AS608 用户手册

——野火指纹识别模块

修订历史

日期	版本	更新内容
2018/5/2	1.0.0	

文档说明

本手册旨在帮助用户正确构建 AS608 模块的使用环境，引导用户快速使用该模块。

关于传输协议与硬件参数请参考文档《AS60x 指纹识别 SOC 用户手册 V10.pdf》。

目录

1. YH-AS608 模块说明

1.1 YH-AS608 简介

YH-AS608 是野火设计的高性能光学指纹识别模块。它采用了杭州晟元芯片技术有限司(Synochip)AS60x 高性能指纹识别芯片，芯片内置 DSP 运算单元并集成了先进的指纹识别算法，具有较高识别精度。模块内部内置了手指探测电路，用户可读取状态引脚(TouchOut)判断有无手指按下。另外模块是通过串口通信向单片机系统和电脑输出指纹识别信息的，使用简单方便，其外观见图 1-1。

图 1-1 YH-AS608 指纹识别模块

1.2 产品特性参数

YH-AS608 模块产品特性参数见表格 1-1。

表格 1-1 YH-AS608 模块产品特性

特性	说明
基本功能	❑ 指纹录入(特征、用户 ID) ❑ 指纹识别 ❑ 指纹库操作（删除指定用户指纹特征等）
工作电压(V)	3.0~3.6V，板载 3.3V
工作电流(mA)	30~60mA
传感器图像大小(pixel)	256*288
图像处理时间(S)	<0.4S
指纹特征检索时间(S)	<0.3(S)
上电延时(S)	<0.1(S),上电后初始化约 0.1S
指纹库容量	300 (ID:0~299)
拒真率(FRR)	<1%
认假率(FAR)	<0.001%
串口（USART）	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)
工作环境	温度(° C):-20~60 ， 湿度<90%

1.3 YH-AS608 模块引脚说明

YH-AS608 模块的引脚说明见图 1-2 及表格 1-2。

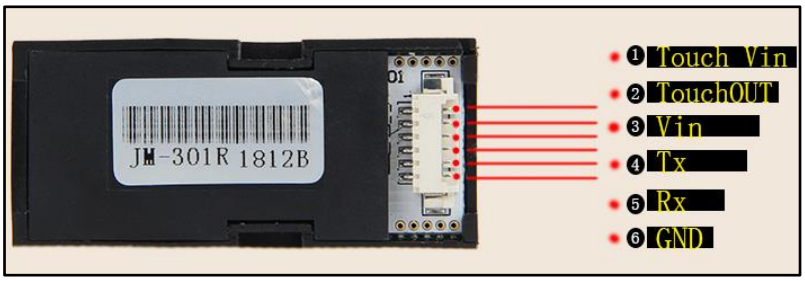

图 1-2 YH-AS608 模块引脚说明

表格 1-2 YH-AS608 模块引脚说明

编号	名称	类型	说明
1	TouchVin	In	手指触摸感应电源输入端，3.3V 供电
2	TouchOut	Out	手指触摸感应信号输出，默认高电平有效
3	Vin	-In	模块 电源正输入端
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

1.4 模块资源

在指纹模块内部内置了 DSP 运算模块和 FLASH 存储区。特别的，芯片内设有一个 72K 字节的图像缓冲区与二个 512 bytes(256 字)大小的特征文件缓冲区，名字分别称为：ImageBuffer，CharBuffer1，CharBuffer2。用户可以通过指令读写任意一个缓冲区。CharBuffer1 或 CharBuffer2 既可以用于存放录取指纹时生成的特征文件，也可以用于存放录取指纹后所生成模板的特征文件。通过 UART 口上传或下载图像时为加快速度，只用到像素字节的高四位，即将两个像素合成一个字节传送。指纹库容量根据挂接的 FLASH 容量不同而改变，系统会自动判别。指纹模板按照序号存放，序号定义为：0—N-1（N 指指纹库容量，这里 N=300）。用户只能根据序号访问指纹库内容。

YH-AS608 指纹识别模块性能稳定，模块配备了串口通信接口，由于是内置了算法处理程序，用户不需要深入了解算法处理原理。只需按照模块既定的通信协议，通过串口向模块发送指令，就可以对指纹模块进行控制和操作了。

2. 上位机硬件测试

使用电脑来测试 YH-AS608 模块非常方便，为此，我们首先要准备好软硬件环境。主要是准备好 USB 转串口 TTL 模块以及安装和打开 SYDemo.exe 测试软件。

测试时需要使用 USB 转串口 TTL 模块连接电脑与 YH-AS608 模块，然后使用 SYDemo 上位机进行指令输入控制。

2.1 安装 USB 转串口驱动

YH-AS608 模块使用串口通讯，且通讯引脚电平为 TTL 类型，所以与电脑通讯时需要使用 USB 转串口 TTL 的串口线作为媒介，在各款开发板上都集成了 USB 转串口的 USB 口。在使用时直接用 USB 线连接到电脑上即可。

该 USB 转串口的驱动芯片为 CH340，在使用前需要给电脑安装好驱动，在 YH-AS608 模块配套资料压缩包的“配套软件”文件夹下可找到该驱动的安装文件，该驱动支持 XP、WIN7、WIN10，非 XP 用户使用用管理员身份安装即可。

安装完成后，把 USB 线接入电脑，如果 USB 转串口驱动安装成功，那么可在计算机->管理->设备管理器->端口中可查看识别到串口。

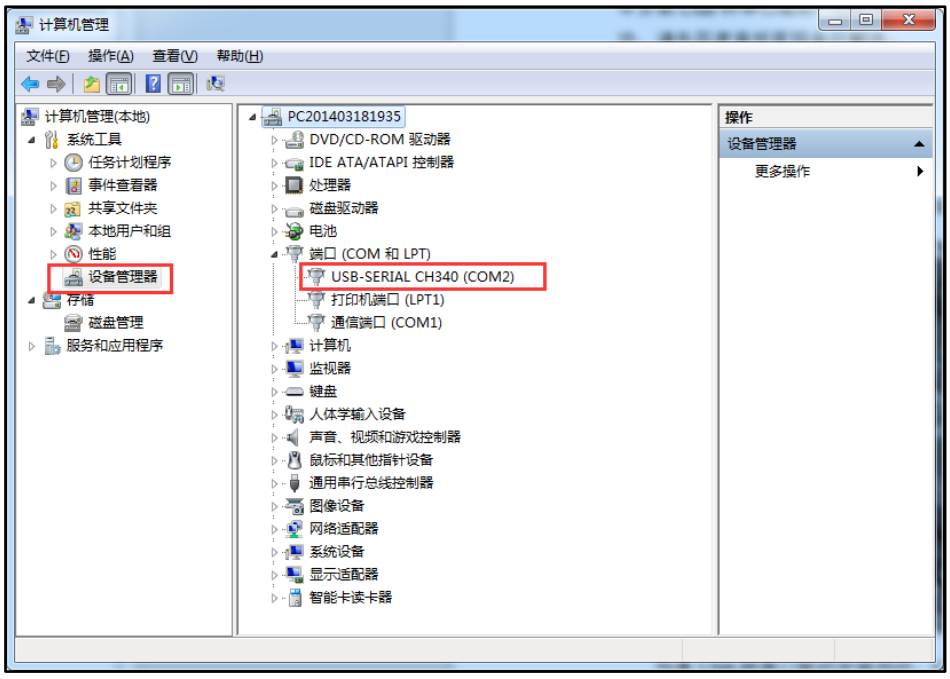

图 2-1 USB 转串口驱动安装成功

如果识别不了串口，请接到电脑的另一个 USB 接口，并重新安装驱动。

2.2 硬件电路连接

指纹识别模块通信使用的是 6 针杜邦线，如，在上位机硬件测试中仅使用 5 根线即可（指纹模块 2 号引脚空闲），在接线之前，先要把开发板上 USB 转串口排针处跳帽全部摘除（全系列开发板都一样）。接线见表格 2-1；

表格 2-1 野火开发板全系列与 YH-AS608 模块连接

全系列开发板	YH-AS608 模块	引脚编号
5V/3.3V	TouchVin	1
空	TouchOut	2
5V/3.3V	Vin	3
RXD	Tx	4
TXD	Rx	5
GND	GND	6

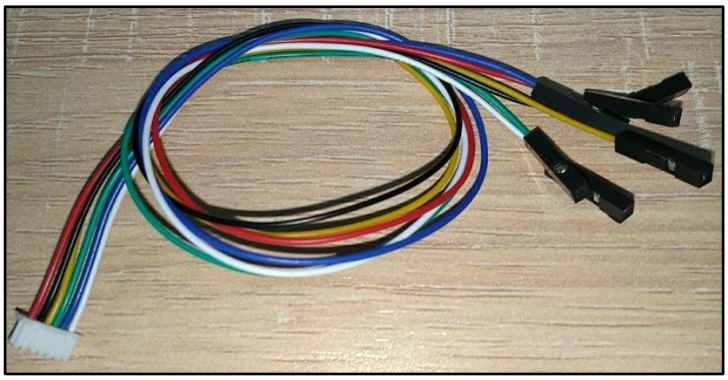

图 2-2 6 针杜邦线

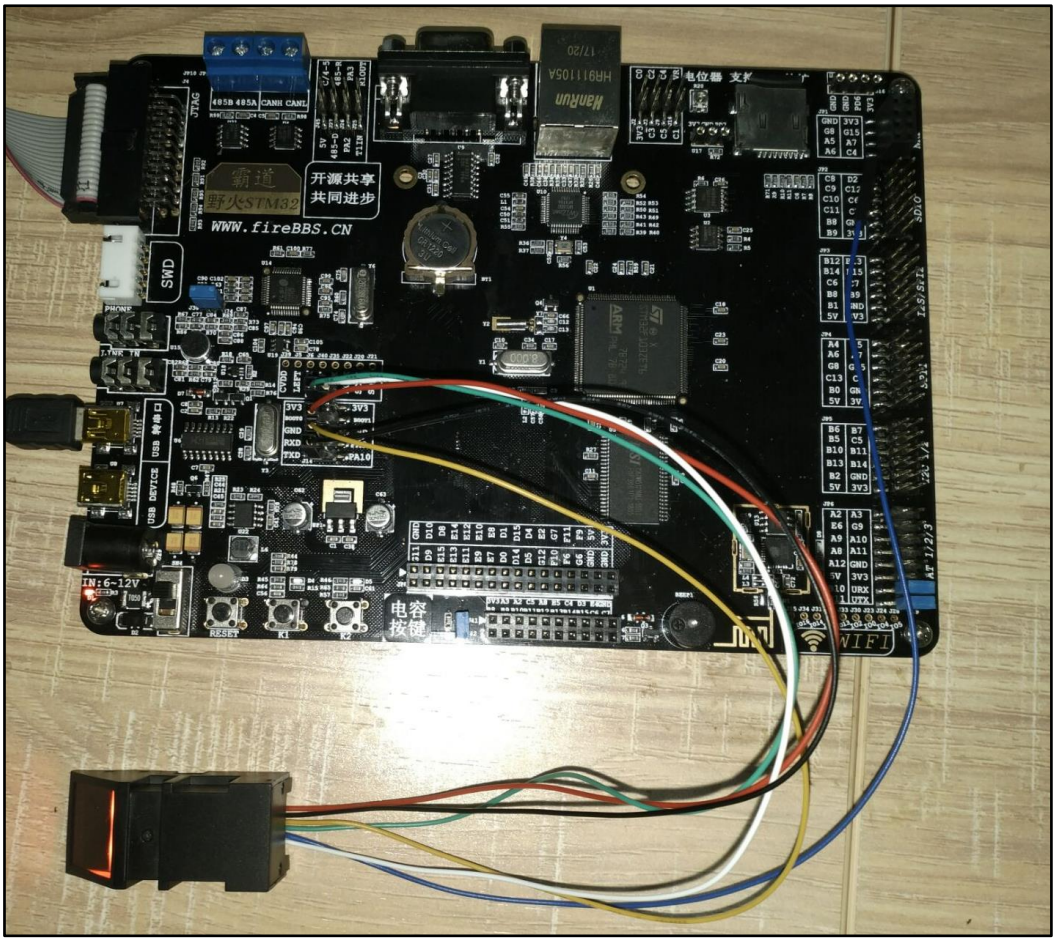

图 2-3 F103 霸道系列开发板接线图

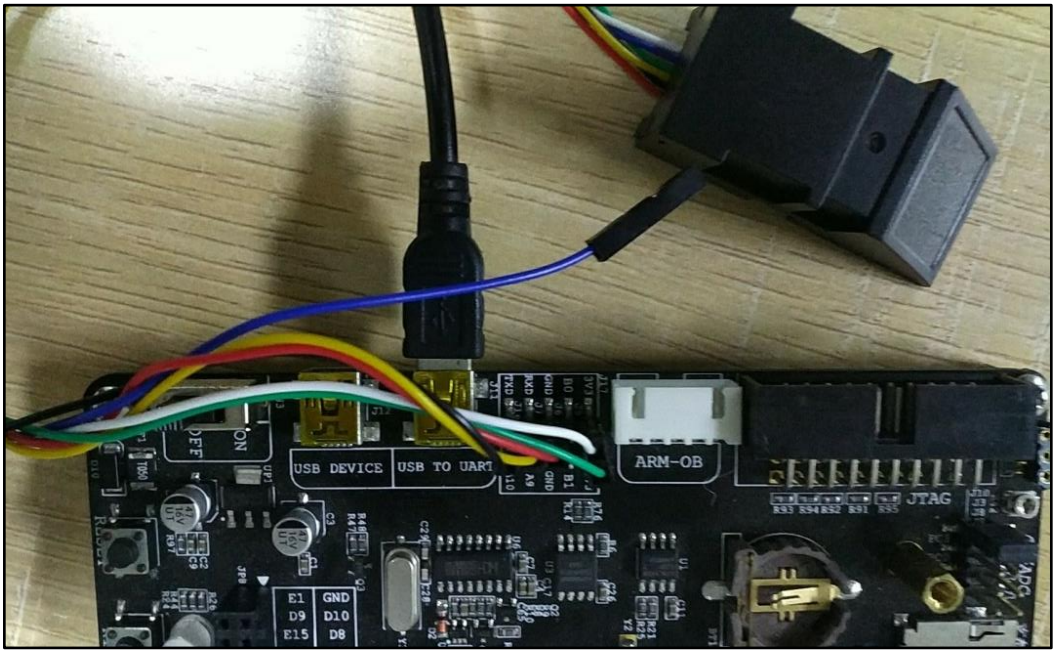

图 2-4 F103 指南者系列开发板接线图

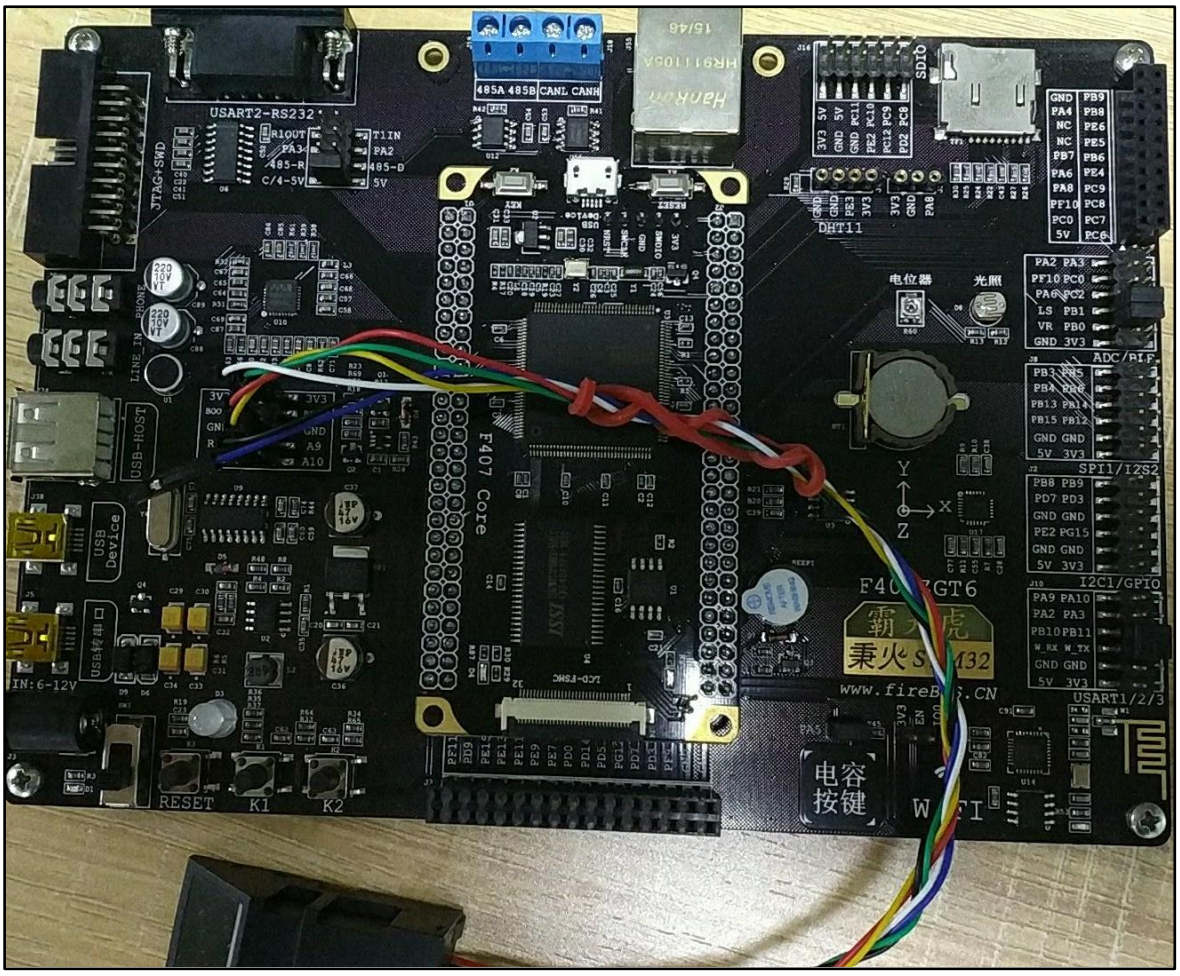

图 2-5 F407 霸天虎系列开发板接线图

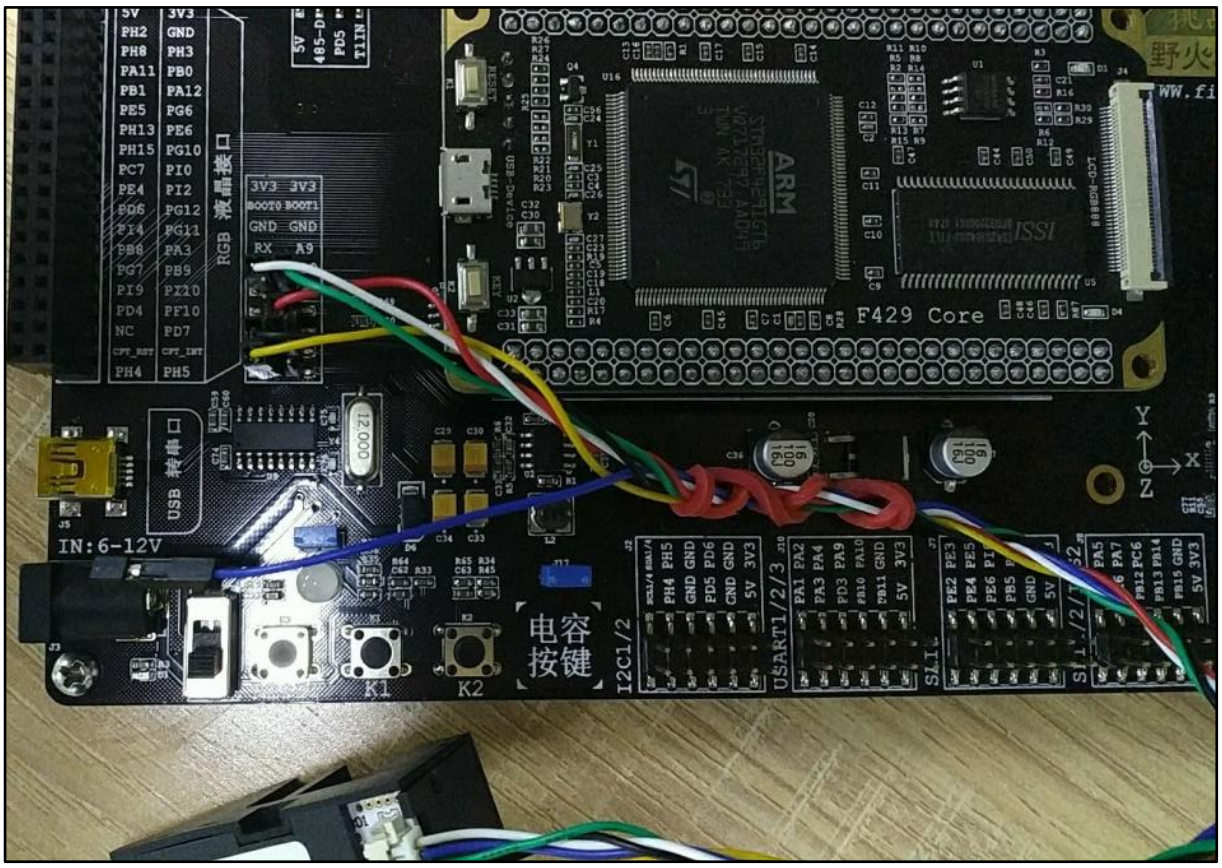

图 2-6 F429 挑战者系列开发板接线图

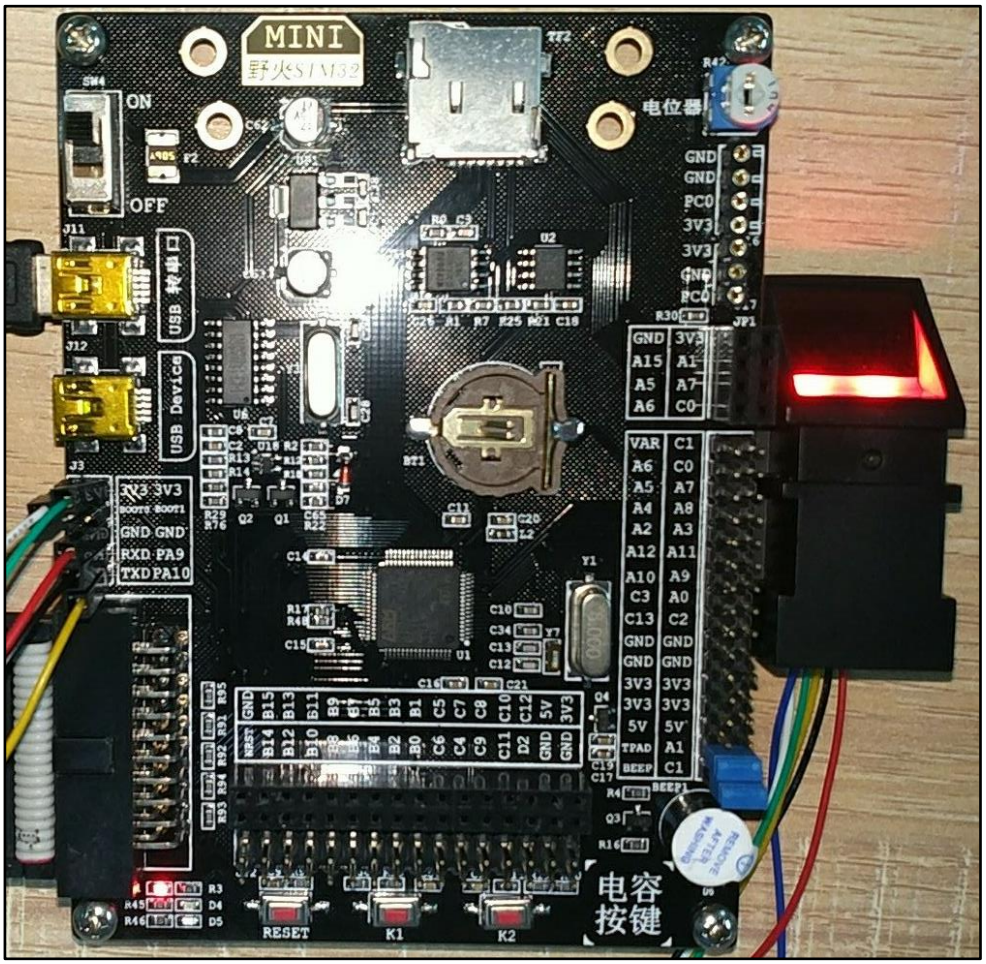

图 2-7 F103 MINI 系列开发板接线图

2.3 安装 SYDemo 上位机与测试

2.3.1 软件简介

为了方便用户调试及使用 YH-AS608 模块，野火提供了多功能测试软件。配合 YH-AS608 模块，该软件能显示指纹识别模块通过 USB 转 TTL 线发送和传回的原始信息。该软件是绿色免安装的，直接打开即可使用。

图 2-8 上位机界面

2.3.2 使用方法

在模块配套资料包的“配套软件”文件夹可找到 SYDemo.exe 安装软件包，安装后打开该软件即可看到测试软件界面见图 2-8。把 YH-AS608 模块通过 USB 转串口 TTL 线连接到电脑。在测试软件界面，先点击打开设备按钮，然后选择串口号（COM4），即可完成上位机与指纹模块的通信，见图 2-9。

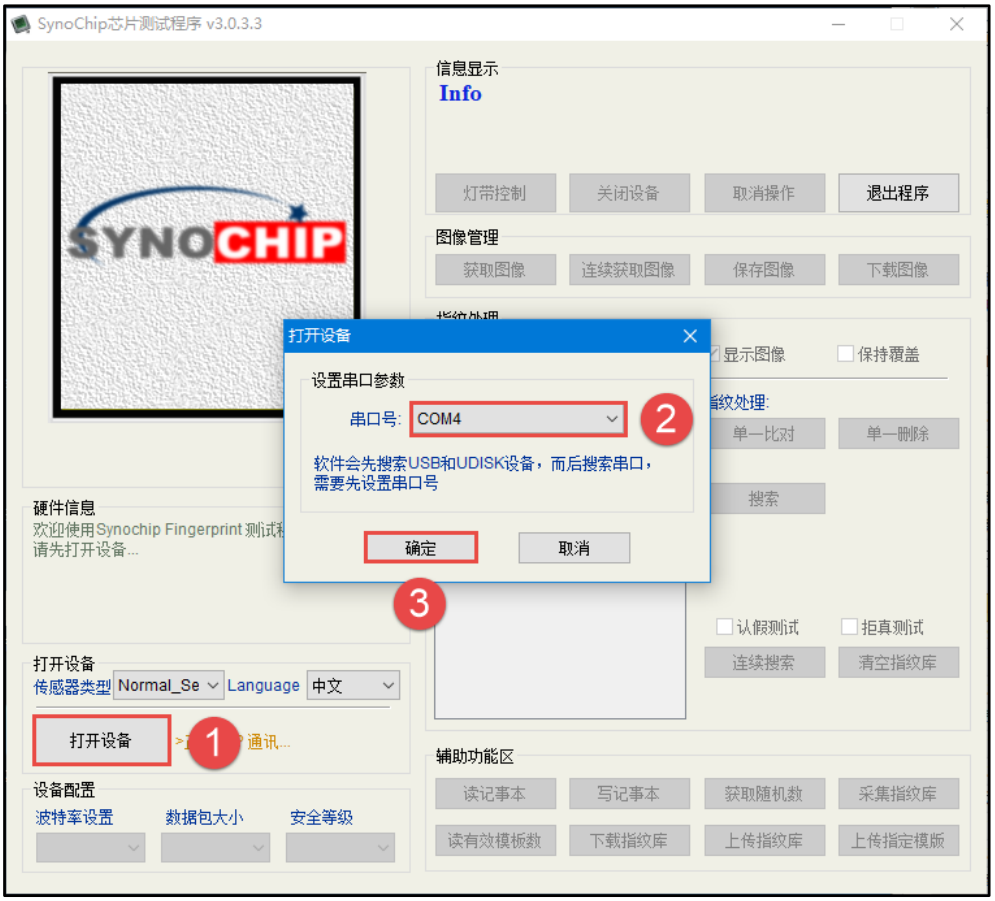

图 2-9 上位机与指纹识别模块通信

使用上位机测试软件可方便地测试 YH-AS608 模块是否正常，具体测试步骤如下：

1. 点击上图“确定”按钮后， 通讯成功如图 2-10 所示，上位机会显示硬件信息、波特率等参数。

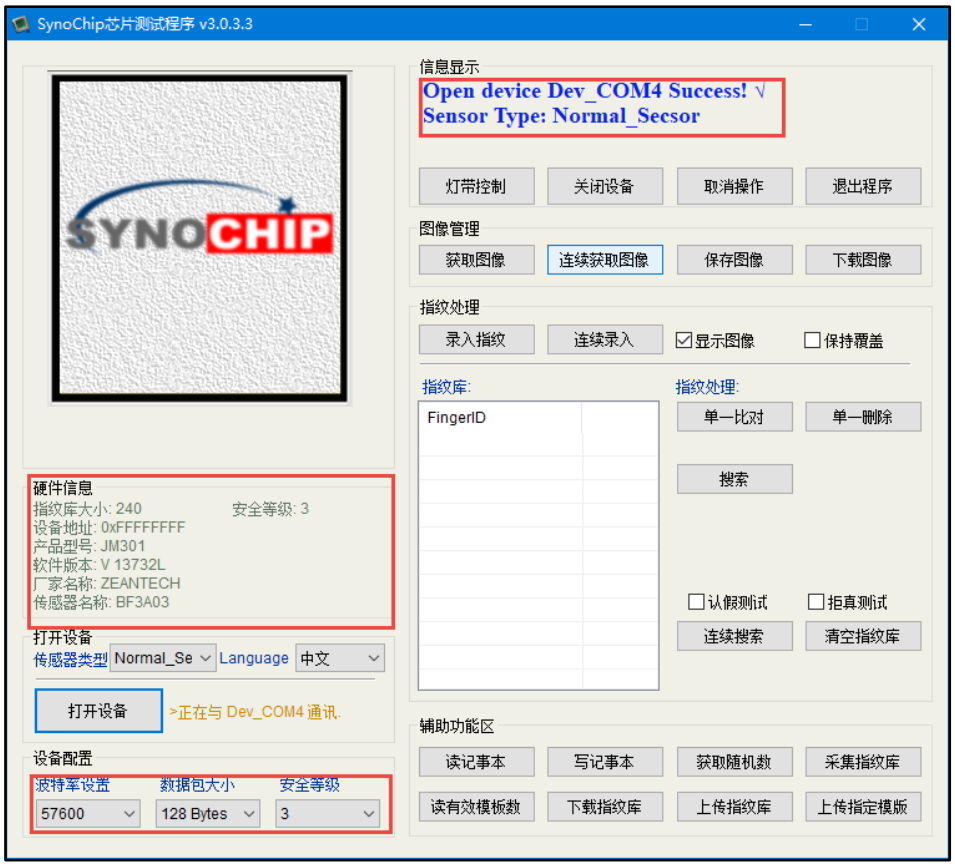

图 2-10 上位机参数显示界面

2. 下面我们来进行录指纹操作，点击“录入指纹”按钮，然后输入指纹储存的 ID 编号，如图 2-11 所示。

图 2-11 录指纹操作

3. 点击“OK”后，上位机会提示“请将手指平放在传感器上”的信息。如图 2-12。

图 2-12 录指纹指令

4. 将手指平放到传感器采集指纹信息。等待指纹信息上传。上传时间会稍久，如图 2-13。当第一次录入图像成功后，上位机会显示指纹特征，并会发出“2.请将手指平放在传感器上”的信息，如图 2-14。

图 2-13 第一次录入指纹

图 2-14 第二次录指纹指令

5. 当第二次指纹录入且两次指纹对比匹配成功后，上位机会提示用户添加成功和指纹 ID 编号，如图 2-15。

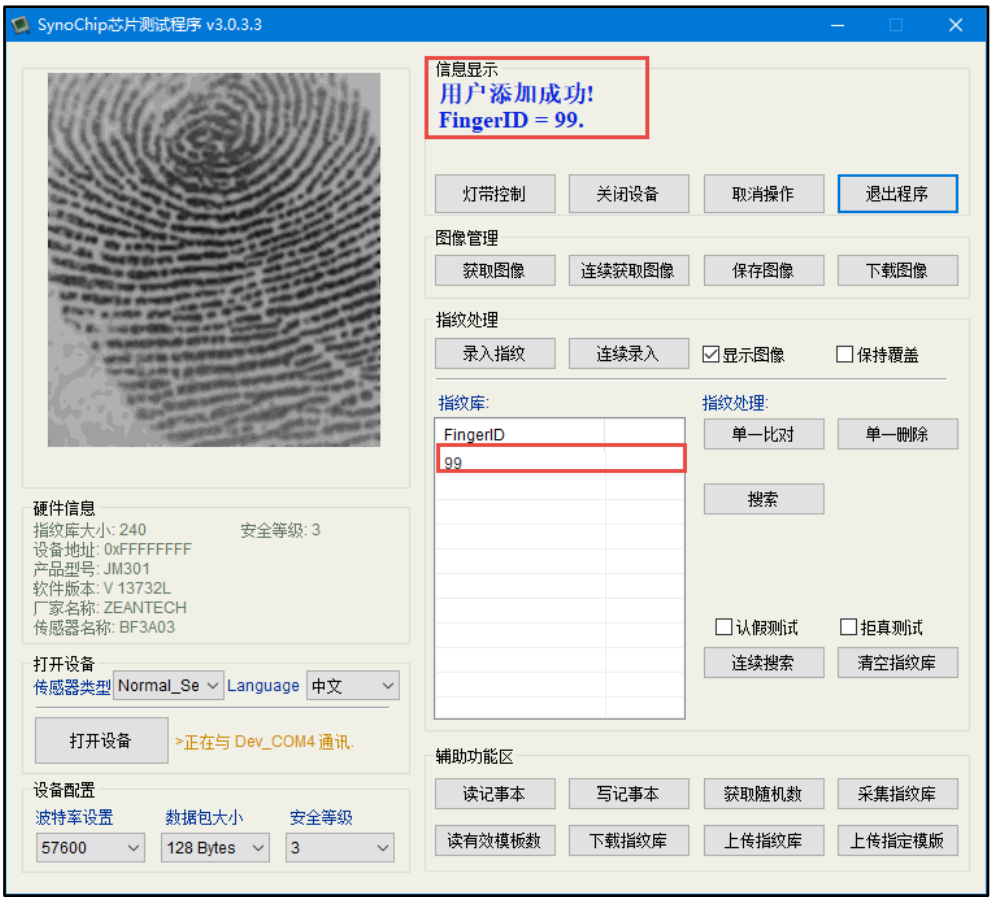

图 2-15 录指纹成功

6. 上面进行了录指纹操作，现在我们来进行验证指纹的操作。点击上位机的“搜索”按钮，而后上位机会提示“请将手指平放在传感器上”，如图 2-16。

图 2-16 比对指纹指令

7. 按照提示将手指放在传感器上录入图像成功之后，系统对比录入图像与指纹库，如果对比成功，则提示“找到相同手指， FingerID= ……”，如图 2-17。

图 2-17 比对指纹成功

说明：如果通讯不成功，可能是接线方式错误，正确方式是模块 Tx、 Rx 分别接到 USB 转串口设备的 Rx、 Tx。 另外，如果更改了模块地址必须更改回默认 OxFFFFFFFF，同时口令也必须是默认值 0，这样才能正常通讯！ 特别注意在使用上位机之前检查是否已经成功安装了 CH340 驱动程序。

3. AS60x 通信协议

YH-AS608 模块通过 TTL 串口输入输出信息，这些信息遵从 AS60x 通信协议。指纹识别模块始终处于从属地位（Slave mode），主机（Host）需要通过不同的指令让模块完成各种功能。主机的指令、模块的应答以及数据交换都是按照规定格式的数据包来进行的。指令只能由上位机下给模块，模块向上位机应答。主机必须按照下述格式封装要发送的指令或数据，也必须按下述格式解析收到的数据包。具体指令可查阅《AS60x 指纹识别 SOC 通讯手册》。

3.1 指令包与数据包格式

指令/数据包共分为三类：

(1) 包标识=01 命令包；

(2) 包标识=02 数据包，且有后续包；

(3) 包标识=08 最后一个数据包，即结束包。

所有的数据包都要加包头：0xEF01。

01 命令包格式：

字节数	2bytes	4bytes	1 byte	2 bytes	1byte	…			2 bytes
名称	包头	芯片地址	包标识	包长度	指令	参数 1	…	参数 n	校验和
内容	0xEF01	xxxx	01	N=					

02 数据包格式：

字节数	2bytes	4bytes	1 byte	2 bytes	N bytes	2 bytes
名称	包头	芯片地址	包标识	包长度	数据	校验和
内容	0xEF01	xxxx	02			

08 结束包格式：

字节数	2bytes	4bytes	1 byte	2 bytes	N bytes	2 bytes
名称	包头	芯片地址	包标识	包长度	数据	校验和
内容	0xEF01	xxxx	08			

说明：

1. 数据包不能单独进入执行流程，必须跟在指令包或应答包后面。

2. 下传或上传的数据包格式相同。

3. 包长度 = 包长度至校验和（指令、参数或数据）的总字节数，包含校验和，但不包含包长度本身的字节数。

4. 校验和是从包标识至校验和之间所有字节之和，超出 2 字节的进位忽略。

5. 芯片地址在没有生成之前为缺省的 0xFFFFFFFF，一旦上位机通过指令生成了芯片地址，则所有的数据包都必须按照生成的地址收发。芯片将拒绝地址错误的数据包。

6. 对于多字节的高字节在前低字节在后 （如 2bytes 的 00 06 表示 0006，不是 0600）。

3.2 指令应答格式

应答是将有关命令执行情况与结果上报给上位机，应答包含有参数，并可跟后续数据包。上位机只有在收到 SOC 的应答包后才能确认 SOC 收包情况与指令执行情况。

应答包格式：

2 bytes	4bytes	1 byte	2 byte	1 bytes	N bytes	2 bytes
0xEF01	芯片地址	包标识 07	包长度	确认码	返回参数	校验和

4. 使用单片机系统控制 YH-AS608 模块

4.1 通用控制说明

YH-AS608 支持 TTL 电平的串口通讯标准，非常方便使用单片机系统来控制。本小节以野火 STM32 开发板为例子说明如何使用 STM32 来控制 YH-AS608 模块。

单片机系统通过串口与 YH-AS608 模块通讯，与模块连接时，只要通过模块引出的排针连接好如下六根线即可，注意**模块与单片机的收发引脚要交叉连接**，见。

表格 4-1 单片机系统与 YH-AS608 接线

单片机系统	YH-AS608 模块
5V 或 3.3V	Vin
GND	GND
串口 RXD	Tx
串口 TXD	Rx
5V 或 3.3V	TouchVin
I/O 引脚	TouchOut

4.2 野火 STM32 开发板控制说明

YH-AS608 模块配套有适用于野火 STM32 开发板的源码，用户可以参考它来编写自己的应用。

4.2.1 连接模块

1.与野火 F103 霸道、F103MINI、F407 霸天虎开发板的连接

野火 F103 霸道、F103MINI、F407 霸天虎板子配套的例程，都是通过 STM32 的 USART2 外设控制 YH-AS608 模块的，连接引脚说明见表格 4-2,表格 4-3。

表格 4-2 野火 F103 霸道、MINI 开发板与 YH-AS608 模块连接

F103 霸道/F103 MINI	YH-AS608 模块	引脚编号
5V/3.3V	TouchVin	1
PA8	TouchOut	2
5V/3.3V	Vin	3
PA3/USART2_RX	Tx	4
PA2/USART2_TX	Rx	5
GND	GND	6

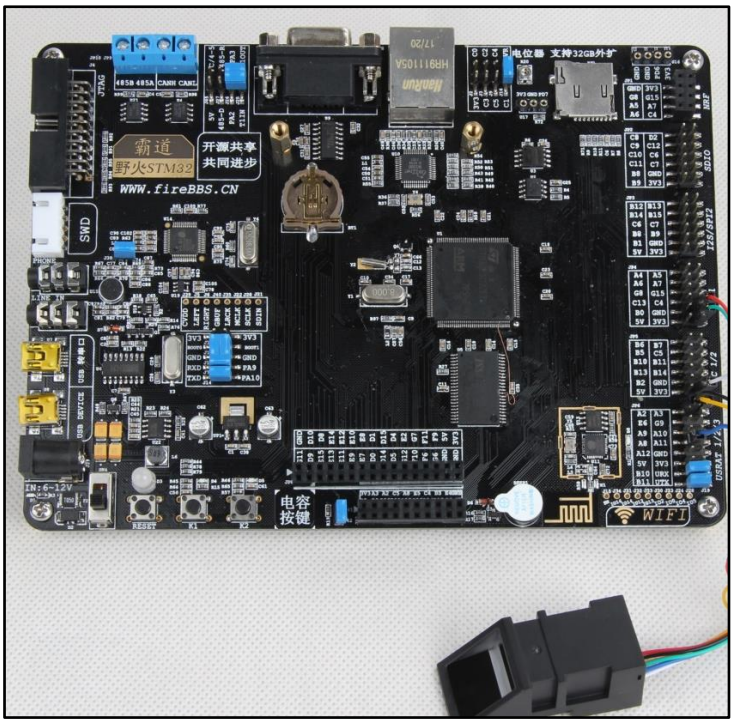

图 4-1 野火 F103 霸道开发板与 YH-AS608 模块连接

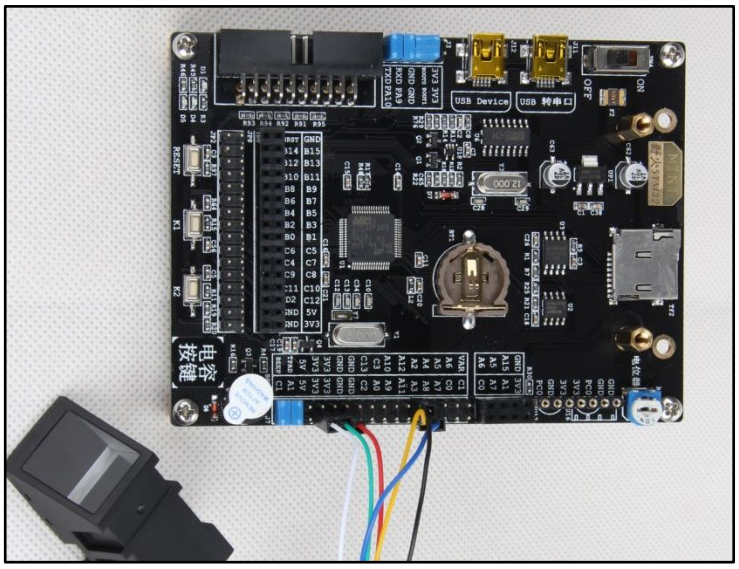

图 4-2 野火 F103 MINI 开发板与 YH-AS608 模块连接

表格 4-3 野火 407 霸天虎开发板与 YH-AS608 模块连接

F407 霸天虎	YH-AS608 模块	引脚编号
5V/3.3V	TouchVin	1
PB8	TouchOut	2
5V/3.3V	Vin	3
PA3/USART2_RX	Tx	4
PA2/USART2_TX	Rx	5
GND	GND	6

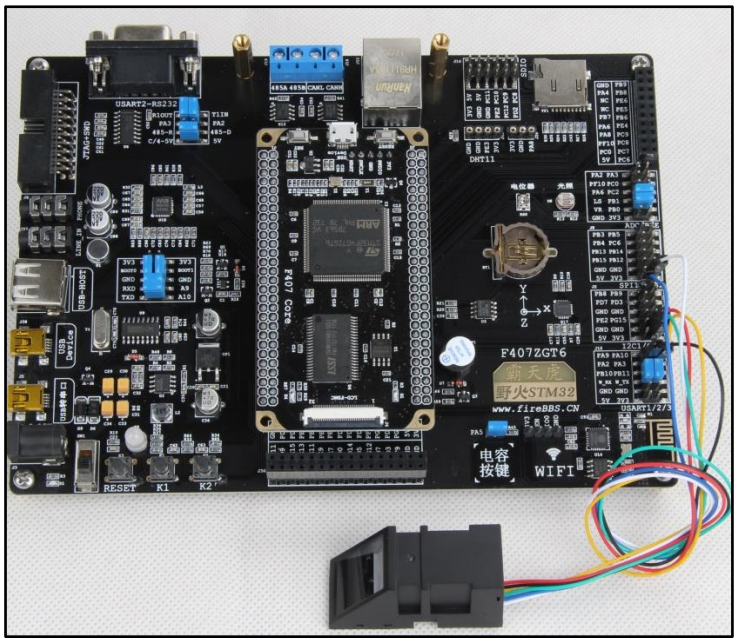

图 4-3 野火 F407 霸天虎开发板与 YH-AS608 模块连接

注意：由于 F103 霸道及 F407 霸天虎开发板的 PA3/USART2_RX 及 PA2/USART2_TX 引脚与开发板上的 MAX3232 串口芯片相连，为了防止引脚共用的影响，请把 F103 霸道及 F407 霸天虎开发板左上角在表格 4-4 中的跳线帽拔掉。另外，在 F103 霸道开发板平台，在使用 YH-AS608 模块的时候不要接入摄像头。

表格 4-4 要摘除的跳帽

编号	丝印
1	R1OUT <---/---> PA3
2	T1IN <---/---> PA2

2.与野火 F103 指南者、F429 挑战者开发板的连接

野火 F103 指南者、F429 挑战者开发板配套的例程，都是通过 STM32 的 USART2 外设控制 YH-AS608 模块的，

表格 4-5 野火 F103 指南者、F429 挑战者开发板与 YH-AS608 模块连接

F103 指南者 /F429 挑战者	YH-AS608 模块	引脚编号
5V/3.3V	TouchVin	1
PA4	TouchOut	2
5V/3.3V	Vin	3
PA3/USART2_RX	Tx	4
PA2/USART2_TX	Rx	5
GND	GND	6

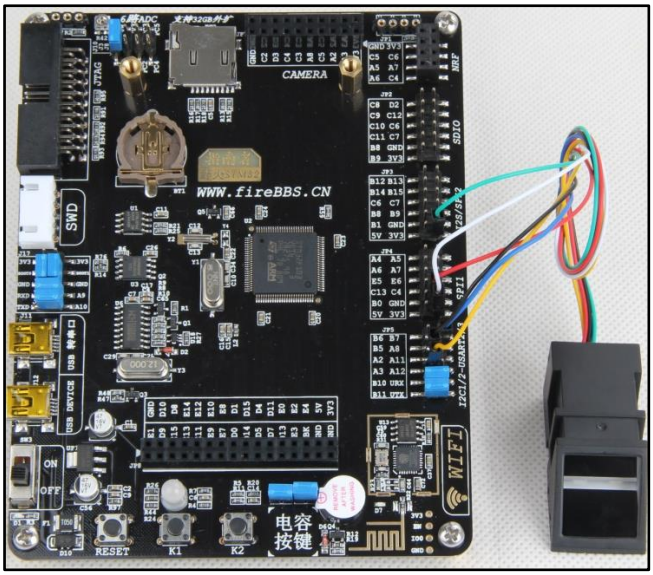

图 4-4 野火 F103 指南者开发板与 YH-AS608 模块连接

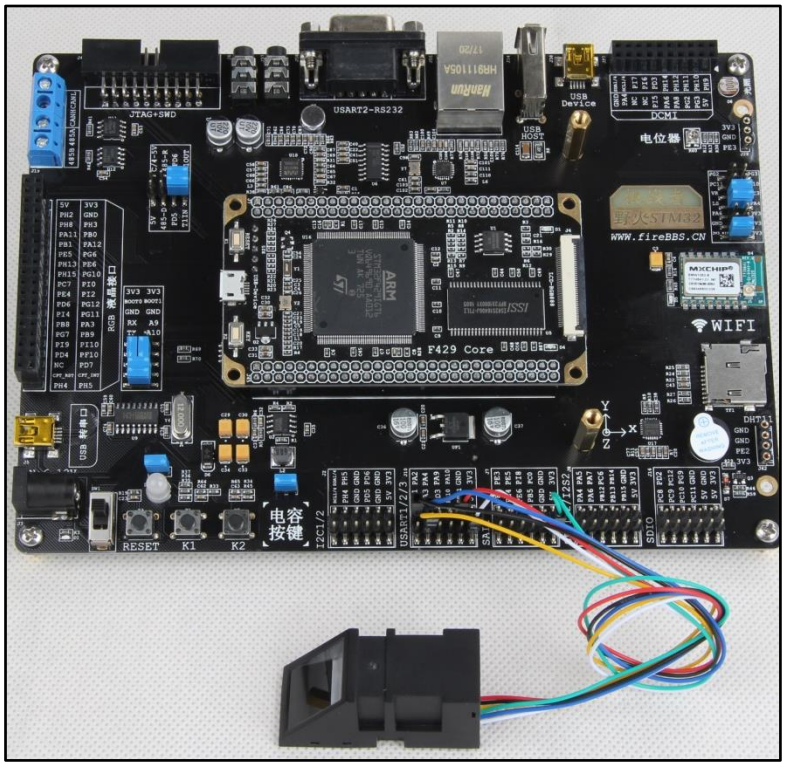

图 4-5 指纹识别模块与 F429 挑战者开发板连接

4.2.2 程序简介

下面以 F103 系列开发板的程序为例进行介绍，F4 的代码类似。

解压野火 YH-AS608 资料后，在如下路径可以找到配套各个开发板的例程：AS608\2-开发板配套例程\ F103-霸道开发板配套例程。各个开发板配套例程的功能和指纹识别模块的驱动是基本一致的，只是不同平台使用的引脚，根据自己使用的开发板，下载对应的程序即可。

4.2.3 实验现象

1.指纹识别实验

当把程序下载到开发板，打开串口调试助手后。串口调试助手会提示操作步骤信息，见图 4-6，按照提示信息即可完成操作实验，实验展示的是，先录入指纹，然后比对指纹，最后清空指纹库，流程做简要介绍，见图 4-7,图 4-9,图 4-9。

特别注意，因为输入数据时调用 scanf 库函数，由于 scanf 函数功能特性，往往需要先输入指令后，在输入窗口打回车键（enter 键），再点击发送，才能将指令发送出去。见图图 4-7，注意光标位置；

图 4-6 指纹识别模块实验串口调试助手初始化界面

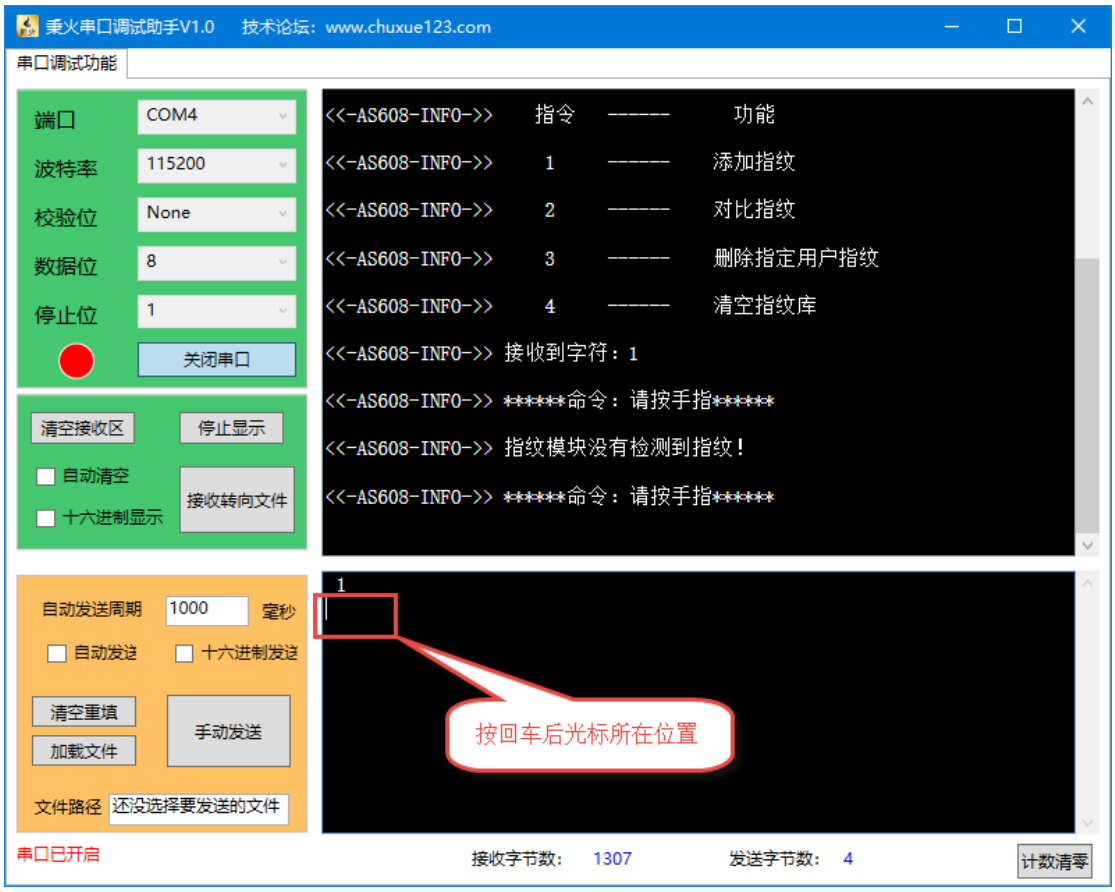

图 4-7 指纹识别模块添加指纹命令

图 4-8 指纹识别模块比对（识别）指纹

图 4-9 指纹识别模块清空指纹库后结果图

5. 代码分析

在本小节中我们将分析如何使用串口调试助手向 STM32 发送控制指令信息，STM32 如何解析调试助手指令和指纹模块反馈的应答信息，以及环形缓冲区的使用。

5.1 实验描述及工程文件清单

5.1.1 实验描述

使用串口调试助手向 STM32 发送命令，STM32 通过解析调试助手发送的命令，向指纹模块发送控制指令，而后等待指纹模块向 STM32 反馈的应答信息，STM32 接收到反馈信息后，通过指令解析向串行助手显示指纹模块的处理结果。该实验使用了 2 个中断，中断 1 使用串口接收中断，采用环形缓冲方式缓存来自指纹模块反馈的应答包。中断 2 使用 EXTI 中断方式，通过指纹模块的 TouchOut 线电压变化检测有无手指按下，发现有手指按下指示灯亮，反之则灭。

5.1.2 主要工程文件

指纹模块实验的主要文件清单如表格 5-1：

表格 5-1 主要文件清单

工程名称	as608
库文件	Libraries
用户编写的文件	USER/main USER/ stm32f10x_it.c USER/as608 USER/led USER/ SysTick USER/ usart

5.2 程序分析

5.2.1 主程序

本例程对一些重要的指纹功能进行了设计，包括指纹录入、指纹比对（识别）、删除指定用户的指纹以及清空指纹库。

代码清单 5-1 主函数（main.c 文件）

```
/**
  * @brief  主函数
  * @param  无
  * @retval 无
  */
int main(void)
{
    /*初始化 USART 配置模式为 115200 8-N-1，中断接收*/
    USART_Config();

    LED_GPIO_Config();

    /*初始化环形缓冲区*/
    rx_queue_init();

    /*初始化指纹模块配置*/
    AS608_Config();


    /*测试 STM32 与指纹模块的通信*/
    Connect_Test();

    while (1) {
        FR_Task();
    }
}
```

主程序中进行串口、LED 灯、环形缓冲区和指纹模块的初始化，其中 Connect_Test()函数用来检测 STM32 与指纹模块的通信，而 FR_Task()函数则是进行指纹模块功能测试。

另外，在配置 as608 时要特别注意，串口通信的中断配置，除了要使能接收中断外，还要使能总线空闲中断，目的是为了使环形缓冲区能完整的把来自指纹识别模块的应打包数据记录下来，防止丢包。

代码清单 5-2 AS608 配置（as608.c 文件）

```
1  /**
2  * @brief  AS608_配置
3  * @param  无
4  * @retval 无
5  */
6  void AS608_Config(void)
7  {
8      GPIO_InitTypeDef  GPIO_InitStructure;
9      EXTI_InitTypeDef  EXTI_InitStructure;
10     USART_InitTypeDef USART_InitStructure;
11
12     /*开启串口 GPIO 口的时钟*/
13     RCC_APB2PeriphClockCmd(AS608_WAK_INT_GPIO_CLK,ENABLE);
14     AS608_USART_GPIO_APBxClkCmd(AS608_USART_GPIO_CLK, ENABLE);
15     /*打开串口外设的时钟*/
16     AS608_USART_APBxClkCmd(AS608_USART_CLK, ENABLE);
17
18     .......
19
20     USART_Init(AS608_USART, &USART_InitStructure);
21
22     /*使能串口接收中断，使能串口总线空闲中断*/
23     USART_ITConfig(AS608_USART, USART_IT_RXNE, ENABLE);
24     USART_ITConfig(AS608_USART, USART_IT_IDLE, ENABLE );
26     /*使能串口*/
27     USART_Cmd(AS608_USART, ENABLE);
29 }
```

5.2.1 中断

在该实验中使用了 2 个中断，一个是 STM32 USART 串口接收中断,用以接收来自指纹识别模块反馈的应答包，并将数据缓存到环行缓冲区中，QUEUE_DATA_TYPE 是一类环形数据结构体类型。环形缓冲区，顾名思义该缓冲区是环形结构，就是缓冲区利用写指针不断缓存接受到的数据时，当写指针访问到缓冲区的最后一个内存位置的时候，当来新的数据需要缓存的时候，写指针回到环形缓冲区的起点位置。缓存过程就类似一个环一样。代码 data_p->head 中是指向环形缓冲区的写指针的头位置，从头指针位置开始，每当缓冲区缓存一个字节，写指针向下偏移一个位置，如此一个一个缓存应答包数据（第 40 行）。

另一个是 EXTI 上升沿/下降沿触发中断（第 9 行代码），来一次中断信号灯反转一次：灯初始化配置为熄灭状态，当手指放入指纹识别模块时，会产生一个边沿触发，灯反转（第 11 行代码）变亮，当手指从传感器移开时，再次触发一个边沿中断信号，灯再次反转熄灭，如此来探测传感器上有无手指触摸。另外，串口通信中断优先级高于 EXTI 中断，代码详见代码清单 5-3：

代码清单 5-3 中断函数（F1 系列 stm32f10x_it.c 文件，F4 系列 stm32f4xx_it.c 文件）

```
1  /**
2    * @brief  TOUCHOUT 引脚 EXTI 中断
```

```
  * @param  None
  * @retval None
  */
void AS608_TOUCHOUT_IRQHandler(void)
{
    /*确保是否产生了 EXTI Line 中断*/
    if (EXTI_GetITStatus(AS608_TOUCHOUT_INT_EXTI_LINE) != RESET) {
        /*LED 反转*/
        LED1_TOGGLE;

        EXTI_ClearITPendingBit(AS608_TOUCHOUT_INT_EXTI_LINE);
    }
}

/**
  * @brief  串口中断服务函数,把接收到的数据写入缓冲区,
            在 main 函数中轮询缓冲区输出数据。
  * @param  None
  * @retval None
  */
void AS608_USART_IRQHandler(void)
{
    uint8_t ucCh;
    QUEUE_DATA_TYPE *data_p;

    if (USART_GetITStatus(AS608_USART,USART_IT_RXNE)!=RESET) {
        ucCh  = USART_ReceiveData( AS608_USART );

        /*获取写缓冲区指针，准备写入新数据*/
        data_p = cbWrite(&rx_queue);

        if (data_p != NULL) { /*若缓冲队列未满，开始传输*/

            /*往缓冲区写入数据，如使用串口接收、dma 写入等方式*/
            *(data_p->head + data_p->len) = ucCh;

            if ( ++data_p->len >= QUEUE_NODE_DATA_LEN) {
                cbWriteFinish(&rx_queue);
            }
        } else return;
    }
    /*数据帧接收完毕*/
    if ( USART_GetITStatus( AS608_USART, USART_IT_IDLE ) == SET )
{
        /*写入缓冲区完毕*/
        cbWriteFinish(&rx_queue);
        /*由软件序列清除中断标志位(先读 USART_SR，然后读 USART_DR)*/
        ucCh = USART_ReceiveData( AS608_USART );
    }
}
```

5.2.2 指纹测试程序

在指纹测试中可以实现多种测试的功能，由于测试程序中，录指纹的过程最为复杂，且流程较多，这里我们仅介绍录指纹过程。在录指纹过程中，秉火串口调试助手会按照流程提示引导用户进行操作，具体流程见流程图：

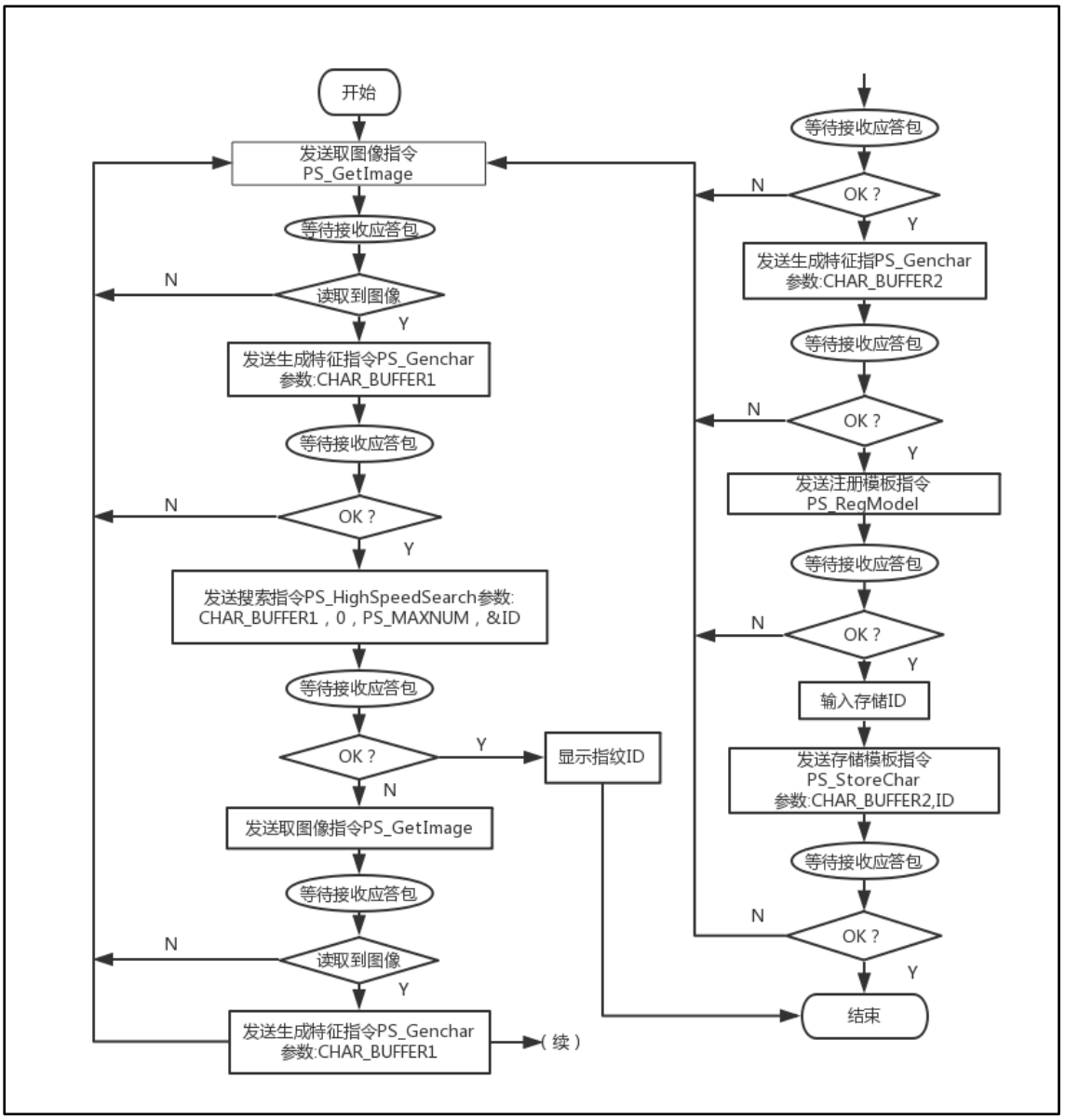

图 5-1 录指纹流程图

根据流程图，在指纹库不存在用户指纹模版时，录指纹过程可以分为以下几个步骤：

(1) 录取 2 次指纹，并分别将 2 次录指纹所生成的指纹特征存放于 CharBuffer1 和 CharBuffer2 中；

(2) 精确比对这两次指纹特征；

(3) 比对成功后,合并特征并生成用户指纹模版；

(4) 输入用户指纹存储 ID，保存指纹模版。

如果指纹库中已经存在某用户的指纹模版，再次录入该用户指纹模版，这样会浪费指纹库容量，所以在录第一次指纹后，先比对（识别）指纹，搜索指纹库中是否存在用户指纹模版，如果存在就结束录指纹的程序，不存在就继续完成录取指纹操作。另外，在录指纹特征的某些特殊情况下，比如一段时间内未识别到指纹或者录取指纹过程连续失败 4 次，程序就会退出录指纹的程序。具体详见代码清单 5-4：

代码清单 5-4 录指纹程序（as608_test.c 文件）

```
1 /**
```

```
  * @brief  录指纹
  * @param  无
  * @retval 无
  */
void  Add_FR(void)
{
    uint16_t  i,j,sure,ID;
    i=j=0;
    while (1)
    {
        switch (j) {
        case 0:                                     /*执行第 1 步*/
            i++;
            AS608_INFO("******命令：请按手指******\r\n");
            sure=PS_GetImage();                     /*录入图像*/
            if (sure == 0x00) {
                sure=PS_GenChar(CHAR_BUFFER1); /*生成特征 1*/
                if (sure==0x00) {
                    AS608_INFO("输入指纹 1 正常！！！\r\n");
                    sure=PS_HighSpeedSearch(CHAR_BUFFER1,\
                                            0,PS_MAXNUM,&ID);
                    if (sure==0x00) {
                        AS608_INFO ("指纹已存在，指纹 ID：%d！！！\r\n\r\n",ID);
                        return ;
                    } else {
                        i=0;
                        j=1;                        /*跳转到第 2 步*/
                    }
                } else {
                    ShowErrMessage(sure);
                }
            } else {
                ShowErrMessage(sure);
            }
        break;


        case 1:
                i++;
            AS608_INFO("******命令：请再按一次手指******\r\n");
            sure=PS_GetImage();
            if (sure==0x00) {
                sure=PS_GenChar(CHAR_BUFFER2);/*生成特征 2*/
                if (sure==0x00) {
                    AS608_INFO("输入指纹 2 正常！！！\r\n");
                    i=0;
                    j=2;                        /*跳转到第 3 步*/
                } else {
                    ShowErrMessage(sure);
                }
            } else {
                ShowErrMessage(sure);
            }
        break;


        case 2:
            AS608_INFO("正在对比两次输入的指纹\r\n");
            sure=PS_Match();                    /*精确比对两枚指纹特征*/
```

```
59              if (sure==0x00) {
60                  AS608_INFO("输入指纹对比成功！！！\r\n");
61                  j=3;                                /*跳转到第 4 步*/
62              } else {
63                  AS608_INFO("对比失败，请重新录入指纹！！！\r\n");
64                  ShowErrMessage(sure);
65                  i=0;
66                  j=0;
67              }
68          break;
69
70
71          case 3:
72              AS608_INFO("正在生成指纹模块\r\n");
73              sure=PS_RegModel();                 /*合并特征（生成模板）*/
74              if (sure==0x00) {
75                  AS608_INFO("生成指纹模块成功！！！\r\n");
76                  j=4;                               /*跳转到第 5 步*/
77              } else {
78                  j =0;
79                  ShowErrMessage(sure);
80              }
81          break;
82
83
84          case 4:
86              do {
87                  AS608_INFO (
88                  "******命令：请输入存储 ID，范围为 0—299******\r\n");
89                  ID=GET_NUM();
90              } while (!(ID<PS_MAXNUM));
92              sure=PS_StoreChar(CHAR_BUFFER2,ID); /*储存模板*/
93              if (sure==0x00) {
94                  AS608_INFO("录入指纹模块成功！！！\r\n\r\n");
95                  return ;
96              } else {
97                  j =0;
98                  ShowErrMessage(sure);
99              }
100          break;
101          }
102
103
104          SysTick_Delay_Ms(1000);
105          if (i==4) {                        /*超过 4 次没有按手指则退出*/
106              AS608_INFO("录指纹失败！\r\n\r\n\r\n");
107              break;
108          }
109      }
110 }
```

由于录指纹的过程较多，且存在许多判断与跳转，故使用 switch 语句结构，其中每一个 case 代表录指纹过程中一个步骤，在某项操作失败之后，可以跳回原处操作，比如在录入指纹 1 特征失败后，可跳回原处重新录入指纹 1 操作，但不可连续超过 4 次失败（第 105 行代码）。

特别地，当向指纹识别模块发送指令后，我们采用时间延时的方式等待指纹模块的应答响应。应答响应后反馈回来的应答包，则通过环形缓冲的方式将数据缓存起来。最后通

过对缓冲区的特指纹识别数据的读取和解析，并将结果反馈给调试助手，这样我们就可以知道指纹模块的运行状态和处理结果了。代码清单 5-5 是从环形缓冲区读取确认码位的 ReturnFlag()函数代码：

代码清单 5-5　从环形缓冲区读取确认码位（as608_test.c 文件）

```
1  /**
2    * @brief   从缓冲区读出确认码
3    * @param   *i:返回值（确认码）
4    * @retval 无
5    */
6  uint16_t  ReturnFlag( uint16_t *i)
7  {
8      QUEUE_DATA_TYPE *rx_data;
9
10     rx_data = cbRead(&rx_queue);
11     /*从缓冲区读取数据，进行处理*/
12
13     if (rx_data != NULL) {          /*缓冲队列非空*/
14         /*打印环行接收到的数据*/
15         QUEUE_DEBUG_ARRAY((uint8_t*)rx_data->head,rx_data->len);
16
17         i= rx_data->head+9;       /*确认码*/
18
19         cbReadFinish(&rx_queue);
20         /*使用完数据必须调用 cbReadFinish 更新读指针*/
21
22         return *i;
23     } else {
24         *i=0xff;
25         cbReadFinish(&rx_queue);
26         return *i;
27     }
28 }
```

代码中 rx_data->head 是指向环形缓冲区的读指针的头位置，该位置缓存着应答包的第一个数据。当环形缓冲区调用（使用）完毕之后，要重新更新读指针的位置（如第 19 行代码），即上一次写指针的最后位置，作为下一次使用环形缓冲区的头指针。

根据 AS60x 通信协议我们知道我们所需要的确认码位于应答数据包的第 9 位，于是我们从环形缓冲区读取出第 9 位（第 17 行代码）数值，就可以知道指纹处理模块的运行状态了。

另外如果我们想在串口调试助手上打印出应答包的所有数据，我们可以通过启动（如图 5-2）中的打印输出开关 QUEUE_DEBUG_ARRAY_ON（默认为 0，即关闭状态），即可在串口调试助手上打印出应答包数据了。

```
rx_data_queue.h  main.c
40  extern QueueBuffer rx_queue;
41
42
43
44
45  /*信息输出*/
46  #define QUEUE_DEBUG_ON          0
47  #define QUEUE_DEBUG_ARRAY_ON    0
48
49  #define QUEUE_INFO(fmt,arg...)          printf("<<-QUEUE-INFO->> "fmt"\n",##arg)
50  #define QUEUE_ERROR(fmt,arg...)         printf("<<-QUEUE-ERROR->> "fmt"\n",##arg)
51  #define QUEUE_DEBUG(fmt,arg...)         do{\
52                                           if(QUEUE_DEBUG_ON)\
53                                           printf("<<-QUEUE-DEBUG->> [%d]"fmt"\n",__LINE__, ##arg);\
54                                           }while(0)
55
56  #define QUEUE_DEBUG_ARRAY(array, num)   do{\
57                                                  int32_t i;\
58                                                  uint8_t* a = array;\
59                                                  if(QUEUE_DEBUG_ARRAY_ON)\
60                                                  {\
61                                                     printf("\n<<-QUEUE-DEBUG-ARRAY->>\n");\
62                                                     for (i = 0; i < (num); i++)\
63                                                     {\
64                                                         printf("%02x   ", (a)[i]);\
65                                                         if ((i + 1 ) %10 == 0)\
66                                                         {\
67                                                             printf("\n");\
68                                                         }\
69                                                     }\
70                                                     printf("\n");\
71                                                  }\
72                                                  }while(0)
73
74  //输出队列的状态信息
75  #define cbPrint(cb)         DATA_QUEUE_LOG("size=0x%x, read=%d, write=%d\n", cb.size, cb.read, cb.write);\
76      DATA_QUEUE_LOG("size=0x%x, read_using=%d, write_using=%d\n", cb.size, cb.read_using, cb.write_using);
77
78
79  QUEUE_DATA_TYPE* cbWrite(QueueBuffer *cb);
```

图 5-2 信息打印输出控制

6. 常见问题

1. Q:为什么在秉火串口调试助手有时输入指令（如 1）后发送不出去？
   答：因为输入数据时调用 scanf 库函数，由于 scanf 函数功能特性，往往需要先输入指令后，在输入窗口打回车键，才能将指令发送出去。

2. Q:为什么将手指放在指纹识别模块探头周边时，手指探测指示灯也会亮？
   答：由于指纹识别模块探测手指实际上是使用了压力传感器，将手指放在探头周边也会激发压力传感器，所以指示灯也会亮。

7. 联系我们

野火公司官网：http://www.embedfire.com

野火电子论坛：http://www.firebbs.cn

野火淘宝店铺：http://fire-stm32.taobao.com

在学习或使用野火产品时遇到问题可在论坛发帖子与我们交流。