AN1601A ATK-AS608 指纹识别模块使用说明

本应用文档（AN1601A）将教大家如何在 ALIENTEK Mini 开发板上使用 ATK-AS608 指纹识别模块。

本文档分为如下几部分：

1， ATK-AS608 指纹识别模块简介
2， 硬件连接
3， 软件实现
4， 验证

1、ATK-AS608 指纹识别模块简介

ATK-AS608 指纹识别模块（以下简称 AS608 模块）是 ALIENTEK 推出的一款高性能的光学指纹识别模块。AS608 模块采用了国内著名指纹识别芯片公司杭州晟元芯片技术有限公司(Synochip)的 AS608 指纹识别芯片。芯片内置 DSP 运算单元，集成了指纹识别算法，能高效快速采集图像并识别指纹特征。模块配备了串口、USB 通讯接口，用户无需研究复杂的图像处理及指纹识别算法，只需通过简单的串口、USB 按照通讯协议便可控制模块。本模块可应用于各种考勤机、保险箱柜、指纹门禁系统、指纹锁等场合。

技术指标：

项目	说明
工作电压(V)	3.0~3.6V，典型值：3.3V
工作电流(mA)	30~60mA，典型值：40mA
USART 通讯	波特率(9600×N)，N=1~12。默认 N=6,bps= 57600 (数据位:8 停止位:1 校验位:none TTL 电平)
USB 通讯	2.0FS (2.0 全速)
传感器图像大小(pixel)	256*288pixel
图像处理时间(S)	<0.4(S)
上电延时(S)	<0.1(S),模块上电后需要约 0.1S 初始化工作
搜索时间(S)	<0.3(S)
拒真率(FRR)	<1%
认假率(FAR)	<0.001%
指纹存容量	300 枚(ID:0~299)
工作环境	温度(°C):-20~60 湿度<90%(无凝露)

2、硬件连接

2.1 模块接口

AS608 模块内部内置了手指探测电路，用户可读取状态引脚(WAK)判断有无手指按下。接口采用 8 芯 1.25 mm 间距单排插座，PCB 如图 2.1.1 所示。

序号	名称	说明
1	Vi	模块电源正输入端。
2	Tx	串行数据输出。 TTL 逻辑电平
3	Rx	串行数据输入。 TTL 逻辑电平
4	GND	信号地。内部与电源地连接
5	WAK	感应信号输出，默认高电平有效
6	Vt	触摸感应电源输入端，3.3v 供电
7	U+	USB D+
8	U-	USB D-

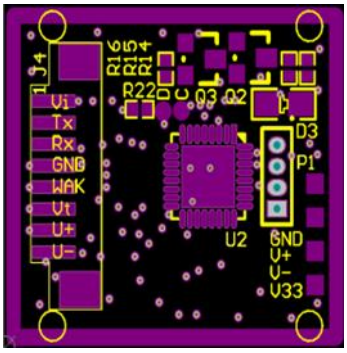


图 2.1.1

2.2 与开发板连接

AS608 模块与 Mini 开发板连接关系图如下图 2.2.1 所示。关系表如表 2.2.1。

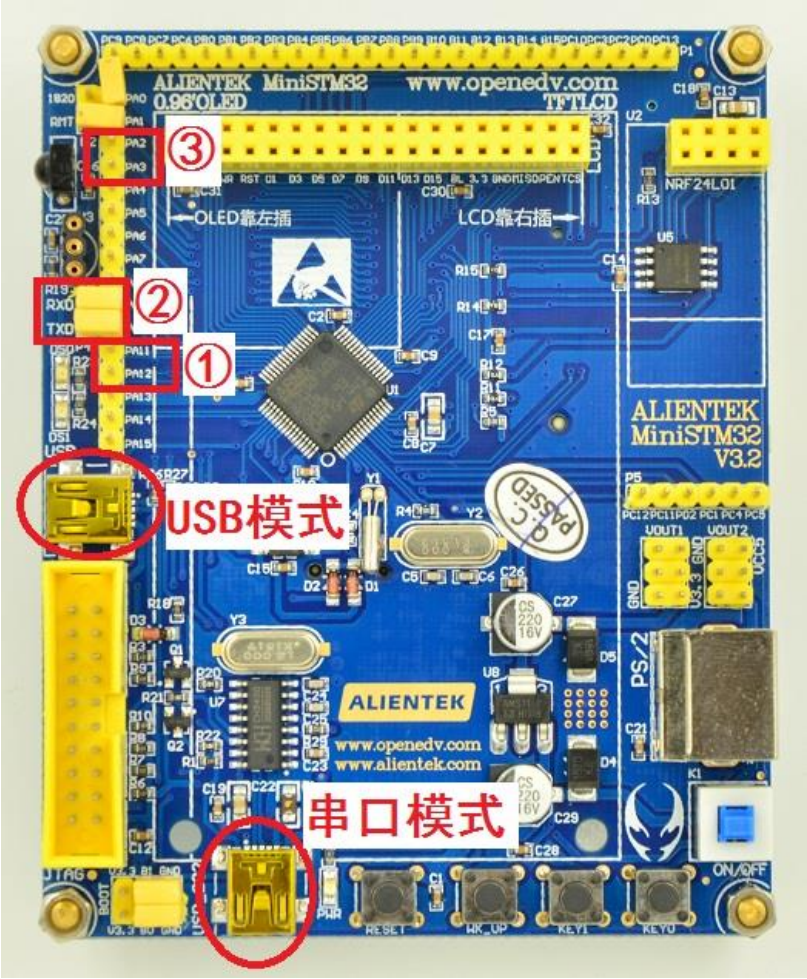


图 2.2.1 AS608 模块与 Mini 开发板连接关系图

- **例程实验演示模式：**AS608 模块的 Tx、Rx 分别连接到③中的 PA3(RX)、PA2(Tx)。
- **上位机 USB 测试：**AS608 模块的 U+、U-分别连接到①中的(PA12)、(PA11)，USB 数据线接到 USB 模式。
- **上位机串口测试：**AS608 模块的 Tx、Rx 分别连接到②中的 RXD、TXD，USB 数据线接到串口模式。

提示：连接方式，取下跳线帽是用杜邦线连接。

AS608 模块指纹识别模块与开发板连接关系									
AS608 模块		Vi	Tx	Rx	GND	WAK	Vt	U+	U-
开发板	例程实验演示模式	3.3V	PA3(RX)	PA2(TX)	GND	PA6	3.3V	—	—
	上位机 USB 测试	3.3V	—	—	GND	—	3.3V	U+ (PA12)	U- (PA11)
	上位机串口测试	3.3V	RXD	TXD	GND	—	3.3V	—	—

表 2.2.1 AS608 模块与 Mini 开发板连接关系表

注：上位机 USB 测试、串口测试及使用串口助手调试的方法在模块资料\ATK-AS608 指纹识别模块用户手册.pdf 中说明。本文档只说明例程实验演示模式。

3、软件实现

本实验主要实现录入指纹、刷指纹（验证指纹）、使用 USMART 读取和修改模块参数等功能。程序是在 Mini 开发板的汉字显示实验和 T9 拼音输入法实验基础上进行修改的，并增加了 usart2.c、as608.c，这里我们使用 usart2.c 与 AS608 模块通讯。注意：由于 Mini 开发板的 USART2（PA2\PA3）连接到了 Flash_CS 和 SD_CS，当每显示一个汉字时需要切换 Flash_CS 状态这就会引起高低电平信号。这个信号在与 AS608 模块握手时被认为错误指令，从而导致握手失败，因此我们在程序流程中均用英文提示信息，除了显示模块返回确认码的信息使用中文之外。usart2.c 参考了在之前的蓝牙例程的 usart3.c（详见：AN1408A ATK-HC05 蓝牙串口模块使用说明）里面介绍过了结合定时器超时接收完成数据的机制。这里，我们就不再介绍 usart2.c,主要看 as608.c 和 main.c 的代码，首先是 as608.c，该文件是 AS608 模块的指令代码，as608.c 部分代码如下：

```
u32 AS608Addr = 0XFFFFFFFF; //默认

//初始化 PA6 为下拉输入
//读摸出感应状态(触摸感应时输出高电平信号)
void PS_StaGPIO_Init(void)
{
   GPIO_InitTypeDef    GPIO_InitStructure;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//使能 GPIOA 时钟
   //初始化读状态引脚 GPIOA
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;
   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;//输入下拉模式
   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;//50MHz
   GPIO_Init(GPIOA, &GPIO_InitStructure);//初始化 GPIO
}
//串口发送一个字节
static void MYUSART_SendData(u8 data)
{
```

```
        while((USART2->SR&0X40)==0);
        USART2->DR = data;
}
//发送包头
static void SendHead(void)
{
        MYUSART_SendData(0xEF);
        MYUSART_SendData(0x01);
}
//发送地址
static void SendAddr(void)
{
        MYUSART_SendData(AS608Addr>>24);
        MYUSART_SendData(AS608Addr>>16);
        MYUSART_SendData(AS608Addr>>8);
        MYUSART_SendData(AS608Addr);
}
//发送包标识,
static void SendFlag(u8 flag)
{
        MYUSART_SendData(flag);
}
//发送包长度
static void SendLength(int length)
{
        MYUSART_SendData(length>>8);
        MYUSART_SendData(length);
}
//发送指令码
static void Sendcmd(u8 cmd)
{
        MYUSART_SendData(cmd);
}
//发送校验和
static void SendCheck(u16 check)
{
        MYUSART_SendData(check>>8);
        MYUSART_SendData(check);
}
//判断中断接收的数组有没有应答包
//waittime 为等待中断接收数据的时间（单位 1ms）
//返回值：数据包首地址
static u8 *JudgeStr(u16 waittime)
{
```

```
        char *data;
        u8 str[8];
        str[0]=0xef;str[1]=0x01;str[2]=AS608Addr>>24;
        str[3]=AS608Addr>>16;str[4]=AS608Addr>>8;
        str[5]=AS608Addr;str[6]=0x07;str[7]='\0';
        while(--waittime)
        {
            delay_ms(1);
            if(USART2_RX_STA&0X8000)//接收到一次数据
            {
                USART2_RX_STA=0;
                data=strstr((const char*)USART2_RX_BUF,(const char*)str);
                if(data)
                    return (u8*)data;
            }
        }
        return 0;
    }
    //录入图像 PS_GetImage
    //功能:探测手指，探测到后录入指纹图像存于 ImageBuffer。
    //模块返回确认字
    u8 PS_GetImage(void)
    {
        u16 temp;
        u8   ensure;
        u8   *data;
        SendHead();
        SendAddr();
        SendFlag(0x01);//命令包标识
        SendLength(0x03);
        Sendcmd(0x01);
        temp = 0x01+0x03+0x01;
        SendCheck(temp);
        data=JudgeStr(2000);
        if(data)
            ensure=data[9];
        else
            ensure=0xff;
        return ensure;
    }
```

as608.c 代码比较多，里面很多都是指令，指令的格式都是一样，所以我们仅贴出部分代码进行讲解一下。

首先，是读 AS608 模块触摸感应状态引脚(WAK)，初始化函数为 void PS_StaGPIO_Init(void)。因为当感应到的时候是输出高电平，所以状态引脚（PA6）配置为下

拉输入模式。往下就是配置串口发送指令的包头、指令码、校验和之类。

第二个函数 static u8 *JudgeStr(u16 waittime)，里面调用了<sting.h>中的 strstr(const char*str1, const char* str2);这个函数是判断 str2 是否包含在 str1 内，如果包含则返回包含数据的首地址，否则返回 NULL。这里我们用于判断串口中断接收的数据包中有没有包含应答指令的包头、模块地址、指令码（07）。参数 waittime 是等待判断的时间单位（1ms）。

第三个函数 u8 PS_GetImage(void)，这个函数是和 AS608 通讯获取图像的指令，里面包含发送包头、地址、校验和，和等待接收模块应答指令 JudgeStr(2000)。AS608.c 中其他指令函数格式都跟这条指令差不多，这里就不一一贴出来讲解。

as608.c 我们就介绍到这里，我们再来看看 main.c 中主函数代码如下：

```
int main(void)
{
    u8 ensure;
    u8 key_num;
    char *str;

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置系统中断优先级分组 2
    delay_init();                   //初始化延时函数
    uart_init(115200);              //初始化串口 1 波特率为 115200，用于支持 USMART
    usart2_init(usart2_baund);      //初始化串口 2,用于与指纹模块通讯
    PS_StaGPIO_Init();              //初始化 FR 读状态引脚
    KEY_Init();                     //按键初始化
    LCD_Init();                     //LCD 初始化
    tp_dev.init();                  //初始化触摸屏
    usmart_dev.init(72);            //初始化 USMART
    mem_init();                     //初始化内存池
    exfuns_init();                  //为 fatfs 相关变量申请内存
    f_mount(fs[1],"1:",1);          //挂载 FLASH.
    POINT_COLOR=RED;
    while(font_init())              //检查字库
    {
        LCD_ShowString(60,50,240,16,16,"Font Error!");
        delay_ms(200);
        LCD_Fill(60,50,240,66,WHITE);//清除显示
        delay_ms(200);
    }
    if(!(tp_dev.touchtype&0x80))//如果是电阻屏
    {
        Show_Str_Mid(0,30,"Adjust the touch screen?",16,240); //是否校准触摸屏？
        POINT_COLOR=BLUE;
        Show_Str_Mid(0,60,"YES: KEY1      NO: KEY0",16,240);
        while(1)
        {
            key_num=KEY_Scan(0);
            if(key_num==KEY0_PRES)
```

```
                break;
            if(key_num==KEY1_PRES)
            {
                LCD_Clear(WHITE);
                TP_Adjust();    //屏幕校准
                TP_Save_Adjdata();//保存校准参数
                break;
            }
        }
    }
    /*加载指纹识别实验界面*/
    LCD_Clear(WHITE);
    POINT_COLOR=RED;
    Show_Str_Mid(0,0,"AS608 Fingerprint module test",16,240);

    Show_Str_Mid(0,20,"Author: @ALIENTEK",16,240);
    POINT_COLOR=BLUE;
    Show_Str_Mid(0,40,"Connect with AS608....",16,240);
    while(PS_HandShake(&AS608Addr))    //与 AS608 模块握手
    {
        LCD_Fill(0,40,240,80,WHITE);
        Show_Str_Mid(0,40,"Cannot connect with AS608!",16,240);
        delay_ms(1000);
        LCD_Fill(0,40,240,80,WHITE);
        Show_Str_Mid(0,40,"Try to connect again....",16,240);
        delay_ms(1000);
    }
    LCD_Fill(0,40,240,320,WHITE);
    Show_Str_Mid(0,40,"Connect success!",16,240);//通讯成功
    str=mymalloc(30);
    sprintf(str,"Baudrate:%d    Addr:%x",usart2_baund,AS608Addr);//显示波特率
    Show_Str(0,60,240,16,(u8*)str,16,0);
    delay_ms(100);
    ensure=PS_ValidTempleteNum(&ValidN);    //读库指纹个数
    if(ensure!=0x00)
        ShowErrMessage(ensure);             //显示确认码错误信息
    ensure=PS_ReadSysPara(&AS608Para);      //读 AS608 模块参数
    if(ensure==0x00)
    {
        mymemset(str,0,50);
        sprintf(str,"RemainNum:%dLevel:%d",AS608Para.PS_max-ValidN,
                AS608Para.PS_level);        //显示剩余指纹数量和安全等级
        Show_Str(0,80,240,16,(u8*)str,16,0);
    }
```

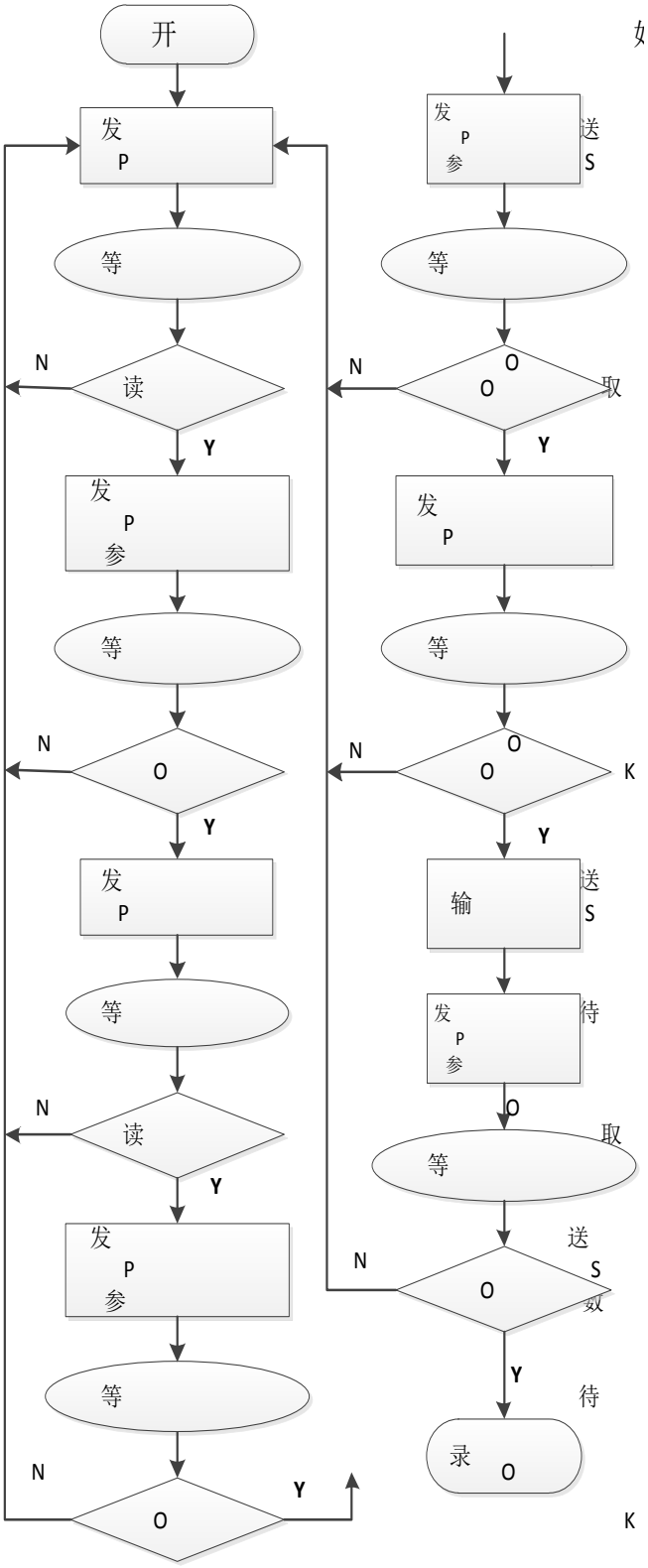
```
        else
            ShowErrMessage(ensure);
        myfree(str);
        AS608_load_keyboard(0,170,(u8**)kbd_menu);//加载虚拟键盘
        while(1)
        {
            key_num=AS608_get_keynum(0,170);
            if(key_num)
            {
                if(key_num==1)Del_FR();          //删指纹
                if(key_num==3)Add_FR();          //录指纹

            }
            if(PS_Sta)  //检测 PS_Sta 状态，如果有手指按下
            {
                press_FR();//刷指纹
            }
        }
    }
```

Main 函数比较简单，初始化硬件→检查字库→是否触摸校准（电阻屏）→与 AS608 模块通讯→通讯成功读取模块参数→显示模块参数→加载虚拟键盘→while 循环获取触摸键值→判断键值进入录指纹或删指纹流程→判断触摸感应状态→进入刷指纹流程。

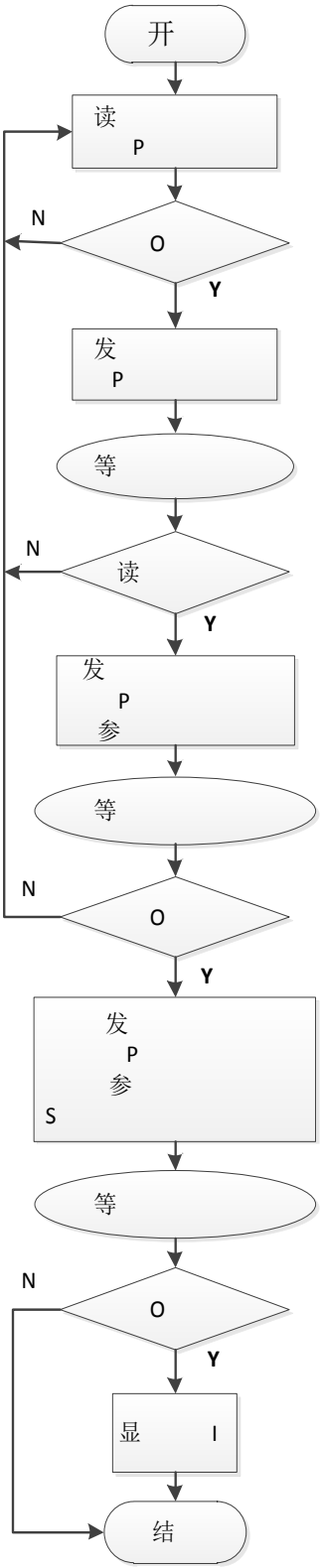
提示：检查字库出错时，需要先下载汉字显示实验更新字库！进入试验后发现触摸不了或者位置不对则复位进行触摸校准！虚拟键盘的制作请参考 T9 拼音输入法实验。

Add_FR();//录入指纹和 Del_FR();//删除指纹的流程如下：

录入指纹流程：



刷指纹流程：



我们的例程代码中分别使用了 2 个 switch 语句按照以上流程图一步步完成录入指纹、刷指纹两个流程。代码简单易懂，这里就不贴出来讲解了，大家可以打开实验工程并参考此流程图理解。

最后，我们讲解一下如何使用 USMART 修改模块地址、修改波特率、读系统参数，首先看一下 usmart_config.c，程序如下：

```
struct _m_usmart_nametab usmart_nametab[]=
{
#if USMART_USE_WRFUNS==1     //如果使能了读写操作
        (void*)read_addr,"u32 read_addr(u32 addr)",
        (void*)write_addr,"void write_addr(u32 addr,u32 val)",
#endif
        /*AS608 指纹识别模块驱动函数*/
        //写系统寄存器",
        (void*)PS_WriteReg,"uint8_t PS_WriteReg(uint8_t RegNum,uint8_t DATA);
        (void*)PS_ReadSysPara,"uint8_t PS_ReadSysPara(SysPara *p); //读系统基本参数",
        (void*)PS_SetAddr,"uint8_t PS_SetAddr(uint32_t addr);    //设置模块地址",
        //读有效模板个数",
        (void*)PS_ValidTempleteNum,"uint8_t PS_ValidTempleteNum(uint16_t *ValidN);
};
```

我们增加了“PS_WriteReg()//写系统寄存器、PS_ReadSysPara()//读系统参数、PS_SetAddr()//设置模块地址、PS_ValidTempleteNum()//读有效模板个数”四个函数。这个四个函数在 as608.c 中，我们可通过 USMART 调用这些函数来读取或修改 AS608 模块的参数。如下图 3.1 和图 3.2 所示。（提示：使用 USMART 先将 PA9>RXD、PA10>TXD 短接）

图 3.1 使用 USMART 修改地址、读参数

图 3.1 中共发送了三条函数，list：读取函数清单、PS_SetAddr(0x12345678)设置地址、PS_ReadSysPara(“”)读取系统参数。（提示：设置地址成功之后需复位开发板才能显示修改后的地址）

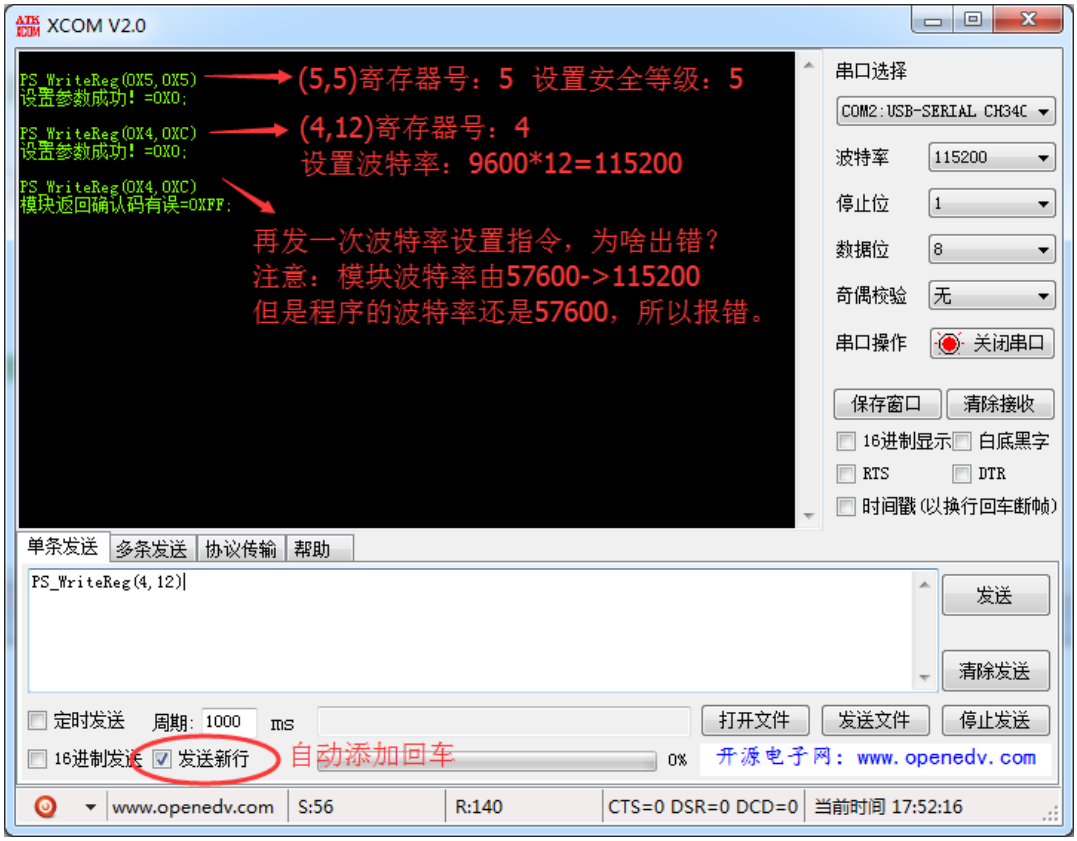


图 3.2 使用 USMART 设置安全等级、波特率

图 3.2 发送了 PS_WriteReg(5,5)、PS_WriteReg(4,12)、PS_WriteReg(4,12)三条函数，PS_WriteReg(uint8_t RegNum,uint8_t DATA)是对 AS608 写寄存器的函数，详细输入参数解析如下图 3.3 写系统寄存器指令所示。

(14) 写系统寄存器 PS_WriteReg

- 功能说明：写模块寄存器
- 输入参数：寄存器序号
- 返回参数：确认字
- 指令代码：0eH
- 指令包格式：

2 bytes	4bytes	1 byte	2 bytes	1 byte	1byte	1byte	2 bytes
包头	芯片地址	包标识	包长度	指令码	寄存器序号	内容	校验和
0xEF01	xxxx	01H	05H	0eH	4/5/6	xx	sum

- 应答包格式：

2 bytes	4bytes	1 byte	2 bytes	1 byte	2 bytes
包头	芯片地址	包标识	包长度	确认码	校验和
0xEF01	xxxx	07H	03H	xxH	sum

寄存器号	寄存器名称	内容说明
4	波特率控制寄存器	9600 的倍数 N
5	比对阀值寄存器	1：level1 2：level2 3：level3 4：level4 5：level5
6	包大小寄存器	0：32bytes 1：64bytes 2：128bytes 3：256bytes

图 3.3 写系统寄存器指令

注意：使用 USMART 必须使用跳线帽将②(PA9>RXD)(PA10>TXD)短接。修改了地址、安全等级必须复位才能显示在 LCD 上，修改了波特率则需要更改程序中 usart2 的初始化波特率，重新编译烧录程序。如果修改了波特率忘记了，则可以通过上位机识别出来。与上位机通讯必须无口令、地址：0xFFFFFFFF。

4、验证

首先，使用杜邦线将模块连接到开发板，连接方式按照上述表 2.2.1 AS608 模块与 Mini 开发板连接关系表中的例程实验演示模式连接。本文档以 ALIENTEK- Mini 开发板及 2.8'LCD 进行实验，下载代码到开发板上，显示如下图：

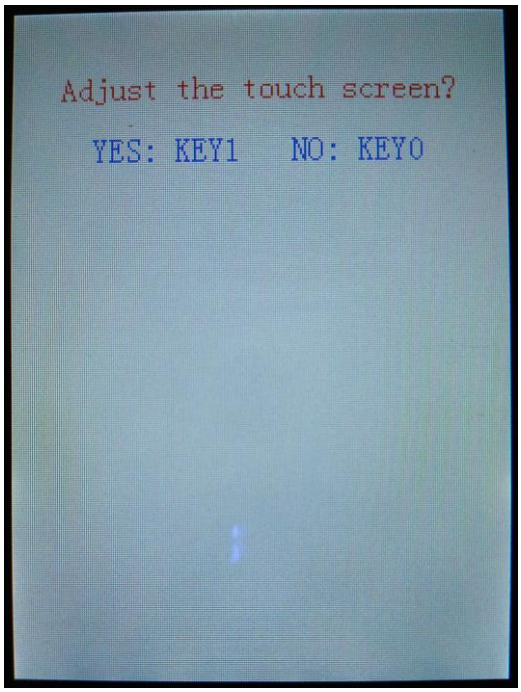


图 4.1 触摸校准界面

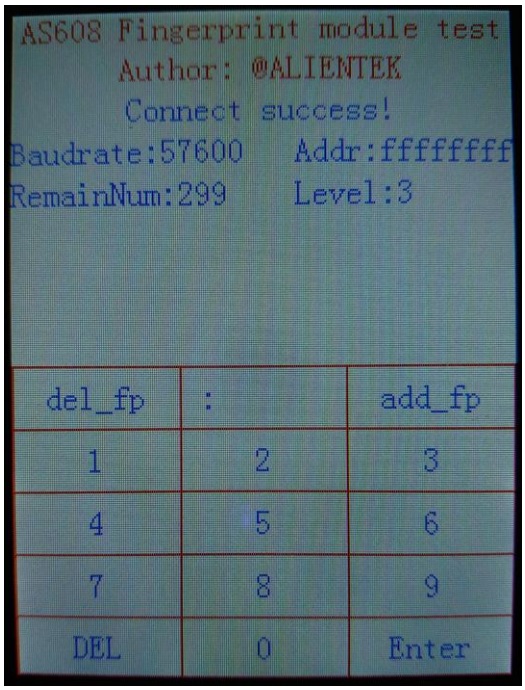


图 4.2 指纹测试主界面

图 4.1 触摸校准界面：当使用电阻屏（2.8'\3.5'LCD）时，字库初始化成功之后弹出此界面，电容屏（4.3'\7'LCD）则无此界面。因为电容屏不需要触摸校准。

图 4.2 指纹测试主界面：当与 AS608 模块通讯成功之后弹出此界面。界面中"RemainNUM"即：当前还剩余多少枚指纹容量。"Level"：安全等级。键盘中冒号空格：显示键入的数值。在此界面可触摸功能键有"del_fp"删除指纹、"add_fp"增加指纹，还可以刷指纹（验证指纹）。

录指纹：按下“del_fp”后按照提示操作如下图：

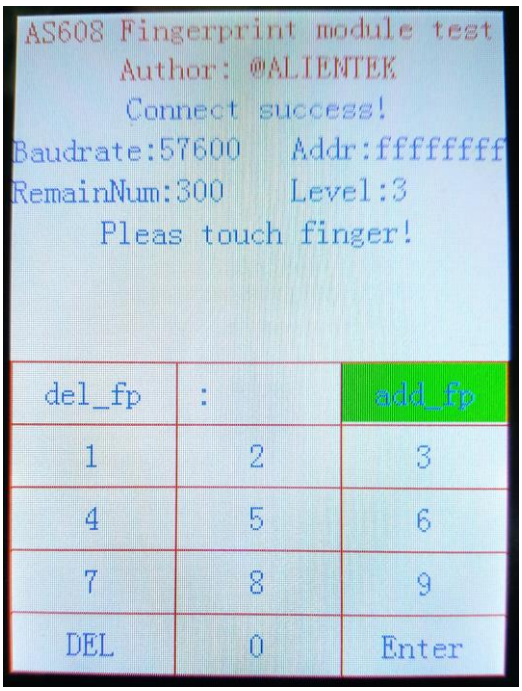


图 4.3 录指纹

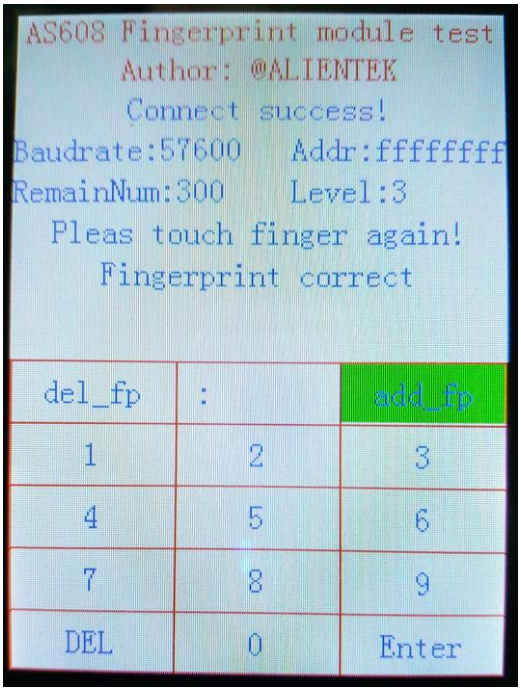


图 4.4 再按一次指纹

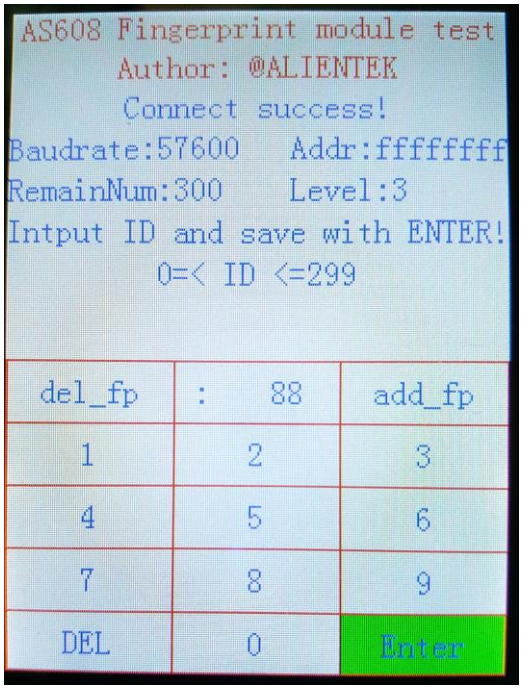


图 4.5 输入存储 ID

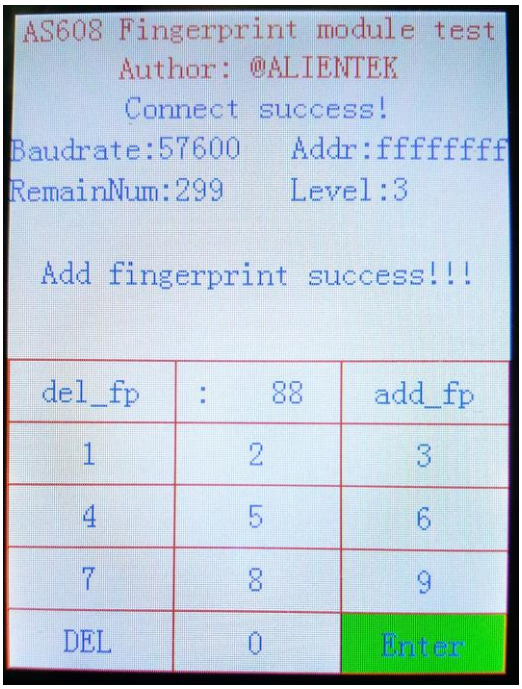


图 4.6 录入指纹成功

在测试主界面按下“add_fp”，系统会提示“Pleas touch finger!”请按手指如图 4.3 所示。然后根据提示，录入第一次指纹，录入成功之后系统会提示“Pleas touch finger again!”请再次按手指如图 4.4 所示。当第二次指纹也录入成功，系统就会提示对比两次指纹，如果两次指纹一致则会生成指纹模板，接着就会提示“Input ID and svae with ENTER”输入 ID 再按“ENTER”保存如图 4.5 所示。最后显示录入指纹成功，如图 4.6 所示。录指纹的每一个步骤成功或失败都会提示。

提示：程序中判断读取指纹图像如果 5 次都没有读到，系统会自动退出录指纹流程。

刷指纹：在测试主界面下，我们按下没有录入指纹的手指和已经录入指纹的手指，结果分别如下图：

图 4.7 按下无指纹的手指

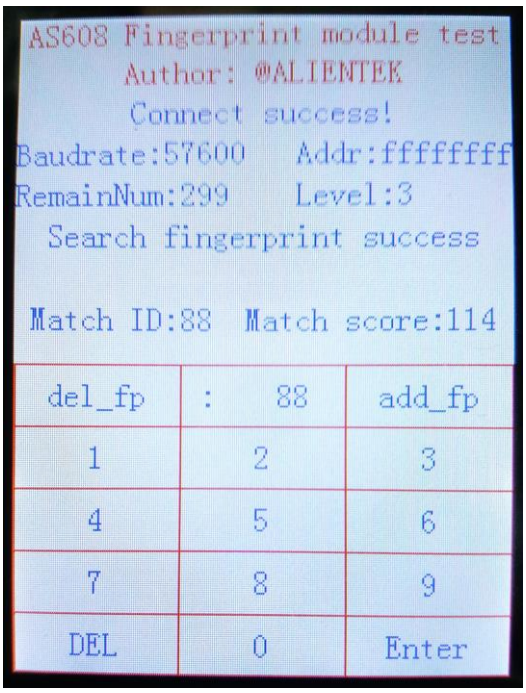


图 4.8 按下有指纹的手指

删指纹：在测试主界面下，按下“del_fp”后按照提示操作如下图：

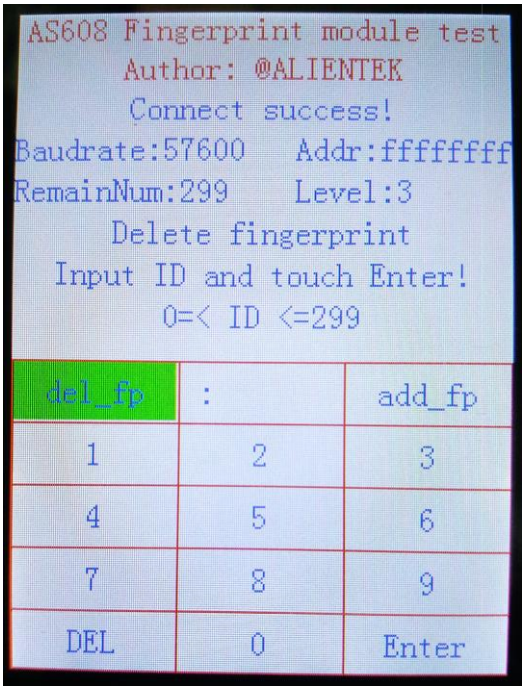


图 4.9 按下“del_fp”

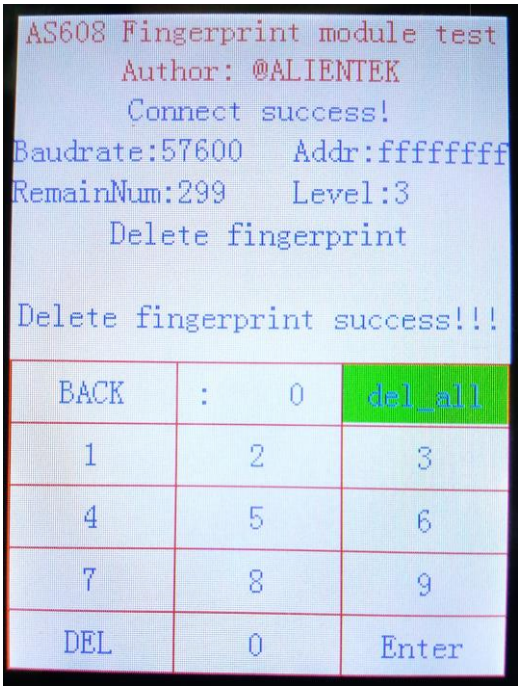


图 4.10 清空指纹库

在删指纹的界面，我们可以删除单个指纹，如图 4.9，我们输入一个指纹 ID 之后再按“Enter”就可以删除单个指纹了。也可以清空指纹库，如图 4.10 所示。如果不想删除了，也可以按“BACK”键返回测试主界面。

至此，关于 ATK-AS608 模块的使用介绍，我们就讲完了，本文档详细介绍了 ATK-AS608 模块的使用和使用过程的注意事项。请大家务必按照文档说明操作，如有疑问可联系我们，谢谢！

正点原子@ALIENTEK
2016-6-20
公司网址：www.alientek.com
技术论坛：www.openedv.com
电话：020-38271790
传真：020-36773971