UniFinger Engine SFR300 SDK Reference Manual

Version 2.5

SUPREMA

Contents

SF_Initialize

The **SF_Initialize** function initializes the SFR300 scanner or SFR200 scanner.

int SF_Initialize(void)

Parameters

This function has no parameters.

Return Values

Nonzero if the initialization succeeds; otherwise 0.

SF_Uninitialize

The **SF_Uninitialize** function uninitialize the SFR300 scanner or SFR200 scanner.

void SF_Uninitialize(void)

Parameters

This function has no parameters.

Return Values

This function has no return values.

SF_SetFastMode

The **SF_SetFatMode** function is to enable or disable the fast mode. The fast mode is used to accelerate the speed of 1:N matching in a little sacrifice of recognition accuracy.

int SF_SetFastMode(int Mode)

Parameters

Mode

SF_NORMAL_MODE : set normal mode (default mode)

SF_FAST_MODE : set fast mode

Return Values

If error is occurred, return -1.

SF_GetDeviceNumber

The **SF_GetDeviceNumber** function returns the number of SFR300 scanner or SFR200 scanner attached to the system.

int SF_GetDeviceNumber(void)

Parameters

This function has no parameters.

Return Values

Number of SFR300 scanner or SFR200 scanner attached to the system.

SF_GetDevice

The **SF_GetDevice** function gets index of SFR300 scanner or SFR200 scanner which is currently selected.

int SF_GetDevice(void)

Parameters

This function has no parameters.

Return Values

Zero-based index of SFR300 scanner or SFR200 scanner which is currently selected.

SF_SetDevice

The **SF_SetDevice** function selects SFR300 scanner or SFR200 scanner.

int SF_SetDevice(int Device)

Parameters

Device

Index of SFR300 scanner or SFR200 scanner being selected. It can be from 0 to number of SFR300 scanner or SFR200 scanner – 1.

Return Values

Nonzero if selection of SFR300 scanner or SFR200 scanner succeeds; otherwise 0.

SF_GetSerial

The **SF_GetSerial** function gets serial number of SFR300 scanner or SFR200 scanner which is currently selected.

int SF_GetSerial(unsigned char* Serial)

Parameters

Serial

32 bytes-long buffer where serial number is returned.

Return Values

Nonzero if serial number is read successfully; otherwise 0.

SF_GetTimeout

The **SF_GetTimeout** function gets timeout for SFR300 scanner or SFR200 scanner to wait finger.

int SF_GetTimeout(void)

Parameters

This function has no parameters.

Return Values

Timeout value in milli-second

SF_SetTimeout

The **SF_SetTimeout** function sets timeout for SFR300 scanner or SFR200 scanner to wait finger.

int SF_SetTimeout(int Timeout)

Parameters

Timeout

Timeout value in milli-second. In case timeout is not needed, it can be 0.

Return Values

Nonzero if setting of timeout succeeds; otherwise 0.

SF_GetBrightness

The **SF_GetBrightness** function gets brightness for SFR300 scanner or SFR200 scanner device.

int SF_GetTimeout(void)

Parameters

This function has no parameters.

Return Values

Brightness value which is from 0 to 255. Higher value means darker image. Default value is 100.

SF_SetBrightness

The **SF_SetBrightness** function sets brightness for SFR300 scanner or SFR200 scanner.

int SF_SetBritheness(int Brightness)

Parameters

Brightness

Brightness value which is from 0 to 255. Higher value means darker image. Default value is 100.

Return Values

Nonzero if setting of brightness succeeds; otherwise 0.

SF_GetSensitivity

The **SF_GetSensitivity** function gets sensitivity for SFR300 scanner or SFR200 scanner.

int SF_GetSensitivity(void)

Parameters

This function has no parameters.

Return Values

Sensitivity value which is from 0 to 7. Higher value means more sensitive. Default value is 4.

SF_SetSensitivity

The **SF_SetSensitivity** function sets sensitivity for SFR300 scanner or SFR200 scanner.

int SF_SetSensitivity(int Sensitivity)

Parameters

Sensitivity

Sensitivity value which is from 0 to 7. Higher value means more sensitive. Default value is 4.

Return Values

Nonzero if setting of sensitivity succeeds; otherwise 0.

SF_IsSensorOn

The **SF_IsSensorOn** function determines whether a SFR300 scanner or SFR200 scanner is attached or not.

int SF_IsSensorOn(void)

Parameters

This function has no parameters.

Return Values

Nonzero if a SFR300 scanner or SFR200 scanner is attached; otherwise 0.

SF_IsFingerOn

The **SF_IsFingerOn** function determines whether a finger is on the SFR300 scanner or SFR200 scanner or not.

int SF_IsFingerOn(void)

Parameters

This function has no parameters.

Return Values

Nonzero if a finger is on the SFR300 scanner or SFR200 scanner; otherwise 0.

SF_Capture

The **SF_Capture** function acquires an image from the SFR300 scanner or SFR200 scanner.

int SF_Capture(void)

Parameters

This function has no parameters.

Return Values

Nonzero if the acquisition of an image succeeds; otherwise 0.

SF_AbortCapturing

The **SF_AbortCapturing** function aborts SFR300 scanner or SFR200 scanner to wait for finger.

void SF_AbortCapturing(void)

Parameters

This function has no parameters.

Return Values

This function has no return values.

SF_GetImageWidth

The **SF_GetImageWidth** function returns width of fingerprint image.

int SF_GetImageWidth(void)

Parameters

This function has no parameters.

Return Values

Width of fingerprint image.

Remarks

Before the **SF_GetImageWidth** function is called, the **SF_Capture** function should be called to acquire a fingerprint image.

SF_GetImageHeight

The **SF_GetImageHeight** function returns height of fingerprint image.

int SF_GetImageHeight(void)

Parameters

This function has no parameters.

Return Values

Height of fingerprint image.

Remarks

Before the **SF_GetImageHeight** function is called, the **SF_Capture** function should be called to acquire a fingerprint image.

SF_GetImageBuffer

The **SF_GetImageBuffer** function returns a pointer to the buffer for fingerprint image.

unsigned char* SF_GetImageBuffer(void)

Parameters

This function has no parameters.

Return Values

A pointer to the buffer for fingerprint image.

Remarks

Before the **SF_GetImageBuffer** function is called, the **SF_Capture** function should be called to acquire a fingerprint image.

Example

```
int Width, Height;
unsigned char* Buffer;
if ( !SF_Capture() ) {
        return; // Error: The acquisition of an image fails.
}
Width = SF_GetImageWidth();
Height = SF_GetImageHeight();
Buffer = ( unsigned char* ) malloc( Width * Height );
memcpy( Buffer, SF_GetImageBuffer(), Width * Height );
```

SF_Clear

The **SF_Clear** function clears the fingerprint image.

void SF_Clear(void)

Parameters

This function has no parameters.

Return Values

This function has no return values.

SF_Draw

The **SF_Draw** function draws the fingerprint image which is acquired using the **SF_Capture** function.

void SF_Draw(HWND hWnd, int l, int t, int r, int b, int bCore)

Parameters

hWnd

Handle to the window where the fingerprint image is drawn.

l

Specifies the logical x-coordinate of the upper-left corner of the rectangle.

t

Specifies the logical y-coordinate of the upper-left corner of the rectangle.

r

Specifies the logical x-coordinate of the lower-right corner of the rectangle.

b

Specifies the logical y-coordinate of the lower-right corner of the rectangle.

bCore

Specifies whether the core of fingerprint is drawn of not.

Return Values

This function has no return values.

Remarks

The core of fingerprint can be drawn in proper position, only after the **SF_Enroll**, **SF_EnrollWithVerify**, **SF_Verify**, or **SF_Identify** function is called.

SF_Enroll

The **SF_Enroll** function yields the template of fingerprint from the fingerprint image which is acquired using the **SF_Capture** function.

int SF_Enroll(unsigned char* Template, int* TemplateSize, int bCoreDetect)

Parameters

Template

A pointer to the buffer that stores the template of fingerprint. SF_TEMPLATESIZE bytes should be allocated to this buffer at least.

TemplateSize

A pointer to the integer variable that contains the size of template. If the exact size of template is not needed, *TemplateSize* can be NULL.

bCoreDetect

Specifies whether the core of fingerprint is detected for enrollment. If *bCoreDetect* is TRUE, the enrollment succeeds only when the core of fingerprint is detected in the center of SFR300 scanner or SFR200 scanner. If *bCoreDetect* is FALSE, the enrollment succeeds as long as the fingerprint image is good enough.

Return Values

SF_SUCCESS

The enrollment succeeds.

SF_NOTGOODIMAGE

The fingerprint image is not good enough for enrollment.

The following return values are available only if *bCoreDetect* is TRUE.

SF_CORETOCENTER

The core of fingerprint is not detected.

SF_CORETOLEFT

SF_CORETORIGHT

SF_CORETOTOP

SF_CORETOBOTTOM

The core of fingerprint is detected, but should be placed in the center of SFR300 scanner or

SFR200 scanner. The return values can be two combinations of these 4 values.

Remarks

Before the **SF_Enroll** function is called, the **SF_Capture** function should be called to acquire a fingerprint image to be enrolled.

Example

```
int ret;
unsigned char m_EnrollTemplate[ SF_TEMPLATESIZE ];
while ( 1 ) {
        if ( !SF_Capture() ) {
                return; // Error: The acquisition of an image fails.
        }
        ret = SF_Enroll( m_EnrollTemplate, NULL, TRUE );
        if ( ret == SF_SUCCESS ) {
                break; // Enrollment succeeds.
        }
        if ( ret == SF_NOTGOODIMAGE ) {
                // Warning: The fingerprint image is not good enough for enrollment.
        }
        else {
                // Warning: The core of fingerprint should be placed in the center of SFR300 scanner or
SFR200 scanner.
        }
}
```

SF_EnrollWithVerify

The **SF_EnrollWithVerify** function yields the template of fingerprint using both the fingerprint image which is acquired using the **SF_Capture** function and the template of fingerprint enrolled formerly using the **SF_Enroll** function.

int SF_EnrollWithVerify(unsigned char* Template, int* TemplateSize, int bCoreDetect)

Parameters

Template

A pointer to the buffer that stores the template of fingerprint enrolled formerly. The new template of fingerprint is also stored in this buffer. SF_TEMPLATESIZE bytes should be allocated to this buffer at least.

TemplateSize

A pointer to the integer variable that contains the size of template. If the exact size of template is not needed, *TemplateSize* can be NULL.

bCoreDetect

Specifies whether the core of fingerprint is detected for enrollment. If *bCoreDetect* is TRUE, the enrollment succeeds only when the core of fingerprint is detected in the center of SFR300 scanner or SFR200 scanner. If *bCoreDetect* is FALSE, the enrollment succeeds as long as the fingerprint image is good enough.

Return Values

SF_SUCCESS

The enrollment succeeds.

SF_VERIFYFAIL

Two fingerprints are not matched with each other.

SF_NOTGOODIMAGE

The fingerprint image is not good enough for enrollment.

The following return values are available only if *bCoreDetect* is TRUE.

SF_CORETOCENTER

The core of fingerprint is not detected.

SF_CORETOLEFT

SF_CORETORIGHT
SF_CORETOTOP
SF_CORETOBOTTOM

The core of fingerprint is detected, but should be placed in the center of SFR300 scanner or SFR200 scanner. The return values can be two combinations of these 4 values.

Remarks

This function can be used to improve the quality of an enrolled fingerprint. It is recommended that two fingerprints are acquired for enrollment. A finger should be removed on the SFR300 scanner or SFR200 scanner in between two acquisitions. If these two fingerprints match with each other, better fingerprint is finally enrolled.
Before the **SF_EnrollWithVerify** function is called, the **SF_Capture** function should be called to acquire a fingerprint image to be enrolled.

Example

```
int ret;
unsigned char m_EnrollTemplate[ SF_TEMPLATESIZE ];
while ( 1 ) {
    if ( !SF_Capture() ) {
        return; // Error: The acquisition of an image fails.
    }
    ret = SF_Enroll( m_EnrollTemplate, NULL, TRUE );
    if ( ret == SF_SUCCESS ) {
        break; // 1st enrollment succeeds.
    }
    if ( ret == SF_NOTGOODIMAGE ) {
        // Warning: The fingerprint image is not good enough for enrollment.
    }
    else {
        // Warning: The core of fingerprint should be placed in the center of SFR300 scanner or
SFR200 scanner.
    }
}
while ( SF_IsFingerOn() );// Wait until a finger is removed on the SFR300 scanner or SFR200 scanner
while ( 1 ) {
    if ( !SF_Capture() ) {
        return; // Error: The acquisition of an image fails.
    }
    ret = SF_EnrollWithVerify( m_EnrollTemplate, NULL, TRUE );
```

```
	if ( ret == SF_SUCCESS ) {
		break; // Enrollment succeeds.
	}
	if ( ret == SF_VERIFYFAIL ) {
		return; // Error: Two fingerprints are not matched with each other.
	}
	if ( ret == SF_NOTGOODIMAGE ) {
		// Warning: The fingerprint image is not good enough for enrollment.
	}
	else {
		// Warning: The core of fingerprint should be placed in the center of SFR300 scanner or
SFR200 scanner.
	}
}
```

SF_Verify

The **SF_Verify** function verifies the fingerprint image acquired using the **SF_Capture** function with the template of fingerprint enrolled formerly.

int SF_Verify(unsigned char* Template, int SecurityLevel, int bCoreDetect)

Parameters

Template

A pointer to the buffer that stores the template of fingerprint enrolled formerly.

SecurityLevel

Specifies the level of security. This is in the range 1 to 7. If *SecurityLevel* is 7, the most secure verification is guaranteed. The value 4 is suitable in general case. Physical meaning of *SecurityLevel* is as follows.

SecurityLevel	False Accept Rate (FAR)
1	Below 1 % (1e-2)
2	Below 0.1 % (1e-3)
3	Below 0.01 % (1e-4)
4	Below 0.001 % (1e-5)
5	Below 0.0001 % (1e-6)
6	Below 0.00001 % (1e-7)
7	Below 0.000001 % (1e-8)

bCoreDetect

Specifies whether the core of fingerprint is detected for verification. If *bCoreDetect* is TRUE, the verification fails only when the core of fingerprint is not detected in the center of SFR300 scanner or SFR200 scanner. If *bCoreDetect* is FALSE, the verification may fail as long as the fingerprint image is good enough.

Return Values

SF_SUCCESS

The verification succeeds.

SF_VERIFYFAIL

The verification fails.

SF_NOTGOODIMAGE

The fingerprint image is not good enough for verification.

The following return values are available only if *bCoreDetect* is TRUE.

SF_CORETOCENTER

The core of fingerprint is not detected.

SF_CORETOLEFT

SF_CORETORIGHT

SF_CORETOTOP

SF_CORETOBOTTOM

The core of fingerprint is detected, but should be placed in the center of SFR300 scanner or SFR200 scanner. The return values can be two combinations of these 4 values.

Remarks

Before the **SF_Verify** function is called, the **SF_Capture** function should be called to acquire a fingerprint image to be verified. This function can be used for 1:N matching as well as 1:1 matching.

Example

For 1:1 matching

```
int ret;
unsigned char m_EnrollTemplate[ SF_TEMPLATESIZE ];
while ( 1 ) {
        if ( !SF_Capture() ) {
                return; // Error: The acquisition of an image fails.
        }
        ret = SF_Verify( m_EnrollTemplate, 3, TRUE );
        if ( ret == SF_SUCCESS ) {
                return; // Verification succeeds.
        }
        if ( ret == SF_VERIFYFAIL ) {
                return; // Verification fails.
        }
        if ( ret == SF_NOTGOODIMAGE ) {
                // Warning: The fingerprint image is not good enough for verification.
        }
        else {
                // Warning: The core of fingerprint should be placed in the center of SFR300 scanner or
SFR200 scanner.
        }
}
```

For 1:N matching

```
int i, ret;
unsigned char m_EnrollTemplateDB[ 10 ][ SF_TEMPLATESIZE ];
while ( 1 ) {
        if ( !SF_Capture() ) {
                return; // Error: The acquisition of an image fails.
        }
        for ( i = 0 ; i < 10 ; i++ ) {
                ret = SF_Verify( m_EnrollTemplateDB[ i ], 3, TRUE );
                if ( ret == SF_SUCCESS ) {
                        return; // Verification succeeds.
                }
                if ( ret == SF_NOTGOODIMAGE ) {
                        break;
                }
        }
        if ( ret == SF_VERIFYFAIL ) {
                return; // Verification fails.
        }
        if ( ret == SF_NOTGOODIMAGE ) {
                // Warning: The fingerprint image is not good enough for verification.
        }
        else {
                // Warning: The core of fingerprint should be placed in the center of SFR300 scanner or
SFR200 scanner.
        }
}
```

SF_Identify

The **SF_Identify** function identifies the fingerprint image acquired using the **SF_Capture** function with the fingerprint templates enrolled formerly.

int SF_Idenify(unsigned char Templates, int Count, int* Match, int SecurityLevel, int bCoreDetect, int Timeout)**

Parameters

Templates

A 2-d pointer to the buffer that stores the fingerprint templates enrolled formerly.

Count

Specifies a number of the fingerprint templates enrolled formerly.

Match

A pointer to the value that stores the order of templates matched.

SecurityLevel

Specifies the level of security. This is in the range 1 to 7. If *SecurityLevel* is 7, the most secure verification is guaranteed. The value 4 is suitable in general case. Physical meaning of *SecurityLevel* is as follows.

SecurityLevel	False Accept Rate (FAR)
1	Below 1 % (1e-2)
2	Below 0.1 % (1e-3)
3	Below 0.01 % (1e-4)
4	Below 0.001 % (1e-5)
5	Below 0.0001 % (1e-6)
6	Below 0.00001 % (1e-7)
7	Below 0.000001 % (1e-8)

bCoreDetect

Specifies whether the core of fingerprint is detected for identification. If *bCoreDetect* is TRUE, the identification fails only when the core of fingerprint is not detected in the center of SFR300 scanner or SFR200 scanner. If *bCoreDetect* is FALSE, the identification may fail as long as the fingerprint image is good enough.

Timeout

Specifies maximum time for matching in milliseconds. If elapsed time for matching exceeds timeout, function stops further matching and returns error code. This time is only for matching, not for extraction. If *Timeout* is 0, no timeout occurs.

Return Values

SF_SUCCESS

The identification succeeds.

SF_MATCHFAIL

The identification fails.

SF_NOTGOODIMAGE

The fingerprint image is not good enough for identification.

SF_MATCHTIMEOUT

The elapsed time for matching exceeds specified timeout.

The following return values are available only if *bCoreDetect* is TRUE.

SF_CORETOCENTER

The core of fingerprint is not detected.

SF_CORETOLEFT

SF_CORETORIGHT

SF_CORETOTOP

SF_CORETOBOTTOM

The core of fingerprint is detected, but should be placed in the center of SFR300 scanner or SFR200 scanner. The return values can be two combinations of these 4 values.

Remarks

Before the **SF_Identify** function is called, the **SF_Capture** function should be called to acquire a fingerprint image to be identified. This function can be used for fast 1:N matching.

Example

```
const int DB_SIZE = 10;
unsigned char m_EnrollTemplateDB[ DB_SIZE ][ SF_TEMPLATESIZE ];
int i, ret;
unsigned char* Template[ DB_SIZE ];
for ( i = 0 ; i < DB_SIZE ; i++ ) {
    Template[ i ] = m_EnrollTemplateDB[ i ]; // copy the pointer to templates
}
```

```
while ( 1 ) {
        if ( !SF_Capture() ) {
                return; // Error: The acquisition of an image fails.
        }
        if ( ( ret = SF_Identify( Template, DB_SIZE, &i, 3, TRUE, 1000 ) ) == SF_SUCCESS ) {
                return; // Identification succeeds at i-th template.
        }
        else if ( ret == SF_MATCHFAIL ) {
                return; // Identification fails.
        }
        else if ( ret == SF_MATCHTIMEOUT ) {
                return; // Identification fails within 1 second ( Timeout parameter 1000 milliseconds ).
        }
        if ( ret == SF_NOTGOODIMAGE ) {
                // Warning: The fingerprint image is not good enough for verification.
        }
        else {
                // Warning: The core of fingerprint should be placed in the center of SFR300 scanner or
SFR200 scanner.
        }
}
```

SF_IdentifyMT

The **SF_IdentifyMT** function identifies the fingerprint image acquired using the **SF_Capture** function with the fingerprint templates enrolled formerly. This function is a multi thread version of the **SF_Identify**

int SF_IdenifyMT(unsigned char Templates, int Count, int* Match, int SecurityLevel, int bCoreDetect, int Timeout)**

Parameters

Templates

A 2-d pointer to the buffer that stores the fingerprint templates enrolled formerly.

Count

Specifies a number of the fingerprint templates enrolled formerly.

Match

A pointer to the value that stores the order of templates matched.

SecurityLevel

Specifies the level of security. This is in the range 1 to 7. If *SecurityLevel* is 7, the most secure verification is guaranteed. The value 4 is suitable in general case. Physical meaning of *SecurityLevel* is as follows.

SecurityLevel	False Accept Rate (FAR)
1	Below 1 % (1e-2)
2	Below 0.1 % (1e-3)
3	Below 0.01 % (1e-4)
4	Below 0.001 % (1e-5)
5	Below 0.0001 % (1e-6)
6	Below 0.00001 % (1e-7)
7	Below 0.000001 % (1e-8)

bCoreDetect

Specifies whether the core of fingerprint is detected for identification. If *bCoreDetect* is TRUE, the identification fails only when the core of fingerprint is not detected in the center of SFR300 scanner or SFR200 scanner. If *bCoreDetect* is FALSE, the identification may fail as long as the fingerprint image is good enough.

Timeout

Specifies maximum time for matching in milliseconds. If elapsed time for matching exceeds timeout, function stops further matching and returns error code. This time is only for matching, not for extraction. If *Timeout* is 0, no timeout occurs.

Return Values

SF_SUCCESS

The identification succeeds.

SF_MATCHFAIL

The identification fails.

SF_NOTGOODIMAGE

The fingerprint image is not good enough for identification.

SF_MATCHTIMEOUT

The elapsed time for matching exceeds specified timeout.

The following return values are available only if *bCoreDetect* is TRUE.

SF_CORETOCENTER

The core of fingerprint is not detected.

SF_CORETOLEFT

SF_CORETORIGHT

SF_CORETOTOP

SF_CORETOBOTTOM

The core of fingerprint is detected, but should be placed in the center of SFR300 scanner or SFR200 scanner. The return values can be two combinations of these 4 values.

Remarks

Before the **SF_IdentifyMT** function is called, the **SF_Capture** function should be called to acquire a fingerprint image to be identified. This function can be used for fast 1:N matching.

SF_GetEnrollQuality

The **SF_GetEnrollQuality** function returns quality of processed image.

int SF_GetEnrollQuality(void)

Parameters

This function has no parameters.

Return Values

Quality value which is from 1 to 100. Typically this value should be above 30 for further processing such as enroll and matching. Especially in case of enrollment, the use of good quality image (above 50) is highly recommended.

Remarks

This function returns valid value, only after the **SF_Enroll**, **SF_EnrollWithVerify**, **SF_Verify**, or **SF_Identify** function is called.

SF_AbortMatching

The **SF_AbortMatching** function aborts matching process of The **SF_Identify** function.

void SF_AbortMatching(void)

Parameters

This function has no parameters.

Return Values

This function has no return values.

SF_RotateTemplate

The **SF_RotateTemplate** function rotates the template of fingerprint enrolled formerly to the amount of 180 degrees.

void SF_RotateTemplate(unsigned char* Template)

Parameters

Template

A pointer to the buffer that stores the template of fingerprint extracted formerly.

Return Values

This function has no return values.

Remarks

The **SF_RotateTemplate** function can be used to support 360 degree rotational invariant matching. The **SF_Verify** and **SF_Identify** function permits 180 degree (-90 ~ +90 degree) rotation.

Example

```
unsigned char m_EnrollTemplate[ SF_TEMPLATESIZE ]; // original enrolled template
unsigned char m_RotatedTemplate[ SF_TEMPLATESIZE ];
memcpy( m_RotatedTemplate, m_EnrollTemplate, SF_TEMPLATESIZE );
RotateTemplate( m_RotatedTemplate );
// For the SF_Verify function call, both m_EnrollTemplate and m_RotatedTemplate should be used.
```

Constant List

Constant Name	Value
SF_TEMPLATESIZE	384
SF_VERIFYFAIL	0
SF_MATCHFAIL	0
SF_SUCCESS	1
SF_MATCHTIMEOUT	128
SF_MATCHABORT	129
SF_NOTGOODIMAGE	2
SF_CORETOCENTER	4
SF_CORETOLEFT	8
SF_CORETORIGHT	16
SF_CORETOTOP	32
SF_CORETOBOTTOM	64
SF_NORMAL_MODE	0
SF_FAST_MODE	1
SF_INVALID_MODE	-1

Revision Notes

V2.0	2005-2-5	Created.
V2.5	2006-5-3	SF_SetFastMode function is added.
		SF_IdentifyMT function is added.
		SF_GetDeviceNumber function is added.
		SF_GetDevice function is added.
		SF_SetDevice function is added.
		SF_GetSerial function is added.
		Constant List is updated.

Contact

Suprema Inc.

Dongcheon Bldg., 13-21 Yangjae-dong, Seocho-gu, Seoul, 137-130, Korea
TEL: +82-2-571-9305
FAX: +82-2-571-9306
E-mail: support@supremainc.com, sales@supremainc.com
Web: www.supremainc.com