

UFIS Software Development Kit Users Manual

ABS Applied Biometric Systems GmbH
SDK Version 3.40
June 15, 2005

Table of Contents

<i>Table of Contents</i>	<i>1</i>
<i>Software Development Kit</i>	<i>2</i>
Purpose	2
SDK Files	2
SDK Testing	2
SDK Functions	3
ufisOpenDevice	3
ufisOpenDeviceT	3
ufisCloseDevice	3
ufisIsOpenDevice	4
ufisStartupSensor	4
ufisIsFingerPresent	4
ufisGetImage	4
ufisConvertToBmp24	4
ufisGetLengthBmp24	5
ufisGetHeight	5
ufisGetWidth	5
ufisGetLength	5
ufisWhatOpened	5
ufisGetDeviceID	6
ufisControlLED	6
ufisFlashLED	6
Return Codes	7
Code Samples	8
Output the fingerprint to screen	9

Software Development Kit

Purpose

This document describes the UFIS Software Development Kit (SDK) and the functions exported by the SDK for use by a calling application.

The SDK is a “thin” software interface between a calling application and UFIS scanners.

The SDK is designed for the fingerprint image acquisition from the UFIS scanner, its quality estimation, minutia extraction, fingerprint matching, template creation and storing in a database, etc.

SDK supports the Windows 98SE /2000 / XP operating environments and serves the UFIS scanner with USB interface.

SDK Files

The SDK consists of following files:

- | | |
|-------------------|-----------------------|
| 1. ufisc.h | SDK header file |
| 2. ufisc.lib | SDK import library |
| 3. ufisc.dll | SDK |
| 4. FingerChip.dll | functions for UFIS112 |

SDK Testing

This SDK version was tested with UFIS (USB interface) scanners running firmware versions 0.1, 1.1, 2.1 under Windows 98SE /2000 / XP.

SDK Functions

This section lists all the externally visible SDK functions

ufisOpenDevice

Syntax: `int ufisOpenDevice(unsigned long ID = 0xffffffff)`

Description: Opens and initializes the UFIS scanner. If you don't set ID, the function tries to open any USB fingerprint scanner. If you set ID, the function tries to open the fingerprint scanner with given ID.

Parameters: ID – device identification number.

Return values: handle of device
rcERROR_OPEN_USBDEVICE.
rcERROR_MEMORY_ALLOCATION

ufisOpenDeviceT

Syntax: `int ufisOpenDeviceT(int type_scanner, unsigned long ID = 0xffffffff)`

Description: Opens and initializes the UFIS scanner. If you don't set ID, the function tries to open any USB fingerprint scanner. If you set ID, the function tries to open the fingerprint scanner with given ID.

Parameters: type_scanner – type of scanner, ID – device identification number.
TYPE_SCANNER_UFIS100 – UFIS100 scanner
TYPE_SCANNER_UFIS110 – UFIS110 scanner
TYPE_SCANNER_UFIS111 – UFIS111 scanner
TYPE_SCANNER_UFIS112 – UFIS112 scanner
TYPE_SCANNER_UFIS210 – UFIS210 scanner
TYPE_SCANNER_UFIS310 – UFIS310 scanner

ID – device identification number.
If you would like to open any scanner, ID is 0xFFFFFFFF

Return values: handle of device
rcERROR_OPEN_USBDEVICE.
rcERROR_MEMORY_ALLOCATION

ufisCloseDevice

Syntax: `int ufisCloseDevice (int handle)`

Description: Closes the UFIS scanner. If you don't set ID, the function tries to close the opened USB fingerprint scanner. If you set ID, the function tries to close the fingerprint scanner with given ID.

Parameters: handle - handle of the device

Return values: rcOK,
rcERROR_WRITE_USBDEVICE,
rcERROR_CLOSE_USBDEVICE.

ufisIsOpenDevice

Syntax: `int ufisIsOpenDevice (int handle, unsigned long ID = 0xffffffff)`

Description: Check activity of device with current ID.

Parameters: handle - handle of the device, ID – device identification number.

Return values: rcOK,
rcERROR_WRITE_USBDEVICE
rcERROR_OPEN_USBDEVICE.

ufisStartupSensor

Syntax: `int ufisStartupSensor (int handle)`

Description: Startups the UFIS scanner. This is first function after ufisOpenDevice as usual.

Parameters: handle - handle of the device

Return values: rcOK,
rcERROR_WRITE_USBDEVICE
rcERROR_OPEN_USBDEVICE.

ufisIsFingerPresent

Syntax: `int ufisIsFingerPresent (int handle)`

Description: Detects the finger presence on the sensor's surface. We must call this function after ufisStartupSensor.

Parameters: handle - handle of the device

Return values: rcOK,
rcFINGER_NOT_PRESENT
rcERROR_READ_USBDEVICE,
rcERROR_OPEN_USBDEVICE.

ufisGetImage

Syntax: `unsigned char * ufisGetImage(int handle)`

Description: This function grabs the **image of a fingerprint** from the fingerprint scanner.

Parameters: handle - handle of the device

Return values: Pointer to the image buffer,
NULL – any errors.

ufisConvertToBmp24

Syntax: `unsigned char * ufisConvertToBmp24(int handle)`

Description: This function converts the row image to bitmap image (24 bits).

Parameters: handle - handle of the device

Return values: Pointer to the bitmap image buffer,
NULL – any errors.

ufisGetLengthBmp24

Syntax: `int ufisGetLengthBmp24(int handle)`

Description: This function returns the length of the bitmap image (24 bits).

Parameters: handle - handle of the device

Return values: the length of the bitmap image

ufisGetHeight

Syntax: `int ufisGetHeight(int handle);`

Description: The function returns the height of the image;

Parameters: handle - handle of the device

Return values: The number of pixel

ufisGetWidth

Syntax: `int ufisGetWidth(int handle);`

Description: The function returns the width of the image;

Parameters: handle - handle of the device

Return values: The number of pixel

ufisGetLength

Syntax: `int ufisGetLength(int handle);`

Description: The function returns the length of row image

Parameters: handle - handle of the device

Return values: The length of the image

ufisWhatOpened

Syntax: `int ufisWhatOpened (int handle)`

Description: The function returns the code of the open device

Parameters: handle - handle of the device

Return values: `TYPESCANNER_UFIS100` - The device code
`TYPESCANNER_UFIS110` - The device code

TYPESCANNER_UFIS111 - The device code
 TYPESCANNER_UFIS112 - The device code
 TYPESCANNER_UFIS210 - The device code
 TYPESCANNER_UFIS310 - The device code

ufisGetDeviceID

Syntax: `long ufisGetDeviceID(int handle);`

Description: The function returns the ID device

Parameters: handle - handle of the device

Return values: ID of device

ufisControlLED

Syntax: `int ufisControlLED(int handle, unsigned char set);`

Description: The function provides to control the LEDs on the UFIS210

Parameters: handle - handle of the device
 set is the control parametr
 LED_GREEN_ON - turn on the green LED
 LED_RED_ON - turn on the red LED
 LED_GREEN_OFF - turn off the green LED
 LED_RED_OFF - turn off the red LED

Return values: rcOK is OK
 rcERROR_WRITE_USBDEVICE is error

ufisFlashLED

Syntax: `int ufisFlashLED(int handle, int type, int msec);`

Description: The function provides to flash the LEDs on the UFIS210

Parameters: handle - handle of the device
 type is the choice of LEDs
 1 - GREEN
 2 - RED
 msec - flash time in millisecond

Return values: rcOK is OK
 rcERROR_WRITE_USBDEVICE is error

Return Codes

rcFINGER_NOT_PRESENT	= 1	Finger does not present.
rcOK	= 0	Operation successful.
rcERROR_OPEN_USBDEVICE	= -2	Error open USB device.
rcERROR_CLOSE_USBDEVICE	= -3	Error close USB device.
rcERROR_READ_USBDEVICE	= -4	Error read USB device.
rcERROR_WRITE_USBDEVICE	= -5	Error write USB device.
rcERROR_MEMORY_ALLOCATION	= -6	Memory allocation error.
rcERROR_CARD_NOT_FOUND	= -7	Card was not detected
rcERROR_CARD_TYPE	= -8	No correct card type
rcERROR_WRITECARD	= -9	Error write to card
rcERROR_CARDDRIVER_NOT_PRESENTS	= -10	Scanner does not support cardreader
ISNOTPRESENT	= 0	

Code Samples

This section contains a sample program the using of the complete set of SDK calls.

```
#include <stdlib.h>
#include <stdio.h>
#include "ufis.h"

void main(void)
{
    short rc;
    byte *pimg;
    int     handle, length;

    handle = ufisOpenDeviceT(TYPESCANNER_UFIS111, 0xffffffff);

    if ( handle < 0 ) {
        printf("Error OpenDevice\n");
        exit(1);
    }

    if ( ufisStartupSensor(handle) != rcOK ) {
        printf("Error StartupSensor \n");
        exit(1);
    }

    while(1) {
        if ( ufisIsFingerPresent(handle) == rcOK ) {
            pimg = ufisGetImage(handle);
            if ( pimg == NULL ) {
                printf("Error GetImage \n");
                break;
            }
            printf("Fingerprint row image is received\n");
            /*
             here is the place for image processing routine
            */
            //      Please, copy the row image to your array

            //      Convert row image to bitmap
            pimg = ufisConvertToBmp24(handle);
            length = ufisGetLengthBmp24(handle);

            if ( ufisStartupSensor(handle) != rcOK ) {
                printf("Error StartupSensor \n");
            }
        }
    }
    ufisCloseDevice(handle);
}
```

Output the fingerprint to screen

Variant 1:

```
unsigned char * bitmap;
int H, W;

bitmap = ufisGetImage(handle);
H = ufisGetHeight(handle);
W = ufisGetWidth(handle);

void Show(unsigned char * bitmap, int W, int H, CClientDC & ClientDC)
{
    int i, j;
    short color;

    for(i = 0; i < H; i++) {
        for(j = 0; j < W; j++) {
            color = (unsigned char)bitmap[j+i*W];
            ClientDC.SetPixel(j,i,RGB(color,color,color));
        }
    }
}
```

Variant 2:

```
unsigned char * pimg;
int Himg;
int Wimg;

bitmap = ufisGetImage(handle);
H = ufisGetHeight(handle);
W = ufisGetWidth(handle);

/* for example */
void CAdminVictoriaDlg::MyPaint(void)
{
    short color;
    /* IDC_IMAGE - GROUP BOX */
    CWnd* pWnd = GetDlgItem(IDC_IMAGE);
    CClientDC pControlDC(pWnd);
    RECT rect;
    int deltaX=1;
    int deltaY=4;

    pWnd->GetWindowRect(&rect);
    rect.left=5;
    rect.top=15;
    rect.bottom = rect.top + Himg-deltaY;
    rect.right = rect.left + Wimg-deltaX;
    CBitmap * oldBitmap;
    CRect clientRect;
```

```

CDC * m_pdcDisplayMemory = new CDC;
CBitmap * m_pBitmap = new CBitmap;

if (m_pdcDisplayMemory->GetSafeHdc() == NULL) {
    m_pdcDisplayMemory->CreateCompatibleDC(&pControlDC);
    m_pBitmap->CreateCompatibleBitmap(&pControlDC, Wimg-deltaX, Himg-deltaY);
    oldBitmap = m_pdcDisplayMemory->SelectObject(m_pBitmap);
}

int x0 = rect.left;
int y0 = rect.top;
int Beg=0;

for(int j=0; j<(unsigned)(Himg-deltaY); j++) {
    for(int i=0; i<(unsigned)(Wimg-deltaX); i++) {
        color = pimg[Beg+i+j*Wimg];
        m_pdcDisplayMemory->SetPixel(i,j,RGB(color,color,color));
    }
}
pControlDC.StretchBlt(x0, y0, Wimg-deltaX , Himg-deltaY, m_pdcDisplayMemory, 0, 0,
    Wimg-deltaX , Himg-deltaY, SRCCOPY);

delete m_pdcDisplayMemory;
delete m_pBitmap;
}

```