

单片机外设报告

外设名称：FPC1020A指纹识别模块

姓 名 林宇航

学 号 201906060308

指导老师 郭方洪

专业班级 自动化1901

学 院 信息工程学院

提交日期 2022.1.1

**一、模块基本介绍**

FPC1020A是为单片机设计的指纹识别二次开发模块，具有体积小、功耗低、接口简单、可靠性高、指纹模板小( 200字节)、大容量指纹识别( 1000枚指纹识别响应时间小于1秒)等优点，可以非常方便将其嵌入用户系统，组成满足客户需求的指纹识别产品。

尤其具备自学习功能，指纹识别过程中，最新提取的指纹特征值识别成功后将该特征值融合到指纹数据库中，使用户在使用过程中越用越好用。1020A模块通讯接口为UART或USB，本模块作为从设备，由主设备发送相关命令对其进行控制。

该模块具有可调节的安全等级功能、指纹特征数据的读/写功能和指纹图像的读/写功能,识别方式为1:N识别或1:1验证。

**二、模块引脚图**

模块与用户设备的串行通讯时，接口引脚定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **引脚号** | **名 称** | **类型** | **功 能 描 述** |
| 1 | VT | In | 手指检测电源正输入端，3.3V |
| 2 | T | Out | 手指检测信号输出端 |
| 3 | V | in | 电源正输入端,3.3V |
| 4 | TX | out | 串行数据输出，TTL逻辑电平 |
| 5 | RX | in | 串行数据输入，TTL逻辑电平 |
| 6 | GND | － | 信号地，内部与电源地连接 |

表格 1 引脚功能定义

需要注意的是此模块需要2组电源单独供电，一组为模块串口电源，一组为感应上电芯片电源，共用地。1、6为感应芯片电源，需一直供电；3、6为模块串口电源。当手指靠近指纹窗口时，管脚2产生一个高电平，MCU接收到这个电平时，通过模块串口电源控制电路（三极管或MOS管）给模块3、6脚上电。

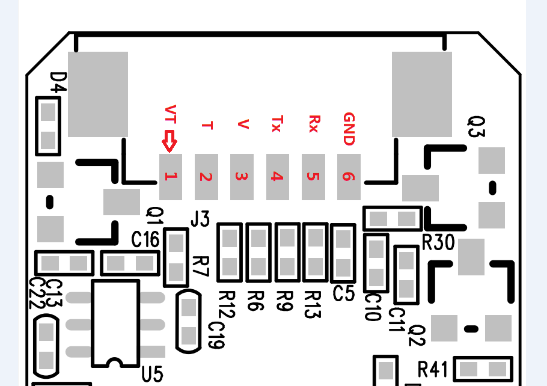


图 1 模块引脚序号图

**三、模块通讯协议**

因为在大作业中只涉及到UART的协议，所以USB协议在此不做介绍，且UART协议仅列出常用命令。

默认UART波特率为19200bps 1起始位 1停止位 （无校验位）

主设备发送的命令及DSP模块的应答按数据长度可分为两类：

**1）= 8字节，数据格式如下：**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | CMD | P1 | P2 | P3 | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | CMD | Q1 | Q2 | Q3 | 0 | CHK | 0xF5 |

说明：

CMD： 命令/应答类型

P1，P2，P3：命令参数

Q1，Q2，Q3：应答参数，

Q3多用于返回操作的有效性信息，此时可有如下取值：

#define ACK\_SUCCESS 0x00 //操作成功

#define ACK\_FAIL 0x01 //操作失败

#define ACK\_FULL 0x04 //指纹数据库已满

#define ACK\_NOUSER 0x05 //无此用户

#define ACK\_USER\_OCCUPIED

　　　　　0x06　　//此ID用户已存在

#define ACK\_USER\_EXIST 0x07 //用户已存在

#define ACK\_TIMEOUT 0x08 //采集超时

0x09 //采集图像有关的操作时， //代表无指纹按压，这时要重复发

CHK： 校验和，为第2字节到第6字节的异或值

**2）> 8字节，数据由两部分组成：数据头+数据包**

数据头格式：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | CMD | Hi(Len) | Low( Len) | 0 | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | CMD | Hi(Len) | Low(Len) | Q3 | 0 | CHK | 0xF5 |

说明：

CMD，Q3的定义同上

Len： 数据包内有效数据长度，16位，由两字节组成

Hi(Len)： 数据包长度高8位

Low(Len)： 数据包长度低8位

CHK： 校验和，为第2字节到第6字节的异或值

数据包格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字节 | 1 | 2…Len + 1 | Len + 2 | Len + 3 |
| 命令 | 0xF5 | Data | CHK | 0xF5 |
| 应答 | 0xF5 | Data | CHK | 0xF5 |

说明：

Len即为Data的字节数；

CHK： 校验和，为第2字节到第Len + 1字节的异或值

发送完数据头后紧接着发送数据包。

1. **命令类型**

**2．1 使模块进入休眠状态（命令/应答均为8字节）**

命令数据格式：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x2C | 0 | 0 | 0 | 0 | CHK | 0xF5 |

应答数据格式：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 应答 | 0xF5 | 0x2C | 0 | 0 | 0 | 0 | CHK | 0xF5 |

**2．2设置/读取指纹添加模式（命令/应答均为8字节）**

指纹添加分两种模式：允许重复模式/禁止重复模式, 在”禁止重复模式”下，同一枚手指只能添加一个用户，若强行进行第二轮添加将返回错误信息。上电后系统处于禁止重复模式。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x2D | 0 | Byte5=0：  0：允许重复1：禁止重复  Byte5=1：0 | 0：设置新的添加模式  1：读取当前添加模式 | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x2D | 0 | 当前添加模式 | ACK\_SUCCUSS  ACK\_FAIL | 0 | CHK | 0xF5 |

**2．3添加指纹（命令/应答均为8字节）**

为确保有效性，用户必须录入2~6次指纹，主机须向DSP模块发送2~6次命令。录入时6次，可提高比对通过率。添加时，0x01 -> 0x02 -> 0x02 -> 0x02 -> 0x02-> 0x03, 0x02可发0~4次。（3位和4位分别为用户号的高8位和低8位）

1. 第1次

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x01 | ~ | *~* | 用户权限（1/2/3） | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x01 | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL  ACK\_FULL  ACK\_TIMEOUT  ACK\_USER\_EXIST | 0 | CHK | 0xF5 |

1. 第2次

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x02 | ~ | ~ | 用户权限(1/2/3) | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x02 | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL  ACK\_TIMEOUT | 0 | CHK | 0xF5 |

1. 第3次

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x02 | ~ | ~ | 用户权限(1/2/3) | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x02 | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL  ACK\_TIMEOUT | 0 | CHK | 0xF5 |

1. 第4次

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x02 | ~ | ~ | 用户权限(1/2/3) | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x02 | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL  ACK\_TIMEOUT | 0 | CHK | 0xF5 |

1. 第5次

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x02 | ~ | ~ | 用户权限(1/2/3) | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x02 | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL  ACK\_TIMEOUT | 0 | CHK | 0xF5 |

1. 第6次

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x03 | ~ | ~ | 用户权限(1/2/3) | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x03 | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL  ACK\_TIMEOUT | 0 | CHK | 0xF5 |

说明：

6次命令中用户号与用户权限应为相同值，指纹重复登录时，返回之前存在的用户ID号。如果只需要采集2次，发第1次和第6次即可。1020A小面阵传感器建议采集6次。

**2．4删除指定用户（命令/应答均为8字节）**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x04 | 用户号（高8位） | 用户号（低8位） | 0 | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x04 | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL | 0 | CHK | 0xF5 |

**2．5删除所有用户（命令/应答均为8字节）**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x05 | 0 | 0 | 0 | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x05 | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL | 0 | CHK | 0xF5 |

**2．6取用户总数（命令/应答均为8字节）**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x09 | 0 | 0 | 0 | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x09 | 用户数（高8位） | 用户数（低8位） | ACK\_SUCCESS  ACK\_FAIL | 0 | CHK | 0xF5 |

**2．7比对1：1（命令/应答均为8字节）**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x0B | 用户号（高8位） | 用户号（低8位） | 0 | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x0B | 0 | 0 | ACK\_SUCCESS  ACK\_FAIL  ACK\_TIMEOUT | 0 | CHK | 0xF5 |

**2．8比对1：N（命令/应答均为8字节）此命令把探测手指、采集、生成特征值、比对集合到一起了，默认不超时，有协议可设，随便再发条命令如取总数就打出来了。**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字节 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 命令 | 0xF5 | 0x0C | 0 | 0 | 0 | 0 | CHK | 0xF5 |
| 应答 | 0xF5 | 0x0C | 用户号（高8位） | 用户号（低8位） | 用户权限(1/2/3)  ACK\_NOUSER  ACK\_TIMEOUT | 0 | CHK | 0xF5 |

**四、单片机与模块通讯代码**

**1）功能函数**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*功能：返回信息处理

\*\*参数： cmd 不同命令不同处理

\*\*返回：处理结果

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **Check\_Package**(**unsigned** **char** cmd)

{

**unsigned** **char** flag = FALSE;

**if**(!WaitFpData()) **return** flag; //等待接收返回信息

p = 0 ;

**if**(g\_ucUartRxEnd)

g\_ucUartRxEnd = 0;//清数据包接收标志

**else**

**return** flag;

**if**(rBuf[0] != DATA\_START)**return** flag;

**if**(rBuf[1] != cmd)**return** flag;

**if**(rBuf[6] != CmdGenCHK(g\_ucUartRxLen - 3, &rBuf[1]))**return** flag;

**switch**(cmd)

{

**case** CMD\_ENROLL1:

**case** CMD\_ENROLL2:

**case** CMD\_ENROLL3:

**if**(ACK\_SUCCESS == rBuf[4])flag = TRUE;

**else** **if**(ACK\_USER\_EXIST == rBuf[4])

{

// Spk\_HaveUser();

Delay\_ms(1500);

}

**break**;

**case** CMD\_DELETE: //删除指定编号指纹

**case** CMD\_CLEAR: //清空所有指纹

**case** CMD\_IDENTIFY: //1:1比对

**if**(ACK\_SUCCESS == rBuf[4])flag = TRUE;

**break**;

**case** CMD\_USERNUMB: //取用户总数

**if**(ACK\_SUCCESS == rBuf[4])

{

flag = TRUE;

l\_ucFPID = rBuf[3];

}

**break**;

**case** CMD\_SEARCH: //1:N比对

**if**((1 == rBuf[4])||(2 == rBuf[4])||(3 == rBuf[4]))

{

flag = TRUE;

l\_ucFPID = rBuf[3];

}

**break**;

**default**:

**break**;

}

**return** flag;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*功能：等待数据包发送完成

\*\*参数：

\*\*返回：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **WaitFpData**(**void**)

{

**unsigned** **char** i;

**for**(i=200; i>0; i--)//等待指纹芯片返回

{

Delay\_ms(40);

**if**(g\_ucUartRxEnd)**break**;

}

**if**(i==0)**return** FALSE;//指纹芯片没有返回

**else** **return** TRUE;

}

**void** **UartSend**(**unsigned** **char** \*Datar,**unsigned** **char** cLength)

{

**do**

{

**UARTCharPut**(UART6\_BASE,\*(Datar++));//默认用UART6的P0P1脚

} **while** (--cLength != 0);

}

**1）指纹录入**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*注册指纹

\*\*输入两次指纹注册一个指纹模板

\*\*参数：UserID 指纹号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **Enroll\_Step1**(**unsigned** **int** u\_id)

{

**unsigned** **char** buf[BUF\_N];

\*buf = CMD\_ENROLL1;

\*(buf+1) = u\_id>>8;

\*(buf+2) = u\_id&0xff;

\*(buf+3) = 1;

\*(buf+4) = 0x00;

UART\_SendPackage(5, buf);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*注册指纹

\*\*输入三次指纹注册一个指纹模板

\*\*参数：UserID 指纹号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **Finger\_Enroll**(**unsigned** **int** u\_id)

{

Enroll\_Step1(u\_id);

**if**(FALSE == Check\_Package(CMD\_ENROLL1))**return** FALSE;//一

Delay\_ms(200);

Enroll\_Step2(u\_id);

**if**(FALSE == Check\_Package(CMD\_ENROLL2))**return** FALSE; //二

Delay\_ms(200);

Enroll\_Step3(u\_id);//三

Delay\_ms(200);

**return** Check\_Package(CMD\_ENROLL3);

}

**2）指纹删除**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*功能：删除指定编号指纹

\*\*参数：u\_id

\*\*返回：void

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **FP\_Delete**(**unsigned** **int** u\_id)

{

**unsigned** **char** buf[BUF\_N];

\*buf = CMD\_DELETE;

\*(buf+1) = u\_id>>8;

\*(buf+2) = u\_id&0xff;

\*(buf+3) = 0x00;

\*(buf+4) = 0x00;

UART\_SendPackage(5, buf);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*删除指定指纹

\*\*

\*\*参数：UserID 指纹号

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **Finger\_Delete**(**unsigned** **int** u\_id)

{

FP\_Delete(u\_id);

// if(FALSE == WaitFpData())return FALSE;

**return** Check\_Package(CMD\_DELETE);

}

**3）指纹识别**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*功能：以CharBuffer1 或CharBuffer2 中的特征文件搜索整个或部分指纹库

\*\*参数：

\*\*返回：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **FP\_Search**(**void**)

{

**unsigned** **char** buf[BUF\_N];

\*buf = CMD\_SEARCH; //1:N比对

\*(buf+1) = 0x00;

\*(buf+2) = 0x00;

\*(buf+3) = 0x00;

\*(buf+4) = 0x00;

UART\_SendPackage(5, buf);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*读取用户总数

\*\*

\*\*参数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**unsigned** **char** **Finger\_Search**(**void**)

{

FP\_Search();

**return** Check\_Package(CMD\_SEARCH);

}

**五、模块使用心得**

第一次使用指纹模块还是十分生疏的，但当你真正掌握了那种使用模块的方法，其他不管是何种模块，只要他有一个较好的通讯协议，使用起来也能比较方便。拿到这种模块你对它一无所知，首先就得去找对应的某宝卖家下载资料，大多数的卖家都会把资料放在介绍页面里，比如一个网盘链接，如果没有就直接问客服要，这个能力是必须的，不如寸步难行。拿到资料后，大致了解他的通讯协议，搞明白它的引脚连接，硬件上也得细心，不然也可能会成为你的绊脚石。一般卖这种模块的开发者都会有一款测试的上位机，通过电脑与模块进行交互来确保模块的正常运行。当你上位机的测试也通过后，连接到单片机，这时候应该根据店家提供的通讯协议自己编写程序吗？答案是否定的，你应该先找一下店家有没有提供例程，即使和你使用的单片机型号不同也没有关系，你只需要改一下通讯的函数就可以了，当然是建立在你已经看懂例程的逻辑的基础上的。一般到这里不出意外模块基本上能使用了，最后再根据你的需求来编写应用层。当然不是一切都一帆风顺的，有时候就是会卡你很久，不妨试试换个模块。

模块化就是提高其使用的灵活性，降低系统的设计难度，模块化的思想在生活中很常见，不仅仅是一个系统，在日后的工作也时刻体现出这个思想，每个人负责一部分，最后再组合起来，变成一个庞大的系统。