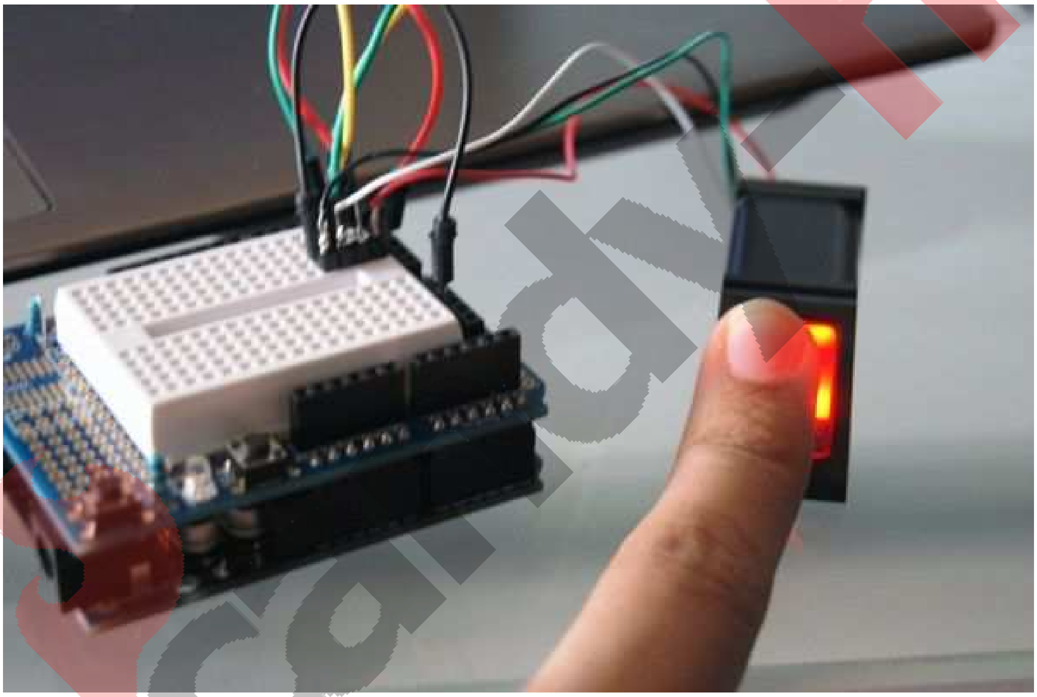
www.candy-ho.com | Contactanos | 1139685940 | **ventas@candy-ho.com** | Mejico 3941 Unidad 1, Villa Martelli | | Lunes a Viernes 10:00 a 18:00 |

Sensor lector de huella digital para Arduino

Ahora puedes asegurar tus proyectos electrónicos incluyendo identificación biométrica con el **sensor lector de huella digital para Arduino**. Este módulo integra toda la electrónica y algoritmos necesarios para hacer la verificación de una huella digital una tarea super simple. Este tipo de módulos se utilizan habitualmente en cajas de seguridad, equipos checadores y controles de acceso. El módulo integra un DSP que realiza todo el procesamiento de imagenes, localización de minucias, creación y comparación de templates para ubicar a los usuarios enrolados, etc. Esto facilita muchísimo la tarea del intergrador, pues no hay que preocuparse por algoritmos complejos.

La conexión con el Lector de huella digital para Arduino se realiza con una interfaz seria asíncrona con niveles TTL. Existen comandos en la interfaz serial para capturar imagenes, detectar huellas y buscar en la base de datos del módulo. También se pueden enrolar huellas directamente en el módulo. Se pueden guardar hasta 1000 usuarios en la memoria FLASH interna del Lector de huella digital para Arduino e identificarlos posteriormente. El módulo cuenta con un led que asiste en la lectura de la huella y que permite saber que esta funcionado.

El sensor biométrico de huella digital es ideal para realizar un sistema capaz de proteger lo que tu requieras por medio del análisis de tu huella digital. El sistema realiza procesamiento digital de imágenes interno con un DSP ademas de incluir capacidades de comparación en base de datos y actualización de la misma. El dispositivo funciona con el protocolo serial, por lo que puede ser utilizado con cualquier microcontrolador o tarjeta de desarrollo.



candy-ho

Características del Lector de huella digital para Arduino

- Módulo Lector de huella digital para Arduino y otros microcontroladores con interfaz UART
- Fuente de alimentación: 3.8 – 7.0V
- Corriente en operación típica: 65 mA
- Interfaz: UART TTL

- Baudrate: 9600-115200 bps
- Tiempo de adquisición: <1S
- Capacidad de almacenamiento: 1000 usuarios
- FAR: <0.001%
- FRR: <1.0%
- Tiempo de búsqueda promedio: <1S
- Character file: 256 bytes
- Template file: 512 bytes
- Nivel de seguridad: 5
- Condiciones de trabajo: -20°C – +60; RH: 40%-85%
- Condiciones de almacenamiento: -40°C – +85°C; RH: <85%
- Dimensiones del panel táctil: 14.5*19.4 mm (0.57*0.76”)
- Dimensiones del módulo: 54*20*20.5 mm (2.13*0.79*0.81”)

El dispositivo tiene la capacidad de almacenar hasta 162 huellas dactilares en su memoria FLASH interna. El LED del dispositivo se ilumina cada que se encuentra tomando imagenes en busca de huellas digitales.

Para poder utilizar el dispositivo es necesario guardar las huellas en la base de datos del mismo. Estas huellas se les asigna un ID. Posteriormente se puede iniciar la secuencia de lectura y comparación para verificar las huellas de los usuarios y así poder discernir y ejecutar acciones en base al resultado.

candy-ho

Precauciones

Para utilizar el código de ejemplo es necesario primero tomar en cuenta que los cables del dispositivo no tienen acoplado ningún conector por lo que se recomienda soldar pines macho o pines hembra, ya que los headers de arduino o de un protoboard no hacen bien contacto y son motivo de que el dispositivo no funcione correctamente!.

candy-ho

Conexiones Con Arduino Y Biblioteca

Conexiones:

- GND : Negro
- Pin D2 : Verde
- Pin D3: Blanco
- 5V: Rojo

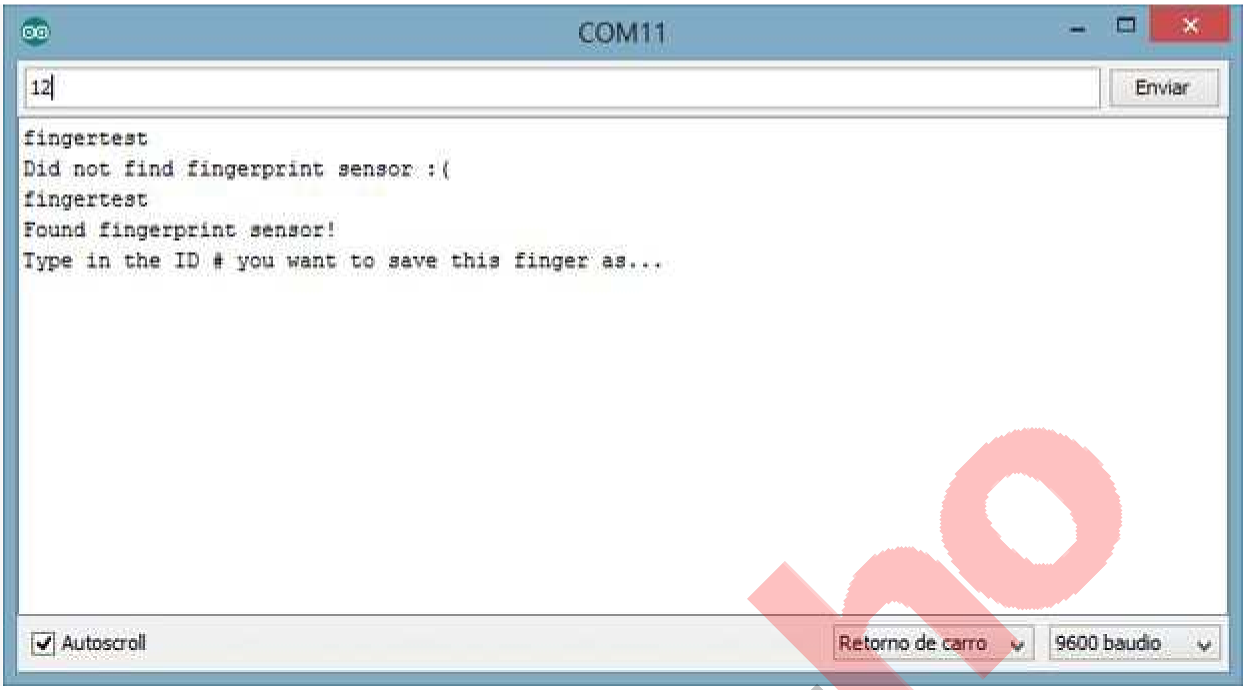
Primero procedemos a descargar la biblioteca para Arduino del siguiente link:

https://github.com/adafruit/Adafruit-Fingerprint-S...

Una vez descargada, se descomprime la biblioteca y se guarda dentro de: C:\Program Files (x86)\Arduino\libraries\ Es necesario renombrar la carpeta de la biblioteca en caso de que se encuentre con un nombre diferente que el archivo ".cpp" que se encuentra en la misma.

candy-ho

Cargar Huellas En El Sensor

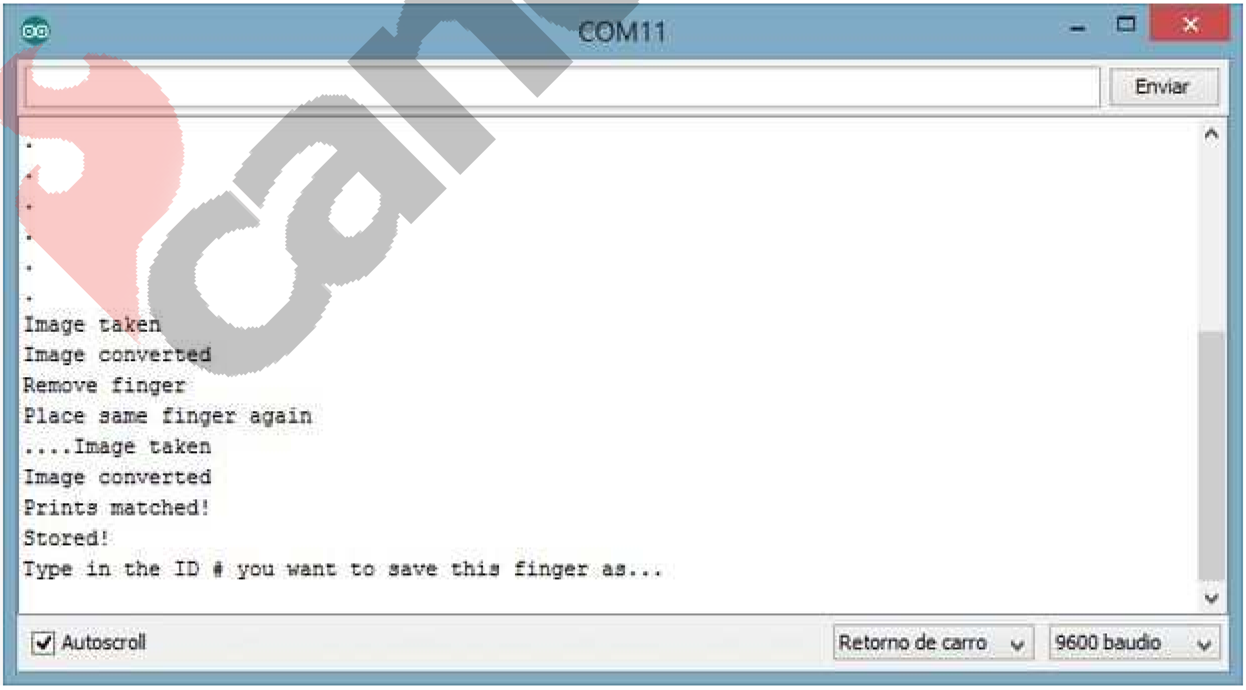


12

```
fingertest
Did not find fingerprint sensor :(
fingertest
Found fingerprint sensor!
Type in the ID # you want to save this finger as...
```

Abrimos el IDE de Arduino y seleccionamos Archivo-Ejemplos- y buscamos la biblioteca que acabamos de instalar y seleccionamos el ejemplo deenroll. En este ejemplo primero identificara si el sensor se encuentra conectado. Si lo detecta primero preguntara una ID para asignarla a la huella a introducir.

Cargar Mas Huellas

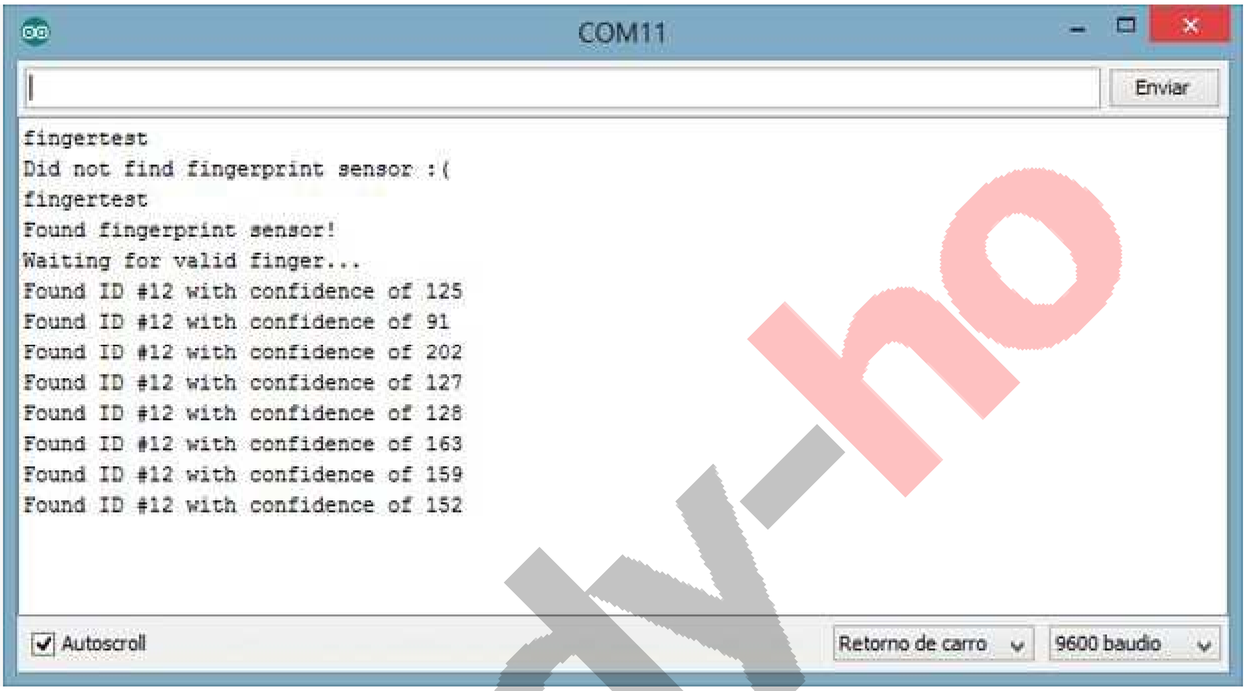


```
.
.
.
.
.
.
Image taken
Image converted
Remove finger
Place same finger again
....Image taken
Image converted
Prints matched!
Stored!
Type in the ID # you want to save this finger as...
```

candy-ho

Una vez enviado el ID deseado ponemos la huella en el sensor, la retiramos y la volvemos a poner para tomar una captura redundante de 2 imágenes. Podemos seguir este proceso para dar de alta en la base de datos del dispositivo todas las huellas que se requieran, ya que estas se guardan en la memoria interna del mismo.

Leer Huellas



```
fingertest
Did not find fingerprint sensor :(
fingertest
Found fingerprint sensor!
Waiting for valid finger...
Found ID #12 with confidence of 125
Found ID #12 with confidence of 91
Found ID #12 with confidence of 202
Found ID #12 with confidence of 127
Found ID #12 with confidence of 128
Found ID #12 with confidence of 163
Found ID #12 with confidence of 159
Found ID #12 with confidence of 152
```

Una vez dadas de altas las huellas, se puede cargar el ejemplo fingerprint, este lee la huella del sensor y nos dice que tan coherente es un resultado con su base de datos, siempre y cuando haya sido un resultado positivo, si no encuentra huella, no envía nada.

Codigo

El sensor funciona a 57600 baudios, se puede configurar pero esta es la velocidad por defecto, al hacer uso del serial, el arduino utiliza la biblioteca de serial por software.

```
#include <SoftwareSerial.h>
```

Si se requiere cambiar de pines el serial por software se puede hacer en la siguiente instrucción:

```
SoftwareSerial mySerial(2, 3);
```

candy-ho

Para el ejemplo de fingerprint, si se requieres que el arduino ejecute una accion al haber encontrado una huella, es necesario indicarlo en esta seccion de codigo:

```
Serial.print("Found ID #");

Serial.print(finger.fingerID);

Serial.print("With confidence of");

Serial.println(finger.confidence);

// Escribir el codigo aqui

return finger.fingerID;
```

candy-ho

UART Fingerprint Reader User Manual

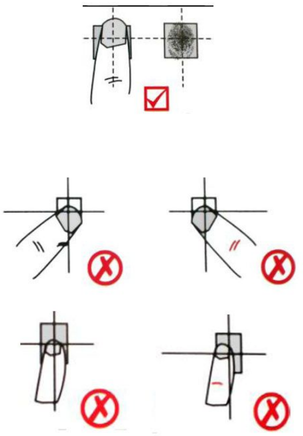
Contents

candy-ho

Overview

Note:
Module mentioned in this manual are using imported high-precision components, when gathering the fingerprint, your fingers only need to touch the acquisition area gently, no need to push it down, fingerprint module can be identified quickly.
If you want to power by battery, it is recommended to use alkaline batteries.

The correct use of the fingerprint module:



Hardware connection:
VCC ------ 3.3V or 5V
GND ------ GND
TXD (serial port of fingerprint module send) ------ RXD (serial port of PC or microcontroller receive)
RXD (serial port of fingerprint module receive) ------ TXD (serial port of PC or microcontroller send)
BL (fingerprint of backlight, connection is not a must) ------ IO port
RST (fingerprint module reset, connection is not a must) ------ IO port

After get the module, you can first use the test software \UART-Fingerprint-Reader\software\Demo Software\ UART Fingerprint Reader.exe) to test the module.

After the hardware connection is completed, run (\UART-Fingerprint-Reader\software\Demo Software\Register Controls. bat) registration control first, and then open (\UART-Fingerprint-Reader\software\Demo Software\ UART Fingerprint Reader.exe)

For details, please refer to (\UART-Fingerprint-Reader\software\Demo Software\ UART Fingerprint Reader.exe))

After get a certain understanding of the module, the below development protocol can be used for secondary development.

candy-ho

Fingerprint Module Development Protocol

1. Communication

DSP module works as a slave device, the master device control it by sending related commands.
Command interface: 19200bps; 1 start bits; 1 stop bits (parity bits: none)

Commands sent by master device and DSP module response can be divided into two categories according to data length:

1) = 8 bytes, data format as below:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	CMD	P1	P2	P3	0	CHK	0xF5
Respond	0xF5	CMD	Q1	Q2	Q3	0	CHK	0xF5

Note:

CMD: Command / response type

P1, P2, P3: Command parameter

Q1, Q2, Q3: Response parameter

Q3 mainly used to return the effective operating information, there will be the following values:

```
#define ACK_SUCCESS      0x00     //Operation successfully
#define ACK_FAIL         0x01     // Operation failed
#define ACK_FULL         0x04     // Fingerprint database is full
#define ACK_NOUSER       0x05     //No such user
#define ACK_USER_EXIST 0x06       // User already exists
#define ACK_FIN_EXIST    0x07     // Fingerprint  already exists
#define ACK_TIMEOUT      0x08     // Acquisition timeout
```

CHK: checksum value, XOR value for the second byte to the sixth byte

2) > 8 bytes, data includes two parts: data head + data package

Data header format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	CMD	Hi(Len)	Low(Len)	0	0	CHK	0xF5
Response	0xF5	CMD	Hi(Len)	Low(Len)	Q3	0	CHK	0xF5

Note:

CMD: Q3 definition as the above.

Len: Effective length of the data is 16 bits, consists two bytes

Hi (Len): Data packet length high 8-bit

Low (Len): Data packet length low 8-bit

CHK: checksum value, XOR value for the second byte to the sixth byte

Data packet format:

Byte	1	2…Len + 1	Len + 2	Len + 3
Command	0xF5	Data	CHK	0xF5
Response	0xF5	Data	CHK	0xF5

sandy-ho

Note:

Len is the number of byte of the Data;

CHK: checksum value, XOR value for the second byte to the Len+1 byte

Send data packet immediately after send the data header.

2. Description of each communication protocol command

2.1 Enable the module into a dormant state (Both command and response are 8 bytes)

Command data format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x2C	0	0	0	0	CHK	0xF5

Response data format:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x2C	0	0	0	0	CHK	0xF5

2.2 Set / read the fingerprint add mode (Both command and response are 8 bytes)

There are two modes for dding fingerprint: Allow repeat mode / prohibit repeat mode, in the "prohibit repeat mode", the same finger can add one user only, if forced to add a second user will return an error message. After power, the system is in prohibiting repeat mode.

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x2D	0	Byte5=0: 0: allow repeat 1: prohibit repeat Byte5=1: 0	0: set new add mode 1: read current add mode	0	CHK	0xF5
Response	0xF5	0x2D	0	Current add mode	ACK_SUCCUSS ACK_FAIL	0	CHK	0xF5

2.3 Add fingerprint (Both command and response are 8 bytes)

To ensure the effectiveness, user must input a fingerprint three times, the host is required to send command to the DSP module three times.

(1) The first time

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x01	User ID(high 8-bit)	User ID(low 8-bit)	User privilege(1/2/3)	0	CHK	0xF5
Response	0xF5	0x01	0	0	ACK_SUCCESS ACK_FAIL ACK_FULL ACK_TIMEOUT	0	CHK	0xF5

Note:

candy-ho

Range of user number is 1 - 0xFFF;

Range of User privilege is 1, 2, 3, its meaning is defined by secondary developers themselves.

(2) The second time:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x02	User ID(high 8-bit)	User ID(low 8-bit)	User privilege(1/2/3)	0	CHK	0xF5
Response	0xF5	0x02	0	0	ACK_SUCCESS ACK_FAIL ACK_TIMEOUT	0	CHK	0xF5

(3) The third time:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x03	User ID(high 8-bit)	User ID(low 8-bit)	user privilege(1/2/3)	0	CHK	0xF5
Response	0xF5	0x03	0	0	ACK_SUCCESS ACK_FAIL ACK_USER_EXIST ACK_TIMEOUT	0	CHK	0xF5

Note: User ID and user privilege should be in the same value in the three commands.

2.4 Delete specified user (Both command and response are 8 bytes)

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x04	User ID(high 8-bit)	User ID(low 8-bit)	0	0	CHK	0xF5
Response	0xF5	0x04	0	0	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2.5 Delete all users (Both command and response are 8 bytes)

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x05	0	0	0	0	CHK	0xF5
Response	0xF5	0x05	0	0	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2.6 Acquire the total number of users (Both command and response are 8 bytes)

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x09	0	0	0	0	CHK	0xF5
Response	0xF5	0x09	User number(high 8-bit)	User number(low 8-bit)	ACK_SUCCESS	0	CHK	0xF5

candy-ho

					ACK_FAIL				

2.7 Compare 1:1 (Both command and response are 8 bytes)

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x0B	User ID(high 8-bit)	User ID(low 8-bit)	0	0	CHK	0xF5
Response	0xF5	0x0B	0	0	ACK_SUCCESS ACK_FAIL ACK_TIMEOUT	0	CHK	0xF5

2.8 Compare 1: N (Both command and response are 8 bytes)

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x0C	0	0	0	0	CHK	0xF5
Response	0xF5	0x0C	User ID(high 8-bit)	User ID(low 8-bit)	user privilege(1/2/3) ACK_NOUSER ACK_TIMEOUT	0	CHK	0xF5

2.9 Acquire user privilege (Both command and response are 8 bytes)

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x0A	User ID(high 8-bit)	User ID(low 8-bit)	0	0	CHK	0xF5
Response	0xF5	0x0A	0	0	user privilege(1/2/3) ACK_NOUSER	0	CHK	0xF5

2.10 Acquire DSP module version number (command = 8 bytes, and response > 8 bytes)

Command data format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x26	0	0	0	0	CHK	0xF5

Response data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Respond	0xF5	0x26	Hi(Len)	Low(Len)	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2) Data packet:

Byte	1	2 --- Len + 1	Len + 2	Len + 3

candy-ho

Response	0xF5	Version data	CHK	0xF5

Note: This protocol won’t public currently.

2.11Set/ read comparison level (Both command and response are 8 bytes)

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x28	0	Byte5=0: New comparison level; Byte5=1: 0	0:Set new comparison level; 1:Read current comparison level	0	CHK	0xF5
Response	0xF5	0x28	0	Current comparison level	ACK_SUCCUSS ACK_FAIL	0	CHK	0xF5

Note: Range of the comparison level is 0-9, the greater the value, the more strict in comparison, default value is 5.

2.12 Acquire and upload images (Command = 8 bytes, response > 8 bytes)

Command data format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x24	0	0	0	0	CHK	0xF5

Response data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x24	Hi(Len)	Low(Len)	ACK_SUCCESS ACK_FAIL ACK_TIMEOUT	0	CHK	0xF5

2) Data packet:

Byte	1	2 --- Len + 1	Len + 2	Len + 3
Response	0xF5	Image data	CHK	0xF5

Note:

In DSP module, the fingerprint image is 248*296 pixels, grayness of each pixel is represented by 8 bits. During the upload process, in order to reduce the amount of data, jump pixel sampling in the horizontal / vertical direction, so that the image becomes 124*148, and take the grayness for high 4-bit, each two pixels composited into one byte for transferring (previous pixel low 4-bit, last pixel high 4-bit).

Transmission starts line by line from the first line, each line starts from the first pixel, totally transfer 124* 148/ 2 bytes of data.

Data length of image is fixed of 9176 bytes.

2.13 Upload acquired images and extracted eigenvalue (Command = 8 bytes, and response > 8 bytes)

Command data format:

candy-ho

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x23	0	0	0	0	CHK	0xF5

Response data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x23	Hi(Len)	Low(Len)	ACK_SUCCESS ACK_FAIL ACK_TIMEOUT	0	CHK	0xF5

2) Data packet:

Byte	1	2	3	4	5 --- Len + 1	Len + 2	Len + 3
Response	0xF5	0	0	0	Eigenvalues data	CHK	0xF5

Note: Eigenvalues data length Len-3 is fixed 193 bytes.

2.14 Download eigenvalues and acquire fingerprint comparison (Command > 8 bytes, response = 8 bytes)

Command data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x44	Hi(Len)	Low(Len)	0	0	CHK	0xF5

2) Data packet

Byte	1	2	3	4	5 --- Len + 1	Len + 2	Len + 3
Command	0xF5	0	0	0	Eigenvalues data	CHK	0xF5

Note: Eigenvalues data length Len-3 is fixed 193 bytes.

Response data format:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x44	0	0	ACK_SUCCESS ACK_FAIL ACK_TIMEOUT	0	CHK	0xF5

2.15 Download the fingerprint eigenvalues and DSP module database fingerprint compare 1: 1 (command>8 bytes/response=8 bytes)

Command data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x42	Hi(Len)	Low(Len)	0	0	CHK	0xF5

candy-ho

2) Data packet:

Byte	1	2	3	4	5 --- Len + 1	Len + 2	Len + 3
Command	0xF5	User ID(high 8-bit)	User ID(low 8-bit)	0	Eigenvalues data	CHK	0xF5

Note: Eigenvalues data length Len-3 is fixed 193 bytes.

Response data format:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x42	0	0	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2.16 Download the fingerprint eigenvalues and DSP module database fingerprint compare 1: N (command>8 bytes / response=8 bytes)

Command data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x43	Hi(Len)	Low(Len)	0	0	CHK	0xF5

2) Data packet:

Byte	1	2	3	4	5 --- Len + 1	Len + 2	Len + 3
Command	0xF5	0	0	0	Eigenvalues data	CHK	0xF5

Note: Eigenvalues data length Len-3 is fixed 193 bytes.

Response data format:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x43	User ID(high 8-bit)	User ID(low 8-bit)	User privilege(1/2/3) ACK_NOUSER	0	CHK	0xF5

candy-ho

2.17 Upload the DSP module database specified user eigenvalue (command = 8 bytes, response > 8 bytes)

Command data format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x31	User ID(high 8-bit)	User ID(low 8-bit)	0	0	CHK	0xF5

Response data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x31	Hi(Len)	Low(Len)	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

					ACK_NOUSER			

2) Data packet:

Byte	1	2	3	4	5 --- Len + 1	Len + 2	Len + 3
Response	0xF5	User ID(high 8-bit)	User ID(low 8-bit)	User privilege(1/2/3)	Eigenvalues data	CHK	0xF5

Note: Eigenvalues data length Len-3 is fixed 193 bytes.

2.18 Download the eigenvalue and save to the DSP module database according to the specified user number (command>8 bytes/response=8 bytes)

Command data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x41	Hi(Len)	Low(Len)		0	CHK	0xF5

2) Data packet:

Byte	1	2	3	4	5--- Len + 1	Len + 2	Len + 3
Command	0xF5	User ID(high 8-bit)	User ID(low 8-bit)	User privilege(1/2/3)	Eigenvalues data	CHK	0xF5

Note: Eigenvalues data length Len-3 is fixed 193 bytes.

Response data format:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x41	User ID(high 8-bit)	User ID(low 8-bit)	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2.19 Acquire all logged in user numbers and user privilege (command = 8 bytes, response > 8 bytes)

Command data format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x2B	0	0	0	0	CHK	0xF5

Response data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x2B	Hi(Len)	Low(Len)	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2) Data packet:

Byte	1	2	3	4 --- Len + 1	Len + 2	Len + 3

candy-ho

Response	0xF5	User number(high 8-bit)	User number(low 8-bit)	User information data(User ID and privilege)	CHK	0xF5

Note:

Data length Len in the Data packet is fixed “3”, the user number is fixed “+2”.

User information data format as below:

Byte	4	5	6	7	8	9	…
Data	User ID1(high 8-bit)	User ID1(high 8-bit)	User ID1(high 8-bit)	User ID2(high 8-bit)	User ID2(high 8-bit)	User ID2(high 8-bit)	…

2.20 Acquire a single record data (command = 8 bytes, response >8 bytes)—Note: The module won’t provide the protocol currently.

This protocol return data records in the library which is designated by the "record location".

Command data format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x38	Record location(high 8-bit)	Record location(low 8-bit)	0	0	CHK	0xF5

Response data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x38	Hi(Len)	Low(Len)	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2）Data packet:

Byte	1	2	3	4	5
Response	0xF5	Digit 7-1:Year Digit:0: Month(Digit 3)	Digit:7-5:Month(Digit2-0) Digit 4-0:Date	Digit 7-2: Hour Digit 1-0: Minute(Digit 5-4)	Digit 7-4: Minute(Digit 3-0) Digit 3-0: Record No.(Digit 21-18)

Byte	6	7	8	9	10	11
Response	Record No.(Digit 17-10)	Record No. (Digit 9-2)	Digit 7-6: Record No.(Digit 1-0) Digit 5-0: User ID(Digit 13-8)	User ID(Digit 7-0)	CHK	0xF5

Note:

Record data length Len is fixed 8;

2.21 Acquire new record data (command = 8 bytes, response > 8 bytes) – Note: The module won’t provide the protocol currently.

This protocol returns 50 continuous record data which is greater or equal to "minimum record number" in the record library.

candy-ho

Command data format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x39	Digit 7-6:0 Digit 5-0:Minimum Record No.(Digit 21-16)	Minimum Record No.(Digit 15-8)	Minimum Record No.(Digit 7-0)	0	CHK	0xF5

Response data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x39	Hi(Len)	Low(Len)	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2) Data packet:

Byte	1	2---9	10-17	…	Len + 2	Len + 3
Response	0xF5	The first record	The second record	…	CHK	0xF5

Note:

Format of each record in the data packet is the same as byte 2-byte 9 in response data packet in above 2.20.

Data length Len is fixed (8*50=400 bytes).

2.22 Wipe the data record (Both command and response are 8 bytes)—Note: The module won't provide the protocol currently.

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x3A	0	0	0	0	CHK	0xF5
Response	0xF5	0x3A	0	0	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2.23 Set Module time (command > 8 bytes, response = 8 bytes) -- Note: The module won't provide the protocol currently.

1) Data header:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x48	Hi(Len)	Low(Len)	0	0	CHK	0xF5

3) Data packet:

Byte	1	2	3	4	5	6	7	8	9	10
Command	0xF5	Week	Year	Month	Date	Hour	Minute	second	CHK	0xF5

Note:

Time data length Len is fixed 7.

Response data format:

Byte	1	2	3	4	5	6	7	8

candy-ho

Response	0xF5	0x48	0	0	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2.24 Read system time -- Note: The module won't provide the protocol currently.

Command data format:

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x3C	0	0	0	0	CHK	0xF5

Response data format:

1) Data header:

Byte	1	2	3	4	5	6	7	8
Response	0xF5	0x3C	Hi(Len)	Low(Len)	ACK_SUCCESS ACK_FAIL	0	CHK	0xF5

2）Data packet:

Byte	1	2	3	4	5	6	7	8	9	10
Response	0xF5	Week	Year	Month	Date	Hour	Minute	Second	CHK	0xF5

Note:

Time data length Len is fixed 7.

2.25 Set/read fingerprint capture timeout value (Both command and response are 8 bytes)

Byte	1	2	3	4	5	6	7	8
Command	0xF5	0x2E	0	Byte5=0: New timeout value; Byte5=1: 0	0: Set new timeout value 1: Read current timeout value	0	CHK	0xF5
Response	0xF5	0x2E	0	Current timeout value	ACK_SUCCUSS ACK_FAIL	0	CHK	0xF5

Note:

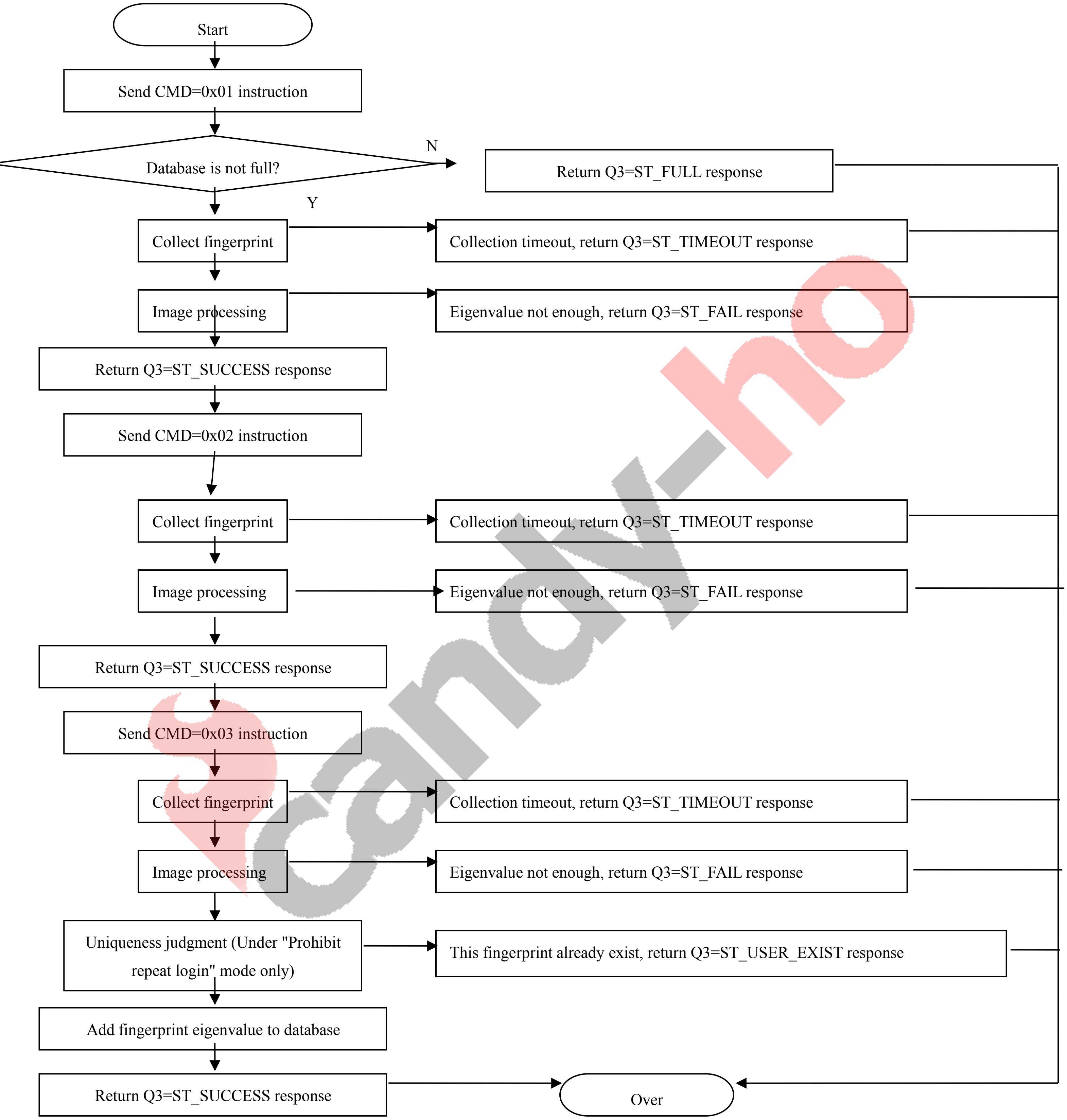
Range of fingerprint waiting timeout (tout) value is 0-255. If the value is 0, the fingerprint acquisition process will keep continue if no fingerprints press on; If the value is not 0, the system will exist for reason of timeout if no fingerprints press on in time tout * T0.

Note: T0 is the time required for collecting/processing an image, usually 0.2- 0.3 s.

candy-ho

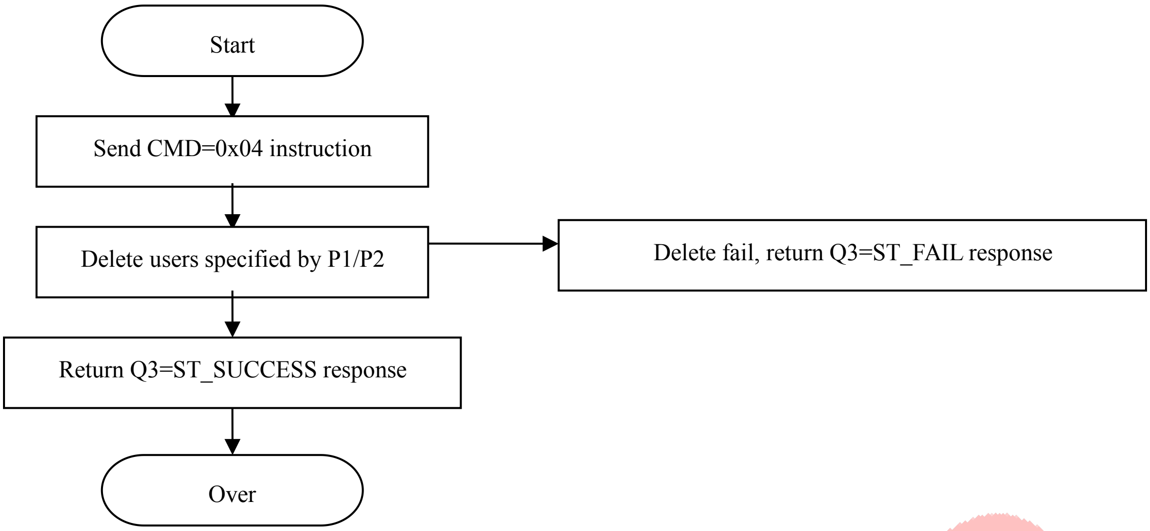
A. Appendix

A.1 Add fingerprint process

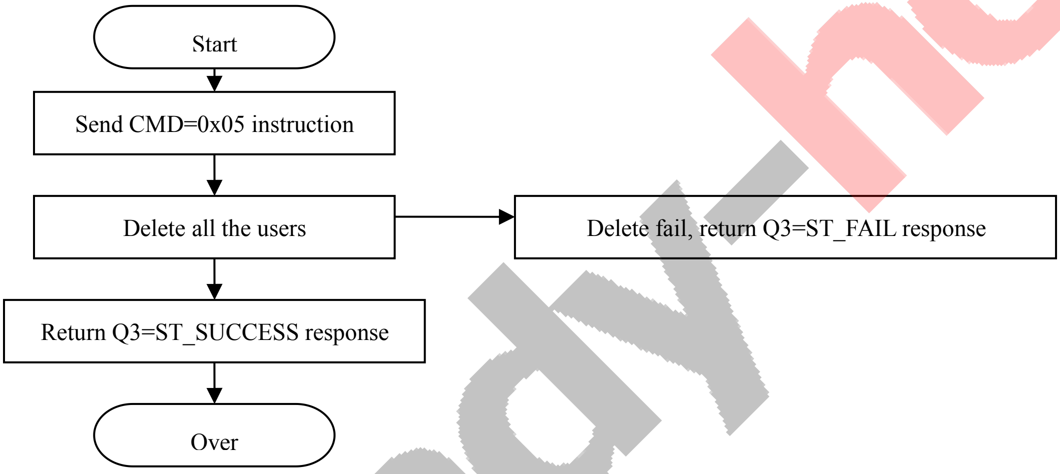

Start
Send CMD=0x01 instruction
Database is not full?
N
Y
Return Q3=ST_FULL response
Collect fingerprint
Collection timeout, return Q3=ST_TIMEOUT response
Image processing
Eigenvalue not enough, return Q3=ST_FAIL response
Return Q3=ST_SUCCESS response
Send CMD=0x02 instruction
Collect fingerprint
Collection timeout, return Q3=ST_TIMEOUT response
Image processing
Eigenvalue not enough, return Q3=ST_FAIL response
Return Q3=ST_SUCCESS response
Send CMD=0x03 instruction
Collect fingerprint
Collection timeout, return Q3=ST_TIMEOUT response
Image processing
Eigenvalue not enough, return Q3=ST_FAIL response
Uniqueness judgment (Under "Prohibit repeat login" mode only)
This fingerprint already exist, return Q3=ST_USER_EXIST response
Add fingerprint eigenvalue to database
Return Q3=ST_SUCCESS response
Over

candy-ho

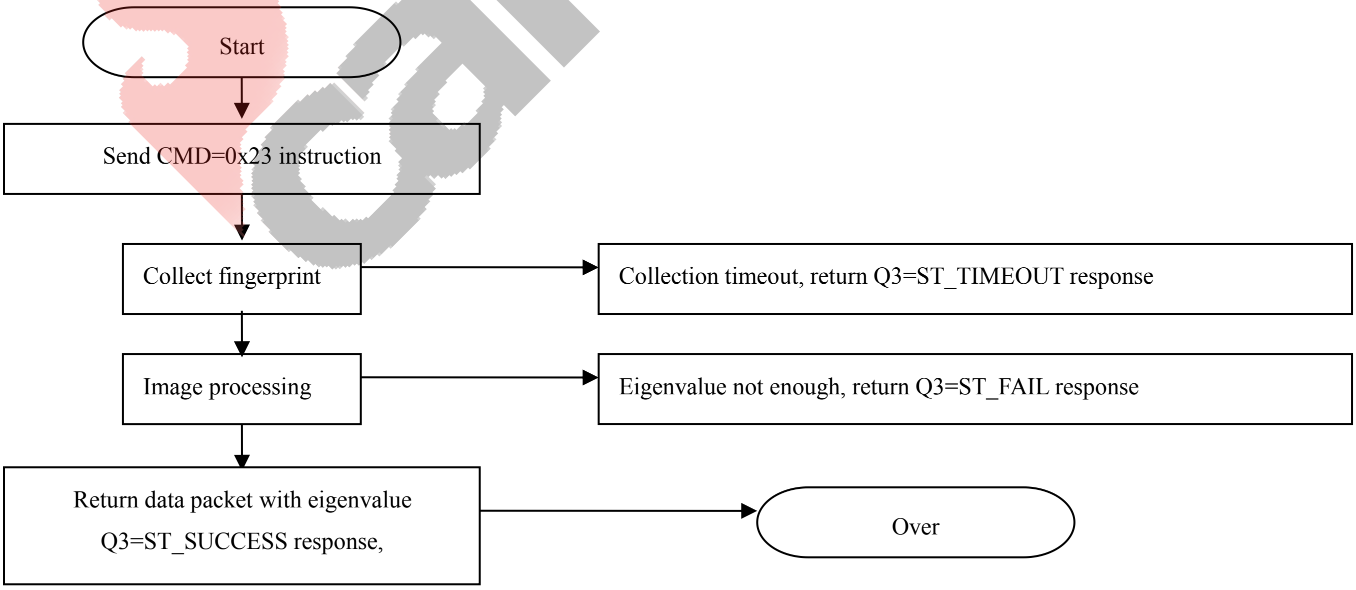
A.2 Delete specified user process


Start
Send CMD=0x04 instruction
Delete users specified by P1/P2
Delete fail, return Q3=ST_FAIL response
Return Q3=ST_SUCCESS response
Over

A.3 Delete all users process


Start
Send CMD=0x05 instruction
Delete all the users
Delete fail, return Q3=ST_FAIL response
Return Q3=ST_SUCCESS response
Over

A.4 Upload collected image and extracted eigenvalue process


Start
Send CMD=0x23 instruction
Collect fingerprint
Collection timeout, return Q3=ST_TIMEOUT response
Image processing
Eigenvalue not enough, return Q3=ST_FAIL response
Return data packet with eigenvalue
Q3=ST_SUCCESS response,
Over

candy-ho