

MRB200 FINGERPRINT MODULE FOR G6103 DEVELOPER'S MANUAL

V1.0.0.2

Editor by Zack

2004-10-17

Chapter 1 Introduction	2
Chapter 2 How to Develop It	3
Chapter 3 Commands for Fingerprint Module	3
1 Basic Commands	5
1.1 CMD_GET_USER_SUM_DB	5
1.2 CMD_GET_USER_RIGHT_DB	5
1.3 CMD_VERIFY_DB	5
1.4 CMD_REG_START_DB	6
1.5 CMD_REG_SECOND_DB	7
1.6 CMD_REG_END_DB	7
1.7 CMD_REG_DELETE_DB	8
1.8 CMD_REG_ALLDEL_DB	8
1.9 CMD_GET_USER_NUMBER_DB	9
1.10 CMD_GET_VALUE	9
1.11 CMD_IDENTIFY_DB	10
1.12 CMD_FROM_VALUE_DB	11
1.13 CMD_TO_VALUE_DB	11
1.14 CMD_FROM_VERIFY_DB	12
1.15 CMD_FROM_VERIFY	12
1.16 CMD_FROM_IDENTIFY_DB	13
1.17 CMD_GET_IMAGE	13
1.18 CMD_SET_BAUD	14
1.19 CMD_PROCESS_IMAGE	15
1.20 CMD_TEST_COMM	15
1.21 CMD_TEST_FINGER	16
1.22 CMD_SERIAL_PROG_UPGRADE	16
1.23 Note	17
2 Internal Commands	18
2.1 CMD_SET_REG	18
2.2 CMD_GET_VERSION	18
2.3 CMD_IDLE	18
2.4 CMD_EXIT_IDLE	18
2.5 CMD_PROG_UPGRADE	18
2.6 CMD_FROM_IMAGE	20
2.7 CMD_GET_LAST_ERROR	20
2.8 CMD_GET_FLG	20
Affix 1 The corresponding definition	21
1 Fingerprint Module command (CMD) Definitions	21
2 Fingerprint Module Answer Code (ACK) Definitions	21
3 Basic Answer Information Definitions	22
4 User Information Definitions	22
5 Transport Speed – Baud Rate Definitions	22
Affix 2 Example	23
Affix 3 Commands List	35

Chapter 1 Introduction

Fingerprint module is the latest module for G6102. Adopting encryption techniques and safety measure for IC card, this system encrypts the data to ensure the security of data and the validity of card, which assures the validity of cardholder and supports vary touch and no touch card such as memory card, and CPU card, including contact and contactless card.

Products application

This system can be applied to the stored information on IC card and contactless card such as fingerprint temporary residence permit, fingerprint ID card, fingerprint admission, traffic duty, social security, driver school, fingerprint finance card, electronic wallet and some other system which need to authenticate the card offline. Together with the perfect technical support and product upgrading service, we can ensure the benefit for our development partner and system integrator.

Features

- It is a portable device can be operated in an easy way, which results from its integrative structure design. So it is fit for the mobile case especially.
- It is compatible with multi-protocol for smart cards. For example, it can operate the card compatible with ISO7816, ISO14443-A/B or ISO15693.
- Improve the safety by SAM card key management method, and it expands the scope of the device, makes the update easier and safer.
- Open developing platform supports redevelopment for different application.
- Footprint identification function is realized by the latest independent footprint identification module of our corporation, which has the advanced dermal detecting function.
- Low power consumption but long halt time results from its power saving design. As a portable terminal device, it is an important feature.
- A Li-ion battery is used, which can be recharged.
- There is a large storage for identification system, and it can be extended according to user's need.

Technical Parameters

Footprint Compare Time	<0.01second
Compare Method	1:1 , 1:N
Storage Capacity	More than 800
Error footprints pass rate	<0.01%~0.001%
Right footprints are rejected rate	<0.1%~0.01%

Chapter 2 How to Develop It

Before you are going to develop the application for G6102 with fingerprint module, you should master the basic way for G6102's development. Fingerprint module is a daughter board for G6102, as a appendant device. It communicates with G6102 by a serial port. No library functions for you to operate fingerprint module, and you just send some commands to operate it, which will give you a introduction in the next chapter. While in the last chapter, we will give you some useful functions to help you to know how to use the fingerprint module smoothly.

At the following chapters, we just give you an introduction about fingerprint module. If you want to know how to use the basic G6102 and contactless module's library functions, please refer to other documents. Further questions please contact with our technical support department.

Chapter 3 Commands for Fingerprint Module

MRB200 fingerprint module is the daughter board for G6102 in fact. It communicates with G6102's main board by asynchronous serial port. Main board sends all kinds of commands (CMD) to fingerprint module to operate it, and fingerprint module will apply the command and answer to main board the result of the operation (ACK).

The parameters for the serial port communication between them show as following:

- 19200bps (Default);
- No checksum;
- One starting bit;
- One end bit;

A commands (CMD) consists of 8 bytes or more bytes. So there are two types of command format. The one is a group of basic commands consist of 8 bytes, the other is a group of commands consist of more than 8 bytes and the first byte would be 0x3X.

Basic commands format:

The first byte is the head byte, and it must be 0xFE.

The second byte is the device number, and it would be 0x00 normally.

The third byte is the command code.

The fourth and fifth bytes are parameter code: P1, P2.

The sixth byte is assistant parameter code: P3.

The seventh byte is check sum, which used to save the ^ (Bit operation symbol) value from the second byte to the sixth byte.

The eighth byte is the end byte, and it must be 0xFD.

Answer (ACK) consists of 8 bytes or more bytes normally, and the format is described as following:

The first byte is the head byte, and it must be 0xFE.

The second byte is the device number, and it would be 0x00 normally.

The third byte is the answer code.

The fourth and fifth bytes are parameter code: P1, P2.

The sixth byte is the answer parameter code AP.

The seventh byte is check sum, which used to save the ^ value from the second byte to the sixth byte.

The eighth byte is the end byte, and it must be 0xFD.

3.1 User Power

The database of the MRB200 fingerprint module supports three level user power. That is, administrator, common user and temporary user. It will give you a convenience way to manage the users. The developer can give the different power to different user according to his requests.

For example, if MRB200 module is used to a fingerprint identify lock, it can give different power to users. Common users can open the door, unlock and so on. The administrator can modify, search and delete information from the fingerprint library except the common operation.

MRB200 fingerprint module will set a default administrator for the first user if the fingerprint library is blank.

1 Basic Commands

1.1 CMD_GET_USER_SUM_DB

- Function

Query the total fingerprint's number in the module.

- CMD Parameters

`CODE = 0x05、 P1 = 0x00、 P2 = 0x00、 P3 = 0x00`

- ACK Parameters

`HEAD + CH + 0x45 + SUM_H + SUM_L + AP + CHK + END`

SUM = The total number of registered fingerprint. The number will be saved as a hex format and use two bytes. The fourth byte will be high byte and the fifth byte will be low byte.

AP must equal EW_SUCCESS, which is defined by a head file.

1.2 CMD_GET_USER_RIGHT_DB

- Function

Send user ID to confirm the user's power.

- CMD Parameters

`CODE = 0x00、 P1 = UserID_H、 P2 = UserID_L、 P3 = 0x00`

NOTE: [P1,P2] is the user ID number, P1 is high byte, P2 is low byte. For example, user ID is 0x1234, so P1 = 0x12, P2 = 0x34.

- ACK Parameters

`HEAD + CH + 0x40 + 0x00 + 0x00 + AP + CHK + END`

AP = User power. If:

- AP=EW_NO_USER User is not occurred in the library.
- AP=EW_GUEST_USER User is a temporary user.
- AP=EW_NORMAL_USER User is a common user.
- AP=EW_MASTER_USER User is the administrator.

1.3 CMD_VERIFY_DB

- Function

Get the user's fingerprint and compare it with the fingerprint selected by command in the fingerprint library. It will confirm if it is the same fingerprint. User must put the finger on the fingerprint collection in five seconds after he send the command, or the module will return the timeout error.

- CMD Parameters

`CODE = 0x01、 P1 = UserID_H、 P2 = UserID_L、 P3 = 0x00`

- ACK Parameters

`HEAD + CH + 0x41 + 0x00 + 0x00 + AP + CHK + END`

AP = Return Value. If:

- AP=EW_TIME_OUT Timeout error.
- AP=EW_SUCCESS Compare successfully.
- AP=EW_FAIL Unknown error.
- EW_FAIL_BMF Collect fingerprint failed.
- EW_FAIL_FEA Get eigenvalue failed.
- EW_FAIL_MATCH Compare failed.
- EW_NO_USER User is not occurred.

1.4 CMD_REG_START_DB

- Function

MRB200 require user input fingerprint for three times to ensure the exact collection in the fingerprint register process. This command will finish the first collection.

- CMD Parameters

`CODE = 0x02、 P1 = UserID_H、 P2 = UserID_L、 P3 = User Power`

NOTE: UserID != 0

- ACK Parameters

`HEAD + CH + 0x42 + 0x00 + 0x00 + AP + CHK + END`

AP = Return information. If:

- AP=EW_TIME_OUT Collection timeout error. User must put his finger on the fingerprint collection in five seconds after command is send, or it will return timeout error.
- AP=EW_SUCCESS Success. It should sent CMD_REG_SECOND_DB right now for the second collection.
- EW_FAIL Unknown error.
- EW_FAIL_ID The user ID should not be 0. This error probably is made by UserID = 0.
- EW_FAIL_FEA The module can't get eigenvalue from the collection fingerprint. The possible reason is the collection is not a fingerprint or the fingerprint is collected is bad.
- EW_FAIL_BMF Collection failed.
- EW_FULL The capacity of the fingerprint's library is full.

1.5 CMD_REG_SECOND_DB

- Function

The second fingerprint collection in the fingerprint register process. It should be send after the command CMD_REG_START_DB. A timeout error will be send if user couldn't put his finger

on the fingerprint collection in five seconds after the command is send.

If the last command is not CMD_REG_START_DB, it will return register failed right now.

If the user ID or power is not same as the corresponding CMD_REG_START_DB command's parameters, it will use the parameters in the command CMD_REG_START_DB.

- CMD Parameters

```
CODE = 0x04, P1 = UserID_H, P2 = UserID_L, P3 = User power
```

- ACK Parameters

```
HEAD + CH + 0x44 + 0x00 + 0x00 + AP + CHK + END
```

AP = Return value. If :

- EW_TIME_OUT Timeout, while it can send this command again.
- EW_SUCCESS Success. It should send command CMD_REG_END_DB right now.
- EW_FAIL Register failed. And if you want to collect a fingerprint again. You have to send command CMD_REG_START_DB again.
- EW_FAIL_REG Last command is not CMD_REG_START_DB
- EW_FAIL_FEA Get eigenvalue failed
- CMD_FAIL_MATCH Fingerprint is not match between this one and last one.

1.6 CMD_REG_END_DB

- Function

The third fingerprint collection in the fingerprint register process. It should be send after the command CMD_REG_SECOND_DB. A timeout error will be send if user couldn't put his finger on the fingerprint collection in five seconds after the command is send. Module will compare the fingerprint collect this time and the ones of last two times. If they are the same, it will save it into the fingerprint library, and then return a successful sign.

If the last command is not CMD_REG_SECOND_DB, it will return register failed right now.

If the user ID or power is not same as the corresponding CMD_REG_START_DB command's parameters, it will use the parameters in the command CMD_REG_START_DB

- CMD Parameters

`CODE = 0x03、 P1 = UserID_H、 P2 = UserID_L、 P3 = User Power`

- ACK Parameters

`HEAD + CH + 0x43 + 0x00 + 0x00 + AP + CHK + END`

AP = Return value. If:

- EW_TIME_OUT Timeout, while it can send this command again.
- EW_SUCCESS Success.
- EW_FAIL Register failed. While it can send this command again
- EW_FAIL_REG Last command is not CMD_REG_SECOND_DB
- EW_FAIL_FEA Get eigenvalue failed
- CMD_FAIL_MATCH Fingerprint is not match between this one and last one.
- CMD_FAIL_FLASH Save fingerprint information failed.

1.7 CMD_REG_DELETE_DB

- Function

Delete the user specified by user number and return the result.

[NOTE] Developers should think about the problem of user's power, because this command will change the fingerprint database. Developers should check the user's power before this command is send in their program.

- CMD Parameters

`CODE = 0x20、 P1 = UserID_H、 P2 = UserID_L、 P3 = 0x00`

- ACK Parameters

`HEAD + CH + 0x60 + 0x00 + 0x00 + AP + CHK + END`

AP = Return Value. Always return:

- EW_SUCCESS Delete successfully. The return value just ensure the user is not existed when ACK is returned, while it's not sure if the user has registered.

1.8 CMD_REG_ALLDEL_DB

- Function

Delete the all users that their power is P3, then return the result of the value.

[NOTE] Developers should think about the problem of user's power, because this command will change the fingerprint database. Developers should check the user's power before this command is send in their program.

- CMD Parameters

`CODE = 0x21、 P1 = 0x00、 P2 = 0x00、 P3 = User Power`

- P3 = EW_ALL_USER All user;
 - P3 = EW_GUEST_USER Temporary user;
 - P3 = EW_NORMAL_USER Common user;
 - P3 = EW_MASTER_USER Administrator;
- ACK Parameters

HEAD + CH + 0x61 + 0x00 + 0x00 + AP + CHK + END

 AP = Return value. It always returns:
 - AP = EW_SUCCESS Delete successfully. The return value just ensure the user is not existed when ACK is returned, while it's not sure if the user has registered

1.9 CMD_GET_USER_NUMBER_DB

- Function

Get P3(It is a number parameter) users' number and their power level from the user pointed by P1, P2 in the fingerprint library.

- CMD Parameters

CODE = 0x10, [P1, P2] = Get from which user, P3 = Want to get how many users

For example: Return five users' number and their power level from No. 10 user. It should be:

P1=0x00, P2=0x10. P3=0x05;

NOTE: P3 should not more than 30. If P3 is more than 30, module will let it equal 30 forcedly.

- ACK Parameters

HEAD + CH + 0x50 + 0x00 + Length + AP + CHK + END

Length: The total length of the data including users' number and their power level.

AP = EW_SUCCESS It is a fixed value

And then, module will send the data including users' number and their power level and other three bytes code, whose format is listed as following:

HEAD +Data (Length bytes) + CHK + END

Three bytes store one user's number and his level. For example, the second, third and fourth bytes store the first user's information. The fifth, sixth and seventh bytes store the second user's information, and so on until the No. P3 user.

1.10 CMD_GET_VALUE

- Function

Collect a fingerprint and pick up the fingerprint eigenvalue. User must put his finger onto the collecting unit in five seconds after this command is sent, or the module will return a

TIMEOUT error. After module finishes picking up the fingerprint eigenvalue, it will return ACK, and then return the fingerprint eigenvalue.

- CMD Parameters

`CODE = 0x26, P1 = P2 = P3 = 0x00`

- ACK Format

`HEAD + CH + 0x66 + Length_H + Length_L + AP + CHK + END`

Length = [Length_H, Length_L] is the length of eigenvalue.

AP = Status. If:

- AP = EW_TIME_OUT Timeout error.
- AP = EW_SUCCESS Success.
- AP = EW_FAIL Failed.

(1) EW_FAIL_BMF Collect failed (2) EW_FAIL_FEA Fail to pick up the eigenvalue

(3) EW_FAIL_MATCH, Compare failed.

If it is successful, the module will send the fingerprint eigenvalue, whose total length Length + 3.

`HEAD + Fingerprint eigenvalue data(Length bytes) + CHK + END`

1.11 CMD_IDENTIFY_DB

- Function

Collect a fingerprint, and compare with the all of the fingerprint in the fingerprint library, then return the result and user's number. User must put his finger onto the collecting unit in five seconds after this command is sent, or the module will return a TIMEOUT error.

- CMD Parameters

`CODE = 0x12, P1 = P2 = P3 = 0x00`

- ACK Format

`HEAD + CH + 0x52 + P1 + P2 + AP + CHK + END`

The fourth and fifth bytes [P1, P2] = User number. For example, the user's number is 0x1234, so P1=0x12, P2=0x34.

The sixth byte AP = User power level. If:

- AP = EW_NO_USER The user is not occurred
- AP = EW_TIME_OUT Timeout error
- AP = EW_GUEST_USER Temporary user
- AP = EW_NORMAL_USER Common user
- AP = EW_MASTER_USER Administrator

1.12 CMD_FROM_VALUE_DB

- Function

Send a fingerprint eigenvalue and user's number to the module. The module will save the eigenvalue to the fingerprint library according to the user's number and return the processing result and user's number.

When sending the command to the module, it blanks the buffer and waits for receiving the fingerprint eigenvalue. Then the eigenvalue is send, the module saves it according to the user's number pointed by command. At last, the module will return the ACK.

- CMD Paramters

$\boxed{\text{CODE} = 0x31, [P1, P2] = (\text{The length of the fingerprint eigenvalue}+3) \text{ (P1 HSB, P2 LSB)}, P3 = 00}$

Then send the fingerprint eigenvalue:

$\boxed{\text{HEAD} + \text{UserID_H} + \text{UserID_L} + \text{User power} + \text{Fingerprint eigenvalue} + \text{CHK} + \text{END}}$

- ACK Format

$\boxed{\text{HEAD} + \text{CH} + 0x71 + P1 + P2 + \text{AP} + \text{CHK} + \text{END}}$

The fourth, fifth bytes $[P1, P2] = \text{User number}$. For example, the user number is 0x1234, then $P1=0x12H, P2=0x34H$.

The sixth byte $\text{AP} = \text{Processing result}$. If:

- $\text{AP}=\text{EW_SUCCESS}$ Success;
- $\text{AP}=\text{EW_FULL}$ The user's capacity is full;
- $\text{AP}=\text{EW_FAIL}$ Failed

1.13 CMD_TO_VALUE_DB

- Function

Get the fingerprint eigenvalue in the fingerprint library according to the user's number pointed in the command.

- CMD Parameters

$\boxed{\text{CODE} = 0x22, [P1, P2] = \text{用户号}, P3 = 0x00}$

- ACK Format

$\boxed{\text{HEAD} + \text{CH} + 0x62 + P1 + P2 + \text{AP} + \text{CHK} + \text{END}}$

The fourth and fifth bytes is $[P1, P2] = \text{The length of the eigenvalue}$.

The sixth byte $\text{AP} = \text{Result}$. If:

- $\text{AP}=\text{EW_NO_USER}$ The user does not exist;
- $\text{AP}=\text{EW_GUEST_USER}$ Temporary User;
- $\text{AP}=\text{EW_NORMAL_USER}$ Common User;

➤ AP=EW_MASTER_USER Administrator;

If the user exists, then it sends the eigenvalue:

`HEAD + Eigenvalue + CHK + END`

1.14 CMD_FROM_VERIFY_DB

- Function

Send a fingerprint eigenvalue and user's number to the module. The module will compare the eigenvalue with the one in the library according to the user's number, then return the processing result.

When sending the command to the module, it blanks the buffer and waits for receiving the fingerprint eigenvalue. Then the eigenvalue is send, the module saves it according to the user's number pointed by command. At last, the module will return the ACK.

- CMD Paramters

`CODE = 0x31、 [P1, P2] = (The length of the fingerprint eigenvalue+3) (P1 HSB, P2 LSB)、 P3 = 00`

Then send the fingerprint eigenvalue:

`HEAD + UserID_H + UserID_L + User power + Fingerprint eigenvalue + CHK + END`

- ACK Format

`HEAD + CH + 0x73 + 0x00 + 0x00 + AP + CHK + END`

The fourth byte AP = Processing result. If:

- AP=EW_SUCCESS Success;
- AP=EW_NO_USER The user doesn't exist;
- AP=EW_FAIL Failed

1.15 CMD_FROM_VERIFY

- Function

Send a fingerprint eigenvalue to the module. The module will send a command to fingerprint collector to collect a fingerprint, and then compare the eigenvalue with the one collected just now,. At last return the processing result.

When sending the command to the module, it blanks the buffer and waits for receiving the fingerprint eigenvalue. After the eigenvalue is send, the module will send a command to fingerprint collector, and collect the fingerprint eigenvalue in five seconds. Then compare them and return the ACK.

- CMD Parameters

`CODE = 0x31、 [P1, P2] = (The length of the fingerprint eigenvalue+3) (P1 HSB, P2 LSB)、 P3 = 00`

Then send the fingerprint eigenvalue:

```
HEAD + 0x00 + 0x00 + 0x00 + Fingerprint eigenvalue + CHK + END
```

- ACK Format

```
HEAD + CH + 0x75 + 0x00 + 0x00 + AP + CHK + END
```

The fourth byte AP = Processing result. If:

- AP=EW_SUCCESS Success;
- AP=EW_TIMEOUT Timeout error;
- AP=EW_FAIL Failed

1.16 CMD_FROM_IDENTIFY_DB

- Function

Send a fingerprint eigenvalue to fingerprint module, and it will compare the eigenvalue with all of the eigenvalue in the library. At last, return the processing result and the user's number.

Send the command and the module will blank the buffer. Then send the eigenvalue. The module will compare it with all of the eigenvalue in the library and then return the ACK.

- CMD Parameters

```
CODE = 0x34, [P1,P2] = (The length of fingerprint eigenvalue+3) (P1 HSB, P2 LSB),
```

```
P3 = 0x00
```

Then send the fingerprint eigenvalue:

```
HEAD + 0x00 + 0x00 + 0x00 + ([P1,P2]-3) Fingerprint eigenvalue + CHK + END
```

- ACK Format

```
HEAD + CH + 0x74 + P1 + P2 + AP + CHK + END
```

The fourth, fifth bytes [P1, P2] = User number. For example, the user number is 0x1234, then P1=0x12H, P2=0x34H.

The sixth byte AP = Processing result. If:

- AP=EW_NO_USER The user does not exist, and [P1, P2] is meaningless;
- AP=EW_GUEST_USER Temporary User;
- AP=EW_NORMAL_USER Common User;
- AP=EW_MASTER_USER Administrator;

1.17 CMD_GET_IMAGE

- Function

Collect a piece of fingerprint image by the module, and get the data of the image outline.

- CMD Parameters

`CODE = 0x27、 P1 = P2 = P3 = 0x00`

- ACK Format

`HEAD + CH + 0x67 + Length_H + Length_L + AP + CHK + END`

Length = Data Length, Length_H HSB, Length_L LSB

AP = Return Value. If:

- AP=EW_TIME_OUT Timeout error.
- EW_TIME_OUT_BMF Collect failed
- AP=EW_SUCCESS Success
- AP=EW_FAIL Failed

If it is successful, the module will return the fingerprint image outline right now. The format is following:

`HEAD + Length (data) + END`

Image outline's properties: 256 Gray scale (8 bits), Width: 256 pixel, High: 256 pixel

Data format: Send the image data line by line. Every byte is represent two pixels. HSB is the former pixel's HSB and LSB is the later pixel's HSB. That is, the precision of the image is 4 bits. The LSB of pixel needs to be filled by user.

1.18 CMD_SET_BAUD

- Function

Set the baud rate for communication. The fingerprint module's baud rate is 19200bps after it is powered on. If the user want to change its baud rate, this command should be sent in the baud rate as 19200bps. After a success sign is returned, it represents that the module has set the baud rate successfully. The user can communicate with the module by new baud rate until the next time the module is powered on.

- CMD Parameters

`CODE = 0x0F、 P1 = P2 = 0x00、 P3 = New baud rate`

- P3 = EW_BAUD_9600, Baud rate is 9600bps
- P3 = EW_BAUD_19200, Baud rate is 19200bps (Default value)
- P3 = EW_BAUD_38400, Baud rate is 38400bps
- P3 = EW_BAUD_57600, Baud rate is 57600bps
- P3 = EW_BAUD_115200, Baud rate is 115200bps(Not command to use this value.

Too fast speed would make the module instable.

- ACK Format

`HEAD + CH + 0x4F + 0x00 + 0x00 + AP + CHK + END`

The sixth byte : AP = Old baud rate.

- P3 = EW_BAUD_9600, Baud rate is 9600bps
- P3 = EW_BAUD_19200, Baud rate is 19200bps (Default value)
- P3 = EW_BAUD_38400, Baud rate is 38400bps
- P3 = EW_BAUD_57600, Baud rate is 57600bps
- P3 = EW_BAUD_115200, Baud rate is 115200bps(Not command to use this value.
Too fast speed would make the module instable.

1.19 CMD_PROCESS_IMAGE

- Function

Get the eigenvalue data of the fingerprint image saved in the module's image buffer.

[NOTE] The fingerprint image is collected last time, and the module will make mistakes if the image data is picked up or compared, because these operation will change the content of the buffer.

- CMD Parameters

`CODE = 0x2D、 P1 = P2 = P3 = 0x00`

- ACK Format

`HEAD + CH + 0x6D + Length_H + Length_L + AP + CHK + END`

Length = [Length_H, Length_L] The length of the eigenvalue

AP = Return value. If:

- AP = EW_SUCCESS Success
- AP = EW_FAIL failed
- EW_FAIL_FEA Failed to pick up the eigenvalue

After then, the fingerprint eigenvalue is sent, the format is following:

`HEAD + Length(fingerprint eigenvalue data) + CHK + END`

1.20 CMD_TEST_COMM

- Function

Test if the communication works well.

- CMD Parameters

`CODE = 0x2C、 P1 = P2 = P3 = 0x00`

- ACK Format

`HEAD + CH + 0x6C + 0x00 + 0x00 + AP + CHK + END`

AP = Return Value. If:

- AP = EW_SUCCESS Communication is OK.

- AP = Others Communication error.

Communication error will result from: (1) No response for a long time. (2) The return value of the module is not compatible with the communication protocol, which includes the number of the data and the definition of the data.

Reasons for these error result from possibility: (1) The baud rate is different from the control baud rate. (2) The module can't work well.

1.21 CMD_TEST_FINGER

- Function

Confirm it the finger is stay on the sensor.

- CMD Parameters

CODE = 0x06、 P1 = P2 = P3 = 0x00

- ACK Format

HEAD + CH + 0x46 + 0x00 + 0x00 + AP + CHK + END

AP = Return value. If:

- AP = EW_SUCCESS The finger is on the sensor.
- AP = EW_TIME_OUT The finger is not on the sensor
- EW_TIME_OUT_BMF Collect failed

1.22 CMD_SERIAL_PROG_UPGRADE

- Function

Send executable file and the its save address to the module, which is not more than 264 bytes. It is convenience for program update in the serial flash. (20bit = 11-bit page address (0x800 pages per chip) + 9-bit byte address (264 bytes per page))

After sending command to fingerprint module, continue to send data for a moment. If the module receives the command, it will blank the buffer, and ready to receive the data. When it finishes receiving the data, it will save them according to the command's parameters and returns ACK.

The executive address (byte) is 0x27FFE ~ 0x27FFF and Flash address is (word) 0x2007FF. It is the bit-OR'ing check value. If it is not correct, the program can't run smoothly. The only function of it is to update program.

- CMD Parameters

CODE = 0x3B、 [P1, P2] = (Program data length+3) (P1 HSB, P2 LSB)、 P3 =PageAddr

[P1,P2]=The data length will be send (byte) $\leq (264+3)$. If it is more than 264+3, it can't not operate Serial Flash, and return UPGRADE_LEN_ERROR

if $[P1,P2] \leq 3$, it can be erased according to page. (The format is just like the method specified in this document.

For example: If $[P1,P2]=1$, so it can send command HEAD + DEVICE + CODE + P1 + P2 + P3 + CHK + END at first, and then send HEAD+DEVICE+Any Value+CHK+END.

P3=The 11bit page address in the Serial Flash. address, and the const list is in the page 125 ~ 185. So only 8 bits is enough

	page	Flash IP adress(byte)	(byte)
Program data	0	0x00000~0x00108	0x020000~0x020108

	124	0x0F800~0x0F908	
Const It msut let the program 'const_move' to finish the copy task located in 0x20100.	125	0x0FA00	Limited by const_move
		
	185	0x17200	

Then send data:

HEAD + DEVICE + ByteAddr_H + ByteAddr_L + Data + CHK + END

ByteAddr_H is the bit 8 of the beginning data byte address, ByteAddr_L is the bit7~bit0 of the beginning data byte address. Or it doesn't process the flash operation and return EW_FULL.

● ACK Format

HEAD + CH + 0x7A + 0x00 + 0x00 + AP + CHK + END

The sixth byte AP = Processing result. If:

- AP=EW_SUCCESS Success
- AP=EW_FAIL Failed
- AP=EW_FULL The number of the data check sum or offset can't content the requests.

1.23 Note

- All of the fingerprint eigenvalue send to or got from the fingerprint module are encrypted. It need to the special API and develop kit support by our company.
- For simplify, the module will not judge if the user is occurred when it adds the new user. It should be judged by another way.

2 Internal Commands

2.1 CMD_SET_REG

- Function
Set BCT100 control register. Be unavailable in this version.

2.2 CMD_GET_VERSION

- Function
Check the version information of the module. Be unavailable in this version.

2.3 CMD_IDLE

- Function
Let the module run in the low power resuming status (5mA & 5V). After the module entering the status, all of the commands will process the same operation defined as following but CMD_EXIT_IDLE, and it will get the same return value.

- CMD Parameters

`CODE = 0x07, P1 = P2 = P3 = 0x00`

- ACK Format

`HEAD + CH + 0x47 + 0x00 + 0x00 + AP + CHK + END`

- AP = EW_SUCCESS Success. The module will return ACK before it entering the status, so it always returns success.

2.4 CMD_EXIT_IDLE

- Function
Let the module quit the low power resuming status.

- CMD Parameters

`CODE = 0x08, P1 = P2 = P3 = 0x00`

- ACK Format

`HEAD + CH + 0x48 + 0x00 + 0x00 + AP + CHK + END`

- AP = EW_SUCCESS Success. The module will return ACK before it entering the status, so it always returns success.

2.5 CMD_PROG_UPGRADE

- Function
Send executable file and the address where to save it to the module, and update the

program in the flash. The program should be less than 2K words.

After sending command to fingerprint module, continue to send data in a moment. If the module receives the command, it will blank the buffer, and ready to receive the data. When it finishes receiving the data, it will save them according to the command's parameters and returns ACK.

The executive address (byte) is 0x27FFE ~ 0x27FFF and Flash address is (word) 0x2007FF. It is the bit-OR'ing check value. If it is not correct, the program can't run smoothly. The only function of it is to update program.

● CMD Parameters

$\boxed{\text{CODE} = 0x3A, [P1, P2] = (\text{Program data length}+3) \text{ (P1 HSB, P2 LSB)}, P3 = \text{Save sector 's number}}$

$[P1,P2]=$ The data length will be send (byte) $\leq (0x1000+3)$. If it is more than 0x1000+3, it can't not operate Serial Flash, and return EW_FULL

if $[P1,P2] \leq 2$, it only erases sector. (It still sends the data according to the format).

For example: If $[P1,P2]=1$, so it can send command HEAD + DEVICE + CODE + P1 + P2 + P3 + CHK + END at first, and then send HEAD + DEVICE + Any Value + CHK + END.

$P3=$ The internal sector number in the flash. $0 \leq P3 \leq 13$, or it returns EW_FAIL. And:

	page	Flash address(byte)	Executive address (byte)
Program data	0	0x200000~0x2007FF	0x020000~0x02FFF

	7	0x203800~0x203FFF	0x027000~0x027FFF
Const data It must let the program 'const_move' to finish copying task located in 0x20100h.	8	0x205000	Limited by const_move in the program
		
	13	0x207800	

Then send data:

$\boxed{\text{HEAD} + \text{DEVICE} + \text{Offset}_H + \text{Offset}_L + \text{Data} + \text{CHK} + \text{END}}$

Offset is the initial data's offset in the sector (count by word). $0x0000 \leq \text{Offset} \leq 0x800 - (\text{ceil}([P1,P2] - 3) / 2)$. Or it doesn't process the flash operation and return EW_FULL.

● ACK Format

$\boxed{\text{HEAD} + \text{CH} + 0x7A + 0x00 + 0x00 + \text{AP} + \text{CHK} + \text{END}}$

The sixth byte AP = Processing result. If:

- AP=EW_SUCCESS Success
- AP=EW_FAIL Failed
- AP=EW_FULL The number of the data or offset can't content the request

2.6 CMD_FROM_IMAGE

- Function
Send the fingerprint image to DSP. Be unavailable in this version.

2.7 CMD_GET_LAST_ERROR

- Function
Get the exact error code of the last one.
- CMD Parameters
`CODE = 0x09、 P1 = P2 = P3 = 0x00`
- ACK Format
`HEAD + CH + 0x49 + P1 + P2 + AP + CHK + END`
The sixth byte AP = Processing Result. If:
 - AP=EW_SUCCESS Success. P1 = HSB of the error code. P2 = LSB of the error code.
 - AP=EW_FAIL Failed.

2.8 CMD_GET_FLG

- Function
Get the value marked by program.
- CMD Parameters
`CODE = 0x0B、 P1 = P2 = P3 = 0x00`
- ACK Format
`HEAD + CH + 0x4B + P1 + P2 + AP + CHK + END`
The sixth byte AP = Processing Result. If:
 - AP=EW_SUCCESS Success. P1 = HSB of the program mark. P2 = LSB of the program mark.
 - AP=EW_FAIL Failed

Affix 1 The corresponding definition

1. Fingerprint Module command (CMD) Definitions

```
#define CMD_GET_USER_SUM_DB          0x05
#define CMD_GET_USER_RIGHT_DB        0x00
#define CMD_VERIFY_DB                0x01
#define CMD_REG_START_DB             0x02
#define CMD_REG_SECOND_DB            0x04
#define CMD_REG_END_DB               0x03
#define CMD_REG_DELETE_DB            0x20
#define CMD_REG_ALLDEL_DB            0x21
#define CMD_GET_USER_NUMBER_DB       0x10
#define CMD_GET_VALUE                 0x26
#define CMD_IDENTIFY_DB              0x12
#define CMD_TO_VALUE_DB              0x22
#define CMD_FROM_VALUE_DB            0x31
#define CMD_FROM_VERIFY_DB           0x33
#define CMD_FROM_IDENTIFY_DB         0x34
#define CMD_FROM_VERIFY              0x35
#define CMD_GET_IMAGE                 0x27
#define CMD_SET_BAUD                  0x0F
#define CMD_GET_VERSION               0x2A
#define CMD_PROCESS_IMAGE             0x2D
#define CMD_TEST_COMM                 0x2C
#define CMD_TEST_FINGER               0x06
```

2. Fingerprint Module Answer Code (ACK) Definitions

```
#define ACK_GET_USER_SUM_DB          0x45
#define ACK_GET_USER_RIGHT_DB        0x40
#define ACK_VERIFY_DB                0x41
#define ACK_REG_START_DB             0x42
#define ACK_REG_SECOND_DB            0x44
#define ACK_REG_END_DB               0x43
#define ACK_REG_DELETE_DB            0x60
#define ACK_REG_ALLDEL_DB            0x61
#define ACK_GET_USER_NUMBER_DB       0x50
#define ACK_GET_VALUE                 0x66
```

```
#define ACK_IDENTIFY_DB          0x52
#define ACK_TO_VALUE_DB         0x62
#define ACK_FROM_VALUE_DB       0x71
#define ACK_FROM_VERIFY_DB      0x73
#define ACK_FROM_IDENTIFY_DB    0x74
#define ACK_FROM_VERIFY         0x75
#define ACK_GET_IMAGE           0x67
#define ACK_SET_BAUD            0x4F
#define ACK_GET_VERSION         0x6A
#define ACK_PROCESS_IMAGE       0x6D
#define ACK_COMM_TEST           0x6C
#define ACK_TEST_FINGER         0x46
```

3. Basic Answer Information Definitions

```
#define EW_SUCCESS              0x00
#define EW_FAIL                 0x01
#define EW_FULL                 0x02
#define EW_ROLLED_USER          0x03
#define EW_NO_USER              0x04
#define EW_TIME_OUT            0x0F
```

4. User Information Definitions

```
#define EW_GUEST_USER           0x01
#define EW_NORMAL_USER          0x02
#define EW_MASTER_USER          0x03
#define EW_ALL_USER             0x04
```

5. Transport Speed – Baud Rate Definitions

```
#define EW_BAUD_9600            0x01
#define EW_BAUD_19200           0x02
#define EW_BAUD_38400           0x03
#define EW_BAUD_57600           0x04
#define EW_BAUD_115200          0x05
```

Affix 2 Example

```
//ShenZhen GrandLand

//Fingerprint application based on EH0318 handpos machine

//Demo source code for Verifying one's fingerprint

//just open Modem Vcc, and then using Uart functions to communication with fingerprint
module

//you must be familiar with the communication protocol between G6102 and fingerprint
module

//the following code is just an example: how to power on the finger module and

//how to send or recieve data beteen POS and EWAYTEK finger module

/*****
*****/

#include <console.h>

#include <mcard.h>

//Please read the document for G6102

/*****
*/

//define constants

#define WR_CARD_SUCCESS 0x33
#define WR_CARD_FAIL    0x32
#define VERI_CARD_SUCCESS 0x31
#define VERI_CARD_FAIL    0x30
#define FIRST_FINGER_START_ADDR 0x072
#define SECOND_FINGER_START_ADDR 0x174

//define constants

#define MAX_TEMP_USED_LENGTH 256
#define USER_INFO_LEN 559

#define DATA_HEAD 0xfe
#define DATA_CH 0x00
#define DATA_END 0xfd

#define FINGER_DATA_LEN 256
```

```
#define DELETE_RECORD_CODE  0x87
#define UPLOAD_RECORD_CODE  0x88
#define UPLOAD_RECORD_ACK   0x89
#define DOWNLOAD_FINGER_CODE 0x98
#define DOWNLOAD_FINGER_ACK  0x99

#define COM_TEST_CODE       0x2C
#define COM_TEST_ACK       0x6C
#define EW_RETURN_SUCCESS   0x00
#define EW_RETURN_FAIL     0x01
#define EW_TIME_OUT        0x0f

#define FINGER_VERIFY_CODE  0x35
#define FINGER_VERIFY_ACK   0x75
#define GET_VALUE_CODE     0x26
#define GET_VALUE_ACK      0x66

#define IC_START_ADDR      0x020

/*****
*****/

//Function: UART send a char
//input: ch( char that will be send to UART)
//output:none
void UARTSendChar(char ch)
{
    typ_UART_stat_word Usw;
    int i;

    UART_send_char(ch);

    do{
        Usw.l_word = UART_stat();
```

```
    } while (Usw.bits.out_busy);

    return ;

}

//

/*****
*****/

//Function: UART send a string array
//input: pBuf(the pointer points to the string array that will be sent to UART)
//output:none
void UARTSendStr(char *pBuf)
{
    typ_UART_stat_word Usw;
    int i;

    for (i=0;pBuf[i];i++)
    {
        UART_send_char(pBuf[i]);
        do{
            Usw.l_word = UART_stat();
        } while (Usw.bits.out_busy);
    }
    return ;
}

//

/*****
*****/

//Function: receive bytes from UART
//input: iLen (received bytes length)
//output: if success return 1 or return 0
int ReceiveUART(short iLen ,char *pReceiveBuff)
{
    typ_msg_word msg;           //system message
    typ_UART_stat_word Usw;     //Uart state
```

```
int iCounter;
unsigned char ch;

iCounter=0;
SPT_set(640);

while(1)
{
    msg.s_word = sys_msg(SM_STAY_AWAKE);
    if (msg.bits.comm_data)           //comm data detecting
    {
        do{
            ch = (unsigned char)UART_get_char();
            pReceiveBuff[iCounter++]=ch;
            if (iCounter>iLen-1) break;
            Usw.l_word = UART_stat();
        }while(Usw.bits.buff_data_available);

    }
    if(iCounter==iLen)
    {
        delay_n_ms(1);
        return 1;
    }

    if (msg.bits.time_out)
    {
        SPT_set(640);
        return 0;
    }
}
//
```

```

/*****
*****/

//Function: clr received buffer after opening UART

//input:none

//output:none

void ClrUART(void)

{

int i;

for(i=0;i<8;i++)

{

    delay_n_ms(100);

    UART_get_char();

}

//

}

//

/*****
*****/

```

```

/*****
*****/

//Function: UART send a finger command

//input: pBuf(the pointer points to the command array) ,

//      chCode(CMD or ACK code) ,chLenHI(data length high byte),

//      chLenLO(data length low byte) ,chAP(ACK return)

//output:none

void UARTSendFingerCmd(char *pBuf, char chCode,char chLenHI,char chLenLO,char
chAP)

{

typ_UART_stat_word Usw;

int i;

int iCheckSUM;

//

```

```

pBuf[0x00]=DATA_HEAD;
pBuf[0x01]=DATA_CH;
pBuf[0x02]=chCode;
pBuf[0x03]=chLenHI;
pBuf[0x04]=chLenLO;
pBuf[0x05]=chAP;
//
iCheckSUM=0;
for(i=0x01;i<0x01+5;i++)
{
    iCheckSUM=iCheckSUM^pBuf[i];
}
pBuf[0x06]=iCheckSUM;
//
pBuf[0x07]=DATA_END;
//
for (i=0x00;i<0x00+8;i++)
{
    UART_send_char(pBuf[i]);
    do{
        Usw.l_word = UART_stat();
    } while (Usw.bits.out_busy);
}
return ;
}
//
/*****
*****/

//Function: UART send finger_value

//input: pBuf(the pointer points to the string array represents finger values), iLen(length of the array)

//output:none

void UARTSendFingerValue(char *pBuf,short iLen)
{

```

```

typ_UART_stat_word Usw;
int i;
int iCheckSUM=0;

for(i=0x000 ;i<0x000 + iLen;i++)
{
    iCheckSUM=iCheckSUM^pBuf[i];
}
//
UART_send_char(DATA_HEAD);
UART_send_char(0x00);
UART_send_char(0x00);
UART_send_char(0x00);
//
for (i=0x000 ;i<0x000 + iLen;i++)
{
    UART_send_char(pBuf[i]);
    do{
        Usw.l_word = UART_stat();
    } while (Usw.bits.out_busy);
}
//
UART_send_char(DATA_END);
return ;
}
//
/*****
*****/
/*****/
//Function : finger verification, comm with DSP and return a verification result
//input:none
//output: 1(fail) or 0(success)
unsigned short VerifyFinger(void)

```

```
{
//initialization UART

UART_init(UART_MODEM_ON|UART_ON|UART_8_DATA_BITS|UART_BAUD_1920
0);
UART_fcntl(UART_fcntl(UART_F_INQ)|UART_F_NO_CTS);
delay_n_ms(100);
    ClrUART();

UARTSendFingerCmd(g_SendCmdBuff,
                FINGER_VERIFY_CODE,
                0x00,
                FINGER_DATA_LEN + 0x03,
                0x00);
UARTSendFingerValue(g_SendDataBuff, FINGER_DATA_LEN);

clear_console();
move_cursor(3,3);
puts("Pls. put your finger");

if(ReceiveUART(0x08,g_ReceBuff))
{
    if (g_ReceBuff[0]= =DATA_HEAD && g_ReceBuff[2]= =FINGER_VERIFY_ACK
&& g_ReceBuff[7]= =DATA_END)
    {
        if (g_ReceBuff[5]==EW_RETURN_SUCCESS)
        {
            move_cursor(3,3);
            puts("Compare Successfully");
            delay_n_ms(1000);
            return 0;
        }

        if (g_ReceBuff[5]==EW_RETURN_FAIL)
```

```
    {
        move_cursor(3,3);
        puts("Compare failed");
        delay_n_ms(1000);
        return 1;
    }

    if (g_ReceBuff[5]==EW_TIME_OUT)
    {
        move_cursor(3,3);
        puts("Timeout");
        delay_n_ms(1000);
        return 1;
    }
    //
}
else
{
    move_cursor(2,3);
    puts("Communication parameters error");
    delay_n_ms(1000);
    return 1;
}
}
else
{
    move_cursor(3,3);
    puts("Serial port error");
    delay_n_ms(1000);
    return 1;
}
}
UART_init(UART_OFF);
//
```

```

}
//
/*****
/*****
//Function: write finger values into IC card from DSP
//input: none
//output: if success return 1, or return 0
short WriteFingerIntoCard(void)
{
//initialization UART
UART_init(UART_MODEM_ON|UART_ON|UART_8_DATA_BITS|UART_BAUD_1920
0);
UART_fcntl(UART_fcntl(UART_F_INQ)|UART_F_NO_CTS);
delay_n_ms(100);
ClrUART();

UARTSendFingerCmd(g_SendCmdBuff,GET_VALUE_CODE,0x00,0x00,0x00);

clear_console();
move_cursor(3,3);
puts("Pls. put your finger");
//
//256 bytes finger values and ACK command(8bytes) +HEAD,END,CHK
if(ReceiveUART(FINGER_DATA_LEN + 0x08 + 0x03 ,g_ReceBuff)
{
if ( g_ReceBuff[0]= =DATA_HEAD &&
g_ReceBuff[2]= =GET_VALUE_ACK &&
g_ReceBuff[7]= =DATA_END)
{
if(g_ReceBuff[5]==EW_RETURN_SUCCESS)
{
memcpy(g_TempBuff,g_ReceBuff + 0x08 + 0x02 -
0x01 ,FINGER_DATA_LEN);
if(!WriteCard( FIRST_FINGER_START_ADDR,

```

```
        FINGER_DATA_LEN,
        g_TempBuff)
    {
        clear_console();
        move_cursor(3,3);
        puts("Write card failed");
        delay_n_ms(1000);
        return 0;
    }
    else
    {
        clear_console();
        move_cursor(3,3);
        puts("Finish writing");
        return 1;
    }
}
else
{
    clear_console();
    move_cursor(3,3);
    puts("Collect failed");
    return 0;
}
}
else
{
    clear_console();
    move_cursor(1,3);
    puts("Communication parameters error");
    return 0;
}
}
```

```
else
{
    clear_console();
    move_cursor(3,3);
    puts("Serial port error");
    return 0;
}
UART_init(UART_OFF);
}
/*****
```

Affix 3 Commands List

Command Name	Content	Command(CMD)										Answer(ACK)	
		Format										Return Value	Format
CMD_GET_USER_SUM_DB	Query the user's fingerprint number in the module	HE	CH	C	P1	P2	P3	CHK	EN				
CMD_GET_USER_RIGHT_DB	Query the user's power	FE	CH	05	00	00	00	CHK	FD			AP=User power	FE+CH+40+00+00+A P+CHK+FD
CMD_VERIFY_DB	Compare the fingerprint collected with the one in the module.	FE	CH	00	P1	P2	00	CH	K	P1P2 is the user's number		AP=Return processing result.	FE+CH+41+00+00+A P+CHK+FD
CMD_REG_START_DB	Collect the fingerprint for the first time	FE	CH	01	P1	P2	00	CH	K	P1P2 is the user's number		AP= Return processing result.	FE+CH+42+00+00+A P+CHK+FD
CMD_REG_SECOND_DB	Collect the fingerprint for the second time.	FE	CH	02	P1	P2	P3	CH	K	P1P2 is the user's number and P3 is the user's power		AP= Return processing result.	FE+CH+44+00+00+A P+CHK+FD
CMD_REG_END_DB	Collect the fingerprint for the third time	FE	CH	03	P1	P2	P3	CH	K	P1P2 is the user's number and P3 is the user's power		AP= Return processing result.	FE+CH+43+00+00+A P+CHK+FD

[NOTE] All of the data is the hex data in the following list

CMD_REG_DELETE_DB	Delete the appointed user.	FE CH 20 P1 P2 00 CH FD PIP2 is the user's number.	AP= Return processing result.	FE+CH+60+00+00+A P+CHK+FD
CMD_REG_ALLDEL_DB	Delete all of the users	FE CH 21 00 00 P3 CH FD P3 is the user's power	AP= Return processing result.	FE+CH+61+00+00+A P+CHK+FD
CMD_GET_USER_NUMBER_DB	Show the user's number and his power who has registered.	FE CH 10 P1 P2 P3 CH FD PIP2 is the beginning user's record number. P3 is the fixed return number, witch should be less than 30.	Return the user's number P1+P2 and his power AP.	FE+CH+50+00+XX+ AP+CHK+FD FE+P1 ₁ +P2 ₁ +AP ₁ + +P1 _{p3} +P2 _{p3} +AP _{p3} +CH K+FD P1+P2 is the user's number Ap is user's power The length form P1 ₁ to AP _{p3} is XX
CMD_GET_VALUE	Collect the fingerprint image and get fingerprint eigenvalue.	FE CH 26 00 00 00 CH FD	AP= Return processing result.	FE+CH+66+00+XX+ AP+CHK+FD FE+XX (eigenvalue) +CHK+FD
CMD_IDENTIFY_DB	Collect the fingerprint image and compare it with all of the fingerprints in the library.	FE CH 12 00 00 00 CH FD	Return the compare result AP and user's number P1+P2	FE+CH+52+P1+P2+A P+CHK+FD
CMD_FROM_VALUE_DB	Send the eigenvalue and save it to the fingerprint library.	FE CH 31 XX XX 00 CHK FD FE+P1+P2+P+(XXXX-3) fingerprint eigenvalue +CHK+FD PIP2 is the user's number and P3 is the user's power XXXX is the length of the transported characters.	Return processing result.	FE+CH+71+P1+P2+A P+CHK+FD PIP2 is user's number and AP is the result.

CMD_TO_VALUE_DB	Get the appointed user's eigenvalue in the module database.	<table border="1"> <tr> <td>FE</td> <td>CH</td> <td>22</td> <td>P1</td> <td>P2</td> <td>00</td> <td>CH</td> <td>FD</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>K</td> <td></td> </tr> </table> <p>PIP2 is the user's number.</p>	FE	CH	22	P1	P2	00	CH	FD							K		Return processing result and user's number	FE+CH+62+00+XX+AP+CHK+FD FE+XX (eigenvalue)+CHK+FD
FE	CH	22	P1	P2	00	CH	FD													
						K														
CMD_FROM_VERIFY_DB	Send fingerprint eigenvalue to the module and compare it with the appointed one in the library.	<table border="1"> <tr> <td>FE</td> <td>CH</td> <td>33</td> <td>XX</td> <td>XX</td> <td>00</td> <td>CH</td> <td>FD</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>K</td> <td></td> </tr> </table> <p>FE+P1+P2+P3+ (XXXX-3) fingerprint eigenvalue +CHK+FD PIP2 is the user's number and P3 is the user's power XXXX is the length of the transported characters.</p>	FE	CH	33	XX	XX	00	CH	FD							K		Return processing result AP.	FE+CH+73+00+00+AP+CHK+FD
FE	CH	33	XX	XX	00	CH	FD													
						K														
CMD_FROM_VERIFY	Send the fingerprint eigenvalue to the module and compare it with the one collected by the module.	<table border="1"> <tr> <td>FE</td> <td>CH</td> <td>35</td> <td>XX</td> <td>XX</td> <td>00</td> <td>CH</td> <td>FD</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>K</td> <td></td> </tr> </table> <p>FE+00+00+00+ (XXXX-3) fingerprint eigenvalue +CHK+FD XXXX is the length of the transported characters.</p>	FE	CH	35	XX	XX	00	CH	FD							K		Return processing result AP.	FE+CH+75+00+00+AP+CHK+FD
FE	CH	35	XX	XX	00	CH	FD													
						K														
CMD_FROM_IDENTIFY_DB	Send the fingerprint eigenvalue to the module and compare it with all of the ones in the library.	<table border="1"> <tr> <td>FE</td> <td>CH</td> <td>34</td> <td>XX</td> <td>XX</td> <td>00</td> <td>CH</td> <td>FD</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>K</td> <td></td> </tr> </table> <p>FE+00+00+00+ (XXXX-3) fingerprint eigenvalue +CHK+FD XXXX is the length of the transported characters</p>	FE	CH	34	XX	XX	00	CH	FD							K		Return processing result AP and the user's number P1+P2	FE+CH+74+P1+P2+AP+CHK+FD
FE	CH	34	XX	XX	00	CH	FD													
						K														
CMD_GET_IMAGE	Collect the fingerprint image and get its outline.	<table border="1"> <tr> <td>FE</td> <td>CH</td> <td>27</td> <td>00</td> <td>00</td> <td>00</td> <td>CH</td> <td>FD</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>K</td> <td></td> </tr> </table>	FE	CH	27	00	00	00	CH	FD							K		Return processing result AP.	FE+CH+67+XX+XX+AP+CHK+FD FE+XXXX (Image value)+FD No check for image
FE	CH	27	00	00	00	CH	FD													
						K														
CMD_SET_BAUD	Set transport speed.	<table border="1"> <tr> <td>FE</td> <td>CH</td> <td>0F</td> <td>00</td> <td>00</td> <td>P3</td> <td>CH</td> <td>FD</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>K</td> <td></td> </tr> </table> <p>P3: New transport speed</p>	FE	CH	0F	00	00	P3	CH	FD							K		AP=Return the original speed.	FE+CH+4F+00+00+AP+CHK+FD
FE	CH	0F	00	00	P3	CH	FD													
						K														

CMD_GET_VERSION	Query the module's version information.	<table border="1" data-bbox="288 819 355 1489"> <tr> <td data-bbox="288 819 355 913">FE</td> <td data-bbox="288 913 355 987">CH</td> <td data-bbox="288 987 355 1061">2A</td> <td data-bbox="288 1061 355 1135">00</td> <td data-bbox="288 1135 355 1209">00</td> <td data-bbox="288 1209 355 1283">00</td> <td data-bbox="288 1283 355 1357">00</td> <td data-bbox="288 1357 355 1431">CH</td> <td data-bbox="288 1431 355 1489">K</td> <td data-bbox="288 1489 355 1563">FD</td> </tr> </table>	FE	CH	2A	00	00	00	00	CH	K	FD	Return the version information.	FE+CH+6A+00+XX+AP+CHK+FD FE+00XX (version information)+CHK+FD
FE	CH	2A	00	00	00	00	CH	K	FD					
CMD_PROCESS_IMAGE	Get the images' fingerprint eigenvalues in the module.	<table border="1" data-bbox="459 819 526 1489"> <tr> <td data-bbox="459 819 526 913">FE</td> <td data-bbox="459 913 526 987">CH</td> <td data-bbox="459 987 526 1061">2D</td> <td data-bbox="459 1061 526 1135">00</td> <td data-bbox="459 1135 526 1209">00</td> <td data-bbox="459 1209 526 1283">00</td> <td data-bbox="459 1283 526 1357">00</td> <td data-bbox="459 1357 526 1431">CH</td> <td data-bbox="459 1431 526 1489">K</td> <td data-bbox="459 1489 526 1563">FD</td> </tr> </table>	FE	CH	2D	00	00	00	00	CH	K	FD	Return processing result AP.	FE+CH+6D+00+XX+AP+CHK+FD FE+XX (eigenvalue)+CHK+FD
FE	CH	2D	00	00	00	00	CH	K	FD					
CMD_TEST_COMM	Test if the communication works well.	<table border="1" data-bbox="584 819 651 1489"> <tr> <td data-bbox="584 819 651 913">FE</td> <td data-bbox="584 913 651 987">CH</td> <td data-bbox="584 987 651 1061">2C</td> <td data-bbox="584 1061 651 1135">00</td> <td data-bbox="584 1135 651 1209">00</td> <td data-bbox="584 1209 651 1283">00</td> <td data-bbox="584 1283 651 1357">00</td> <td data-bbox="584 1357 651 1431">CH</td> <td data-bbox="584 1431 651 1489">K</td> <td data-bbox="584 1489 651 1563">FD</td> </tr> </table>	FE	CH	2C	00	00	00	00	CH	K	FD	Return processing result AP.	FE+CH+6C+00+00+AP+CHK+FD
FE	CH	2C	00	00	00	00	CH	K	FD					