Capacitive Fingerprint Reader (B) Serial Command Manual

CONTENT

1. Overview

1.1. Serial Parameter

Parameters of Serial port:

Start bit: 1bit

Data bit: 8bit

Stop bit: 1bit

Parity bit: None

Baud rate: 9600/19200/38400/57600/115200/230400/460800/921600 (default 115200)

Data transmitting order:

The data transmission (Byte and Word) follows the order than the LSB first transmitted.

Host should only send commands after getting the response from Target Module

1.2. Main Functions

Add fingerprint: Add the fingerprint to the internal database of Capacitive Fingerprint Reader (B) (hereafter Module), the capacity of the internal database is 3000 fingerprints.

Delete fingerprint: Detect single fingerprint according to ID or delete all the fingerprints.

Verify fingerprint (1:1): Verify the fingerprint with the certain fingerprint in database according to the ID

Verify fingerprint (1: N): Verify the fingerprint with all the fingerprints in database

Other functions:

1. Upload/Download eigenvalue: The Module support to upload the eigenvalue of fingerprints to Host or Download from Host to the Module.
2. Upload/Download image: The Module support to upload the image of fingerprint to Host or Download from Host to the Module
3. Check if the fingerprint data is broken in range (ID)
4. Read the total of fingerprints and the details in database
5. Set/Read the configuration values of the Module (Security level, enable/disable auto-learn, etc.)
6. Set/Read the Module SN
7. Read the information of the Module (Hardware, Software)

For more information about the Module, please refer to the Command Table Chapter

2. Communication Protocol

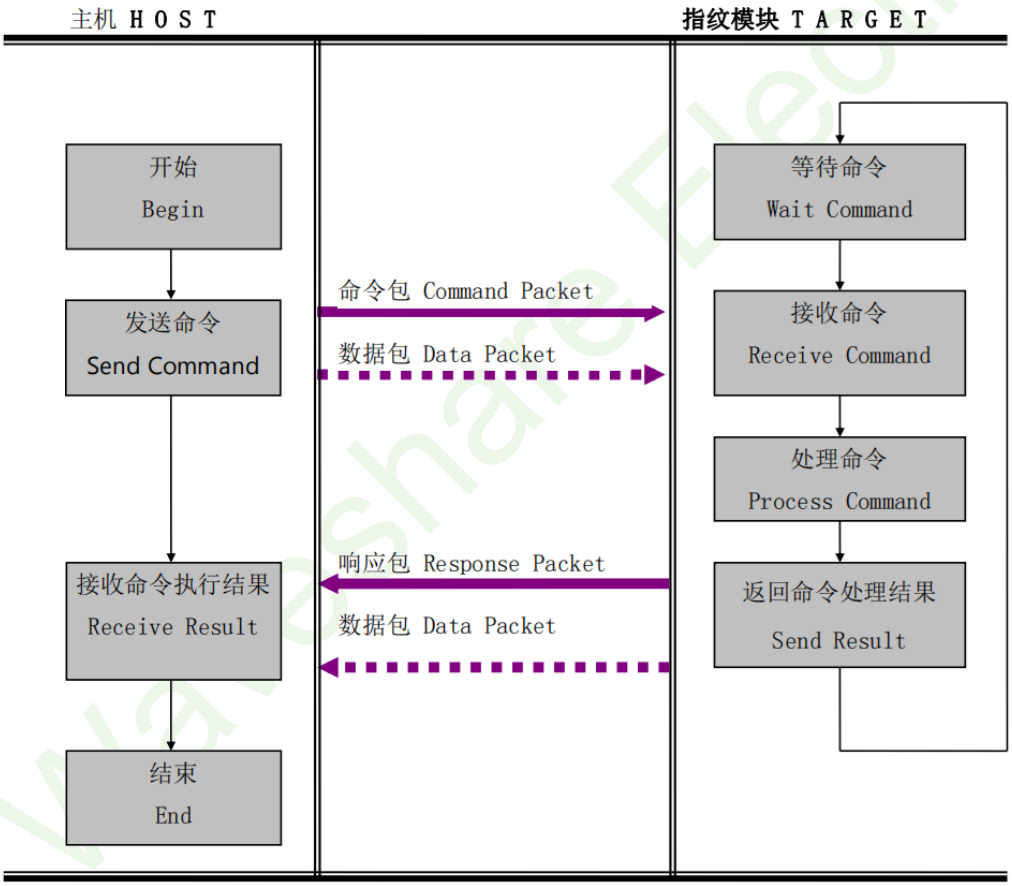
After powering, the firmware of Module requires time to boot and initialize, the Host could only send commands after the Module initializes.

The Module will send a handshake signal (0x55) to the HOST via UART after initializing.

The Host can go into working mode after getting the handshake signal

For the Host, it can also add a 280ms delay before sending commands after powering instead of waiting for the handshake signal.

2.1. Communication Flow Chat



Note:

All the commands should be sent/received one by one.

The Host should only send commands after getting the response from the Module.

2.2. Communication Packet

2.2.1. Command Packet

- The Command Packet contains the commands which are sent from Host to Module
- All the commands sent from the Host should be transmitted via the Command Packet
- The length of the Command Packet is 26 bytes.

2.2.2. Response Packet

- The Response Packet contains the response data which are sent from Module to Host
- All the commands are completed/handled after receiving the Response Packet
- The length of the Response Packet is 26 bytes

2.2.3. Data Packet

- If the contents of the command or response are larger then 16 bytes, it uses the Data Packet to transmit
- Before sending the Data Packet, the Host should use the Command Packet to inform the Module the length of the Data Packet
- The maximum length of the Data Packet is 500 bytes.

2.3. Packet Format

2.3.1. Identifier Code of the Packet

The first 2 bytes of the Packet is the Identifier Code of the Packet, which identify the type of the Packet.

Packet Type	Identifier Code
Command Packet	0xAA55
Response Packet	0x55AA
Data Packet (command)	0xA55A
Data Packet (data)	0x5AA5

TABLE 1 PACKET IDENTIFIER CODE

2.3.2. Command Packet Format

Note: L means the Low bits of the bytes, H means the High bits of the bytes

PREFIX		SID	DID	CMD		LEN		DATA				CKS	
0x55	0xAA	Source ID	Target ID	L	H	L	H	D0	D1	…	D15	L	H
0	1	2	3	4	5	6	7	8	9	…	23	24	25

The format of the Command Packet:

OFFSET	TYPE	DATA TYPE	SIZE	DESCRIPTION
0	PREFIX	WORD	2 bytes	Packet Identify code

2	SID	BYTE	1 byte	Source Device ID
3	DID	BYTE	1 byte	Destination Device ID
4	CMD	WORD	2 bytes	Command Code
6	LEN	WORD	2 bytes (=n, n<16)	Length of DATA
8	DATA	Byte Array	16 bytes	Command Parameter
24	CKS	WORD	2 bytes	Check Sum: The last 2 bytes of the check sum (PREFIX to all the DATA)

2.3.3. Response Packet Format

PREFIX		SID	DID	RCM		LEN		RET		DATA			CKS		
0xAA	0x55	Source ID	Target ID	L	H	L	H	L	H	D0	D1	...	D15	L	H
0	1	2	3	4	5	6	7	8	9	10	11	...	23	24	25

The format of the response packet

OFFSET	TYPE	DATA TYPE	SIZE	DESCRIPTION
0	PREFIX	WORD	2 bytes	Packet Identify code
2	SID	BYTE	1 byte	Source Device ID
3	DID	BYTE	1 byte	Destination Device ID
4	RCM	WORD	2 bytes	Response Code
6	LEN	WORD	2 bytes (=n, n<16)	Length of RET and DATA
8	RET	WORD	2 bytes	Result Code (0: Success; 1: Fail)
10	DATA	Byte Array	14 bytes	Response Data
24	CKS	WORD	2 bytes	Check Sum: The last 2 bytes of the checksum (PREFIX to all the DATA)

2.3.4. Command Data Packet Format

PREFIX		SID	DID	CMD		LEN		DATA				CKS	
0x5A	0xA5	Source ID	Target ID	L	H	L	H	D0	D1	...	Dn-1	L	H
0	1	2	3	4	5	6	7	8	9	...	8+n-1	8+n	8+n+1

The format of the Command Data Packet:

OFFSET	TYPE	DATA TYPE	SIZE	DESCRIPTION
0	PREFIX	WORD	2 bytes	Packet Identify code
2	SID	BYTE	1 byte	Source Device ID
3	DID	BYTE	1 byte	Destination Device ID
4	CMD	WORD	2 bytes	Command Code

6	LEN	WORD	2 bytes (=n, n<500)	Length of RET and DATA
8	DATA	Byte Array	n bytes	Command Parameter
24	CKS	WORD	2 bytes	Check Sum: The last 2 bytes of the checksum (PREFIX to all the DATA)

The Host should first send the Command packet before the Command Data Packet, it will make the Module enter the waiting status.

In the DATA field of the Command packet, it should be set with the length of Command Data Packet.

The Host should only send the Command Data Packet after confirming that the Module is in the status that waiting for Data Packet.

2.3.5. Response Data Packet Format

PREFIX		SID	DID	RCM		LEN		RET		DATA				CKS	
0xA5	0x5A	Source ID	Target ID	L	H	L	H	L	H	D0	D1	…	Dn-3	L	H
0	1	2	3	4	5	6	7	8	9	10	11	…	8+n-1	8+n	8+n+1

The format of the Response Data Packet Format

OFFSET	TYPE	DATA TYPE	SIZE	DESCRIPTION
0	PREFIX	WORD	2 bytes	Packet Identify code
2	SID	BYTE	1 byte	Source Device ID
3	DID	BYTE	1 byte	Destination Device ID
4	RCM	WORD	2 bytes	Response Code
6	LEN	WORD	2 bytes (=n, n<500)	Length of RET and DATA
8	RET	WORD	2 bytes	Result Code (0: Success; 1: Fail)
10	DATA	Byte Array	n-2 bytes	Response Data
24	CKS	WORD	2 bytes	Check Sum: The last 2 bytes of the checksum (PREFIX to all the DATA)

Note: The module would send the Response Data Packet if the data sent is larger than 14 bytes.

3. About the Command

3.1. Definition

ImageBuffer: This is the buffer for saving the fingerprint image while communicating

RamBuffer: This is the buffer for saving the fingerprint eigenvalue while communicating.

There are three RamBuffer in the Module: Ram Buffer0, Ram Buffer1 and the Ram Buffer2

Note: The data in ImageBuffer and RamBuffer lose after powering off

Database: All the fingerprint data are saved in the FLASH Memory, which is caused as database hereby, users can write/read the fingerprint data by CMD_STORE_CHAR / CMD_LOAD_CHAR commands.

3.2. Command List

No.	Command Name	Code	Function
1	CMD_TEST_CONNECTION	0x0001	Test the connection of Module and Host
2	CMD_SET_PARAM	0x0002	Set parameters of Module
3	CMD_GET_PARAM	0x0003	Get the parameters of Module
4	CMD_GET IMAGE	0x0020	Capture image from the sensor and save it to ImageBuffer
5	CMD_FINGER_DETECT	0x0021	Detect the finger (Finger touched or not)
6	CMD_UP_IMAGE	0x0022	Upload the image to Host from the ImageBuffer
7	CMD_DOWN_IMAGE	0x0023	Download the image from Host to the ImageBuffer
8	CMD_SOTRE_CHAR	0x0040	Save the fingerprint data from certain Ram Buffer to the database with ID
9	CMD_LOAD_CHAR	0x0041	Read the fingerprint and save it to Ram Buffer (0/1/2) according to ID
10	CMD_UP_CHAR	0x0042	Upload the fingerprint from Ram Buffer to the Host
11	CMD_DOWN_CHAR	0x0043	Download the fingerprint from Host to the certain Ram Buffer (0/1/2)
12	CMD_DEL_CHAT	0x0044	Detect the fingerprints from database in certain range
13	CMD_GET_EMPTY ID	0x0045	Get the minimum ID which is not used
14	CMD_GET_STATUS	0x0046	Get the status of the ID
15	CMD_GET_BROKENT_ID	0x0047	Check the broken fingerprints in the certain range

16	CMD_GET_ENROLL_COUNT	0x0048	Check the numbers of fingerprint in database
17	CMD_GENERATE	0x0060	Generates fingerprint data according to the image in ImageBuffer and save it to Ram Buffer
18	GMD_MERGE	0x0061	Generate a new fingerprint template by the fingerprints data in the Ram Buffer (two or three data) Note: The new fingerprint template will be saved in Ram Buffer0
19	GMD_MATCH	0x0062	Verify two fingerprints in Ram Buffer (1:1)
20	CMD_SEARCH	0x0063	Verify the fingerprint in Ram Buffer with all the fingerprints in database (1: N)
21	CMD_VERIFY	0x0064	Verify the fingerprint in Ram Buffer with a certain fingerprint in database (1:1)
22	CMD_SET_MODULE_SN	0x0008	Set the SN code of Module
23	CMD_GET_MODULE_SN	0x0009	Get the SN code of Module
24	CMD_GET_ENROLLED_ID_LIST	0x0049	Get the list of fingerprints in database
25	CMD_ENTER_STANDY_STATE	0x000C	Set the Module enter sleep mode

4. Commands Descriptions

4.1. CMD_TEST_CONNECTION (0x0001)

[Function]

Check the connection status between Host and the Module.

The Host should fist send this command to check the connection status.

If the module response Fail or doesn't response, it is taken as disconnect, abnormally working or incorrect baud rate.

[Sequence]

The Module response ERR_SUCCESS if the connect successfully.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0001	
LEN	0	
DATA	None	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0001	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Example]

Host command: 55 AA 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01

Module response: AA 55 01 00 01 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 01

4.2. CMD_SET_PARAM (0x0002)

[Function]

According to the Parameter Type, set the parameters of Module, like Device ID, Security Level, Baudrate, Duplication Check and Auto Learn.

[Sequence]

1. If the Parameter Type is invalid, the Module response ERR_INVALID_PARAM
2. If the Parameter Value is invalid, the Module response ERR_INVALID_PARAM
3. If all the Type and value is valid, the Module is set and response result

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0002	
LEN	5	
DATA	1 byte	Parameter Type
	5 bytes	Parameter Value
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0002	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Parameter Type]

Parameter Type	Parameter Value
0	The Device ID of Module, Range: 1~255
1	The Security Level of Module, range: 1~5 (default: 3) Higher the Security Level, lower the FAR and higher the FRR, that is if the Level is higher, the correct rate of verification is higher, however, it will make the fingerprint difficult to verify.
2	Enable/Disable the Duplication Check. 1: Enable, it will check if the same fingerprint was added while using CMD STORE CHAR command. 0: Disable the checking.
3	Set the Baudrate of UART, range: 1~8 (default 5) 1: 9600bps; 2: 19200bps; 3: 38400bps; 4: 57600bps; 5:115200bps; 6: 230400bps; 7: 460800bps; 8: 921600bps;
4	Enable/Disable the Auto Learn Function

	1: Enable, the Module will smart update the fingerprint data while using the CMD_SEARCH and CMD_VERIFY command 0: Disable.

[Example 1] Set the Baudrate to 57600bps

Host Command: 55 AA 00 00 02 00 05 00 03 04 00 00 00 00 00 00 00 00 00 00 00 00 00 0d 01

Module Response: AA 55 01 00 02 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 01

[Example 2] Set the Baudrate to 115200bps

Host Command: 55 AA 00 00 02 00 05 00 03 05 00 00 00 00 00 00 00 00 00 00 00 00 00 0e 01

Module Response: AA 55 01 00 02 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 01

4.3. CMD_GET_PARAM (0x0003)

[Function]

According to the Parameter Type, read parameters of the Module, like the Device ID, Security Level, Baudrate, Duplication Check and the Auto Learn. Please refer to the last section about the Parameter Type.

[Sequence]

1. If the Parameter Type is invalid, the Module responses ERR_INVALID_PARAM
2. If the Parameter Type is valid, the Module response related parameter value.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0003	
LEN	5	
DATA	1 byte	Parameter Type
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0003	
LEN	Fail: 6; Success: 2	
RET	Result Code	
DATA	4 bytes	Success: Parameter Value
CKS	2 bytes	Checksum

[Examples] Read the current Security Level (Response Level 3)

Host Command: 55 AA 00 00 03 00 01 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 01

Module Response : AA 55 01 00 03 00 06 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00 0C 01

4.4. CMD_GET_IMAGE (0x0020)

[Function]

Capture image from the sensor and save it to the ImageBuffer.

[Sequence]

Capture image from the sensor, if the image is captured successfully, the Module returns ERR_SUCCESS, otherwise, it returns error code.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0020	
LEN	0	
DATA	None	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0020	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Example] Capture image

Host Command: 55 AA 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1F 01

Module Response : AA 55 01 00 20 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 22 01

4.5. CMD_FINGER_DETECT (0x0021)

[Function]

Check if the finger touches the sensor.

[Sequence]

The Module will return if the finger is detected while receiving the command

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0021	
LEN	0	
DATA	None	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0021	
LEN	Success: 3; Fail: 2	
RET	Result Code	
DATA	1 byte	Success: status 1: Finger is detected 0: Finger is not detected
CKS	2 bytes	Checksum

[Example 1] Finger is not detected

Host Command: 55 AA 00 00 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 01

Module Response : AA 55 01 00 21 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 24 01

[Example 2] Finger is detected

Host Command: 55 AA 00 00 21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 01

Module Response: AA 55 01 00 21 00 03 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 25 01

4.6. CMD_UP_IMAGE_CODE (0x0022)

[Function]

According to the Image Type, upload the image data from ImageBuffer to Host. If the Image Type is 0, the Module sends Full image (242*266), if it is 1, the Module sends Quarter image (keep 1 of 4 pixels): 121*133

[Sequence]

1. If the Image Type is invalid, the Module return ERR_INVALID_PARAM
2. Use the Command/Response Packet, the Module send the size of image to Host
3. Use the Response Data Packet, the Module divide the image data send them (496bytes per packet) to Host

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0022	
LEN	1	
DATA	1 byte	Image Type (0: Full; 1: Quarter)
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0022	
LEN	Success: 6; Fail: 2	
RET	Result Code	
DATA	2 bytes	Success: Width of the Image data Full: 242 Quarter: 121
	2 bytes	Success: Height of the Image data Full: 266 Quarter: 133
CKS	2 bytes	Checksum

If the Module response Success, it will send the Response Data Packet to Host, every Packet include 496 bytes until all the image data are sent.

PREFIX	0x5AA5	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0022	
LEN	4 + Length of the Image data (maximum 496 bytes)	
RET	ERR_SUCCESS	
DATA	Length of Image data (2 bytes) + Image data	
CKS	2 bytes	Checksum

Note:

1. Before you use this command to send the image, you need to first use the CMD_GET_IMAGE command to save the finger image to ImageBuffer.

2. High resolution mode (Full Mode) Width*Height: 242*266
3. Low resolution mode (Quarter Mode) Width*Height: 121*133

[Examples] Upload the Full Image data

Host Command: 55 AA 00 00 22 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 22 01

Module Response : AA 55 01 00 22 00 06 00 00 00 F2 00 0A 01 00 00 00 00 00 00 00 00 00 00 25 02

Data Packet (Module):

A5 5A 01 00 22 00 F4 01 00 00 F0 01 [Image Data 496 bytes] [2 bytes Checksum]

… … Total 129 packets

A5 5A 01 00 22 00 88 01 00 00 84 01 [The last 388 bytes of image data] [2 bytes Checksum]

The last packet includes 388 bytes image data

4.7. CMD_DOWN_IMAGE (0x0023)

[Function]

Save the Image data sent from Host to ImageBuffer, the Host will send the image data (496 bytes per packet) as well as the ID of the Image data

Note: Image format: resolution: 500DPI; Grayscale: 8 bit; Pixels: 242*266

[Sequence]

1. If the Height of the Width of Image is wrong, the Module return ERR_INVALID_PARAM
2. Use the Response Packet, the Module response ERR_SUCCESS
3. Receive the Command Data Packet and save to ImageBuffer

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0023	
LEN	4	
DATA	2 bytes	Image width: 242
	2 bytes	Image Height: 266
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA

SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0023	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Command Data Packet]

PREFIX	0xA55A	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0023	
LEN	2 + Length of Image data	
DATA	ID of Image data (2 bytes) + Image Data	
CKS	2 bytes	Checksum

[Response Data Packet]

PREFIX	0x5AA5	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0023	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Example]

Host Command: 55 AA 00 00 23 00 04 00 F2 00 0A 01 00 00 00 00 00 00 00 00 00 00 00 00 23 02

Module Response : AA 55 01 00 23 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 25 01

Command Data Packet (Host):

5A A5 00 00 23 00 F2 01 00 00 [Image Data 496 bytes] [2 bytes Checksum]

Response Data Packet (Module):

A5 5A 01 00 23 00 02 00 00 00 25 01

The Length of Response Data Packet is only 12 bytes.

… … Total 129 command data packets which has 496 image data and 129 response packets

Command Data Packet (Host)

5A A5 00 00 23 00 84 01 82 00 [The last 388 bytes of image data] [2 bytes Checksum]

The last packet includes 388 bytes image data

Response Data Packet (Module):

A5 5A 01 00 23 00 02 00 00 00 25 01

4.8. CMD_STORE_CHAR (0x0040)

[Function]

Copy the Fingerprint Template data from Ram Buffer to the database

[Sequence]

1. If the template data is invalid, the Module returns ERR_INVALID_TMPL_NO
2. If the ID of RamBuffer is invalid, the Module returns ERR_INVALID_BUFFER_ID
3. If the Duplication Check is set to OFF, Module will directly save the Template data to database according to the ID and response
4. If the Duplication Check is set to ON, Module will first verify the Template data with all the fingerprints in data.
   If verify successfully, it means that the fingerprint was exist, the Module return ERR_DUPLICATON_ID and the ID of Template data
   If verify failed, the Template data of RamBuffer will be saved in database and response

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0040	
LEN	4	
DATA	2 bytes	Template ID
	2 bytes	Ram Buffer ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA
SID	Source Device ID
DID	Destination Device ID
CMD	0x0040
LEN	The LEN is related to Result Code: 2 or 4 If result code is ERR_DUPLOCATION_ID: LEN = 4 Or: LEN = 2
RET	Result Code

DATA	2 bytes	The data is related to the result code: 0 or the ID of duplication Template If the result code is ERR_DUPLICATION_ID: the ID Or: 0
CKS	2 bytes	Checksum

[Example] Save the Template data from RamBuffer0 to database and set the ID to 1

Host Command: 55 AA 00 00 40 00 04 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 44 01

Module Response : AA 55 01 00 40 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 42 01

4.9. CMD_LOAD_CHAR (0x0041)

[Function]

Get the specified Template data from database and save it to RamBuffer

[Sequence]

1. If the Template ID is invalid, the Module return ERR_INVALID_TMPL_NO
2. If the Template doesn't exist, the Module return ERR_TMPL_EMPY
3. If the RamBuffer ID is invalid, the Module return ERR_INVALID_BUFFER_ID
4. Save the Target ID to the RamBuffer and return ERR_SUCCESS

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0041	
LEN	4	
DATA	2 bytes	Template ID
	2 bytes	Ram Buffer ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0041	
LEN	2	
RET	Result Code	
DATA	2	
CKS	2 bytes	Checksum

[Example] Read the Template data (ID 1) to RamBuffer0

Host Command: 55 AA 00 00 41 00 04 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 01

Module Response : AA 55 01 00 41 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 43 01

4.10. CMD_UP_CHAR (0x0042)

[Function]

Upload the Template data from RamBuffer to Host

[Sequence]

1. If the RamBuffer ID is invalid, the Module returns ERR_INVALID_BUFFER_ID
2. Use the Command/Response Packet, send the size of Template data to Host
3. Use the Response Data Packet, send the Template data to Host

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0042	
LEN	2	
DATA	RamBuffer ID	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0042	
LEN	4	
RET	ERRU_SUCCESS or ERR_FAIL	
DATA	Success: Length of Template data Fail: Error code	
CKS	2 bytes	Checksum

[Response Data Packet]

PREFIX	0x5AA5
SID	Source Device ID
DID	Destination Device ID
CMD	0x0042

LEN	2 + Template Record Size (498)	
RET	ERRU_SUCCESS	
DATA	Template Record Data	
CKS	2 bytes	Checksum

Note:

Before you use this command, you need to first use one of the CMD_GENERATE, GMD_DOWN_CHAR or CMD_LOAD_CHAR commands to save template data to RamBuffer

[Example]

Host Command: 55 AA 00 00 42 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 43 01

Module Response : AA 55 01 00 42 00 04 00 00 00 F2 01 00 00 00 00 00 00 00 00 00 00 00 39 02

Response Data: A5 5A 01 00 42 00 F4 01 00 00 [498 bytes Template data] [2 bytes Checksum]

4.11. CMD_DOWN_CHAR (0x0043)

[Function]

Download Template data from Host and save it to the RamBuffer

[Sequence]

1. The Host send a Command Packet let the Module enter the waiting status. The Length of Template data is included in the Command Packet.
2. The Module will check the Command Packet, if the command is wrong, it returns ERR code and finished the work. If the Length is wrong, it returns ERR_INVALID_PARAM. If the Length is correct, the Module return Response Packet and enter the waiting status.
3. The Host send Template data by Command Data Packet to the Module.
4. If the RamBuffer ID in Command Data Packet is invalid, the Module returns ERR_INVALID_BUFFER_ID.
5. If the Checksum of Command Data Packet is wrong, the Module returns ERR_INVALID_TEMPL_DATA.
6. The Template data received will be saved to specified RamBuffer and the Module returns ERR_SUCCESS.

[Command]

PREFIX	0xAA55
SID	Source Device ID
DID	Destination Device ID
CMD	0x0043
LEN	2
DATA	2 + Length of Template data

CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0043	
LEN	4	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Command Data Packet]

PREFIX	0xA55A	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0043	
LEN	2 + Template Record Size (498)	
DATA	RamBuffer ID + Template data (498 bytes)	
CKS	2 bytes	Checksum

[Response Data Packet]

PREFIX	0x5AA5	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0043	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

Note:

The Template data in RamBuffer2 will be deleted after using CMD_SEARCH, CMD_VERIFY, CMD_GENERATE, CMD_STORE_CHAR, CMD_DEL_CHAR, CMD_GET_EMPTY_ID, CMD_GET_STATUS, GET_BROKEN_ID, CMD_GETN_ENROLL_COUNT commands, in this case, we recommend you use other RamBuffer instead of the RamBuffer2.

[Example]

Host Command: 55 AA 00 00 43 00 02 00 F4 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 39 02

Module Response : AA 55 01 00 43 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 01

Command Data: 5A A5 00 00 43 00 F4 01 00 00 [498 bytes Template data] [2 bytes Checksum]

Response Data: A5 5A 01 00 43 00 02 00 00 00 45 01

4.12. CMD_DEL_CHAT (0x0044)

[Function]

Delete the Template data from database in range (Begin ID ~ End ID)

[Sequence]

1. If the range is invalid, the Module returns ERR_INVALID_PARAM.
2. If the Template data is empty in the range, the Module returns ERR_TMPL_EMPTY.
3. Detect all the Templates in the range and return result.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0044	
LEN	4	
DATA	2 bytes	Begin Template ID
	2 bytes	End Template ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0044	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Example] Detect the Template data in range 1 - 3000 from database

Host Command: 55 AA 00 00 44 00 04 00 01 00 BB 08 00 00 00 00 00 00 00 00 00 00 00 00 0B 02

Module Response : AA 55 01 00 44 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 46 01

4.13. CMD_GET_EMPTY_ID (0x0045)

[Function]

Get the minimum free ID for Template in range (begin ID - end ID).

[Sequence]

1. If the range is invalid, the Module returns ERR_INVALID_PARAM
2. Search the minimum free ID in specified range. If there is free ID, Module returns the ID value, otherwise, returns ERR_EMPTY_ID_NOEXIST.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0045	
LEN	4	
DATA	2 bytes	Begin Template ID
	2 bytes	End Template ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0045	
LEN	Success: 4; Fail: 2	
RET	Result Code	
DATA	2 bytes	Success: The minimum free ID
CKS	2 bytes	Checksum

[Example] Check the free ID in range 1-2000, the result is ID 11

Host Command: 55 AA 00 00 45 00 04 00 01 00 D0 07 00 00 00 00 00 00 00 00 00 00 00 00 20 02

Module Response : AA 55 01 00 45 00 04 00 00 00 0B 00 00 00 00 00 00 00 00 00 00 00 00 00 54 01

4.14. CMD_GET_STATUS (0x0046)

[Function]

Check if the ID was used

[Sequence]

1. If the ID is invalid, the Module returns ERR_INVALID_TMPL_NO
2. If the ID was used, the Module returns 1, otherwise, returns 0

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0046	
LEN	4	
DATA	ID of Template	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0046	
LEN	Success: 3; Fail: 2	
RET	Result Code	
DATA	2 bytes	Success: Status of ID 1: used 0: free
CKS	2 bytes	Checksum

[Example1] Check the ID = 1 status, Response free

Host Command: 55 AA 00 00 46 00 02 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 48 01

Module Response : AA 55 01 00 46 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 49 01

[Example2] Check the ID = 1 status, Response used

Host Command: 55 AA 00 00 46 00 02 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 48 01

Module Response : AA 55 01 00 46 00 03 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 4A 01

4.15. CMD_GET_BROKEN_ID (0x0047)

[Function]

Check if broken template data in the range (begin ID ~ end ID), sometimes the template data may be broken because of suddenly shutdown while writing Flash. Host can check broken template anytime and delete the broken template data.

[Sequence]

1. If the range is valid, the Module returns ERR_INVALID_PARAM

2. If broken templates are detected, the Module returns the number of templates and the ID of first broken template, otherwise, it returns 0

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0047	
LEN	4	
DATA	2 bytes	Begin ID
	2 bytes	End ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0047	
LEN	Success: 6; Fail: 2	
RET	Result Code	
DATA	2 bytes	Success: Number of broken templates
	2 bytes	Success: First ID of the broken template
CKS	2 bytes	Checksum

[Example] Check the broken template ID in range 1-200 (no broken)

Host Command: 55 AA 00 00 47 00 04 00 01 00 C8 00 00 00 00 00 00 00 00 00 00 00 00 00 13 02

Module Response : AA 55 01 00 47 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 4D 01

4.16. CMD_GET_ENROLL_COUNT (0x0048)

[Function]

Check the sum of templates in the range (begin ID ~ end ID)

[Sequence]

1. If the range is invalid, the Module returns ERR_INVALID_PARAM
2. The module returns the number of templates

[Command]

PREFIX	0xAA55
SID	Source Device ID

DID	Destination Device ID	
CMD	0x0048	
LEN	4	
DATA	2 bytes	Begin ID
	2 bytes	End ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0048	
LEN	Success: 4; Fail: 2	
RET	Result Code	
DATA	2 bytes	Success: Number of templates
CKS	2 bytes	Checksum

[Example] Check the templates number in range 1-200 (total 10)

Host Command: 55 AA 00 00 48 00 04 00 01 00 C8 00 00 00 00 00 00 00 00 00 00 00 00 14 02

Module Response : AA 55 01 00 48 00 04 00 00 00 0A 00 00 00 00 00 00 00 00 00 00 00 00 56 01

4.17. CMD_GENRATE (0x0060)

[Function]

Generate template data according to the image data which are saved in ImageBuffer.

[Sequence]

1. If the RamBuffer ID is invalid, the Module returns ERR_INVALID_BUFFER_ID.
2. Check if the image data in ImageBuffer is available, if the image data is bad, the Module returns ERR_ABD_QUALITY.
3. Save the template data to specified RamBuffer and returns ERR_SUCCESS.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0060	
LEN	2	
DATA	2 bytes	RamBuffer ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0060	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

Note:

Because this command should use image data from ImageBuffer to generate template, you should first use CMD_GET_IMAGE command to get image data and save it in ImageBuffer.

[Example] Generate template data from ImageBuffer and save it to RamBuffer0

Host Command: 55 AA 00 00 60 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 61 01

Module Response : AA 55 01 00 60 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 62 01

4.18. CMD_MERGE (0x0061)

[Function]

Merge the templates in RamBuffer and generate a new template, save the new template to specified RamBuffer. The count should be 2 or 3, if the count is 2, the Module merges the templates of RamBuffer0 and RamBuffer1, if the count is 3, the Module merges the templates of three Ram Buffers.

[Sequence]

1. If the RamBuffer ID is invalid, the Module returns ERR_INVALID_BUFFER_ID.
2. If the generate count is invalid, the Module returns ERR_GEN_COUNT.
3. The module generates a template according to the generate count, returns Error code if generated failed.
4. Save the template to specified RamBuffer and returns ERR_SUCCESS.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0061	
LEN	3	
DATA	2 bytes	RamBuffer ID

	1 byte	Count (2/3)
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0061	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Example] Merge treen templates from RamBuffer and generate a new one

Host Command: 55 AA 00 00 61 00 03 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00 66 01

Module Response : AA 55 01 00 61 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 63 01

4.19. CMD_MATCH (0x0062)

[Function]

Match templates of two Ram Buffer

[Sequence]

1. If the RamBuffer IDD are invalid, the Module returns ERR_INVALID_BUFFER_ID
2. Compare the templates of two RamBuffer and returns result. If they match successfully, the Module returns ERR_SUCESS in RET bytes and the smart update result in DATA bytes, otherwise, it returns ERR_VERIFY in RET bytes.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0062	
LEN	4	
DATA	2 bytes	First RamBuffer ID
	2 bytes	Second RamBuffer ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA

SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0062	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Example] Match the templates of RamBuffer 0 and RamBuffer 1, the result is success

Host Command: 55 AA 00 00 62 00 04 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 66 01

Module Response : AA 55 01 00 62 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 64 01

4.20. CMD_SEARCH (0x0063)

[Function]

Compare the template of Ram buffer with all the templates in database in range (begin ID ~ end ID)

[Sequence]

1. If the RamBuffer ID is invalid, the Module returns ERR_INVALID_BUFFER_ID.
2. If the range is invalid, the Module returns ERR_INVALID_BUFFER_ID.
3. If there are not templates in the database, the Module returns ERR_ALL_TMPL_EMPTY.
4. Compare the template of the RamBuffer with templates in database and return the result. If match successfully, the Module returns ERR_SUCCESS in RET bytes and the ID and smart update result in DATA bytes. Otherwise, it returns ERR_IDENTIFY in RET bytes.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0063	
LEN	6	
DATA	2 bytes	RamBuffer ID
	2 bytes	Begin Template ID
	2 bytes	End Template ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA
SID	Source Device ID

DID	Destination Device ID	
CMD	0x0063	
LEN	Success: 5; Fail: 2	
RET	Result Code	
DATA	Success: Template ID (2 bytes) + Smart update result (1 byte)	
CKS	2 bytes	Checksum

[Example] Compare the template of RamBuffer0 with database (1-200), get the result that ID=8

Host Command: 55 AA 00 00 63 00 06 00 00 00 01 00 C8 00 00 00 00 00 00 00 00 00 00 00 31 02

Module Response : AA 55 01 00 63 00 05 00 00 00 08 00 01 00 00 00 00 00 00 00 00 00 00 00 71 01

4.21. CMD_VERIFY (0x0064)

[Function]

Compare the template of RamBuffer with specified Template in database (1:1) and return result

[Sequence]

1. If the Template ID is invalid, the Module returns ERR_INVALID_TMPL_NP
2. If the RamBuffer ID is invalid, the Module returns ERR_INVALID_BUFFER_ID.
3. If the database is empty, the Module returns ERR_TEMPL_EMPTY.
4. Compare the template of RamBuffer with the specified template in database and return the result. If success, the Module returns ERR_SUCCESS in RET bytes, template ID and smart update result in DATA bytes. If fail, returns ERR_VERIFY in RET bytes.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0064	
LEN	4	
DATA	2 bytes	Template ID
	2 bytes	RamBuffer ID
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA
SID	Source Device ID
DID	Destination Device ID
CMD	0x0064

LEN	Success: 5; Fail: 2	
RET	Result Code	
DATA	Success: Template ID (2 bytes) + Smart update result (1 byte)	
CKS	2 bytes	Checksum

[Example] Compare the template of RamBuffer0 with template (ID=8) in database (1:1)

Host Command: 55 AA 00 00 64 00 04 00 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6F 01

Module Response : AA 55 01 00 64 00 05 00 00 00 08 00 01 00 00 00 00 00 00 00 00 00 00 72 01

4.22. CMD_SET_MODULE_SN (0x0008)

[Function]

Get the Module SN from host and save it to the Module, the Length of SN is 16 bytes.

[Sequence]

1. Host send the Command Packet, let the Module enter waiting data, the Length of the Command Data Packet is sent in the DATA bytes
2. Check if the Command Packet is correct. Wrong: return Error code and finished. If the Length is wrong, returns ERR_INVALID_PARAM. Correct: Module send Response Packet to Host and enter waiting status.
3. Host send the Command Data Packet after receiving the Response Packet.
4. Module save the SN and returns result.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0008	
LEN	2	
DATA	16 (Length of the SN)	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA
SID	Source Device ID
DID	Destination Device ID
CMD	0x0008
LEN	2
RET	Result Code
DATA	None

CKS	2 bytes	Checksum

[Command Data Packet]

PREFIX	0xA55A	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0008	
LEN	16 (Length of the SN)	
DATA	SN (16 bytes)	
CKS	2 bytes	Checksum

[Response Data Packet]

PREFIX	0x5AA5	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0008	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Example] Set the Module SN as Waveshare

Host Command: 55 AA 00 00 08 00 02 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 19 01

Module Response : AA 55 01 00 08 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A 01

Command Data: 5A A5 00 00 08 00 10 00 77 61 76 65 73 68 61 72 65 00 00 00 00 00 00 00 DD 04

Response Data: A5 5A 01 00 08 00 02 00 00 00 0A 01

4.23. CMD_GET_MODULE_SN (0x0009)

[Function]

Read the SN of Module

[Sequence]

1. Send the Length of SN code to Host by Command/Response Packet
2. Send the SN code by Response Data Packet

[Command]

PREFIX	0xAA55
SID	Source Device ID
DID	Destination Device ID

CMD	0x0009	
LEN	0	
DATA	None	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0009	
LEN	4	
RET	Result Code	
DATA	Success: Length of the SN (16 bytes) Fail: Error Code	
CKS	2 bytes	Checksum

Success: [Response Data Packet]

PREFIX	0x5AA5	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0009	
LEN	2 + SN Code Length (16)	
RET	ERR_SUCCESS	
DATA	SN code (16 bytes)	
CKS	2 bytes	Checksum

[Example] Read the SN code as Waveshare

Host Command: 55 AA 00 00 09 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 01

Module Response : AA 55 01 00 09 00 04 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 1D 01

Response Data: A5 5A 01 00 09 00 12 00 00 00 77 61 76 65 73 68 61 72 65 00 00 00 00 00 00 00 E1 04

4.24. CMD_GET_ENROLLED_ID_LIST (0x0049)

[Function]

Send the ID list in database to Host

The structure of ID list is as below:

Every bit of every byte means the status of Template x (x = byte >=0) * 8 + y (y = bit >=0). If it is 0, it means that template is not enrolled, if it is 1, it means that the template was enrolled.

For example: If the second byte of the ID list is 01000001 (BIN), then it means that (Note that we count it from right to left):

bit 0 (1): 8+2+0 = 16 (ID 16 has fingerprint enrolled)

bit 1(0): 8*2+1 = 17 (ID 17 is a free ID without fingerprint)

…

bit 6 (1): 8*2+6 = 22 (ID 22 has fingerprint enrolled)

bit 7 (0): 8*2+7 = 23 (ID 23 is a free ID without fingerprint)

[Sequence]

1. Host send the request by Command Packet
2. The Module response and send the Length of ID list by Response Packet
3. The Module send the ID list by Response Data Packet

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0049	
LEN	0	
DATA	None	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x0049	
LEN	4	
RET	Result Code	
DATA	Success: Length of the SN (400 bytes) Fail: Error Code	
CKS	2 bytes	Checksum

Success: [Response Data Packet]

PREFIX	0x5AA5
SID	Source Device ID
DID	Destination Device ID
CMD	0x0049
LEN	402

RET	ERR_SUCCESS	
DATA	ID List Information	
CKS	2 bytes	Checksum

[Example] Get the ID list in range 1~3000

Host Command: 55 AA 00 00 49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 48 01

Module Response : AA 55 01 00 49 00 04 00 00 00 90 01 00 00 00 00 00 00 00 00 00 00 00 00 DE 01

Response Data: A5 5A 01 00 49 00 92 01 00 00 0E 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 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 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 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 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 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 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 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 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 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 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
00 00 00 00 00 00 00 00 00 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 EB 01

In the ID list:

The byte 0 is 0x0E = 0000 1110, it means that:

Bit 0 = 0: ID = 0 is free (0 is not used always)

Bit 1 = 1: ID = 1 was used

Bit 2 = 1: ID = 2 was used

…

Bit 7 = 0: ID = 7 is free

……

The byte 375 is 0x01 = 0000 0001 means that the Bit 0, ID = 30000 was used.

4.25. CMD_ENTER_STANDBY_STATE (0x000C)

[Function]

Set the Module enter sleep mode

[Sequence]

The Module response ERR_SUCCESS and enter the sleep mode after receiving the Command Packet.

[Command]

PREFIX	0xAA55	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x000C	
LEN	0	
DATA	None	
CKS	2 bytes	Checksum

[Response]

PREFIX	0x55AA	
SID	Source Device ID	
DID	Destination Device ID	
CMD	0x000C	
LEN	2	
RET	Result Code	
DATA	None	
CKS	2 bytes	Checksum

[Example] Set the Module enter the Sleep mode

Host Command: 55 AA 00 00 0C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0B 01

Module Response: AA 55 01 00 0C 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0E 01

Note:

We recommend you send the Sleep command before cutting off the power of Module.

5. Response And Error Code

No.	Response/Error Code	Value	Description
1	ERR_SUCCESS	0x00	Command is handled successfully
2	ERR_FAIL	0x01	Command is failed to be handled
3	ERR_VERIFY	0x10	Template is verified fail (1:1)
4	ERR_INDENTIFY	0x11	The template doesn't exist (1: N)
5	ERR_TMPL_TEMPTY	0x12	The ID is free without fingerprint
6	ERR_TMPL_NOT_EMPTY	0x13	The ID was used
7	ERR_ALL_TMPL_EMPTY	0x14	There are no templates in database
8	ERR_EMPTY_ID_NOEXIST	0x15	There is not free ID
9	ERR_BROKEN_ID_NOEXIST	0x16	There is not broken template
10	ERR_INVALID_TMPL_DATA	0x17	The template data is invalid
11	ERR_DUPLICATION_ID	0x18	The fingerprint has existed
12	ERR_BAD_QUALITY	0x19	The quality of image is bad
13	ERR_MERGE_FAIL	0x1A	Fail to merge templates
14	ERR_INVALID_TMPL_NO	0x1D	The template ID is invalid
15	ERR_INVALID_PARAM	0x22	The parameter is invalid
16	ERR_GEN_COUNT	0x25	The merge count is invalid
17	ERR_INVALID_BUFFER_ID	0x26	The Buffer ID is invalid
18	ERR_FP_NOT_DETECTED	0x28	No finger is detected
19	ERR_FP_CANCLE	0x41	The command is cancelled