开源共享　　携手共进

深圳普中科技有限公司

官方网站：www.prechin.cn

技术论坛：www.prechin.net

技术 QQ：2489019400

咨询电话：0755-61139052

PZ-HC05 蓝牙串口模块开发手册

本手册我们将向大家介绍 PZ-HC 蓝牙模块及其在普中 STM32F1 开发板上的使用。本手册我们将使用 PZ-HC05 蓝牙串口模块实现蓝牙串口通信， 并和手机连接，实现手机控制开发板。本章分为如下几部分内容：

1 PZ-HC05 蓝牙串口模块介绍

2 硬件设计

3 软件设计

4 实验现象

1 PZ-HC05 蓝牙串口模块介绍

1.1 特性参数

PZ-HC05 是一款高性能的主从一体蓝牙串口模块，可以同各种带蓝牙功能的电脑、蓝牙主机、手机、 PDA、 PSP 等智能终端配对，该模块支持非常宽的波特率范围：4800~1382400，并且兼容 5V 或 3.3V 单片机系统，可以很方便与您的产品进行连接，使用非常灵活、方便。该模块各参数如下所示：

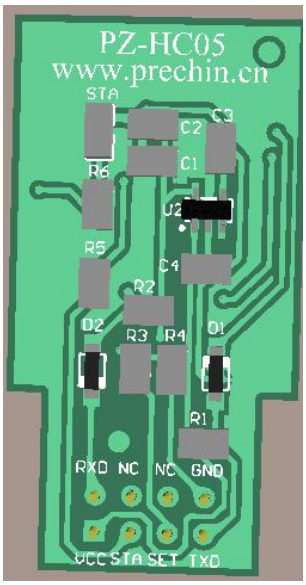
项目	说明
接口特性	TTL，兼容 3.3V/5V 单片机系统
支持波特率	4800、 9600（默认）、 19200、 38400、 57600、115200、 230400、 460800、 921600、1382400
其他特性	主从一体，指令切换，默认为从机。带状态指示灯，带配对状态输出
通信距离	10M（空旷地）
工作温度	-25℃~75℃
模块尺寸	20.37mm*38.84mm
工作电压	DC3.3V~5.0V
工作电流	配对中： 30~40mA； 配对完毕未通信： 1~8mA；通信中： 5~20mA
Voh	3.3V@VCC=3.3V 3.7V@VCC=5.0V
Vol	0.4V(Max)
Vih	2.4V(Min)
Vil	0.4V(Max)

注意：通信中的电流和你的串口通信频繁程度成正比，如果单位时间内的数据通信量越大，电流则越高；反之，单位时间内的数据通信量越小，电流则越低（接近配对未通信的电流）。

1.2 模块说明

1.2.1 模块引脚说明

PZ-HC05 模块非常小巧（20.37mm*38.84mm），模块通过 1 个 2*4 的间距为 2.54 的排针与外部连接，模块外观如图所示：



从图中可以看到，从右到左依次为模块引出的 8 个脚，其中 NC 代表未与模块相连即待扩展脚，可用的管脚只有 6 个，各引脚的详细描述如图所示：

管脚名称	功能说明
VCC	电源（ 3.3V~5.0V）
GND	地
TXD	模块串口发送脚（ TTL 电平，不能直接接 RS232 电平!），可接单片机的 RXD
RXD	模块串口接收脚（ TTL 电平，不能直接接 RS232 电平!），可接单片机的 TXD
SET	用于进入 AT 状态；高电平有效（悬空默认为低电平）
STA	配对状态输出；配对成功输出高电平，未配对则输出低电平

另外，模块自带了一个状态指示灯： STA。该灯有 3 种状态，分别为：

1,在模块上电的同时(也可以是之前),将 SET 脚 设置为高电平(接 VCC),此时 STA 慢闪（ 1 秒亮 1 次），模块进入 AT 状态，且此时波特率固定为 38400。

2，在模块上电的时候，将 SET 脚 悬空或接 GND，此时 STA 快闪（ 1 秒 2 次），表示模块进入可配对状态。如果此时将 SET 脚 再拉高，模块也会进入 AT 状态，但是 STA 依旧保持快闪。

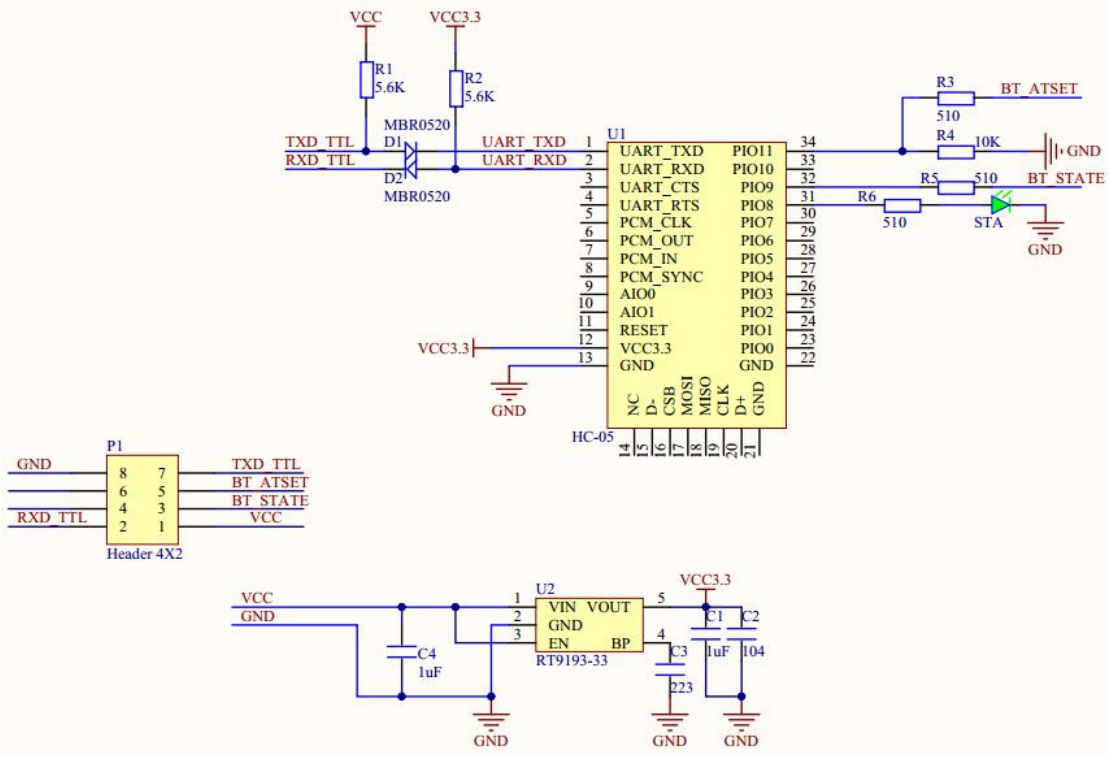
3，模块配对成功，此时 STA 双闪（一次闪 2 下， 2 秒闪一次）。

有了 STA 指示灯，我们就可以很方便的判断模块的当前状态，方便大家使用。

PZ-HC05 蓝牙串口模块所有功能都是通过 AT 指令集控制，比较简单，该部分使用的详细信息，请参考 HC05 蓝牙指令集.pdf。

使用 PZ-HC05 蓝牙串口模块，任何单片机（ 3.3V/5V 电源）都可以很方便

的实现蓝牙通信，从而与包括电脑、手机、平板电脑等各种带蓝牙的设备连接。PZ-HC05 蓝牙串口模块的原理图如图所示：



1.2.2 模块使用说明

（1）AT 指令说明及测试

PZ-HC05 蓝牙串口模块所有功能都是通过 AT 指令集控制， 这我们仅介绍用户常用的几个 AT 指令，详细的指令集，请参考 HC05 蓝牙指令集.pdf 这个文档。

1.进入 AT 状态

有 2 种方法使模块进入 AT 指令状态：

①上电同时/上电之前将 SET 脚设置为 VCC，上电后，模块即进入 AT 指令状态。

②模块上电后，通过将 SET 脚接 VCC，使模块进入 AT 状态。

方法 1（ 推荐）进入 AT 状态后，模块的波特率为： 38400（ 8 位数据位，1 位停止位）。方法 2 进入 AT 状态后，模块波特率和通信波特率一致。

2.指令结构

模块的指令结构为： AT+<CMD><=PARAM>，其中 CMD（指令）和 PARAM（参数）都是可选的，不过切记在发送末尾添加回车符（ \r\n），否则模块不响应，比如我们要查看模块的版本：

串口发送： AT+VERSION?\r\n

模块回应： +VERSION:2.0-20100601

OK

3.常用指令说明及测试

注意，这里我们通过将模块连接电脑串口，来测试模块的指令，注意模块不能和 RS232 串口直连。

①修改模块主从指令

AT+ROLE=0 或 1,该指令来设置模块为从机或主机,并且可以通过 AT+ROLE?来查看模块的主从状态，如图所示：

我们模块出厂默认设置为从机，所以发送 AT+ROLE?，得到的返回值为：+ROLE:0，发送 AT+ROLE=1，即可设置模块为主机，设置成功模块返回 OK 作为应答。注意串口调试助手要勾选发送新行，这样就会自动发送回车了。

②设置记忆指令

AT+CMODE=1， 该指令设置模块可以对任意地址的蓝牙模块进行配对， 模块

默认设置为该参数。AT+CMODE=0， 该指令设置模块为指定地址配对，如果先设置模块为任意地址，然后配对，接下去使用该指令，则模块会记忆最后一次配对的地址，下次上电会一直搜索该地址的模块，直到搜索到为止。

③修改通信波特率指令

AT+UART=<Param1>,<Param2>,<Param3>， 该指令用于设置串口波特率、停止位、校验位等。Param1 为波特率，可选范围为：4800、9600、19200、38400、57600、 115200、230400、 460800、 921600、 1382400； Param2 为停止位选择， 0 表示 1 位停止位， 1 表示 2 位停止位； Param3 为校验位选择， 0 表示没有校验位（ None）， 1 表示奇校验（ Odd），2 表示偶校验（ Even）。

比如我们发送： AT+UART=9600,0,0，则是设置通信波特率为 9600， 1 位停止位，没有校验位，这也是我们模块的默认设置。

④修改密码指令

AT+PSWD=<password>，该指令用于设置模块的配对密码， password 必须为 4 个字节长度。

⑤修改蓝牙模块名字

AT+NAME=<name>，该指令用于设置模块的名字， name 为你要设置的名字，必须为 ASCII 字符，且最长不能超过 32 个字符。模块默认的名字为 HC05。比如发送：AT+NAME=PZ-HC05，即可设置模块名字为“PZ-HC05”。

2 硬件设计

本实验使用到硬件资源如下：

（1）D1、D2 指示灯

（2）K_UP、K_DOWN 按键

（3）串口 1、串口 2

（4）TFTLCD 模块

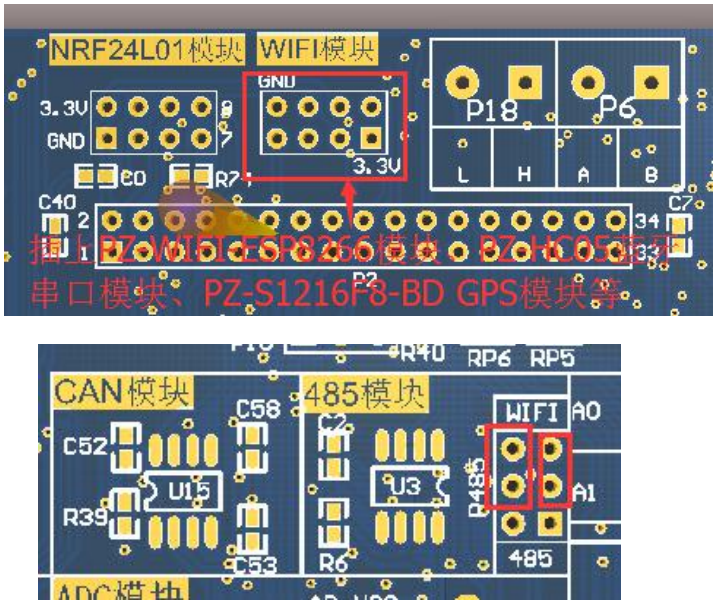
（5）PZ-HC05 蓝牙串口模块

前四部分电路在前面章节都介绍过，这里就不多说，前面我们介绍了该模块的接口管脚功能，下面我们来看下 PZ-HC05 蓝牙串口模块与开发板如何连接的。

我们 STM32 开发板上板载了一个 WIFI 接口（适用于 PZ-WIFI-ESP8266 模块、PZ-HC05 蓝牙串口模块、PZ-S1216F8-BD GPS 模块等），直接将该模块插上 WIFI 模块接口处即可，其内部管脚连接关系如图所示：

PZ-HC05蓝牙串口模块	普中STM32F1开发板（IO）
VCC	3.3V
GND	GND
RXD	PA2(TXD)
TXD	PA3(RXD)
SET	PG7
STA	PE6

注意：因为开发板上的 RS485 模块也是使用串口 2，所以使用 PZ-WIFI-ESP8266 模块、PZ-HC05 蓝牙串口模块、PZ-S1216F8-BD GPS 模块等需要将开发板 RS485 模块处的 P485 端子切换到 WIFI（丝印）处。如图所示：



如果是使用其他板子需要将模块连接到单片机管脚上的画，一定要注意模块的 TXD 要连接单片机的 RXD，模块的 RXD 要连接单片机的 TXD 管脚。

3 软件设计

本实验所实现的功能为：开机检测 PZ-HC05 蓝牙模块是否存在，如果检测不成功，则报错。检测成功之后，显示模块的主从状态，并显示模块是否处于连接状态， D1 指示灯闪烁提示程序运行正常。按 K_DOWN 按键，可以开启/关闭自动发送数据（通过蓝牙模块发送）；按 K_UP 按键可以切换模块的主从状态。

蓝牙模块接收到的数据，将直接显示在 LCD 上（仅支持 ASCII 字符显示）。结合手机端蓝牙软件(蓝牙串口助手 V0.16.apk)， 可以实现手机无线控制开发板(点亮和关闭 D2 指示灯)。

我们打开本实验工程，可以看到我们的工程 APP 列表中多了 usart2.c 和 hc05.c 源文件，以及头文件 usart2.h、 hc05.h。首先，我们来看 usart2.c 里面代码，如下：

```
#include "usart2.h"
#include "stdarg.h"
#include "stdio.h"
#include "string.h"
#include "time.h"


//串口接收缓存区
u8 USART2_RX_BUF[USART2_MAX_RECV_LEN];  // 接收缓冲，最大 USART3_MAX_RECV_LEN 个字节.
u8  USART2_TX_BUF[USART2_MAX_SEND_LEN]; // 发送缓冲，最大 USART3_MAX_SEND_LEN 字节


//通过判断接收连续 2 个字符之间的时间差不大于 10ms 来决定是不是一次连续的数据.
//如果 2 个字符接收间隔超过 10ms,则认为不是 1 次连续数据.也就是超过 10ms 没有接收到
//任何数据,则表示此次接收完毕.
//接收到的数据状态
//[15]:0,没有接收到数据;1,接收到了一批数据.
//[14:0]:接收到的数据长度
u16 USART2_RX_STA=0;
```

```
void USART2_Init(u32 bound)
{

   GPIO_InitTypeDef GPIO_InitStructure;
   USART_InitTypeDef USART_InitStructure;
   NVIC_InitTypeDef NVIC_InitStructure;

   RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2; //PA2  TXD
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;  //复用推挽输
出
   GPIO_Init(GPIOA, &GPIO_InitStructure);


   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;//PA3  RXD
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;//浮空输入
   GPIO_Init(GPIOA, &GPIO_InitStructure);//初始化 GPIOA3

   //Usart2 NVIC 配置
   NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=2 ;//抢占
优先级 0
   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;      //子优
先级 0
   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;         //IRQ
```

```
通道使能
        NVIC_Init(&NVIC_InitStructure); //根据指定的参数初始化 VIC 寄存
器


            //USART2 初始化设置
        USART_InitStructure.USART_BaudRate = bound;//串口波特率
        USART_InitStructure.USART_WordLength  =  USART_WordLength_8b;//
字长为 8 位数据格式
        USART_InitStructure.USART_StopBits = USART_StopBits_1;//一个停
止位
        USART_InitStructure.USART_Parity = USART_Parity_No;//无奇偶校验
位
        USART_InitStructure.USART_HardwareFlowControl                   =
USART_HardwareFlowControl_None;//无硬件数据流控制
        USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    //收发模式
        USART_Init(USART2, &USART_InitStructure); //初始化串口 2

        USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);//开启串口接受中
断

        USART_Cmd(USART2, ENABLE);                          //使能串口 2

        TIM7_Int_Init(99,7199);      //10ms 中断
        USART2_RX_STA=0;      //清零
        TIM_Cmd(TIM7,DISABLE);           //关闭定时器 7
    }

    void USART2_IRQHandler( void )
```

```
    {
        u8 res;
        if(USART_GetITStatus(USART2, USART_IT_RXNE) != RESET)//接收到数
据
        {
            res =USART_ReceiveData(USART2);
            if((USART2_RX_STA&(1<<15))==0)//接收完的一批数据,还没有被
处理,则不再接收其他数据
            {
                if(USART2_RX_STA<USART2_MAX_RECV_LEN) //还可以接收数据
                {
                    TIM_SetCounter(TIM7,0);//计数器清空
        //计数器清空
                    if(USART2_RX_STA==0)                //使能定时器 7 的
中断
                    {
                        TIM_Cmd(TIM7,ENABLE);//使能定时器 7
                    }
                    USART2_RX_BUF[USART2_RX_STA++]=res;    //记录接收到
的值
                }
                else
                {
                    USART2_RX_STA|=1<<15;               //强制标记接收完
成
                }
            }
        }
    }
```

```
//串口 2,printf 函数
//确保一次发送数据不超过 USART2_MAX_SEND_LEN 字节
void usart2_printf(char* fmt,...)
{
   u16 i,j;
   va_list ap;
   va_start(ap,fmt);
   vsprintf((char*)USART2_TX_BUF,fmt,ap);
   va_end(ap);
   i=strlen((const char*)USART2_TX_BUF);      //此次发送数据的长度
   for(j=0;j<i;j++)                           //循环发送数据
   {
      while(USART_GetFlagStatus(USART2,USART_FLAG_TC)==RESET); //
循环发送,直到发送完毕
      USART_SendData(USART2,USART2_TX_BUF[j]);
   }
}
```

这部分代码主要实现了串口 2 的初始化，以及实现了串口 2 的 printf 函数：usart2_printf，和串口 2 的接收处理。串口 2 的数据接收，采用了定时判断的方法，对于一次连续接收的数据，如果出现连续 10ms 没有接收到任何数据，则表示这次连续接收数据已经结束。

usart2.h 里面的代码我们就不在这里列出了，请大家参考对应源码。

下面我们来看下 hc05.c 里面的代码，如下：

```
#include "SysTick.h"
#include "usart.h"
#include "usart2.h"
#include "hc05.h"
```

```
#include "led.h"
#include "string.h"
#include "math.h"




//初始化 HC05 模块
//返回值:0,成功;1,失败.
u8 HC05_Init(void)
{
    u8 retry=10,t;
    u8 temp=1;

    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE|RCC_APB2Periph_GPIOG,ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;              //  端 口配置
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;         //上拉输入
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;          //IO 口速度为 50MHz
    GPIO_Init(GPIOE, &GPIO_InitStructure);


    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;              //  端 口
```

配置

```
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;          // 推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;          //IO口速度为50MHz
    GPIO_Init(GPIOG, &GPIO_InitStructure);

    HC05_ATSET=1;
    HC05_STATE=1;

    USART2_Init(9600);   //初始化串口2为:9600,波特率.

    while(retry--)
    {
        HC05_ATSET=1;                //KEY置高,进入AT模式
        delay_ms(10);
        usart2_printf("AT\r\n");     //发送AT测试指令
        HC05_ATSET=0;                //KEY拉低,退出AT模式
        for(t=0;t<10;t++)            //最长等待50ms,来接收HC05模块的回应
        {
            if(USART2_RX_STA&0X8000)break;
            delay_ms(5);
        }
        if(USART2_RX_STA&0X8000)//接收到一次数据了
        {
            temp=USART2_RX_STA&0X7FFF;  //得到数据长度
            USART2_RX_STA=0;
```

```
if(temp==4&&USART2_RX_BUF[0]=='O'&&USART2_RX_BUF[1]=='K')
            {
                temp=0;//接收到 OK 响应
                break;
            }
        }
    }
    if(retry==0)temp=1;  //检测失败
    return temp;
}
//获取 HC05 模块的角色
//返回值:0,从机;1,主机;0XFF,获取失败.
u8 HC05_Get_Role(void)
{
    u8 retry=0X0F;
    u8 temp,t;
    while(retry--)
    {
        HC05_ATSET=1;                //KEY 置高,进入 AT 模式
        delay_ms(10);
        usart2_printf("AT+ROLE?\r\n"); //查询角色
        for(t=0;t<20;t++)            //最长等待 200ms,来接收 HC05 模
块的回应
        {
            delay_ms(10);
            if(USART2_RX_STA&0X8000)break;
        }
        HC05_ATSET=0;                //KEY 拉低,退出 AT 模式
        if(USART2_RX_STA&0X8000) //接收到一次数据了
```

```
            {
                temp=USART2_RX_STA&0X7FFF;  //得到数据长度
                USART2_RX_STA=0;
                if(temp==13&&USART2_RX_BUF[0]=='+')//接收到正确的应答了
                {
                    temp=USART2_RX_BUF[6]-'0';//得到主从模式值
                    break;
                }
            }
        }
        if(retry==0)temp=0XFF;//查询失败.
        return temp;
    }
    //HC05 设置命令
    //此函数用于设置 HC05,适用于仅返回 OK 应答的 AT 指令
    //atstr:AT 指令串.比如:"AT+RESET"/"AT+UART=9600,0,0"/"AT+ROLE=0"
等字符串
    //返回值:0,设置成功;其他,设置失败.
    u8 HC05_Set_Cmd(u8* atstr)
    {
        u8 retry=0X0F;
        u8 temp,t;
        while(retry--)
        {
            HC05_ATSET=1;                   //KEY 置高,进入 AT 模式
            delay_ms(10);
            usart2_printf("%s\r\n",atstr); //发送 AT 字符串
            HC05_ATSET=0;                   //KEY 拉低,退出 AT 模式
            for(t=0;t<20;t++)               //最长等待 100ms,来接收 HC05 模
```

块的回应

```
            {
                if(USART2_RX_STA&0X8000)break;
                delay_ms(5);
            }
            if(USART2_RX_STA&0X8000)//接收到一次数据了
            {
                temp=USART2_RX_STA&0X7FFF;  //得到数据长度
                USART2_RX_STA=0;
                if(temp==4&&USART2_RX_BUF[0]=='O')//接收到正确的应答了
                {
                    temp=0;
                    break;
                }
            }
        }
        if(retry==0)temp=0XFF;//设置失败.
        return temp;
    }
```

此部分代码总共 3 个函数：

1，HC05_Init 函数，该函数用于初始化与 PZ-HC05 连接的 IO 口，并通过 AT 指令检测 PZ-HC05 蓝牙模块是否已经连接。

2，HC05_Get_Role 函数，该函数用于获取 PZ-HC05 蓝牙模块的主从状态，这里利用 AT+ROLE?指令获取模块的主从状态。

3，HC05_Set_Cmd 函数，该函数是一个 PZ-HC05 蓝牙模块的通用设置指令，通过调用该函数，可以方便的修改 PZ-HC05 蓝牙串口模块的各种设置。

hc05.h 里面的代码我们也不列出了，请大家参考对应源码。

最后我们打开 main.c 文件，里面的代码如下：

```
//显示 HC05 模块的主从状态
```

```
void HC05_Role_Show(void)
{
   if(HC05_Get_Role()==1)LCD_ShowString(10,140,200,16,16,"ROLE:Ma
ster");   //主机
   else LCD_ShowString(10,140,200,16,16,"ROLE:Slave ");
   //从机
}

//显示 HC05 模块的连接状态
void HC05_Sta_Show(void)
{
   if(HC05_STATE)LCD_ShowString(110,140,120,16,16,"STA:Connected
");       //连接成功
   else LCD_ShowString(110,140,120,16,16,"STA:Disconnect");
       //未连接
}


int main()
{
   u8 t=0;
   u8 key;
   u8 sendmask=0;
   u8 sendcnt=0;
   u8 sendbuf[20];
   u8 reclen=0;

   SysTick_Init(72);
   NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);  //中断优先级
```

分组 分 2 组

```
        LED_Init();
        KEY_Init();
        USART1_Init(9600);
        TFTLCD_Init();           //LCD 初始化


        FRONT_COLOR=RED;
        LCD_ShowString(10,10,tftlcd_data.width,tftlcd_data.height,16,"
PRECHIN");
        LCD_ShowString(10,30,tftlcd_data.width,tftlcd_data.height,16,"
www.prechin.com");
        LCD_ShowString(10,50,tftlcd_data.width,tftlcd_data.height,16,"
BT05 BlueTooth Test");
        delay_ms(1000);          //等待蓝牙模块上电稳定

        while(HC05_Init())       //初始化 HC05 模块
        {
            LCD_ShowString(10,90,200,16,16,"HC05 Error!    ");
            delay_ms(500);
            LCD_ShowString(10,90,200,16,16,"Please Check!!!");
            delay_ms(100);
        }
        LCD_ShowString(10,90,210,16,16,"K_UP:ROLE K_DOWN:SEND/STOP");
        LCD_ShowString(10,110,200,16,16,"HC05 Standby!");
        LCD_ShowString(10,160,200,16,16,"Send:");
        LCD_ShowString(10,180,200,16,16,"Receive:");

        FRONT_COLOR=BLUE;
```

```
HC05_Role_Show();
delay_ms(100);
USART2_RX_STA=0;
while(1)
{
    key=KEY_Scan(0);
    if(key==KEY_UP)                         //切换模块主从设置
    {
        key=HC05_Get_Role();
        if(key!=0XFF)
        {
            key=!key;                       //状态取反
            if(key==0)HC05_Set_Cmd("AT+ROLE=0");
            else HC05_Set_Cmd("AT+ROLE=1");
            HC05_Role_Show();
            HC05_Set_Cmd("AT+RESET");   //复位 HC05 模块
            delay_ms(200);
        }
    }
    else if(key==KEY_DOWN)
    {
        sendmask=!sendmask;             //发送/停止发送
        if(sendmask==0)LCD_Fill(10+40,160,240,160+16,WHITE);//
清除显示
    }
    else delay_ms(10);
    if(t==50)
    {
        if(sendmask)                    //定时发送
```

```
            {
                sprintf((char*)sendbuf,"PREHICN
HC05 %d\r\n",sendcnt);
                LCD_ShowString(10+40,160,200,16,16,sendbuf);  //显示
发送数据
                usart2_printf("PREHICN HC05 %d\r\n",sendcnt);//发送
到蓝牙模块
                sendcnt++;
                if(sendcnt>99)sendcnt=0;
            }
            HC05_Sta_Show();
            t=0;
            led1=!led1;
        }
        if(USART2_RX_STA&0X8000)        //接收到一次数据了
        {
            LCD_Fill(10,200,240,320,WHITE);//清除显示
            reclen=USART2_RX_STA&0X7FFF;    //得到数据长度
            USART2_RX_BUF[reclen]='\0';     //加入结束符
            printf("reclen=%d\r\n",reclen);
            printf("USART2_RX_BUF=%s\r\n",USART2_RX_BUF);
            if(reclen==10||reclen==11)      //控制 D2 检测
            {
                if(strcmp((const           char*)USART2_RX_BUF,"+LED2
ON\r\n")==0)led2=0;//打开 LED2
                if(strcmp((const           char*)USART2_RX_BUF,"+LED2
OFF\r\n")==0)led2=1;//关闭 LED2
            }
            LCD_ShowString(10,200,209,119,16,USART2_RX_BUF);// 显 示
```

接收到的数据

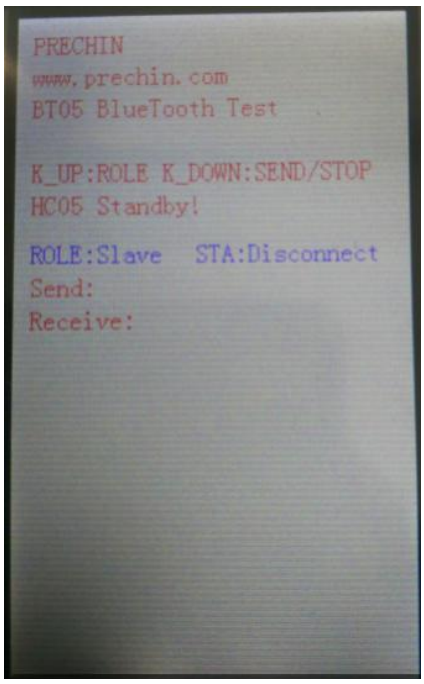
```
                USART2_RX_STA=0;
            }

            t++;
        }
    }
```

此部分代码比较简单，主要是调用前面编写的函数，加上额外的显示控制等，这里我们就不多说，大家可以查看源码。

4 实验现象

将工程程序编译下载到开发板内并且将模块连接到开发板上，注意对于 STM32F1 开发板要将 RS485 模块的端子短接到 485 测，这个在前面已经说过。可以看到 TFTLCD 上显示如下界面：



可以看到，此时模块的状态是从机（ Slave），未连接（ Disconnect）。发送和接收区都没有数据，同时蓝牙模块的 STA 指示灯快闪（ 1 秒 2 次），表示模块进入可配对状态，目前尚未连接。

下面我们来演示一下 PZ-HC05 蓝牙串口模块与手机蓝牙连接，这里我们先设置蓝牙模块为从机（ Slave）角色，以便和手机连接。然后在手机上安装蓝牙串口助手 V0.16.apk 这个软件， 安装完软件后，我们打开该软件，进入搜索蓝牙设备界面，第一次连接蓝牙设备的时候会让你输入配对码，输入完成后一段时间就可以连接成功，如图所示：

从上图可以看出，手机已经搜索到我们的模块了， PZ-HC05，点击这个设备，即进入，点击连接设备，连接成功后，TFTLCD 上会显示“Connected”,并且手机界面如图所示：

这几种模式都可以用来测试 PZ-HC05 蓝牙串口模块，我们选择“键盘模式”，打开后界面如下：

从图中可以看到，有一个 LED2 亮和 LED2 灭按键，这个是我们自定义的，初始安装这个软件的时候并没有此功能按键，可以点击右上角的配置按钮，选择配置键值，如图所示：

然后点击“点我”即可对其配置，比如我们设置按键名称为“LED2 亮”，按下发送值为“+LED2 ON”，然后点击确定即可。设置完成后再次点击右上角的配置按钮，选择保存键盘配置即可。我们设置了两个按键功能，一个用于点亮 LED2，一个用于熄灭 LED2。

然后我们点击按钮就可以通过蓝牙发送对应的键值，PZ-HC05 蓝牙串口模块就会接收手机发送的数据并显示在 LCD 上，同时控制开发板上的 D2 指示灯的亮灭。

大家如果需要使用手机通过蓝牙控制更多 STM32 外部设备，可以增加相应的按键功能即可。至此我们就介绍完 PZ-HC05 蓝牙串口模块的使用，大家稍作改进就可以通过 PZ-HC05 蓝牙串口模块做很多有意思的东西。