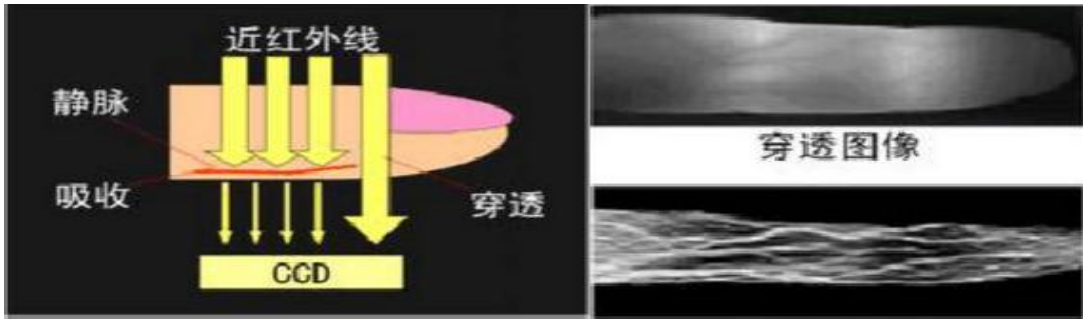
指静脉模块通讯协议

序号	版本	更新日期	更新说明
1	V01	2015.04.21	首次版本
2	V02	2017.04.28	增加上传模板，下载模板

目录

1 指静脉技术原理

指静脉识别技术的基本原理是： 手指中动态流过静脉血管中的血红蛋白， 对波长在700~1000nm 之间的近红外光线有一定得吸收作用，所以当近红外光照射手指后，手指静脉附近透过的近红外光线较少，而其它位置较多，利用 CCD 图形传感器采集透射手指后的近红外光线，将会产生具有独特性质的静脉血管纹路图像。经过算法提取静脉特征值后，将采集 的图像与标准样品特征值进行对比，从而实现了指静脉识别认证的过程。



1.1 主要特性

- 手指内部静脉血管特征，人人可用，活体特征，几乎无法伪造；
- 非接触特征采集，无读头磨损问题；
- 静脉特征采集与静脉识别二合一，集成度高；
- 指静脉模板<4K；
- 1:N 识别 及 1:1 认证功能；
- 可脱机使用，无需主机控制即可自动识别；
- 可适当调节的安全等级；
- 可设置的设备编号；
- 根据手指肌肉和骨骼特性，专业设计近红外光源，自动调节光源强度，很好适应大小粗细不同手指；
- 具有自学习功能，可有效解决由于温度或正常生长造成的血管膨胀和萎缩问题；
- 专利技术，自主静脉识别算法，专用 SOC。

1.2 应用领域：

指静脉门禁控制
指静脉签到、指静脉考勤机等
指静脉锁、指静脉保险柜等
指静脉 POS 终端机等手持设备应

2 使用注意事项

1、使用时手指自然伸直，保证指尖接触到感应器，且整个手指贴合在采集窗口上；不得按压手指，不得手指置于采集窗口内。

2、测试时，手握模块会引起模块误判，请将模块固定在桌面再进行操作；

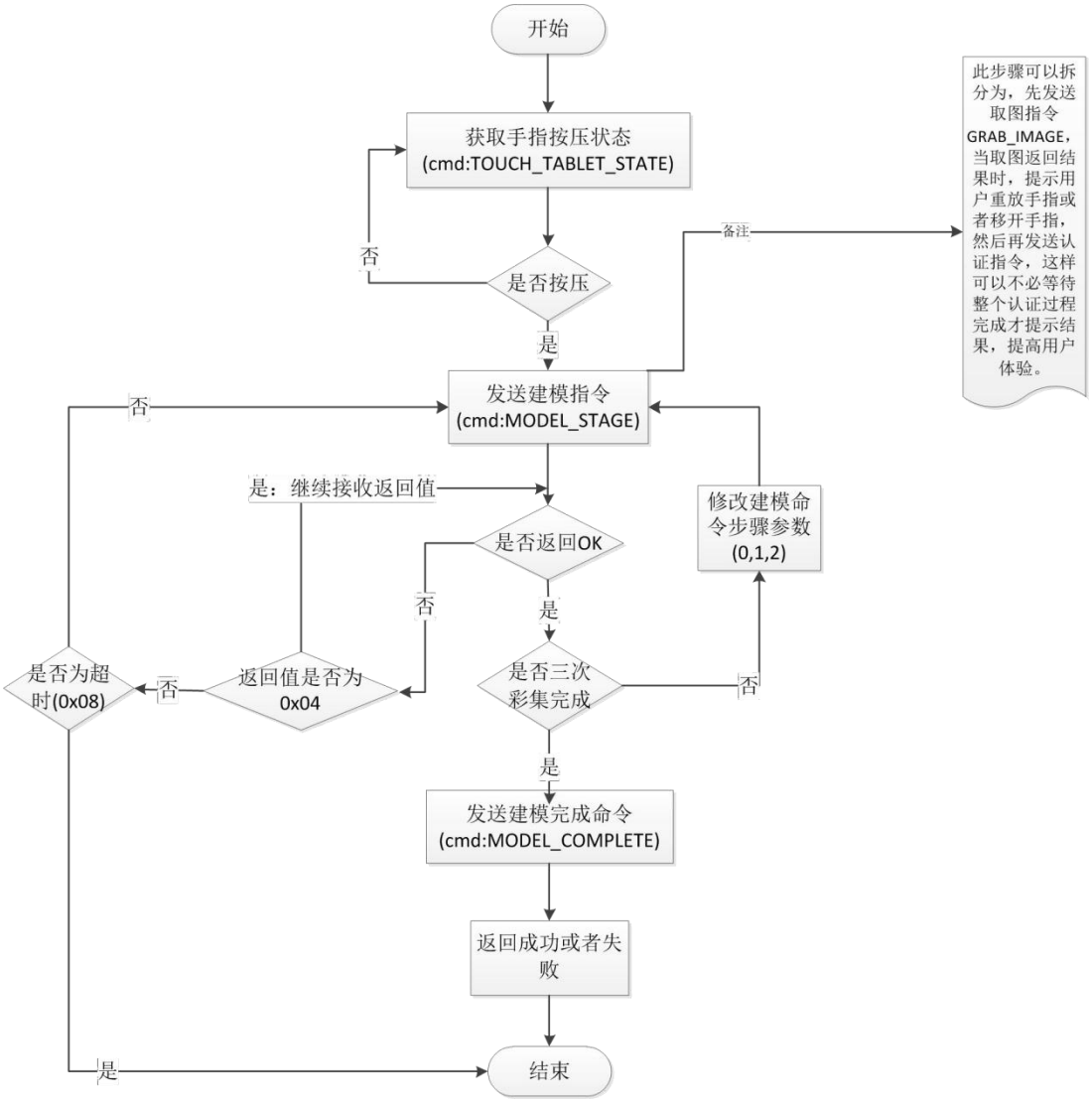
3、建议采集手指为食指，中指，无名指；

4、金属对触摸有很强干扰，请保证模块指尖所对应的整面远离金属 2mm 以上，且手指自然放置时该手指远离金属 2mm 以上。

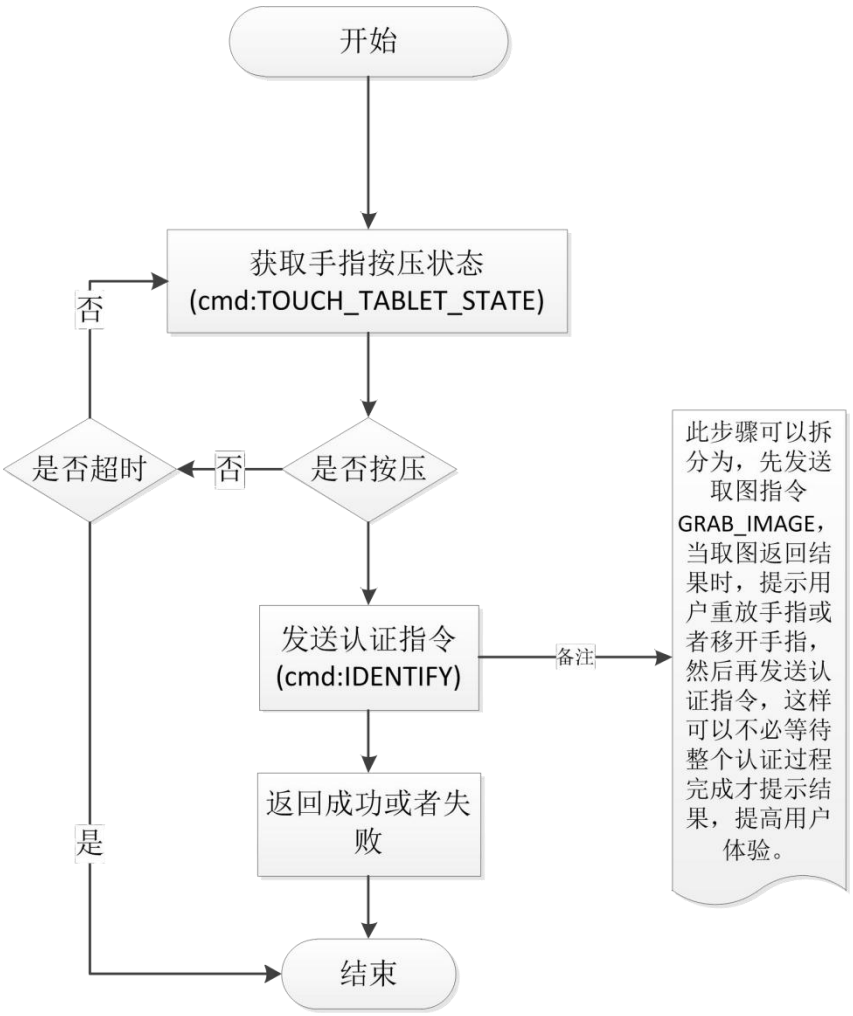
3 设备工作流程和串口调试工具

3.1 设备工作流程简介

3.1.1 建模流程



3.1.2 认证流程


开始
获取手指按压状态
(cmd:TOUCH_TABLET_STATE)
否
是否超时
否
是否按压
发送认证指令
(cmd:IDENTIFY)
备注
此步骤可以拆分为，先发送取图指令GRAB_IMAGE，当取图返回结果时，提示用户重放手指或者移开手指，然后再发送认证指令，这样可以不必等待整个认证过程完成才提示结果，提高用户体验。
返回成功或者失败
是
结束

3.2 串口模式测试 DEMO 使用简介

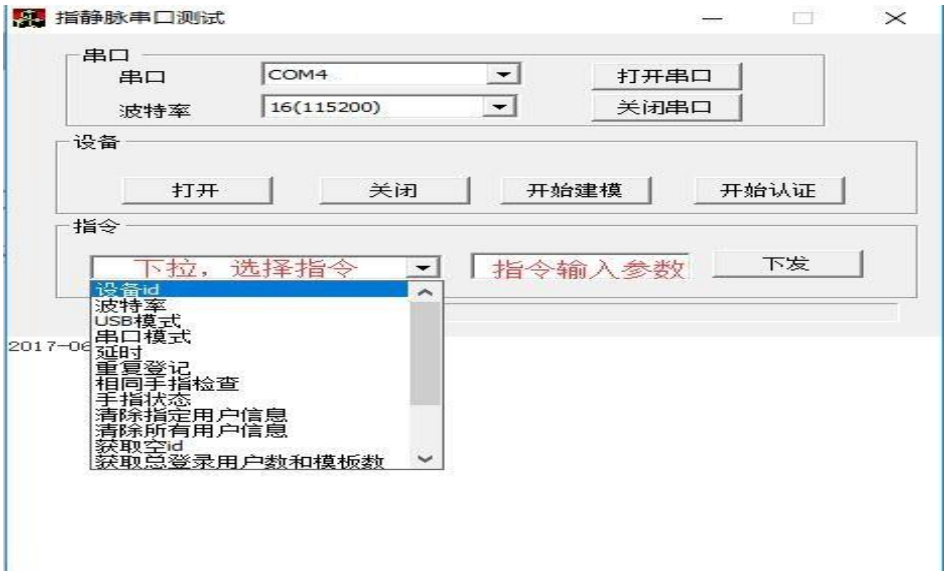
3.2.1 建模

- 选择正确的串口和波特率（默认 115200）
- 打开设备
- 开始建模

3.2.2 认证

- 选择正确的串口和波特率（默认 115200）
- 打开设备
- 开始认证

指令测试



有些指令需要参数，可以在后面的编辑框填入

指令名称	参数
设备 id	设置设备 id,范围(1-255)
波特率	按层分，13 级对应 576000,14 级对应 921600,15 级对应 1000000,16 级对应 1152000
usb 模式	设备转成 usb 模式
串口模式	设备转成串口模式
延时	建模和认证时的超时时间（单位秒,默认 10）
重复登记	是否检查登录的同一个人，默认 1(0,1)
相同手指检查	建模时是否检查是同一个手指，默认 1(0,1)
手指状态	返回当前手指状态
清除指定用户信息	用户 id
清除所有用户信息	
获取空 id	获取一个空id
获取总登录用户数和模板数	返回总登录用户数模板数
获取指定 id 登录信息	返回该 id 登录的模板个数
恢复出厂	
获取设备唯一 ID	每个设备出厂都有一个全球独立的 ID

4 协议定义

4.1 协议格式

串口数据包由包头、数据段、校验和三部分组成。包类型包含以下四种类型：

```
typedef enum
{
    COMMOND_PACKET   = 0x01,//命令包
    DATA_PACKET      = 0x02,//数
    据包RESPOND_PACKET      =
    0x07,//应答包
    DATA_FINISH_PACKET = 0x08 //数据结束包
}packet_type_t;
```

此类型对应协议中的第 6 字节(包类型)，其中命令包和应答包都是固定 24 字节的长度，也就是“数据长度” 域(第 7-8 字节) 固定为 15 ， 它包含了校验和在内， 即填写 0x0F00, 协议采用小端模式， 因此pack[7]=0x0F,pack[8]=0x00。协议中不用的域请填充 0x00。数据包和数据结束包长度不固定，详情请参考以下协议说明。

4.1.1 命令包格式

	包头域					数据域	校验和域
域分类	起始字节	包序号	地址	包类型	数据长度	包含命令，命令参数	checksum
对应字节	[0]-[1]	[2]-[3]	[4]-[5]	[6]	[7]-[8]	[9]-[21]	[22]-[23]
备注	[0]=0x01 [1]=0xEF	[2]=0x00 [3]=0x00	[4]=0xFF [5]=0xFF	命令包计算校验和包含的域范围			

校验和计算方法：

```
unsigned int checksum = 0;
checksum = packet[6] + packet [7] + packet [8];
for (i = 0; i < len - 2; i++)//len 为“数据长度”,len=([8] << 8) + [7]
    checksum += packet [i + 9];
packet [len + 7] = checksum & 0xFF;
packet [len + 8] = checksum >> 8
```

包序号可以默认为 00 00， 也可以从 0 开始递增，应答中会原样返回对应的序号。

4.1.2 应答包格式

	包头域	数据域	校验和域

域分类	起始位	序号	地址	包类型	数据长度	错误码	参数	checksum
对应字节	[0]-[1]	[2]-[3]	[4]-[5]	[6]	[7]-[8]	[9]	[10]-[21]	[22]-[23]
备注				计算校验和				

4.2 参数定义

4.2.1 包类型定义

名称	值
命令包	0x01
数据包	0x02
应答包	0x07
数据结束包	0x08

4.2.2 包命令定义

名称	值
打开设备	0x00
关闭设备	0x01
设置 LED 灯(未用)	0x02
获取触摸状态	0x03
建模	0x04
建模完成	0x05
保留	
认证	0x07
保留	
删除单个模板	0x09
删除所有模板	0x0A
设置超时	0x0E
设置检查重复注册	0x0F
设置检查同一手指	0x10
设置波特率	0x11
查看是否存在指定模板	0x12
取消当前正在执行的动作	0x13
取图	0x14
设置设备 id	0x17
重启设备	0x18
设置安全级别	0x19
查询设备可用的模板 ID	0x1A
恢复出厂设置	0x1C
获取系统信息	0x1D
获取用户注册信息	0x1E
读取信息	0x1F
写入信息	0x20
读取数据	0x21
写入数据	0x22
设置通讯模式(USB 或者串口)	0x28
获取设备唯一 ID	0x39

4.2.3 错误码定义

名称	简称	值
成功	RESULT_OK	0x00
没有找到可用的设备	RESULT_DEVICE_NOT_FOUND	0x01
设备没有打开	RESULT_DEVICE_NOT_OPEN	0x02
设备已经打开	RESULT_DEVICE_OPENED	0x03
指静脉质量评估太差	RESULT_QUALITY_LOW	0x04
放入了不同的手指	RESULT_DIFF_FINGER	0x05
手指已经注册	RESULT_FINGER_ROLLED	0x06
参数错误	RESULT_WRONG_PARA	0x07
超时退出	RESULT_TIMEOUT	0x08
包校验和错误	RESULT_CHECKCHECKSUM_ERROR	0x09
注册的 id 已经存在	RESULT_ID_EXIST	0x0A
数据校验出错	RESULT_DATA_ERROR	0x0B
用户 id 不存在	RESULT_EMPTY_ID	0x0C
还没采图	RESULT_NO_FRAME	0x0D
错误的图片	RESULT_ERROR_IMAGE	0x10
认证失败	RESULT_IDENTIFY_FAIL	0x11
前后两次放置的手指相似度太高	RESULT_HIGH_SIMILARITY	0x12
被取消	RESULT_BE_CANCELED	0x13
特征提取失败	RESULT_EXTRACTION_FAIL	0x14
未知错误	RESULT_UNKOWN_ERROR	0xFF

5 命令数据包详细定义

5.1 打开设备

发送包:

超始字节	包序列号	访问地址	包类型	数据长度	命令	参数	其它数据	校验和
[0]-[1]	[2]-[3]	[4]-[5]	[6]	[7]-[8]	[9]	[10]	[11]-[21]	[22]-[23]
[0]=0x01 [1]=0xEF	[2]=0x00 [3]=0x00	[4]=0xFF [5]=0xFF	[6]=0x01	[7]=0x0F [8]=0x00	[9]=0x00	[10]=0x01	全部填0x00	[22]=0x11 [23]=0x00

校验码 checkchecksum 为从【包类型】到【其它数据】的字节算术和，请参考 7.1.1~7.1.2,以下不再重复说明。

应答包:

超始字节	包序列号	访问地址	包类型	数据长度	结果	其它数据	校验和
[0]-[1]	[2]-[3]	[4]-[5]	[6]	[7]-[8]	[9]	[10]-[21]	[22]-[23]

[0]=0x01 [1]=0xEF	[2]=0x00 [3]=0x00	[4]=0xFF [5]=0xFF	[6]=0x07	[7]=0x0F [8]=0x00	[9]=0x00,OK [9]=0x01,失败	全部为0x00	

成功应答时，校验和[22]=0x11，[23]=0x00

代码示例：

```
void openDev()
{
    int len;
    unsigned char packet [24];
    unsigned char recvData[24];
    unsigned int checkchecksum =
    0;
    //包头
    data[0]=0x0
    1;
    data[1]=0xe
    f;
    //序号
    data[2]=0x0
    1;
    data[3]=0x0
    0;
    //地址
    data[4]=0xf
    f;
    data[5]=0xf
    f;
    //类型
    data[6]= COMMOND_PACKET;
    //长度
    data[7]=0x0
    f;
    data[8]=0x0
    0;
    //命令号
    data[9] = CMD_DEV_OPEN;
    //参数
    data[10]=OFFLINE_M
    ODE;
    data[11]=0;
    //计算校验和
```

```
len = ([8] << 8) + [7];
checksum = packet[6] + packet [7] + packet [8];
for (i = 0; i < len - 2; i++)//len 为“数据长度”,len=([8] << 8) + [7]
    checksum += packet [i + 9];
packet [len + 7] = checksum & 0xFF;
packet [len + 8] = checksum >> 8;
//发送命令
rs232_send(packet);
if(rs232_recv(recvData))
{
    Checksum();
    if (recvData[9] == RESULT_OK)
        { printf("设备打开成功");
        return TRUE;
    } else {
        printf("设备打开失败");
    }
```

```
        }
    }
```

5.2 设备关闭

发送的包：

超始字节	包序列号	访问地址	包类型	数据长度	命令	其它数据	校验和
[0]-[1]	[2]-[3]	[4]-[5]	[6]	[7]-[8]	[9]	[10]-[21]	[22]-[23]
[0]=0x01 [1]=0xEF	[2]=0x00 [3]=0x00	[4]=0xFF [5]=0xFF	[6]=0x01	[7]=0x0F [8]=0x00	[9]=0x01	全部填0x00	[22]=0x11 [23]=0x00

设备回复：

超始字节	包序列号	访问地址	包类型	数据长度	结果	其它数据	校验和
[0]-[1]	[2]-[3]	[4]-[5]	[6]	[7]-[8]	[9]	[10]-[21]	[22]-[23]
[0]=0x01 [1]=0xEF	[2]=0x00 [3]=0x00	[4]=0xFF [5]=0xFF	[6]=0x07	[7]=0x0F [8]=0x00	[9]=0x00,OK [9]=0x01,失败	全部为0x00	

5.3 设备灯的状态

发送的包：

超始字节	包序列号	访问地址	包类型	数据长度	命令	颜色	闪烁	其它数据	校验和
[0]-[1]	[2]-[3]	[4]-[5]	[6]	[7]-[8]	[9]	[10]-[11]	[12]-[13]	[14]-[21]	[22]-[23]
[0]=0x01 [1]=0xEF	[2]=0x00 [3]=0x00	[4]=0xFF [5]=0xFF	[6]=0x01	[7]=0x0F [8]=0x00	[9]=0x01	[10]=0x00:红灯 [10]=0x01:绿灯	[12]=0x00:常亮 [12]=0x01:闪烁	全部填0x00	

注：上表命令中，未作特殊说明的字节都填 0x00，由于word 页宽有限以后的命令说明采用十六进制表示，比如这里的颜色[10]-[11]，绿灯表示为 0x0001，小端模式。

设备回复：

超始字节	包序列号	访问地址	包类型	数据长度	结果	其它数据	校验和
[0]-[1]	[2]-[3]	[4]-[5]	[6]	[7]-[8]	[9]	[10]-[21]	[22]-[23]
[0]=0x01 [1]=0xEF	[2]=0x00 [3]=0x00	[4]=0xFF [5]=0xFF	[6]=0x07	[7]=0x0F [8]=0x00	[9]=0x00,OK [9]=0x01,失败	全部为0x00	

5.4 获取触摸状态

发送的包：

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x03	\	

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	state	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00（ok）	0：未按 1:按下去		

5.5 建模

发送的包:

sig	Seq	adr	Type	length	cmd	curModel	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x04	当前建模步骤(0~2)		

校验码为从 type 到结束的字节算术和

设备回复:

sig	Seq	adr	Type	length	Result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok		
					0x04:需要调整手指		
					0x05:手指不同		
					0x06:手指已注册		
					0x07:参数不对		
					0x08:超时		

注意：当回复状态码是 0x04(需要调整手指)，这个时候当前指令还在继续

5.6 完成建模

发送的包:

sig	Seq	adr	Type	length	cmd	Id(4)	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-13	14-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x05	0:设备自定 其它：上位机产生		

校验码为从 type 到结束的字节算术和

设备回复:

Sig	Seq	adr	type	length	result	Index	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-13	14-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok 0x07:建模失败 0x0a:指定id 已经存在 0x0D:还没采图	模板 id		

5.7 认证

发送的包：

sig	Seq	adr	Type	length	cmd	IdMin	IdMax	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-13	14-17	18-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x07	可选	可选		

IdMin 和 IdMax 为可选参数，如果有设定，并且 IdMax>IdMin,那么就会从只搜索从 IdMin 到 IdMax 的模板

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Index	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-13	14-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok 0x04:调整手指 0x08:超时 0xff:认证失败	模板 id		

5.8 验证指定 id 是否建模

发送的包：

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x12		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	count	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok 0xff:没有	个数		

5.9 取消等待

正对正在建模采集中，或认证，机器会处于等待状态加了这条取消的指令

发送的包：

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x13		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok		

5.10 删除所有模板

发送的包：

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x0a		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00 （ok）		

5.11 设置 usb 模式

发送的包：

sig	Seq	adr	Type	length	cmd	Mode	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x0a	1:串口 2:adb 的usb 模式		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00（ok）		

5.12 获取可用的模板 ID

发送的包：

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x1A		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Id(4)	Data(8)	Checksum
0-1	2-3	4-5	6	7-8	9	10-13	14-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok 0xff:fa il	用户 id		

5.13 删除指定模板

发送的包：

sig	Seq	adr	Type	length	cmd	Id	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-13	14-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x09	模板 id		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok		

5.14 设置波特率

波特率按以下数组定义

```
const int
arBaud[]={50,75,110,134,150,200,300,600,1200,1800,2400,4800,9600,19200,38400,5
7600,1152 00,230400,460800,500000,
    576000,921600,1000000,1152000,1500000,2000000,2500000,3000000,3500000,4000000};
```

设置和获取时得到的波特率是该数组的索引，比如默认波特率是设置成 16,实际波特率是 115200

发送的包：

sig	Seq	adr	Type	length	cmd	Id(1)	Data(11)	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x11	波特率		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00（ok）		

5.15 设置延时

建模或认证时，如果手指一直没有按下去，超过这个时间也
会退出发送的包：

sig	Seq	adr	Type	length	cmd	Id	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x0e	延时(单位秒)		

校验码为从 type 到结束的字节算术和
设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok		

5.16 获取用户数

发送的包：

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x1e		

校验码为从 type 到结束的字节算术和
设备回复：

Sig	Seq	adr	type	length	result	userCount	modelCount	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-13	14-17	18-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok	用户数	模板数		

5.17 恢复出厂设置

发送的包：

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x1c		

校验码为从 type 到结束的字节算术和
设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok		

5.18 获取系统信息

发送的包：

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x1d		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Content	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-20	21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok	10:main version		
						11:sub version		
						12:device id		
						13:baud rate		
						14-15:level		
						16:time out		
						17:check dup		
						18:check same finger		
						19:usb mode		
						20:read error		

5.19 上传模板

1. 设置将写入的数据信息

发送的包：

sig	Seq	adr	Type	length	cmd	dataType	Size	Index	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-14	15-18	19-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x20	1:模板	数据大小	模板 id		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23

0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok 0x07:参数不对		

2. 写入数据

循环写发的数据包：

发送的包:

sig	Seq	adr	Type	length	Data	Checksum
0-1	2-3	4-5	6	7-8	9-?	
0xef01	0x0000	写的数据位置	0x02	小于 512		

结束时发送的包：

sig	Seq	adr	Type	length	Data	Checksum
0-1	2-3	4-5	6	7-8	9-?	
0xef01	0x0000	写的数据位置	0x08	小于 512		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok		

5.20 下载模板

分为两步：

1. 设置要读的模板信息： 发送的包：

sig	Seq	adr	Type	length	cmd	dataType	Size	Index	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-14	15-18	19-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x1f	1:模板	每次读取的大小	模板id		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok 0x0c:空 id		

2. 读取数据

循环读取发的包：

发送的包：

sig	Seq	adr	Type	length	cmd	dataType	Size	Index	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10	11-14	15-18	19-21	22-23

0xef01	0x0000	读取的数据位置	0x01	0x000f	0x21	1:模板	模板大小	模板id		

校验码为从 type 到结束的字节算术和

出错时设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x07:参数不对		

读取成功:

Sig	Seq	adr	type	length	Data	Checksum
0-1	2-3	4-5	6	7-8		
0xef01	0x0000	位置	0x02:数据包 0x08:数据结束	不定长		

5.21 采集图像

建模或认证，可以先发本指令进行采图，然后再按流程操作，作用是采完图后可以提示用户拿开手指，而不必等待建模或认证的指令完成

1.17 以上版本支持

发送的包:

sig	Seq	adr	Type	length	cmd	current	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-11	12-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x14	采集第几张图		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok 0x04:需要调整 0x07:参数不对 0x08:超时		

5.22 获取设备唯一 ID

模块装进锁之后，首先用该指令设置背景。1.34 以上版本支持

发送的包:

sig	Seq	adr	Type	length	cmd	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-21	22-23
0xef01	0x0000	0xffff	0x01	0x000f	0x39		

校验码为从 type 到结束的字节算术和

设备回复：

Sig	Seq	adr	type	length	result	Data	Checksum
0-1	2-3	4-5	6	7-8	9	10-25	26-27
0xef01	0x0000	0xffff	0x07	0x000f	0x00:ok	设备唯一 ID	