\0'**）。默认为“TAvvvv-xxxxxxxx\0”(16byte)，vvvv 为固件版本号，xxxxxxxx 为 MAC 地址后四个字节。

FF92 为模块连接间隔设置通道。可以通过对此通道进行写操作，来设定移动设备和模块之间的连接间隔，借此可以灵活控制设备功耗，以及数据吞吐量。为了提高连接速度，连接间隔参数不保存，上电后总以默认值(20ms)工作。测试表明，使用 iphone4s(IOS 5.1.1)从连接间隔为 500ms 修改为其他连接间隔，需要大约 30s 的等待时间。相反从高频度的连接间隔(如 20ms)进行变更，会有很高的执行效率(BLE 协议导致)。

FF93 为模块串口波特率设置通道。可以通过对此通道进行读写操作，来设定模块通用串口波特率，两秒后开始启用新的波特率，掉电保存。出厂设置默认为1(9600 bps)。

FF94 为远程复位恢复控制通道，通过写入不同值，可以实现不同的控制功能。

1. 对此通道写入 **0x55**，对模块进行软件复位。

2.过此通道写入 **0x35**，对模块进行浅恢复，所有用户参数将恢复到出厂设置控制，包括 IO 输出口的状态，PWM 的初始化模式，以及用户密码，之后会复位模块。

3.过此通道写入 **0x36**，对模块进行深度恢复，所有系统参数将恢复到出厂设置控制，

之后会复位模块。

FF95 为模块广播周期设置通道。可以通过对此通道进行读写操作，来设定模块广播周期。此参数掉电保存，出厂设置默认为 0 (200ms)。

FF96 为模块产品识别码设置通道。可以通过对此通道进行读写操作，来设定模块识别码，APP 端可以通过此识别码来进行过滤和连接指定的产品类型，此参数掉电保存。出厂设置默认为 0x0000。

FF97 为模块发射功率设置通道。可以通过对此通道进行写操作，来设定模块发射功率，此参数掉电不保存。出厂设置默认为 1 (0 dBm)。

FF98 为模块广播内容设置通道。可以通过对此通道进行写操作，来自定义模块的广播数据，此参数掉电保存。当数据为全 0（16 byte）时，认为不使用自定义广播数据，而使用默认的广播数据，详见《广播数据设置》章节。

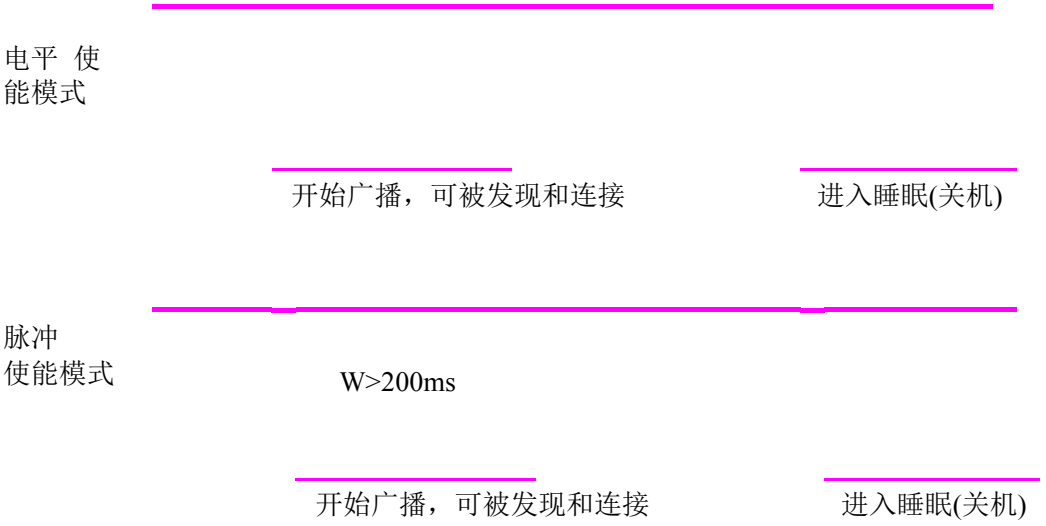
FF99 为远程控制扩展通道，通过写入不同值，可以实现特定的控制功能。对此通道写入 0x01，来触发模块保存当前除 IO0 以外的 IO 配置以及输出状态，在重新上电后模块总是使用保存过的 IO 配置以及输出状态初始化 IO7～IO1，IO0 却总是上电后默认为输入状态，用来做恢复出厂设置的触发 IO，上电之后，IO0 和其他 IO 一样可以被配置成输出 IO 使用。

对此通道写入 0x02，当在脉冲使能模式下，可对模块进行远程关机，在电平使能模式下无效。

FF9A 为系统功能使能开关通道，通过 BIT0~BIT7 的写操作可以开启和关闭系统的特定功能。1 为开启，0 为关闭。默认为全 0b00000000。此设置掉电保存。

BIT0：使能模式默认为 0，对应低电平使能（开始广播），高电平睡眠（0.4uA）。此位被置成 **1**，则模块会被设置为脉冲使能模式，每得到一次合法脉宽（W>200ms），模块将轮流在开（开始广播）和关（深度睡眠 0.4uA）之间切换。如果模块处于正连接状态，关无效。如果正处于广播状态，关有效。

BIT1~BIT7:保留。



电平使能模式和脉冲使能模式示意 在电平使能

模式下，广播(此状态下，可被发现，被连接)有以下特性：

1．如果 EN 脚被使能后（置低），模块会保持一直广播，直到被连接，或者 EN 被置高。

2．正常断开或者超时断开后，只要 EN 置低，模块总会保持广播，直到再次被连接。

在脉冲使能模式下，广播(此状态下，可被发现，被连接)有以下特性： 1．如果使能后持续广播 30 秒，仍没被连接，模块会停止广播进入关机状态。

2．正常断开后持续广播 30 秒，仍没被连接，模块会停止广播进入关机状态。

3．连接超时断开后会一直保持广播，直到再次被连接，EN 关机无效。

在电平使能模式下，IO6 做为信号提示引脚时（IO6 默认为蓝牙连接状态提示），当已连接输出低电平，当蓝牙未连接或者断开(超时断开和 APP 主动断开)后处于未连接状态时，输出高电平。

在脉冲使能模式下，IO6 做为信号提示引脚时（IO6 默认为蓝牙连接状态提示），输出信号有以下特性：

1．当已连接，会输出低电平脉冲（1s）一次。

2．当蓝牙正常断开（APP 主动断开）时，会输出低电平脉冲（0.5s）一次。

3．当蓝牙超时断开时，会输出 2Hz 的方波，这种提示会持续 2 分钟，期间会一直保持广播，并不可关机，直到模块重新连接上主设备。

不同使能模式下的广播状态和IO6的提示方式：

模块状态	使能后未连接		连接		正常断开		超时断开	
	IO6 提示方式	广播状态	IO6 提示方式	广播状态	IO6 提示方式	广播状态	IO6 提示方式	广播状态
电平使能模式	高电平	保持广播	低电平	停止广播	高电平	保持广播	高电平	保持广播
脉冲使能模式	高电平	广播 30 秒	一个低电平脉冲 w=1 秒	停止广播	一个低电平脉冲 w=0.5 秒	广播 30 秒	2Hz 方波持续 2 分钟	保持广播

➢设备信息【服务 UUID：0x180A】

特征值 UUID	可执行的操作	字节数	默认值	备注
2A23 (handle: 0x0003)	Read	8	xxxxxx0000xxxxxx (Hex)	系统 ID,xxxxxxxxxxxx 为模块芯片物理地址，低字节在前
2A26 (handle: 0x0005)	Read	5	V2.2u (ASCII)	模块软件版本号

说明：模块信息读取通道。

2A23 为模块信息获取通道，可以通过对此通道进行读操作，来获取此模块 ID。格式如 xxxxxx0000xxxxxx，其中 xx 部分为模块芯片的物理地址MAC，六个字节，低字节在前。

2A26 为模块软件版本号读取通道，可以通过对此通道进行读操作，来获取模块软件版本，格式为 Vx.xx。x.xx 为固件版本号。

➢端口定时事件配置【服务 UUID：0xFE00】

端口定时事件配置服务，用于设置 IO 或 PWM 端口的定时事件。这个服务提供了设置定时任务的功能，即：某个执行主体在某个时刻执行某个动作。执行主体可以是 10 个端口中的一个，包括 6 个突变输出口，和 4 路可渐变 PWM 输出通道，执行的动作类型可以是突变，或者是渐变。

1. 定时事件设置：

此服务提供了 32 个定时事件可以设置，事件是指在某个时刻执行某个特定动作。

定时事件**(EVT)** ＝执行时间+动作类型可通过事件读写通道（**UUID：0xFE03**）进行设置，其包含以下参数：

• 事件索引号，1 个字节，用来指示修改或设置的事件索引号；

• 执行时间（定时时间），7 个字节，用来指示事件触发的时间；

• 动作类型，1 个字节，用来指示当定时溢出时执行的动作，包含输出高电平，输出低电平，电平翻转，PWM 突变，PWM 渐变；

• 操作参数，3 个字节，用来设置 PWM 的目标占空比和渐变的开销时间，此参数和 6 个突变输出口无关，专门用来定义四路 PWM 通道的渐变行为的参数；

2. 定时任务设置

定时任务＝某个执行主体+某个定时事件

可配置成执行定时事件的端口（执行主体）包括 6 个 IO 口和 4 个 PWM 输出口。端口开启定时事件后，便形成定时任务。在定时事件触发时，端口将按照事件的定义执行动作。每个端口均可最多配置 32 个定时事件，并且有单独的响应开关，端口间的设置互不冲突，多个端口可以同时配置成相同的定时事件，但如果事件的动作类型对端口无效时，端口将忽略此定时操作，如 IO0 端口(无 PWM 输出功能)开启了某种 PWM 渐变事件，定时事件触发时，IO0 将忽略此事件。可通过端口事件读写通道（**UUID：0xFE05**）进行

设置，其包含以下参数：

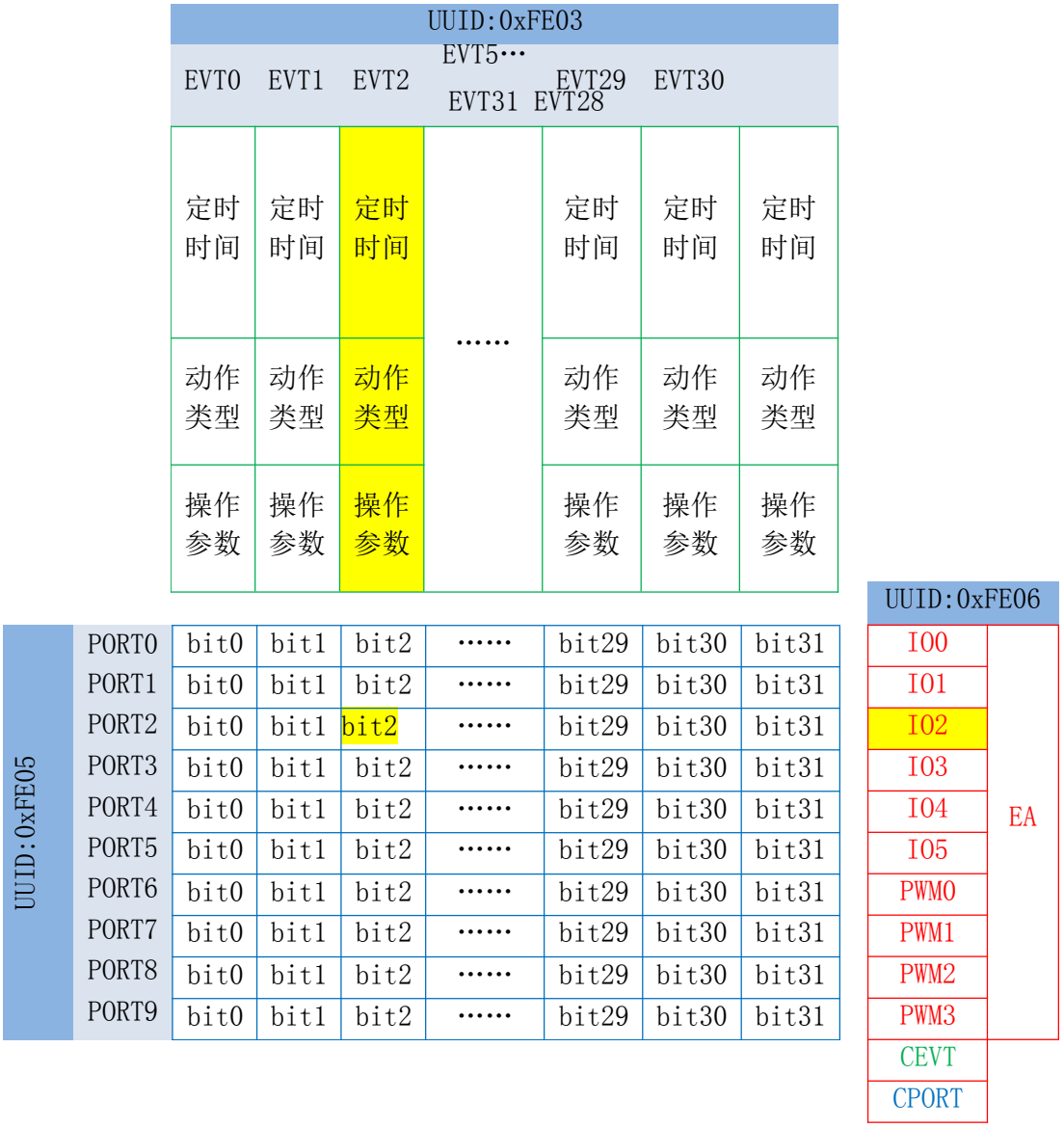
• 端口索引号，1 个字节，用来指示修改或设置的端口；

• 事件开启位，4 个字节，共 32 位，分别控制 32 个定时事件的响应开关，设置是否响应某个事件；

事件读写通道（UUID：0xFE03）和端口事件读写通道（UUID：0xFE05）是复用写入接口，每次被写入时，将第一个字节的“事件索引号”或“端口索引号”指向需要设置的事件或端口（相当于指针），后面其他字节为设置的具体细节。若想获取某个事件的定义或者某个端口的设置信息时，需先通过读事件指针通道（UUID：0xFE02）或读端口事件通道（UUID：0xFE04）写入希望读取的索引号后，再读取事件读写通道（UUID：0xFE03）和端口事件读写通道（UUID：0xFE05），以获取指定索引的相关信息。

3. 定时任务使能设置事件端口配置通道（UUID：0xFE06）用来控制定时任务（端口定时事件）的使能开

关，包括所有定时任务的总使能位（EA），六个 IO 口和 4 个 PWM 口的定时事件单独使能位，同时包括定时事件清空控制位（CEVT）和端口定时事件清空控制位（CPORT）。定时事件清空控制位和定时任务清空控制位被置位后，将清除所有 32 个定时事件和 10 个端口定时事件配置信息。



定时事件，定时任务，定时任务使能配置示意图

4. 定时事件 EVT 响应条件：
   i. 定时事件 EVT 的定时时间溢出触发；
   ii. 响应端口开启了指向定时事件 EVT 的响应开关位 BIT；
   iii. 指向响应端口的定时事件单独使能位 IO/PWM 使能； iv. 定时事件总使能位 EA 使能；

如上表，EVT2 定时触发，若 PORT2 开启了对应的 bit2，同时 IO2 位和 EA 位使能时，IO2 将触发 EVT2 的动作类型操作。

5. 关于优先级：

低索引事件或端口的动作会优先执行。如定时事件 1 和定时事件 2 的定时时间相同，端口 0 和端口 1 同时开启了两个定时事件，若两个定时事件在同一时刻触发时，执行端口的定时事件优先级如下：

1. 端口 0 的定时事件 1；
2. 端口 1 的定时事件 1；
3. 端口 0 的定时事件 2；
4. 端口 1 的定时事件 2；

注：

- ✓ 若想定时控制 IO 口，需先将相应 IO 口配置为输出口，可参看本文 BLE 协议说明的《可编程 IO（8 路）》章节。
- ✓ 定时功能在模块处于连接状态或非连接状态下均有效。
- ✓ 定时功能配置信息在模块掉电后不保存，如果丢失，可以用 APP 同步刷新。
- ✓ 为了避免 RTC 时钟的误差，建议 APP 连接或者断开前对 RTC 时钟进行同步更新。

特征值 UUID	可执行的操作	字节数	字节序号	默认值	含义	备注
FE01 (handle: 0x0086)	R/W	7	BYTE7 ～ BYTE0	0x07D001 01000000	秒分时日月 年(L)年(H)	RTC 时钟操作通道。 通过这个通道可以方便地读取和修改当前系统时钟。 取值范围： 秒：0~59； 分：0~59； 时：0~23； 日：1~31； 月：1~12； 年：2000 以上 默认时间为 2000 年 1 月 1 日 0 时 0 分 0 秒。
FE02 (handle: 0x0089)	R/W	1	BYTE0	0x00	事件索引号	读事件指针。 读取某个事件之前，必须设定此指针，让其指向需要读取的事件，再对 FE03 通道进行读操作。 取值范围： 0~31，分别表示 32 个定 时事件。

FE03 (handle: 0x008C)	R/W	12	BYTE0	0x00	事件索引号	事件读写通道。 对此通道进行读写，可以获取和设置定时 事件。 • 事件索引号： 取值范围： 0~31：分别表示 32 个定 时事件； • 定时时间（秒分时日月年）： 取值范围与 RTC 时钟的一样，若其中一个字节无效(FF 表示无效)，低位的有效字节时间将作为循环定时。如以下定时时间（Hex）： 00 FF 01 01 01 D0 07 此定时事件将在任意分钟的 0 秒时刻触发； • 动作类型： 取值范围： 0：无动作； 1：IO 输出低电平； 2：IO 输出高电平； 3：IO 电平翻转； 4：PWM 突变； 5：PWM 渐变；
			BYTE7 ～ BYTE1	0x000000 00000000	秒分时日月 年(L)年(H)	
			BYTE8	0x00	动作类型	
			BYTE9	0x00	PWM 新占空值	

			BYTE10	0x00	PWM 渐变时长的低字节	• PWM 新占空值： 动作类型为 4 或 5时有效，变化后的目标占空值，取值范围：0~255；0 时占空比为 0%，即全低电平，255 时占空比为 100%，即全高电平； • PWM 渐变时长：操作值为 5 时有效，从当前占空值变化到新的占空值所开销的时间，此值越大变化得越慢，此值越小变化得越快。取值范围：0~65535，单位为 100ms；
			BYTE11	0x00	PWM 渐变时长的高字节	
FE04 (handle: 0x008F)	R/W	1	BYTE0	0x00	端口索引号	端口事件读指针。 端口索引：读取某个端口所有适用的事件之前，必须设定此指针，让其指向需要读取的端口，再对 FF05 通道进行读操作。 取值范围： 0~9：分别表示 IO0~IO5 和 PWM0~PWM3 的定时端口。
FE05 (handle: 0x0092)	R/W	5	BYTE0	0x00	端口索引号	端口事件读写通道。 • 端口索引号 取值范围： 0~9：分别表示 IO0~IO5 和 PWM0~PWM3 的定时端 口。 • 适用事件使能开关，不同位分别对应 0~31

<table>
<tr><td></td><td></td><td></td><td>BYTE4
～
BYTE1</td><td>0b00000000
00000000
00000000
00000000</td><td>定时事件使能
0~31</td><td>个定时事件：
取值范围：
1：开启；
0：关闭；</td></tr>
<tr><td rowspan="16">FE06
(handle:
0x0095)</td><td rowspan="16">R/W</td><td rowspan="16">2</td><td rowspan="8">BYTE0</td><td rowspan="8">0b00000000</td><td>Bit0:定时总使能</td><td rowspan="16">事件端口配置字。

取值范围：
1：使能；
0：关闭；

• BYTE0:BIT0,为端口定时事件总使能（EA）；

• BYTE0:BIT1 ～ BIT7,BYTE1:BIT0～BIT2,为端口定时事件单独使能；

• BYTE1:BIT3,为定时事件清空控制位，清除所有已设置定时事件；(CEVT)

• BYTE1:BIT4,为定时任务清空控制位，清除所有端口对任意定时事件的响应配置；(CPORT)</td></tr>
<tr><td>Bit1:IO0 使能</td></tr>
<tr><td>Bit2:IO1 使能</td></tr>
<tr><td>Bit3:IO2 使能</td></tr>
<tr><td>Bit4:IO3 使能</td></tr>
<tr><td>Bit5:IO4 使能</td></tr>
<tr><td>Bit6:IO5 使能</td></tr>
<tr><td>Bit7:PWM0 使能</td></tr>
<tr><td rowspan="8">BYTE1</td><td rowspan="8">0b00000000</td><td>Bit0:PWM1 使能</td></tr>
<tr><td>Bit1:PWM2 使能</td></tr>
<tr><td>Bit2:PWM3 使能</td></tr>
<tr><td>Bit3:定时事件清空控制</td></tr>
<tr><td>Bit4:定时任务清空控制</td></tr>
<tr><td>-</td></tr>
<tr><td>-</td></tr>
<tr><td>-</td></tr>
</table>

提示：如果想实现一个定时任务的四个步骤： 1.设计一个或者多个定时事件（指定什么时间执行什么动作）（写 0xFE03）；

2.指定由哪个 IO 口来执行这个定时事件（建立定时事件和执行主体的关系）（写 0xFE05）；

3. 打开这个 IO 口对定时任务的响应使能开关（允许响应）（写 0xFE06）；

4. 打开定时任务总使能开关（写 0xFE06）。

举例：

例 1：如下图，在 2013 年 1 月 2 日 4 时 4 分 5 秒时刻设置 IO2 翻转一次。

IO2

2013 年 1 月 2 日
3 时 4 分 5 秒

2013 年 1 月 2 日
4 时 4 分 5 秒

BLE 主设备对从模块的操作步骤如下：

- ✓ 向 IO 配置字（UUID：0xFFF1）写入 0x04，配置 IO2 为输出口；
- ✓ 向事件读写通道（UUID：0xFE03）写入如下数据，设置定时事件 0 在 2013 年 1 月 2 日 4 时 4 分 5 秒时刻触发 IO 翻转操作： Hex（低字节在前）：00 0504 0402 01 DD 07 0300 0000。
- ✓ 向端口事件读写通道（UUID：0xFE05）写入如下数据，开启 IO2 定时端口的定时事件 0：

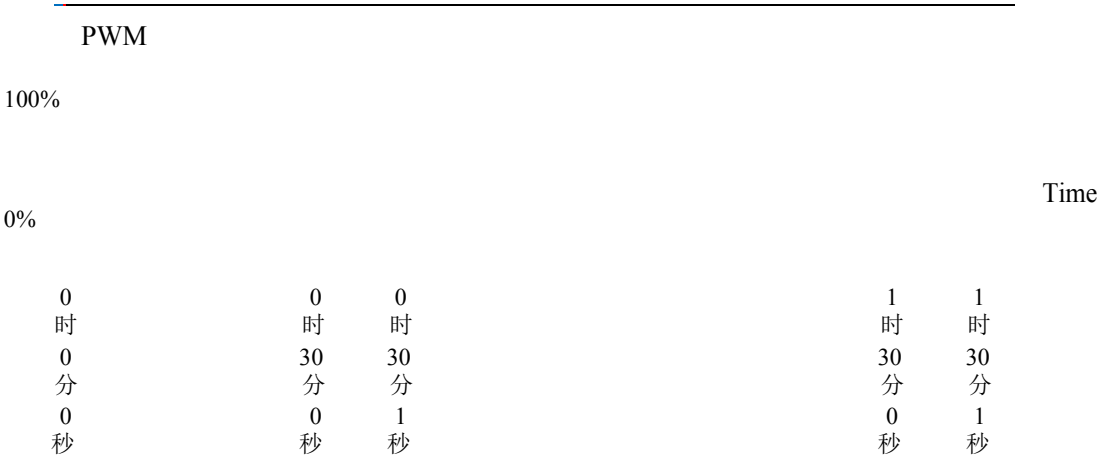
  Hex（低字节在前）：02 0100 0000。
- ✓ 向事件端口配置字（UUID：0xFE06）写入如下数据，使能端口定时事件总使能位和 IO2 使能位：

  Hex（低字节在前）：09 00。
- ✓ 向 RTC 时钟操作通道（UUID：0xFE01）写入如下数据，更新模块 RTC 时钟，如 2013 年 1 月 2 日 3 时 4 分 5 秒：

  Hex（低字节在前）：05 04 03 02 01 DD 07。

至此，已完成定时功能设置，等待定时事件触发。

例 2：如下图，在每小时的 30 分 0 秒时刻设置 PWM0 占空比突变到 100%，然后占空比渐变到 0%，渐变的开销时间为 1 秒钟。



BLE 主设备对从模块的操作步骤如下： 1.向事件读写通道（UUID：0xFE03）写入如下数据，设置定时事件 1 在每小时的 30 分 0

秒时刻触发 PWM 突变操作，占空比为 100%： Hex（低到高

字节）：01 00 1EFF FF FF FFFF 04 FF00 00。

2.向事件读写通道（UUID：0xFE03）写入如下数据，设置定时事件 2 在每小时的 30 分 0

秒时刻触发 PWM 渐变操作，占空比为 0%，渐变时长为 1 秒钟： Hex（低到高字节）：

02 00 1EFF FF FF FFFF 05 00 E8 03。

3.向端口事件读写通道（UUID：0xFE05）写入如下数据，开启 PWM0 定时端口的定时事

件 1 和定时事件 2：

Hex（低到高字节）：06 06 00 00 00。 4.向事件端口配置字（UUID：0xFE06）

写入如下数据，使能端口定时事件总使能位和

PWM0 使能位： Hex（低到高

字节）：81 00。

5.向 RTC 时钟操作通道（UUID：0xFE01）写入如下数据，更新模块 RTC 时钟，如 2013 年 1

月 2 日 3 时 4 分 5 秒：

Hex（低到高字节）：05 04 03 02 01 DD 07。

至此，已完成定时功能设置，等待定时事件触发。

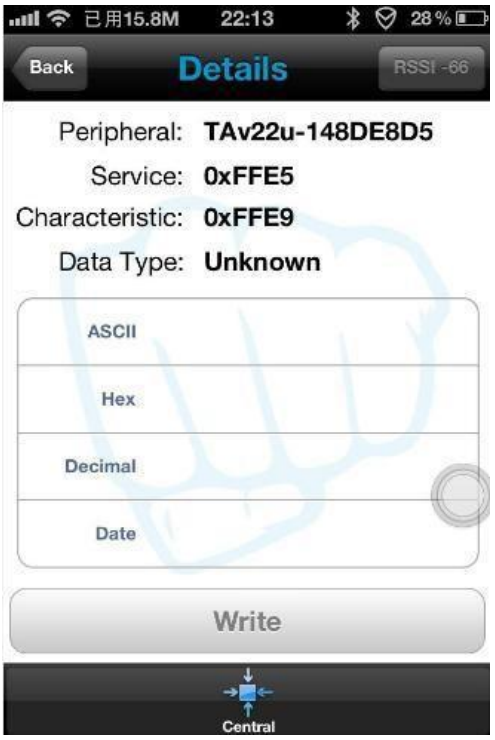
●用 APP 测试透传功能

使用 iPHONE 4S，iPHONE5，iPHONE5S　IPADMINI 或者 iPAD4 安装测试软件 LightBule。

APP 打开后会自动进行扫描，扫描到的设备会出现列表中(或许会提示需要打开蓝牙)，点击某个设备，会进行连接，连接成功后会跳转到服务控制主界面。

接收流程：选择 0xFFE0->0xFFE4,然后点击 StartNotiry，便可接收来自主机发来的字符。

发送流程：选择 0xFFE5->0xFFE9,然后点击 Write，便可向主机发送字符。



●用 USBDongle 及 Btool 测试

BLE 模块可使用 TI 官方 CC2540MinDK 开发套件中的 USBDongle 模拟手机配合安装目录下的 C:\TexasInstruments\BLE-CC254x-1.2.1\Projects\Btool\Btool.exe 进行蓝牙通讯测试。

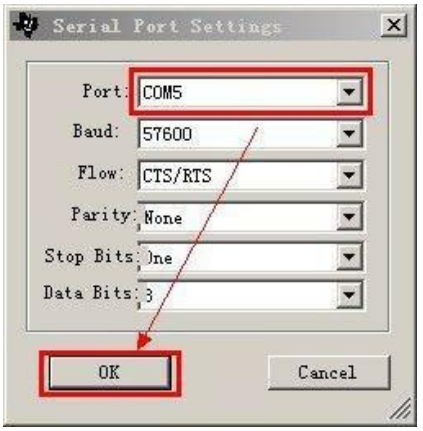
这个 USBDongle 需要使用安装目录下 C:\TexasInstruments\BLE-CC254x-1.2.1\Projects\ble\HostTestApp\CC2540 的工程项目。 编译下载到 USBdongle 中。具体的 BTOOL 的使用详情请参考官方说明文档

CC2540MiniDevelopmentKit User'sGuide(Rev.B).pdf

➢连接 BLE 模块

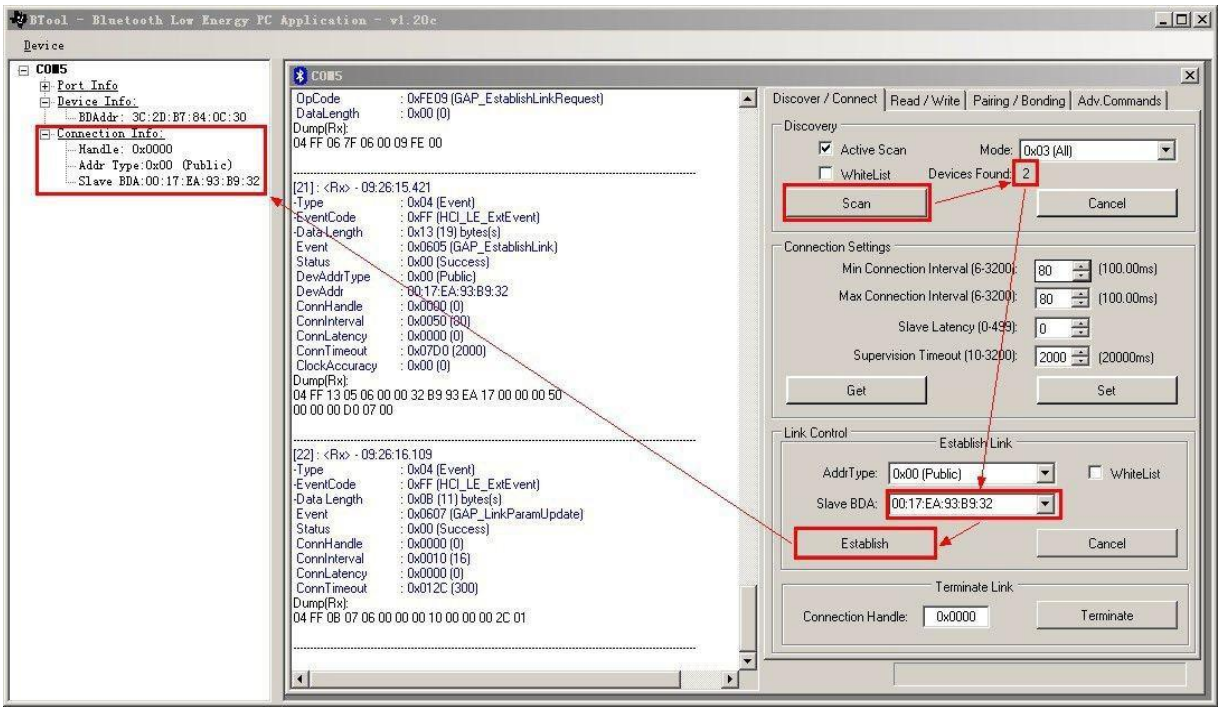
USBDongle 和模块的连接是通讯的基础，扫描连接的操作步骤如下：

1. 打开 C:\TexasInstruments\BLE-CC254x-1.2.1\Projects\ble\HostTestApp 目录下的工程文件，编译，下载到 USBDongle 中；
2. 将模块上电(3 ～ 3 . 3 v)；
3. 将模块使能脚 EN 下地，模块开始广播；
4. 将 USBDongle 插入 PCUSB 口，会在硬件管理中出现一个串口设备（如：COM5）；
5. 打开 C:\TexasInstruments\BLE-CC254x-1.2.1\Projects\Btool\Btool.exe；
6. 菜单 Device->NewDevice，选择 4 中发现的串口，选默认设置，OK；



7. 扫描连接,安照箭头的方向进行扫描，连接，其中 00:17:ea:93:b9:32 的模块的物理地址。连接前请确认是目标模块。

8. 连接成功后，左边会出现已经连接的模块信息 ConnectionInfo。



这样就已经成功连接了，下面就可以开始测试直驱功能以及蓝牙串口转发功能(透传)。

➢测试直驱功能

使用 Btool 和 Dongle 可以访问任何协议内定义的通道，经典步骤如下：

1) 通过 UUID 发现通道的 handle
2) 记住通道对应的 handle
3) 利用 **handle+1** 打开通道通知开关
4) 利用 UUID 或者 handle 对通道读
5) 利用 handle 对通道写

其中，如果直接使用《BLE 协议说明(APP 接口)》中的特征值信息表中提供的 handle, 可以跳过 1，2 两步。通道的通知开关必须通过 **handle+1** 指针操作。这里说的通道，就是指特征值(Characteristic)。写通道时，字节数必须和协议定义长度一致，否则会认为不合法而失败。

重要提示：BTool 最核心的操作就是先连接，再对某个通道的 handle 进行读和写，以及打开通知开关三个动作。前面几步(1,2)是为了寻找到对应的 handle，在 IOS 编程中，不需要寻找 handle，而是直接通过通道 UUID 进行读写。

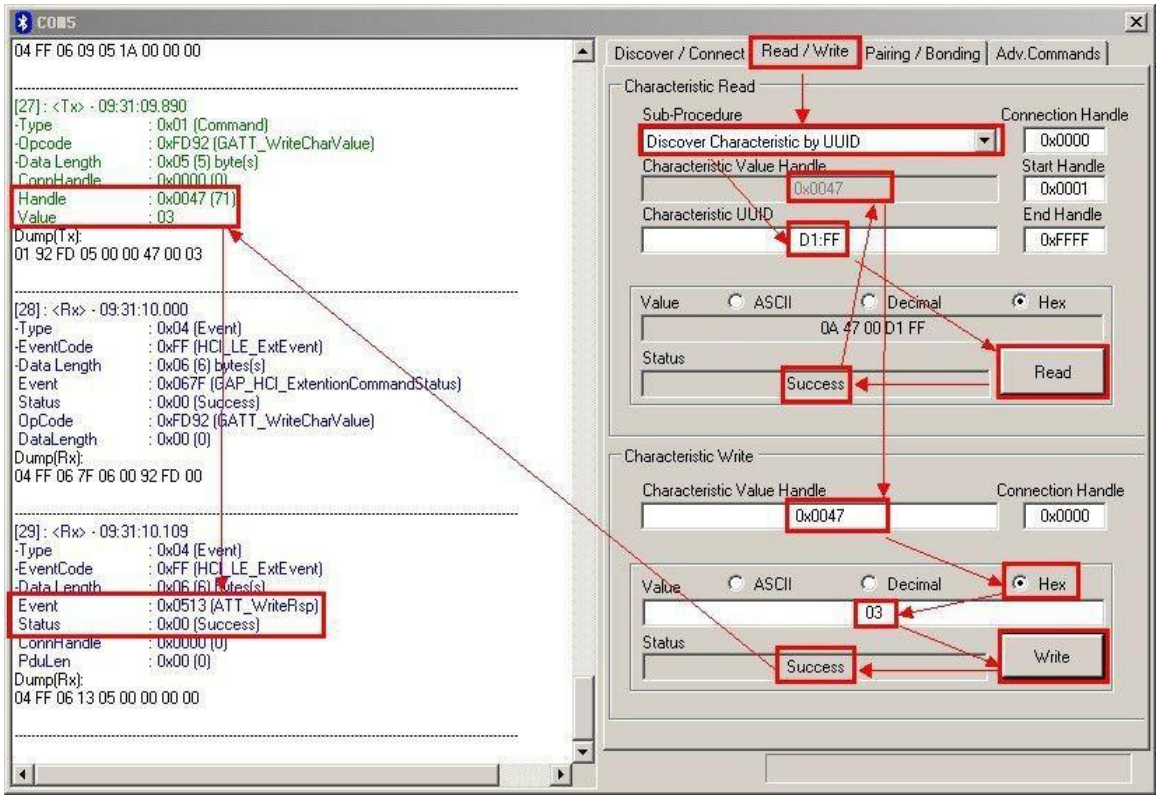
现以 ADC 为例，详述如何使用 USBDongle 以及 Btool 软件工具来直接驱动蓝牙模块。下例以英科联 BLE 透传模块 V2.0 为例，对应 Handle 和其他版本可能略有不同，但操作流程完全一致。其他功能测试方法类似，只是对应的通道 **UUID** 不同，读写方法相同。

ADC 输入(2 路) 【服务 UUID：0xFFD0】

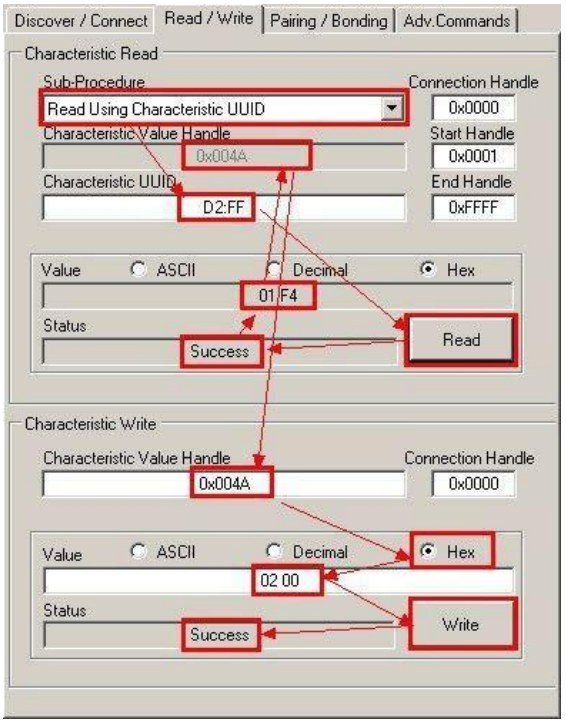
特征值 UUID	可执行的操作	字节数	默认值	备注
FFD1 (handle: 0x0047)	Read/write	1	0x00	使能控制。 0x00:关闭两个 ADC 通道 0x01:打开 ADC0 通道 0x02:打开 ADC1 通道 0x03:打开两个 ADC 通道
FFD2 (handle: 0x004A)	Read/write	2	0x01F4	采集周期，单位 ms 如 0x01F4 对应 500 ms
FFD3 (handle: 0x004D)	Read/notify	2	0x0000	ADC0 采集结果,最大值 0x01FFF
FFD4 (handle: 0x0051)	Read/notify	2	0x0000	ADC1 采集结果,最大值 0x01FFF

1. 打开 ADC0 和 ADC1。如图所示，后续操作都可以参考红色箭头的方向进行操作。

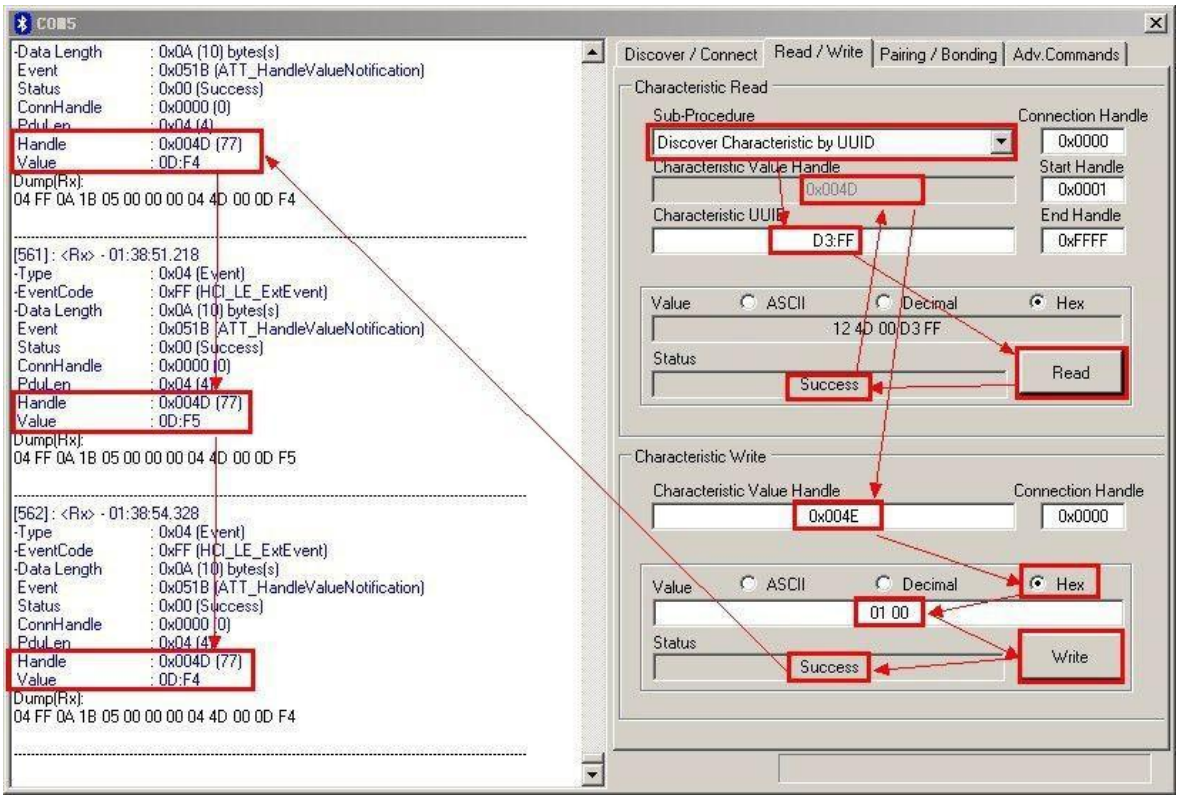
首先寻找需要控制的通道(UUID)，输入需要寻找的 UUID，低位在前 D1:FF，读取成功后，得到 handle=0x0047,向 0x0047 写入 03，这样就打开了 ADC0 和 ADC1,见上表。



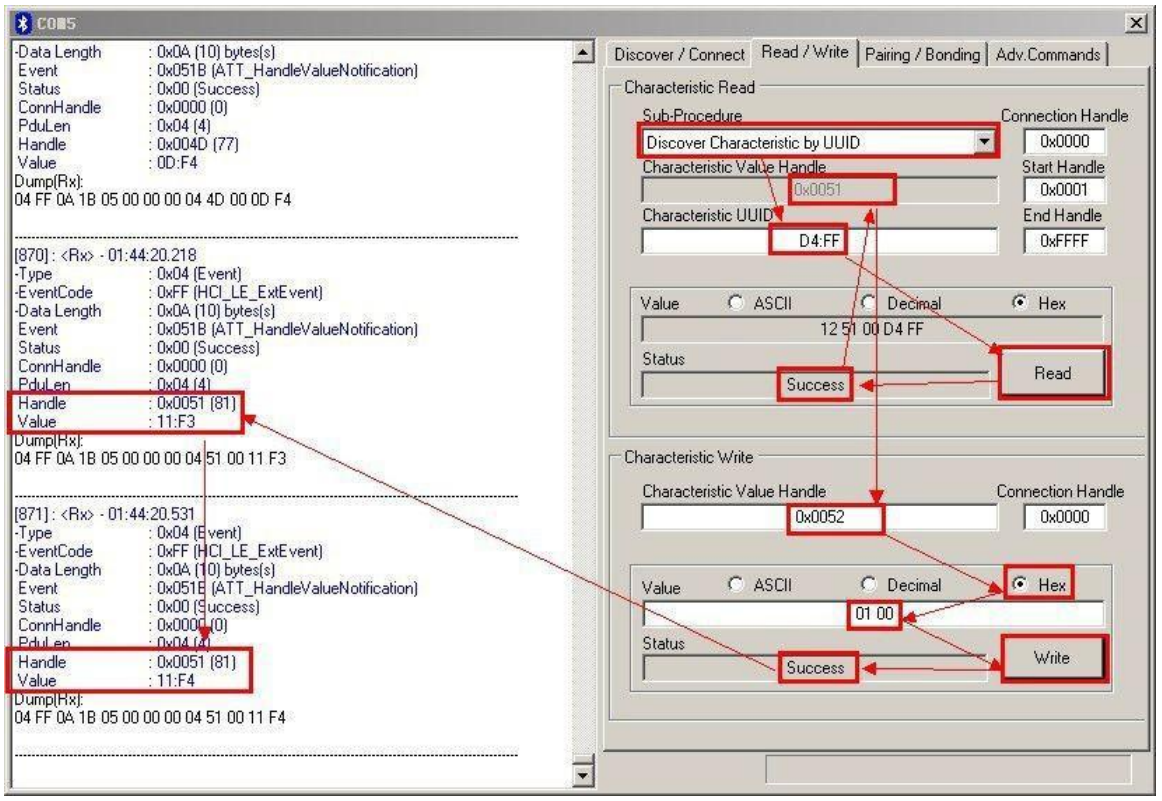
2. 读取和重新设定采集周期，先读取 FFD2 通道，可以得到 01F4，表示默认为 500ms，通过得到的 handle=0x004A 写入 02 00 高位在前，（0x0200 =512 ms），设置每 512ms 采集一次。



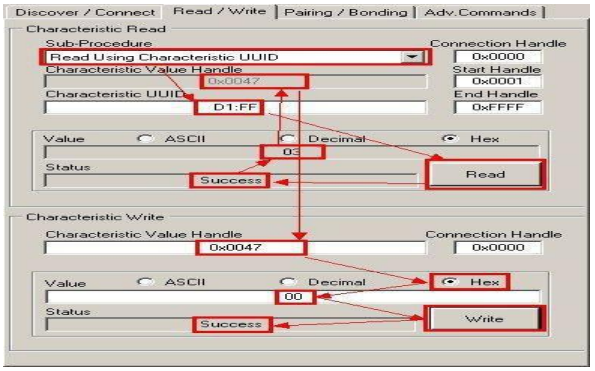
3. 打开 ADC0 通道通知使能开关，通知开关的地址为 **handle+1**，0x004D+1=0x004E，此后，ADC0 每次采集值不同于上一次，则会产生一次新的通知



4. 打开 ADC1 通道通知使能开关，通知开关的地址为 **handle+1**，0x0051+1=0x0052，此后，ADC1 每次采集值不同于上一次，则会产生一次新的通知。



5. 停止 ADC0 和 ADC1。和打开 ADC0 和 ADC1 类似。向 0x0047 写入 00，这样就关闭了 ADC0 和 ADC1。

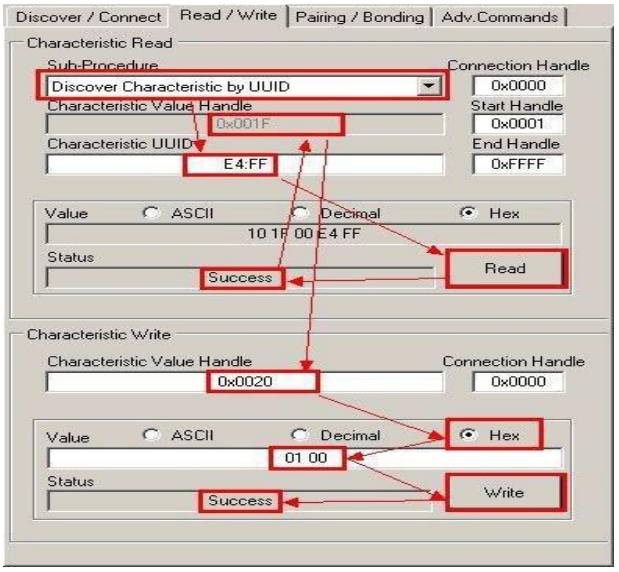


➢测试透传功能

将模块如系统示意图中的桥接模式，连接到串口终端或者单片机，便可以进行蓝牙 串口转发测试。注：**BRTS** 必须被置低，否则串口数据无法被模块 **RX** 接收。

1. 使用 BTool 使 BLE 模块与 USBDongle 建立连接后（连接过程参考上节说明），通过对 Handle: 0x0020 写入 01:00，来打开串口数据通道的自动通知开关，如下图所示。

如果主机将合法数据包发送到 BLE 模块的 RX 端，模块将会自动以通知的形式发到 BTool，左侧的显示栏会显示具体的数据。MCU 发给模块的串口数据可以是 200 字节以内的任意长度。

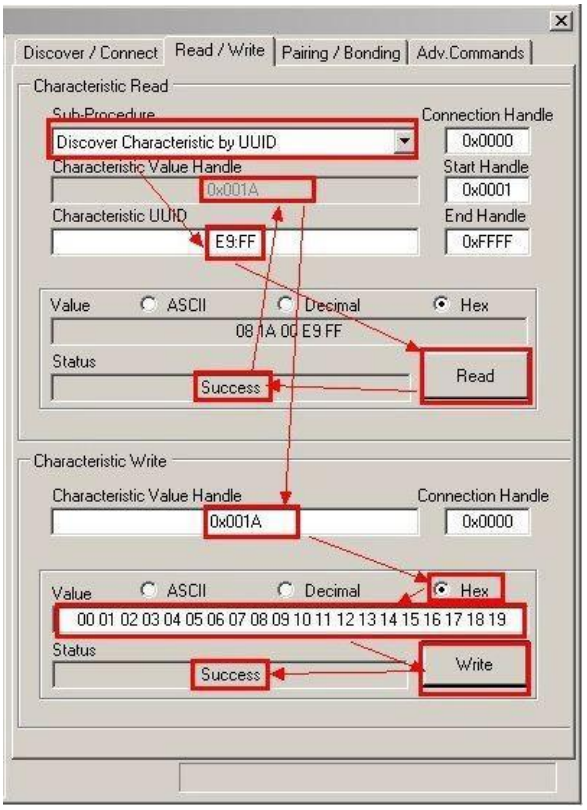


模块发送至移动设备使用串口数据通道，对应特征值(通道)的 UUID 如下：

名称	无线包数据长度	UUID	Handle	Notification EnableHandle
串口数据通道	20Bytes	0xFFE4	0x001F	0x0020

2. 通过 BTool 写 1-20 字节数据到模块。当模块收到来自手机的写操作，模块会通过串口发送到 MCU。用户可以通过读取 MCU 检验数据是否正确，也可以通过串口助手显示手机写入模块的数据。

   例如：写 7 个字节的数据到模块，是通过 Handle0x001A 写入，如下图所示。



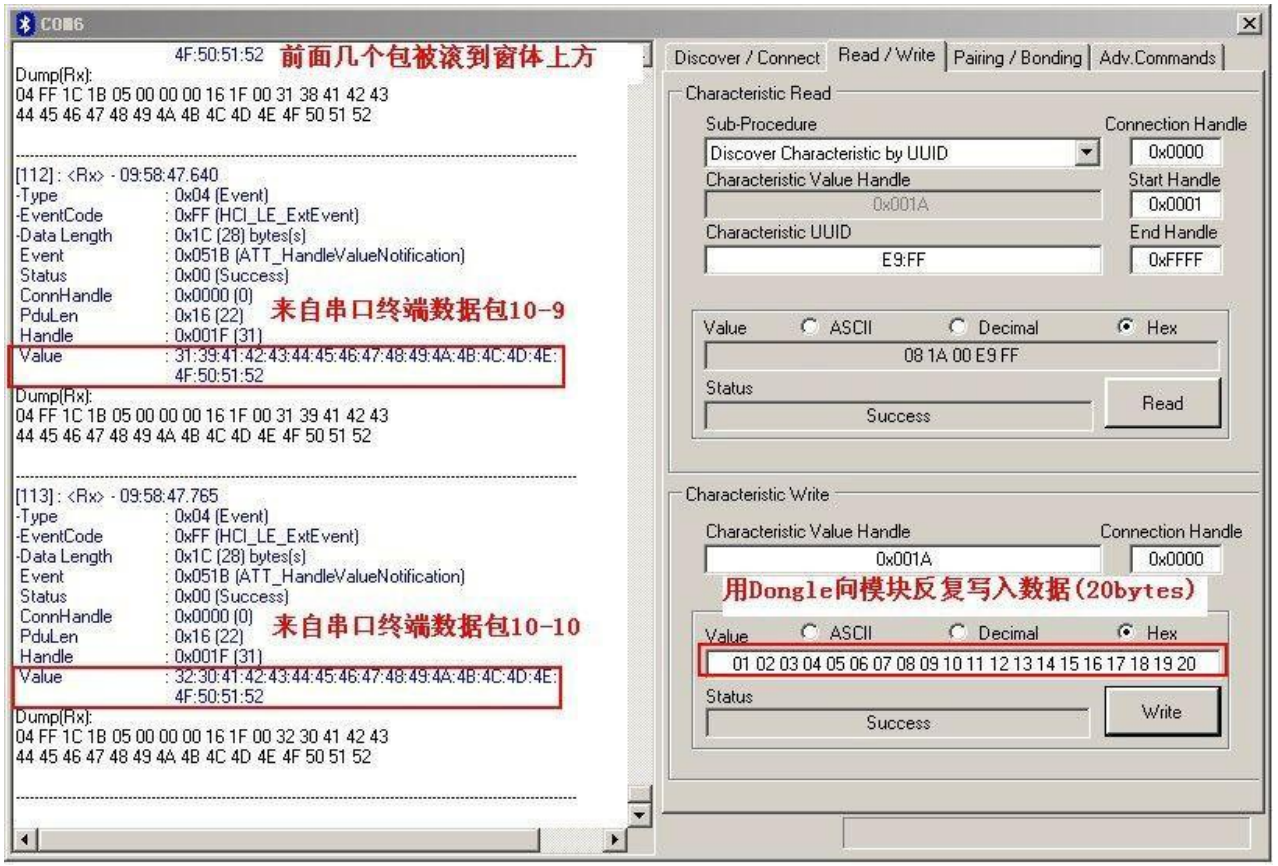
注：可写入 1-20 个字节到模块，但不能超过20 个字节，因此在手机端编程时，必须自行分包发送，每包长度不得超过 20 字节。移动设备发往模块通过蓝牙数据通道，对应特征值(通道)的 UUID 如下：

名称	无线包数据长度	UUID	Handle
蓝牙数据通道	20Bytes	0xFFE9	0x001A

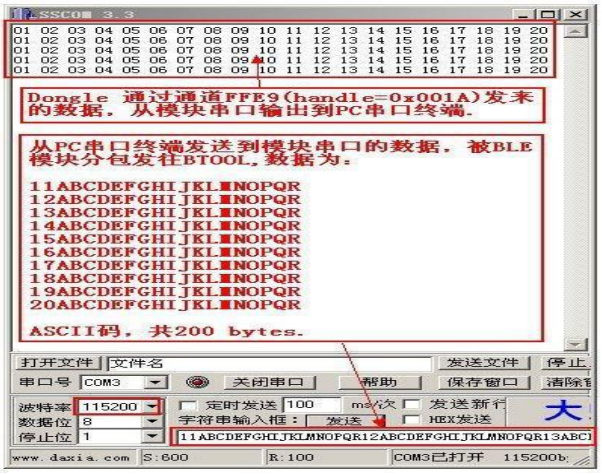
透传功能的测试，可以通过电平转换模块直连 PC 串口，通过串口终端来测试。

参考截图如下：

1，BTool 收发数据截屏。



2，PC 终端连接透传模块截屏，注 BRTS 必须被置低，否则串口数据无法被模块接收。



●主机参考代码（透传）

逻辑关系：模块间是用 BCTS,BRTS 两个 IO 口进行发送接收的通知和控制。

这两个 IO 常态高位，置低触发，如果模块有数据要发，置低 BCTS 通知单片机接收，如果单片机有数据要发，置低 BRTS 通知模块接收。示意性代码如下：

```
voidmain(void)
{
  EN=0;                                          //使能 EN，开始广播
  while(!BLEMoudleAck("TTM:OK\r\n\0"));          //等待手机端扫描，连接
                                                 //等待连接成功，也可加入限时等待
                                                 //也可判断连接提示信号线的电平
  BRTS=0;                                        //BRTS 置低通知 2540 模块准备接收
  halMcuWaitMs(2);                               //延迟 2ms
  UARTWrite(HAL_UART_PORT_0,"TTM:CIT-100ms",14);
                                                 //修改连接间隔，从串口得到确认：
  halMcuWaitMs(5);                               //延迟 5ms,确保数据已经发出
  BRTS=1;                                        //RTS 置高，发送完毕
  while(!BLEMoudleAck("TTM:OK\r\n\0"));          //等待设置成功，也可加入限时等待
  while(1){                                      //循环收发测试
   while(1){
    if(BCTS== 0){                                //检测，若 BCTS 置低则准备接收
      while(BCTS==0);                            //等待发送完毕，也可限时等待
      if(UARTRead(uartBuffer)==SUCCESS)          //串口读取数据
       {.....}                                   //使用数据
    }
    BRTS=0;                                      //RTS 置低通知 2540 模块准备接收
    halMcuWaitMs(2);                             //延迟 2ms
    send_TX("1234567890");                       //发送任意数据（200byte以内）
    halMcuWaitMs(5);                             //延迟 5ms,确保数据已经发出
    BRTS=1;                                      //RTS 置高，发送完毕
    halMcuWaitMs(20);                            //延迟再发下一个包，延时视包大小而定
    }
   }
}
}
```