C300 Series Intelligent Face Recognition Module

Communication Protocol B Description

Version: V2.2-20231218

Product Documentation Revision History

Version	illustrate	Date Revision	Author
V0.1	Create document, add basic content	20230610	ssliu
V1.0	Improved communication protocol description	20230630	ssliu
V2.0	Added palm vein protocol content	20231115	watery
V2.1	Face registration command is compatible with palm vein registration	20231117	watery
V2.2	Module name changed to C300	20231218	watery

Table of contents

1 Introduction

With the rapid development of artificial intelligence technology, and based on the uniqueness and convenience of face and palm ID, Sunny Wenqiao's proprietary facial recognition and palm vein recognition technologies have been deployed in various scenarios since 2018.

Applied in various identity recognition scenarios such as door locks, finance, and access control.

Full protection has become the development trend of smart products.

The smart lock industry has developed rapidly in the past two years, and currently fingerprint unlocking is the main method.

Accurate facial recognition technology and palm vein recognition technology provide door lock products with facial authentication that is both secure and accurate

The palm vein authentication function has also been added. By entering the face or palm feature information in advance, the module can

Quickly start and quickly compare face or palm information to unlock. Compared with other unlocking methods, face and palm

Authentication can bring users a more convenient, fast, accurate and secure biometric experience.

2 Basic product introduction

2.1 Product form

Intelligent Optics C300 series module products consist of 3D TOF, structured light, IR binocular module + Intelligent Optics 3D

It consists of face recognition algorithm + palm vein recognition algorithm.

2.2 Product Advantages

- Security

  Intelligent optics widely implements payment-level 3D face recognition algorithm and palm vein recognition algorithm, which can

  It can effectively protect the security of user information and unlocking, and can achieve one in a million unlocking rates under the premise of 99% unlocking rate.

  3D liveness detection can effectively block various attacks such as photos, 3D head models, masks, etc.

  The improvement in security level is unattainable by traditional fingerprint recognition solutions.

- Ease of use

  ÿ C300 series smart door locks can provide users with a contactless, seamless door unlocking experience, saving

  It eliminates the tedious process of users pressing fingerprints or entering passwords, and also solves the problem for some people.

  For example, the problem of difficulty in unlocking caused by the unclear fingerprint features of the elderly or children.

  ÿ Intelligent optical computer vision algorithm is deeply optimized based on embedded platform, single unlocking time

  Less than 1 second (calculated from power-on), providing users with an extremely fast door opening experience.

  ÿ In face mode, the large vertical field of view allows the door lock to support a height range from 130cm to

  200cm, can cover both children and people of particularly tall stature.

- reliability

  ÿ Mature and stable 3D visual hardware solution.

  ÿ Strict product quality inspection.

  ÿ Wide temperature range.

- Ultra-low power consumption

  The average power consumption of the module when working is ~0.7W.

- Simple interface for easy integration

From a hardware perspective, the existing door lock solution only needs to power the C300 series module and connect to the

For details, please refer to the detailed specifications of each project.

The connection and the easy-to-use information transmission protocol on the software make it easy for lock companies to integrate.

With less workload, you can get the fastest door lock upgrade and the safest unlocking function.

3 Communication Protocol Description

The C300 series modules and the main control use serial port communication. This chapter will explain the serial port configuration and communication protocol.

3.1 Serial port configuration

Table 3-1 Serial port configuration

Configuration items	illustrate
Baud rate	Supports most baud rates, default is 115200
Hardware/software flow control	Not used
Data bits	8
Stop bits	1
Parity bit	Not used

3.2 Communication message format

The basic message format for communication between the master control and C300 series modules is shown in Table 3-2.

Table 3-2 Message format

SyncWord	MsgID	Size	Data	ParityCheck
2 bytes	1 byte	2 bytes	N bytes	1 byte

Table 3-3 is a detailed description of each field.

Table 3-3 Message field description

Field length		illustrate
SyncWord 2 bytes Fixed message start synchronization word 0xEF 0xAA		
MsgID	1 byte message ID (e.g. RESET)	
Size	2bytes	Data size, in bytes
Data	N bytes of data corresponding to the message, such as the parameters corresponding to the command message.	65535>=N>=0, N=0 means this message has no parameters.
Parity Check	1 byte The parity check code of the protocol is calculated by excluding the Sync Word part of the entire protocol.	After the division, the remaining bytes are bitwise XORed.

3.3 Communication message details

3.3.1 Message header file definition

The message header file definition is as shown in Appendix 1 (example only, the specific details are subject to the actual agreement of the current protocol).

3.3.2 Message List

Table 3-4 lists all the communication messages between the module (M) and the master (H). Each message contains an ID and message data. Each message processing has a corresponding timeout definition. Timeout refers to the time when the master waits for the module to process. The longest processing time, that is, the time it takes for the module feedback processing to be completed.

When the instruction processing times out, the master can use the following processing methods:

1. Send MID_GETSTATUS command

This message can quickly obtain the working status of the module. If it crashes, there is no return and the main control can be powered off directly.

2. Send the RESET command

Stop all current processing, the module enters the standby state, and waits for new instructions from the master.

3. Think the module is dead

The main control can adopt the power-off restart processing method.

Table 3-4 Communication message summary

MsgID	**Code** direction **Data** structure **Timeout** (ms)				illustrate
MID_REPLY	0x00	M»H	s_msg_reply_data	N/A	Reply is the module's reply to the master. The response to the command of the master For each command, the module will eventually enter Reply
MID_NOTE	0x01	M»H	s_msg_note_data	N/A	Note is sent by the module to the main control Information, according to note id to determine the deletion Information type and adapted data structure
MID_IMAGE	0x02	M»H	image(raw)	N/A	The module sends pictures to the main control
MID_RESET	0x10	H»M	N/A	1000	Stop all currently processing messages, The module enters the standby state
MID_GETSTATUS	0x11	H»M	N/A	200	Return the current status of the module immediately
MID_VERIFY	0x12	H»M	s_msg_verify_data	10000	Authentication unlock (face or palm)

MID_ENROLL	0x13	H»M	s_msg_enroll_data	10000	Multi-directional face or palm recording for new users enter
MID_SNAPIMAGE	0x16	H»M	s_msg_snap_image_data	N/A	Capture pictures and store them locally
MID_GETSAVEDIMAGE	0x17	H»M	s_msg_get_saved_image_dat a	500	Get the size of the image to be uploaded
MID_UPLOADIMAGE	0x18	H»M	s_msg_upload_image_data	10000	Upload locally stored images to the main control
MID_ENROLL_SINGLE	0x1D	H»M	s_msg_enroll_data	10000	New user single face or palm registration
MID_DELUSER	0x20	H»M	s_msg_deluser_data	200	Deleting a registered user
MID_DELALL	0x21	H»M	N/A	200	Delete all registered users
MID_GETUSERINFO	0x22	H»M	s_msg_getuserinfo_data	200	Get information about a registered user
MID_FACERESET	0x23	H»M	N/A	200	Reset algorithm status if in progress Interactive input will clear the recorded Direction of entry
MID_GET_ALL_USERID	0x24	H»M	N/A	500	Get the number of all registered users and ID
MID_ENROLL_ITG	0x26	H»M	s_msg_enroll_itg	10000	Comprehensive entry
MID_GET_VERSION	0x30	H»M	N/A	1000	Get software version information
MID_START_OTA	0x40	H»M	s_msg_startota_data	N/A	Enter OTA upgrade mode
MID_STOP_OTA	0x41	H»M	N/A	N/A	Exit OTA mode and restart the module.
MID_GET_OTA_STATUS	0x42	H»M	N/A	N/A	Get OTA status and transfer upgrades The starting packet number of the packet
MID_OTA_HEADER	0x43	H»M	s_msg_otaheader_data	N/A	The size of the upgrade package sent, the total number of packages, The size of the sub-package, the md5 of the upgrade package value
MID_OTA_PACKET	0x44	H»M	s_msg_otapacket_data	N/A	Send upgrade package: package number, package size Small, packed data.
MID_INIT_ENCRYPTION	0x50	H»M	s_msg_init_encryption_data	1000	Set the encryption random number
MID_CONFIG_BAUDRATE	0x51	H»M	s_note_config_baudrate	100	Set the communication port baud in OTA mode Rate
MID_SET_RELEASE_ENC_ KEY	0x52	H»M	s_msg_enc_key_number_dat a	N/A	Set the production encryption key sequence, Electricity will be saved
MID_SET_DEBUG_ENC_K EY	0x53	H»M	s_msg_enc_key_number_dat a	N/A	Set the debug encryption key sequence, Power loss
MID_GET_LOGFILE	0x60	H»M	s_msg_reply_get_log_data	1000	Save log to memory

MID_UPLOAD_LOGFILE	0x61	H»M	s_msg_upload_logfile_data	N/A	Upload the saved log to the master control
MID_ENROLL_BY_PIC	0x71	H»M	s_msg_enroll_data	10000	Still image capture
MID_SET_ENRROL_FOR ME	0x72	H»M	s_msg_param_data	200	Set the input parameters
MID_GET_ENROL_FOR ME	0x7A	H»M	s_msg_param_data	500	Get input parameters
MID_VERIFY_BY_PIC	0x82	H»M	N/A	10000	Static image recognition
MID_SET_THRESHOLD_L EVEL	0xD4	H»M	s_msg_algo_threshold_level	200	Setting the Threshold Level
MID_GET_THRESHOLD_L EVEL	0xD5	H»M	s_msg_algo_threshold_level	200	Get threshold level
MID_GETLIBRARY_VERS ION	0xF3	H»M	N/A	200	Get the algorithm version number
MID_POWERDOWN	0xED	H»M	N/A	1000	The module is ready to shut down, ending the current Processing tasks, clear the file system cache
MID_DEMOMODE	0xFE	H»M	N/A	100	Entering Demo Mode
MID_ENROLL_PALM	0xA0	H»M	s_msg_enroll_data	10000	Interactive palm entry
MID_ENROLL_SINGLE_P ALM	0xA1	H»M	s_msg_enroll_data	10000	Single frame recording palm
MID_ENROLL_ITG_PALM	0xA2	H»M	s_msg_enroll_itg	200	Integrate support and expand all palm recorders Entry method
MID_PALMRESET	0xA3	H»M	N/A	200	Reset the palm algorithm status, if it is During interactive input, it will be cleared Entered direction
MID_DELUSER_PALM	0xA4	H»M	s_msg_deluser_data	200	Delete a palm registered user
MID_DELALL_PALM	0xA5	H»M	N/A	200	Delete all registered users
MID_GETUSERINFO_PAL M	0xA6	H»M	s_msg_getuserinfo_data	500	Get the information of a registered user of a palm breath
MID_GET_ALL_USERID_P ALM	0xA7	H»M	N/A	10000	Get all registered palm users Quantity and ID
MID_GET_VERSION_PAL M	0xA8	H»M	N/A	1000	Get the palm software version information

3.4 Sending and receiving module messages

3.4.1 Message Reception (H>>M)

The complete protocol format received by the module is shown in Table 3-5. The master control should send commands to the module according to this format.

Table 3-5 Message Protocol

NameSyncWord	MsgID	Size		Data	ParityCheck
Bytes 2 bytes	1 byte	2 bytes		N bytes	1 byte
Content 0xEFAA command		high eight Bit	Low eight Bit	data	checksum

The structure of the command sent is shown as follows.

The command and data definitions are shown in Table 3-6.

Table 3-6 Message description

command (* indicates carrying data)	code	data		illustrate
		Structure	content	
MID_RESET	0x10	none		none
MID_GET_STATUS	0x11	none		none
MID_VERIFAND*	0x12 s_msg_verify_data		pd_rightaway (1byte)	After successful unlocking No, cut off the power immediately (yes:1 no:0)
			timeout (1byte)	Unlock timeout (Unit: s)
MID_ENROLL*	0x13 s_msg_enroll_data		admin(1byte)	Is it set to pipe? Manager (yes:1 no:0)

			user_name(32bytes) user_name[0]:id_heb user_name[1]:id_leb	Enter user ID, Name Note: ID is reserved and can be Command switch algorithm return ID Or the master sends the ID
			face_dir or palm_dir (1byte)	User needs to enter face or palm Direction, specific parameters As shown in Table 3-7
			timeout (1 byte)	Input timeout (Unit: s)
MID_SNAPIMAGE	0x16	none		Image capture
MID_GETSAVEDIMAGAND	0x17	none		none
MID_UPLOADIMAGE*	0x18	s_msg_upload_image_data	upload_image_offset (4 byte)	Logs to be uploaded Offset
			upload_image_size (4 byte)	This upload log Size, maximum 4K
MID_ENROLL_SINGLE*	0x1D	Same as MID_ENROLL (Only one registration of face or hand direction is required)		Single posture input person Face or palms
MID_DELETE_USER*	0x20	s_msg_deluser_dafacing	user_id_heb (1byte)	To delete the user ID
			user_id_leb (1byte)	
MID_DELETE_THE_ALL	0x21	none		Delete All Users (including faces and palm)
MID_GET_	0x22	s_msg_getuserinfo	user_id_heb	Registered users

USER_INFO *		_data	(1byte)	ID
			user_id_leb (1byte)	
MID_FACE _RESET	0x23	none		Clear entry status
MID_GET_ ALL_USERS D	0x24	none		Get all registered Number of registered users and ID
MID_ENRO LL_ITG*	0x26 s_msg_enroll_itg		admin (1byte)	Is it set to pipe? Manager (yes:1 no:0)
			user_name (32byte)	Enter user name
			direction (1byte)	Not used
			enroll_type (1byte)	Registration Type: Various Single pose Status Entry
			enable_duplicate (1byte)	Is it repeated entry?
			timeout (1byte)	Input timeout (Unit: s)
			reserved (3byte)	Reserved seat
MID_GET_ VERSION	0x30	none		Get software version
MID_STAR T_OTA*	0x40	s_msg_startota_da facing	v_primary (1 byte)	Firmware version number
			v_secondary (1 byte)	
			v_revision (1 byte)	

MID_STOP_ORDER	0x41	none		none
MID_GET_OTA_STATUS	0x42	none		Get OTA updates state
MID_OTA_HEADER*	0x43	s_msg_otaheader_data	fsize_b (4 byte)	Firmware size, high in front
			num_pkt (4 bytes)	Total number of packages, high in forward
			pkt_size (2 bytes)	Pack size, high in front
			md5_sum (32 byte)	Firmware MD5 value
MID_OTA_PACKET*	0x44	s_msg_otapacket_data	pid (2 bytes)	Sub-package number
			psize (2 bytes)	Package size
			data (n is changed)	Subcontracting content
MID_INIT_ENCRYPTITHE*	0x50	s_msg_init_encryption_data	seed[4] (4 bytes)	The master sends a Random Numbers
			mode	
MID_CONFIG_BAUDRATHE*	0x51	s_msg_config_baudrate	baudrate_index (1 bytes)	Baud rate switching, 1: 115200; 2: 230400; 3: 460800; 4: 1500000
MID_SET_RELEASE_ENC_KEY*	0x52	s_msg_enc_key_number_data	enc_key_number[16] encryption sequence	
MID_SET_DEBUG_ENC	0x53	Similar to MID_SET_RELEASE_ENC_KEY		

_KEY*				
MID_GET_ LOGFILE	0x60	none		Get the length of the log file
MID_UPLO AD_LOGFI NO*	0x61	s_msg_upload _logfile_data	upload_logfile _offset[4] 4 bytes)	Logs to be uploaded Offset
			upload_logfile _size[4] (4 bytes)	This upload log Size, maximum 4K
MID_ENRO LL_BY_PIC *	0x71	s_msg_enroll_ data	admin(1byte)	Is it set to pipe? Manager (yes:1 no:0)
			user_name(32bytes) user_name[0]:id_heb user_name[1]:id_leb	Enter user ID, Name Note: ID is reserved and can be Command switch algorithm return ID Or the master sends the ID
			face_dir or palm_dir (1byte)	Just set it to 0
			timeout (1 byte)	Input timeout (Unit: s)
MID_SET_E NRROL_PA COPPER*	0x72	s_msg_param_dat a	param_data (1 byte)	Enter parameter settings, A total of 5 modes are supported Formula, detailed description See Table 3-8
MID_GET_ ENRROL_P WIRE	0x7A	none s_msg_param_data param_data (1 byte)		Get input parameters
MID_VERIF Y_BY_PIC	0x82	none		Still Image Registration

MID_SET_THRESHOLD_LEVEL*	0xD4	s_msg_algo_threshold_level	verify_threshold_level(1 byte)	Comparison threshold setting, The value range is 0~4. The higher the value, the higher the success rate Will reduce
			aliveness_threshold_live l(1 byte)	Liveness threshold setting, The value range is 0~4. The higher the value, the higher the success rate Will reduce
MID_GET_THRESHOLD_LEVEL	0xD5	none		Get Threshold
MID_POWER_DOWN	0xED	none		none
MID_GETLIBRARY_VERSION	0xF3	none		none
ONE_DEMO_MODE*	0xFE	s_msg_demomode_data	enable (1byte)	enable:1 disable: 0
MID_ENROLL_PALM	0xA0	s_msg_enroll_data	admin(1byte)	Is it set to pipe? Manager (yes:1 no:0)
			user_name(32bytes) user_name[0]:id_heb user_name[1]:id_leb	Palm Entry User ID, name Note: ID is reserved and can be Command switch algorithm return ID Or the master sends the ID
			palm_dir (1byte)	User needs to enter The direction of the palm Body parameters are shown in Table 3-7 Shown
			timeout (1 byte)	Input timeout (Unit: s)

MID_ENROLL_SINGLE_PALM	0xA1	Same as MID_ENROLL_PALM (only need to send the palm positive direction registration once)		
MID_ENROLL_ITG_PALM	0xA2	s_msg_enroll_itg	admin (1byte)	Is it set to pipe? Manager (yes:1 no:0)
			user_name (32byte)	Palm Entry User Name
			direction (1byte)	Not used
			enroll_type (1byte)	Palm registration type: Multi-gesture input or Single gesture recording
			enable_duplicate (1byte)	Is it repeated entry?
			timeout (1byte)	Input timeout (Unit: s)
			reserved (3byte)	Reserved seat
MID_PALM_RESET	0xA3	none		Clear entry status
MID_DELUSER_PALM	0xA4	s_msg_deluser_da facing	user_id_heb (1byte)	To be deleted palm User ID
			user_id_leb (1byte)	
MID_DELALL_PALM	0xA5	none		Delete Palm All user
MID_CAPABILITY SERINFO_PALM	0xA6	s_msg_getuserinfo _data	user_id_heb (1byte)	Registered palm User ID
			user_id_leb (1byte)	
MID_GET_	0xA7	none		Get all palms

ALL_USERS D_PALM			Registered users Quantity and ID
MID_GET_ VERSION_P ALM	0xA8	none	Get the palm algorithm Software Version

The input direction definition is shown in Table 3-7.

Table 3-7 Direction definition

face_dir (Face direction definition)	code	illustrate
FACE_DIRECTION_UP	0x10	Entering an upward-facing face
FACE_DIRECTION_DOWN	0x08	Recording downward-facing faces
FACE_DIRECTION_LEFT	0x04	Enter a left-facing face
FACE_DIRECTION_RIGHT	0x02	Enter a right-facing face
FACE_DIRECTION_MIDDLE	0x01	Enter a positive face
FACE_DIRECTION_UNDEFINE	0x00	Undefined, default is positive
palm_dir (Palm direction definition)	code	illustrate
PALM_DIRECTION_UP	0x10	Enter the palm facing up
PALM_DIRECTION_DOWN	0x08	Enter the palm facing down
PALM_DIRECTION_LEFT	0x04	Enter the left-facing palm
PALM_DIRECTION_RIGHT	0x02	Enter right-facing palm
PALM_DIRECTION_MIDDLE	0x01	Enter the positive palm
PALM_DIRECTION_UNDEFINE	0x00	Undefined, default is positive

Example: 0xEF 0xAA 0x10 0x00 0x00 0x10

name say	SyncWord MsgID		Size		Data	ParityCheck
Inside Allow	0xEFAA	0x10 (MID_RE SET)	0x00	0x00	none	0x10

This message is the RESET message sent by the master to the module, and the data length is 0.

3.4.2 Message sending (M>>H)

The module mainly sends three types of messages: REPLY, NOTE, and IMAGE.

The structure of the message is shown below.

(1) Sending a REPLY message

The complete protocol of the REPLY message sent by the module to the master is shown in Table 3-8.

Table 3-8 REPLY message protocol

NameSyncWordMsgID			Size		Data			ParityCheck
byte number	2 bytes	1 byte	2 bytes		N bytes			1 byte
Content 0xEFAA		MID_REPLY (0x00)	high eight Bit	Low eight Bit	s_msg_reply_data			checksum
					mid (1byte)	result (1byte)	data[0] (n bytes)	

mid indicates the task that the module is currently processing. For example, when mid = MID_ENROLL (0x13), it means that the

The message is the message that the module replies after processing the enroll task. The details of mid are shown in Table 3-9.

Table 3-9 Description of each message reply

mid (* indicates data is carried)	code	data[0]		illustrate
		Structure	content	
MID_RESET	0x10	none		none
MID_GETSTATUS*	0x11	s_msg_repy_get status_data	status (1byte)	Module status include: IDLE(0),BUSY(1),ERROR (2)
MID_VERIFY* (Only when result user_id is only available when it is MR_SUCCESS)	0x12	s_msg_reply_verify_data	user_id_heb (1byte)	Verified User ID
			user_id_leb (1byte)	
			user_name (32 bytes)	Username
			admin	Is it management

			(1byte)	member
			unlockStatus (1 byte)	Unlock process status Status (reserved)
MID_ENROLL* (Only when all directions The results are MR_SUCCESS only has user_id)	0x13	s_msg_reply_en roll_data	user_id_heb (1byte)	Registered User ID
			user_id_leb (1byte)	
			direction(1 byte)	People from all directions Face or palm Entry Status
MID_SNAPIMAG AND	0x16	none		Snap a picture picture
MID_GETSAVED IMAGE*	0x17	s_msg_reply_ge t_saved_image_ data	image_size[4] (4 bytes)	Pictures to be uploaded size, image_size[0] Put the length high 8-bit
MID_UPLOADIM AGE	0x18	image data	According to the size required by the master control Upload image	Actual image
MID_ENROLL_SI English	0x1D	Same as MID_ENROLL		none
MID_DELUSER 0x20		none		none
MID_DELALL 0x21		none		none
MID_GETUSER_I NFO*	0x22	s_msg_reply_ge tuserinfo_data	user_id_heb (1byte)	Registered User ID
			user_id_leb (1byte)	
			user_name (32bytes)	Registered User Name
			admin(1byte)	Is it management member

MID_FACERESET	0x23	none		none
MID_GET_ALL_USERS*	0x24	s_msg_reply_all_userid_data	user_counts (1 byte)	Registered User quantity
			users_id[40] (40 bytes)	All registered User ID, Use two consecutive Byte storage one ID, save first High eight bits
MID_ENROLL_ITG*	0x26	Same as MID_ENROLL		none
MID_GET_VERSION ON*	0x30	s_msg_reply_version_data	version_info[32] (32 bytes)	Version Information
MID_START_OTA	0x40	none		none
MID_STOP_OTA 0x41		none		none
MID_GET_OTA_STATUS	0x42	s_msg_reply_getotastatus_data	take_status (1 byte)	4 means in OTA status
			next_pid_e (2 bytes)	Transferring firmware packages From Starting number
MID_OTA_HEADIS	0x43	none		none
MID_OTA_PACKAND	0x44	none		none
MID_INIT_ENCRYPTION	0x50	s_msg_reply_init_encryption_d minutes	device_id[20]	Device ID (Device unique ID Other code)
MID_CONFIG_BAUDRATE	0x51	none		none
MID_SET_RELEA 0x52		none		none

SE_ENC_KEY				
MID_SET_DEBUG_ENC_KEY	0x53	none		none
MID_GET_LOGFILE	0x60	s_msg_reply_get_log_data	log_size[4] (4 bytes)	File to be transferred Piece length, log_size[0] Length: 8 Bit
MID_UPLOAD_LANDFILE	0x61	image data	According to the master control requirements size of uploaded image	Actual image
MID_ENROLL_BY_PIC	0x71	s_msg_reply_enroll_data	user_id_heb (1byte)	Registered User ID
			user_id_leb (1byte)	
			direction(1 byte)	Ignore
MID_SET_ENRROL_PARAME	0x72	none		none
MID_GET_ENRROL_PARAME	0x7A	s_msg_param_data	param (1 byte)	Get registration parameters number
MID_VERIFY_BY_PIC	0x82	Same as MID_VERIFY		none
MID_SET_THRESHOLD_LEVEL	0xD4	none		
MID_GET_THRESHOLD_LEVEL	0xD5	s_msg_algo_threshold_level	verify_threshold_lewell (1 byte)	
			aliveness_threshold_level (1 byte)	
MID_GETLIBRARY_VERSION	0xF3	s_msg_reply_library_version_data library_version_info[20] (20 byte)		Get the algorithm version This issue

MID_POWERDOWN	0xED	none		none
MID_DEMOMODAND	0xFE	none		none
MID_ENROLL_PALM (only when each Direction result Both MR_SUCCESS Only user_id)	0xA0	s_msg_reply_enroll_data	user_id_heb (1byte)	Registered palm
			user_id_leb User ID (1byte)	
			direction(1 byte)	Hands in all directions Palm entry status state
MID_ENROLL_SINGLE_PALM	0xA1	Same as MID_ENROLL_PALM		none
MID_ENROLL_ITG_PALM	0xA2	Same as MID_ENROLL_PALM		none
MID_PALMRESET	0xA3	none		none
MID_DELUSER_PALM	0xA4	none		none
MID_DELALL_PALM	0xA5	none		none
MID_GETUSERINFO_PALM	0xA6	s_msg_reply_getuserinfo_data	user_id_heb (1byte)	Registered palm
			user_id_leb User ID (1byte)	
			user_name (32bytes)	Registered palm User's name
			admin(1byte)	Is it management member
MID_GET_ALL_USERID_PALM	0xA7	s_msg_reply_all_us special_data	user_counts (1 byte)	Registered palm Number of users

			users_id[40] (40 bytes)	All palms have Registered User ID, use consecutive Two bytes store Store an ID, Store the high 8 bits first
MID_GET_VERSI ON_PALM	0xA8	s_msg_reply_versio n_data	version_info[32] (32 bytes)	Version Information

result indicates the execution result of the command, as shown in Table 3-10.

Table 3-10 Result List

result	code	illustrate
MR_SUCCESS	0	Success
MR_FAILED4_UNKNOWNREASON	5	Unknown error
MR_FAILED4_INVALIDPARAM	6	Invalid parameter
MR_FAILED4_NOMEMORY	7	Insufficient Memory
MR_FAILED4_UNKNOWNUSER	8	No user has been entered (no People & Not in Curry)
MR_FAILED4_MAXUSER	9	The maximum number of users entered has exceeded
MR_FAILED4_USERENROLLED	10	people have been entered
MR_FAILED4_LIVENESSCHECK	12	Liveness detection failed
MR_FAILED4_TIMEOUT	13	Entry or unlock timeout
MR_FAILED4_READ_FILE	19	Failed to read file
MR_FAILED4_WRITE_FILE	20	Failed to write file
MR_FAILED4_USER_REGISTER_ERR	22	User registration error
MR_FAILED4_PIC_ERROR	28	Still image reception error
MR_FAILED4_OTA_PACKET_MD5	29	OTA upgrade package receiving verification failed
MR_FAILED4_FACE_INIT_ERROR	32	Algorithm face database initialization failed

Example: 0xEF 0xAA 0x00 0x00 0x04 0x13 0x00 0x00 0x01 0x11

Table 3-11 Example of personnel entering a reply

name say	SyncWord MsgID		Size	Data	Ditch yCh

									eck
Inside Allow	0xEFAA	0x00 (MID_R EPLY)	0x00 0x04		s_msg_reply_data				0x11
					0x13 (MID_ ENROL L)	0x00 (MR_S UCCC SS)	0x00 (user _id_h eb)	0x01 (user _id_l eb)	

This message indicates that this is the REPLY message sent by the module to the main control. The data part occupies 4 bytes. The entry is successful.

And the entered user ID is 1.

(2) Sending NOTE messages

The main function of NOTE message is to actively return some information to the master control. Currently, NOTE message mainly occurs in three situations

Send: a) When booting, the module sends NID_READY to the main control; b) During the input process, the module sends the face signal to the main control

c) The module sends face information to the main control during the unlocking process. The complete protocol of the NOTE message sent by the module to the main control

As shown in Table 3-12.

Table 3-12 NOTE message protocol

NameSyncWordMsgID			Size		Data		ParityCheck
Bytes 2 bytes		1 byte	2 bytes		N bytes		1 byte
Content 0xEFAA		MID_NO THE (0x01)	high eight Bit	Low eight Bit	s_msg_note_data		checksum
					not (1byte)	data[0] (n-bytes)	

nid indicates the execution result of the algorithm in the enroll process, etc., as follows:

Table 3-13

nid (* indicates data is carried)	code	illustrate
NOT_READY	0	Module is ready
NID_FACE_STATE*	1	The algorithm is executed successfully and returns the face information s_note_data
NOT_UNKNOWNERROR	2	Unknown error
NID_OTA_DONE*	3	The data field is 0 (1 byte)
NID_PALM_STATE*	4	The algorithm is executed successfully and returns the palm status information s_note_data
NOT_PIC_RCV	5	The module is ready to receive images

The data carried by NID_FACE_STATE mainly stores face information, as follows:

Table 3-14

structure	s_note_data (left~roll not output, judged based on state)

Contents state	left	top	right	bottom	yaw	pitch	roll
Type int16_t	int16_t	int16_t	int16_t	int16_t	int16_t	int16_t	int16_t
Bytes 2bytes	2bytes	2bytes	2bytes	2bytes	2bytes	2bytes	2bytes

state: indicates the current state of the face, mainly including the following situations:

Table 3-15

Face state	code	illustrate
FACE_STATE_NORMAL	0	Normal face
FACE_STATE_NOFACE	1	No face detected
FACE_STATE_TOOUP	2	The face is too close to the image Edge, failed to record
FACE_STATE_TOODOWN	3	The face is too close to the bottom of the picture Edge, failed to record
FACE_STATE_TOOLEFT	4	The face is too close to the left of the image Edge, failed to record
FACE_STATE_TOORIGHT	5	The face is too close to the right of the image Edge, failed to record
FACE_STATE_FAR	6	The face is too far away to be Entry
FACE_STATE_CLOSE	7	The face is too close to Entry
FACE_STATE_FACE_OCCLUSION	10	Face occlusion
FACE_STATE_DIRECTION_ERROR	11	Wrong face orientation
FACE_STATE_FACE_ADD_NOT_ENOUGH	19	The number of registered faces is not enough Enough
FACE_STATE_DETECT_ERROR	20	Face detection failed
FACE_STATE_FACE_BIGANGLE	21	The face angle is too big
FACE_STATE_UNKNOWNREASON	30	Unknown error
Palm state	code	illustrate
PALM_STATE_NORMAL	0	Palm detection
PALM_STATE_NOPALM	1	Palm not detected

PALM_STATE_TOOUP	2	The palm is too close to the picture Edge, failed to record
PALM_STATE_TOODOWN	3	The palm is too close to the picture Edge, failed to record
PALM_STATE_TOOLEFT	4	The palm is too close to the left of the picture Edge, failed to record
PALM_STATE_TOORIGHT	5	The palm is too close to the right of the picture Edge, failed to record
PALM_STATE_FAR	6	The palms are too far away to Entry
PALM_STATE_CLOSE	7	The palms are too close to Entry
PALM_STATE_PALM_OCCLUSION	8	Palm occlusion
PALM_STATE_DIRECTION_ERROR	9. Wrong palm direction entered	
PALM_STATE_PALM_POSE_ERRO R	10	Wrong hand posture

Example: 0xEF 0xAA 0x01 0x00 0x01 0x00 0x00

name say	SyncWord MsgID		Size		Data		ParityCheck
Inside Allow	0xEFAA	0x01 (MID_N OTE)	0x00 0x01		s_msg_note_data		0x00
					0x00 (NOT_READY)	none	

This message indicates that this is a NOTE message sent by the module to the master. The data length is 1 byte. The module is ready to

Be prepared to receive additional commands.

3.5 Master control receiving message process

The message sent by the master receiving module can follow the process shown in Figure 3-1.

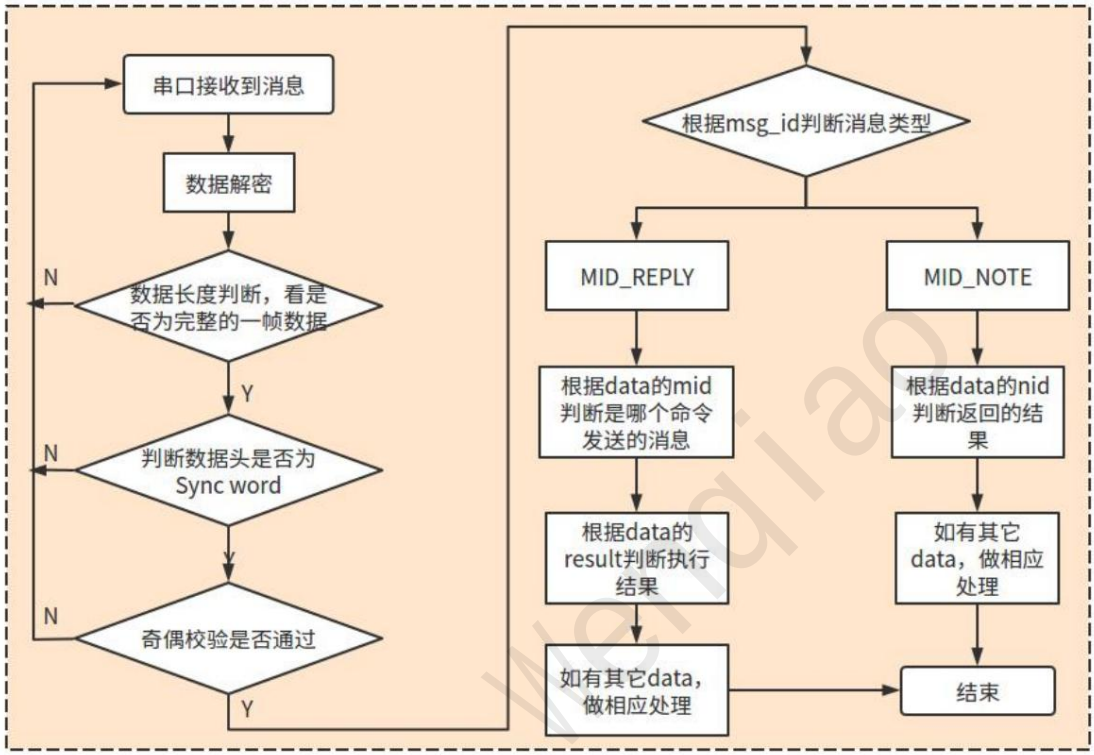


Figure 3-1 Master control message receiving process

3.6 General message processing flow

Figure 3-2 General message processing flow

3.7 Power on and off process

The master actively controls the power on and off of the module, as shown in Figure 3-3. When the module is powered on, it enters the startup process.

When the module enters the standby state, it will notify the master control MSG::NOTE::READY, and the master control will start sending command messages.

Process and return the result. When there is no message to send, the master calls MSG::POWERDOWN, and the module processes and returns.

The main control can be powered off.

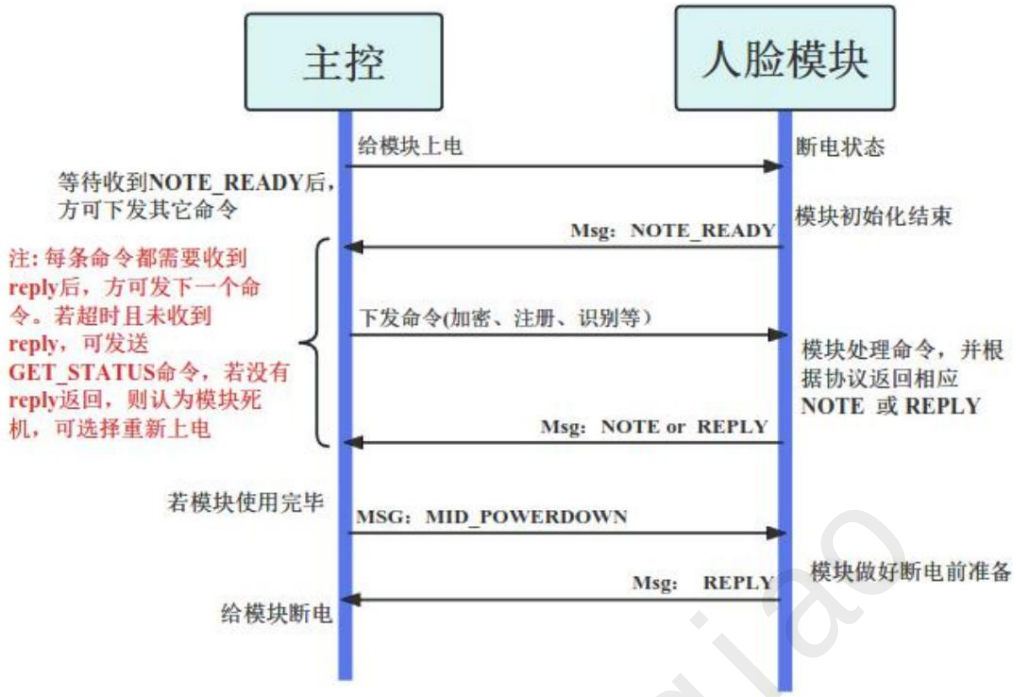


Figure 3-3 Power on and off process

3.8 Multi-posture input

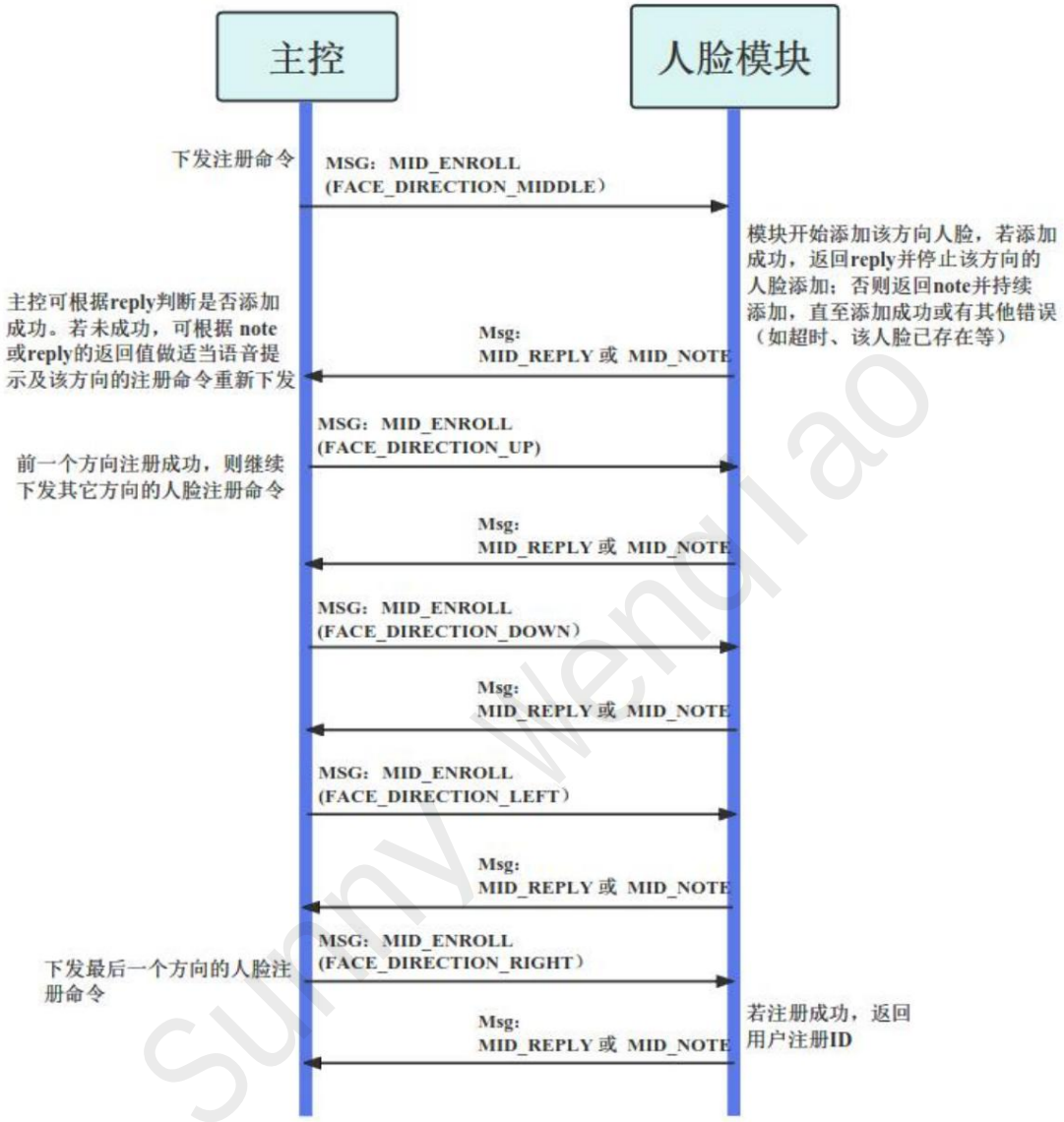


Figure 3-4 Face entry process interaction flow

During the recording process, the master needs to send 5 registration commands, each time changing the registration direction (the first direction specifies FACE_DIRECTION_MIDDLE, and the order of the remaining four directions can be arbitrary). Each direction is entered successfully or failed

If the module fails, it will return the input result in the form of a REPLY message; otherwise, the module will return the input result in the form of a NOTE message.

Only when the master receives the reply from 5 directions and the result of 5 directions = MR_SUCCESS,

The registration is successful, and face and palm entry are supported.

If there is a sudden power outage during the data entry process, the previously entered information will not be saved.

3.9 Single posture input

The main difference between single-frame recording and recording process is that single-frame recording only requires one image to be recorded successfully.

There is no need to interact five times to complete registration, and face and palm entry are supported.

3.10 Static Image Import

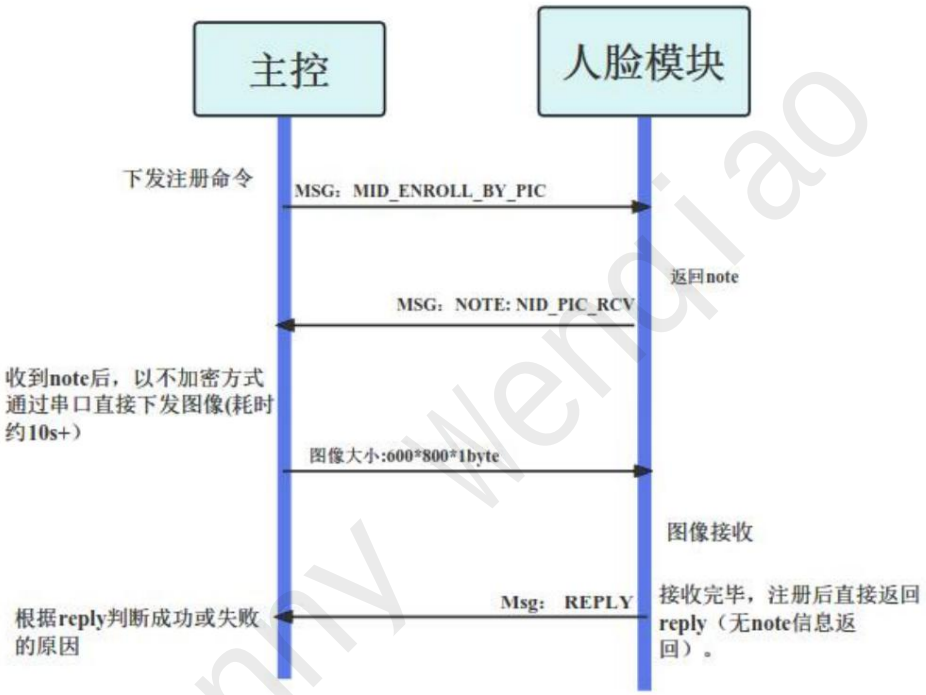


Figure 3-5 Static image registration interaction process

Static image registration only requires one registration command, and the registration direction can be set to 0. This image is only used for pre-registration.

Only when the real person in the picture reaches the door lock, the module detects the person and then performs live registration and recognition.

If the image size is not 600*800 bytes, the module will return a reply of result = MR_FAILED4_PIC_ERROR. After receiving this reply, the master control can choose to

resend the static image registration command or other operations.

3.11 Static Image Recognition

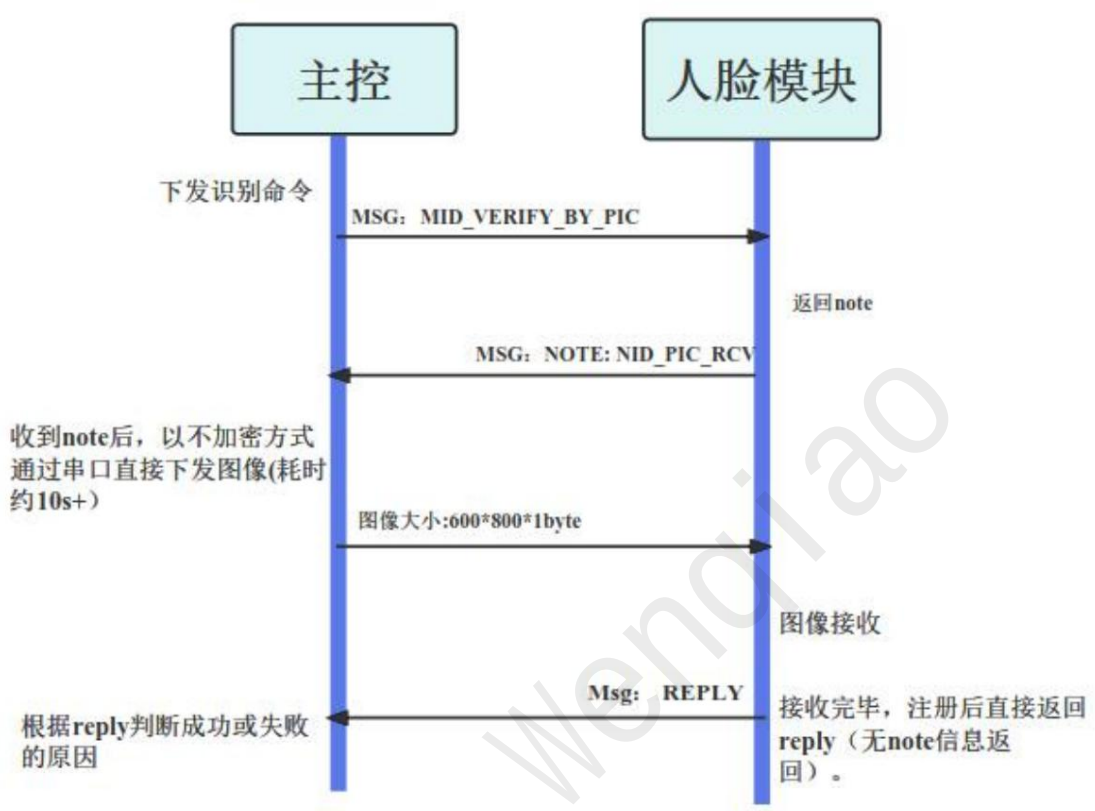


Figure 3-6 Static image recognition process

The module supports static image recognition, which does not perform liveness detection but only face recognition.

If the size of the received image is not 600*800 bytes, the module will return result = MR_FAILED4_PIC_ERROR

After receiving this reply, the master control can choose to resend the static image registration command or perform other operations.

3.12 Unlocking Process

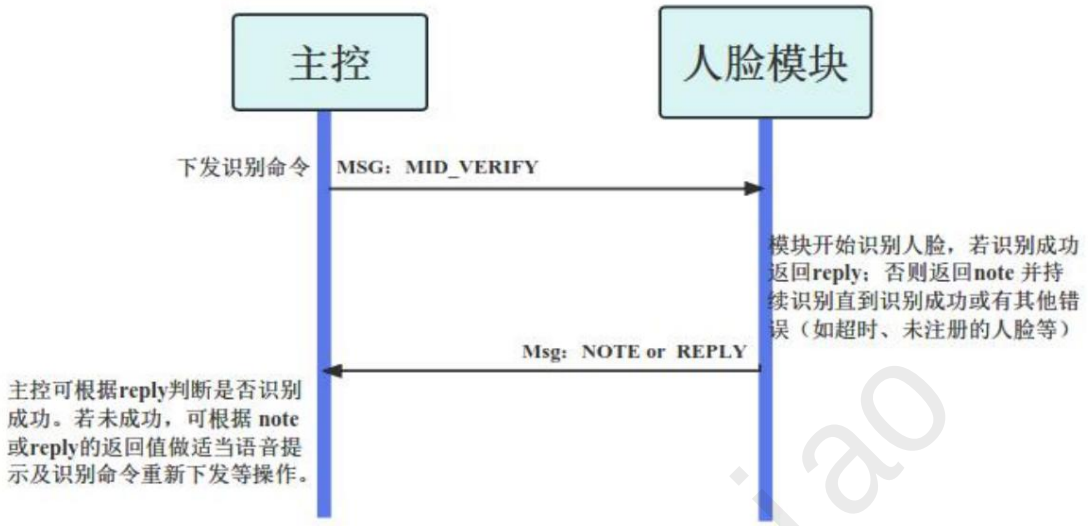


Figure 3-7 Face recognition interaction process

The unlocking process switches to the corresponding recognition mode to unlock the door according to whether it is a palm or a face.

3.13 Encrypted Communication Process

The module currently supports two encryption modes: simple encryption and AES. The encrypted communication process is shown in Figure 3-7.

1 The master controller powers on the module;

2. The module sends NOTE READY (plain text) to the main control;

3. The master controller needs to send the MID_SET_RELEASE_ENC_KEY (plain text) command to the module when it powers on for the first time.

Set the 16 Bytes sequence, and no need to send it again after power-on.

4 The master sends MID_INIT_ENCRYPTION (plain text) with 4 Bytes random sequence and encryption mode to

Modules;

5 The module and the main control both use a custom private protocol to generate a 16-Byte password based on the random sequence.

Both parties use the same algorithm to generate the same password. After that, both parties will use this password for encryption and decryption in this session;

6 The module returns the status and uses random password AES/SMPL to encrypt and send the product serial number to the main control;

7. The master decrypts and identifies the device ID. After confirming the identity, it starts processing the required instructions, such as input or unlocking;

8 The master uses a randomly generated password and AES/SMPL encryption to encrypt instructions and data and send them to

Modules;

9 The module decrypts the command using the password decrypted in step 4, determines the legitimacy of the command and processes the command;

10 The module encrypts the instruction processing result and data with the password decrypted in step 4, and sends them to the main control.

3.14 OTA Upgrade

ÿ The master sends MID_START_OTA (encrypted text) to notify the module to enter OTA mode; ÿ The module responds OK (plain text, all subsequent operations are in plain text mode), and the module enters OTA mode; ÿ The master sends MID_CONFIG_BAUDRATE to configure a higher baud rate. After the module responds OK, the host synchronously sets the same baud rate. Serial port data transmission can only start after a delay of 100ms. (This configuration

The baud rate is optional during the configuration process, and the default baud rate is 115200)

ÿ The master sends MID_OTA_HEADER and sends the firmware-related information to the module; ÿ The module responds with OK; ÿ The master sends MID_GET_OTA_STATUS (this configuration process is optional); ÿ The module responds with OTA status and the starting packet number (this configuration process is optional); ÿ The master sends the upgrade package MID_OTA_PACKET in a loop, and after receiving OK, it continues to the next package. ÿ After the module receives the firmware, it starts upgrading. ÿ After the module upgrade is completed, it responds to the master with a NOTE message carrying the upgrade result.

(NID_OTA_DONE), the module automatically restarts after 10 seconds. The timeout period for the master control to wait for the module to reply with NID_OTA_DONE information after sending the upgrade package must be greater than 120 seconds.

As shown in the figure, during the OTA upgrade process, you can use MID_GET_OTA_STATUS to get the OTA upgrade status. When the status is in red, it means that the received upgrade file is incorrect (transmission error). or files not required by the module).

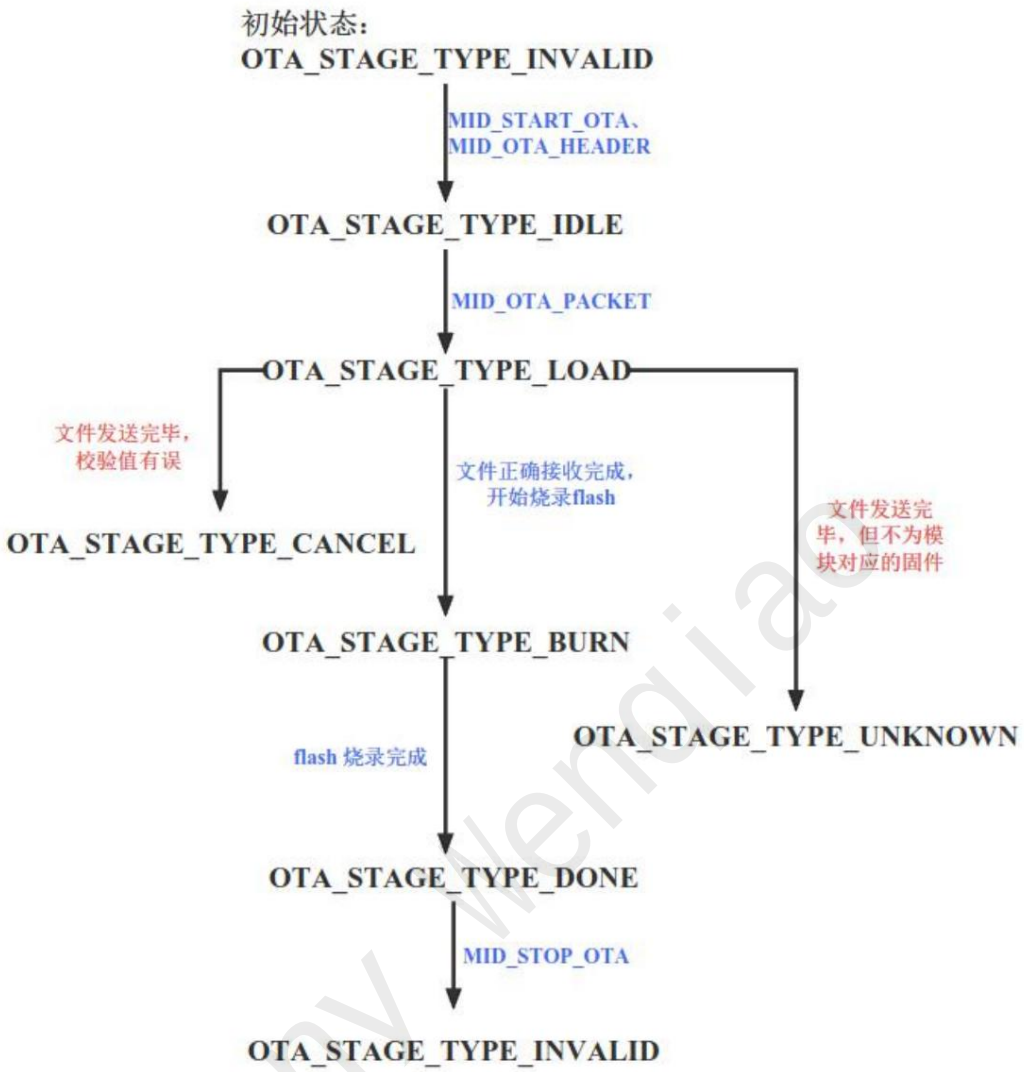


Figure 3-8 OTA status diagram

3.15 Image Capture

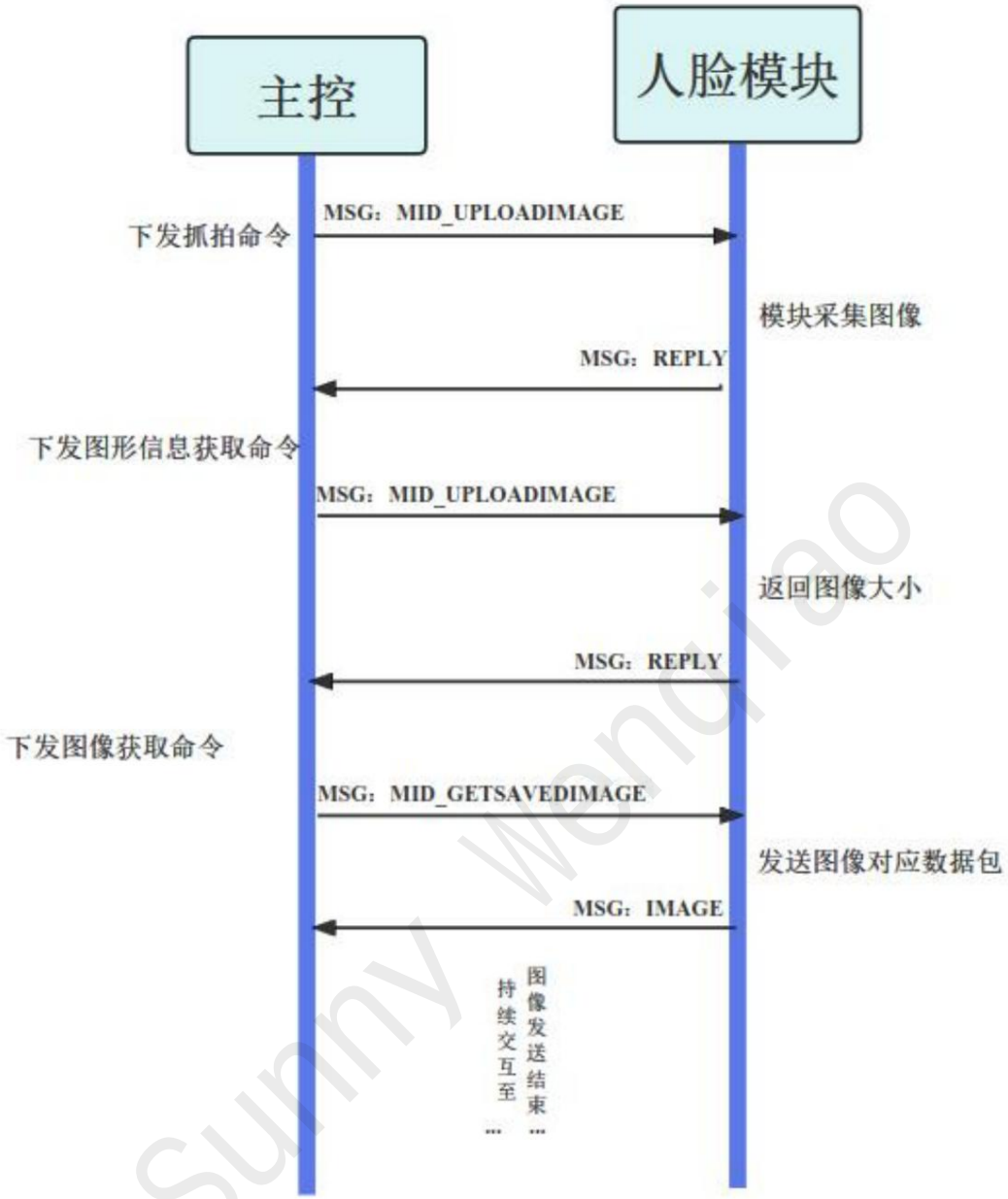


Figure 3-9 Image capture interaction process

The picture capture process is shown in Figure 3-9. This function is used to trigger the main control when abnormalities such as identification or registration occur.

After receiving the image, the master control can save the image to analyze the cause. To shorten the image transmission time,

before increasing the baud rate.

3.16 Supplementary Notes

3.16.1 MID_REPLY

Each command sent by the master to the module will eventually receive a MID_REPLY reply from the module, which contains

The command mid sent by the master, the execution result of the command result, and the data that may be returned data data.

```
typedef struct {
    uint8_t mid; // the command(message) id to reply (the request message ID)
    uint8_t result; // command handling result: success or failed -> s_msg_result
    uint8_t data[0];
} s_msg_reply_data;
```

The command execution result result may be the following:

```
/* message result code */
typedef enum {
    MR_SUCCESS = 0, // success
    MR_FAILED4_UNKNOWNREASON = 5, // UNKNOWN_ERROR
    MR_FAILED4_INVALIDPARAM = 6,                    // invalid param
    MR_FAILED4_NOMEMORY = 7,                        // no enough memory
    MR_FAILED4_UNKNOWNUSER = 8, // no user enrolled
    MR_FAILED4_MAXUSER = 9,                         // exceed maximum user number
    MR_FAILED4_USERENROLLED = 10, // this user has been enrolled
    MR_FAILED4_LIVENESSCHECK = 12,
    MR_FAILED4_TIMEOUT = 13,                        // exceed the time limit
    MR_FAILED4_READ_FILE = 19, // read file failed
    MR_FAILED4_WRITE_FILE = 20, // write file failed
    MR_FAILED4_USER_REGISTER_ERR = 22,
    MR_FAILED4_PIC_ERROR = 28,
    MR_FAILED4_OTA_PACKET_MD5 = 29,
    MR_FAILED4_ALG_INIT_ERROR = 32,
```

```
} msg_result; /* message result code end */
```

The returned data varies depending on the mid. In the header file message.h, it starts with "s_msg_reply_"

The structures are all the data carried by each instruction when sending a reply message. Take verify as an example:

```
/* message reply data definitions */
typedef struct {
        uint8_t user_id_heb;
        uint8_t user_id_leb;
        uint8_t user_name[USER_NAME_SIZE];
        uint8_t admin;
        uint8_t unlockStatus;
} s_msg_reply_verify_data;
```

3.16.2 MID_NOTE

MSG::NOTE is the information that the module actively reports to the main control. For example, when the module is powered on, it will actively send a message to the main control.

A NOTE message indicates that it is READY and can receive commands from the master. After the master receives the NOTE message,

You can start sending entry or unlock commands.

The data contained in the NOTE message is:

```
typedef struct {
        uint8_t not;                // note id
        uint8_t data[0];
} s_msg_note_data;
/* module -> host note end */
```

3.16.3 MID_RESET

After the master sends this command, the module will cancel the previously executed command (such as input, unlock, etc.) and return to

Return to STANDBY status.

3.16.4 MID_GETSTATUS

After the master sends the command, the module returns its current status, which mainly includes:

MS_STANDBY(0): The module is in idle state, waiting for the master control command

MS_BUSY(1): The module is in working state

MS_ERROR(2): Module error, cannot work normally

MS_OTA(4): The module is in OTA mode

3.16.5 MID_VERIFY

MSG::VERIFY is the most commonly used function, which switches during the recognition process based on whether a palm or a face is detected. Enter the corresponding recognition mode. Once a palm is detected, it enters the palm recognition mode.

During the unlocking process, the module returns two messages: NOTE and REPLY.

NOTE: It mainly returns the status of the face or palm in the current frame.

REPLY mainly returns the final result after the unlocking process is completed. The messages that may be returned include: MR_SUCCESS, MR_FAILED4_USERENROLLED, etc., see Table 3-11 for details.

MR_SUCCESS indicates that the unlocking is successful. REPALY will carry the unlocking method (reserved function).

The default is normal unlocking. The data structure is as follows:

```
typedef struct {
    uint8_t user_id_heb;
    uint8_t user_id_leb;
    uint8_t user_name[USER_NAME_SIZE];
    uint8_t admin;
    uint8_t unlockStatus;
} s_msg_reply_verify_data;
```

[unlockStatus] represents the eye open or closed state when unlocking (reserved function, currently not returned).

3.16.6 MID_ENROLL/MID_ENROLL_SINGLE

MSG::ENROLL is a commonly used function that supports face and palm input. If a face is detected, it will enter Face registration, once the palm is detected, it will enter palm registration, and the accompanying data is as follows:

```
// enroll user
```

```
typedef struct {
        uint8_t admin; // the user will be set to admin
        uint8_t user_name[32];
         uint8_t direction;
         uint8_t timeout;
} s_msg_enroll_data;
```

[admin] indicates that the person entering the data is an administrator; [timeout] is the timeout period during the entry process (in seconds).

The default value is 10s, which can be set when sending the input command. The timeout period can be set by the user at will, and the maximum value is no more than 255s;

[direction] indicates the direction of the face or palm that the user wants to enter. There are five valid directions. The definition of face direction is as follows:

```
// msg face direction
typedef enum {
        FACE_DIRECTION_UP = 0x10,                       // face up
        FACE_DIRECTION_DOWN = 0x08,                     // face down
        FACE_DIRECTION_LEFT = 0x04,                     // face left
        FACE_DIRECTION_RIGHT = 0x02, // face right
        FACE_DIRECTION_MIDDLE = 0x01, // face middle
        FACE_DIRECTION_UNDEFINE = 0x00 // face undefine
} face_dir;
```

The palm direction is defined as follows:

```
// msg palm direction
typedef enum {
        PALM_DIRECTION_UP = 0x10,                       // palm up
        PALM_DIRECTION_DOWN = 0x08,                     // palm down
        PALM_DIRECTION_LEFT = 0x04,                     // palm left
        PALM_DIRECTION_RIGHT = 0x02, // palm right
        PALM_DIRECTION_MIDDLE = 0x01, // palm middle
        PALM_DIRECTION_UNDEFINE = 0x00 // palm undefine
} palm_dir;
```

During the input process, two messages, NOTE and REPLY, will also be returned.

NOTE: It mainly returns the status of the face or palm in the current frame.

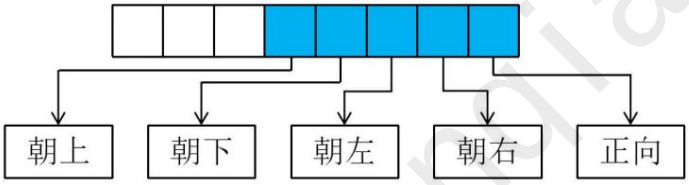
REPLY mainly returns the final result after the input process is completed. For details of the possible returned messages, please see Table 3-11.

When MR_SUCCESSS is returned, it means the entry is successful. REPLY will also return some information, as follows

As shown:

```
typedef struct {
    uint8_t user_id_heb;
    uint8_t user_id_leb;
    uint8_t direction;
} s_msg_reply_enroll_data;
```

[direction] indicates the face or palm input status, and the lower 5 bits of the data represent the face direction from high to low. Up, down, left, right and positive, as shown in the figure below, 1 means the direction has been entered, 0 means the direction Not entered.



3.16.7 MID_ENROLL_ITG

MSG::ENROLL_ITG is a supplement and extension of the 1.15.6 ENROLL command. The entry type can be specified in the parameter structure, and the user can choose whether to repeat the entry;

The accompanying data is as follows:

```
// enroll user intergrated
typedef struct {
    uint8_t admin; // the user will be set to admin, 1:admin user, 0:normal user
    uint8_t user_name[USER_NAME_SIZE];
    uint8_t direction; // reference FACE_DIRECTION_*
    uint8_t enroll_type; // reference FACE_ENROLL_TYPE_*
    uint8_t enable_duplicate; // enable user enroll duplicatly, 1:enable, 0:disable
// when enroll_type is equal to FACE_ENROLL_TYPE_RGBÿ the
enable_duplicate can be set to 2, it means that cant duplicate enroll with
username
    uint8_t timeout; // timeout unit second default 10s
```

```
    uint8_t reserved[3]; // reserved faild
} s_msg_enroll_itg;
```

[enroll_type] FACE_ENROLL_TYPE_INTERACTIVE: interactive enrollment;

FACE_ENROLL_TYPE_SINGLE is for single frame recording.

[enable_duplicate] has the following

functions: 0: The same person cannot be recorded, but the user name can be repeated.

1: The same person can be entered repeatedly, and the user name can also be repeated.

2: The same person can register, but the user name cannot be repeated (only valid when registering in RGB).

3.16.8 MID_DELUSER

MSG::DELUSER is used to delete a registered face user, specified by user_id.

3.16.9 MID_DELALL

MSG::DELALL is to delete all registered face and palm user information.

3.16.10 MID_GETUSERINFO

The master specifies the registered user to be returned through the user id. After receiving the command, the module will return the information of the specified user.

Information mainly includes the following:

```
typedef struct {
    uint8_t user_id_heb;
    uint8_t user_id_leb;
    uint8_t user_name[32];
    uint8_t admin;
} s_msg_reply_getuserinfo_data;
```

3.16.11 MID_FACERESET

MSG::FACERESET message is used to reset the SDK algorithm status. If interactive input is in progress, send this message.

The message will reset the entered status and all entered directions will be cleared.

3.16.12 MID_POWERDOWN

Before the master controller powers off the module, it must first send the MID_POWERDOWN command. After receiving the command, the module will

The master controller will respond to the power failure and send a reply message to the master controller. After receiving the REPLY message, the master controller needs to wait for 100ms before

Power off, forced power off may cause damage to the hardware.

3.16.13 MID_DEMOMODE

After the master sends this command, the module enters the demonstration mode. In this mode, no feature authentication will be performed during the module authentication process.

Comparison, that is, everyone can unlock, but liveness detection will still be performed.

3.16.14 MID_ENROLL_BY_PIC

MSG::ENROLL_RY_PIC is used for temporary user remote registration, that is, when the owner is not at home, he wants to authorize others

This command can be used to unlock the door through face recognition. In this case, the authorized person needs to take a face photo and send it to the owner.

When the authorized person reaches the door lock, the module automatically completes the actual live body registration and recognizes the person.

Don't unlock it.

```
// enroll user
typedef struct {
    uint8_t admin; // the user will be set to admin
    uint8_t user_name[32];
     face_dir direction;
     uint8_t timeout;
} s_msg_enroll_data;
```

[admin] indicates whether the person entering the data is an administrator; [timeout] is the timeout period during the entry process (in seconds).

Since it includes the time for image transmission (10s), it is recommended to set it to more than 15s. It can be set when sending the registration command,

and the maximum value is no more than 255s; since only one registration command needs to be sent, [direction] can use 0.

Since it is not a dynamic entry, the image entry only returns a REPLY message.

REPLY mainly returns the final result after the input process is completed. For details of the possible returned messages, please see Table 3-11.

When MR_SUCCESSS is returned, it means the entry is successful. REPLY will also return some information, as follows

As shown:

```
typedef struct {
    uint8_t user_id_heb;
    uint8_t user_id_leb;
    uint8_t direction;
} s_msg_reply_enroll_data;
```

[direction] indicates the face registration status. When registration is successful, the value is 0x1f.

3.16.15 MID_SET_ENRROL_PARAME & MID_GET_ENRROL_PARAME

This command is used to set the input parameters. The structure is defined as follows:

```
// param
typedef struct {
    uint8_t param;
} s_msg_param_data;
```

The values that can be set for param and their corresponding descriptions are as follows in Table 3-16:

Table 3-16

Mode param_data value	illustrate
Mode 1 0x00	Add mode: automatically generate ID, no repeated registration is allowed; If the face exists, an error and the corresponding ID are returned
Mode 2 (Default mode)	0x01 Added mode: automatically generate ID, allow repeated registration
Mode 3 0x02	Added mode: The master controller issues the ID, and repeated registration is not allowed; If the ID exists, the existing ID is returned; If the face exists, an error and the corresponding ID are returned.

Mode 4 0x03 Added mode: The master controller sends the ID and allows repeated registration;		
Mode 5 0x04 Update mode: The master must send the ID to update the specified ID information; If there is no one in the face database, no face will be returned; If the ID does not exist, an error message indicating that there is no corresponding ID is returned; If the ID exists, delete the original specified ID user information and update it to Current user information.		

The setting of this parameter can satisfy both the testing of the lock control manufacturer and the use of the final product.

In the test phase, you can choose to use 5 modes, 0 to 4. Mode 0 to 1, the module automatically generates an ID, and the master can select it by itself

Whether to back up the user ID and other information, or only manage and back up the user ID and other information by the module. Mode 2~4

The ID is sent by the master control. At this time, the first two bytes of the user_name field of s_msg_enroll_data are reused to store the ID.

Therefore, the actual name needs to be offset two bytes backwards.

3.16.16 MID_SET_THRESHOLD_LEVEL&MID_GET_THRESHOLD_LEVEL

This interface is used to modify the algorithm security level. Different manufacturers and users have different understandings of security requirements.

The standards are inconsistent, so the module opens a security level interface to set the security level of liveness and verify.

0-4 represent: low, lower, normal, higher, and high respectively.

The system default security level is: 2 Normal.

The structure sent by the master control is as follows:

```
typedef struct {
    uint8_t verify_threshold_level;                 // level 0~4, safety from low to high, default 2
    uint8_t liveness_threshold_level; // level 0~4, safety from low to high, default 2
} s_msg_algo_threshold_level;
```

Disclaimer: It is recommended to use the default settings. A security level that is too high may lead to a lower pass rate; a security level that is too low may lead to

Please use this setting with caution as it may cause misidentification issues.

3.16.17 MID_GET_ALL_USERID

Get the number of all registered users and the IDs of all registered users.

3.16.18 MID_SET_RELEASE_ENC_KEY

The master controller must send this command when it is powered on for the first time. The module will encrypt and save the data.

The encrypted communication described in 3.12.15 is performed. The data will not be lost when power is off, and the data will not be lost.

```
typedef struct {
    uint8_t enc_key_number[16]; //
} s_msg_enc_key_number_data;
```

3.16.19 MID_INIT_ENCRYPTION

The master sends a random number, and the module generates a password after receiving the random number, and sends a reply according to the new password.

news.

The structure of the random number sent by the master is as follows:

```
typedef struct {
    uint8_t seed[4];
    uint8_t mode;
} s_msg_init_encryption_data;
```

The data structure returned after the module encryption is completed is as follows:

```
typedef struct {
    uint8_t device_id[20];
} s_msg_reply_init_encryption_data;
```

3.16.20 MID_SNAPIMAGE

This command is to capture a picture. When the main control issues this command, the module will immediately capture the picture.

Save the image in memory (invalid after power failure).

3.16.21 MID_GETSAVDIMAGE & MID_UPLOADIMAGE

The captured images will be stored in the local path of the module. When the master obtains the captured images, it first calls the MID_GETSAVEDIMAGE command to obtain the size of the image to be uploaded, and then calls MID_UPLOADEIMAGE multiple times to obtain the image as needed. The data upload_image_offset[4] carried in the MID_UPLOADEIMAGE instruction indicates the offset of the uploaded image (for example, when the offset is 0, the image will be transmitted from the beginning), and upload_image_size[4] indicates the size of the uploaded image. The maximum Supports 4096 bytes.

3.16.22 MID_GET_LOGFILE & MID_UPLOAD_LOGFILE

Get the log file size to be uploaded through MID_GET_LOGFILE. MID_UPLOAD_LOGFILE is used to upload the log to the master. The usage of these two instructions is exactly the same as that of MID_GETSAVDIMAGE & MID_UPLOADIMAGE.

reference.

Note: The log is stored in the memory and is volatile when power is off.

3.16.23 MID_GETLIBRARY_VERSION

This message can obtain the version number information of the face algorithm library in the current module.

The structure sent by the master control is as follows:

```
typedef struct {
  uint8_t library_version_info[20];
} s_msg_reply_library_version_data;
```

3.16.24 CMD

The messages from HOST >> MODULE and MODULE >> HOST are sent (received) according to the following structure:

```
typedef struct zng_cmd_s {
        uint8_t flag_h;
        uint8_t flag_l;
```

```
    uint8_t msg_id;
    uint8_t length_h;
    uint8_t length_l;
    uint8_t data[4096];
    uint8_t ParityCheck;
} zng_cmd_t;
```

3.16.25 MID_ENROLL_SINGLE_PALM

This command is mainly used for palm registration, single image registration method, and the accompanying data is as follows:

```
// enroll user
typedef struct {
    uint8_t admin; // the user will be set to admin
    uint8_t user_name[32];
     palm_dir direction;
     uint8_t timeout;
} s_msg_enroll_data;
```

[admin] indicates that the person entering the data is an administrator; [timeout] is the timeout period during the entry process (in seconds).

The default value is 10s, which can be set when sending the input command. The timeout period can be set by the user at will, and the maximum value is no more than 255s;

[direction] indicates the direction of the palm that the user wants to enter. There are 5 valid palm directions (please keep your palm aligned with the camera at present.

The header can be registered in parallel), and is defined as follows:

```
/* msg palm direction */
typedef enum {
    PALM_DIRECTION_UP = 0x10,                    // palm up
    PALM_DIRECTION_DOWN = 0x08,                  // palm down
    PALM_DIRECTION_LEFT = 0x04,                  // palm left
    PALM_DIRECTION_RIGHT = 0x02, // palm right
    PALM_DIRECTION_MIDDLE = 0x01, // palm middle
    PALM_DIRECTION_UNDEFINE = 0x00 // palm undefine
} palm_dir;
```

During the input process, two messages, NOTE and REPLY, will also be returned.

NOTE: It mainly returns the state of the palm in the current frame.

REPLY mainly returns the final result after the input process is completed. For details of the possible returned messages, please see Table 3-11.

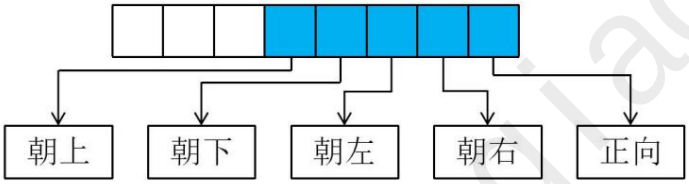
When MR_SUCCESSS is returned, it means the entry is successful. REPLY will also return some information, as follows

As shown:

```
typedef struct {
    uint8_t user_id_heb;
    uint8_t user_id_leb;
    uint8_t direction;
} s_msg_reply_enroll_data;
```

[direction] indicates the palm input status, and the lower 5 bits of the data represent the palm direction from high to low, respectively.

Downward, left, right and forward, as shown in the figure below, 1 means the direction has been entered, and 0 means the direction has not been entered.



3.16.26 MID_DELUSER_PALM

MID_DELUSER_PALM is to delete a palm registered user, specified by user_id.

3.16.27 MID_DELALL_PALM

MID_DELALL_PALM is to delete the user information registered for all palms.

3.16.28 MID_GETUSERINFO_PALM

The master specifies the registered user to be returned through the user id. After receiving the command, the module will return the specified palm user

The information mainly includes the following information:

```
typedef struct {
    uint8_t user_id_heb;
    uint8_t user_id_leb;
    uint8_t user_name[32];
    uint8_t admin;
```

```
} s_msg_reply_getuserinfo_data;
```

3.16.29 MID_GET_VERSION_PALM

This message can obtain the version number information of the palm algorithm library in the current module

The structure sent by the master control is as follows:

```
typedef struct {
  uint8_t library_version_info[20];
} s_msg_reply_library_version_data;
```

3.17 Module Basic Protocol Application Example

Due to the complexity of actual use, the following only shows the actual serial port when registration/deletion/recognition is successful.

Sending and receiving data is taken as an example, and other situations are not described in detail.

3.17.1 Registration

a. Clear the entry status, MID_FACERESET, the following data are all in hexadecimal.

(H>>M):ef aa 23 00 00 23

(M>>H):ef aa 00 00 02 23 00 21

b. Enter the face, MID_ENROLL, this process needs to be entered 5 times in a row. If a single entry fails

If it fails, it will be re-entered until it times out. The following data are all in hexadecimal.

(H>>M):ef aa 13 00 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 01 0a 3b

(M>>H):ef aa 00 00 05 13 00 ff ff 01 17

(H>>M):ef aa 13 00 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 10 0a 2a

(M>>H):ef aa 00 00 05 13 00 ff ff 11 07

(H>>M):ef aa 13 00 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 08 0a 32

(M>>H):ef aa 00 00 05 13 00 ff ff 19 0f

(H>>M):ef aa 13 00 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 04 0a 3e

(M>>H):ef aa 00 00 05 13 00 ff ff 1d 0b

(H>>M):ef aa 13 00 23 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 02 0a 38

(M>>H):ef aa 00 00 05 13 00 00 01 1f 08

Special note: Failed due to face: (M>>H):ef aa 01 00 11 01 01 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 10

3.17.2 Delete

Delete all users, MID_DELALL, the following data are all in hexadecimal.

(H>>M):ef aa 21 00 00 21

(M>>H):ef aa 00 00 02 21 00 23

3.17.3 Identification

Identify the user, MID_VERIFY, the following data are all in hexadecimal.

(H>>M):ef aa 12 00 02 00 0a 1a

(M>>H):ef aa 00 00 26 12 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 c8 fd

4. Notes on using the module

Please pay attention to the following matters when using the product:

1. Relative humidity of this product: 10%~93%, no condensation.

2. The operating temperature of this product: -20°~60°

3. Storage temperature of this product: -40°~70°

4. Static electricity treatment: The whole product is grounded, supporting contact discharge ±8KV; human body contact chip ±2KV; whole machine, assembly

   And anti-static treatment during transportation.