HC05 配套例程使用说明

YH-HC05 模块一共配套了 2 个 HC05AT 指令测试例程，用户可根据需求选择相应的程序来学习。

例程所在目录：6-模块配套资料\蓝牙\野火【蓝牙_HC05】模块资料\2.开发板配套例程

表 1 HC05 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	接 3.3V 或 5V	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	KEY	模式引脚，悬空时默认为低电平 高电平时模块进入 AT 命令模式 低电平时模块为串口透传模式	PB14	PB11	PA5
6	INT	配对状态输出 配对状态时输出为高电平 未配对时输出为低电平	PB13	PB10	PA7

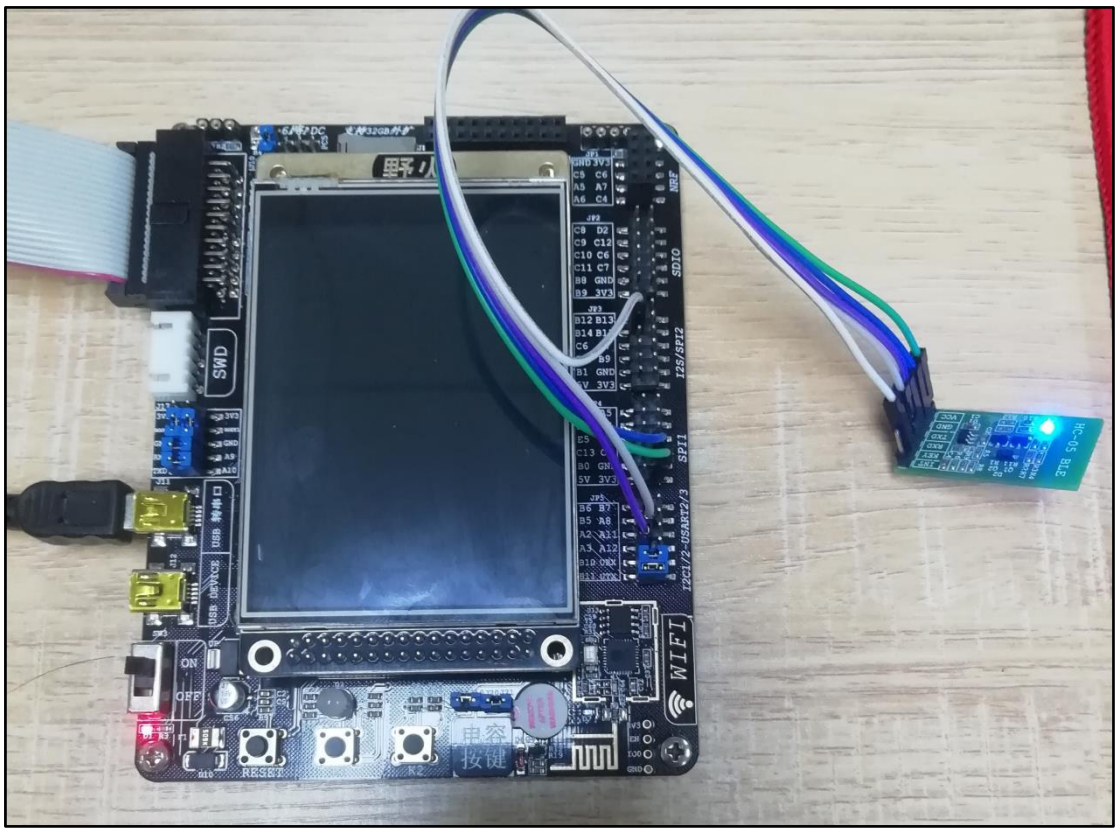

图 1 蓝牙模块与指南者开发板连接图

1. HC05AT 指令测试程序

测试步骤：

（1）将蓝牙模块和开发板使用杜邦线连接起来；

（2）将程序使用下载器或者是串口下载到开发板中；

（3）打开串口调试助手，波特率设置为 38400，并按下开发板复位键向串口调试助手发送“AT”接收窗口中收到蓝牙模块的回复的“OK”代表电脑串口调试助手与蓝牙模块的通信正常，（注意：AT 后面一定要加上回车并且蓝牙模块此时不要连接手机）其他 AT 指令请详见模块配套资料中官方文档文件夹里面的《AT 指令说明》。

图三 开发板向串口输出的信息

（4）用串口调试助手或者蓝牙调试助手发送：“RED_LED”,每发送一次“RED_LED”开发板上面的红灯的状态就会取反一次。

（5）如果下载的是带液晶的版本，会在屏幕上显示接收到串口调试助手和蓝牙模块发的信息。

（6）配套程序讲解

（7）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 指南者开发板“1.HC05AT 指令测试(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）。程序实现了使用串口调试助手发送 AT 指令，配置蓝牙模块的功能。基本思路如下：STM32 的 USART1 从电脑串口调试助手接收数据，接收完数据之后转发到串口 USART2，当 USART2 从蓝牙模块接收完数据之后转发到串口 USART1 上面，就可以使用串口调试助手发送 AT 指令配置蓝牙模块了。

（8） HC05 的驱动包含了 bsp_usart_blt.c 及 bsp_hc05.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 HC05 模块处接收到的数据，进行基本处理。 bsp_hc05.c 文件包含命令发送、设备管理等 HC05 功能函数。

（9） bsp_usart_blt.c 程序中控制 STM32 使用 USART 与 HC05 模块通讯，使用 USART 接收中断模式来处理 HC05 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
2 //中断缓存串口数据
3 ReceiveData BLT_USART_ReceiveData;
4 void bsp_USART_Process(void)
5 {
6    uint8_t ucCh;
```

```
    if (USART_GetITStatus(BLT_USARTx, USART_IT_RXNE) != RESET) {
        ucCh = USART_ReceiveData(BLT_USARTx);
        if (BLT_USART_ReceiveData.datanum < UART_BUFF_SIZE) {
            if ((ucCh != 0x0a) && (ucCh != 0x0d)) {

          BLT_USART_ReceiveData.uart_buff[BLT_USART_ReceiveData.datanum] = ucCh;
          //不接收换行回车
                BLT_USART_ReceiveData.datanum++;
            }
        }
    }
    if (USART_GetITStatus( BLT_USARTx, USART_IT_IDLE ) == SET )
{                                   //数据帧接收完毕
        BLT_USART_ReceiveData.receive_data_flag = 1;
        USART_ReceiveData( BLT_USARTx );
//由软件序列清除中断标志位(先读 USART_SR，然后读 USART_DR)
    }
}

//获取接收到的数据和长度
char *get_rebuff(uint16_t *len)
{
    *len = BLT_USART_ReceiveData.datanum;
    return (char *)&BLT_USART_ReceiveData.uart_buff;
}

//清空缓冲区
void clean_rebuff(void)
{

    uint16_t i=UART_BUFF_SIZE+1;
    BLT_USART_ReceiveData.datanum = 0;
    BLT_USART_ReceiveData.receive_data_flag = 0;
    while (i)
        BLT_USART_ReceiveData.uart_buff[--i]=0;
}

```

（10）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 BLT_USART_ReceiveData.uart_buff 中，并用 BLT_USART_ReceiveData.datanum 表示接收到的数据长度。当接收完一数据后，单片机会进入串口空闲中断，此时将 BLT_USART_ReceiveData.receive_data_flag 标志置位，代表一帧数据接收完成。get_rebuff 函数则用于返回 BLT_USART_ReceiveData.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 BLT_USART_ReceiveData.uart_buff 缓冲的数据。 clean_rebuff 函数则用于清空 BLT_USART_ReceiveData.uart_buff 的数据，一般在处理完缓冲数据后调用。

（11）USART1 接收处理流程如下代码 USART1 接收处理流程与 USART2 接收处理流程类似，这里不再赘述。

代码清单 4-2 USART1 中断接收缓冲 核心代码（位于 stm32f10x_it.c 文件）

```
ReceiveData DEBUG_USART_ReceiveData;
// 串口中断服务函数
void DEBUG_USART_IRQHandler(void)
{
    uint8_t ucCh;
    if (USART_GetITStatus(DEBUG_USARTx, USART_IT_RXNE) != RESET) {
        ucCh = USART_ReceiveData(DEBUG_USARTx);
        if (DEBUG_USART_ReceiveData.datanum < UART_BUFF_SIZE) {
            if ((ucCh != 0x0a) && (ucCh != 0x0d)) {

          DEBUG_USART_ReceiveData.uart_buff[DEBUG_USART_ReceiveData.datanum] =
          ucCh;                 //不接收换行回车
                DEBUG_USART_ReceiveData.datanum++;
```

```
12              }
13          }
14      }
15      if (USART_GetITStatus( DEBUG_USARTx, USART_IT_IDLE ) == SET )
        {                                       //数据帧接收完毕
16          DEBUG_USART_ReceiveData.receive_data_flag = 1;
17          USART_ReceiveData( DEBUG_USARTx );
                    //由软件序列清除中断标志位(先读 USART_SR，然后读 USART_DR)
18      }
19  }
```

（12）main 函数实现了 USART1 与 USART2 串口数据互相转发的功能。当串口 1 收到数据之后，DEBUG_USART_ReceiveData.receive_data_flag 标志会置位，之后会用 strstr 函数判断数据是不是以“AT”开头的，如果是，就把蓝牙模块 KEY 引脚电平置高，使蓝牙模块进入 AT 指令响应模式，并把数据发送到蓝牙模块上面；如果发送了字符串“RED_LED”就会将开发板红灯的状态取反一次。其他数据就会将蓝牙模块 KEY 引脚电平置低，使蓝牙模块进入透传模式，此时如果蓝牙模块连接了手机，模块就会将数据发送至手机上，然后在清屏，并将接收到的数据显示在屏幕上面。下面从 USART2 接收的数据处理过程同上。

代码清单 4-3 main 函数数据处理 核心代码（位于 main.c 文件）

```
1  while (1)
2  {
3      if (DEBUG_USART_ReceiveData.receive_data_flag == 1) {
4          DEBUG_USART_ReceiveData.uart_buff[DEBUG_USART_ReceiveData.datanum] =
                                                                          0;
5          if (strstr((char *)DEBUG_USART_ReceiveData.uart_buff,"AT")) { //如果
               数据是以 AT 开头的，就把 KEY 置高，设置蓝牙模块
6              BLT_KEY_LOW;
7              delay_ms(20);
8
         Usart_SendStr_length(BLT_USARTx,DEBUG_USART_ReceiveData.uart_buff,DE
         BUG_USART_ReceiveData.datanum);
9              Usart_SendStr_length(BLT_USARTx,"\r\n",2);
10             BLT_KEY_HIGHT;
11         }else if (strstr((char*)DEBUG_USART_ReceiveData.uart_buff,"RED_LED"))
           {
12             LED1_TOGGLE;//将红灯状态取反
13         } else {
14             BLT_KEY_LOW;
15
         Usart_SendStr_length(BLT_USARTx,DEBUG_USART_ReceiveData.uart_buff,DE
         BUG_USART_ReceiveData.datanum);
16         }
17         LCD_ClearLine(LINE(5));//清屏
18         LCD_ClearLine(LINE(6));
19         LCD_ClearLine(LINE(7));
20         LCD_ClearLine(LINE(8));
21
           ILI9341_DispStringLine_EN( (LINE(5)),(char*)DEBUG_USART_ReceiveDat
           a.uart_buff );//显示从 USART1 接收到的数据
22         DEBUG_USART_ReceiveData.receive_data_flag = 0;    //接收数据标志清零
23         DEBUG_USART_ReceiveData.datanum = 0;
24     }
25     if (BLT_USART_ReceiveData.receive_data_flag == 1) {
26         BLT_USART_ReceiveData.uart_buff[BLT_USART_ReceiveData.datanum] = 0;
27         if (strstr((char *)BLT_USART_ReceiveData.uart_buff,"RED_LED")) { //
           在这里可以自己定义想要接收的字符串然后处理
28             LED1_TOGGLE; //这里接收到串口调试助手或者是手机发来的 “RED_LED”就会把板
           子上面的红灯取反一次
29         } else {
```

```
30
    Usart_SendStr_length(DEBUG_USARTx,BLT_USART_ReceiveData.uart_buff,BLT_U
    SART_ReceiveData.datanum);
31          Usart_SendStr_length(DEBUG_USARTx,"\r\n",2);
32      }
33      LCD_ClearLine(LINE(10));
34      LCD_ClearLine(LINE(11));
35      LCD_ClearLine(LINE(12));
36      LCD_ClearLine(LINE(13));
37
        ILI9341_DispStringLine_EN( (LINE(10)),(char*)BLT_USART_ReceiveData
        .uart_buff );
38      clean_rebuff();
39    }
40 }
```