人脸识别模组用户手册

文档版本：V1.0.3

发布日期：2023.3.16

型号：HLK-FRxxxx

前言及声明

感谢您购买深圳市海凌科电子有限公司的人脸识别模组。

本手册针对软、硬件应用开发工程师编写，包含硬件接口、系统资源、指令系统、安装信息等内容。为了确保应用开发顺利进行，在进行模块系统集成前请仔细阅读本用户手册。

请妥善保存手册，以便遇到问题时快速查阅。

本文件包含海凌科电子有限公司的私有信息，在未经本公司书面许可的情况下，第三方不得擅自复制或修改；当然，任何在没有授权、特殊条件、限制或告知的情况下对此信息的复制和擅自修改都是侵权行为。

我们已尽最大努力以保证本手册的准确性。然而，如您有任何疑问或者发现错误，可直接与我公司联系，我们将十分感激。

因我公司奉行不断完善改进产品的宗旨，在任何时间，无需告知任何方的情况下，海凌科电子有限公司有权对本公司产品和服务进行更改、添加、删除、改进以及其他任何变更。如有需要，请您访问我公司的网站或电话联系，以获取最新信息。

未经本公司书面许可，任何单位及个人不得以任何方式或理由对本公司产品、服务、信息、材料的任何部分进行使用、复制、修改、抄录、传播或与其它产品捆绑使用、销售。

海凌科电子有限公司对其发行的或与合作公司共同发行的包括但不限于产品或服务的全部内容及我司网站上的材料拥有版权等知识产权，受法律保护。

在对本公司产品的使用中，海凌科电子有限公司不背负任何责任或者义务；而第三方在使用中则不得侵害任何专利或者其他知识产权。

所有产品的售出都受制于本公司在定购承认书里的销售条款和条件。本公司利用测试、工具、质量控制等技术手段来支持产品的相关性能符合所需规格的一定程度的保证。除了明确的政府书面要求外，没必要执行每款产品的所有参数测试。

凡侵犯本公司版权等知识产权的，本公司必依法追究其法律责任。

本公司法律事务部受本公司指示，特此郑重法律声明！

海凌科电子有限公司

联系我们

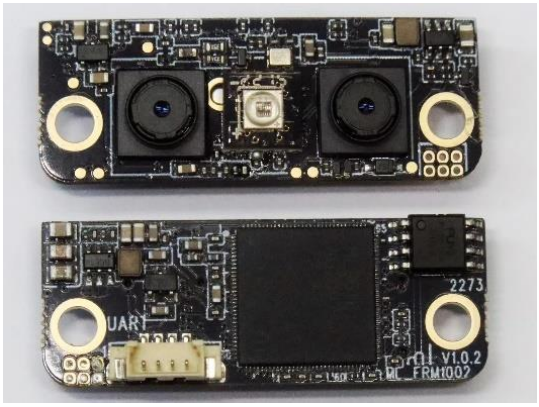
深圳市海凌科电子有限公司
地址：深圳市龙华区民治街道民乐社区星河 WORLD 二期 E 栋 1705、1706、1709A
电话：0755-82557290

目录

1. 概述

HLK_FRMxxxx 人脸识别模组解决方案以高性能应用处理器为硬件平台，配合双目传感器进行活体检测，具有启动速度快、金融级的识别能力、超低使用功耗等特点。凭借超低功耗、强大的运算速度，在多种应用领域中，为各行业赋能。

HLK_FRMxxxx 系列人脸识别模组具备完整的人脸处理能力，可以在无需上位机参与的情况下，完成人脸录入，图像处理，人脸比对，人脸特征储存等功能。和同类人脸产品相比，具有以下特色：



- 高性能算法芯片
- 双目 3D 摄像头模组
- 广角低畸变镜头
- 红外成像
- 金融支付级别识别算法
- 活体检测，抗各种攻击
- UART 通信接口
- 支持 5.0V~14.0V 供电
- 多种通用尺寸，可适配不同结构

安全性

- 内置自研嵌入式系统 3D 深度还原算法、3D 人脸识别算法及多模态活体防范算法，能够有效保障用户信息及解锁安全性，在 98%通过率的前提下，能够做到低于十万分之一的误识率。同时使用多模态活体防伪算法，能有效屏蔽照片、视频及各种头模和假人的攻击。

易用性

- 模组构建的智能门锁能够向用户提供非接触式的解锁开门体验，省去了用户按压指纹或者输入密码的麻烦；
- 从冷启动上电到识别结果输入的单次解锁时间最快可达到 1.0 秒以内，给用户提供了秒开流畅通行开门体验；
- 摄像头模组中使用的广角镜头，使得门锁能支持身高范围广且可调节，小孩和身高特别高的人群都能覆盖。

可靠性

- 成熟稳定的芯片硬件方案；
- 严格的产品质量检验；
- 宽泛的温度适应范围（-20℃~+60℃）。

超低功耗

- 模组工作时的峰值功耗低于 2.0W。

超快速度

- 模组从关机状态下启动，单次解锁最快速度 1.0s。

接口简单易于集成

- 硬件采用行业通用的 PCB 尺寸和固定螺丝孔位设计，提供标准串口供客户能够快速完成整合。主板与摄像头采用一体化设计，满足整机产品的不同结构的安装要求。供电部分支持输入范围为 5.0V~14.0V。

应用范围

HLK-FRXXXX 系列人脸识别模组应用广泛，适合各类人脸识别系统，例如：

- ❖ 智能门锁，智能门禁系统，考勤机等；
- ❖ 刷脸支付系统：3D 人脸支付；
- ❖ 智能设备解锁与人机交互应用；
- ❖ 低功耗电池人脸识别系统。

客户可以根据本手册提供的技术资料，开发出相应的人脸识别应用系统。

技术支持

海凌科拥有完备的技术团队，所有员工均来自物联网行业的专业人才，可以对用户开发提供良好的技术支持和售后服务工作。

2. 主要技术指标

类型	描述	规格
算法核心板	CPU	高性能处理器
	接口	UART，行业标准线序
	主板尺寸	40mm*15mm*4.1mm
摄像头模组	规格	双目摄像头
	视场角	D80.0° H53.3° V67.5°
识别性能	识别时间	1.0s 解锁
	人脸数	100 人
	FAR	<0.001%（十万分之一）
	FRR	<2%
	识别距离	0.4~1.0m
	识别身高	1.20~2.00m(@1.20m 镜头高度，20° 倾角)
工作条件	电源输入	DC5.0~14.0V，典型电压：7.4V（两节锂电池电压）
	功耗	低于 2.0W
	工作温度	-20℃~+60℃
	环境湿度	10%~90%无凝露
	存储条件	温度 23±5℃，相对湿度 RH35~70%
	存储期限	在推荐存储条件下，建议 6 个月内使用

表 2.1 技术指标表

3. 硬件接口

3.1.硬件接口定义

电源和通讯接口采用 1.25-4Pin 插座方式连接,管脚定义如表 3.1 所示。

管脚	信号功能定义			
信号	信号定义	信号类型	LEVEL	备注
1	GND	GND		接地
2	UART_RX	IN	3.3V	通信串口 Rx（人脸识别模组输入）
3	UART_TX	OUT	3.3V	通信串口 Tx（人脸识别模组输出）
4	DC-IN	POWER		电源输入范围 5.0~14.0V（默认 2 节锂电池输入）

表 3.1 接口管脚信号定义

电源和通信需要满足如下上电时序：t>400ms，模块上电后，约需 400ms 时间进行初始化工作。在此期间，模块不能响应上位机命令。

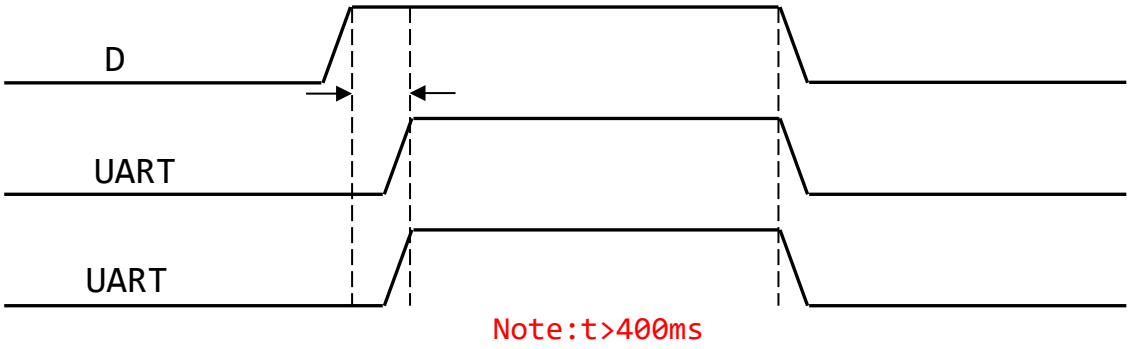


图 3.1 上电时序图

3.2.串行协议

采用半双工异步串口通讯，默认波特率为 115200bps。传送的帧格式为 10 位，一位 0 电平起始位，8 位数据位（低位在前）和一位停止位，无校验位。

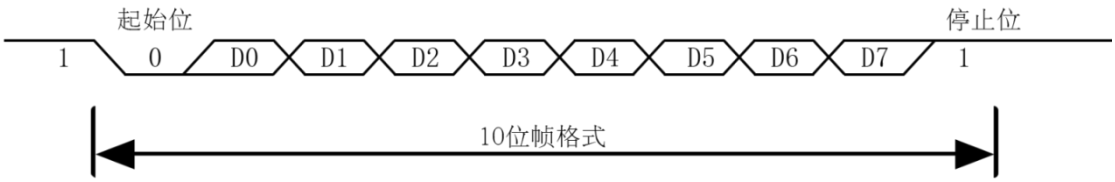


图 3.2 串行协议视图

4. 通讯协议

通讯协议定义了 ML-FRM 系列模板与上位机之间进行数据交换的规则。硬件上采用 UART 接口，软件实现上参考如下通讯协议和指令集。

在传输协议中定义的 2 字节或者 4 字节，采用大字节序（bigendian），高位在前低位在后（例如 0x12345678 实际传输方式为 0x120x340x560x78，不是 0x780x560x340x12）。

链路层描述（UART）

UART 链路层为半双工点对点工作方式，所有命令必须由上位机发起，人脸模组进行响应。

UART 上位机发送到模组的数据包格式：

格式	包头	上位机命令	应用层数据长度	应用层数据内容	除包头外的数据校验位
长度/Bytes	2	1	2	N1	1

表 4.1 上位机数据包格式表

UART 模组响应到上位机的数据包格式：

格式	包头	消息类型	数据长度	响应命令	错误码	数据内容	除包头外的数据校验位
长度/Bytes	2	1	2	1	1	N2	1

表 4.2 模组响应数据包格式表

说明：

包头：UART 起始头界定，无论是上位机发送数据还是人脸模组发送数据，包头数据固定为 0xEF0xAA；

消息类型：分为三种，分别是 REPLY，NOTE，IMAGE，对应数据分别为 0x00，0x01，0x02，具体参考消息类型定义表。

数据长度：描述应用层实际数据的长度，上位机发送的数据包中数据长度不包含包头、上位机命令、应用层数据长度以及最后一位校验位 4 部分；模组响应的数据包中数据长度不包含包头、模组响应位、应用层数据长度以及最后一位校验位。按照不同命令，数据包长度不同，但是每条指令为固定长度。注意上位机发送的数据长度为（N1）Bytes，模组响应的数据长度为（N2+2）Bytes。

响应命令：与上位机发送的命令字段对应，表示收到了响应的命令。

错误码：返回此命令执行的结果，0x00 表示无错误，其他参考 REPLY 错误信息定义表以及 NOTE 错误信息定义表。

除包头外的数据校验位：将除了包头和校验位自身的所有数据进行校验位运算，用来检查数据长度是否有误。

消息类型	代码	描述
REPLY	**0x00**	对上位机发送的命令的响应回复
NOTE	**0x01**	主动向主控返回一些信息，主要在三种情况下发送：a)开机时模组向主控发送 NID_READY；b)录入过程中模组向主控发送人脸信息；c)解锁过程中模组向主控发送人脸信息
IMAGE	**0x02**	模组向主控发送图片，一包数据最大支持 4000 字节数据的传输

表 4.3 消息类型定义表

代码	说明
0x00	成功
0x01	模组拒绝该命令
0x02	录入/匹配算法已终止
0x03	发送消息错误
0x04	相机打开失败
0x05	未知错误
0x06	无效的参数
0x07	内存不足
0x08	没有已录入的用户
0x09	录入超过最大用户数量
0x0A	人脸已录入
0x0C	活体检测失败
0x0D	录入或解锁超时
0x0E	加密芯片授权失败
0x13	读文件失败
0x14	写文件失败
0x15	通信协议未加密
0x17	RGB 图像没有 ready
0xFD	无效信息 ID，key 未写入
0xFE	检测错误
0xFF	编码错误

表 4.4 REPLY 错误信息定义表

代码	说明
0x00	人脸正常
0x01	未检测到人脸
0x02	人脸太靠近图片上边沿，未能录入
0x03	人脸太靠近图片下边沿，未能录入
0x04	人脸太靠近图片左边沿，未能录入
0x05	人脸太靠近图片右边沿，未能录入
0x06	人脸距离太远，未能录入
0x07	人脸距离太近，未能录入
0x08	眉毛遮挡
0x09	眼睛遮挡
0x0A	脸部遮挡
0x0B	录入人脸方向错误
0x0C	在闭眼模式检测到睁眼状态
0x0D	闭眼状态
0x0E	在闭眼模式检测中无法判定睁闭眼状态

表 4.5　NOTE 错误信息定义表

5. 命令集汇总

上位机					人脸模组	
命令	代码	命令数据结构	说明	数据长度	响应数据结构	说明
停止所有当前在处理的消息,模组进入standby 状态	0x10	无	无	0	无	无
获取模组当前状态	0x11	无	无	0	is_working（1byte）	模组是否在工作状态
人脸匹配	0x12	timeout_heb(1byte)	匹配超时时间(单位 s)	2	user_id_heb(1byte)	匹配到的用户 ID
					user_id_leb(1byte)	
					user_name(32bytes)	匹配到的用户名字
		timeout_leb(1byte)			is_admin(1byte)	是否为管理员
					unlockStatus(1bytes)	匹配过程中睁闭眼状态
单方向人脸注册	0x13	is_admin(1byte)	是否设置为管理员	35	user_id(2byte)(只有当录入完成且错误码为 0x00 时才有 user_id)	成功注册的用户 ID
		user_name(32bytes)	用户名			
		face_dir(1byte)	录入方向		face_dir(1bytes)	注册方向
		timeout(1byte)	录入超时时间(单位 s)			
删除指定 ID 号的用户	0x20	user_id_heb(1byte)	已录入的用户 ID	2	无	无
		user_id_leb(1byte)				
删除所有已录入的用户	0x21	无	无	0	无	无
停止所有	0x23	无	无	0	无	无

当前在处理的消息,模组进入 standby 状态						
获取所有已注册用户的数量及 id	0x24	无	无	0	id_sum(1byte)	用户总数
					id_list(200bytes) (每个 id 号为双字节)	Id 列表
人脸注册	0x26	is_admin(1byte)	是否设置为管理员	40	user_id(2byte) (只有当录入完成且错误码为 0x00 时才有 user_id)	成功注册的用户 ID
		user_name(32bytes)	用户名			
		face_dir(1byte)	录入方向			
		enroll_type(1byte)	录入类型			
		enable_duplicate (1byte)	是否重复录入		face_dir(1bytes)	注册方向
		timeout (1byte)	录入超时时间(单位 s)			
		reserved (3byte)	预留位			
获取当前人脸模组版本号	0x30	无	无	0	version(32bytes)	芯片固件版本信息
获取当前人脸模组 UUID	0x93	无	无	0	uuid(32bytes)	芯片 UUID
模组下电命令	0xED	无	无	0	无	无
演示模式	0xFE	Enable（1byte）	开关演示模式	1	无	无

表 5.1 命令集汇总表

5.1.模组回归 standby 状态

(1) 发送包格式示例（Byte）

包头		命令	数据长度		校验位
0xEF	0xAA	0x23	0x00	0x00	0x23

表 5.2 模组回归 standby 状态发送包示例表

(2) 响应包格式示例（Byte）

包头		消息类型	数据长度		响应命令	错误码	校验位
0xEF	0xAA	0x00	0x00	0x02	0x23	0x00	021

表 5.3 模组响应包示例表

5.2.获取模组当前状态

(1) 发送包格式示例（Byte）

包头		命令	数据长度		校验位
0xEF	0xAA	0x11	0x00	0x00	0x11

表 5.4 获取模组当前状态发送包示例表

(2) 响应包格式示例（Byte）

在是否工作的数据位上，0x00 表示模组正在等待上位机命令发送，0x01 表示模组正在进行录入或者匹配等工作。

包头		消息类型	数据长度		响应命令	错误码	是否在工作	校验位
0xEF	0xAA	0x00	0x00	0x03	0x11	0x00	0x00	0x12

表 5.5 获取模组当前状态响应包示例表

5.3.人脸匹配

(1) 发送包格式（Byte）

包头		命令	数据长度		匹配超时时间		校验位
0xEF	0xAA	0x12	0x00	0x02	(1byte)	(1byte)	(1byte)

表 5.6 人脸匹配发送包格式表

(2) 响应包格式（Byte）

匹配成功示例：

包头		消息类型	数据长度		响应命令	错误码	数据内容	校验位
0xEF	0xAA	0x00	0x00	0x26	0x12	0x00	(36bytes)	0x23

表 5.7 人脸匹配成功响应示例

匹配失败示例匹配（匹配超时等错误码）：

包头		消息类型	数据长度		响应命令	错误码	数据内容	校验位
0xEF	0xAA	0x00	0x00	0x26	0x12	0x0D	(36bytes)	0x23

表 5.8 人脸匹配失败响应示例

正在匹配(会在未超时时间内多次发送该数据包)：

包头		消息类型	数据长度		响应命令	错误码	数据内容	校验位
0xEF	0xAA	0x01	0x00	0x11	0x01	0x01	(17bytes)	0x10

表 5.9 人脸正在匹配响应示例表

5.4.删除指定 ID 号的用户

用户 ID 是从 01 开始进行递增计数，删除某个 ID 的用户后其他剩余用户的 ID 号不会受到影响。

(1) 发送包格式（Byte）

包头		命令	数据长度		删除 ID		校验位
0xEF	0xAA	0x20	0x00	0x02	(1byte)	(1byte)	(1byte)

表 5.10 删除指定 ID 号的用户发送格式表

(2) 响应包格式（Byte）

删除成功示例：

包头		消息类型	数据长度		响应命令	错误码	校验位
0xEF	0xAA	0x00	0x00	0x02	0x20	0x00	0x22

表 5.11 删除 ID 成功响应示例表

删除失败示例（不存在该用户等错误码）：

包头		消息类型	数据长度		响应命令	错误码	校验位
0xEF	0xAA	0x00	0x00	0x02	0x20	0x01	0x23

表 5.12 删除 ID 失败响应示例表

5.5.删除所有已录入的用户

(1) 发送包格式示例（Byte）

包头		命令	数据长度		校验位
0xEF	0xAA	0x21	0x00	0x00	0x21

表 5.13 删除所有用户命令格式表

(2) 响应包格式示例（Byte）

包头		消息类型	数据长度		响应命令	错误码	校验位
0xEF	0xAA	0x00	0x00	0x02	0x21	0x00	0x23

表 5.14 删除所有用户成功响应示例

5.6.人脸注册（包括 0x13 和 0x26）

命令 0x26 人脸注册有两种模式，分别为单向注册和五向注册，两者对应的代码分别为 0x01 和 0x00。而命令 0x13 人脸注册只有五向注册。注册时的用户视角朝向对应的代码以及模组响应的方向代码如表 5.11 所示，其中无论单向还是五向注册的直视摄像头的发送和响应代码均相同。示例表中均用命令 0x26 作为例子。

是否录入重复位中，0x00 为禁止录入相同人脸，0x01 为允许录入相同人脸；

是否为管理员位中，0x00 为非管理员，0x01 为管理员。

0x26 中的保留位不做处理，其三位均是 0x00

	直视	抬头看	低头看	朝左看	朝右看
发送	0x01	0x10	0x08	0x04	0x02
响应	0x01	0x11	0x19	0x1D	0x1F

表 5.15 发送与响应的视角代码

(1) 发送包格式（Byte）：

当命令为 0x13 时

包头		命令	数据长度		是否为管理员	用户名	注册方向	录入超时时间	校验位
0xEF	0xAA	0x13	0x00	0x23	(1byte)	(32bytes)	(1byte)	(1byte)	(1bytes)

当命令为 0x26 时

包头		命令	数据长度		是否为管理员	用户名	注册方向	注册类型	是否重复录入	录入超时时间	保留位	校验位
0xEF	0xAA	0x26	0x00	0x28	(1byte)	(32bytes)	(1byte)	(1byte)	(1byte)	(1byte)	(3bytes)	(1bytes)

表 5.16 人脸注册发送包格式表

(2) 响应包格式示例(0x26 为例)

单向注册成功响应示例

包头		消息类型	数据长度		响应命令	错误码	用户 id		录入方向	校验码
0xEF	0xAA	0x00	0x00	0x05	0x26	0x00	0x00	0x01	0x01	0x23

表 5.17 单向注册成功响应示例表

五向注册低头视角成功响应示例

包头		消息类型	数据长度		响应命令	错误码	用户 id		录入方向	校验码
0xEF	0xAA	0x00	0x00	0x05	0x26	0x00	0xFF	0xFF	0x19	0x3A

表 5.18 五向注册低头视角成功响应示例表

在返回的错误码为 0x00 时的五向注册中，只有当最后一个方向的注册成功时，用户 id 才会出现在响应包中，其余方向注册成功后均返回 0xFF,0xFF。

注册等待时的 NOTE 与匹配等待时的 NOTE 一样。

注册失败时单向和五向返回的用户 id 均为 0x00,0x00；录入方向位的代码也为 0x00。

除了响应命令外，两个注册命令的响应包格式相同

5.7.获取当前人脸模组版本号

(1) 发送包格式示例（Byte）

包头		命令	数据长度		校验位
0xEF	0xAA	0x30	0x00	0x00	0x30

表 5.19 获取当前模组版本号发送包示例表

(2) 响应包格式示例（Byte）

包头		消息类型	数据长度		响应命令	错误码	版本数据	校验位
0xEF	0xAA	0x00	0x00	0x22	0x30	0x00	(32bytes)	(1byte)

5.8.获取当前人脸模组 UUID

(1) 发送包格式示例（Byte）

包头		命令	数据长度		校验位
0xEF	0xAA	0x93	0x00	0x00	0x93

表 5.21 获取当前模组 UUID 发送包

(2) 响应包格式示例（Byte）

包头		消息类型	数据长度		响应命令	错误码	版本数据	校验位
0xEF	0xAA	0x00	0x00	0x22	0x93	0x00	(32bytes)	(1byte)

表 5.22 模组响应包示例表

5.9.获取所有已注册的用户数量及 id

(1) 发送包格式示例（Byte）

包头		命令	数据长度		校验位
0xEF	0xAA	0x24	0x00	0x00	0x24

表 5.23 获取所有已注册的用户数量及 id 发

(2) 响应包格式示例（Byte）

包头		消息类型	数据长度		响 应命令	错误码	用户数量	大字节序的所有用户 id（2bytes/user）	校验位
0xEF	0xAA	0x00	0x00	0xCB	0x24	0x00	(1byte)	(200bytes)	(1byte)

表 5.24 获取所有已注册的用户数量及 id 响应包示

5.10. 演示模式

(1) 发送包格式示例（Byte）

包头		命令	数据长度		演 示 模 式开关位	校验位
0xEF	0xAA	0xFE	0x00	0x01	0x01	0xFE

(2) 响应包格式示例（Byte）表 5.25 演示模式发送包示例表

包头		消息类型	数据长度		响应命令	错误码	校验位
0xEF	0xAA	0x00	0x00	0x02	0xFE	0x00	0xFC

表 5.26 演示模式响应包示例表

5.11. 模组下电命令

(1) 发送包格式示例（Byte）

包头		命令	数据长度		校验位
0xEF	0xAA	0xED	0x00	0x00	0xED

表 5.25 下电命令发送包示例表

(2) 响应包格式示例（Byte）

包头		消息类型	数据长度		响应命令	错误码	校验位
0xEF	0xAA	0x00	0x00	0x02	0xED	0x00	0xEF

表 5.26 下电命令响应包示例表

5.12. 上传下载文件功能

模组提供人脸特征上传（模组—>上位机）以及下载（上位机—>模组）功能。该功能总命令为 0xF8，采用不同子命令实现不同功能，统一的命令格式如下：

发送包格式示例（Byte）

数据	0xEF	0xAA	0xF8	size	sub_cmd	data	check_num
长度	1	1	1	2	1	N	1

表 5.27 上传下载发送包示例表

size：数据长度，包括 sub_cmd(1 字节)+data(N 字节)；
sub_cmd：上传下载流程的子命令，不同子命令代表不同功能；
data：数据域，长度可变；
check_num：校验值；

响应包格式示例（Byte）

数据	0xEF	0xAA	0x00	size	0xF8	result	sub_cmd	data	check_num
长度	1	1	1	2	1	1	1	N	1

表 5.28 上传下载响应包示例表

result:错误码；
sub_cmd：子命令；
data：数据；
check_num：校验值；

(1) 特征文件上传流程

上位机和模组进行上传功能的流程图如图 5.1 所示。

在执行上传功能之前，需要向模组发送准备上传命令，用于配置分块大小、获取文件总大小、块数等信息。

然后上位机可以分包获取文件数据，获取完成后进行文件的完整性校验。

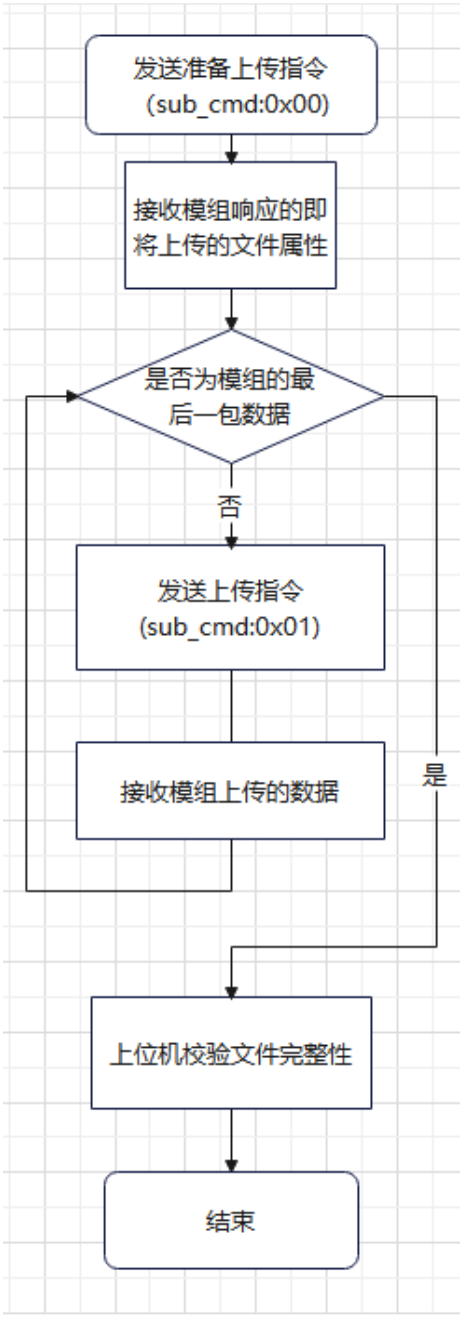


图 5.1 上传功能的流程图

1．准备上传数据命令

上层发送命令格式示例（Byte）：

数据	0xEF	0xAA	0xF8	size	0x00	id	per_block_size	check_num
长度	1	1	1	2	1	2	2	1

表 5.29 上层准备接受上传命令示例表

id：需要人脸模组上传的特征数据的目标 ID，ID 取值范围为 1 至最大的人脸存储数量（默认 100）；

per_block_size：模组每个发送帧的特征数据块长度；

check_num：校验值；

模组响应数据示例（Byte）：

数据	0xEF	0xAA	0x00	size	0xF8	result	0x00	total_length	block_num	last_block_szie	check_num
长度	1	1	1	2	1	1	1	4	4	2	1

表 5.30 模组准备上传响应示例表

total_length：人脸特征数据总长度；

block_num：需要接收的块数量；

last_block_size：最后一帧的人脸特征数据块长度；

result：

0x00：已准备好；

0x01：ID 不存在；

0x02：发送长度不合理；

0x10：忙；

2．上传数据命令

上层发送命令格式示例（Byte）：

数据	0xEF	0xAA	0xF8	size	0x01	block_index	check_num
长度	1	1	1	2	1	4	1

表 5.31 上层接受上传命令示例表

block_index：当前需要发送块区的索引号。

模组响应数据示例（Byte）：

数据	0xEF	0xAA	0x00	size	0xF8	result	0x01	id	block index	face data	check num
长度	1	1	1	2	1	1	1	2	4	N	1

表 5.32 模组上传数据响应示例表

face_id：当前发送的人脸数据的 ID。

block_index：当前发送的块区索引号。

face_data：人脸数据。

result：

0x00：完成。

0x01：未发送准备命令（sub_cmd:0x00）。

0x02：不存在目标索引号。

0x10：忙；

(2) 特征文件下载流程

上位机和模组进行下载功能的流程图如图 5.2 所示。

在执行下载功能之前，需要向模组发送下载准备命令，用于配置文件大小、分块大小、块数等信息。

模组响应正常后，进行分包下载。

下载完成，上位机发送下载完成命令通知模组将下载的文件进行校验和存储。

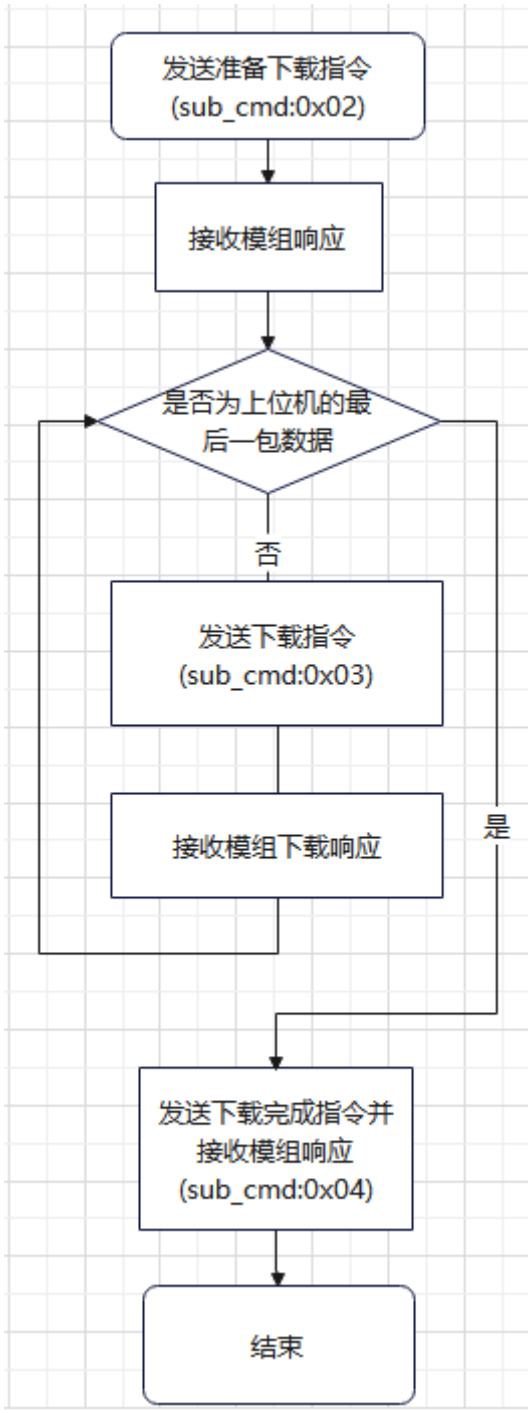


图 5.2 下载功能的流程图

1．准备下载数据命令

上层发送命令格式示例（Byte）：

数据	0xEF	0xAA	0xF8	size	0x02	id	total_length	block_num	per_block_size	last_block_size	check_num
长度	1	1	1	2	1	2	4	4	2	2	1

表 5.33 上层准备下载命令示例表

id：插入的特征 ID，ID 取值范围为 1 至最大的人脸存储数量（默认 100）；

total_length：特征总长度。

block_num：块数。

per_block_size：发送数据帧的特征数据块长度。

last_block_size：最后一块的数据长度。

模组响应数据示例（Byte）：

数据	0xEF	0xAA	0x00	size	0xF8	result	0x02	check_sum
长度	1	1	1	2	1	1	1	1

表 5.34 模组准备接受下载响应示例表

result：

0x00：OK。

0x01：ID 已被使用。

0x02：ID 超出范围。

0x03：接收的数据长度不合理。

0x04：block_num 数据不合法。

0x10：当前忙。

2．下载数据命令

上层发送命令格式示例（Byte）：

数据	0xEF	0xAA	0xF8	size	0x03	Block_index	data	Check_num
长度	1	1	1	2	1	4	N	1

表 5.35 上层下载数据命令示例表

block_index：发送的块区索引号。

data：发送的数据

模组响应数据示例（Byte）：

数据	0xEF	0xAA	0x00	size	0xF8	result	0x03	Block_index	Check_num
长度	1	1	1	2	1	1	1	4	1

表 5.36 模组接收下载数据响应示例表

block_index：当前接收到的块索引。

result：

0x00：OK。

0x01：未发送 ready(sub_cm:0x02)

0x01：数据长度有误。

0x02：index 有误。

0x10：当前忙。

3．下载完成命令

上层发送命令格式示例（Byte）：

数据	0xEF	0xAA	0xF8	size	0x04	Check_sum
长度	1	1	1	2	1	1

表 5.37 上层下载完成命令示例表

模组响应数据示例（Byte）：

数据	0xEF	0xAA	0x00	size	0xF8	result	0x04	id	Check_sum
长度	1	1	1	2	1	1	1	2	1

表 5.38 模组接收下载数据完成响应示例表

face_id：操作的 ID

result：

0x00：OK。

0x01：发送数据错误。

0x02：数据存储发生错误。

0x10：当前忙。

6. 接口函数说明

6.1. RECV_MSG，*P_RECV_MSG 结构

模组发给上位机的每条响应，都会经过协议格式的对比收录于 RECV_MSG 结构体中，该结构体需要初始化后提前对里面的 uint8_tcmd 项进行设置，以便将该结构体的指针传入功能函数后函数能让接收到的命令位与 cmd 进行对比，判断模组是否响应出错。

```
typedefstruct
{
    uint8_tdata_length[2];
    uint8_tcmd;
    uint8_tstate;
    uint8_tdata[MAX_SIZE];
    uint8_tparity;
}RECV_MSG,*P_RECV_MSG;
```

模组返回的数据存放于 data 项中，其中不包括包头、消息类型、数据长度、命令位、错误码以及校验位，可供用户以及功能函数进行直接的数据分析。

6.2. USER，*P_USER 结构

USER 结构体用于存储使用结构函数的用户的名字、id 号以及是否为管理员身份，例如在模组人脸匹配功能中可以将响应数据通过对比 id 项判断是否匹配出错。

```
typedefstruct
{
    uint32_tisAdm;
    uint8_tname[32];
    uint8_tid[2];
}USER,*P_USER;
```

6.3. Receive_Interruption

Receive_Interruption 函数用于 UART 接收中断期间对接收的数据进行处理的函数，该函数参数为每次中断接收到的一位字节数据以及存储数据用的 RECV_MSG 结构体，函数内部会对接收的数据分析其在协议中的位置。

```
voidReceive_Interruption(uint8_tg_fUART1_Byte,P_RECV_MSGp_function);
```

6.4. GetParityCheck

校验函数 GetParityCheck 需要传入即将发送到模组的指令指针，初始设置校验码为 0，除了包头的 EF、AA，校验码与剩余的指令按位异或运算，每一位异或运算所得结果与下一位继续按位异或运算，最后得到的则为该指令的校验码。

```
unsignedcharGetParityCheck(uint8_t*p,intlen)
{
    uint8_tnParityCheck=0;
    for(inti=2;i<len-1;++i)
    {
        nParityCheck^=p[i];
    }

    returnnParityCheck;
}
```

6.5. ML_API_DeleteID

ML_API_DeleteID 函数用于删除指定 id 的用户，需要传入用户的 id 号以及 P_RECV_MSG 结构体指针，后者用于获取命令发送后存储模组的响应，判断是否删除成功。

```
intML_API_DeleteID(uint8_tid,P_RECV_MSGp_function);
```

6.6. ML_API_DeleteAll

ML_API_DeleteAll 函数用于删除所有已注册的用户，需要传入 P_RECV_MSG 结构体指针，用于获取命令发送后存储模组的响应，判断是否删除成功。

```
intML_API_DeleteAll(P_RECV_MSGp_function);
```

6.7. ML_API_GetFirmwareVersion

ML_API_GetFirmwareVersion 函数用于获取当前模组的固件版本号，版本号数据总长 32 字节，需要传入 P_RECV_MSG 结构体指针，用于获取命令发送后存储模组的响应，并判断该命令发送后模组响应状态以及版本数据

```
intML_API_GetFirmwareVersion(P_RECV_MSGp_function);
```

6.8. ML_API_Match

ML_API_Match 函数用于进行人脸匹配功能，需要传入 P_USER 结构体指针以及 P_RECV_MSG 结构体指针，前者是存取匹配成功后返回来的 ID 以及其用户名，后者是对模组响应的数据进行整理。

```
intML_API_Match(P_USERp_user,P_RECV_MSGp_function);
```

6.9. ML_API_Enrollment_0x13

ML_API_Enrollment_0x13 函数用于实现人脸的注册录入功能，需要传入 P_RECV_MSG 结构体指针、P_USER 结构体指针、录入方向 dir（0<=dir<=4）。函数返回的整数值代表录入结果，0 代表录入超时，1 代表该方向录入成功，2 代表用户录入成功。建议录入方向从 0 开始依次递增，若返回 1 再进行下一个方向的录入，且当 dir=4 时函数仅返回 0 或是 2。

intML_API_Enrollment_0x13(P_RECV_MSGp_function,P_USERp_user,uint8_tdir);

6.10. ML_API_GetUUID

ML_API_GetUUID 函数用于获取当前模组的 UUID，数据总长 32 字节，需要传入 P_RECV_MSG 结构体指针，用于获取命令发送后存储模组的响应，并判断该命令发送后模组响应状态以及 UUID 数据.

intML_API_GetUUID(P_RECV_MSGp_function);

6.11. ML_API_Enrollment_0x26

ML_API_Enrollment_0x26 函数用于实现人脸的注册录入功能，需要传入 P_RECV_MSG 结构体指针、P_USER 结构体指针、录入方向 dir（0<=dir<=4）以及录入模式参数 enrollment_parttern（enrollment_pattern=1 或 5，分别代表单向录入和五向录入）。函数返回的整数值代表录入结果，0 代表录入超时，1 代表该方向录入成功，2 代表用户录入成功。当录入模式为单向时该函数不会返回 1 的结果，当录入模式为五向时建议录入方向从 0 开始依次递增，若返回 1 再进行下一个方向的录入，且当 dir=4 时函数仅返回 0 或是 2。

intML_API_Enrollment_0x26(P_RECV_MSGp_function,P_USERenrollmentpattern,uint8_tdir,uint8_tenrollment_pattern);

6.12. ML_API_GetUUID

ML_API_GetUUID 函数用于获取当前模组的 UUID，数据总长 32 字节，需要传入 P_RECV_MSG 结构体指针，用于获取命令发送后存储模组的响应，并判断该命令发送后模组响应状态以及 UUID 数据.

intML_API_GetUUID(P_RECV_MSGp_function);

6.13. ML_API_GetAllUserID

ML_API_GetAllUserID 函数用于获取模组内所有已注册的用户的数量以及他们的 id，在响应的数据中每个用户的 id 均为双字节表示，该函数需要传入 P_RECV_MSG 结构体指针用于存储模组响应的数据，并且会返回一个 uint8_t 类型的数作为提取到的所有已注册用户的数量。所有用户 id 号可使用 P_RECV_MSG 结构体指针查询。

intML_API_GetAllUserID(P_RECV_MSGp_function);

6.14. ML_API_ResetAndStandby

ML_API_ResetAndStandby 函数用于将正在进行录入或者匹配的状态的模组复位回到未进行匹配或者录入的状态，该命令针对于在五向注册过程中某一个方向的注册成功但却收不到下一个方向的命令，这时如果直接重新从直视方向进行录入则会反馈方向错误导致注册失败的情况，该函数的命令发送后会将模组重新归位于直视方向。此函数需要传入 P_RECV_MSG 结构体指针用于存储模组响应的数据。

intML_API_ResetAndStandby(P_RECV_MSGp_function);

6.15. ML_API_GetModuleState

ML_API_GetModuleState 函数用于获取当前模组是否处于匹配或者录入等工作状态，需要传入 P_RECV_MSG 结构体指针，用于获取命令发送后存储模组的响应，在响应包数据位中 0x01 表示模组正在工作，0x00 则表示模组处于等待命令的 standby 状态。该函数可在模组向上位机发送 NOTE 消息时使用。

intML_API_GetModuleState(P_RECV_MSGp_function);

6.16. ML_API_PowerDown

ML_API_PowerDown 函数用于通知模组即将下电，下电动作由主板进行，目的是为了通知模组下电前进行数据保存等行为，具体行为设置由开发者决定。

intML_API_PowerDown(P_RECV_MSGp_function);

6.17. ML_API_PresentationMode

ML_API_GetModuleState 函数用于设置模组是否进入演示模式的工作状态，需要传入 P_RECV_MSG 结构体指针，用于获取命令发送后存储模组的响应，在发送包数据位中 0x01 表示命令模组进入演示模式，0x00 则表示命令模组退出演示模式。演示模式与匹配模式不同在于，前者进入后一旦捕捉到人脸，无论匹配是否正确，模组均会返回成功的消息，以便对客户演示人脸捕捉的功能。

intML_API_PresentationMode(P_RECV_MSGp_function);

7. 修订记录

Version	Revisions	Date	Modifiedby
V1.0.0	Firstdraft.	2022/10/18	admin
V1.0.1	Modifyenrollmentcmd	2022/11/4	admin
V1.0.2	Upload&DownloadcmdAppended	2023/1/5	admin
V1.0.3	AddPowerDownCMD ModifyFunctionType Modify0x13enrollmentcmd Add0x23CMDasStandby	2023/3/16	admin