AN1408 ATK-HC05 蓝牙串口模块使用

本应用文档（AN1408，对应**战舰 STM32 开发板扩展实验 1/MiniSTM32 开发板 V3.0 扩展实验 11**）将教大家如何在 ALIENTEK STM32 开发板上使用 ATK-HC05 蓝牙串口模块（**注意，本文档同时适用 ALIENTEK 战舰和 MiniSTM32 两款开发板**）。本文档我们将使用 ATK-HC05 蓝牙串口模实现蓝牙串口通信，并和手机连接，实现手机控制开发板。

本文档分为如下几部分：

1， ATK-HC05 蓝牙串口模块简介

2， 硬件连接

3， 软件实现

4， 验证

1、ATK-HC05 蓝牙串口模块简介

ATK-HC05 模块，是 ALIENTEK 生成的一款高性能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，并且模块兼容 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。

ATK-HC05 模块非常小巧（16mm*32mm），模块通过 6 个 2.54mm 间距的排针与外部连接，模块外观如图 1.1 所示：

图 1.1 ATK-HC05 模块外观图

图 1.1 中，从右到左，依次为模块引出的 PIN1~PIN6 脚，各引脚的详细描述如表 1.1 所示：

序号	名称	说明
1	LED	配对状态输出；配对成功输出高电平，未配对则输出低电平。

2	KEY	用于进入 AT 状态；高电平有效（悬空默认为低电平）。
3	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
4	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
5	GND	地
6	VCC	电源（3.3V~5.0V）

表 1.1 ATK-HC05 模块各引脚功能描述

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 KEY 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 KEY 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 KEY 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。

ATK-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，该部分使用以及模块的详细参数等信息，请参考 **ATK-HC05-V11 用户手册.pdf** 和 **HC05 蓝牙指令集.pdf**。

通过 ATK-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。ATK-HC05 蓝牙串口模块的原理图如图 1.2 所示：

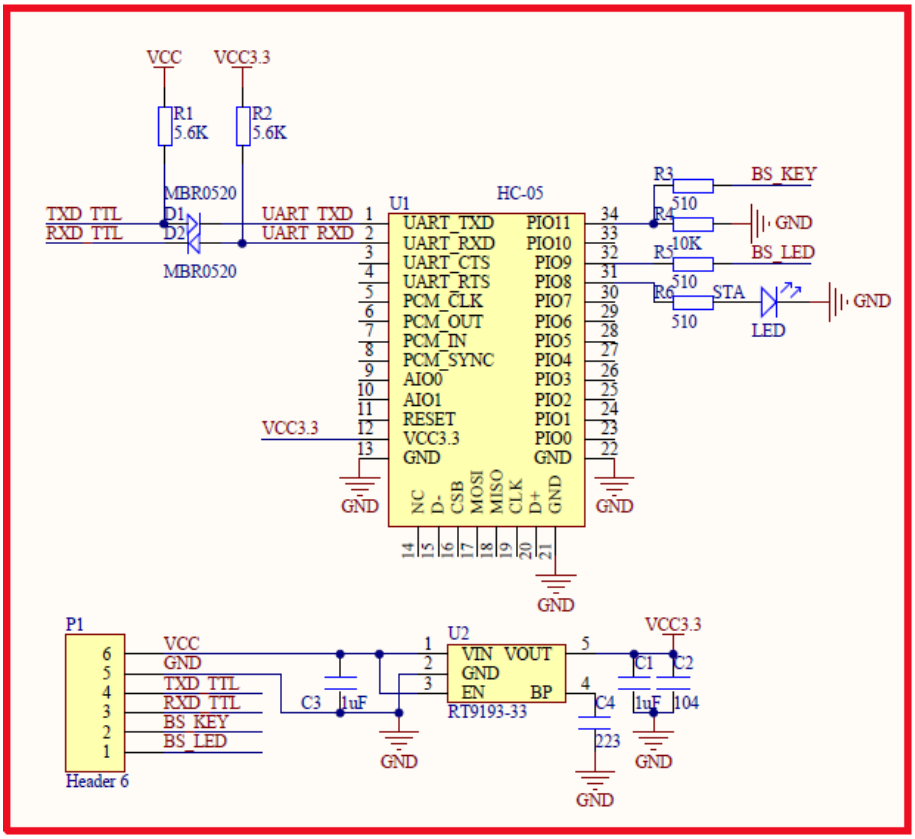

图 1.2 ATK-HC05 蓝牙串口模块原理图

2、硬件连接

本实验功能简介：开机检测 ATK-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，DS0 闪烁，提示程序运行正常。按 KEY0 按键，可以开启/关闭自动发送数据（通过蓝牙模块发送）；按 WK_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）。同时，我们还可以通过 USMART 对 ATK-HC05 蓝牙模块进行 AT 指令查询和设置。结合手机端蓝牙软件(蓝牙串口助手 v1.97.apk)，可以实现手机无线控制开发板(点亮和关闭 LED1)。

所要用到的硬件资源如下：

1， 指示灯 DS0 、DS1

2， KEY0/WK_UP 两个按键

3， 串口 1、串口 2

4， TFTLCD 模块

5， ATK-HC05-V11 蓝牙串口模块

接下来，我们看看 ATK-HC05 蓝牙串口模块同 ALIENTEK STM32 开发板的连接，前面我们介绍了 ATK-HC05 蓝牙串口模块的接口，我们通过杜邦线连接 ATK-HC05 模块和开发板的相应端口，连接关系如表 2.1 所示：

ATK-HC05 蓝牙模块与开发板连接关系						
ATK-HC05 蓝牙串口模块	VCC	GND	TXD	RXD	KEY	LED
ALIENTEK STM32 开发板	3.3V/5V	GND	PA3	PA2	PC4	PA4

表 2.1 ATK-HC05 蓝牙模块同 ALIENTEK STM32 开发板连接关系表

表中 ATK-HC05 蓝牙串口模块的 VCC，因为我们的模块是可以 3.3V 或 5V 供电的，所以可以接开发板的 3.3V 电源，也可以接开发板的 5V 电源，这个随便大家自己选择。

为了测试蓝牙模块的所有功能，上表我们用了 6 根线连接开发板，在实际使用的时候，如果不需要进入 AT 设置和状态指示，则只需要 4 根线连接即可：VCC/GND/TXD/RXD。

3、软件实现

本实验（注：这里仅以战舰板代码为例进行介绍，MiniSTM32 开发板对应代码几乎一模一样，详见 MiniSTM32 开发板 V3.0 扩展实验 11），我们在标准例程：**USMART 调试组件实验**的基础上修改，首先删掉一些本例程用不到的代码：beep.c、exti.c、wdg.c、timer.c、tpad.c 和 oled.c 等。

然后，在 HARDWARE 文件夹里面新建 USART2 和 HC05 两个文件夹，并分存放 usart2.c，usart2.h 和 hc05.c，hc05.h 等几个文件。并在工程工程 HARDWARE 组里面添加 usart2.c 和 hc05.c 两个文件，并在工程添加 usart2.h 和 hc05.h 的头文件包含路径。

在 usart2.c 里面，我们输入如下代码：

```
#include "delay.h"
#include "usart2.h"
#include "stdarg.h"
#include "stdio.h"
#include "string.h"
//串口发送缓存区
__align(8) u8 USART2_TX_BUF[USART2_MAX_SEND_LEN];     //发送缓冲
#ifdef USART2_RX_EN                                   //如果使能了接收
//串口接收缓存区
u8 USART2_RX_BUF[USART2_MAX_RECV_LEN];                //接收缓冲
//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_IRQHandler(void)
{
    u8 res;
    if(USART2->SR&(1<<5))//接收到数据
    {
        res=USART2->DR;
        if(USART2_RX_STA<USART2_MAX_RECV_LEN)//还可以接收数据
        {
            TIM4->CNT=0;                                    //计数器清空
            if(USART2_RX_STA==0)TIM4_Set(1);                //使能定时器 4 的中断
            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
        }
    }
}
//初始化 IO 串口 2
//pclk1:PCLK1 时钟频率(Mhz)
//bound:波特率
void USART2_Init(u32 pclk1,u32 bound)
{
    RCC->APB2ENR|=1<<2;         //使能 PORTA 口时钟
    GPIOA->CRL&=0XFFFF00FF;     //IO 状态设置
    GPIOA->CRL|=0X00008B00;     //IO 状态设置
    RCC->APB1ENR|=1<<17;        //使能串口时钟
    RCC->APB1RSTR|=1<<17;       //复位串口 2
    RCC->APB1RSTR&=~(1<<17);//停止复位
    //波特率设置
    USART2->BRR=(pclk1*1000000)/(bound);// 波特率设置
    USART2->CR1|=0X200C;    //1 位停止,无校验位.
    USART2->CR3=1<<7;       //使能串口 2 的 DMA 发送
    UART_DMA_Config(DMA1_Channel7,(u32)&USART2->DR,
    (u32)USART2_TX_BUF);//DMA1 通道 7,外设为串口 2,存储器为 USART2_TX_BUF
#ifdef USART2_RX_EN                 //如果使能了接收
    //使能接收中断
    USART2->CR1|=1<<8;      //PE 中断使能
    USART2->CR1|=1<<5;      //接收缓冲区非空中断使能
```

```
    MY_NVIC_Init(2,3,USART2_IRQChannel,2);//组 2，最低优先级
    TIM4_Init(99,7199);          //10ms 中断
    USART2_RX_STA=0;             //清零
    TIM4_Set(0);                 //关闭定时器 4
#endif
}
//串口 2,printf 函数
//确保一次发送数据不超过 USART2_MAX_SEND_LEN 字节
void u2_printf(char* fmt,...)
{
    va_list ap;
    va_start(ap,fmt);
    vsprintf((char*)USART2_TX_BUF,fmt,ap);
    va_end(ap);
    while(DMA1_Channel7->CNDTR!=0);      //等待通道 7 传输完成
    UART_DMA_Enable(DMA1_Channel7,strlen((const char*)USART2_TX_BUF)); \
    //通过 dma 发送出去
}
//定时器 4 中断服务程序
void TIM4_IRQHandler(void)
{
    if(TIM4->SR&0X01)//是更新中断
    {
        USART2_RX_STA|=1<<15; //标记接收完成
        TIM4->SR&=~(1<<0);       //清除中断标志位
        TIM4_Set(0);             //关闭 TIM4
    }
}
//设置 TIM4 的开关
//sta:0，关闭;1,开启;
void TIM4_Set(u8 sta)
{
    if(sta)
    {
        TIM4->CNT=0;              //计数器清空
        TIM4->CR1|=1<<0;         //使能定时器 4
    }else TIM4->CR1&=~(1<<0);//关闭定时器 4
}
//通用定时器中断初始化
//这里始终选择为 APB1 的 2 倍，而 APB1 为 36M
//arr：自动重装值。
//psc：时钟预分频数
void TIM4_Init(u16 arr,u16 psc)
{
```

```
        RCC->APB1ENR|=1<<2;     //TIM4 时钟使能
        TIM4->ARR=arr;          //设定计数器自动重装值
        TIM4->PSC=psc;          //预分频器
        TIM4->DIER|=1<<0;       //允许更新中断
        TIM4->CR1|=0x01;        //使能定时器 4
        MY_NVIC_Init(1,3,TIM4_IRQChannel,2);//抢占 2，子优先级 3，组 2 在 2 中优先级最低
    }
    #endif
    ////////////////////////////////////USART2 DMA 发送配置部分////////////////////////////////
    //DMA1 的各通道配置
    //这里的传输形式是固定的,这点要根据不同的情况来修改
    //从存储器->外设模式/8 位数据宽度/存储器增量模式
    //DMA_CHx:DMA 通道 CHx
    //cpar:外设地址
    //cmar:存储器地址
    void UART_DMA_Config(DMA_Channel_TypeDef*DMA_CHx,u32 cpar,u32 cmar)
    {
        RCC->AHBENR|=1<<0;                  //开启 DMA1 时钟
        delay_us(5);
        DMA_CHx->CPAR=cpar;             //DMA1 外设地址
        DMA_CHx->CMAR=cmar;             //DMA1,存储器地址
        DMA_CHx->CCR=0X00000000; //复位
        DMA_CHx->CCR|=1<<4;             //从存储器读
        DMA_CHx->CCR|=0<<5;             //普通模式
        DMA_CHx->CCR|=0<<6;             //外设地址非增量模式
        DMA_CHx->CCR|=1<<7;             //存储器增量模式
        DMA_CHx->CCR|=0<<8;             //外设数据宽度为 8 位
        DMA_CHx->CCR|=0<<10;            //存储器数据宽度 8 位
        DMA_CHx->CCR|=1<<12;            //中等优先级
        DMA_CHx->CCR|=0<<14;            //非存储器到存储器模式
    }
    //开启一次 DMA 传输
    void UART_DMA_Enable(DMA_Channel_TypeDef*DMA_CHx,u16 len)
    {
        DMA_CHx->CCR&=~(1<<0);              //关闭 DMA 传输
        DMA_CHx->CNDTR=len;                 //DMA1,传输数据量
        DMA_CHx->CCR|=1<<0;                 //开启 DMA 传输
    }
```

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：u2_printf，和串口 2 的接收处理。串口 2 这里我们发送数据采用 DMA 发送，以提高系统实时性。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。此种方法判断串口数据结束不同于我们串口实验里面的判断回车结束，据有更广泛的通用性，希望大家好好掌握。

usart2.h 里面的代码我们就不在这里列出了，请大家参考本文档对应源码（**扩展实验 1 ATK-HC05 蓝牙串口模块实验**），我们在 hc05.c 里面，输入如下代码：

```
#include "delay.h"
#include "usart.h"
#include "usart2.h"
#include "hc05.h"
#include "led.h"
#include "string.h"
#include "math.h"
//初始化 ATK-HC05 模块
//返回值:0,成功;1,失败.
u8 HC05_Init(void)
{
    u8 retry=10,t;
    u8 temp=1;
    RCC->APB2ENR|=1<<2;         //使能 PORTA 时钟
    RCC->APB2ENR|=1<<4;         //使能 PORTC 时钟
    GPIOA->CRL&=0XFFF0FFFF;     //PA4,输入
    GPIOA->CRL|=0X00080000;
    GPIOA->ODR|=1<<4;           //PA4 上拉
    GPIOC->CRL&=0XFFF0FFFF;     //PC4,推挽输出
    GPIOC->CRL|=0X00030000;
    GPIOC->ODR|=1<<4;           //PC4 输出 1
    USART2_Init(36,9600); //初始化串口 2 为:9600,波特率.
    while(retry--)
    {
        HC05_KEY=1;                     //KEY 置高,进入 AT 模式
        delay_ms(10);
        u2_printf("AT\r\n");            //发送 AT 测试指令
        HC05_KEY=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 ATK-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_KEY=1;                         //KEY 置高,进入 AT 模式
            delay_ms(10);
            u2_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_KEY=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;
}
//ATK-HC05 设置命令
//此函数用于设置 ATK-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_KEY=1;                 //KEY 置高,进入 AT 模式
            delay_ms(10);
            u2_printf("%s\r\n",atstr);   //发送 AT 字符串
            HC05_KEY=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;
}
//通过该函数,可以利用 USMART,调试接在串口 2 上的 ATK-HC05 模块
//str:命令串.(这里注意不再需要再输入回车符)
void HC05_CFG_CMD(u8 *str)
{
        u8 temp;
        u8 t;
        HC05_KEY=1;                         //KEY 置高,进入 AT 模式
        delay_ms(10);
        u2_printf("%s\r\n",(char*)str); //发送指令
        for(t=0;t<50;t++)                   //最长等待 500ms,来接收 HC05 模块的回应
        {
            if(USART2_RX_STA&0X8000)break;
            delay_ms(10);
        }
        HC05_KEY=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;
```

```
            USART2_RX_BUF[temp]=0;          //加结束符
            printf("\r\n%s",USART2_RX_BUF);//发送回应数据到串口 1
        }
    }
```

此部分代码总共 4 个函数：1，HC05_Init 函数，该函数用于初始化与 ATK-HC05 连接的 IO 口，并通过 AT 指令检测 ATK-HC05 蓝牙模块是否已经连接。2 ，HC05_Get_Role 函数，该函数用于获取 ATK-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 ATK-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 ATK-HC05 蓝牙串口模块的各种设置。4，HC05_CFG_CMD 函数，该函数专为 USMART 调试组件提供，专用于 USMART 测试 ATK-HC05 蓝牙串口模块的 AT 指令，在不需要 USMART 调试的时候，该函数可以去掉。**注意要将 HC05_CFG_CMD 添加到 usmart_nametab 里面，才能通过 USMART 调用该函数哦！**

hc05.h 里面的代码我们也不列出了，请大家参考本文档对应源码。

最后在 test.c 里面，我们修改代码如下：

```
//显示 ATK-HC05 模块的主从状态
void HC05_Role_Show(void)
{
    if(HC05_Get_Role()==1)LCD_ShowString(30,140,200,16,16,"ROLE:Master");//主机
    else LCD_ShowString(30,140,200,16,16,"ROLE:Slave ");                 //从机
}
//显示 ATK-HC05 模块的连接状态
void HC05_Sta_Show(void)
{
    if(HC05_LED)LCD_ShowString(120,140,120,16,16,"STA:Connected ");  //连接成功
    else LCD_ShowString(120,140,120,16,16,"STA:Disconnect");         //未连接
}
int main(void)
{
    u8 t; u8 key;
    u8 sendmask=0; u8 sendcnt=0;
    u8 sendbuf[20]; u8 reclen=0;
    Stm32_Clock_Init(9);    //系统时钟设置
    delay_init(72);         //延时初始化
    uart_init(72,9600);     //串口 1 初始化为 9600
    LED_Init();             //初始化与 LED 连接的硬件接口
    KEY_Init();             //初始化按键
    LCD_Init();             //初始化 LCD
    usmart_dev.init(72);    //初始化 USMART
    POINT_COLOR=RED;
    LCD_ShowString(30,30,200,16,16,"ALIENTEK STM32 ^_^");
    LCD_ShowString(30,50,200,16,16,"HC05 BLUETOOTH COM TEST");
    LCD_ShowString(30,70,200,16,16,"ATOM@ALIENTEK");
    while(HC05_Init())          //初始化 ATK-HC05 模块
    {
```

```
        LCD_ShowString(30,90,200,16,16,"ATK-HC05 Error!");
        delay_ms(500);
        LCD_ShowString(30,90,200,16,16,"Please Check!!!");
        delay_ms(100);
    }
    LCD_ShowString(30,90,200,16,16,"WK_UP:ROLE KEY0:SEND/STOP");
    LCD_ShowString(30,110,200,16,16,"ATK-HC05 Standby!");
    LCD_ShowString(30,160,200,16,16,"Send:");
    LCD_ShowString(30,180,200,16,16,"Receive:");
    POINT_COLOR=BLUE;
    HC05_Role_Show();
    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 ATK-HC05 模块
            }
        }else if(key==KEY_RIGHT)
        {
            sendmask=!sendmask;                 //发送/停止发送
            if(sendmask==0)LCD_Fill(30+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,"ALIENTEK HC05 %d\r\n",sendcnt);
                LCD_ShowString(30+40,160,200,16,16,sendbuf);//显示发送数据
                u2_printf("ALIENTEK HC05 %d\r\n",sendcnt);  //发送到蓝牙模块
                sendcnt++;
                if(sendcnt>99)sendcnt=0;
            }
            HC05_Sta_Show();
            t=0; LED0=!LED0;
        }
        if(USART2_RX_STA&0X8000)                //接收到一次数据了
```

```
            {
                LCD_Fill(30,200,240,320,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
                if(reclen==9||reclen==8)              //控制 DS1 检测
                {
                    if(strcmp((const char*)USART2_RX_BUF,"+LED1 ON")==0)LED1=0;
                    //打开 LED1
                    if(strcmp((const char*)USART2_RX_BUF,"+LED1 OFF")==0)LED1=1;
                    //关闭 LED1
                }
                LCD_ShowString(30,200,209,119,16,USART2_RX_BUF);//显示接收数据
                USART2_RX_STA=0;
            }
            t++;
        }
    }
```

此部分代码，实现了我们在前面提到的本节所要实现的功能。代码比较简单，我们就不啰嗦了，接下来看代码验证。

4、验证

特别注意：对于战舰板，请务必先把 P9 的两个跳线帽拔了（PA2/PA3 与 48T/48R 的跳线帽），否则开发板无法连接到蓝牙模块（有干扰）。

在代码编译成功之后，我们下载代码到我们的 STM32 开发板上（假设 ATK-HC05 蓝牙串口模块已经连接上开发板），LCD 显示如图 4.1 所示界面：

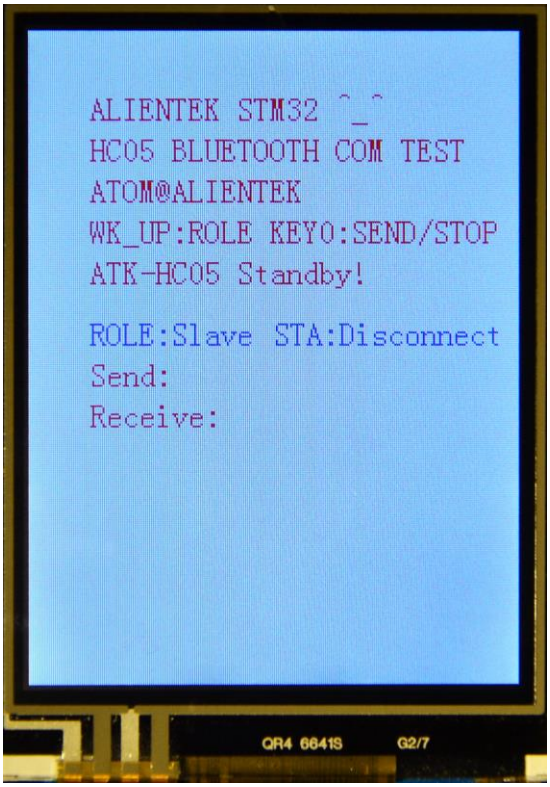

图 4.1 初始界面

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。

本实验，我们将演示两个 ATK-HC05 蓝牙串口模块的对接以及一个 ATK-HC05 蓝牙模块和手机（带蓝牙功能）的连接并通过手机控制开发板的 LED1（DS1）的亮灭。

首先我们来看两个 ATK-HC05 蓝牙串口模块的对接，两个 ATK-HC05 蓝牙模块的对接非常简单，因为 ATK-HC05 蓝牙串口模块出厂默认都是 Slave 状态的，所以我们只需要将另外一个 ATK-HC05 蓝牙串口模块上电，然后按一下开发板的 WK_UP 按键，将连接开发板的 ATK-HC05 蓝牙串口模块设置为主机（Master），稍等片刻后，两个 ATK-HC05 蓝牙模块就会自动连接成功，同时液晶显示状态为 Connected，如图 4.2 所示：

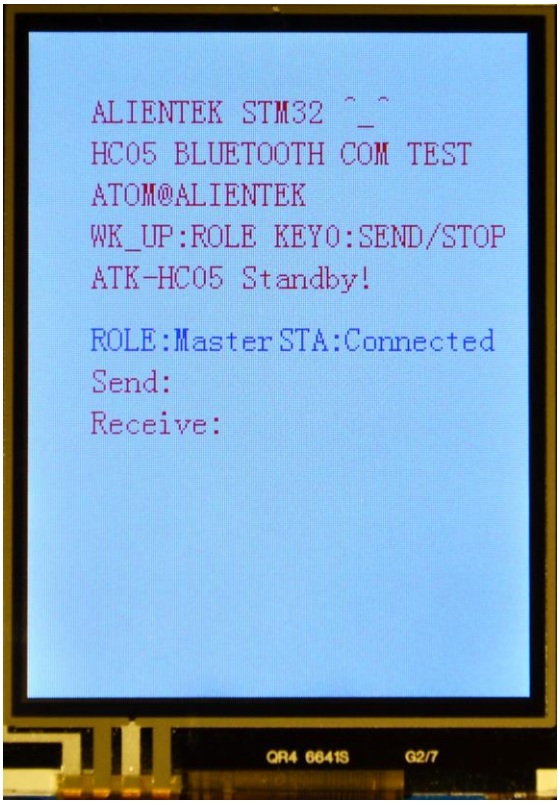

图 4.2 两个 ATK-HC05 蓝牙模块连接成功

此时，可以看到两个蓝牙模块的 STA 指示灯都是双闪（一次闪 2 下，2 秒闪一次），表示连接成功，我们通过串口助手（连接蓝牙从机）向开发板发送数据，也可以收到来自开发板的数据（按 KEY0，开启/关闭自动发送数据），如图 4.3 所示：

图 4.3 ATK-HC05 蓝牙串口模块从机发送和接收数据

点击串口调试助手的发送，我们就可以在开发板的液晶上，看到来自蓝牙从机发过来的数据，如图 4.4 所示：

图 4.4 接收到来自从机的数据

以上就是 2 个 ATK-HC05 蓝牙串口模块的对接通信。

接下来，我们看看 ATK-HC05 蓝牙串口模块同手机（必须带蓝牙功能）的连接，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 v1.97.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，如图 4.5 所示：

图 4.5 搜索蓝牙设备

从上图可以看出，手机已经搜索到我们的模块了，ATK-HC05，点击这个设备，即进入选择操作模式，如图 4.6 所示：

图 4.6 选择操作模式

这里我们选择：键盘模式（PS:实时模式在 ATK-HC05-V11 用户手册里面有介绍）。选择

模式后，我们输入密码（仅第一次连接需要设置），完成配对，如图 4.7 所示：

图 4.7 输入配对密码

在输入密码之后，等待一段时间，即可连接成功，如图 4.8 所示：

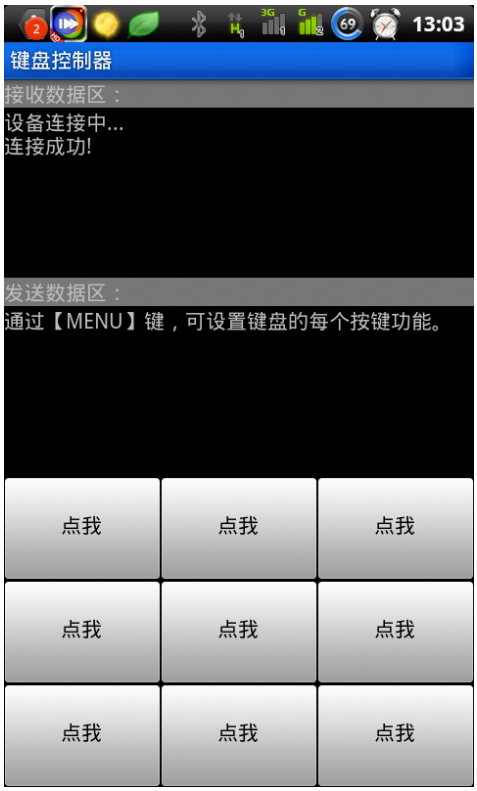

图 4.8 键盘模式连接成功

可以看到，键盘模式界面总共有 9 个按键，可以用来设置，我们点击手机的 menu 键，就可以对按键进行设置，这里我们设置前两个按键，如图 4.9 所示：

图 4.9 设置两个按键按钮名字和发送内容

在 main 函数里面，我们是通过判断是否接收"+LED1 ON"或"+LED1 OFF"字符串来决定 LED1（DS1）的亮灭的,所以我们设置两个按键的发送内容分别设置为"+LED1 ON"和"+LED1 OFF"，就可以实现对 LED1 的亮灭控制了。设置完成后，我们就可以通过手机控制开发板 LED1 的亮灭了，同时该软件还是可以接收来自开发板的数据，如图 4.10 所示：

图 4.10 手机控制开发板

通过点击 LED1 亮和 LED1 灭这两个按键，我们就可以实现对开发板 LED1(DS1)的亮灭

控制。

至此，关于 ATK-HC05 蓝牙串口模块的介绍，我们就讲完了，我们实现了两个 ATK-HC05 模块的互联以及手机通过 ATK-HC05 模块控制开发板，大家稍作改进，就可以通过 ATK-HC05 蓝牙串口模块，做很多有意思的东西。

正点原子@ALIENTEK
2014-03-29
公司网址：www.alientek.com
技术论坛：www.openedv.com
电话：020-38271790
传真：020-36773971