SilkID-SDK Interfaces

Version: V2.3

Contents

A. Interface Functions

1. Connecting

Interface: Connect

Function: Connecting the module.

Parameter: dev**: Serial device. type:1-232 communication,0-usb communication.**

Note**: When the communication type is USB, no need to set "dev".When the communication type is 232, if the input value for "dev" is not set, the SDK will open "/dev/ttys0".**

Returned value: 0-false, >0-true

```
int Connect(char *dev, int type);
```

2. Disconnecting

Interface: Disconnect

Function: Disconnecting the communication with the module.

Parameters: None

Note**: The module support 232 and USB communication. When the USB communication is connected, the Host can't connect the module by 232.The Host needs disconnecting the USB communication. If 232 communication is connected first, the Host also disconnects the 232 communication.**

Returned value: 0**-false, >**0**-true**

```
int DisConnect(void);
```

3. Enrolling User Templates

Interface: EnrollUserByScan

Function: Enrolling user templates

Description: A user needs to press a finger three times on the fingerprint reader of the module for enrollment.

Note: **During enrollment, a fingerprint template may fail to be enrolled but normally the specified user will be created.**

Parameter: userID**: user ID**

Returned value: 0**-false,** >0**-true**

int EnrollUserByScan (int userID);

4. Enrolling Fingerprint by Image

Interface: EnrollTemplateByImage

Function: Enrolling fingerprint by image (over the extended communication protocol).

Description: The host sends a fingerprint image to the module for fingerprint enrollment. The extended communication protocol is used to transmit fingerprint images.

Note: **An original image in the module may be different from the enrolled fingerprint image. One possible cause is that the fingerprint image is rotated and therefore the image data organization sequence is different from the original fingerprint image data when being sent. The transmitted image data is the correct image data.**

Parameters: int userID**: user ID;** data**: image data;** dataSize**: image data size.**

Returned value: 0**-false,** >0**-true.**

int EnrollTemplateByImage (int userID, unsigned char * data, int dataSize).

5. Getting Fingerprint Images from the Module

Interface: GetFingerImage

Function: Getting fingerprint images from the module

Description: After this command is sent, a user needs to press a finger on the module. After the finger is successfully pressed, the module sends the fingerprint image to the host over the extended protocol. The obtained data are correct image data.

Parameters: width**: image width;** height**: image height;** data**: image data**

Returned value: 0**-false,** >0**-true**

int GetFingerImage (int *width, int *height, unsigned char * data);

6. Identifying Users by Fingerprint Image

Interface: IdentifyByImage

Function: Identifying users by fingerprint image.

Description: After receiving the image identification command, the module sends a response for acknowledgment. Then, the module enters the state of receiving image data. In this case, the extended protocol is used for data transmission.

Parameters: imageSize**: fingerprint image size;** userID**: user ID; index: fingerprint** index**;** data**: fingerprint image.**

Returned value: 0**-false,** >0**-true**

```
int IdentifyByImage (int imageSize, int *userID, int *index, unsigned char * data);
```

7. Clearing All Data

Interface: ClearDB

Function: Clearing all data.

Description: When this interface is executed, the module deletes all fingerprint templates, user information and verified fingerprint logs.

Parameters: None

Returned value: 0**-false,** >0**-true**

```
int ClearDB (void);
```

8. Reading All Verified Logs

Interface: ReadAllLogs

Function: Reading all verified logs from the module to the host.

Description: This interface needs to be called in cycles to read all verified logs from the module. When logs are read, 1 is returned; otherwise, 0 is returned.

Parameters: userID**: user ID;**

event: Event occurring in storing logs;

verified: Verification method;

date: date;

time: time;

Reserved: This field can be read or set by the MD_LC **command.**

Returned value: 0-**false,** >0-**true**

```
int ReadAllLogs (int *userID,    char *event, char *verified, unsigned long *date, unsigned long *time, char *reserved);
```

Note: **The parameter** date **is time data which is not analyzed and it can be analyzed by the TimeAnalyse function.**

Interface: TimeAnalyse

Function: Analyzing the date and time in a data packet.

Description: Reading and analyzing data by calling the ReadAllLogs interface.

Parameters: date**: date data packet;** time**: time data packet;**

Year**: year;** Month**: month;** Day**: date;** Hour**: hour;** Minute**: minute;** Second**: second**

Returned value: None

```
void TimeAnalyse(unsigned long date, unsigned long time,
int *Year, int *Month, int *Day,    int *Hour, int *Minute, int *Second);
```

9. Setting Module Time

Interface: SetTime

Function: Setting module time

Parameters: Year**: year;** Month**: month;** Day**: date;** Hour**: hour;** Minute**: minute;** Second**: second**

Returned value: 0-**false,** >0-**true**

```
int SetTime (int Year, int Month, int Day, int Hour, int Minute, int Second);
```

10. Getting Time

Interface: GetTime

Function: Getting the current time of the module

Note: **The year must be plus** 2000 **when the date is displayed in the format of** YYYY/MM/DD**.**

Parameters: Year**: year;** Month**: month;** Day**: date;** Hour**: hour;** Minute**: minute;** Second**: second**

Returned value: 0**-false,** >0**-true**

int GetTime (int *Year, int *Month, int *Day, int *Hour, int *Minute, int *Second);

11. Getting System Status

Interface: GetStatus

Function: Getting system status.

Note**: The current module always responds as busy status.**

Parameters: None

Returned value: 0**-false,** >0**-true**

int GetStatus (void);

12. Getting Parameters

Interface: GetParameter

Function: Getting parameters from the module.

Parameters: flag**: Reads values based on the parameter ID;** value**: parameter value**

Parameter IDs:

0x36 - Save log

0x82 - Auto response when free scan

0x62 - Time out

0x6E - Firmware version,

0x71 – Baudrate 115200

0x73 - Current number of Enrolled fp,

0x74 - The available number of fp that can be enrolled

0x79 - Max user count

0x7C - Log number

0x7B- Maximum Log Count

0x89 - Build number

0x6D - Module id

Returned value: 0-false, >0-true

```
int GetParameter (int flag, int *value);
```

13. Setting Parameters

Interface: SetParameter

Function: Setting system parameters

Description: This command is used only to set parameters in the memory, but not to save the parameters. If parameters need to be saved, the [14]--->SaveParameter interface needs to be called.

Parameters: Please view the following list.

flag	value
0x36 – Save log	on-0x31, off-0x30
0x82 – Auto response when free scan	on-0x31, off-0x30
0x62 –Time out	0x31- 1 second 0x32- 2 seconds … Max seconds:255 seconds
0x50 – Working mode	0x30: Matching mode; 0x31: Reader(Picture) mode; 0x32: Reader(Template) mode
0x51 – Template format	0x30: ZK，0x32:Ansi 378，0x33:ISO 19794-2

Returned value: 0-false, >0-true

```
int SetParameter (int value, int flag);
```

14. Saving Parameters

Interface: SaveParameter

Function: Saving parameter values to a file

Parameters: None

Returned value: 0-false, >0-true

```
int SaveParameter (void);
```

15. Deleting All User Information

Interface: DeleteAllUsers

Function: When all users are deleted, all fingerprint templates are also deleted.

Parameters: None

Returned value: 0-false, >0-true

```
int DeleteAllUsers (void);
```

16. Deleting a User

Interface: DeleteUser

Function: Deleting a user based on a specified user ID and deleting the fingerprint templates of this user.

Note: **Before a user is deleted, the fingerprint template of this user is deleted. If the module powers off suddenly when a template is being deleted, the user may fail to be deleted or partial templates are deleted. In this case, the interface can be called again for deletion.**

Parameter: userID**: user ID**

Returned value: 0-false, >0-true

```
int DeleteUser (int userID);
```

17. Downloading All User Information

Interface: ReadAllUser.

Function: Downloading all user information from the module to the host.

Description: This interface needs to be called in cycles to read all user information. If user data are available, 1 is returned; otherwise, 0 is returned.

Note: **Only** userID**, name, and** fingerprintNum **among the returned parameters are valid; other returned results may appear in later versions.**

Parameters: userID**: user ID;**

name**: name;**

password**: password;**

secLevel**: User encryption levels 0 and 1;**

PIN2**: Secondary ID of a user;**

privilege**: permissions;**

Card**: Card number, used to store the number of the corresponding ID card;**

fingerprintNum**: Number of fingerprint templates enrolled for a user.**

Returned value: 0-false, >0-true

```
int ReadAllUser (int *userID, char *name, char *password,
unsigned short *secLevel, unsigned long *PIN2, unsigned char *privilege,
unsigned char *fingerprintNum, unsigned char *Card);
```

18. Getting Information of a User

Interface: GetUser

Function: Reading specified user information from the module.

Note: **Only** userID, name, **and** fingerprintNum **among the returned parameters are valid; other returned results may appear in later versions.**

Parameters: userID**: user ID;**

name: Name;

password: Password;

secLevel: User encryption levels 0 and 1;

PIN2: Secondary ID of a user;

privilege: Permissions;

Card: Card number, used to store the number of the corresponding ID card;

fingerprintNum: Number of fingerprint templates enrolled for a user.

Returned value: 0-false, >0-true

int GetUser (int userID, char *name, char *password,

unsigned short *secLevel, unsigned long *PIN2, unsigned char *privilege,

unsigned char *fingerprintNum, unsigned char *Card);

19. Adding and Modifying User Information

Interface: ModifyUser

Function: The module checks whether a user exists according to received user information. If the user exists, the module modifies the user information; if the user does not exist, the module adds the user.

Note: **Only** userID, name, **and** fingerprintNum **among the returned parameters are valid; other returned results may appear in later versions.**

Parameters: userID**: user ID;**

name: name;

password: password;

secLevel: User encryption levels 0 and 1;

PIN2: secondary ID of a user;

privilege: Permissions;

Card: Card number, used to store the number of the corresponding ID card;

fingerprintNum: Number of fingerprint templates enrolled for a user.

Returned value: 0-false, >0-true

int ModifyUser (int userID, char *name, char *password,

unsigned short secLevel, unsigned long PIN2, unsigned char privilege,

unsigned char fingerprintNum, char *Card);

20. Uploading Fingerprint Templates

Interface: SetTemplates

Function: Uploading fingerprint templates from the host to the module.

Parameters: userID: **user ID;** flag: **parameter ID;** data: **fingerprint data;** dataSize: **fingerprint data size.**

Parameter ID: 0: **None;**

1: **CHECK_FINGER (check whether the fingerprint exists.)**

Returned value: 0-**false,** >0-**true**

int SetTemplates (int userID, int flag, unsigned char *data, int dataSize);

21. Deleting Fingerprint Templates of a User

Interface: DeleteTemplates

Function: Deleting fingerprint templates according to a specified user ID and fingerprint index.

Parameters: userID: **user ID;** index: **fingerprint index (0-9);** flag: **parameter ID.**

Parameter ID: 0 – **Indicates all fingerprint templates of the user.**

1 –**(DELETE_ONLY_ONE), Deletes only the fingerprint template of a specified index.**

Returned value: 0-**false,** >0-**true**

int DeleteTemplates (int userID, int index, int flag);

22. Deleting All Fingerprint Templates

Interface: DeleteAllTemplates

Function: Deleting the fingerprint templates of all users.

Note: **The module always responds with success for this command.**

Parameters: None

Returned value: 0-**false,** >0-**true**

int DeleteAllTemplates (void);

23. Downloading Specified Fingerprint Templates

Interface: ReadTemplates

Function: Downloading fingerprint templates from the module to the host according to a specified user ID and fingerprint indexes.

Description: This interface needs to be called in cycles to read all fingerprint templates of a user. If the fingerprint template data of the user is available, 1 is returned; otherwise, 0 is returned.

Parameters: userID**: user ID;** index**: fingerprint index;** flag**: parameter ID;**

data**: Data of a template**

Parameter ID: 0 **– Ignores the fingerprint index;**

1 **– downloads fingerprint templates according to fingerprint indexes.**

Returned value: 0**-false,** >0**-true**

int ReadTemplates(int userID, int index, int flag, unsigned char *data);

24. Downloading All Fingerprint Templates

Interface: ReadAllTemplates

Function: Downloading all fingerprint templates from the module to the host.

Description: This interface needs to be called in cycles to read all fingerprint templates from the module. If fingerprint template data is available, 1 **is returned; otherwise,** 0 **is returned.**

Parameter: data**: data of a template**

Returned value: 0**-false,** >0**-true**

int ReadAllTemplates (unsigned char *data);

25. Verifying Fingerprint

Interface: Verify

Function: Waiting until a user presses a finger for 1:1 fingerprint template verification.

Description: When this interface is called, a user ID needs to be transmitted to the module. After receiving this command, the module waits until the user presses a finger for 1:1 fingerprint template verification.

Parameter: userID**: user ID**

Returned value: 0**-false, >**0**-true**

```
int Verify (int userID);
```

26. Deleting All Verified logs

Interface: DeleteAllLogs

Function: Deleting all verified logs.

Note**: The module always responds with success for this interface.**

Returned value: 0**-false, >**0**-true**

Parameters: None

```
int DeleteAllLogs (void);
```

27. Scanning a Fingerprint Template

Interface: ScanTemplate

Function: Scanning a fingerprint template.

Description: After this command is called, the module will check a fingerprint template from an inner buffer. If the inner buffer includes a template, the module returns the fingerprint template. If the inner buffer doesn't include a template, the module also return. The maximum length of a template is 2KB.

Note: When using the function, we need to set the module mode as "Matching" mode.

Parameter: data: Fingerprint template data.

Returned value: 0-false, >1-true

```
void ScanTemplate(unsigned char *data);
```

28. Resetting the Module

Interface: Reset

Function: Reset the module.

Returned value: 0-false, >0-true

Parameters: None

```
int Reset(void);
```

29. Disabling the module

Interface: DisableDevice

Function: The module doesn't send message of matching result to host after using the function.

Returned value: 0-false, >0-true

Parameters: None

```
int DisableDevice(void);
```

30. Enabling the Module

Interface: EnableDevice

Function: The module sends the message of matching result to host after the fingerprint is verified.

Returned value: 0-false, >0-true

Parameters: None

```
int EnableDevice(void);
```

31. Controlling the LED

Interface: SetParameter

Function: Control three kind of LED in module.

Returned value: 0-false, >0-true

Parameters: flag-0x31;value-Low 8 bits of word control the showing time and the unit is second. High 8 bits of word control which color LED to show. 0x80-Green LED,0xC0-Yellow LED, 0x40-Red LED

```
int SetParameter (int value, int flag);
```

32. Upgrading the firmware

Interface: Upgrade

Function: Upgrading the firmware version.

Returned value: 0-false, >0-true

Parameters: fw: Content of the firmware size: The size of the firmware.

```
int Upgrade (unsigned char *fw, int size);
```

33. Uploading Template Storage File

Interface: UploadTemplatesFile

Purpose: To upload the template storage file on the host to the module.

Parameter: fw: fingerprint template data; size: template data size

Returned value: 0 – false, >0 – true

```
int UploadTemplatesFile (unsigned char *fw, int size);
```

34. Uploading User Information Storage File

Interface: UploadUserFile

Purpose: To upload the user information storage file on the host to the module.

Parameter: fw: user information data; size: user information data size

Returned value: 0 – false, >0 – true

```
int UploadUserFile (unsigned char *fw, int size);
```

35. Downloading User Information Storage File

Interface: DownloadUserFile

Purpose: To download the user information storage file from the module to the host.

Parameter: None.

Returned value: 0 – false, >0 – true

```
int DownloadUserFile (void);
```

36. Downloading Template Storage File

Interface: DownloadTemplatesFile

Purpose: To download the fingerprint template storage file from the module to the host.

Parameter: None.

Returned value: 0 – false, >0 – true

```
int DownloadTemplatesFile (void);
```

37. Real-time Fingerprint Template Comparison

Interface: FreeScan

Purpose: To transfer data indicating whether comparison on the fingerprint sensor is successful to the host in real time.

Parameter: None.

Returned value: 0 – false, -1 – no, >0 – UserID

```
int FreeScan (void);
```

B. Notice

1. **The scope of the user ID:1~65535**

2. **When the mode of the module is configured as "Matching" mode, then if a user presses a finger on the fingerprint reader for comparison the module sends a message to the host.**

3. **You need to UploadUserFile first when UploadTemplatesFile.**

4. **Restart the module after updated the firmware to avoiding unexpected error.**

5. **under authentication mode, you need to receive the result at any time,it means int FreeScan (void). suggestion: build a thread to detecting the result. the communication will disconnect if you don't receive the command of int FreeScan (void)**

C. FAQ

1. How to obtain all users' fingerprint templates from Module A and upload them to Module B?

Answer: Method 1:

1) First call the interface DownloadTemplatesFile to download the registered fingerprint templates of all users to the host. .

2) And then call the interface UploadTemplatesFile to upload the fingerprint templates stored on the host to the module (all fingerprint templates within the module will be cleared before upload).

3) Restart the module (via soft reset or removal and insertion) to make the data take effect.

Method 2:

1) First call the interface ReadTemplates to download the fingerprint template of a designated user to the host.
2) Then call the interface SetTemplates and upload the downloaded fingerprint template of the designated user on the host to the module. This operation is completed.

Note: The interface DownloadTemplatesFile must be used with the interface UploadTemplateFile.

2. How to enroll a user via the fingerprint image?

Answer: First prepare the fingerprint image of the user, and then call the interface EnrollTemplateByImage for enrollment.

3.How to enroll a user?

Answer: Call the interface EnrollUserByScan, and then press a finger for three times based on the module indicator to complete the user enrollment via the fingerprint template.