



## BH-SPP-CA 蓝牙透传模块 用户手册

### 修订历史

日期	版本	更新内容
2016/11/14	1.0.0	-
2017/8/8	1.0.1	新增 407、MINI 开发板说明





## 文档说明

本手册旨在帮助用户正确构建 BH-SPP-CA 蓝牙模块的使用环境，引导用户快速使用该模块。

关于模块的原理图、机械尺寸等说明请参考《SPP-CA-蓝牙模块原理图》文档。

关于 SPP-CA 模块的 AT 指令说明，请参考《SPP-C 蓝牙模块 AT 指令集》文档。



## 目录

BH-SPP-CA 蓝牙透传模块 .....	1
用户手册.....	1
文档说明.....	2
目录 .....	3
1. SPP-CA 简介 .....	4
1.1 SPP-CA 简介 .....	4
1.2 特性参数.....	4
2. 硬件测试.....	5
2.1 硬件连接.....	5
2.2 测试流程.....	7
2.2.1 开发板与安卓手机进行蓝牙通讯.....	8
2.2.2 电脑与安卓手机进行蓝牙通讯.....	11
3. SPP-CA 注意事项及常见问题 .....	14
4. 配套程序说明.....	16
4.1 SPP-CA 驱动 .....	16
4.2 main 函数执行流程.....	20
5. 产品更新及售后支持.....	21



## 1. SPP-CA 简介

### 1.1 SPP-CA 简介

BH-SPP-CA 是秉火科技推出的蓝牙串口模块, 它采用蓝牙 2.1 协议, 可与任何版本的蓝牙兼容通讯, 包括与具有蓝牙功能的电脑、蓝牙主机、手机、PDA、PSP 等终端配对, 可实现串口透传功能。驱动 SPP-CA 模块时只需要使用 TTL 电平标准的串口即可(5V/3.3V 电压均可), 支持的波特率范围为 4800~230400, 非常适合用于单片机系统扩展蓝牙特性。

注意 SPP-CA 模块仅支持蓝牙的从模式通讯, 即如果使用两个 SPP-CA 模块, 它们之间是无法建立蓝牙配对连接的。若需要扩展主模式的蓝牙, 可以选用我们的 BH-HC05 模块。

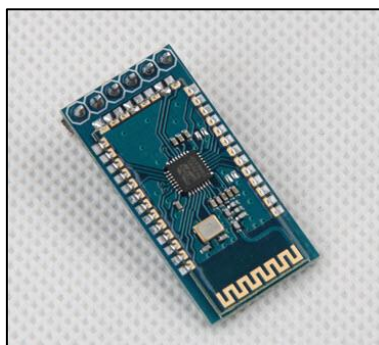


图 1-1 SPP-CA 外观图

### 1.2 特性参数

参数	说明
供电	3.3V-5V
通讯接口	TTL 串口, 支持波特率 4800~230400, 默认波特率为 9600。
通讯距离	10 米
频段	2.40GHz-2.48GHz
蓝牙协议	蓝牙 2.1。带 EDR, 2Mbps-3Mbps 调制度
天线	板载 2.4GHz 天线, 自适应跳频技术
无线发射功率	-4~6dBm(蓝牙 Class2 功率级别)
无线接收灵敏度	-80dBm
误码率	0。但会在传输链路产生误码, 如 RS232 和 TTL 线路处理中。
应用领域	蓝牙转串口透传数据。
工作温度	-40~ +150℃
功耗	以下数据为波特率 38400、从机、串口透传模式下的功耗 通讯中: 10.3mA



已配对未通讯: 4.6mA 休眠电流: 小于 1mA
-------------------------------

## 2. 硬件测试

本模块配套 STM32 驱动程序, 可直接使用秉火 F103 指南者、霸道及 F429 挑战者开发板进行测试。按要求使用杜邦线把模块连接到开发板, 并下载对应的程序即可。

### 2.1 硬件连接

BH-SPP-CA 模块外观见图 2-1, 模块引出了 6 个引脚, 在其背面标有引脚名称的丝印。



图 2-1 BH-SPP-CA 模块背面丝印



表 2-1 SPP-CA 模块引脚说明

序号	引脚名称	说明	与 F103 霸道、指南者及 F407 霸天虎连接	与 F429 挑战者连接	与 F103-MINI 开发板连接
1	VCC	3.3/5V 电源输入	接 3.3V 或 5V	接 3.3V 或 5V	接 3.3V 或 5V
2	GND	地线	GND	GND	GND
3	TXD	串口数据发送引脚, TTL 电平	PA3 (注意跳帽)	PD6 (注意跳帽)	PA3 (注意跳帽)
4	RXD	串口数据接收引脚, TTL 电平	PA2 (注意跳帽)	PD5 (注意跳帽)	PA2 (注意跳帽)
5	NC	空引脚, 因兼容而预留, 不需要连接。	空引脚不用接	空引脚不用接	空引脚不用接
6	INT	配对状态输出 <input type="checkbox"/> 配对状态时输出为高电平 <input type="checkbox"/> 未配对时输出为低电平	PB13	PB10	PA7

**注意跳帽:** 在 F103 霸道及 F407 霸天虎开发板 USART2 的 PA2/PA3, F429 挑战者开发板 USART2 的 PD5/PD6 默认都通过跳线帽连接到了板载的 MAX3232 芯片, 存在引脚共用的问题, 使用蓝牙模块时, 必须把板子 JTAG 接口附近的这部分跳线帽拔掉, 见表 2-2。

表 2-2 F103 霸道、F407 霸天虎及 F429 挑战者需要拔掉的跳线帽

F103 霸道及 F407 霸天虎开发板	F429 挑战者开发板
拔掉跳帽: PA2<->T1IN、	拔掉跳帽: PD5<->T1IN、
拔掉跳帽: PA3<->R1OUT	拔掉跳帽: PD6<->R1OUT

在 F103 指南者及 F103-MINI 开发板没有这个问题, 只要连接好杜邦线即可。

按照表 2-1 把 BH-SPP-CA 模块与各个开始板连接后的效果见图 2-2。



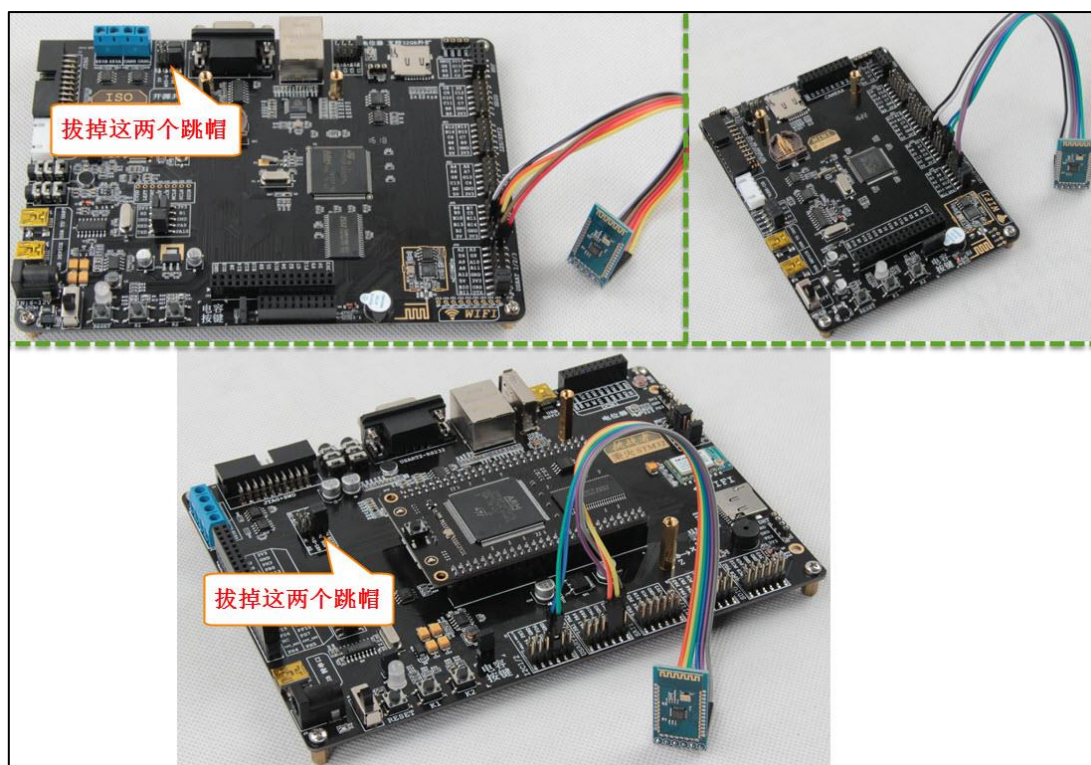


图 2-2 BH-SPP-CA 模块与各个开发板的连接图 1

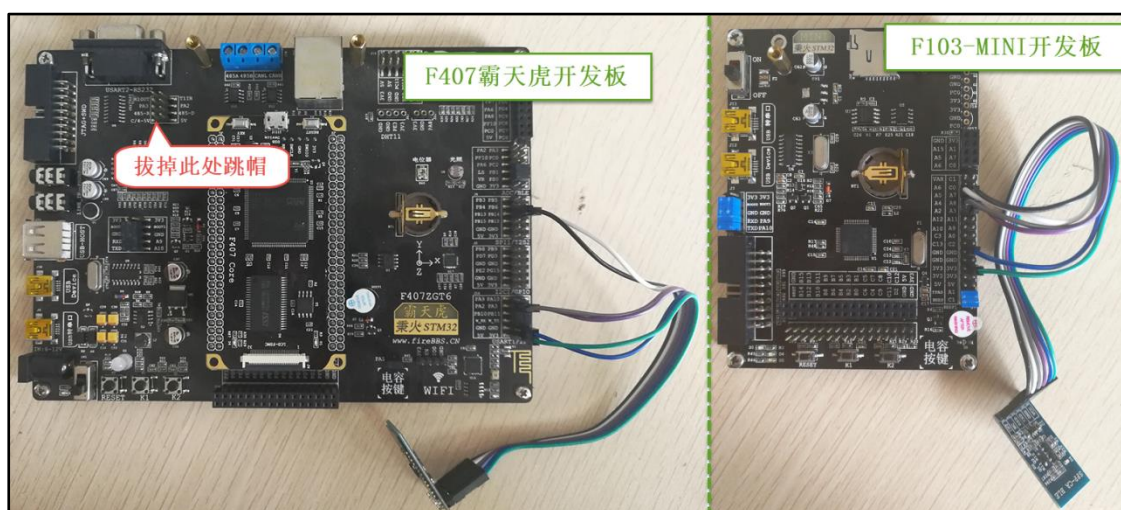


图 2-3 BH-SPP-CA 模块与各个开发板的连接图 2

## 2.2 测试流程

SPP-CA 模块支持两种测试方式:

- ❑ 方法 2: 使用开发板连接 SPP-CA 模块, 开发板输出 SPP-CA 模块的信息。再使用安卓手机与 SPP-CA 模块进行蓝牙通讯。



- ❑ 方法 1: 使用 USB 转 TTL 串口线, 把 SPP-CA 模块与电脑连接起来, 通过电脑的串口调试助手获取 SPP-CA 模块的信息。再使用安卓手机与 SPP-CA 模块进行蓝牙通讯。

## 2.2.1 开发板与安卓手机进行蓝牙通讯

### 1. 准备测试环境

- ❑ 在安卓手机上安装“BTClient”APP。  
BTClient APP 所在目录: “配套软件\HC-PDA-ANDROID.apk”。
- ❑ 按表 2-1 连接好模块后, 找到配套资料里的例程“1. SPP-CA 测试程序(带液晶)和 2. SPP-CA 测试程序(不带液晶)”这两个程序均可, 根据自己的实验平台选择, 带液晶的程序需要给开发板接上配套的液晶, 不带液晶的程序通过串口输出运行状态信息。使用 MDK 编译并下载该程序到开发板, 复位开发板让程序运行。  
工程所在目录: “开发板配套例程\霸道版本[或指南者、挑战者版本]”。

### 2. 正常运行的实验现象

- ❑ 开发板的液晶屏会显示“SPP-CA BlueTooth demo, SPP-CA module detected! Device name :BT\_FIRE”三行文字, 见图 2-4; 可以长按开发板的 KEY2 按键(注意不要按 KEY1), 会随机生成一个新名字。

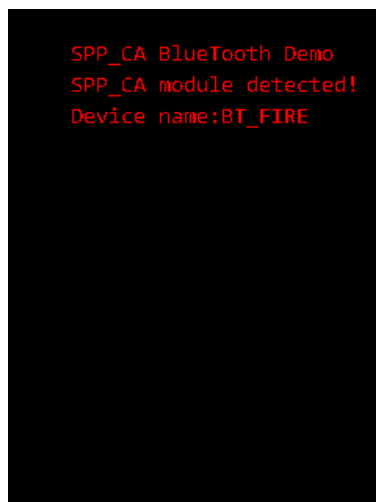


图 2-4 SPP-CA 模块正常运行时的液晶显示

- ❑ 打开手机的 BTClient APP, 打开手机的蓝牙设备, 点击界面左下角的“连接”按钮, 会弹出“查找设备中...”的窗口, 稍等一下会出现名为“BT\_FIRE”或“BT04-A”的蓝牙设备, 这就是我们的 SPP-CA 模块的名字, 点击该设备, APP 会与 SPP-CA 尝试连接, 输入配对码“1234”, 即可连接成功。APP 的操作见图 2-5, 开发板的液晶显示见图 2-6。



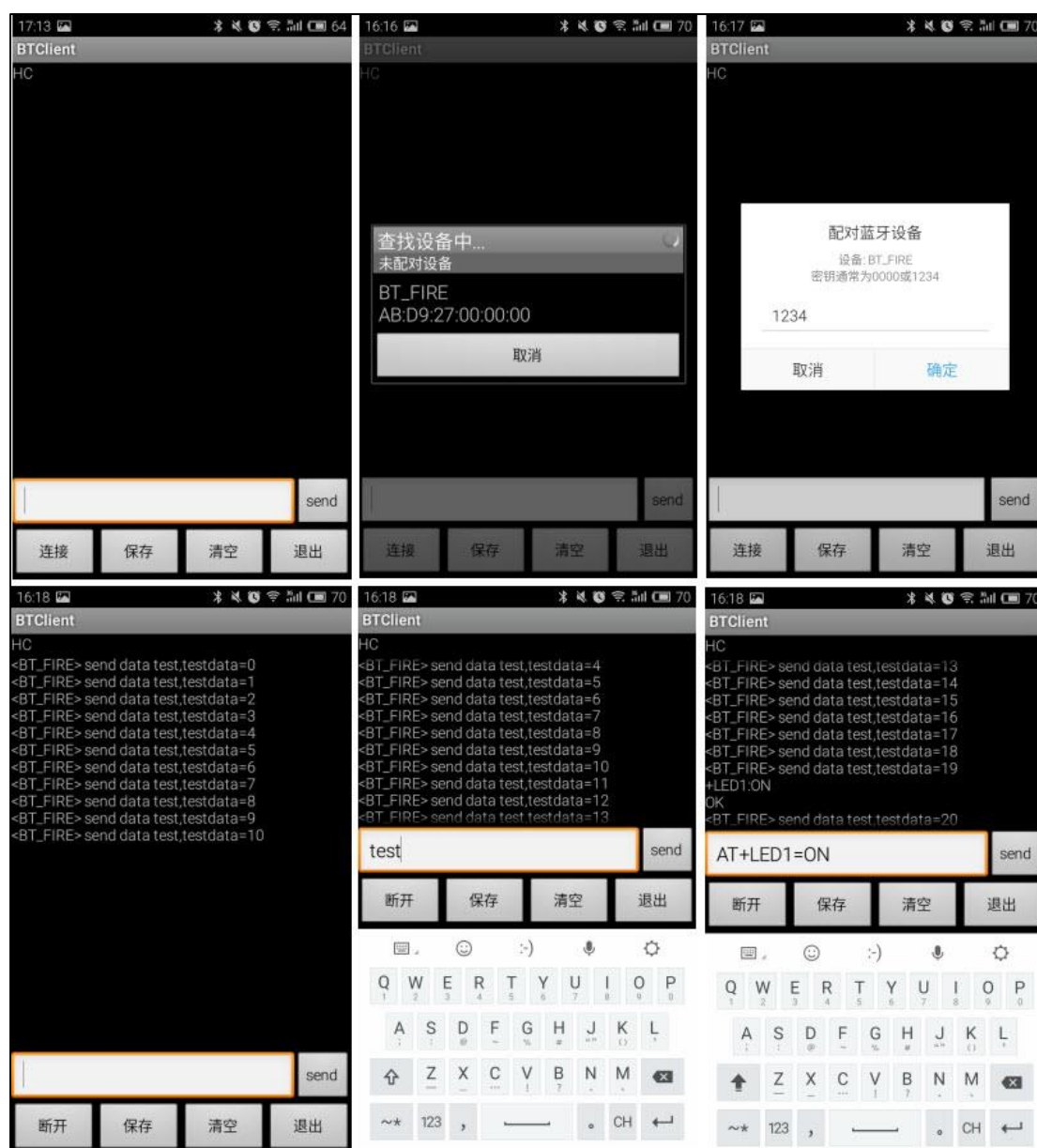


图 2-5 手机上 BTClient 的操作

- ❑ APP 与模块连接上后, SPP-CA 模块会主动向 APP 发送类似 “<Device Name> send data test, test data=<x>” 的字样, 其中<Device Name>为 SPP-CA 模块的名字, <x>则为一个自加的字符串编号。
- ❑ APP 与模块连接上后, 可以使用 APP 通过蓝牙向开发板发送数据。点击界面中的输入框, 输入字符串, 然后点击 “send” 发送数据, 开发板接收到数据后会在液晶上原样显示出来。(可输入回车, 但程序中的液晶处理只显示第一行数据。)
- ❑ 除了发送普通字符串, 还可以向开发板发送命令控制板子上的 LED 灯。开发板配套的程序支持两条命令: “AT+LED1=ON” 和 “AT+LED1=OFF”, 分别用于点亮开发板上的 LED1 和关闭 LED1。开发板接收到命令后, 会在液晶上



显示接收到的命令，并且控制 LED 灯。如果需要其它命令，用户可修改配套的 STM32 程序，实现自定义的功能。

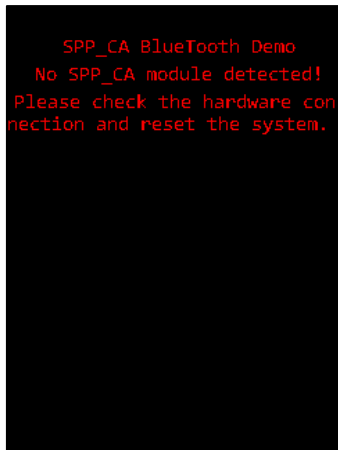


图 2-6 与手机 APP 通讯时的液晶显示界面

实际运行时，液晶界面显示的“receive data”内容可能有细微差异，它不影响程序状态。

### 3. 不正常运行时故障排查

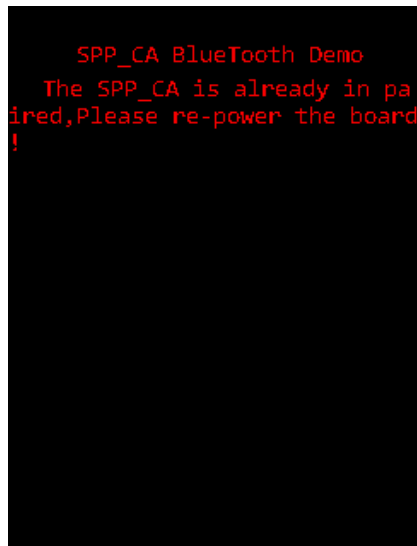
液晶屏显示“No SPP-CA detected!”，见图 2-7。这时说明开发板检测不到 SPP-CA 模块，请参照引脚连接表重新检查模块与开发板之间的连线，并给开发板上电重新测试。



```
SPP_CA BlueTooth Demo
No SPP_CA module detected!
Please check the hardware connection and reset the system.
```

图 2-7 模块运行不正常时的液晶显示

若模块当前已与蓝牙主机处于配对连接状态，再复位开发板，液晶屏上可能会输出图 2-8 中的提示信息。



```
SPP_CA BlueTooth Demo
The SPP_CA is already in paired, Please re-power the board!
```

图 2-8 模块处于配对状态下再复位 STM32 程序时的输出

出现该提示的原因是 SPP-CA 模块配对后，它会把从串口接收到的所有内容都直接通过蓝牙传输出去，因此它会把 STM32 程序发送给它的 AT 命令当作普通字符串外发，而不对命令作出响应。解决方法是通过手机端控制蓝牙断开连接，或者直接重新给开发板（蓝牙模块）重新上电。

## 2.2.2 电脑与安卓手机进行蓝牙通讯

使用电脑与安卓手机进行蓝牙通讯的测试方式比较方便，不需要开发板及下载程序，只要用 USB 转串口 TTL 线连接起来即可。

### 1. 准备测试环境

使用 USB 转串口 TTL 线连接模块，接上电源和地并把 SPP-CA 模块与 USB 转串口的 TXD、RXD 线交叉相连即可，模块的 NC 和 INT 引脚不需要连接。



表 2-3 SPP-CA 模块与 USB 转串口 TTL 线的连接

SPP-CA 模块	USB 转串口 TTL 线
VCC	3.3 或 5V
GND	GND
TXD	RXD
RXD	TXD
NC 和 INT 引脚不需要连接	-

把 USB 转串口线接入到电脑，然后打开串口调试助手软件，选择对应的 COM 口，串口设置为 9600-N-8-1。

## 2. 正常运行的实验现象

首先，使用串口调试助手向 SPP-CA 模块发送“AT\r\n”命令测试模块是否正常连接，见图 2-4；



图 2-9 使用串口调试助手向模块发送 AT 命令测试

当 SPP-CA 模块处于非配对状态，并且连接正常的情况下，当它的串口接收到字符串“AT\r\n”时，它会返回字符串“OK”。

需要注意的是，SPP-CA 的所有命令都以“\r\n”结束，即在串口调试助手输入框里，输入完命令字符后，要确保在它的后面加上一个回车。

若该模块对“AT\r\n”命令返回响应“OK”，说明模块已正常工作并准备就绪，此时可以使用前面介绍的安卓手机客户端与模块连接并进行数据传输。若之前运行过 STM32 程序控制模块，它的名字会被改为“BT\_FIRE”，若并未修改过，则它的默认名字为“BT04-A”，手机客户端搜索到后使用默认配对码“1234”直接连接即可。



连接后, 串口调试助手会接收到模块返回的“+CONNECTING<<地址 CONNECTED”字符, 并且模块背面的 LED 指示灯常亮, 见图 2-6。

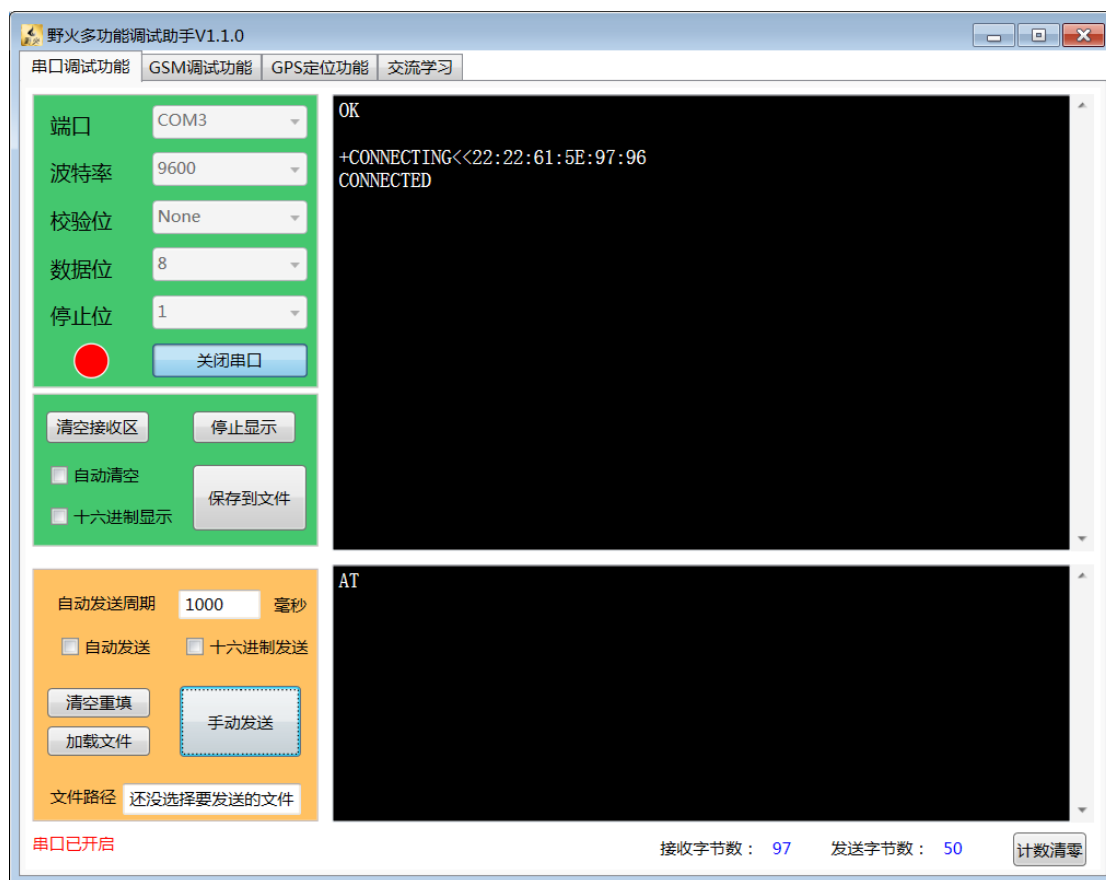


图 2-10 模块与手机端配对连接成功

连接成功后, 在串口调试助手输入的字符串会被传输到手机端, 手机端也可以往模块发送数据, 其内容会显示在电脑的串口调试助手中, 见图 2-11。

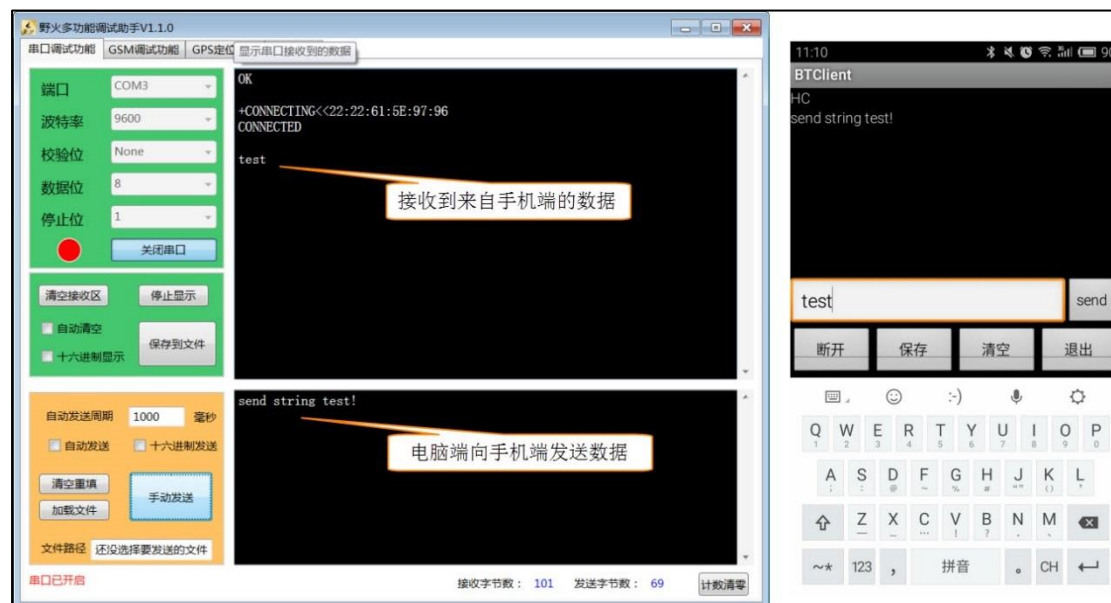


图 2-11 电脑端与手机端互相发送数据



模块处于配对状态时,所有字符串都会直接发送通过蓝牙向外转发,若想使用 AT 命令控制模块时,必须要先断开连接(可通过手机端断开或直接给 SPP-CA 模块断电)。

### 3. 不正常运行时故障排查

若使用“ATr\n”命令测试模块没有返回“OK”,需要检查如下事项:

- ❑ 模块与 USB 转串口 TTL 线是否连接正常,注意它们的 TXD 与 RXD 要交叉连接;
- ❑ 确认串口调试助手打开的 COM 是 USB 转串口 TTL 线对应的端口,且波特率是 9600;
- ❑ 确认模块背面的指示灯是否处于闪烁状态,若指示灯不亮,检查模块的供电,若指示灯常亮,说明模块处于配对状态,给模块重新上电即可。

## 3. SPP-CA 注意事项及常见问题

### 1. AT 命令

关于控制 SPP-CA 模块的 AT 命令可以查询配套资料里的《SPP-CA 蓝牙模块 AT 指令集文档》。

部分命令与该文档稍有差异,如 AT+VERSION 命令的响应返回值,设置串口波特率时不支持 A、B、C 选项表示的 460800、921600 及 1382400 波特率,且成功更改模块的波特率配置后,控制端的串口波特率也要做相应的修改,才能继续发送命令。

当模块处于配对状态时,模块不会响应 AT 指令,它接收到的所有字符串都会通过蓝牙对外发送出去,若希望模块响应 AT 命令,必须断开它与其它模块的蓝牙连接(可通过与之连接的蓝牙断开或直接给本模块重新上电断开)。

### 2. 主机(MASTER)与从机(SLAVE)有什么区别?

蓝牙模块只有主机才能主动发起扫描、配对及连接的操作。连接后主机和从机都可以主动互相发送数据,通信都是全双工的,也就是收发可同时进行。

SPP-CA 模块仅支持从机模式,所以它没有扫描、连接等控制命令。由于只能工作于从机模式,两个 SPP-CA 模块之间不能建立连接,若需要支持主机模式的蓝牙模块,可以选用我们的 BH-HC05 模块。

### 3. 关于模块上的 LED 灯

模块上的 LED 是配对状态指示灯。

- ❑ 刚上电时模块快闪一次;
- ❑ 可配对状态:指示灯均匀慢速闪烁(800ms 亮,800ms 灭)
- ❑ 配对状态:指示灯常亮。





## 4. 关于模块上的 INT 和 NC 引脚

模块上的 NC 引脚是我们为了兼容其它产品的 PCB 预留的位置，它本身没有与 SPP-CA 的任何有效引脚相连，使用时直接忽略即可。

模块上的 INT 引脚用于指示配对状态：

- ☐ 高电平：模块处于配对状态；
- ☐ 低电平：模块未处于配对状态。



## 4. 配套程序说明

BH-SPP-CA 模块一共配套了 2 个例程, 用户可根据需求选择相应的程序来学习。

例程所在目录: “开发板配套例程\霸道版本[或指南者、挑战者版本]”。

程序	说明
1.SPP-CA 测试程序(带液晶)	SPP-CA 模块测试程序, 带液晶状态显示, 方便测试
2. SPP-CA 测试程序(不带液晶)	SPP-CA 模块测试程序, 不带液晶状态显示, 可使用串口调试助手查看测试信息。由于这个程序不包含液晶显示驱动, 十分方便移植

### 4.1 SPP-CA 驱动

配套的两个程序除了液晶信息输出, 其它部分是完全一样的, 这里我们以 F103 霸道开发板 “1. SPP-CA 测试程序(带液晶)” 来讲解 (F429 挑战者程序中的液晶显示函数稍有区别, 其它内容一致)。

SPP-CA 的驱动包含了 `bsp_usart_blt.c` 及 `bsp_spp_ca.c` 文件中。`bsp_usart_blt.c` 文件中主要是配置控制蓝牙模块的 `usart` 工作模式, 以及处理从 SPP-CA 模块处接收到的数据, 进行基本处理。`bsp_spp_ca.c` 文件包含命令发送、设备管理等 SPP-CA 功能函数。

`bsp_usart_blt.c` 程序中控制 STM32 使用 USART 与 SPP-CA 模块通讯, 使用 USART 接收中断模式来处理 SPP-CA 发送给 STM32 的数据。它在中断服务函数中把接收到的数据存储到一个静态缓冲区中, 核心代码见代码清单 4-1。

代码清单 4-1 USART2 中断接收缓冲 核心代码 (位于 `bsp_usart_blt.c` 文件)

```
1 //中断缓存串口数据
2 #define UART_BUFF_SIZE      1024
3 volatile uint16_t uart_p = 0;
4 uint8_t      uart_buff[UART_BUFF_SIZE];
5
6 void bsp_USART_Process(void)
7 {
8     if (uart_p < UART_BUFF_SIZE) {
9         if (USART_GetITStatus(BLT_USARTx, USART_IT_RXNE) != RESET) {
10             uart_buff[uart_p] = USART_ReceiveData(USART2);
11             uart_p++;
12         }
13     } else {
14         USART_ClearITPendingBit(BLT_USARTx, USART_IT_RXNE);
15         clean_rebuff();
16     }
17 }
18
19
20
21 //获取接收到的数据和长度
22 char *get_rebuff(uint16_t *len)
23 {
24     *len = uart_p;
25     return (char *)&uart_buff;
26 }
```



```
27
28 //清空缓冲区
29 void clean_rebuff(void)
30 {
31
32     uint16_t i=UART_BUFF_SIZE+1;
33     uart_p = 0;
34     while (i)
35         uart_buff[--i]=0;
36
37 }
38
```

其中的 **bsp\_USART\_Process** 函数直接在 USART 的接收中断服务函数中调用, 把每个接收到的字节数据都存储在静态变量 `uart_buff` 中, 并用 `uart_p` 表示接收到的数据长度。**get\_rebuff** 函数则用于返回 `uart_buff` 的指针及其长度, 在需要处理接收数据的时候, 使用该函数获取 `uart_buff` 缓冲的数据。**clean\_rebuff** 函数则用于清空 `uart_buff` 的数据, 一般在处理完缓冲数据后调用。

`bsp_spp_ca.c` 文件的核心函数为 `SPP_CA_Send_CMD`、`SPP_CA_Get_CMD` 及 `SPP_CA_SendString` 函数, 见代码清单 4-2。这些个函数分别用于向 SPP-CA 模块发送命令及蓝牙透传数据。

代码清单 4-2 SPP-CA 驱动核心代码 (位于 `bsp_spp_ca.c` 文件)

```
1 /**
2  * @brief 向 SPP_CA 模块发送命令并检查 OK。只适用于具有 OK 应答的命令
3  * @param cmd:命令的完整字符串, 需要加\r\n。
4  * @param clean 命令结束后是否清除接收缓冲区, 1 清除, 0 不清除
5  * @template 复位命令: SPP_CA_Send_CMD("AT+RESET\r\n",1);
6  * @retval 0, 设置成功; 其他, 设置失败。
7  */
8 uint8_t SPP_CA_Send_CMD(char* cmd, uint8_t clean)
9 {
10     uint8_t retry=5;
11     uint8_t i,result=1;
12
13     while (retry--) {
14         delay_ms(10);
15         Usart_SendString(SPP_CA_USART, (uint8_t *)cmd);
16
17         for (i=0; i<20; i++) {
18             uint16_t len;
19             char * redata;
20
21             delay_ms(10);
22
23             redata = get_rebuff(&len);
24             if (len>0) {
25                 if (redata[0]!=0) {
26                     SPP_CA_DEBUG("send CMD: %s",cmd);
27
28                     SPP_CA_DEBUG("receive %s",redata);
29                 }
30                 if (strstr(redata,"OK")) {
31
32                     if (clean==1)
33                         clean_rebuff();
34                     return 0;
35                 } else {
36                     //clean_rebuff();
37                 }
38             } else {
```



```
39         delay_ms(100);
40     }
41 }
42     SPP_CA_DEBUG("SPP_CA send CMD fail %d times",retry);
43 }
44
45     SPP_CA_DEBUG("SPP_CA send CMD fail ");
46
47     if (clean==1)
48         clean_rebuff();
49
50     return result ;
51 }
52 }
53
54
55
56 /**
57  * @brief   向 SPP_CA 模块发送获取信息的命令
58  * @param   respon_string:存储接收到的响应
59  * @note    SPP_CA 模块的获取信息命令的正常返回都不带 OK:
60           如 : AT+VERSION
61           返回: +VERSION+=BOLUTEK Firmware V2.2, Bluetooth V2.1
62  * @retval  0,设置成功;其他,设置失败.
63  */
64 uint8_t SPP_CA_Get_CMD(char* cmd,char* respon_string)
65 {
66     uint8_t retry=5;
67     uint8_t i,result=1;
68     char check_parten[30];
69
70     char *p;
71     char cmd_temp[30];
72
73     strcpy(cmd_temp,cmd);
74
75     while (retry--){
76         delay_ms(10);
77         Usart_SendString(SPP_CA_USART,(uint8_t *)cmd);
78
79         for (i=0; i<20; i++){
80             uint16_t len;
81             char * redata;
82
83             delay_ms(10);
84
85             redata = get_rebuff(&len);
86             if (len>0 ) {
87                 if (redata[0] != 0) {
88                     SPP_CA_DEBUG("send CMD: %s\r\n",cmd);
89
90                     SPP_CA_DEBUG("receive %s",redata);
91                 }
92                 {
93                     //删除命令字符串的回车字符
94                     p = strstr(cmd_temp,"\r\n");
95                     if (p != NULL)
96                         *p = '\0';
97
98                     //组成的 check_parten 用于检查命令返回的信息
99                     //cmd_temp 地址+2 是为了去除命令前的"AT"，响应不带"AT"字符
100                     sprintf(check_parten,"%s=",cmd_temp+2);
101
102                     SPP_CA_DEBUG("check patern :%s",check_parten);
103
104                     //正常的命令响应以 check_parten 开头，以"\r\n"结尾
```



```
105         if (strstr(redata,check_parten) && strstr(redata,"\r\n")) {
106             //以接收完整的命令响应,拷贝、退出
107             strcpy(respon_string,redata);
108             clean_rebuff();
109             return 0;
110         } else {
111             //clean_rebuff();
112         }
113     }
114     } else {
115         delay_ms(100);
116     }
117 }
118 SPP_CA_DEBUG("SPP_CA send CMD fail %d times",retry);
119 }
120 SPP_CA_DEBUG("SPP_CA send CMD fail ");
121 clean_rebuff();
122 return result ;
123 }
124
125
126 /**
127  * @brief  使用 SPP_CA 透传字符串数据
128  * @param  str,要传输的字符串
129  * @retval 无
130  */
131 void SPP_CA_SendString(char* str)
132 {
133     Uart_SendString(SPP_CA_USART,(uint8_t *)str);
134 }
135
136
```

为方便使用,代码中把不同类型的命令分成两种情况来发送,其中 SPP\_CA\_Send\_CMD 用于发送 SPP-CA 的设置命令,模块接收到这些命令后,会作出相应的配置,并且返回带“OK”的响应, SPP\_CA\_Send\_CMD 函数发送命令后,会对模块的响应进行判断,若响应中包含“OK”,则返回 0 表示命令控制成功,否则将多次重发命令进行尝试。

例如,“AT\r\n”命令正常的响应为“OK”,那么使用 SPP\_CA\_Send\_CMD(“AT\r\n”,1) 语句向模块发送命令,若函数返回 0,即表示接收到模块返回的“OK”响应,可以直接利用该语句来检测模块是否连接正常。

SPP\_CA\_Get\_CMD 函数则用于发送获取 SPP-CA 信息的命令,模块接收到这类命令后,会通过串口返回对应的信息,这些信息中不包含“OK”字符,所以不使用 SPP\_CA\_Send\_CMD 函数来发送, SPP\_CA\_Get\_CMD 函数通过判断响应的头部字符串及结尾的回车来判断是否接收到正常的响应,若不正常,则重发多次命令。

例如:“AT+VERSION\r\n”命令的响应为“+VERSION=BOLUTEK Firmware V2.2,Bluetooth V2.1\r\n”,所以该函数判断接收到的字符串中是否包含“+VERSION”及“\r\n”字符来判断响应的正确性。应用时使用 SPP\_CA\_Get\_CMD(“AT+VERSION\r\n”,respon\_string)的形式来获取信息,函数的返回值表示响应是否正常, respon\_string 参数会存储响应正常时的响应字符串。

某些 SPP-CA 的命令比较特殊,其本身包含有设置状态及获取信息的功能。如设置模块名称的命令“AT+NAME”,当要设置模块名称为 FIRE 时,使用“AT+NAME<名称>”的形式,它的正常响应是“OK”,所以应用时使用 SPP\_CA\_Send\_CMD(“AT+NAMEFIRE\r\n”)来



发送设置并检查;而当要获取模块当前的名称时,它的正常响应是“+NAME=名称”,所以此时应使用形如 `SPP_CA_Get_CMD(“AT+NAME\r\n”,respon_string)` 的语句来接收模块返回的包含名称的响应字符串。

当模块处于配对状态时,不会响应 AT 命令,这时可直接使用 `SPP_CA_SendString` 函数通过蓝牙对外发送数据,该函数就是一个串口发送代码的接口,`SPP-CA` 模块会把它串口接收到的所有数据转发出去。

## 4.2 main 函数执行流程

`main` 函数代码较长,这里就不帖代码了,请读者打开工程直接对照阅读。了解 `main` 函数的执行流程有助于读者掌握 `SPP-CA` 模块的初始化及各种功能的使用方法。它具体的执行流程如下:

- ❑ 初始化 `Systick` 定时器,每 1ms 产生一次定时中断。
- ❑ 初始化液晶、`USART1`、板子上的 LED 灯、按键。
- ❑ 调用 `SPP_CA_Init` 函数初始化 `SPP-CA`,若检测到 `SPP-CA` 模块,则程序继续正常往下执行。
- ❑ 调用 `SPP_CA_Send_CMD` 函数,对模块进行复位(RESET)、恢复出厂设置(ORGL)。
- ❑ 调用 `SPP_CA_Get_CMD` 函数,演示如何向 `SPP-CA` 模块发送各种命令,获取信息。默认这部分是演示是不在液晶屏上显示出来的,用串口调试助手可查看命令执行结果。
- ❑ 使用 `SPP_CA_Send_CMD` 执行命令把模块的名字设置为“BT\_FIRE”。

接着,在 `while` 循环里定时执行各种操作:

- ❑ 若模块未处于连接状态,每 5s 向串口调试助手发送提示信息。
- ❑ 若模块处于连接状态,定时检查接收缓冲区,判断接收到的字符串是否为“AT+LED1=ON”和“AT+LED1=OFF”命令,若是则进行相应的操作,若不是则把接收缓冲区的第一行字符串显示出来。在这里用户可参照该范例自定义其它的命令。
- ❑ 若模块处于连接状态,则每隔几秒向配对的蓝牙设备发送一段字符串,用于演示。
- ❑ 若 `KEY2` 被长按下,则更换 `SPP-CA` 模块的名字,防止与周边的蓝牙设备名重复(其实名字重复也没问题,只是方便用户识别而已)。





## 5. 产品更新及售后支持

秉火的产品资料更新会第一时间发布到论坛: <http://www.firebbs.com>

购买秉火产品请到秉火官方淘宝店铺: <https://fire-stm32.taobao.com>

在学习或使用秉火产品时遇到问题可在论坛发帖子与我们交流。