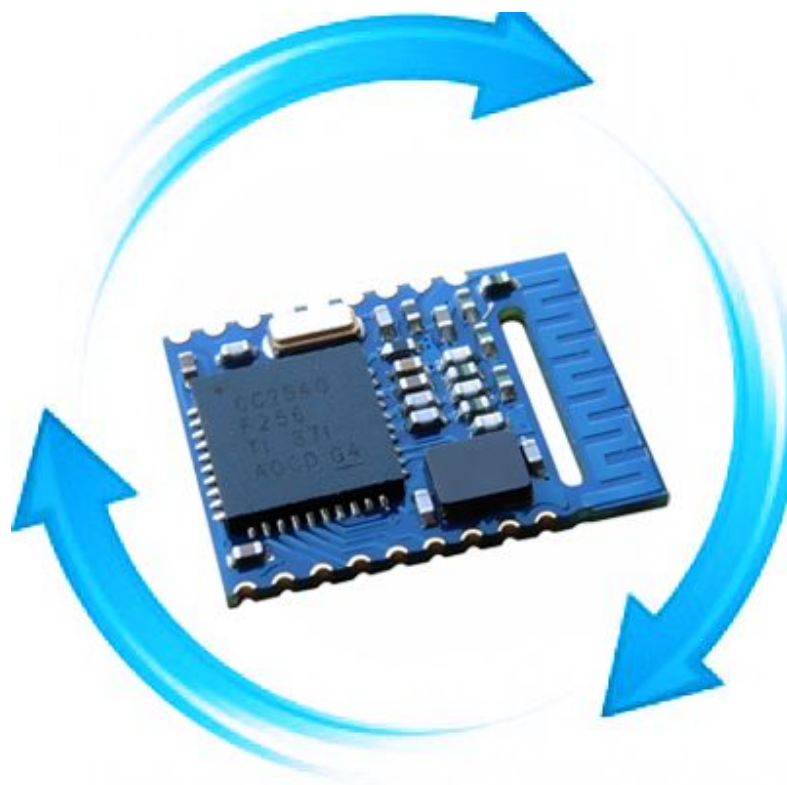


无线 UART 串口通信

## RF-BM-S02 蓝牙 4.0 ( BLE ) 模块

透明传输模式&直驱模式参考手册 ( Ver. 2.21u )



## 目录 Catalog

前言.....	3
概述.....	5
模块工作示意图.....	9
封装尺寸脚位定义.....	10
串口透传协议说明(桥接模式).....	13
串口 AT 指令.....	16
广播数据设置.....	22
系统复位与恢复.....	23
IOS APP 编程参考.....	25
BLE 协议说明(APP 接口).....	28
用 APP 测试透传功能.....	69
用 USB Dongle 及 Btool 测试.....	71
主机参考代码（透传）.....	82
Prefix.....	85
Overview.....	86
Schematic Diagram of Working Mode.....	90
Packaging size and Pins.....	91
UART transparent transparent transmission(Bridge mode).....	93
UART AT Command.....	96
Broadcast Data Setting.....	100
System Reset and Recovery.....	101
IOS APP programming reference.....	102
BLE protocol description (APP interface ).....	104
Transmission test by APP.....	130
Test by USB Dongle and Btool.....	132
Master reference code (transparent transmission).....	141
Contact Us.....	143
附录 A：模块原理图.....	144

# 前言

## 如何快速低成本地开发智能手机新外设

-论低功耗蓝牙技术在智能移动设备中的应用-

USB 协议的产生，让个人电脑的外设如雨后春笋般地涌现。同样，做为智能手机最新开放的低功耗蓝牙(BLE)无线应用技术，也有异曲同工之妙。BLE 技术给电子产品桥接智能手机提供了可能。相对 WIFI，Bluetooth 2.0 等无线技术，有着能耗低，连接迅速，通讯距离更远等优势，让智能手机的外围电子设备有了更开阔的发展前景。

做为国际蓝牙联盟(BT-SIG)成员之一，德州仪器(TI)于 2012 年强势推出 CC254x 系列单芯片(SOC)低功耗蓝牙收发器，经典 51 内核，最强优势在于丰富的外围(21 个 IO,UART,SPI,USB2.0,PWM,ADC,analog comparator,op-amp)，超宽的工作电压(2v-3.6v)，极低的能耗(<0.4uA)，极小的唤醒延时(4us)。

为方便低功耗蓝牙(BLE)应用技术在各个行业产品中的移植和使用，中国 TI 无线领域战略合作伙伴信驰达科技特别推出了低功耗蓝牙透传模块，已经有两款模块成功通过了蓝牙技术联盟 BQB (EPL)，FCC，CE，ROHS 认证：

### **RF-BM-S01 v1.1(全引脚):**

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

### **RF-BM-S02 (小尺寸精简版，非全引脚):**

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

其他模块认证中。

模块做为智能手机外设的桥梁，使得主机端应用开发异常简单。在桥接模式下(串口)，用户的现有产品或者方案配合此透传模块，能十分方便地和移动设备(需支持蓝牙 4.0)相互通讯，实现超强的智能化控制和管理。而在直驱模式下，用户直接使用模块扩展简单外围，就能快速设计出方案甚至产品，以最低成本最高效地推出特有的个性化移动设备新外设。

**RF-BM-S01** 低功耗蓝牙模块，采用 TI 的 CC2540 作为核心处理器。模块运行在 2.4 GHz ISM band，GFSK 调制方式（高斯频移键控），40 频道 2 MHz 的通道间隙，3 个固定的广播通道，37 个自适应自动跳频数据通道，物理层可以和经典蓝牙 RF 组合成双模设备，2 MHz 间隙能更好地防止相邻频道的干扰。宽输出功率调节(-23 dBm ~ 4dBm)，-93 dBm 高增益接收灵敏度。

此模块的设计目的是迅速桥接电子产品和智能移动设备，可广泛应用于有此需求的各种电子设备，如仪器仪表，物流跟踪，健康医疗，智能家居，运动计量，汽车电子，休闲玩具等。随着安卓 4.3 智能设备对 BLE 技术的集成，智能手机标配 BLE 必将成为时尚，手机外设的市场需求将成级数倍增。用户可借此模块，以最短的开发周期整合现有方案或产品，以最快的速度占领市场，同时为企业的发展注入崭新的技术力量。

## 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(14 bit),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 (  $61.036\text{ Hz} \leq f \leq 8\text{ kHz}$ ，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，0~0xFFFFFFFF ms (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）字符串提示；
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	0.35uA	0.3-0.4uA	-	EN 悬空
广播	202uA	0.14 ~ 0.54mA	3.85ms	广播周期 250ms
连接事件	243uA	0.41 mA	2.25ms	连接周期 100ms
单次 BLE 数据接收事件	332uA	0.65 mA	3.0ms	(20bytes,10 次/秒)
模块接收数据 并串口发送	497uA	2.68mA	5.1ms	(20bytes,10 次/秒)
单次 BLE 数据发送事件	342uA	0.69mA	3.2ms	(20bytes,10 次/秒)

\*1 注：官方测试方式：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。

以上数据为信驰达模块 **RF-CC2540A1** 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。



## 模块工作示意图

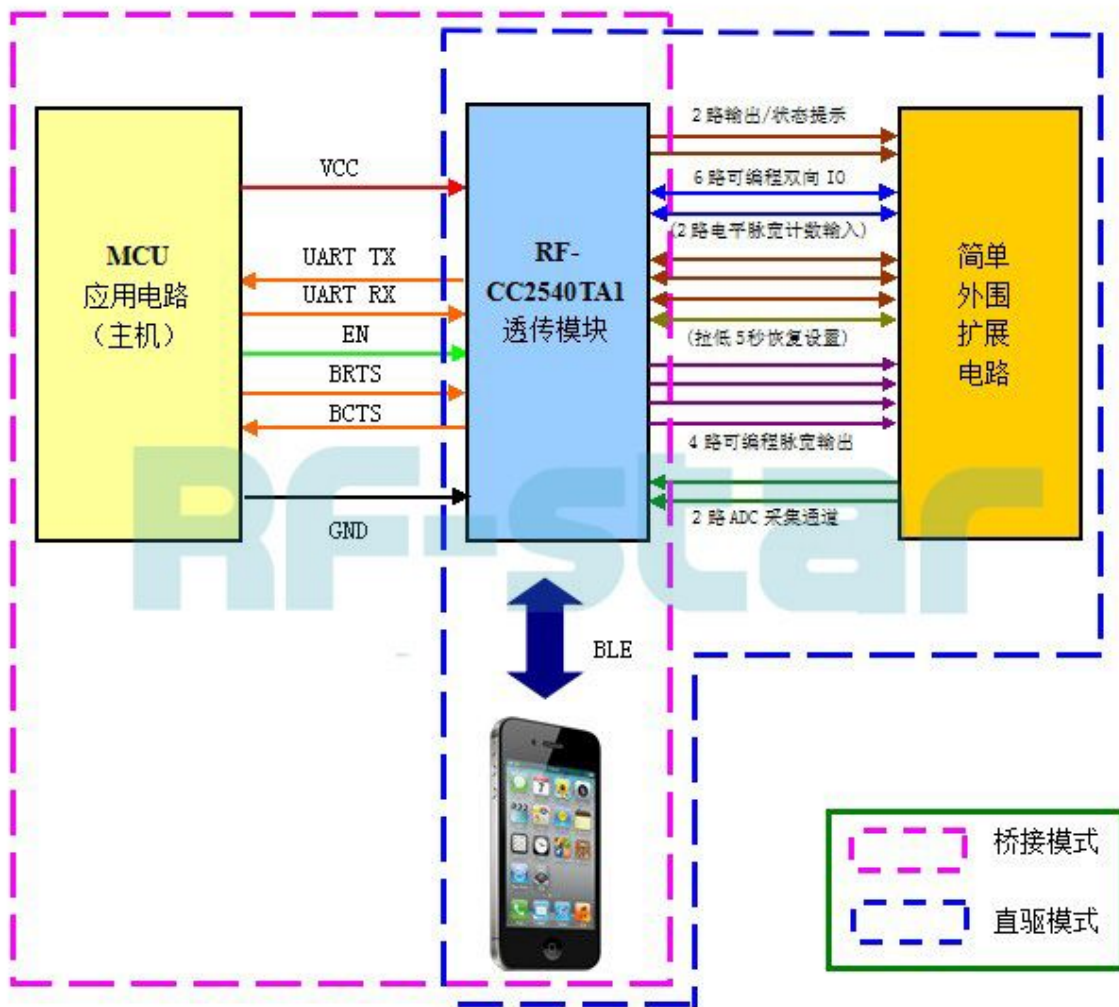
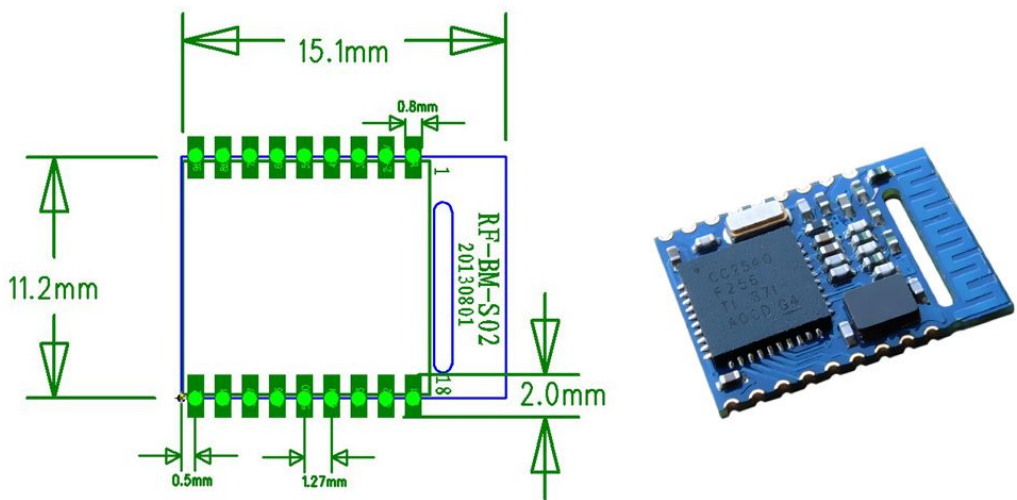


图 1：模块 (V2.20)桥接模式和直驱模式示意图

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

RF-BM-S02 版 ( BQB 认证 , 四层板工艺 )



BM-S02 版 ( BQB 认证 )

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。

				<p>0：模块开始广播，直到连接到移动设备</p> <p>1：无论模块当前状态，立即进入完全睡眠状态 (0.4uA)</p> <p>➤ 脉冲触发模式，每收到一次脉冲 (W&gt;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</p> <p>(关于模式的切换请参考《模块参数设置》相关章节)</p>
Pin7	IO5	P1.7	I/O	<p>➤ 可编程双向 IO，可通过 BLE 协议设置成输入或输出使用</p> <p>➤ 当做为输入时，可做为电平脉宽计数输入端</p>
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	<p>恢复出厂设置触发或可编程双向 IO</p> <p>➤ <b>上电后 30 秒内</b>，保持此引脚低电平 <b>5s</b>，系统会恢复部分参数（浅恢复），若保持 <b>20s</b> 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）</p> <p>➤ 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</p>
Pin11	PWM 1	P1.1	O	PWM 输出通道 1
Pin12	PWM 3	P0.7	O	PWM 输出通道 3
Pin13	PWM	P0.6	O	PWM 输出通道 4

	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：BM-S02 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 20 ms ,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 **100us** 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 **500us** 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\n0**" 字串，可以根据此字串来确定是否可以正常转发操作。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\n0**" 字串提示，如果是非正常断开，会从 TX 给出 **TTM:DISCONNET FOR TIMEOUT\r\n0** 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：

$$V = 80 \times 1000 / T \quad (V \text{ 只和 } T \text{ 有关})$$

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 = 4K byte/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”开头的字符串会当成 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 输入以下字符串，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 iPhone 4s ( IOS 5.1.1 ) 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 输入以下字符串，- 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 IOS5 中无法立即变更。此名称掉电保存。

#### ➤ 波特率设定

向串口 RX 输入以下字符串，- 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 BLE APP 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 输入以下字符串：

"TTM:MAC-?"

会从 TX 收到：

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

字符串后面"xxxxxxxxxxx"为 6 字节模块蓝牙地址。

#### ➤ 模块复位

向串口 RX 输入以下字符串：

"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  $0 < L \leq 16$ 。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

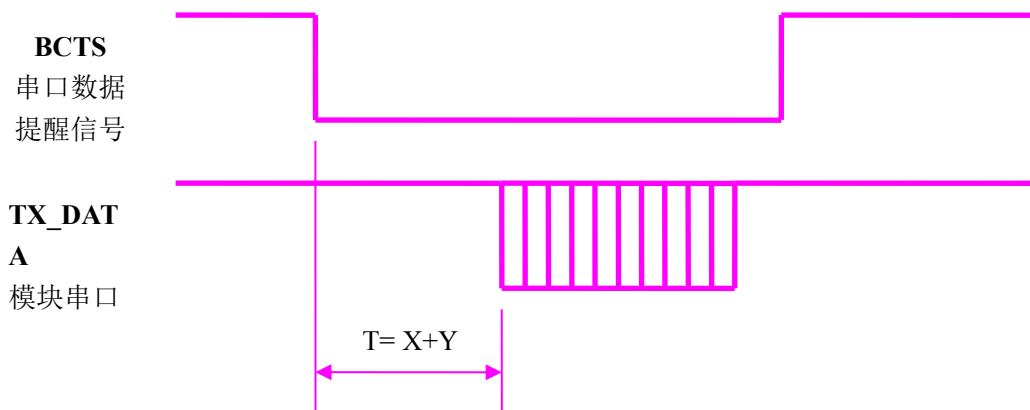
向串口 RX 输入以下字串，设置 BCTS 输出低到串口 TX 输出数据之间的延时，单位 ms

"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  $T = (X + Y) \text{ ms}$ ，其中  $500\mu\text{s} < Y < 1\text{ms}$ 。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\n0" "TTM:OK\r\n0" "TTM:ERP\r\n0"	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 字节以内的任意字符串。	"TTM:OK\r\n0" "TTM:ERP\r\n0"	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:BPS SET AFTER 2S ...\r\n0" "TTM:ERP\r\n0"	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-xxxxxxx xxxx"    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 00 00；	"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-Xms"	是	X="0","2","5","10","15", "20","25" 设置 BCTS 输出低到串口	"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.

		输出数据之间的延时，单位 ms		
--	--	-----------------	--	--

*\* 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 (16 byte)，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) - 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。



## IOS APP 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：

```
/*!  
 * @method writeValue:forCharacteristic:withResponse:  
 * @param data The value to write.  
 * @param characteristic The characteristic on which to perform the write operation.  
 * @param type The type of write to be executed.  
 * @discussion Write the value of a characteristic.  
 * The passed data is copied and can be disposed of after the call finishes.  
 * The relevant delegate callback will then be invoked with the status of the request.  
 * @see peripheral:didWriteValueForCharacteristic:error:  
 */  
  
- (void)writeValue:(NSData *)data forCharacteristic:(CBCharacteristic *)characteristic type:(C  
BCharacteristicWriteType)type;
```

**说明：对某个特征值进行写操作。**

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

*[p writeValue:d*

*forCharacteristic:c*

*type:CBCharacteristicWriteWithoutResponse];*

*/\*!*

- \* @method readValueForCharacteristic:
- \* @param characteristic The characteristic for which the value needs to be read.
- \* @discussion Fetch the value of a characteristic.
- \* The relevant delegate callback will then be invoked with the status of the request.
- \* @see peripheral:didUpdateValueForCharacteristic:error:

*\*/*

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

**说明：读取某个特征值。**

*[p readValueForCharacteristic:c];*

*/\*!*

- \* @method setNotifyValue:forCharacteristic:
- \* @param notifyValue The value to set the client configuration descriptor to.
- \* @param characteristic The characteristic containing the client configuration.
- \* @discussion Ask to start/stop receiving notifications for a characteristic.
- \* The relevant delegate callback will then be invoked with the status of the request.
- \* @see peripheral:didUpdateNotificationStateForCharacteristic:error:

\*/

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

**说明：打开特征值通知使能开关。**

***[self setNotifyValue:YES forCharacteristic:c]; //打开通知使能开关***

***[self setNotifyValue:NO forCharacteristic:c]; //关闭通知使能开关***

/\*

\* @method didUpdateValueForCharacteristic

\* @param peripheral Peripheral that got updated

\* @param characteristic Characteristic that got updated

\* @error error Error message if something went wrong

\* @discussion didUpdateValueForCharacteristic is called when CoreBluetooth has updated

a

\* characteristic for a peripheral. All reads and notifications come here to be processed.

\*

\*/

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

**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。**

有关设备的扫描，连接以及其他通讯细节，可以参考信驰达科技提供的基于 IOS 的透传模块测试 APP 源码( ble Transmit Moudel v1.29)。里面实现了对 FFE9 和 FFE4 转发蓝牙数据到串口，转发串口数

据到蓝牙两个通道(特征值)的操作 ( 通知和写操作 ), 其他直驱功能控制方法类似 , 都是通过对某个通道(特征值)的读写来实现。只是通道 UUID 以及读写字节数不同。( 相关源码请向业务索取 )

## 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 BLE API 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（4 bytes）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（2 bytes）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  $500 \leq w \leq 65535$ ，一个单位对应  $0.00000025s$ ，对应方波周期： $0.000125s \leq T \leq 0.01638375s$ ，因此方波信号频率的可调范围： $61.036Hz \leq f \leq 8kHz$ ，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  $0x8235 * 0.00000025s = 0.00833325s$ ，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  $0x0FA0 * 0.00000025s = 0.001s$ ，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（2 bytes）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： $0 \leq t \leq 65535$ ，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 BLE API 接口向 FFD1 通道写操作，来使能两个 13bit ADC 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 : 0xFF0】

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	<p>IO7~ IO0 的配置字。</p> <p>当相应位被设置为 0 时 :</p> <p>bit7,bit6 表示 IO7,IO6 做为信号提示脚位 , 低电平有效</p> <p>bit5~ bit0 表示 IO5~ IO0 做为输入口</p> <p>当相应位被设置为 1 时 :</p> <p>bit7,bit6 表示 IO7,IO6 做为普通输出口</p> <p>bit5~ bit0 表示 IO5~ IO0 做为输出口</p>
FFF2  (handle:  0x001A)	write	1	--	<p>IO7~ IO0 的输出状态。</p> <p>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</p>

				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 ~ IO0 8 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 ~ IO0 8 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 01 00），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)。

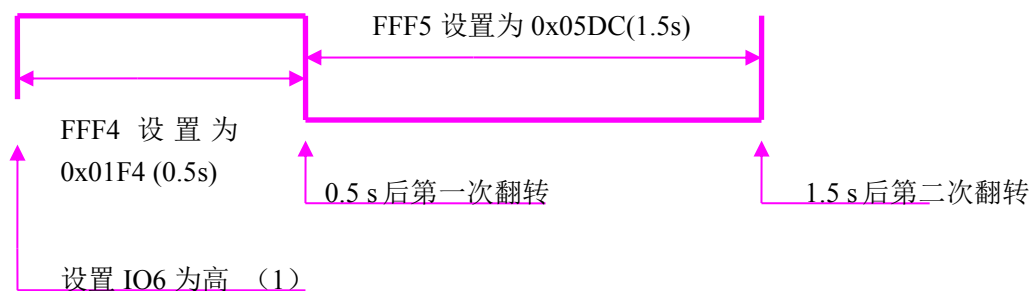
FFF4 通道设定 IO6 第一次翻转的延时时间，FFF5 通道设定 IO6 第二次翻转的延时时间。如果 FFF4 设置为 0，则不启动 IO6 的翻转。如果 FFF4 设置为非 0，而 FFF5 通道设置为 0，则仅启动一次 IO6 的翻转。必须先设置 FFF5 通道，此时翻转未被启动，再设置 FFF4 通道为非 0 值来启动 IO6 的定时翻转。同样，**可以通过对 FFF4 通道写入 0 来关闭 IO6 的定时翻转，此时以前写入 FFF5 通道的任意值将被清零。**单位为 ms，范围为 0 ~ 0xFFFFFFFF ms (4294967295ms,约 1193 小时,约 49.7 天)，换算成十六进制为：

0.5s	1s	1.5s	2s	3s	4s	5s
500ms	1000ms	1500ms	2000ms	3000ms	4000ms	5000ms
<b>0x01F4</b>	0x03E8	<b>0x05DC</b>	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 0bx**1**xxxxxx;
2. 设置 IO6 当前为高(1)，通过向 FFF2 写入 0bx**1**xxxxxx;
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：0xFFFF0】

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 IO 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。

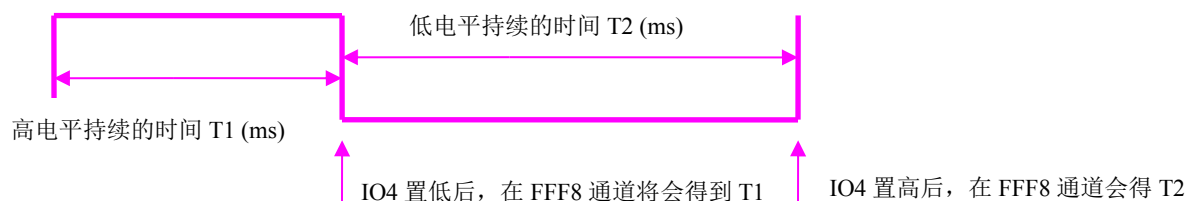


此功能仅在 FFF1 的高两位 BIT5,BIT4 被设为 0 时有效(做为输入口)。

FFF8 通道为 IO4 ( P1.6 ) 电平脉宽计数通知通道 , APP 通过 BLE API 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  $0 \sim 0xFFFFFFFF$  ms (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 BLE API 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  $0 \sim 0xFFFFFFFF$  ms (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  <b>(掉电 保存)</b>	1	"123456123456" (ASCII)	提交当前密码  <b>123456</b> ，新密码和旧
		2	"123456888888" (ASCII)	密码必须一致  把旧密码 <b>123456</b> 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
FFC2  (handle:  0x0048)	notify	1	0 ( PWD_RIGHT_EVENT )	提交密码正确
			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 BTtool 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 BLE API 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 **01 00**），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 BLE API 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 的读值为 signed char 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	可执行的操作	是	字	默认值	备注
-----	--------	---	---	-----	----

UUID		否 保 存	节 数		
FF91  (handle: 0x0062)	Read/write	是	16	TA <sub>v</sub> 22u-xxxxxxx  (带结束符的 ASCII 字符串)	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 : <ul style="list-style-type: none"> <li>➤ 远程复位控制, 写入 <b>0x55</b> 对模块进行复位</li> <li>➤ 远程浅恢复控制, 写入 <b>0x35</b> 对模块进行浅恢复 ( 仅仅恢复用户数据 ), 并复位</li> <li>➤ 远程深度恢复控制, 写入 <b>0x36</b> 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</li> </ul>
FF95 (handle: 0x006E)	Read/write	是	1	0	设定广播周期 : 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	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 : +4 dBm 1 : 0 dBm 2 : -6 dBm 3 : -23 dBm
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, $0 < n \leq 16$
FF99	write	-	1	无	远程控制扩展通道 :



(handle: 0x007A)					<p>➤ <b>0x01</b> : 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</p> <p>➤ <b>0x02</b> : 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</p>
FF9A (handle: 0x007D)	Read/write	是	1	0b <b>00000000</b>	<p>系统功能使能开关：</p> <p><b>BIT0</b>：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 <math>W &gt; 200ms</math>。当广播时间超过 30s，仍未被连接，则会自动进入关机状态。</p>

					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-xxxxxxx\0” (16 byte)，vvvv 为固件版本号，xxxxxxx 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。

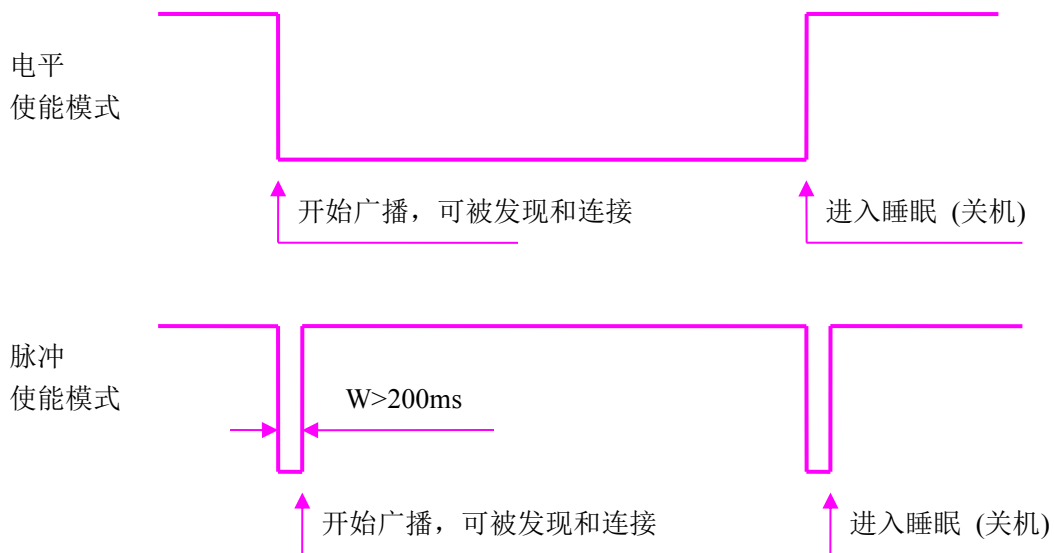
对此通道写入 **0x01**，来触发模块保存当前除 IO0 以外的 IO 配置以及输出状态，在重新上电后模块总是使用保存过的 IO 配置以及输出状态初始化 IO7 ~ IO1，IO0 却总是上电后默认为输入状态，用来做恢复出厂设置的触发 IO，上电之后，IO0 和其他 IO 一样可以被配置成输出 IO 使用。

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（ $W > 200\text{ ms}$ ），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。



UUID:0xFE03						
EVT	EVT	EVT	EVT5...	EVT2	EVT3	EVT3
0	1	2	EVT28	9	0	1
定时 时间	定时 时间	定时 时间	.....	定时 时间	定时 时间	定时 时间
动作 类型	动作 类型	动作 类型		动作 类型	动作 类型	动作 类型
操作 参数	操作 参数	操作 参数		操作 参数	操作 参数	操作 参数

UUID:0xFE05									UUID:0xFE0	
									6	
UUID:0xFE05	POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO0	EA
	T0									
	POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO1	
	T1									
	POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO2	
T2										
POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO3		
T3										
POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO4		

	T4								
	POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	IO5
	T5								
	POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	PWM0
	T6								
	POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	PWM1
	T7								
	POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	PWM2
	T8								
	POR	bit0	bit1	bit2	.....	bit29	bit30	bit31	PWM3
	T9								
									CEVT
									CPOR
									T

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：

- 定时事件 EVT 的定时时间溢出触发；
- 响应端口开启了指向定时事件 EVT 的响应开关位 BIT；
- 指向响应端口的定时事件单独使能位 IO/PWM 使能；
- 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	0x07D00101 000000	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:	R/ W	1	BYTE0	0x00	事件索引号	读事件指针。

0x0089)						<p>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。</p> <p>取值范围: 0~31, 分别表示 32 个定时事件。</p>
FE03 (handle: 0x008C)	R/ W	12	BYTE0	0x00	事件索引号	<p>事件读写通道。</p> <p>对此通道进行读写,可以获得和设置定时事件。</p>

			BYTE7 ~ BYTE1	0x000000 00000000	秒分时日 月 年(L)年(H)	·事件索引号：  取值范围：  0~31：分别表示 32 个定 时事件；
			BYTE8	0x00	动作类型	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。
			BYTE9	0x00	PWM 新占空 值	如以下定时时间 ( Hex )：  00 FF 01 01 01 D0 07
			BYTE10	0x00	PWM 渐变时 长的低字节	此定时事件将在任意分钟 的 0 秒时刻触发；
			BYTE11	0x00	PWM 渐变时长的高 字节	·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；

						<p>4 : PWM 突变 ;</p> <p>5 : PWM 渐变 ;</p> <p>·PWM 新占空值 :</p> <p>动作类型为 4 或 5 时有效 ,</p> <p>变化后的目标占空值 ,取值</p> <p>范围 : 0~255 ;</p> <p>0 时占空比为 0% , 即全低</p> <p>电平 , 255 时占空比为</p> <p>100% , 即全高电平 ;</p> <p>·PWM 渐变时长 :</p> <p>操作值为 5 时有效 ,从当前</p> <p>占空值变化到新的占空值 ,</p> <p>所开销的时间 ,此值越大变</p> <p>化得越慢 ,此值越小变化得</p> <p>越 快 。 取 值 范 围 :</p> <p>0~65535 单位为 100ms ;</p>
FE04 (handle: 0x008F)	R/ W	1	BYTE0	0x00	端口索引号	<p>端口事件读指针。</p> <p>端口索引 :</p>

						<p>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。</p> <p>取值范围：</p> <p>0~9：分别表示 IO0~IO5 和 PWM0~PWM3 的定时端口。</p>
FE05 (handle: 0x0092)	R/ W	5	BYTE0	0x00	端口索引号	<p>端口事件读写通道。</p> <p>·端口索引号</p> <p>取值范围：</p> <p>0~9：分别表示 IO0~IO5 和 PWM0~PWM3 的定时端口。</p>
			BYTE4 ~ BYTE1	0b000000 00000000 00000000 00000000 00	定时事件使能 0~31	<p>·适用事件使能开关，不同位分别对应 0~31 个定时事件：</p>



						取值范围：  1：开启；  0：关闭；
FE06  (handle: 0x0095)	R/ W	2	BYTE0	0b000000  00	Bit0:定时总使能	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 );
					Bit1:IO0 使能	
					Bit2:IO1 使能	
					Bit3:IO2 使能	
					Bit4:IO3 使能	
					Bit5:IO4 使能	
					Bit6:IO5 使能	
					Bit7:PWM0 使能	
			BYTE1	0b000000  00	Bit0:PWM1 使能	·BYTE0:BIT1 ~ BIT7,  BYTE1:BIT0 ~ BIT2,为端口定时事件单独使能；
					Bit1:PWM2 使能	
					Bit2:PWM3 使能	
					Bit3:定时事件清空控制	

					<div>Bit4:定时任务清空控制</div> <div>制</div> <div>-</div> <div>-</div> <div>-</div>	<div>·BYTE1:BIT3,为定时事件清空控制位 ,清除所有已设置定时事件 ; (CEVT)</div> <div>·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)</div>
--	--	--	--	--	------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------

**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。



BLE 主设备对从模块的操作步骤如下：

- ✓ 向 IO 配置字 ( UUID : 0xFF1 ) 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 05 04 04 02 01 DD 07 03 00 00 00**。

- ✓ 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 01 00 00 00**。

- ✓ 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 :

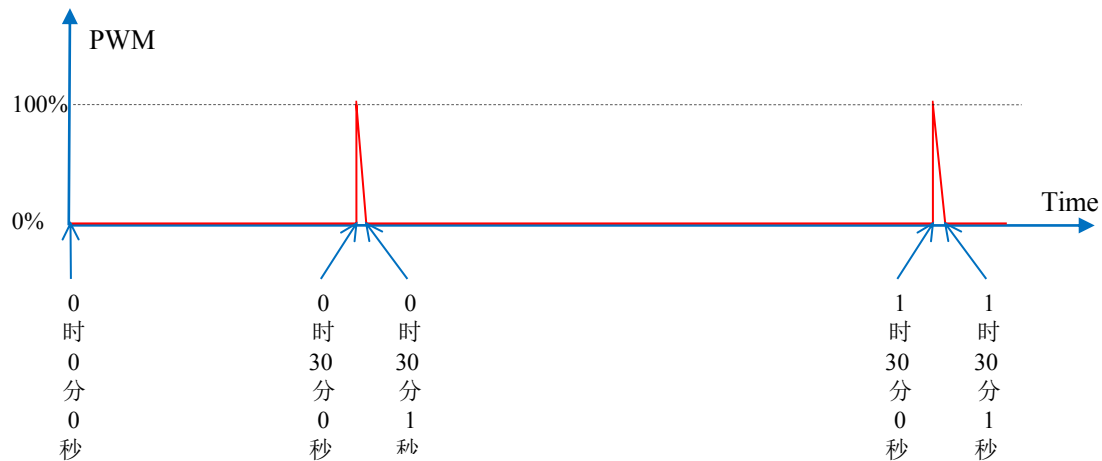
Hex ( 低字节在前 ) : **09 00**。

- ✓ 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 1E FF FF FF FF FF 04 FF 00 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 1E FF FF FF FF FF 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

模块 IOS 平台的测试工具(APP)可以在 AppStore 下载到。打开 iPhone 4S，iPhone 5 或者 iPad4 中的 AppStore，搜索 BLE Transmit\_Moudel (如需源码可向业务索取)，下载安装，进行测试。你有三种方法安装此应用：

1. 从 APP STORE 上搜索下载安装，需要 app 苹果账户，免费申请
2. 使用源码编译下载到你的苹果设备，需要苹果开发者账户
3. 越狱你的苹果设备，到信驰达官网下载 IPA 文件(相当于 windows 的 exe 文件)，使用快用助手，PP 助手，等工具安装

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控制主界面。



接着会自动扫描服务数据，之后提示准备完成。



此时如果模块串口已经就绪（连接了主 CPU，或者串口终端），即可以开始工作可进行手动和自动收发测试。IP:是 iPhone 发出的数据包，PC:是主机 CPU 或者串口终端发出的数据包。



注：如果使用串口终端进行测试，串口终端的数据要发到手机，**必须保持 BRTS 置低**，防止模块进入睡眠。

关于 IOS 编程，根据低功耗蓝牙协议，移动设备发送数据可以通过 **B 通道**(发送)的对应服务（UUID）进行写操作。模块数据到移动设备的数据传送，是通过通知的形式进行，因此在 APP 启动后需要打开 **A 通道**(接收)对应服务(UUID)的通知(Notification)使能，之后模块串口收到的数据包会自动发送到移动设备。具体操作可以参考信驰达科技提供的基于 IOS 的透传模块测试 APP 源码( BLE

Transmit\_Moudel V1.29)。此软件不定期更新，敬请关注信驰达官方网站 (<http://www.szrfstar.com/>) 动态。

## 用 USB Dongle 及 Btool 测试

BLE 模块可使用 TI 官方 CC2540 MinDK 开发套件中的 USB Dongle 模拟手机配合安装目录下的 C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\Btool\Btool.exe 进行蓝牙通讯测试。

这个 USB Dongle 需要使用安装目录下

C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\ble\HostTestApp\CC2540 的工程项目。

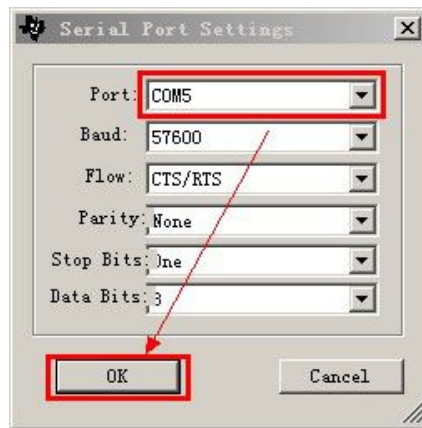
编译下载到 USB dongle 中。具体的 BTOOL 的使用详情请参考官方说明文档

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

### ➤ 连接 BLE 模块

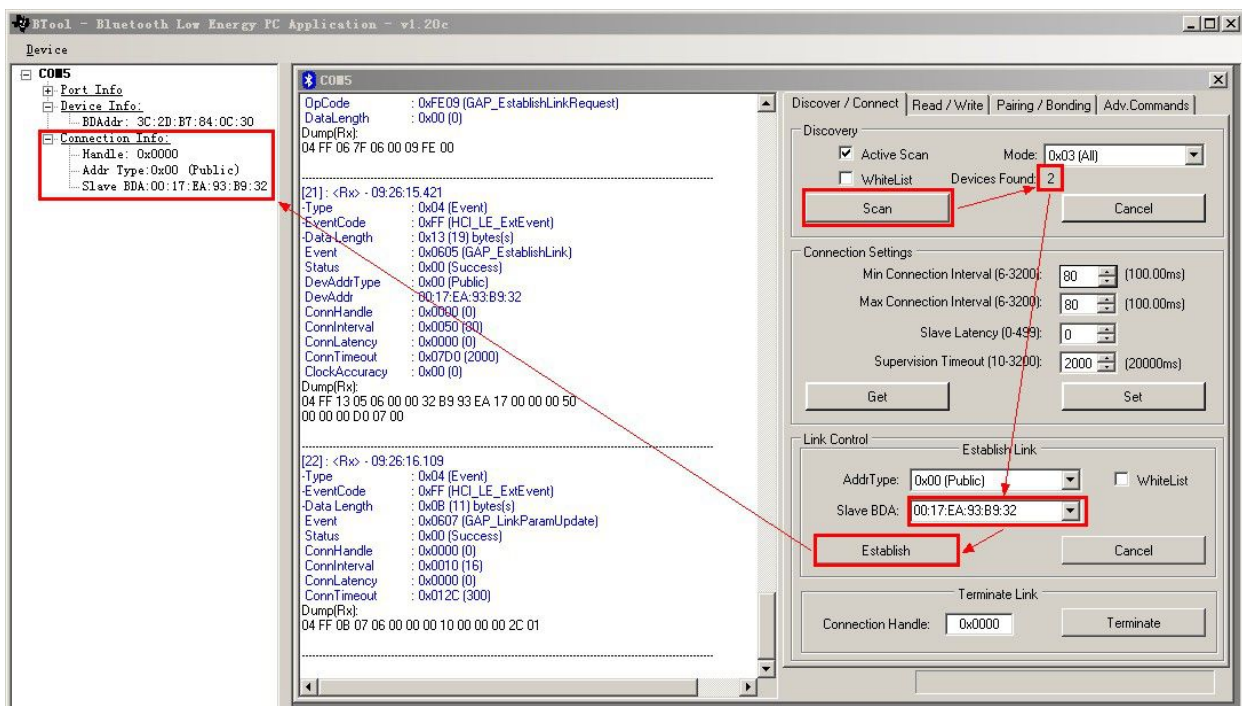
USB Dongle 和模块的连接是通讯的基础，扫描连接的操作步骤如下：

1. 打开 C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\ble\HostTestApp 目录下的工程文件，编译，下载到 USB Dongle 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 USB Dongle 插入 PC USB 口，会在硬件管理中出现一个串口设备（如：COM5）；
5. 打开 C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\Btool\Btool.exe；
6. 菜单 Device -> New Device，选择 4 中发现的串口，选默认设置，OK；



7. 扫描连接,按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 Connection Info。



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。

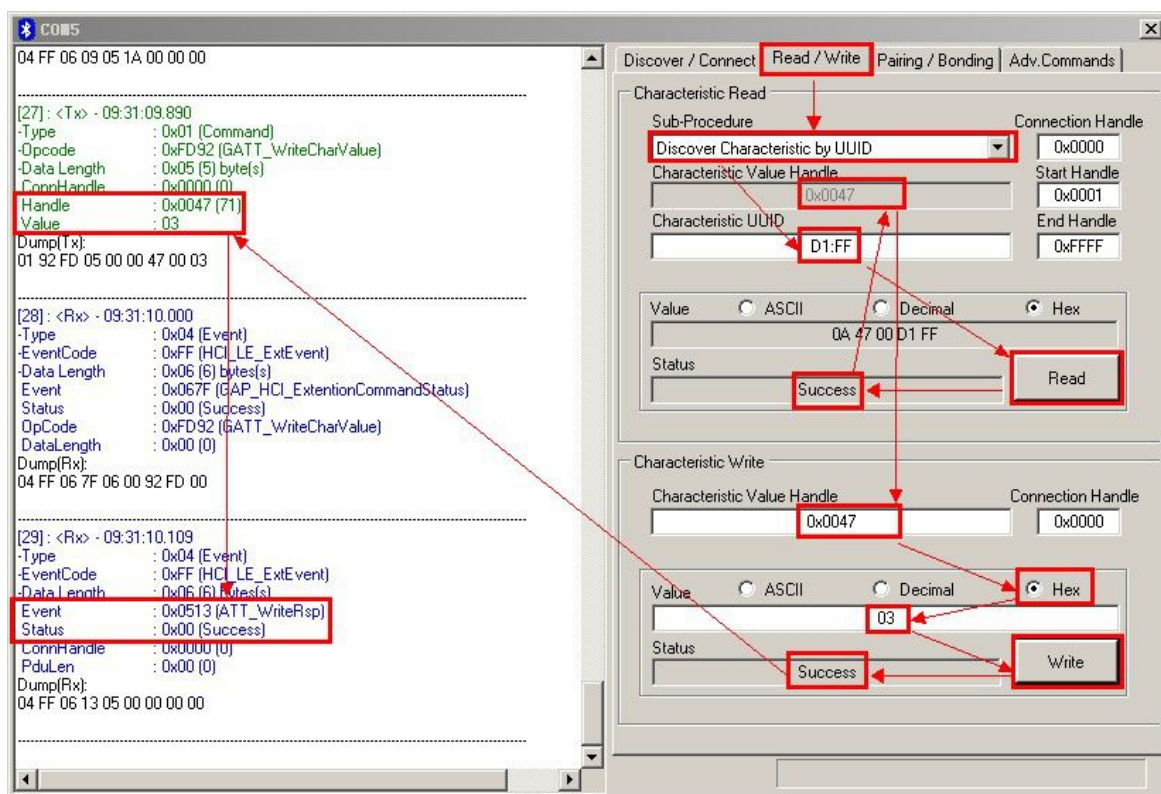
现以 ADC 为例，详述如何使用 USB Dongle 以及 Btool 软件工具来直接驱动蓝牙模块。**下例以信驰达 BLE 透传模块 V2.0 为例 ,对应 Handle 和其他版本可能略有不同 ,但操作流程完全一致。其他功能测试方法类似，只是对应的通道 UUID 不同，读写方法相同。**

#### ADC 输入(2 路) 【服务 UUID：0xFFD0】

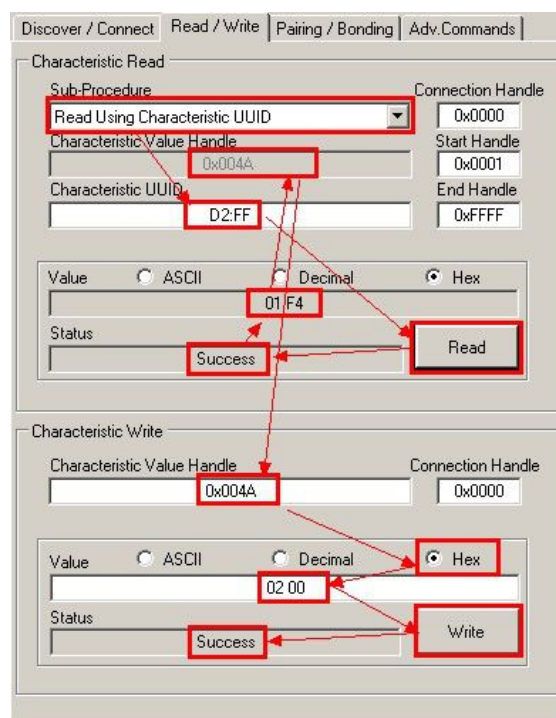
特征值 UUID	可执行的操作	字节数	默认值	备注
FFD1	Read/write	1	0x00	使能控制。

(handle: 0x0047)				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

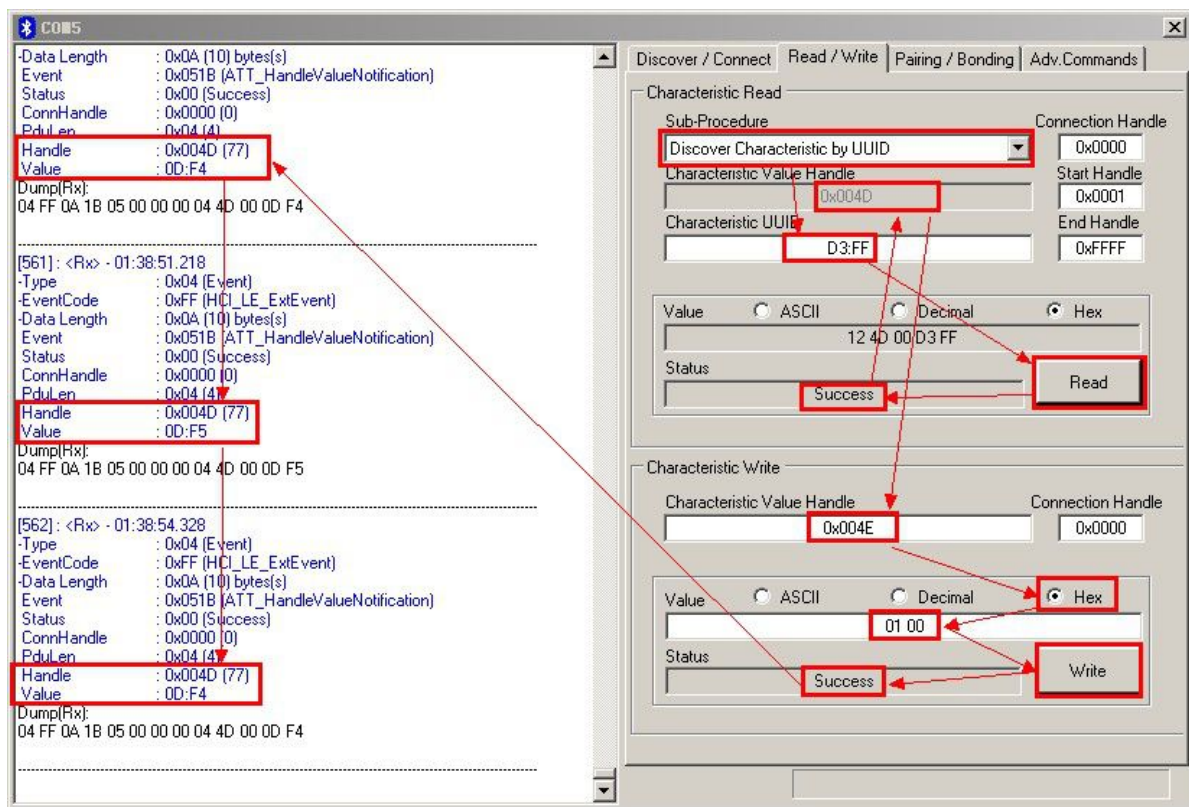
1. 打开 ADC0 和 ADC1。如图所示，后续操作都可以参考红色箭头的方向进行操作。首先寻找需要控制的通道(UUID)，输入需要寻找的 UUID，低位在前 D1:FF，读取成功后，得到 handle = 0x0047,向 0x0047 写入 03，这样就打开了 ADC0 和 ADC1,见上表。



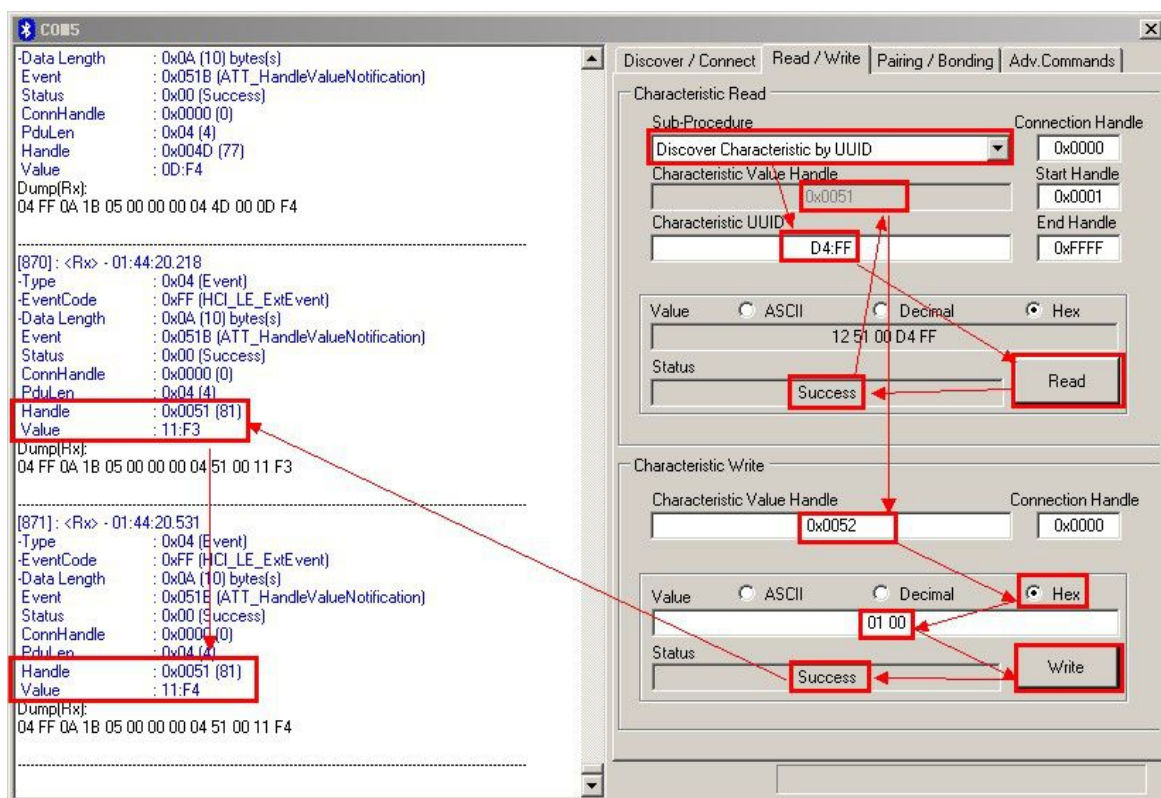
2. 读取和重新设定采集周期，先读取 FFD2 通道，可以得到 01 F4，表示默认为 500ms，通过得到的 handle = 0x004A 写入 02 00 高位在前，( 0x0200 = 512 ms )，设置每 512ms 采集一次。



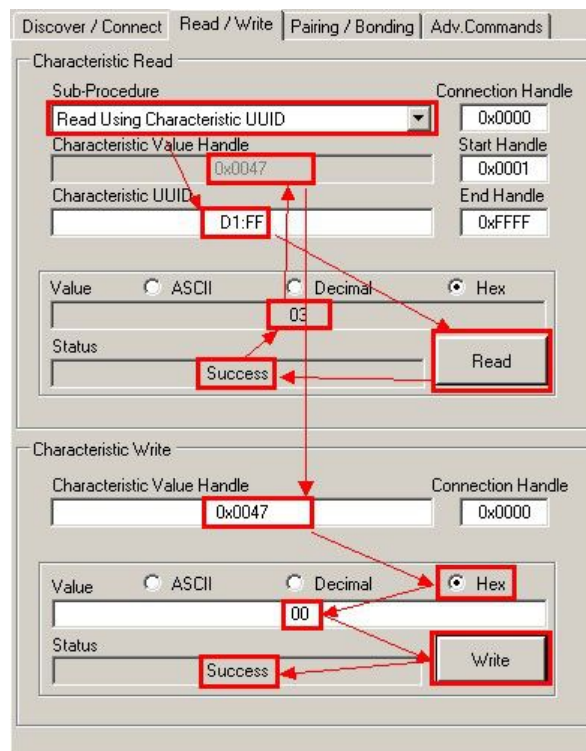
3. 打开 ADC0 通道通知使能开关，通知开关的地址为 **handle+1**， $0x004D+1=0x004E$ ，此后，ADC0 每次采集值不同于上一次，则会产生一次新的通知



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。



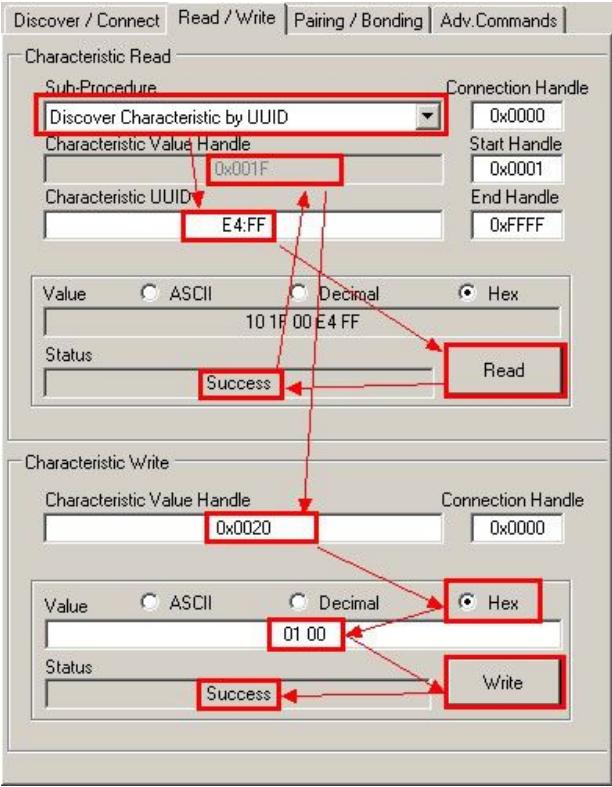


➤ 测试透传功能

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 USB Dongle 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。



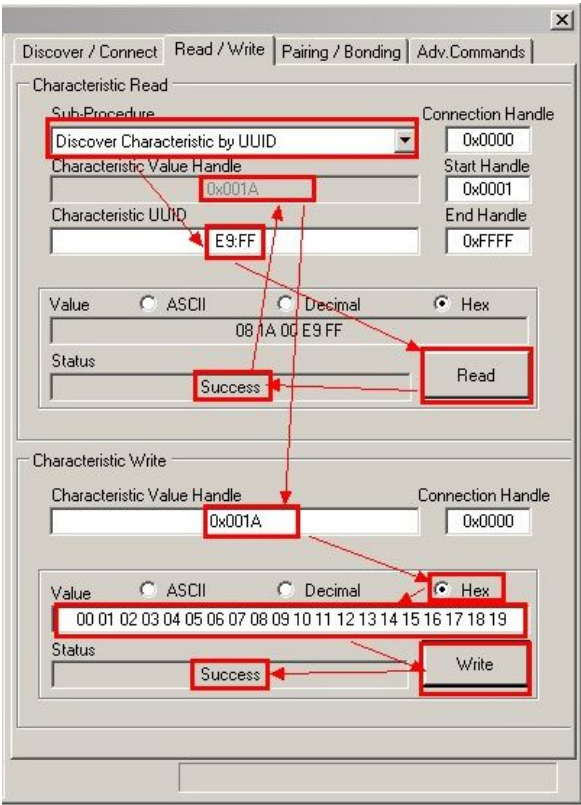
模块发送至移动设备使用**串口数据通道**，对应特征值(通道)的 UUID 如下：

名称	无线包数据长度	UUID	Handle	Notification Enable Handle

串口数据 通道	20 Bytes	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 Handle 0x001A 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
----	---------	------	--------

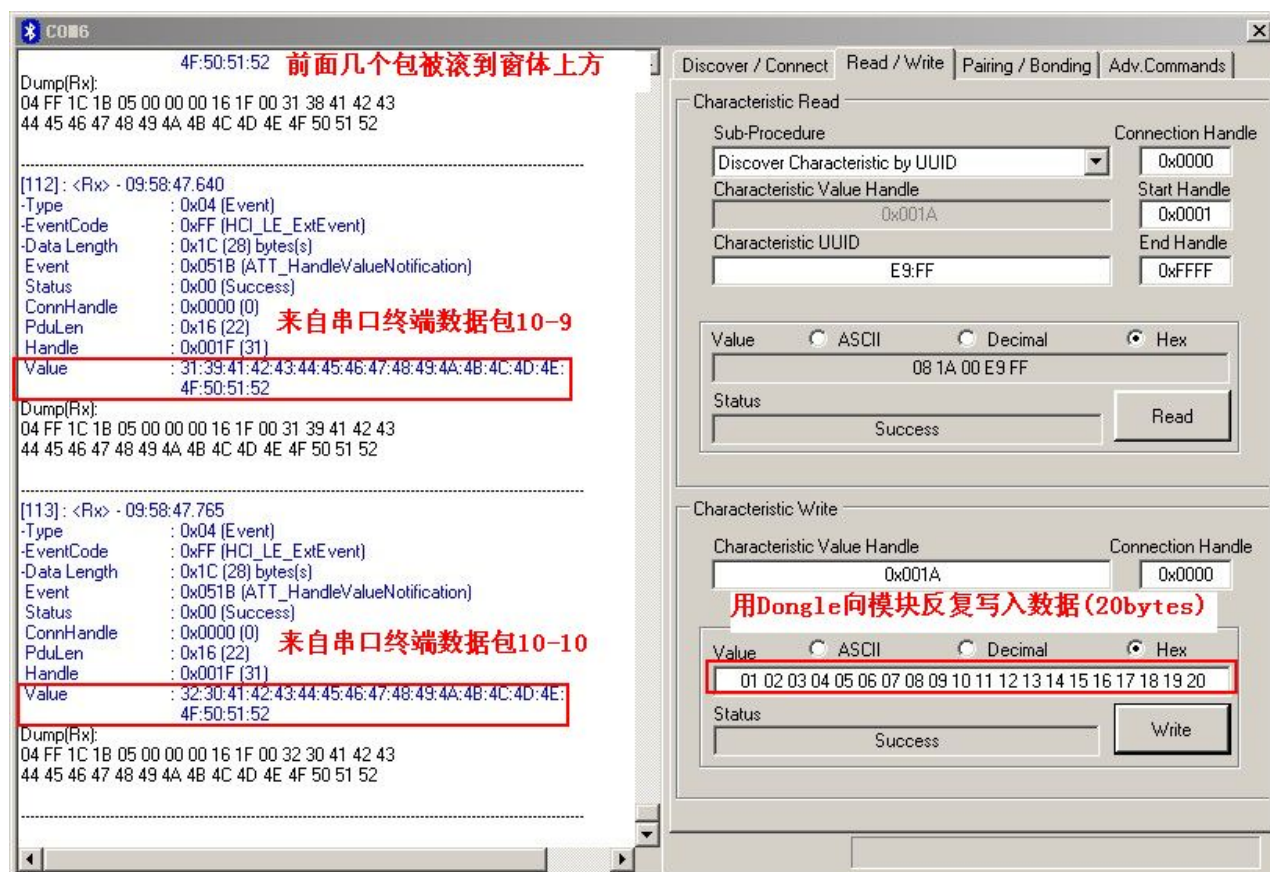


蓝牙数据通道	20 Bytes	0xFFE9	0x001A
--------	----------	--------	--------

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：

```
void main(void)
```

```
{
```

```
    EN = 0; //使能 EN，开始广播
```

```
    while(!BLEMoudleAck("TTM:OK\r\n0")); //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

}

}

}

}
```

## Prefix

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

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



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

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

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

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

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

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

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

(Other modules are under certification process).

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

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

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

## Overview

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

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

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

The detailed meaning of the user data is defined by the up-application. Mobile devices can write to the module through the APP. And the data written will be sent to the user's MCU through UART. Then the module will transmit the data package from user MCU to the mobile devices automatically. In the development under this mode, the user need to undertake the code design for MCU code, and the code design of APP for smart mobile terminals.



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

**Features:**

1. Easy to use, no need of any experience of Bluetooth protocol stack application
2. UART design for user interface, full-duplex bi-directional communication, and supporting the minimum baud rate of 4800 bps;
3. Supporting bridge mode (USART transparent transmission), and direct-drive mode (no additional MCU needed);
4. Default connection interval of 20ms, which makes quick connection;
5. Supporting software reset module by AT command, and access to the MAC address;
6. Supporting the adjustment of Bluetooth connection interval by AT command, and the control of different forwarding rates (dynamic power adjustment);
7. Supporting the adjustment of the transmission power by AT command, the change of broadcasting interval, the customization of broadcasting data, the customization of equipment UDID, the setting of data delay (preparation time of user MCU USART receiving), the change of the USART baud rate, and the change of the module names (all settings can be saved after power-off);
8. The length of the UART packets can be any below or equal to the arbitrary length of 200 byte (large packet automatic distribution);
9. High-speed transparent transmission forward rate maximum up to 4 K/S and the stable rate to be at 2.5 K to 2.5 K (IO5, IO6);
10. Supporting the change of module name by APP in mobile devices, the change of UART baud rate and product UDID, the customization of broadcasting contents and cycle (all settings can be saved after power-off);
11. Supporting the remote reset of module by APP in mobile devices, and the setting of transmission power;
12. Supporting the adjustment of Bluetooth connection interval by APP in mobile devices but the setting cannot be saved after power-off (dynamic power adjustment);
13. All IO port expansion, including debug ports

14. Supporting the connection status and the flexible configuration of broadcasting status prompt pin / general IO;
15. 6 two-way programmable IO port, input check triggered by external interrupt, and low power operation (applied in trigger alarm, lighting control, remote control toys, and various i/o switch);
16. 2 programmable single timing / cycling timing output port (applied in smart timing schedule);
17. 2 ADC inputs (14 bit), EN/BAN, free configuration of sampling cycle (applied in temperature/humidity metering, photometry, & etc.);
18. 4 programmable PWM outputs (120 hz) (applied in dimming control);
19. Module-side RSSI continuous acquisition, APP readable and auto-notifying, EN/BAN, free setting of acquisition frequency (applied in finder, anti-loss and alarm);
20. Supporting battery reading and prompt, able to auto upload (notification of remaining battery);
21. Supporting the password setting, modifying and restoring for anti-hijacking, preventing from malicious connection from a third party. Also the notification of independent crypto-operation result to ease the APP programming;
22. Supporting restoring factory settings by 5-second long press and the APP remote recovery;
23. Supporting the custom of PWM output initialization status (low, full, PWM output status value before power-off);
24. Supporting the custom of PWM frequency ( $61.036 \text{ Hz} \leq f \leq 8 \text{ kHz}$ , default 120 Hz);
25. Real-time system status prompt in broadcasting contents, including battery charge, custom UDID, current output value of 4 PWMs or collection value of 2 ADCs , the current state of IO, and etc.;(suitable for broadcast applications);
26. 2 level pulse-width counting,  $0 \sim 0 \text{ XFFFFFFF ms}$  (about 49.7 days);
27. Supporting internal RTC, which can be synchronized any time from APP side;
28. Supporting 6 IO and 4 PWM timing control (default setting OFF);
29. 4 PWMs to support gradient mode (suitable for dimming control);
30. Supporting the saving of IO port configuration and output status, and the customization of the default initialization status;
31. Supporting the shallow recovery and depth recovery modes, which can recover user data



flexibly while reserve the essential configuration of the product;

32. Supporting string prompts of Bluetooth connection status from TX UART (connection, normal disconnection and timeout disconnection) ;

33. Supporting low-level-enabled mode and pulse-width-enabled mode, and the remote shutdown;

34. Supporting auto shutdown after 30 seconds without connection in the pulse-enabled mode;

35. Supporting timeout (break) prompt by square wave alarm in the pulse-enabled mode;

36. Extremely low power in standby mode (0.4 uA of current as per TI Official for CC2540), and the measured power consumption data as follows:

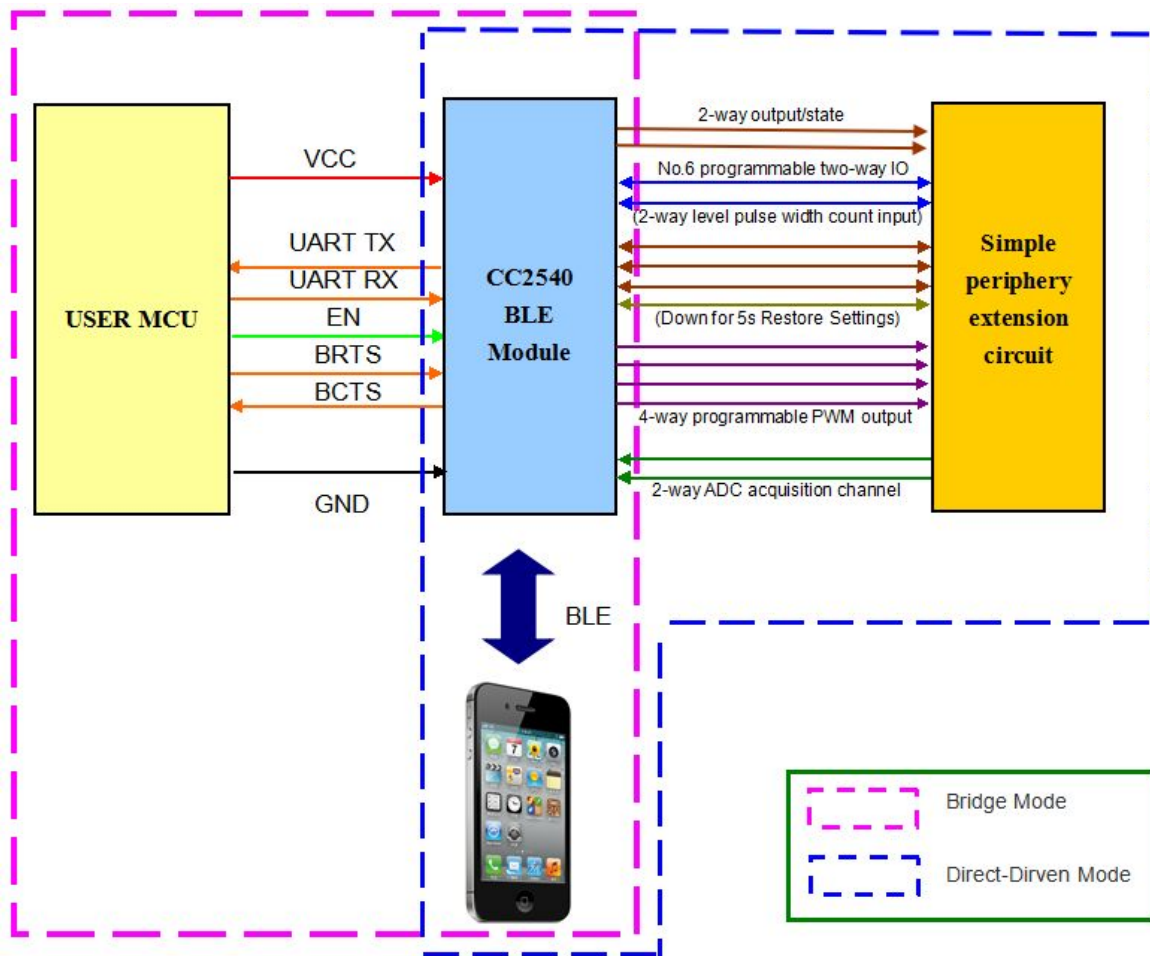
Event	Average current (integral computed*1)	Average current (ammeter measured*2)	Duration	Testing Conditions/Remarks
Sleeping	0.35uA	0.3-0.4uA	—	EN dangling
Broadcasting	202uA	0.14~0.54mA	3.85ms	broadcasting cycle is 250 ms
Connection Event	243uA	0.41 mA	2.25ms	Connection cycle is 100 ms
Single BLE Data Receive Event	332uA	0.65 mA	3.0ms	(20bytes,10 times per second)
Module receiving data and send through USART	497uA	2.68mA	5.1ms	(20bytes,10 times per second)
Single BLE data transmission event	342uA	0.69mA	3.2ms	(20bytes,10 times per second)

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

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

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

## Schematic Diagram of Working Mode

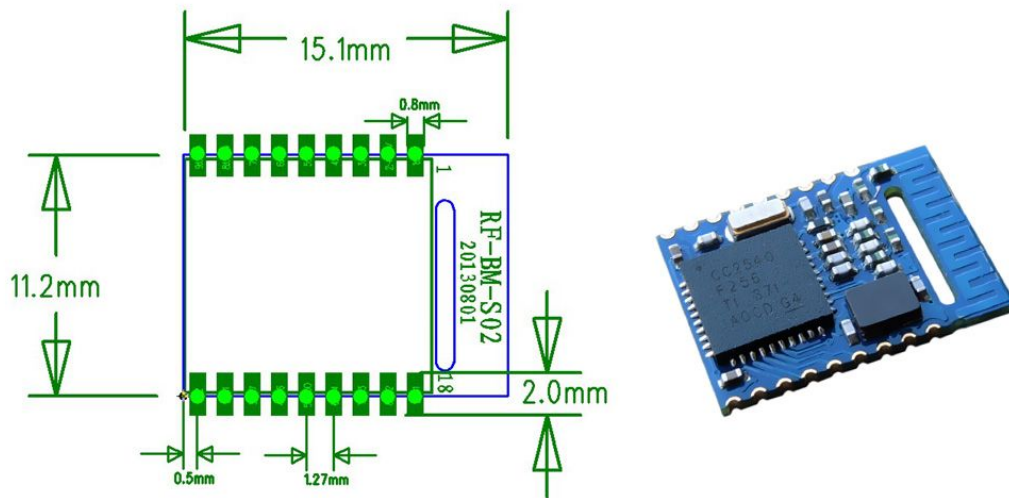


Module Bridge mode and Direct-Driven mode schematic diagram

Note: In order to avoid the output level difference of user MCU's IO and module IO, which will result to high current, a small isolation resistor is suggested to be connected in series in the output signal line TX, BCTS.

## Packaging size and Pins

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



### RF-BM-S02 (BQB Certification)

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

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

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

## UART transparent transmission(Bridge mode)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## UART AT Command

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

### Connection interval setting

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

"TTM:CIT-Xms"

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

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

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

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



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

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

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

## Module Renaming

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

"**TTM:REN-**" + Name

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

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

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

## Baud rate setting

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

"**TTM:BPS-115200**"

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

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

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

## Acquiring physical address MAC

Input the following string to a UART RX:

"**TTM:MAC-?**"

And the following string will be received from TX:

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

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

### ➤ Module resetting

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

"**TTM:RST-SYSTEMRESET**"

### ➤ Broadcast cycle setting

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

"**TTM:ADP(X)**"

Where X = "2", "5", "10", "15", "20", "25", "30", "40" or "50". Confirmation string of "**TTM: OK \r \n \0**"



0" will be received from TX. If the command format incorrect, the following string will be returned:

"TTM: ERP\r\n\0"

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

### **Additional customerized contents of broadcast**

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

"TTM:ADD-"+ Data

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

"TTM:ERP\r\n\0"

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

### **Defining product identification code**

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

"TTM:PID-"+ Data

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

"TTM:ERP\r\n\0"

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

### **Transmission power setting**

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

"TTM:TPL-(X)"

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

"TTM:ERP\r\n\0"

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

### **Data delay setting**

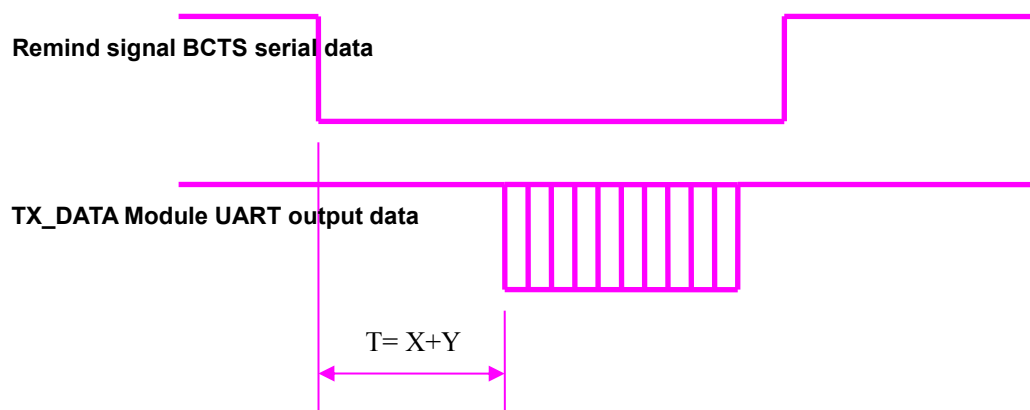
Input the following string to the UART RX to set the delay from when BCTS outputs low to when UART TX outputs data (in ms)

"TTM:CDL-Xms"

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

"TTM:ERP\r\n0"

To make the user CPU have enough time to be waken up from sleep mode and ready to receive data, the module provides this delay (X) setting. The BRTS will be set low before there is data to be sent through the module's UART. While the delay from when BRTS is set low till when the module TX outputs data will be set by this parameter. The actual delay (T) will be  $T = (X + Y)$  ms, if minimum delay is not less than X, while  $500\text{ us} < Y < 1\text{ ms}$ . This setting can be saved after power-off.



Module uart output data delay setting schemes

#### AT command list

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

"TTM:MAC-?"	—	acquire MAC address	"TTM:MAC-xxxxxxx xxxx" xxxxxxxxxxxx for module MAC address	Return with MAC address
"TTM:RST-SY STEMRESET"	—	Reset the module	NO	Resetting the module
"TTM:ADP-(X) "	YES	X = "2", "5", "10", "15", "20", "25", "30", "40", "50"  Set the appropriate broadcast cycle T = X * 100ms  X = "2", "5", "10", "15", "20", "25", "30", "40", "50" set the broadcast cycle T = X * 100 ms	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Setting up the broadcast cycle.(ie.If the parameter set to "5", the cycle will be 500 ms)
"TTM:ADD-" + Data	YES	Data for customerized broadcast Data, and length L < = 16;	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Setting the customerized broadcast contents
"TTM:PID-" + Data	YES	Data for customerized product identification code, Length L = 2, and default value is <b>00</b> <b>00</b> ;	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Setting the customerized product identification code
"TTM:TPL-(X)"	NO	X = "+ 4", " <b>0</b> ", and "6", "- 23" set up the transmission power (in dBm)	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	Transmission power setting
"TTM:CDL-Xm s"	YES	X = " <b>0</b> ", "2", "5", "10", "15", "20", "25" set the delay from when BCTS output is set low till when UART outputs data (in ms)	"TTM:OK\r\n\r\n0" "TTM:ERP\r\n\r\n0"	If minimum delay not less than X, the actual delay will be X + Y ms (500 us < Y < 1 ms).

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

## Broadcast Data Setting

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

```
{
    0x00,0x00,           Customizing equipment type code, default setting 00 00, and can be
                        set by the AT command;

    0x00,0x00,0x00,0x00, Current output status of four PWM (default), or two ADC values;

    0x00,                Percentage of module power suppl (2.0 v = 0%);

    0x00,0x00,           IO configuration, IO output / input status (real-time changing with IO
                        current status);
}
```

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

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

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

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

## System Reset and Recovery

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

4. Use the AT command reset module (see the serial the AT command section);
5. The use of the service channel interface, use the APP to remote reset module.Interface (see the BLE protocol descriptions (APP) - module parameter Settings "section).
6. Using hardware RESTORE foot foot (see definition table), **electricity for 30 seconds**, the foot down after **5 seconds**, module of the system parameters can RESTORE the user level (shallow recovery, release the foot immediately after reset), if continue to lower after **20 seconds** of module, system parameters RESTORE to factory Settings (depth recovery), and immediately reset.The foot with the internal pull-up, not into this mode by default.

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

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

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

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

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

## IOS APP programming reference

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

About BLE under the IOS programming, the key is to **Characteristic value**, (**Characteristic**, **this**

article called channel) to **read**, **write**, and **open switch notice**. To read and write channel through the direct control of the straight drive module function can be realized, no additional CPU. Typical functions that extract is as follows:

```

/*!
 * @method writeValue:forCharacteristic:withResponse:
 * @param data The value to write.
 * @param characteristic The characteristic on which to perform the write operation.
 * @param type The type of write to be executed.
 * @discussion Write the value of a characteristic.
 * The passed data is copied and can be disposed of after the call finishes.
 * The relevant delegate callback will then be invoked with the status of the request.
 * @see peripheral:didWriteValueForCharacteristic:error:
 */
- (void)writeValue:(NSData *)data forCharacteristic:(CBCharacteristic *)characteristic type:(CBCharacteristicWriteType)type;

```

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

```

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

```

```

/*!
 * @method readValueForCharacteristic:
 * @param characteristic The characteristic for which the value needs to be read.
 * @discussion Fetch the value of a characteristic.
 * The relevant delegate callback will then be invoked with the status of the request.
 * @see peripheral:didUpdateValueForCharacteristic:error:
 */
- (void)readValueForCharacteristic:(CBCharacteristic *)characteristic;

```

**Note:** read some characteristic value.

```

[p readValueForCharacteristic:c];

```

```

/*!
 * @method setNotifyValue:forCharacteristic:
 * @param notifyValue The value to set the client configuration descriptor to.
 * @param characteristic The characteristic containing the client configuration.
 * @discussion Ask to start/stop receiving notifications for a characteristic.
 * The relevant delegate callback will then be invoked with the status of the request.
 * @see peripheral:didUpdateNotificationStateForCharacteristic:error:
 */
- (void)setNotifyValue:(BOOL)notifyValue forCharacteristic:(CBCharacteristic *)characteristic;

```

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

```
[self setNotifyValue:YES forCharacteristic:c];//open notice enabled switch.
[self setNotifyValue:NO forCharacteristic:c]; //close notice enabled switch.
```

```
/*
 * @method didUpdateValueForCharacteristic
 * @param peripheral Pheripheral that got updated
 * @param characteristic Characteristic that got updated
 * @error error Error message if something went wrong
 * @discussion didUpdateValueForCharacteristic is called when CoreBluetooth has updated a
 * characteristic for a peripheral. All reads and notifications come here to be processed.
 */
- (void)peripheral:(CBPeripheral *)peripheral didUpdateValueForCharacteristic:(CBCharacteristic *)
characteristic error:(NSError *)error
Note: after each read operations, will perform to the callback function.The application layer
save read data in this function.
```

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

## BLE protocol description (APP interface )

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

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

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

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

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

				devices
--	--	--	--	---------

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

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

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



0x0056)					(sudden change)	
---------	--	--	--	--	-----------------	--

Directions:

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

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

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

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

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

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

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

For example: 0 xFF000000

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

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

Example 1: Output the square wave of 120 Hz. Writing 0x8235 (33333) to FFB3 will set square

wave cycle at  $0x8235 * 0.00000025 = 0.00833325$  s. So the frequency is around 120 Hz;

Example 2: Output the square wave of 1 KHZ. Writing 0x0FAa0 (4000) to FFB3 will set the square wave cycle at  $0x0FA0 * 0.00000025 = 0.001$  s. So the frequency is around 1 kHz;

FFB4 is the channel for length control of 4 PWM output changing time. To write to FFB (2 bytes) can adjust the speed of frequency changes of 4 PWM output square waves. This is a value of time t, and t must meet:  $0 < t \leq 65535$ , (one unit equivalent to 100 ms). The longer the t, the slower the PWM changes from the current value to the target. While the smaller the t, the faster the change takes place. When t is zero, the target will be reached immediately. The changing time of 4 PWM share the same value. Also reading this channel will get the last-time setting, and the setting can be saved after power-off. Factory setting is 0x0000 by default, corresponding conversion mode for immediate mutations.

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

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

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

# Programmable IO (8 Channel) 【Service UUID: 0xFF0】

Characteristic value UUID	Executive Operation	Bytes	Default Value	Remarks
FFF1 (handle: 0x0017)	Read/write	1	0b00000000	<p>IO7 - IO0 configuration bytes.</p> <p>When the corresponding bit is set to 0, <b>bit7,bit6</b> indicate that IO7 and IO6 are signal prompt pin and valid at low level.</p> <p><b>bit5 – bit0</b> indicate that IO5 - IO0 work as input ports</p> <p>When the corresponding bit is set to 1:  <b>bit7, bit6</b> indicate that IO7 and IO6 work as normal output port;  <b>bit5 – bit0</b> indicate that IO5 - IO0 work as output ports</p>
FFF2 (handle: 0x001A)	write	1	--	<p>IO7 ~ IO0 output status.</p> <p>It indicates the output level in IO7 ~ IO0. Bit 7 and bit6 are only valid when IO7 and IO6 work as normal output ports. When IO7,IO6 works as signal prompt pin, bit7 and bit6 are invalid.</p>
FFF3 (handle: 0x001D)	Read/notify	1	0x3F	<p>IO5 ~ IO0 input status.</p> <p>Notifications can be read and received. When notify-enabled switch on, the change of certain input level will be notified to the APP. IO7 and IO6 can only work as output or signal prompt pin, and the corresponding pins are invalid.</p>

Directions: IO configuration and control channel.

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

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

FFF2 is the configuration channel for 8 IO. The 8 bits control IO7 ~ IO0 (8 IO) correspondingly. And they are only valid when the corresponding bits are set to work as output. When certain IO is set to output, corresponding bit in this channel can be written, so the output control of the IO is realized. The bit corresponding to the IO that is set as input will be invalid.

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

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

#### Timed Rollover Output (2 Channel) 【Service UUID: 0xFFE0】

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

FFF6 (handle: 0x0027)	Read/write	4	0x00000000	Setting of first time rollover delay of IO7 0: IO7 rollover not started Non 0: in ms, delay before rollover
FFF7 (handle: 0x002A)	Read/write	4	0x00000000	Setting of second time rollover of IO7 0: No rollover Non 0: in ms, delay before rollover

Directions: Channel to configure timed rollover.

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

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

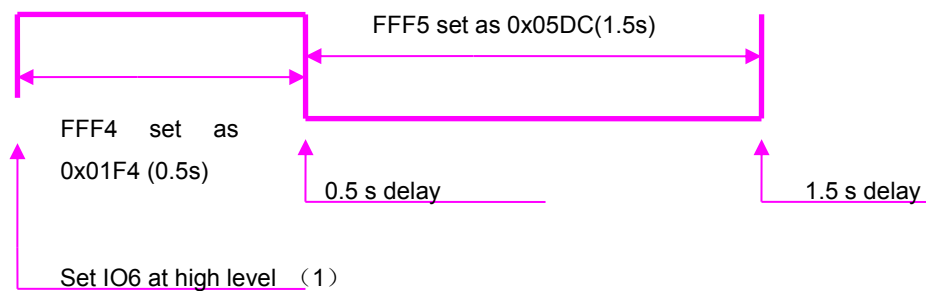
FFF4 is used to set the first time delay of rollover of IO6, and FFF5 is used to set the second time delay of rollover of IO6. If FFF4 is set to 0, rollover of IO6 will not be started. If FFF4 is set to a non-zero value, while FFF5 is set to 0, IO6 rollover is only started for once. FFF5 must be configured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

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

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

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

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



**Rollover Cycle (2 s) Diagram**

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

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

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

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

Directions: counting and notifying channel of IO level duration.

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

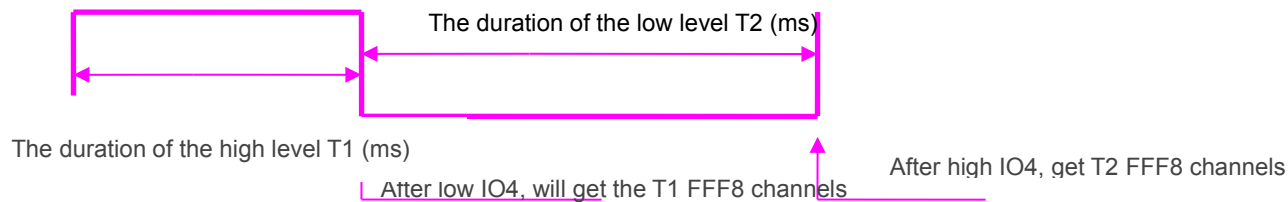
It functions only when the two high bits of FFF1 - BIT5 and BIT4 - are set to zero (as an input port).

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

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

to 4294967295 ms, or about 1193 hours, or about 49.7 days). It can be processed and used by APP in the callback function.

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



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

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

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

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

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

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

Channel UUID	Executa ble Operatio n	B yt es	Example	Remarks
-----------------	---------------------------------	---------------	---------	---------

FFC1 (handle: 0x0045)	write ( <b>Saved</b> after power-of f)	12	"123456123456"(ASCII)	Submit 123456 as the new password , and the new password must be same as the previous one
			"123456888888"(ASCII)	Change the previous password 123456 into the new one of 888888, and the previous password must be correct
			"888888000000"(ASCII)	Cancel the use of password by changing the new password to 000000, and the previous password must be correct
FFC2 (handle: 0x0048)	notify	1	0 (PWD_RIGHT_EVENT)	Password submission correct.
			1 (PWD_ERROR_EVENT)	Password submission error
			2 (PWD_UPDATED_EVENT)	Password update success
			3 (PWD_CANCEL_EVENT)	Cancelling the use of password

#### Description:

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

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

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

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

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

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

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



password is cancelled.

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

Channel UUID	Executable Operation	Bytes	Default Value	Remarks
2A19 (handle: 0x000A)	Read/notify	1	Percentage of power charge	Reading the current battery power percentage, or automatically creating notification

Directions: the channel for battery power reading or notifying.

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

### RSSI Report 【ServiceUUID: 0xFFA0】

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

Directions: RSSI reading and return channel.

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

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

char type.

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

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

Channel UUID	Executable Operation	Whether or not to save	Bytes	Default Value	Remarks
FF91 (handle: 0x0062)	Read/write	YES	16	TA v22u-x xxxxxxx (ASCII string with terminator)	Device name, XXXXXXXX for the last four bytes of the physical address
FF92 (handle: 0x0065)	Read/write	NO	1	0	Bluetooth connection interval: 0: 20ms 1: 50ms 2: 100ms 3: 200ms 4: 300ms 5: 400ms 6: 500ms 7: 1000ms 8: 2000ms
FF93 (handle: 0x0068)	Read/write	YES	1	1	Set the baud rate of UARTs: 0: 4800 bps 1: 9600 bps 2: 19200 bps 3: 38400 bps 4: 57600 bps 5: 115200 bps
FF94 (handle: 0x006B)	write	—	1	no	Channel to control remote reset and recovery:

					<ul style="list-style-type: none"> <li>➤ Remote reset control, by writing <b>0x55</b> to reset the module</li> <li>➤ Remote shallow recovery control, by writing <b>0x35</b> to shallow-recover the module (restoring user data only) and reset</li> </ul> <p>Remote deep recovery control, by writing <b>0x36</b> to deep-recover the module (all back to factory settings) and reset</p>
FF95 (handle: 0x006E)	Read/write	YES	1	0	<p>Set the broadcast cycle:</p> <p>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,</p>
FF96 (handle: 0x0071)	Read/write	YES	2	0x0000	Set product identification number
FF97 (handle: 0x0074)	Read/write	No	1	1	<p>Set the transmission power:</p> <p>0: +4 dBm 1: 0 dBm 2: -6 dBm 3: -23 dBm</p>
FF98 (handle: 0x0077)	Read/write	YES	16	The default broadcast content, See the "broadcast data set" section	<p>Set customized broadcast data</p> <p>,Customizing broadcast data, 0 &lt; n &lt;= 16</p>

FF99 (handle: 0x007A)	write	—	1	无	<p>Remote control extension channel:</p> <ul style="list-style-type: none"> <li>➤ <b>0x01</b>: saving-trigger control of IO configuration output. Writing 0x01 will trigger the saving of current IO configuration and output status. IO7 – IO1 will be initialized to use the saved configuration and output status when power on again. But IO0 is always set as input by default when power on, working as the test port of factory setting recovery.</li> <li>➤ <b>0x02</b>: remote shutdown control. In the pulse-enable mode, writing 0x02 to the channel can shut down the module remotely.</li> </ul>
FF9A (handle: 0x007D)	Read/write	YES	1	0b <b>00000</b> <b>000</b>	<p>System function enabled switch:</p> <p><b>BIT0</b>: Enabled mode setting, 0 by default, corresponding to the level-enabled at low. 1 means pulse-enabled. Every time the EN pin receives a pulse, the module will switch between boot-up (starting broadcast) and shutdown (stopping broadcast) in turn. Effective pulse width T must meet: <math>W &gt; 200 \text{ ms}</math>. When the broadcast time exceeds 30 s but the module is still not connected, it will shut down automatically</p> <p><b>BIT1~BIT7</b>: Temporary unused。</p>

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

Directions: module information configuration channel.

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

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

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

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

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

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

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

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

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

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

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

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

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

Writing **0x01** to this channel will trigger the module to save the current configuration and the output

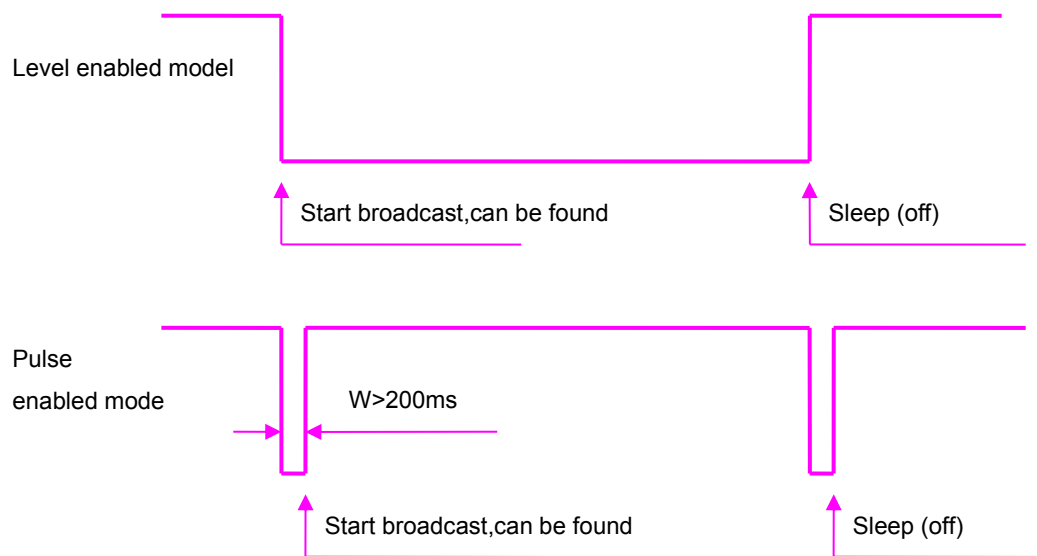
status of all Ios (except IO0). When power on again, the module will always initialize IO7 – IO1 with the saved settings and output status. While IO0 is always set to default input status, to work as the triggering IO of factory setting recovery. But afterwards, IO0 can also be set as output, just as other IOs.

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

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

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

BIT1~BIT7:Reserved.



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

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

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

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

4. If broadcasting for 30 s after enabled, but still not connected, the module will stop

broadcasting and shut down.

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

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

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

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

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

Broadcasting status and IO6 prompt methods in different enabled modes:

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

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

Channel UUID	Executable Operation	Bytes	Default Value	Remarks
2A23 (handle: 0x0003)	Read	8	xxxxxx0000xxxxxx (Hex)	System ID, where xxxxxxxxxxxx is the physical address of

				module chip, with low byte in front
2A26 (handle: 0x0005)	Read	5	V2.2u (ASCII)	Software version number of module

Directions: module information reading channel.

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

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

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

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

#### 6. Timed event setting

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

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

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

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



## 7. Timed task setting

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

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

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

Each port can be configured with up to 32 timed events, and has separate responding switches. There is no conflict among the settings of the ports. And the same timed events can be configured in different port at the same time. But if the action type of the event is invalid for certain ports, the action will be neglected. For example, IO0 port (no PWM output function) is configured to start certain PWM gradual-change event, so IO0 will ignore it when the timed event is triggered. Settings can be done through the reading and writing channel(**UUID:0xFE05**), including the following parameters:

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

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

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

## 8. Timed task enabled Settings

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

UUID:0xFE03						
EVT 0	EVT 1	EVT 2	EVT5 ...EVT 28	EVT2 9	EVT3 0	EVT3 1
Timi ng time	Timi ng time	Timi ng time	.....	Timin g time	Timin g time	Timin g time
Acti on type	Acti on type	Acti on type		Actio n type	Actio n type	Actio n type
Ope ratin g para met ers	Ope ratin g para met ers	Ope ratin g para met ers		Oper ating para meter s	Oper ating para meter s	Oper ating para meter s

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

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

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

- v. Timed event of EVT time overflow triggering;
- vi. Responded port opening up the responding switch BIT which indicates to the timed event EVT;
- vii. Individual enable bit which indicates to the timed event of responding port making IO/PWM enabled;
- viii. General enable bit of timed events EA enabled;

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

#### 5. About the priority:

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

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

#### Notes:

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

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

Characteri stic value	Execut able operati on	Bytes	Byte NO.	Default Value	Definition	Remarks
--------------------------	---------------------------------	-------	-------------	---------------	------------	---------

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

			BYTE 7~ BYTE 1	0x000000 00000000	Second/minute/hour/day/ month/year (L) /year(H)	standing for 32 timed events; ·Timing (in second / minute / hour / day / month / year): Value range same as that for RTC clock. If one of the bytes is invalid (FF), the effective byte time at low level will be treated as loop timing. For example, the following timing (Hex) : 00 FF 01 01 D0 07 01 means the timed event will be triggered in the second of 0 at any minute.
			BYTE 8	0x00	Action Type	·Action type: Value range: 0: no action; 1: IO output low level; 2: IO output high level; 3: IO level rollover; 4: PWM sudden change; 5: PWM gradual change;
			BYTE 9	0x00	PWM New Duty value	·PWM New Duty Value: Action types are effective for 4 or 5, after the change of Duty values, values range: 0 ~ 255; When is 0, the duty ratio is 0%, that is the low level, When is 255 , the duty ratio is 100%, that is the high level;
			BYTE 10	0x00	Low Byte of PWM Gradual Change	·PWM sudden change times: Means the time that it takes from the current duty ratio changing to a new duty ratio. It is valid when the operation value is 5. Greater value means slower change, while
			BYTE E1 1	0x00	High Byte of PWM Gradual Change	

						smaller value means faster changes. Value range: 0 ~ 65535 (in 100 ms);
FE04 (handle: 0x008F)	R/W	1	BYTE 0	0x00	Port index number	Indicator of port event reading Port index: The indicator must be set before reading all the applicable events of certain port, to be directed to the port that will be read, followed by reading to FF05. Value range: 0 ~ 9: respectively IO0 ~ IO5 and PWM0 ~ PWM3 timing ports.
FE05 (handle: 0x0092)	R/W	5	BYTE 0	0x00	Port index number	Port events reading and writing channel.  ·Port index number Value Range: 0 ~ 9: respectively standing for IO0 ~ IO5 and the timing ports of PWM0 ~ PWM3.
			BYTE 4 ~ BYTE 1	0b00000000 0000000000 0000000000 0000000000	Timed events enabled 0 ~ 31	·Enable switch of applicable events, with different bit corresponding to the 31 timed events respectively. Value range: 1: on; 0: off;

FE06 (handle: 0x0095)	R/W	2	B Y T E 0	0b0000000 0	Bit0: General Enable of Timing	Event port configuration.  Value range: 1: enabled; 0: off;  · BYTE0: BIT0, for general enabling of port events (EA). · BYTE0: BIT1 ~ BIT7, BYTE1: BIT0 ~ BIT2, both for separate enabling of timed port events. · BYTE1: BIT3, for control of all timed event clearance;(CEVT) · BYTE1: BIT4, Empty for timing task control bit, remove the response configuration of all ports to any timing event. (CPORT)
					Bit1:IO0 Enable	
					Bit2:IO1 Enable	
					Bit3:IO2 Enable	
					Bit4:IO3 Enable	
					Bit5:IO4 Enable	
					Bit6:IO5 Enable	
					Bit7:PWM0 Enable	
					Bit0:PWM1 Enable	
					Bit1:PWM2 Enable	
					Bit2:PWM3 Enable	
					Bit3:Control of timed event clearance	
					Bit4: Control of timed task clearance	
					-	
					-	
					-	
					-	
					-	

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

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

Examples:

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

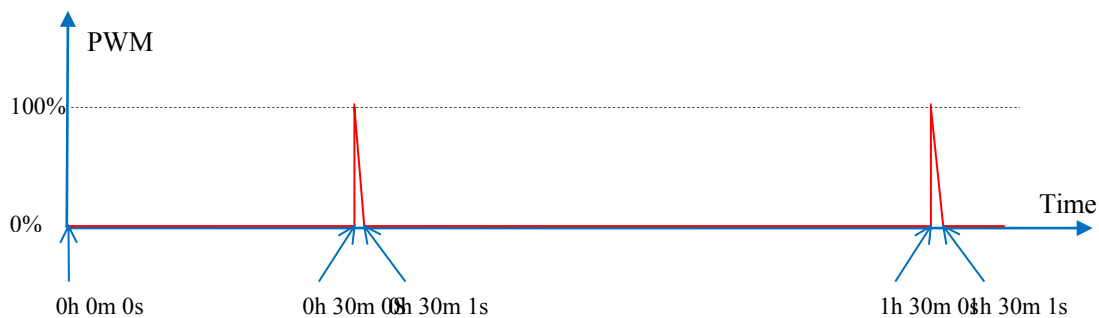


BLE master device operation steps to slave module:

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

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

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



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

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

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



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

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

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

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

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

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

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

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

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

## Transmission test by APP

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

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

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



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



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



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

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

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

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

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

## Test by USB Dongle and Btool

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

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

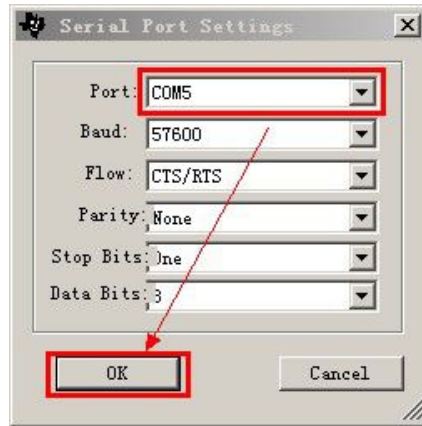
C:\Texas Instruments\BLE-CC254x-1.2.1\Projects\ble\HostTestApp\CC2540 to compile and download the tool into a USB dongle. The details of BTOOL can be found in the official documentation

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

## Connect BLE module

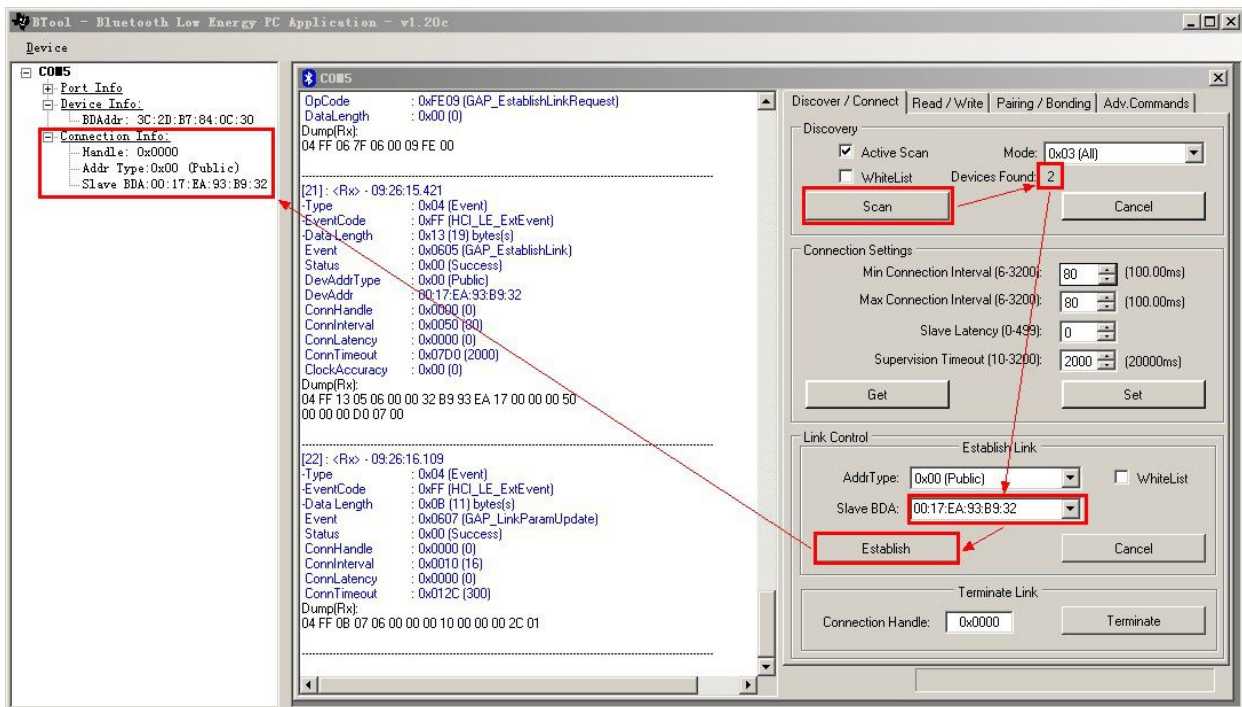
The connection of USB Dongle and the module is basis of communication. The steps of scanning and connecting are as follows:

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



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

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



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

## Testing Direct-Drive function

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

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

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

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

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

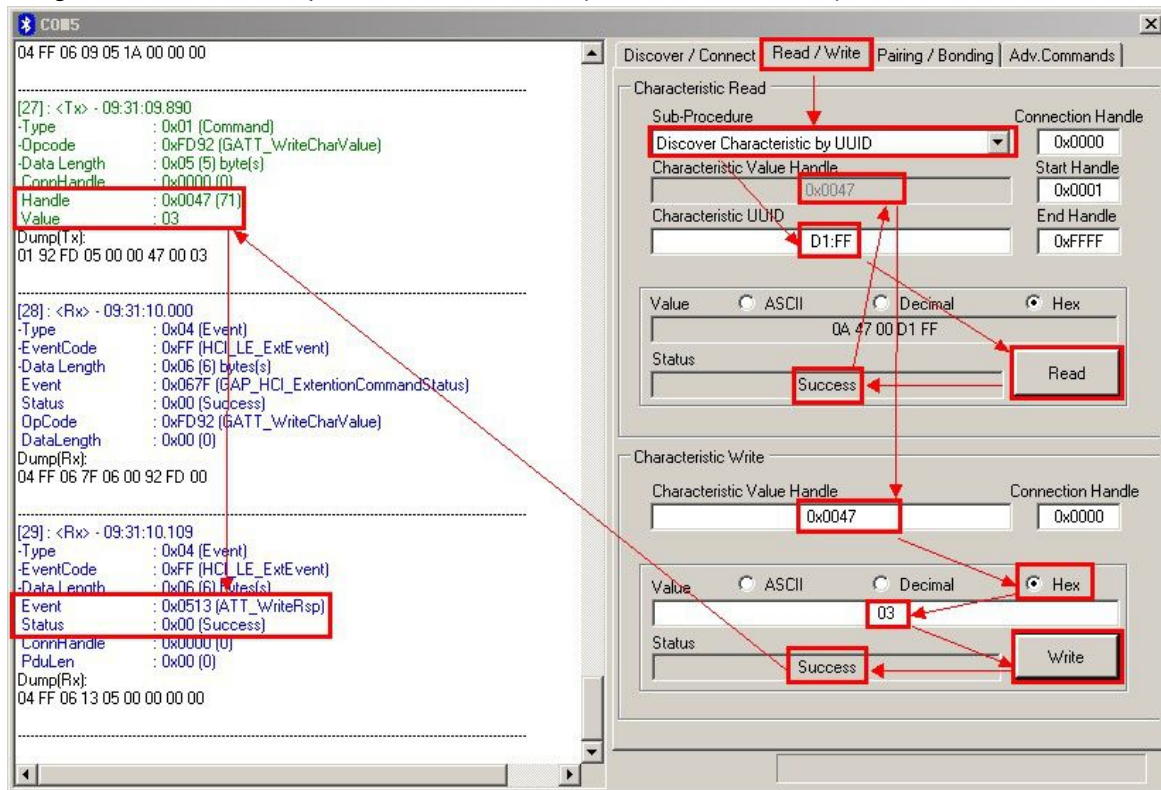
Now take the example of ADC to illustrate hwo to use the USB Dongle and Btool to drive directly the bluetooth module.

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

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

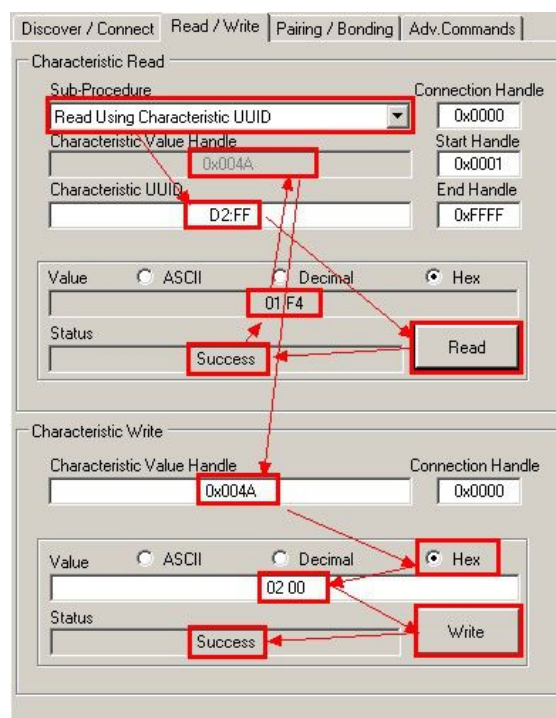
Channel UUID	Executable operation	Bytes	Default value	Remarks
FFD1 (handle: 0x0047)	Read/write	1	0x00	Enable control 0x00:Close the two ADC channels 0x01:Open ADC channels 0x02:Open ADC1 channel  0x03:Open the two ADC channels
FFD2 (handle: 0x004A)	Read/write	2	0x01F4	Sampling cycle ( ms ) ie. 0x01f4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

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

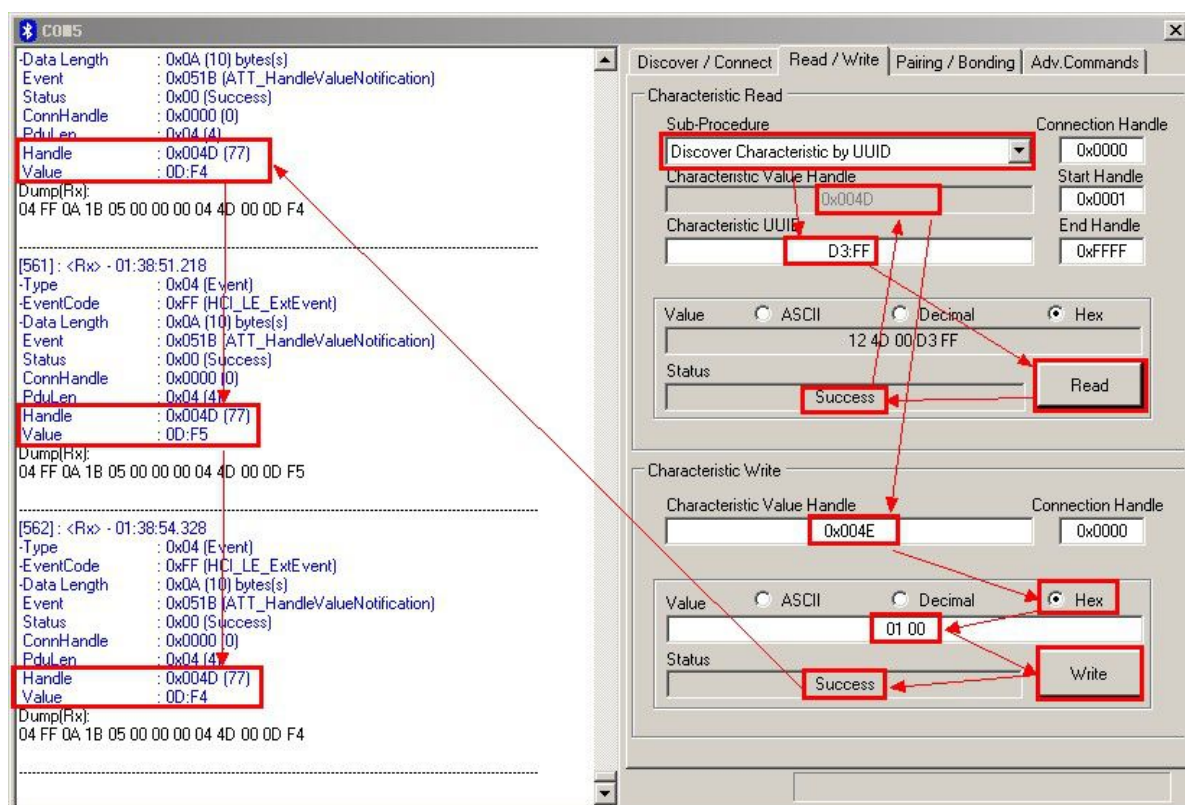


2. Read and reset the sampling cycle. First to read FFD2 channel and get 01 F4, which means the default value is 500 ms. Get the handle = 0x004A and write 0200 (high byte in the front) to the handle. And the the sampling cycle will be set at 512 ms ( 0x0200 = 512 ms).



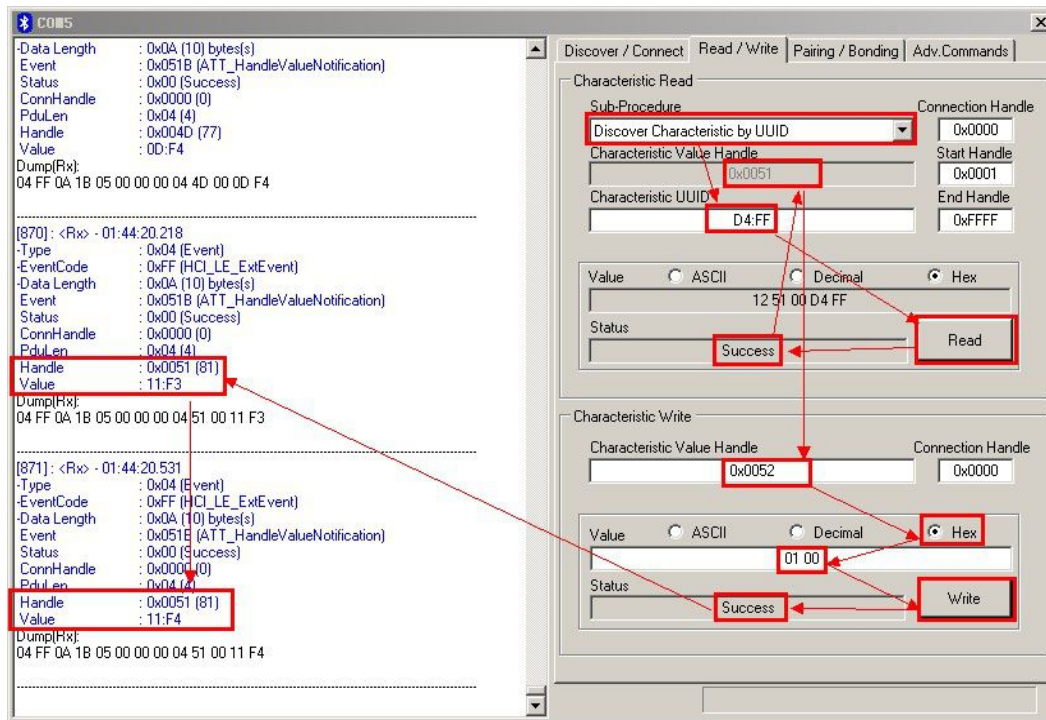


3. Open notify enable switch of ADC0. The address of notify switch is  $\text{handle}+1$ ,  $0x004D + 1 = 0x004E$ . Since then, every time when the sample cycle of ADC0 is different from the last time one, there will be a new notification.

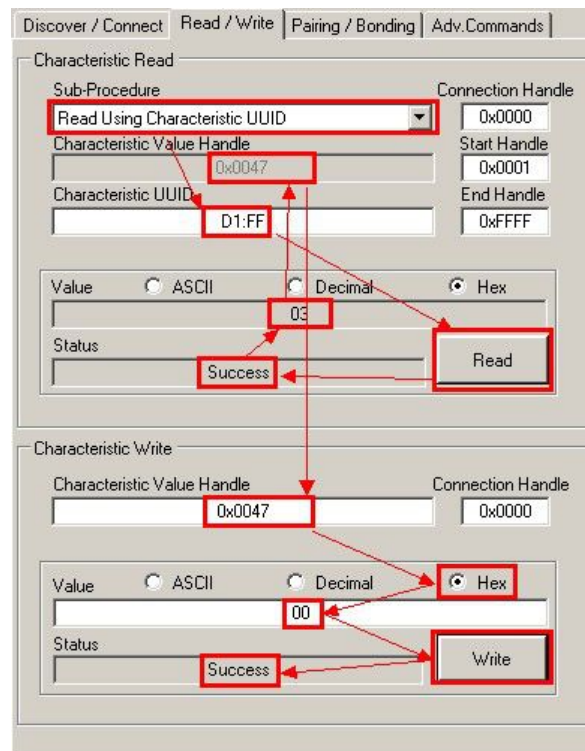


4. Open the notify enable switch of ADC1 channel. The address of notify switch is  $\text{handle}+1$ ,

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



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



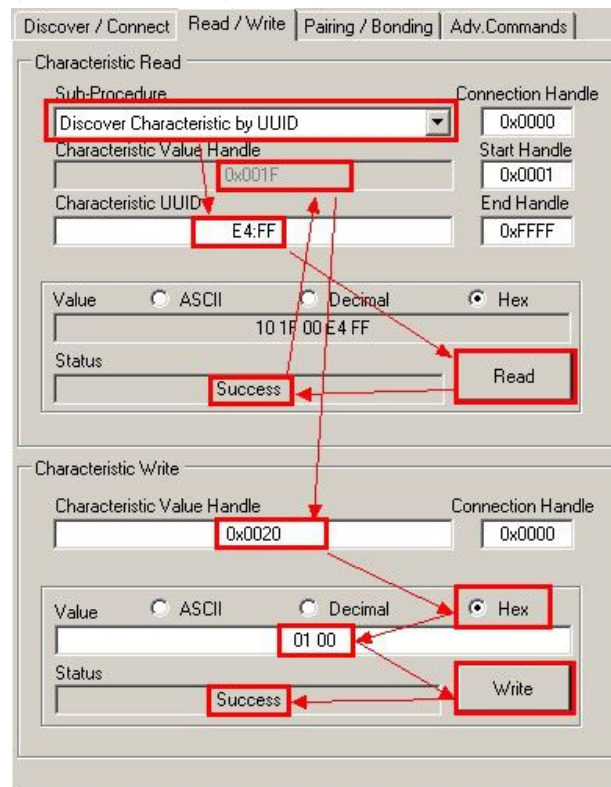
## Testing transparent transmission function



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

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

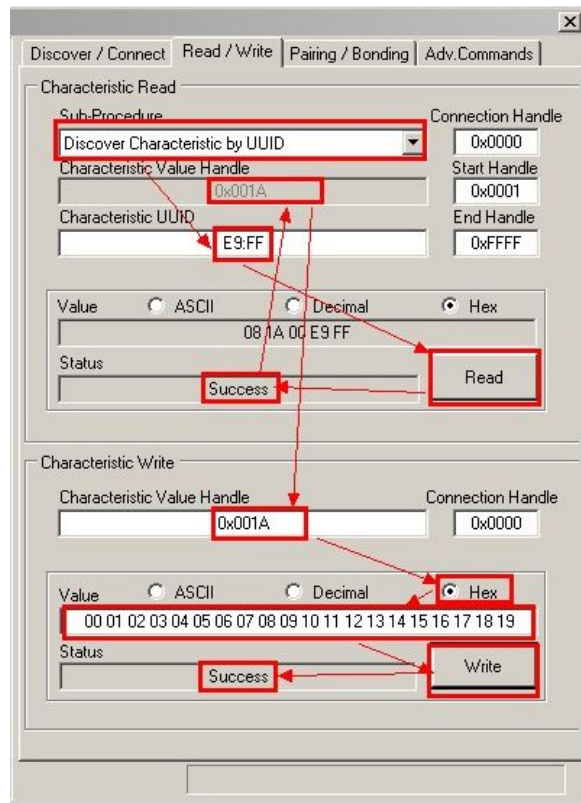
1.The auto-notify switch of serial data channel will be turned on by writing 01:00 to handle: 0x0020, after the BLE module is connected to USB Dongle with Btools (as shown in the diagram below). If the host sends legal data packets to BLE RX, the module will send a notification automatically to Btool,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



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

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

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



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

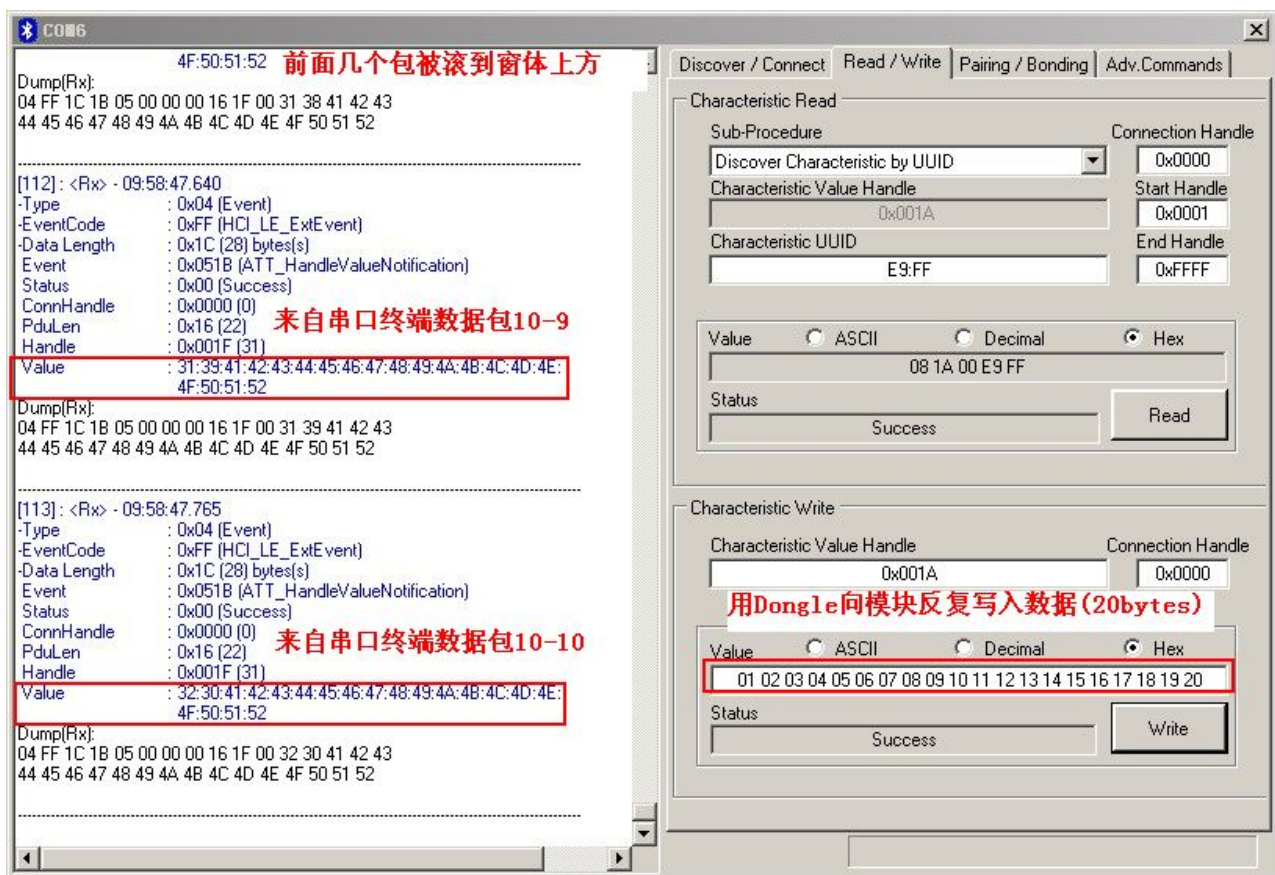
Mobile devices send data to the module through bluetooth data channel, UUID of which is as follows:

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

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

Please refer the screenshots below:

#### 1. Transceiving data with BTool



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



## Master reference code (transparent transmission)

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

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

```
void main(void)
```

```
    EN = 0 ;                                //Enable,start broadcast

    while(!BLEMoudleAck("TTM:OK\r\n0"));    //Waiting for Phone scan,Connecting

                                           //Waiting for the connection is
                                           //successful,Also can add wait time limit

                                           //Also can judge connection prompt signal
                                           //level

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

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

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

    BRTS = 1;                               //RTS high and sent

    while(!BLEMoudleAck("TTM:OK\r\n0"));    //Waiting to set success, also can add wait
                                           //time limit

    while(1){                               //Loop transceiver test
        while(1){
            if(BCTS == 0){                  //Testing,if BCTS low is ready to receive

                while(BCTS==0);              //Waiting to be sent, but also timed wait

                if(UARTRead(uartBuffer) == SUCCESS) //USART to read data

                    {... ...}               //Service data
```

```

    }
    BRTS = 0;                                     //RTS low,and notice CC2540 module is
                                                    ready to receive
    halMcuWaitMs(2);                               //Delay for 2ms

    send_TX("1234567890");                         //Send any data （within 200byte）

    halMcuWaitMs(5);                               //Delay for 5ms,and ensure the data has
                                                    been issued
    BRTS = 1;                                       //RTS high, sent

    halMcuWaitMs(20);                              //Delay again send next packet, and delay
                                                    depending on the packet size
    }
}
}
}

```



## Contact Us

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

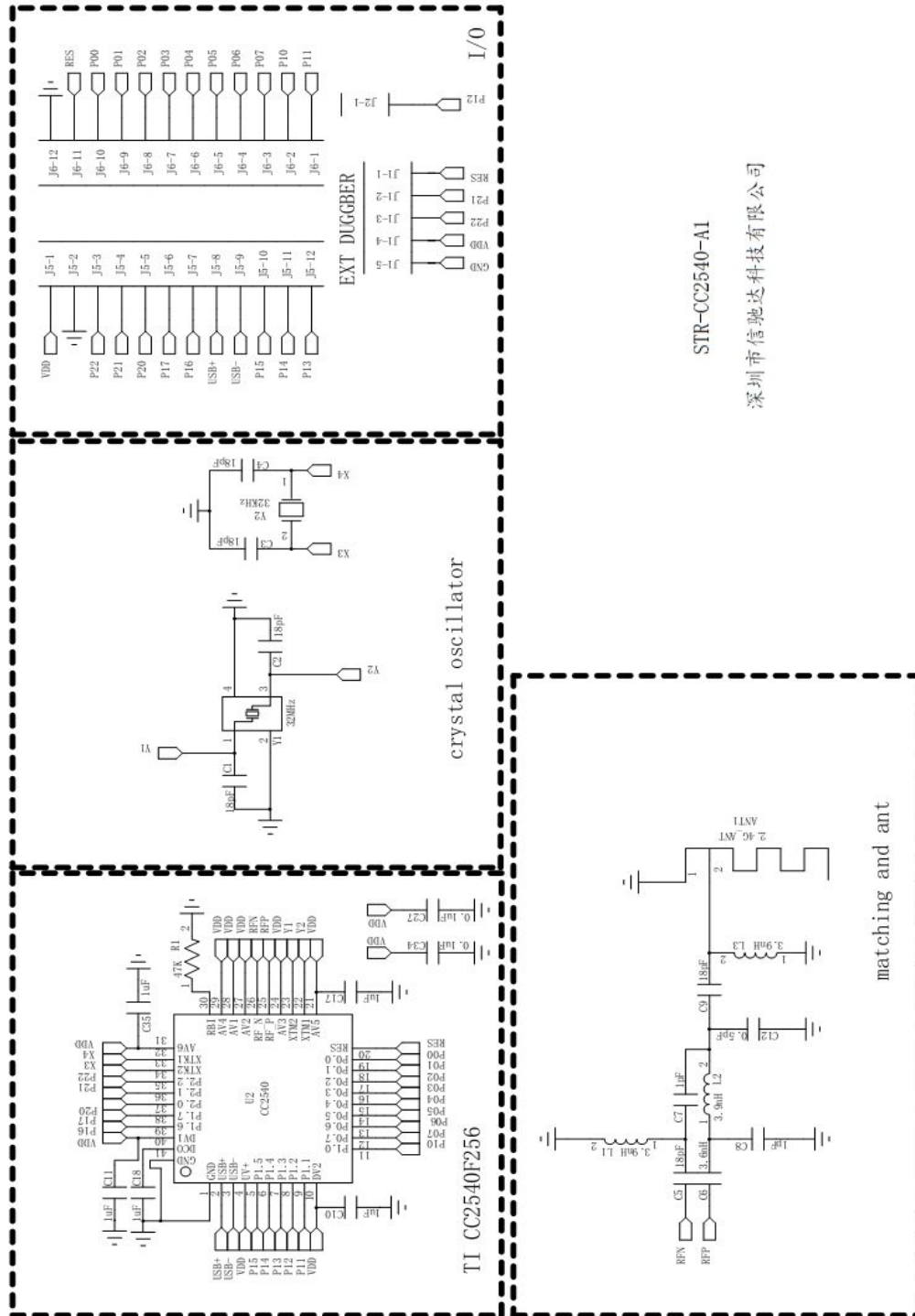
Tel: 0755-8632 9829 Web: [www.szrfstar.com](http://www.szrfstar.com)

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

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



附录 A: 模块原理图  
Appendix B: Schematic Diagram



STR-CC2540-A1

深圳市信驰达科技有限公司

RF-CC2540TA1







145

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

TCB

GRANT OF EQUIPMENT  
AUTHORIZATION

TCB

Certification  
Issued Under the Authority of the  
Federal Communications Commission  
By:

Nemko Canada Inc.  
303 River Road  
Ottawa, Ontario, K1V 1H2  
Canada

Date of Grant: 01/17/2014  
Application Dated: 01/17/2014

ShenZhen RF-STAR Technology CO.,LTD  
2F,BLDG.8,Zone A,BaoAn Internet Industry Base,  
BaoYuan Road,XiXiang, BaoAn DIST,  
ShenZhen,  
China

Attention: Aroo woo

NOT TRANSFERABLE

EQUIPMENT AUTHORIZATION is hereby issued to the named GRANTEE,  
and is VALID ONLY for the equipment identified hereon for use under the  
Commission's Rules and Regulations listed below.

FCC IDENTIFIER: 2ABN2-RFBMS01  
Name of Grantee: ShenZhen RF-STAR Technology  
CO.,LTD  
Equipment Class: Digital Transmission System  
Notes: Bluetooth 4.0 (BLE) Module  
Modular Type: Limited Single Modular


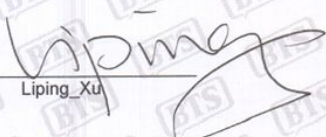

<u>Grant Notes</u>	<u>FCC Rule Parts</u>	<u>Frequency Range (MHZ)</u>	<u>Output Watts</u>	<u>Frequency Tolerance</u>	<u>Emission Designator</u>
	15C	2402.0 - 2480.0	0.00216		

Limited modular approval. Power output listed is conducted.



## 附录 C : RoSH 认证证书

### Appendix C: RoHS Certification

	
<b>TEST REPORT</b>	
REPORT No.: <b>BTS201312302663</b>	Date: 2014-01-09
Page 1 of 8	
Client Company: <b>Shenzhen RF-STAR Technology Co., LTD</b>	
Client Address: <b>2F,BLDG.8, Zone A, BaoAn Internet Industry Base, BaoYuan Road, XiXiang, BaoAn DIST, ShenZhen</b>	
Report on the submitted samples said to be:	
Sample Name	: Bluetooth 4.0 (BLE) Module
Style/ Item No.	: RF-BM-S01; RF-BM-S01_V1.1
Product Rating	: DC3.3V, Max40mA; Max0.132w; 2483.5MHz
Operating Frequency Range	: 2402-2483.5MHz
RF Output Power	: -23dBm-4dBm
Sample Receiving Date	: December 30, 2013
Testing Period	: December 30, 2013 to January 09, 2014
Results	: See next pages
*****	
Summary of Test Results:	
<b>TEST REQUEST</b>	<b>CONCLUSION</b>
A EU RoHS Directive 2011/65/EU and its amendment directives	Pass
*****	
Signed for and on behalf of BTS	
 Liping_Xu	
	
<small>This document is issued by the Company subject to its General Conditions of service printed overleaf, available on request for electronic format documents, subject to terms and conditions for electronic documents. is drawn to the limitation of liability indemnification and jurisdiction issues defined therein. any holder of this document is advised that information contained hereon reflects the company's findings at the time of its intervention only and within the limits of client's instructions. if any, the company's sole responsibility is to its client, and this document does not exonerate parties to a transaction from exercising all their rights and obligations under the transaction documents. this document cannot be reproduced except in full, without prior written approval of the company. any unauthorized alteration, forgery or falsification of the content or appearance of this document is unlawful and offenders may be prosecuted to the fullest extent of the law, unless otherwise stated the results shown in this test report refer only to the sample(s) tested.</small>	
<small>Bestow Technology Service Co., LTD (BTS) 深圳市贝思特技术服务有限公司</small>	<small>716# of General Building, Bantian Hi-Tech, industry park, Longgang District, Shenzhen City, Guangdong Province, P.R.China 中国广东省深圳市龙岗区坂田高新技术产业园综合楼716#</small>
	<small>传真 / fax: 0755-89589101 chaxun@bestow.cn 邮编 / post code: 518129 电话 / telephone: 0755-89500120-8015</small>

附录 D : 蓝牙 BQB-EPL 认证书

Appendix D: End Product Listing

# EPL Bluetooth® End Product Listing

## The Bluetooth SIG Hereby Recognizes

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

Member Company

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

Qualified Design Name

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

Contact Person: **Aroo Wong**

Series: **V1.1**

Publish Date: **13 November 2013**

EPL Type: **Other**

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

