星友科技
STARTEK

Programmer's Guide for FM220

Revision History

Rev. #	Date	Updated By	Description
1.0.0	2008/7/7	Startek Engineering, Inc	Initial
1.0.1	2008/7/8	Holing Hsu	1.Support new function FP_EnrollEx() 2.Change description of FP_CodeMatch() and FP_CodeMatchEx() support multiple fingerprint formats (include original pcode/fpcode, encoded ISO 19794-2 and encoded INCIT 378 formats).
1.0.2	2009/2/26	Holing Hsu	1. Modify function prototype FP_GetTemplate() 2. Add function FP_SaveISOMinutia(), FP_LoadSIOMinuita(), FP_SaveM1Minutia(), FP_LoadM1Minutia() 3.
1.0.3	2009/10/13	Holing Hsu	1.
1.0.4	2010/6/4	Holing Hsu	1.Add function FP_TemplateSelect()
1.0.5	2010/6/8	Holing Hsu	1. Add function FP_SaveWsqFile() 2. Add function FP_ImageQuality()
1.0.6	2011/1/15	Holing Hsu	1.Add function FP_SaveISOImage()
1.0.7	2011/12/20	Holing Hsu	Add function FP_GetISOimage()

1. Function List: *Please load the FM220API.dll and include the FM220API.h to use the API below*:

Device Control Functions	
FP_ConnectCaptureDriver	FP_DisconnectCaptureDriver
Device Diagnosis Functions	
FP_Diagnose	FP_CheckBlank
Fingerprint Capture and Feature Extraction Functions	
FP_CreateCaptureHandle	FP_Capture
FP_DestroyCaptureHandle	FP_Snap
FP_GetPrimaryCode	FP_GetTemplate
FP_ImageQuality	FP_GetISOImage
Fingerprint Image Access Functions	

FP_CreateImageHandle	FP_GetImage
FP_DestroyImageHandle	
Functions of Enrollment	
FP_CreateEnrollHandle	FP_Enroll
FP_DestroyEnrollHandle	FP_EnrollEx
FP_TemplateSelect	
Functions of Verification	
FP_ImageMatch	FP_ImageMatchEx
FP_CodeMatch	FP_CodeMatchEx
Supporting Functions	
FP_SaveImage	FP_Displayimage
FP_SaveISOMinutia	FP_LoadISOMinutia
FP_SaveM1Minutia	FP_LoadM1Minutia
FP_SaveWsqFile	FP_SaveISOImage

2. Funciton Description:

FP_ConnectCaptureDriver

Synopsis

HANDLE WINAPI FP_ConnectCaptureDriver(int reserved)

Description

The **FP_ConnectCaptureDriver()** connects the capture driver of the fingerprint device. Please connect the capture driver when your program is initialized, and disconnect the capture driver before terminating your program.

Parameter

reserved: this value is now not in use. Please let it be 0

Return Value

i. **Handle of the driver** : if the connection succeeds.
ii. **NULL :** if connection failed.

Remarks

This function must be called before the other API is used. Please disconnect the capture driver when program is finished.

FP_DisconnectCaptureDriver

Synopsis

void WINAPI FP_DisconnectCaptureDriver(HANDLE hConnect)

Parameter

hConnect: the handle returned by FP_ConnectCaptureDriver()

Description

The **FP_DisconnectCaptureDriver()** disconnects the capture driver of the fingerprint device.

Return Value

None.

FP_Diagnose

Synopsis

int WINAPI FP_Diagnose(HANDLE hConnect)

Description

Diagnose if the fingerprint device is OK.

Parameter

hConnect: The handle returned by **FP_ConnectCaptureDriver()**

Return Value

i. **OK** The fingerprint device is OK.
ii. **FAIL** There is problem with your fingerprint system.

Remarks

This function diagnoses your fingerprint device. Before testing, please clean the prism and **make sure that there is no finger on the reader**.

FP_CheckBlank

Synopsis

int WINAPI FP_CheckBlank(HANDLE hConnect)

Description

To check if there is any fingerprint on the reader.

Parameter

hConnect: The handle returned by **FP_ConnectCaptureDriver()**

Return Value

i. **OK** There is no fingerprint on the reader.
ii. **FAIL** There is a fingerprint on the reader.
iii. **S_MEM_ERR** Failed to allocate memory.

Remarks

This function is very useful in the enrollment process. To get stable and real features of a fingerprint during the enrollment, the user must remove his finger from the reader once a fingerprint has been snapped and put it down again on the reader after **FP_Enroll** has successfully been processed for this snapped fingerprint image. You can check if a fingerprint has actually been lifted off the reader by using this function.

FP_CreateCaptureHandle

Synopsis

HFPCAPTURE WINAPI FP_CreateCaptureHandle(HANDLE hConnect)

Description

Allocate memory and initialize the parameters for snapping.

Parameter

hConnect: The handle returned by **FP_ConnectCaptureDriver()**

Return Value

HFPCAPTURE	If succeeds, the return value is a handle to the allocated memory and the parameters for snapping.
NULL	Failed to create.

Remarks

This function allocates memory and initializes some parameters for snapping. This function must be called before using **FP_Capture** to snap a fingerprint image to main memory, otherwise the system will crash. When you want to leave the snap process, you **MUST** call **FP_DestroyCaptureHandle** to release the handle.

FP_Capture

Synopsis

HFPCAPTURE WINAPI FP_Capture(HANDLE hConnect, HFPCAPTURE hFPCapture)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
hFPCapture	A handle, which is created by **FP_CreateCaptureHandle()**, to the allocated memory and the parameters for snapping.

Description

To snap a fingerprint from the fingerprint device to the main memory by a fingerprint image quality control process. The fingerprint quality control cycle needs several frames of images and will continuously return the status of the fingerprint after each frame of image captured.

Return Value

OK	a valid fingerprint has successfully been snapped.
S_KEYCARD_CHECK_FAIL	**The driver** is found invalid.
fp_condition	Report the position and moisture condition of the fingerprint. The high-byte of fp_conditon is the moisture information, and the low-byte of fp_condition is the position information. You may use **U_DENSITY_CHECK_MASK** and **U_POSITION_CHECK_MASK** to check them by bit-and operation respectively. The action and its meanings is listed below :

(fp_condition & U_POSITION_CHECK_MASK)

U_POSITION_NO_FP	no fingerprint on the fingerprint reader.
U_POSITION_TOO_LOW	fingerprint is too low relative to the center of optical reader.
U_POSITION_TOO_TOP	fingerprint is too high relative to the optical reader.
U_POSITION_TOO_RIGHT	fingerprint is deviated to the right of the optical reader.
U_POSITION_TOO_LEFT	fingerprint is deviated to the left of the optical reader.
U_POSITION_TOO_LOW_RIGHT	fingerprint is deviated to the lower right of the optical reader.
U_POSITION_TOO_LOW_LEFT	fingerprint is deviated to the lower left of the optical reader.
U_POSITION_TOO_TOP_RIGHT	fingerprint is deviated to the upper right of the optical reader.
U_POSITION_TOO_TOP_LEFT	fingerprint is deviated to the upper left of the optical reader.

(fp_condition & U_DENSITY_CHECK_MASK)

U_DENSITY_TOO_DARK	fingerprint is too dark to identify
U_DENSITY_TOO_LIGHT	fingerprint is too light to identify
U_DENSITY_AMBIGUOUS	fingerprint can not be identified

Remarks

This function snaps a fingerprint image from the fingerprint device to the main memory. You should use a while loop to run this function and stop if a valid fingerprint has successfully been grabbed. This function must be used together with **FP_CreateCaptureHandle** and **FP_DestroyCaptureHandle()**.

The fingerprint device must be connected to the PC and 60K memory is required in run time

FP_DestroyCaptureHandle

Synopsis

int WINAPI FP_DestroyCaptureHandle(HANDLE hConnect, HFPCAPTURE hFPCapture)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
hFPCapture	A handle, which is created by **FP_CreateCaptureHandle**, to the allocated memory and the parameters for snapping.

Description

Release the handle of the allocated memory and the parameters for snapping.

Return Value

i. **OK** if succeeds.
ii. **FAIL** The hFPCapture is invalid.

Remarks

This function releases the handle created by **FP_CreateCaptureHandle()**. Make sure to call this function when you want to leave the snap process.

FP_Snap

Synopsis

int WINAPI FP_Snap(HANDLE hConnect)

Description

To snap a fingerprint from the fingerprint device to the main memory by fingerprint image quality control process. The fingerprint quality control cycle needs several frames of images to judge the quality of the fingerprint. This function will return status of the fingerprint after a cycle of quality judgment.

Return Value

OK	a valid fingerprint has successfully been snapped.
S_KEYCARD_CHECK_FAIL	**The driver** is found invalid.
fp_condition	Report the position and moisture condition of the fingerprint. The high-byte of fp_conditon is the moisture information, and the low-byte of fp_condition is the position information. You may use **U_DENSITY_CHECK_MASK** and **U_POSITION_CHECK_MASK** to check them by bit-and operation respectively. The action and its meanings is listed below :

(fp_condition & U_POSITION_CHECK_MASK)

U_POSITION_NO_FP	no fingerprint on the fingerprint reader.
U_POSITION_TOO_LOW	fingerprint is too low relative to the center of optical reader.
U_POSITION_TOO_TOP	fingerprint is too high relative to the optical reader.
U_POSITION_TOO_RIGHT	fingerprint is deviated to the right of the optical reader.
U_POSITION_TOO_LEFT	fingerprint is deviated to the left of the optical reader.
U_POSITION_TOO_LOW_RIGHT	fingerprint is deviated to the lower right of the optical reader.
U_POSITION_TOO_LOW_LEFT	fingerprint is deviated to the lower left of the optical reader.
U_POSITION_TOO_TOP_RIGHT	fingerprint is deviated to the upper right of the optical reader.
U_POSITION_TOO_TOP_LEFT	fingerprint is deviated to the upper left of the optical reader.

(fp_condition & U_DENSITY_CHECK_MASK)

U_DENSITY_TOO_DARK	fingerprint is too dark to identify
U_DENSITY_TOO_LIGHT	fingerprint is too light to identify
U_DENSITY_AMBIGUOUS	fingerprint can not be identified

Remarks

This function snaps a good-enough fingerprint image from the fingerprint device to the main memory. You should use a while loop to run this function and stop if a valid fingerprint has successfully been grabbed.

FP_GetPrimaryCode

Synopsis

int WINAPI FP_GetPrimaryCode(HANDLE hConnect, BYTE *p_code)

Parameter

HConnect	The handle returned by **FP_ConnectCaptureDriver()**
p_code	The primary code, which is the extracted minutia of the fingerprint from the image of main memory.

Description

This function converts the fingerprint image in main memory to a 256 bytes primary fingerprint code that can roughly represent the feature of a fingerprint.

Return Value

i. **OK** : input image has been processed successfully.
ii. **S_MEM_ERR** : insufficient memory for processing.
iii. **U_LEFT, U_RIGHT, U_UP, U_DOWN** : The process failed due to the bad positioning of the input image.
iv. **S_KEYCARD_CHECK_FAIL** : the driver is invalid.

Remarks

This function converts the fingerprint image in main memory to a 256 bytes primary fingerprint code that can roughly represent the feature of a fingerprint.

i. You should call **FP_Snap() or** first to snap a fingerprint to the main memory.
ii. You should allocate 256 bytes memory for p_code

FP_GetTemplate

Synopsis

int WINAPI FP_GetTemplate(HANDLE hConnect, BYTE *minu_code, int mode, int reserved)

Parameter

HConnect	The handle returned by **FP_ConnectCaptureDriver()**
minu_code	The minutiae code, which is the extracted minutia of the fingerprint from the image of main memory.
mode	Decide which file format for output 0: pcode(same as the output of FP_GetPrimaryCode()) 1: ISO/IEC 19794-2 format 2: ANSI/INCIT 378 format
reserved	Reserved, this value must set 0.

Description

This function converts the fingerprint image in main memory to a specified format of fingerprint code that can represent the feature of a fingerprint.

Return Value

i. **OK** : input image has been processed successfully.
ii. **S_MEM_ERR** : insufficient memory for processing.
iv. **S_KEYCARD_CHECK_FAIL** : the driver is invalid.
v. **other**

Remarks

This function converts the fingerprint image in main memory to a specified format of fingerprint code that can represent the feature of a fingerprint.

i. You should allocate 1024 bytes memory for minu_code

FP_GetImageQuality

Synopsis

int WINAPI FP_GetImageQuality(HANDLE hConnect)

Parameter

HConnect The handle returned by **FP_ConnectCaptureDriver()**

Description

This function reports current fingerprint image quality value in main memory. The quality value will from 1 one to 5. The value1 indicate the best quality and the value 5 indicate the worst quality.

Return Value

i. **1~5** : fingerprint image quality value.1 is the best quality. 5 is the worst quality
ii. **0** : get fingerprint image quality fail.

Remarks

This function converts the fingerprint image in main memory to a specified format of fingerprint code that can represent the feature of a fingerprint.

FP_GetISOImage

Synopsis

int WINAPI FP_GetISOimage(HANDLE hConnect,BYTE ImgCompAlgo,BYTE FpPos,BYTE *img_code)

Parameter

HConnect	The handle returned by **FP_ConnectCaptureDriver()**
ImgCompAlgo	Specify the compression algorithm. Currently only support no compression, set the value with “0”.
FpPos	Specify the finger position code.

Finger position codes

Unknown	0
Right thumb	1
Right index finger	2
Right middle finger	3
Right ring finger	4
Right little finger	5
Left thumb	6
Left index finger	7
Left middle finger	8
Left ring finger	9
Left little finger	10

img_code	The image code with format ISO 19794-4.

Description

This function converts the fingerprint image in main memory to ISO 19794-4 format.

Return Value

i. **OK** : input image has been processed successfully.
ii. **-1** : insufficient memory for processing.

Remarks

This function converts the fingerprint image in main memory to ISO 19794-4 format.

i. You should allocate 82990 bytes memory for **img_code**

FP_CreateImageHandle

Synopsis

HFPIMAGE WINAPI FP_CreateImageHandle(HANDLE hConnect, BYTE mode, WORD size)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
mode	**BIN_IMAGE** (Binary) or **GRAY_IMAGE** (gray)
size	**LARGE** (256x324) or **SMALL** (128x128) in size.

Description

To allocate memory and initialize the parameters for FP_IMAGE_SET defined in the include files with the specified mode and size.

Return Value

i. **HFPIMAGE** If succeeds, return a handle to the FP_IMAGE_SET.
ii. **NULL** Failed to create.

Remarks

This function allocates memory for the desired image. After calling this function, you can call **FP_GetImage()** to get a binary or gray fingerprint image from main memory depending on the specified mode. You may use the simple supporting functions **FP_DisplayImage()** to display the image or **FP_SaveImage()** to save the image as file.

You MUST remember to call **FP_DestroyImageHandle()** to release the system resources when the handle to FP_IMAGE_SET is no longer in use.

FP_GetImage

Synopsis

int WINAPI FP_GetImage(HANDLE hConnect, HFPIMAGE hFPImage)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
hFPImage	A handle, created by **FP_CreateImageHandle**, to FP_IMAGE_SET defined in the include files

Description

Load the fingerprint image from the main memory.

Return Value

i. **OK** : Get a fingerprint image successfully.
ii. **S_FP_INVALID** : The input handle is not valid or illegal.
iii. **S_MEM_ERR** : Unable to allocate memory while processing.
iv. **S_KEYCARD_CHECK_FAIL** : Driver is found invalid.

Remarks

This function gets a fingerprint image depending on the mode (binary or gray) and the size (large or small) specified in **FP_CreateImageHandle()**. You may use the simple support functions like **FP_DisplayImage()** to display the image or **FP_SaveImage()** to save the image as BMP or RAW files.

Please note:

i. You should first snap a fingerprint to the main memory.
ii. You should call **FP_CreateImageHandle()** to get a handle to the image set.
iii. You should call **FP_DestroyImageHandle()** to release the handle when **FP_GetImage()** is no longer in use.

FP_DestroyImageHandle

Synopsis

int WINAPI FP_DestroyImageHandle(HANDLE hConnect ,HFPIMAGE hFPImage)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
hFPImage	A handle created by **FP_CreateImageHandle().**

Description

To release the handle, which is created by **FP_CreateImageHandle(),** to FP_IMAGE_SET defined in the include files.

Return Value

i. **OK** The handle is released successfully.
ii. **S_FP_INVALID** The handle is invalid.

Remarks

Release the handle created by **FP_CreateImageHandle**. The function **MUST** be called when **FP_GetImage()** is no longer in use.

FP_CreateEnrollHandle

Synopsis

HFPENROLL WINAPI FP_CreateEnrollHandle(HANDLE hConnect, BYTE mode)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
mode	A reserved value, it must be set as DEFAULT_MODE.

Description

To allocate memory and initialize the parameters for FP_ENRL_SET defined in the include files with the specified mode.

Return Value

i. **HFPENROLL** If succeeds, a handle to FP_ENRL_SET will be returned.
ii. **NULL** Failed to create **HFPENROLL**.

Remarks

This function creates and initializes a handle for the enrollment set. Only after this function has been called, you can start the enrollment process by calling **FP_Enroll(),** otherwise the system will crash .Make sure to call **FP_DestroyEnrollHandle()** to free the system resources when you want to leave the enrollment process.

FP_Enroll

Synopsis

int WINAPI FP_Enroll(HANDLE hConnect, HFPENROLL hFPEnroll,BYTE *p_code ,BYTE *fp_code)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
hFPFnroll	A handle, created by **FP_CreateEnrollHandle(),** to FP_ENRL_SET defined in the include files
p_code	A primary code of 256 bytes derived from **FP_GetPrimaryCode().**
fp_code	The final fingerprint code to represent the feature of a fingerprint if the enrollment is successful.

Description

Generate a final fingerprint code of 256 bytes.

Return Value

i. **U_CLASS_A, U_CLASS_B, U_CLASS_C, U_CLASS_D, U_CLASS_E** : The quality of enrolled fingerprint.
ii. **U_NOT_YET** : Enrollment is not completed yet.
iii. **S_FP_INVALID** : The p_code is invalid.
iv. **S_FPSET_INVALID** : Invalid input handle.

Remarks

This function generates the final fingerprint code ***fp_code*** from several input ***p_code*** by collecting their common features. The purpose of enrollment is to get enough stable characteristics to represent the corresponding fingerprint.

More explanations will be needed for the working process of enrollment. At first, you must call **FP_CreateEnrollHandle()** to create a pointer to FP_ENRL_SET. Then you should call **FP_DestroyEnrollHandle()** to release the system resource. Basically, the kernel process of enrollment works in a continuous loop as following:

1. Input a ***p_code***, which is generated by calling **FP_Snap()** or **()** and .
2. Send the input ***p_code*** to ***hFPEnroll*** by calling **FP_Enroll()**
3. If the return value is not one of the qualities defined, repeat step 1 and step 2 until the ***quality*** of the fingerprint is derived.

4. If you have tried more than 6 times and still cannot derive the ***quality*** of the finger, it means that the finger you have chosen to enroll may not be good enough. You should change to another finger and restart the enrollment.
5. If you want to improve the enrolled quality, you can continue executing from step 1 to step 3 to get a better final fingerprint code with better ***quality***. Of course, it depends on the requirements of your application.
6. If you have tried to ***enhance*** the enrolled quality more than 3 times but the ***quality*** still remains in a certain class without any improvement, it seems that the enrolled ***quality*** has been stable. Any attempt to enhancement may be in vain. You should stop the enrollment with the stable enrolled ***quality***. If you are not satisfied with the current enrolled ***quality***, choose another finger and restart the enrollment.

FP_EnrollEx

Synopsis

int WINAPI FP_EnrollEx(HANDLE hConnect,HFPENROLL hEnrlSet,BYTE *minu_code1, BYTE *minu_code2,int mode)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
hFPFnroll	A handle, created by **FP_CreateEnrollHandle(),** to FP_ENRL_SET defined in the include files
minu_code1	The input fingerprint code.
minu_code2	The final fingerprint code to represent the feature of a fingerprint if the enrollment is successful.
mode	This flag decide output format of enrolled fingerprint template. 0: pcode(same as the output of FP_Enroll()) 1: encoded ISO/IEC 19794-2 format(same as the output of FP_ISOMinutiaEnroll()) 2: encoded M1 ANSI/INCIT 378 format

Description

Generate a final fingerprint code of 256 bytes.

Return Value

i. **U_CLASS_A, U_CLASS_B, U_CLASS_C, U_CLASS_D, U_CLASS_E** : The quality of enrolled fingerprint.
ii. **U_NOT_YET** : Enrollment is not completed yet.
iii. **S_FP_INVALID** : The p_code is invalid.
iv. **S_FPSET_INVALID** : Invalid input handle.

Remarks

This function generates the final enrolled fingerprint code ***minu_code2*** from several input ***minu_code1*** by collecting their common features. The purpose of enrollment is to get enough stable characteristics to represent the corresponding fingerprint.

More explanations will be needed for the working process of enrollment. At first, you must call **FP_CreateEnrollHandle()** to create a pointer to FP_ENRL_SET. Then you should call **FP_DestroyEnrollHandle()** to release the system resource. Basically, the kernel process of enrollment works in a continuous loop as following:

1. Input a ***minu_code1***, which is generated by calling **FP_GetTemplate()**.
2. Send the input ***minu_code1*** to ***hFPEnroll*** by calling **FP_EnrollEx()**
3. If the return value is not one of the qualities defined, repeat step 1 and step 2 until the ***quality*** of the fingerprint is derived.

4. If you have tried more than 6 times and still cannot derive the ***quality*** of the finger, it means that the finger you have chosen to enroll may not be good enough. You should change to another finger and restart the enrollment.
5. If you want to improve the enrolled quality, you can continue executing from step 1 to step 3 to get a better final fingerprint code with better ***quality***. Of course, it depends on the requirements of your application.
6. If you have tried to ***enhance*** the enrolled quality more than 3 times but the ***quality*** still remains in a certain class without any improvement, it seems that the enrolled ***quality*** has been stable. Any attempt to enhancement may be in vain. You should stop the enrollment with the stable enrolled ***quality***. If you are not satisfied with the current enrolled ***quality***, choose another finger and restart the enrollment.

FP_ DestroyEnrollHandle

Synopsis

int WINAPI FP_DestroyEnrollHandle (HANDLE hConnect, HFPENROLL hFPEnroll)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver().**
HFPEnroll	A handle, which is created by **FP_CreateEnrollHandle(),** to FP_ENRL_SET defined in the include files.

Description

To release the handle, which is created by **FP_CreateEnrollHandle(),** to FP_ENRL_SET defined in the include files.

Return Value

i. **OK** Handle is released successfully.
ii. **S_FPSET_INVALID** Input handle is invalid.

Remarks

This function releases the handle created by **FP_CreateEnrollHandle()**. Call this function only if **FP_Enroll()** is no longer in use.

FP_TemplateSelect

Synopsis

int WINAPI FP_TemplateSelect(HANDLE hConnect,int fp_cnt,BYTE *fp1,BYTE *fp2,BYTE *fp3,BYTE *fp4,BYTE *fp5)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver().**
fp_cnt	The number of fingerprint template. The maximum value is 5
fp1	The 1st input fingerprint code
fp2	The 2nd input fingerprint code
fp3	The 3rd input fingerprint code
fp4	The 4th input fingerprint code
fp5	The 5th input fingerprint code

Description

This function selects one of the best quality fingerprint template through cross validation. The return value will be 0~4 indicate fp1~fp5.

Return Value

i. **0~4** The best fingerprint template value
ii. **Negative value** Template selection fail

Remarks

FP_ImageMatch

Synopsis

int WINAPI FP_ImageMatch(HANDLE hConnect, BYTE *fp_code, WORD security)

Parameter

HConnect	The handle returned by **FP_ConnectCaptureDriver()**
fp_code	The final fingerprint code generated through the enrollment.
Security	A parameter to set the threshold that determines whether the verification can be passed. The 8 security levels with **False Acceptance Rate (FAR)** are ranged from the strictest 1/300,000(**Security A**) to the loosest 1/100(**Security E**).

SECURITY_A	Verification passes as long as the minutiae matching score is over the threshold. The FAR of security A is 1/200,000.
SECURITY_B	The FAR of security B is 1/100,000.
SECURITY_C	The FAR of security C is 1/50,000.
SECURITY_D	The FAR of security D is 1/10,000.
SECURITY_E	The FAR of security E is 1/2,000.
AUTO_SECURITY	Automatically changes the verification security according to the enrolled quality, this is the default security.

Enrolled Quality	Auto-Security
A	C
B	D
C	E
D	E
E	R

Description

To match the fingerprint image in main memory with the final fingerprint code which is generated through enrollment.

Return Value

i. **OK** : The verification of fingerprint image with final fingerprint code meets the requirement of ***security*.**
ii. **FAIL** : The fingerprint image is not identical with the final fingerprint code on the required security.
iii. **S_MEM_ERR :** Insufficient memory for image processing.
iv. **S_FPCODE_INVALID :** the input ***fp_code*** is illegal.
v. **S_SECURITY_ERR :** improper security level setting.

Remarks

This function verifies the fingerprint image in the main memory with the final fingerprint code generated through the enrollment. The argument ***security*** sets the threshold that determines whether this verification can be passed. To call this function, you should first snap a fingerprint to the main memory, and input the fp_code generated by **FP_Enroll()**, but not by **FP_GetPrimaryCode()** only. Please note that ***p_code*** and ***fp_code*** are quite different in their content and cannot be misplaced.

FP_ImageMatchEx

Synopsis

int WINAPI FP_ImageMatchEx(HANDLE hConnect, BYTE *fp_code, WORD security, DWORD *score)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
fp_code	The final fingerprint code generated through the enrollment.
security	A parameter to set the threshold that determines whether the verification can be passed. See **FP_ImageMatch()** for more details.
score	The similarity of two fingerprints to be compared. The higher score means a higher similarity.

Description

Verify the fingerprint image in the main memory with the final enrolled fingerprint code.

Return Value

i. **OK** : The verification of fingerprint image with final fingerprint code meets the requirement of ***security*.**
ii. **FAIL** : The fingerprint image is not identical with the final fingerprint code on the required security.
iii. **S_MEM_ERR :** insufficient memory for image processing.
iv. **S_FPCODE_INVALID :** The input ***fp_code*** is illegal.
v. **S_SECURITY_ERR :** Improper security level setting.

Remarks

This function verifies the fingerprint image in main memory with the final fingerprint code generated through the enrollment. The argument ***security*** sets the threshold that determines whether this verification can be passed. To call this function, you should first snap a fingerprint to the main memory, and input the fp_code generated by **FP_Enroll()**, but not by **FP_GetPrimaryCode()** only. Please note that ***p_code*** and ***fp_code*** are quite different in their content and cannot be misplaced.

FP_CodeMatch

Synopsis

int WINAPI FP_CodeMatch(HANDLE hConnect, BYTE *p_code, BYTE *fp_code, WORD security)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
p_code	The fingerprint code generated by using **FP_GetPrimaryCode().**
fp_code	The final fingerprint code generated through the enrollment.
Security	A parameter to set the threshold that determines whether the verification can be passed. See **FP_ImageMatch()** for details.

Description

Match the input fingerprint code with an existing enrolled fingerprint code.

Return Value

i. **OK** : The verification of fingerprint image with final fingerprint code meets the requirement of ***security*.**
ii. **FAIL** : The fingerprint image is not identical with the final fingerprint code on the required security.
iii. **S_MEM_ERR :** Insufficient memory for image processing.
iv. **S_FPCODE_INVALID :** The input ***fp_code*** is illegal.
v. **S_SECURITY_ERR :** Improper security level setting.

Remarks

This function verifies the p_code generated by **FP_GetPrimaryCode()** with the final fingerprint code ***fp_code*** generated through the enrollment. The argument ***security*** sets the threshold that determines whether this verification can be passed. Please note that ***p_code*** and ***fp_code*** are quite different in their content and cannot be misplaced. After version 1.6.5.3, this function can also support encoded ISO/IEC 19794-2 or encoded ANSI/INCIT 378 format generated by FP_GetTemplate() as input.

FP_CodeMatchEx

Synopsis

int WINAPI FP_CodeMatchEx(HANDLE hConnect, BYTE *p_code, BYTE *fp_code, WORD security , DWORD *score)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
p_code	The fingerprint code generated through **FP_GetPrimaryCode().**
fp_code	The final fingerprint code generated through the enrollment.
security	A parameter to set the threshold that determines where the verification can be passed. See **FP_ImageMatch()** for details.
score	The similarity of two fingerprints to be compared. The higher score means a higher similarity.

Description

To verify two fingerprint code, while one is generated through **()** and the other through the enrollment.

Return Value

i. **OK** : The verification of fingerprint image with final fingerprint code meets the requirement of ***security*.**
ii. **FAIL** : The fingerprint image is not identical with the final fingerprint code on the required security.
iii. **S_MEM_ERR :** Insufficient memory for image processing.
iv. **S_FPCODE_INVALID :** The input ****fp_code*** is illegal.
v. **S_SECURITY_ERR :** Improper security level setting.

Remarks

This function verifies the p_code generated by **FP_GetPrimaryCode()** with the final fingerprint code ***fp_code*** generated through the enrollment . The argument ***security*** sets the threshold that determines whether this verification can be passed. Please note that ***p_code*** and ***fp_code*** are quite different in their content, they cannot be misplaced. After version 1.6.5.3, this function can also support encoded ISO/IEC 19794-2 or encoded ANSI/INCIT 378 format generated by FP_GetTemplate() as input.

FP_SaveImage

Synopsis

int WINAPI FP_SaveImage(HANDLE hConnect, HFPIMAGE hFPImage, WORD FileType, char* Filename)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
hFPImage	The handle to the FP_IMAGE_SET defined in the include files.
FileType	The image can be saved as a bitmap (**BMP**) file or a raw (**RAW**) file.
Filename	The filename to be saved as.

Description

Save the fingerprint image as a BMP or RAW file.

Return Value

i. **OK** : the image is saved successfully.
ii. **S_FILE_ERR** : failed to open the file.
iii. **S_FP_INVALID** : input hFPImage is invalid.
v. **S_MEM_ERR** : failed to allocate memory.

Remarks

This function saves the image as a BMP or RAW file with the specified filename. The size and mode of the image are determined while calling **FP_CreateImageHandle()**.

You must call **FP_CreateImageHandle()** and **FP_GetImage()** first to derive a 'legal' and 'valid' image handle and do remember to call **FP_DestroyImageHandle()** to free the handle.

FP_DisplayImage

Synopsis

int WINAPI FP_DisplayImage(HANDLE hConnect , HDC hDC, HFPIMAGE hFPImage,int nStartX, int nStartY , int nDestWidth, int nDestHeight)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
hDC	Identifies the device context.
HFPImage	The handle to the FP_IMAGE_SET defined in include files
nStartX, nStartY	The start position of the image to be displayed
nDestWidth, nDestHeight	The size of the image to be displayed

Description

Display the image handle on a device context.

Return Value

i. **OK** If succeeds
ii. **FAIL** Otherwise.

Remarks

The function displays the image handle on a device context. To call this function, you should first create a handle to the FP_IMAGE_SET using **FP_CreateImageHandle()** and then call **FP_GetImage()** to load the image from the main memory. Finally **DONOT** forget to call **FP_DestroyImageHandle()** to release the image handle.

FP_SaveISOMinutia

Synopsis

int WINAPI FP_SaveISOminutia(HANDLE hConnect, char*Filename, BYTE *minu_code)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
Filename	The filename to be saved as.
minu_code	The fingerprint template with ISO 19794-2 format.

Description

Save the fingerprint template as ISO 19794-2 format.

Return Value

i. **OK** If succeeds
ii. **FAIL** Otherwise.

Remarks

FP_LoadISOminutia

Synopsis

int WINAPI FP_LoadISOminutia (HANDLE hConnect, char*Filename, BYTE *minu_code)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
Filename	The filename to be loaded from.
minu_code	The fingerprint template with ISO 19794-2 format.

Description

Load the fingerprint template as ISO 19794-2 format.

Return Value

i. **OK** If succeeds
ii. **FAIL** Otherwise.

Remarks

FP_SaveM1Minutia

Synopsis

int WINAPI FP_SaveM1minutia(HANDLE hConnect, char*Filename, BYTE *minu_code)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
Filename	The filename to be saved as.
minu_code	The fingerprint template with ANSI 378 format.

Description

Save the fingerprint template as ANSI 378 format.

Return Value

i. **OK** If succeeds
ii. **FAIL** Otherwise.

Remarks

FP_LoadM1minutia

Synopsis

int WINAPI FP_LoadM1minutia (HANDLE hConnect, char*Filename, BYTE *minu_code)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
Filename	The filename to be loaded from.
minu_code	The fingerprint template with ANSI 378 format.

Description

Load the fingerprint template as ANSI 378 format.

Return Value

i. **OK** If succeeds
ii. **FAIL** Otherwise.

Remarks

FP_SaveWsqFile

Synopsis

int WINAPI FP_SaveWsqFile(HANDLE hConnect, char *filename,int comp_ratio)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
Filename	The filename to be saved as.
comp_ratio	The WSQ compress ratio. 0: about 5:1 compression 1: about 15:1 compression

Description

Save the fingerprint image as WSQ format with specified filename.

Return Value

i. **OK** If succeeds
ii. **FAIL** Otherwise.

Remarks

FP_SaveISOImage

Synopsis

int WINAPI FP_SaveISOImage(HANDLE hConnect,HFPIMAGE hFPImage ,BYTE FileFormat ,char*Filename,BYTE comp_ratio,BYTE FpPos)

Parameter

hConnect	The handle returned by **FP_ConnectCaptureDriver()**
HFPImage	The handle to the FP_IMAGE_SET defined in include files
FileFormat	Th file format 0: ISO 19794-4 format
Filename	The filename to be saved as.
comp_ratio	The WSQ compress ratio. 0: no compression support only
FpPos	Finger position codes.

Finger position codes

Unknown	0
Right thumb	1
Right index finger	2
Right middle finger	3
Right ring finger	4
Right little finger	5
Left thumb	6
Left index finger	7
Left middle finger	8
Left ring finger	9
Left little finger	10

Description

Save the fingerprint image as ISO 19794-4 format with specified filename.

Return Value

i. **OK** If succeeds
ii. **FAIL** Otherwise.

Remarks