

Fingerprint Scanner Interface Specification

Version 1.50

**Hongda Research&Development
Center**

Rev.	Date	Description
1.50	2007/09/03	Support vista, support 64 bit windows
1.42	2007/07/27	Add several functions
1.40	2006/10/11	Created, support for S500E.

1. Introduction

This SDK can be match with USB fingerprint scanner S500E which produce by Hongda company,it can be realize a series of static state scanning ,dynamic stakeout functions for fingerprint image.

This SDK could be keep compatible with the SDK interface of S500/S503,the S500E could use on the application which use the SDK of S500/S503 made without any change.

*** New Functions ***

- ◆ Nondestructive truly original image, without any processing algorithm
- ◆ Using high-quality sensor, the image quality is top-ranking
- ◆ Take full advantage of USB2.0 the high-speed characteristics
- ◆ Automatic adjust parameters of sensor for dry or wet finger
- ◆ Install and uninstall driver simple
- ◆ Allow to connect 8 same model or many different model scanners at the same time
- ◆ Allow multi-application open same scanner at the same time
- ◆ Provide data space of 10K bytes to user
- ◆ Support vista and 64 bit windows system.

Scanner Specification:

Gray scale:	256
Resolution:	500DPI
Image size:	256 × 360
Distortion :	<1%
Frames Per Second:	15 FPS(USB2.0), 5 FPS(USB1.1)

2. Development kits Files

1	FpDrv.dll	SDK DLL File
2	FpDrv.h	C++ language header file
3	FpDrv.lib	Static link of DLL for VC.
4	FpTest.exe	Test application
5	Fp_SDK_Spec.pdf	DLL specification (this document)
6	FpDrv_S503.dll	Used when connect S503 or earlier model scanner
7	FpExample.cpp	Example source file
8	HdS500E.inf	Usage for setup driver
9	HdS500E.sys	Driver file
10	UpdateDev.dll	Used when install/remove driver
11	RunUpdate.exe	Install/remove driver
12	AddCert.exe	Add certification in 64bit system

3. System Requirements

◆ Hardware(PC)

- ◆ CPU: Pentium 300 MHz or higher
- ◆ RAM: 64MB or above
- ◆ Display: Standard VGA, 800×600 16 bit color
- ◆ Other: CD-ROM drive, mouse, USB port

◆ Operation System

- ◆ Windows: Windows98/Me/2000/XP/2003/Vista, 32 or 64bit
- ◆ Language: Chinese and English

4. Function Interface Specification

4.1 Data structure

4.1.1 Fp_Device_Attrib (device attributes)

```
typedef struct _Fp_Attrib
{
    WORD ModelCode;           //The Modelcode of the fingerprint scanner device
    WORD DeviceNum;          //with some hardware interface, you may connect
                             //more than one device
    WORD Interface;          //The fp scanner hardware interface
    WORD InsideType;         //The fp scanner type
    WORD Version;            //version number
    WORD ImageWidth;         //the scanned finger image width
    WORD ImageHeight;        //the scanned finger image height
    WORD ImageFormat;        //the scanned finger image format
    WORD VideoFormat;        //the video format
    WORD Resolution;         //the finger image resolution
    WORD State;              //0 - cap closed 1 - cap stop 2 - cap running
}Fp_Device_Attrib;
```

Description:

Define the full device attributes information, use to select and open current device, ask device status and get device attributes.

Members:

WORD ModelCode	The device identify of the scanner, must be set when initialize, can be one of the following values:
	FP_MODEL_FP7110A (0x0012) S500 scanner
	FP_MODEL_FP503 (0x0050) S503 scanner
	FP_MODEL_FP500E (0x0054) S500E scanner
WORD DeviceNum	Current device number, begin form 0,1,2... If connect with m

		ore scanners, it must be set. The default is 0.
WORD	Interface	The interface between scanner and PC. Return by driver. Uses one of the following values: FP_INTERF_USB (0x0001) USB interface FP_INTERF_AV (0x0002) Video AV interface FP_INTERF_EPP (0x0003) Parallel interface
WORD	InsideType	Scanner inside type, return by driver. Uses one of the following values: FP_TYPE_CHIP (0x0001) Touch single chip FP_TYPE_VIDEO (0x0002) Video FP_TYPE_CARD (0x0003) Card
WORD	Version	The current driver version, return by driver. High byte is main and low byte is minor version.
WORD	ImageWidth	Image width, pixel, return by device
WORD	ImageHeight	Image height, pixel, return by device
WORD	ImageFormat	Image format type, return by driver, Uses one of the following values: FP_IMGFORM_GRAY8 (0x0001) 8 bit gray image FP_IMGFORM_GRAY16(0x0002) 16 bit gray image FP_IMGFORM_RGB24 (0x0003) 24 bit color image
WORD	VideoFormat	Video format, return by driver. Uses one of the following values: FP_VIDFORM_RGB24 (0x0001) 24 bit color FP_VIDFORM_I420 (0x0002) i420 color FP_VIDFORM_GRAY8 (0x0003) 8 bit gray If ModelCode is FP_MODEL_FP503 or earlier, return to FP_VIDFORM_RGB24, if ModelCode is FP_MODEL_FP500E, return to FP_VIDFORM_GRAY8.
WORD	Resolution	Image distinguish (DPI), return by driver
WORD	State	Current state of scanner, return by driver. Uses one of the foll

owing values:

FP_STATE_OFF	(0x0000)	Scanner shut down
FP_STATE_STOP	(0x0001)	Scanner stop
FP_STATE_RUN	(0x0002)	Scanner run

4.1.2 Fp_Image_Attrib (Image Attributes Structure)

```
typedef struct _Img_Attrib
{
    short  FrameRate;           //fingerprint preview framerate
    short  Brightness;         //fingerprint image brightness
    short  Contrast;           //fingerprint image contrast
    short  Exposure;           //fingerprint image exposure
}Fp_Image_Attrib;
```

Description:

Define the full image attributes information, use to consult and adjust image character.

Members:

short FrameRate video image frame per second, unit FPS
short Brightness Image brightness coefficient
short Contrast Image contrast coefficient
short Exposure Image exposure coefficient

Each member value range get from Fp_Img_Attrib_Range structure.

4.1.3 Fp_Img_Attrib_Range (Image Attributes Range)

```
typedef struct _Img_Attrib_Range
{
    BYTE  MinFrameRate;
    BYTE  MaxFrameRate;
    WORD  MinBrightness;
    WORD  MaxBrightness;
    WORD  MinContrast;
    WORD  MaxContrast;
```

```
WORD   MinExpose;  
WORD   MaxExpose;  
DWORD  Reserved;  
}Fp_Img_Attrib_Range;
```

Description:

Define adjust range of members in Fp_Image_Attrib, return by driver.

成员说明:

BYTE	MinFrameRate	Adjustable frame min
BYTE	MaxFrameRate	Adjustable frame max
WORD	MinBrightness	Adjustable brightness min
WORD	MaxBrightness	Adjustable brightness max
WORD	MinContrast	Adjustable contrast min
WORD	MaxContrast	Adjustable contrast max
WORD	MinExpose	Adjustable exposure min
WORD	MaxExpose	Adjustable exposure max
DWORD	Reserved	Reserved; must be zero

If the max is less than min, means the device is not support the property; If the min is equal to max, means the property is not adjustable.

4.2 Function Definition Specification

4.2.1 Fp_Init

Initialize the scanner.

```
BOOL __stdcall Fp_Init(  
    IN    HWND hParent,  
    IN    LPCRECT lprcDispRect,  
    IN OUT Fp_Device_Attrib *pDevAttr,
```

OUT **Fp_Image_Attrib** *pImgAttr
);

Parameters:

hParent

Handle to the parent window. Display monitor images in it, If this parameter is NULL, the desktop window becomes the parent window. If this parameter is INVALID_HANDLE_VALUE, it will not show monitor window.

lprcDispRect

Video monitor windows display rectangle region, pointer to structure with rectangle, The rectangle is relative coordinate of the parent window.

pDevAttr

Pointer to fingerprint scanner device attributes. Through ModelCode and DeviceNum to select device, for more information refer to data structure.

pImgAttr

Pointer to image attributes structure. for more information refer to data structure.

Return Values:

Successful return nonzero value, failure return is FALSE.

Remarks:

If initialize success, return to device info and show image monitor with user's requirements. The member ModelCode of pDevAttr must be set. When ModelCode is FP_MODEL_TEST (0), execute scanner auto test function, If the function return as TRUE,

test success, pDevAttr pointer to device attributes contents and the scanner closed automatically; test failure if return zero. ModelCode return to the current scanner connected. If no special declaration, the following function must be called after this function .

4.2.2 Fp_Uninit

To shut down device, you must call it when you exit application.

```
void __stdcall Fp_Uninit(  
    IN    Fp_Device_Attrib *pDevAttr  
);
```

Parameters:

pDevAttr

Pointer to the current scanner attributes structure.

Return Values:

No

Remarks:

To shut down device, you must call it when you exit application.

4.2.3 Fp_GetImage

Capture an image data to buffer.

```
BOOL __stdcall Fp_GetImage(  
    IN    LPCRECT lprcClipRect,
```

```
    OUT  BYTE * const pBuff  
);
```

Parameters:

lprcClipRect

Pointer to the intercept rectangle region, If it is NULL to get the whole image.

lprcClipRect point to rectangle width is lprcClipRect->right - lprcClipRect->left,height is prcClipRect->bottom - lprcClipRect->top. Supposed the image is 640 * 640, to get the whole image data should be set as

```
lprcClipRect->left = 0;  
lprcClipRect->right = 256;  
lprcClipRect->top = 0;  
lprcClipRect->bottom = 360;
```

pBuff

Pointer to a buffer to receive image data. It must be allocated enough size.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Fp_GetImage capture the image in the appointed rectangle region , which is relative to image region.

4.2.4 Fp_SaveImage

Save the image to bitmap file.

```
BOOL __stdcall Fp_SaveImage(  
    IN    int nImgType,  
    IN    int nWidth,  
    IN    int nHeight,  
    IN    const BYTE *pBuffer,  
    IN    LPCSTR pPath  
);
```

Parameters:

nImgType

Image file type, now support following values:

FP_IMGFILE_BMP (0x0001) BMP type, no compress

FP_IMGFILE_JPG (0x0002) JPEG type, lossy compression

FP_IMGFILE_PNG (0x0003) PNG type, lossless Compression

nWidth

Image width.

nHeight

Image height.

pBuffer

Pointer to the buffer containing the data to be written to the file.

pPath

Pointer to a null-terminated string that names the file to be saved.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

To make the image in buffer save as special image file. pBuffer data is get from [Fp_GetImage](#) function .

4.2.5 Fp_ShowAdjDialog

Show image property adjustable dialog box.

```
BOOL __stdcall Fp_ShowAdjDialog(  
    IN    int nDlgId  
);
```

Parameters:

nDlgId

Identify of the dialog box, this parameter can be one of the following values:

FP_DLG_FILTER(0X0001) filter page

This parameter is not be used now.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Show image property adjustable dialog box. Fp_ShowAdjDialog does not return control until close the modal dialog box. This function is only simple example, user can call other function to realize image property adjustment. After every adjustment, the function save its property value and use for next open device.

4.2.6 Fp_ShowAdjDialog1

Display image property adjustable dialog box , It is Fp_ShowAdjDialog's extending.

BOOL __stdcall Fp_ShowAdjDialog1(

IN int nDlgId,

IN HWND hParent

);

Parameters:

nDlgId

Same as Fp_ShowAdjDialog.

hParent

Handle to parent window.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Show image property adjustable dialog box. Fp_ShowAdjDialog1 does not return control until close the modal dialog box. [Fp_ShowAdjDialog](#) call this function inside.

4.2.7 Fp_GetImgAttrib

To get image's attributes info of current scanner .

```
BOOL __stdcall Fp_GetImgAttrib(  
    OUT    Fp_Image_Attrib *pImgAttr  
);
```

Parameters:

pImgAttr

Pointer to Fp_Image_Attrib struct. Detailed info refer to data structure Specification.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

The image property info is returned by device.

4.2.8 Fp_SetImgAttrib

Changes the image attributess of sensor.

```
BOOL __stdcall Fp_SetImgAttrib(  
    IN    Fp_Image_Attrib *pImgAttr  
);
```

Parameters:

pImgAttr

Pointer to a Fp_Image_Attrib structure. More info refer to data structure.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Changes the image attributess of sensor. The member range return from [Fp_GetImgAttribRange](#).

4.2.9 Fp_GetDevAttr

To get attributess of current device.

```
BOOL __stdcall Fp_GetDevAttr(  
    OUT      Fp_Device_Attrib *pDevAttr  
);
```

Parameters:

pDevAttr

Pointer to Fp_Device_Attrib structure.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

The Fp_GetDevAttr can be called before Fp_Init.

If called before Fp_Init, get the scanner default attributes;

If called after Fp_Init, get the scanner attributes in using.

4.2.10 Fp_SetWindow

Change monitor window.

```
BOOL __stdcall Fp_SetWindow(  
    IN    HWND hParent,  
    IN    LPCRECT lprcDispRect  
);
```

Parameters:

hParent

Handle to parent window of monitor window, can be NULL or
INVALID_HANDLE_VALUE.

lprcDispRect

Pointer to a rectangle structure of monitor region on parent window, can be set to
NULL. If hParent be set to NULL, lprcDispRect is relative coordinate of the parent
window.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Change video window or display region after opened window.

If hParent as NULL, video window display on desktop.

If hParent as INVALID_HANDLE_VALUE, it will not display video window.

If lprcDispRect as NULL, it will default size and place to show video window.

4.2.11 Fp_SetCallback

Sets callback function of capture an image.

```
BOOL __stdcall Fp_SetCallback(  
    IN    Fp_Callback_ pfnCallback  
);
```

Parameters:

pfnCallback

Pointer to a Fp_Callback type function. Fp_Callback_ is defined in FpDrv.h:

```
typedef void (__stdcall *Fp_Callback_)(BYTE * const pBuffer,  
    long lBufferSize);
```

pBuffer pointer to an image buffer, lBufferSize is image size, with byte as unit.

If member VideoFormat in Fp_Device_Attrib structure is FP_VIDFORM_GRAY8, per byte as a pixel, if VideoFormat is FP_VIDFORM_RGB24, the next three bytes as a pixel.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

The callback function will be called after captured an image and can process image data in

it. pfnCallback be called before display and mosaic image, its processed image data will decide the final display result. pfnCallback process the image data has no influence to Fp_GetImage.

pfnCallback should be complete before the next frame image, otherwise it may cause the follow image frames lost .

pfnCallback should be set to NULL if no use.

4.2.12 Fp_Flush

Set next frame as white screen.

BOOL __stdcall Fp_Flush(void);

Parameters:

No

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Fp_Flush set each byte of next image data as 0xFF.

4.2.13 Fp_GetLastError

The Fp_GetLastError function returns the calling thread's last-error code value.

int __stdcall Fp_GetLastError(

OUT **LPSTR** szErrInfo
);

Parameters:

szErrInfo

Pointer to error information, use to get current error information, set to NULL if you don't want to return error information. The size of memory of szErrInfo should no little than 256 bytes.

Return Values:

The return value is the calling thread's last-error code value, you can get particular information from the interface head file FpDrv.H.

Remarks:

When fail to call other fuction, you can via it to get error information and error code. You should call Fp_GetLastError immediately when a function's return value indicates that such a call will return useful data.

4.2.14 Fp_AutoAdjust

Set and get state of auto-adjust image function.

BOOL **__stdcall** Fp_AutoAdjust(
 IN **DWORD** dwState
);

Parameters:

dwState

Time value, MS as unit, means waiting for fingerprint pressing time. With dwState time, if press fingerprint, adjust fingerprint image, then exit the adjustment. The time of adjustment is not include dwState.

If within dwState time not press fingerprint , then adjust by the empty image.

The following values show special meanings:

When the value is 0, close the auto adjustment immediately.

When the value is 1, check is under auto adjustment state, return TRUE to auto adjustment state.

When the value is 0xFFFFFFFF, if not close Fp_AutoAdjust(0), it will under auto adjustment state always.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

When the equipment is under auto adjustment state, no need to hand set up

Fp_Image_Attrib structure of Brightness、 Contrast and Exposure value.

After adjustment, the function return immediately and not wait for adjustment finish .

The default open auto adjustment.

4.2.15 Fp_GetImgAttribRange

To get members range of Fp_Image_Attrib structure.

```
BOOL __stdcall Fp_GetImgAttribRange(  
    OUT    Fp_Img_Attrib_Range *pRange  
);
```

Parameters:

pRange

Pointer to Fp_Img_Attrib_Range structure.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:Confirm members range through the function before call procedure [Fp_SetImgAttrib](#).

4.2.16 Fp_Pause

Pause receive image data.

```
BOOL __stdcall Fp_Pause(  
    IN    BOOL bPause  
);
```

Parameters:

bPause

Set TRUE to pause and false to continue.

Return Values:If *bPause* value is TRUE, Successful returns TRUE and failure returns FALSE.If *bPause* value is FALSE, if the scanner start capture returns TRUE, else returns FALSE.

If no enough times to call Fp_Pause(FALSE), a call to Fp_GetLastError returns

FP_ERROR_STATE_STOP.

Remarks:

Repeat Fp_Pause (TRUE) pause, then must use the same times Fp_Pause (FALSE) to cancel pause and continue capture.

4.2.17 Fp_GetDevNum

Get scanner number of connecting PC.

```
int __stdcall Fp_GetDevNum(  
    IN    WORD    wModelCode,  
    OUT   WORD    wOurDevs[128]  
);
```

Parameters:

wModelCode

Scanner model value to query, refer to data structure explaining. If it is 0, means checking all compatible scanner models.

wOurDevs

Every value of array return to current all connecting scanner models, can be set to NULL.

Return Values:

Return to number of appointed model of scanner. Failure return is 0.

Remarks:

Fp_GetDevNum can be called before Fp_Init function.

4.2.18 Fp_ChangePalette

Change image color palette.

```
BOOL __stdcall Fp_ChangePalette(  
    IN    RGBQUAD *pQuad,  
    IN    DWORD dwQuadNum  
);
```

Parameters:

pQuad

Pointer to a RGBQUAD structure. The structure definition is as following :

```
typedef struct tagRGBQUAD { // rgbq  
    BYTE    rgbBlue;  
    BYTE    rgbGreen;  
    BYTE    rgbRed;  
    BYTE    rgbReserved;  
} RGBQUAD;
```

For more information about RGBQUAD refer to MSDN.

dwQuadNum

Number of pQuad 's member, between [1, 256] .

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

The function only valid when VideoFormat in Fp_Device_Attrib structure is FP_VIDFORM_GRAY8.

4.2.19 Fp_SetDrawImageCallback

Set callback function on draw image on screen.

```
BOOL __stdcall Fp_SetDrawImageCallback(  
    IN    Fp_DrawImage_ pfnCallback  
);
```

Parameters:

pfnCallback

Pointer to callback function of Fp_DrawImage_ type. Fp_DrawImage in FpDrv.h definition as following :

```
typedef void (__stdcall *Fp_DrawImage_)(HDC hdc, RECT rcRect);
```

hdc is device context handle. rcRect is image display position on screen.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

The pfnCallback be called after drawing every frame image. Set to NULL when no use.

4.2.20 Fp_GetUserSpace

Get scanner personal data room size.

```
DWORD __stdcall Fp_GetUserSpace(  
    void  
);
```

Parameters:

No

Return Values:

If the function succeeds, the return value is space size of scanner provided,
In bytes.

If the function fails or no space be provided, the return value is 0.

To get extended error information, call [Fp_GetLastError](#).

Remarks:

Different scanner type or version may provide different space size, specific space size get from this function.

4.2.21 Fp_WriteUserData

Write user's private data.

```
BOOL __stdcall Fp_WriteUserData(  
    IN    LPVOID pBuf,  
    IN    DWORD dwAddr,  
    IN    DWORD dwLen
```

);

Parameters:

pBuf

Pointer to the buffer containing the data to be written to the user's space.

dwAddr

The begin address of space in device, start from 0.

dwLen

Number of bytes to write to the user's space.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

The written data is not over user space region, use [Fp_GetUserSpace](#) to get space size.

Primary data will be overwrite.

4.2.22 Fp_ReadUserData

Read user's private data.

BOOL **stdcall** Fp_ReadUserData(
 OUT **LPVOID** pBuf,
 IN **DWORD** dwAddr,

```
    IN    DWORD dwLen  
);
```

Parameters:

pBuf

Pointer to the buffer that receives the data read from the user space.

dwAddr

The begin address of space in device, start from 0.

dwLen

Number of bytes to be read.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Read the data from the scanner user's space .

4.2.23 Fp_GetUSBType

Get PC USB interface type.

```
int __stdcall Fp_GetUSBType(int SpeedOf);
```

Parameters:

SpeedOf

How to check USB transfer speed, USB_SYSTEM means operation system supporting transfer type. USB_CURTRAN means the current image data transferring type.

Return Values:

If PC USB interface fit for USB 2.0 version, support high speed transfer, return to 2, otherwise return 1.

Remarks:

By using USB_SYSTEM, call it before Fp_Init, only under USB hardware support high speed and drive successfully, return to 2. By using USB_CURTRAN, return to the current actual transfer image date type.

4.2.24 Fp_GetExeRecord

Obtain all path and counts of executable files which ever opened the scanner.

```
BOOL __stdcall Fp_GetExeRecord (  
    OUT void *pBuf,  
    INOUT DWORD *Size,  
    OUT DWORD *ExistsNum  
);
```

Parameters:

pBuf

Pointer to a array buffer that receives the files list, terminated by '\n' characters.

pBuf can be NULL.

Size

Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the *pbuf* parameter. When the function returns, this variable contains the size of the data copied to *pBuf*.

The Size parameter can be NULL only if *pbuf* is NULL.

ExistsNum

Pointer to a variable that receives the number of executable files, Exclude no exist(uninstalled or deleted). This parameter can be NULL.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Judge which procedures to use this type of scanner. Mainly use to uninstall scanner driver, if still have other procedure usages to drive, should stop an uninstallation.

4.2.25 Fp_LoadImage

The **Fp_LoadImage** function loads a fingerprint file saved by Fp_SaveImage.

BOOL __stdcall Fp_LoadImage(

IN int nImgType,

OUT int *pWidth,

OUT int *pHeight,

OUT BYTE *pBuffer,

```
    IN    int nBufferSize,  
    IN    LPCSTR pPath  
);
```

Parameters:

nImgType

Specifies the type of image to be loaded, reference to `Fp_SaveImage`.

pWidth

Pointer to a variable that receives the width. This parameter can be NULL.

pHeight

Pointer to a variable that receives the height. This parameter can be NULL.

pBuffer

Pointer to a buffer that receives the image's data. This parameter can be NULL.

nBufferSize

Number of bytes of pBuffer.

pPath

Pointer to a null-terminated string that names the file to be opened.

Return Values:

Successful return is TRUE, failure return is FALSE.

Remarks:

Read the data of fingerprint image file, no include the information of document format.

Only support 8 gray level images.

5. Program Example

```
#include "fpdrv.h" // you should include the SDK header file in your source code

//function pointers definition
Fp_Init_      MyFp_Init;
Fp_Uninit_    MyFp_Uninit;
Fp_GetImgAttrib_  MyFp_GetImgAttrib;
Fp_SetImgAttrib_  MyFp_SetImgAttrib;
Fp_ShowAdjDialog_ MyFp_ShowAdjDialog;
Fp_GetImage_     MyFp_GetImage;
Fp_SaveImage_    MyFp_SaveImage;
Fp_SetCallback_  MyFp_SetCallback;
Fp_Flush_        MyFp_Flush;
Fp_SetWindow_    MyFp_SetWindow;
Fp_GetDevAttr_   MyFp_GetDevAttr;

//structure variable definition
Fp_Device_Attrib  FpDevAttr;
Fp_Image_Attrib   FpImgAttr;

//dynamic load the FpDrv.DLL
HINSTANCE hFpDrv;
hFpDrv = LoadLibrary("FpDrv.dll");

MyFp_Init = (Fp_Init_)GetProcAddress(hFpDrv, "Fp_Init");
MyFp_Uninit = (Fp_Uninit_)GetProcAddress(hFpDrv, "Fp_Uninit");
MyFp_GetImgAttrib = (Fp_GetImgAttrib_)GetProcAddress(hFpDrv, "Fp_GetImgAttrib");
MyFp_SetImgAttrib = (Fp_SetImgAttrib_)GetProcAddress(hFpDrv, "Fp_SetImgAttrib");
```

```
MyFp_ShowAdjDialog = (Fp_ShowAdjDialog_)GetProcAddress(hFpDrv, "Fp_ShowAdjDialog");
MyFp_GetImage = (Fp_GetImage_)GetProcAddress(hFpDrv, "Fp_GetImage");
MyFp_SaveImage = (Fp_SaveImage_)GetProcAddress(hFpDrv, "Fp_SaveImage");
MyFp_SetCallback = (Fp_SetCallback_)GetProcAddress(hFpDrv, "Fp_SetCallback");
MyFp_Flush = (Fp_Flush_)GetProcAddress(hFpDrv, "Fp_Flush");
MyFp_SetWindow = (Fp_SetWindow_)GetProcAddress(hFpDrv, "Fp_SetWindow");
MyFp_GetDevAttr = (Fp_GetDevAttr_)GetProcAddress(hFpDrv, "Fp_GetDevAttr");
```

```
//first you should initialize the device
```

```
RECT DispRect = {0, 0, 256, 360};
FpDevAttr.ModelCode = FP_MODEL_FP500E;           //set model code
FpDevAttr.DeviceNum = 0;                        //set device emulation
```

```
if (!MyFp_Init(this->GetSafeHwnd(), &DispRect, &FpDevAttr, NULL))
{
    return;
}
```

```
//as you like you can capture one fingerprint image
```

```
BYTE *pBuffer;
char pPath[] = ".\\Finger.bmp";

pBuffer = (BYTE*)malloc(FpDevAttr.ImageWidth * FpDevAttr.ImageHeight);

if (MyFp_GetImage(NULL, pBuffer))
{
    MyFp_SaveImage(FP_IMGFILE_BMP, FpDevAttr.ImageWidth, FpDevAttr.ImageHeight,
    pBuffer, pPath);
}
```

```
free(pBuffer);
```

```
//if you want to adjust the image quality easily, call following function
```

```
MyFp_ShowAdjDialog(0);
```

```
//as last , you must uninitialize the driver
```

```
MyFp_Uninit(&FpDevAttr);
```

```
FreeLibrary(hFpDrv);
```

6. Install and Uninstall Driver

For users of developments, provided very convenient method to install and uninstall the scanner driver. What underneath introduce is manual operation and you can also use the creation tool of InstallShield etc. to easily complete.

6.1 Install Driver

6.1.1 Copy Files

Enter the installed directory(like C:\Program Files\Hongda\S500E), copy following files:

- a. HdS500E.inf -> C:\Windows\Inf (support system directory is C:\Windows);
- b. HdS500E.sys -> C:\Windows\System32\Drivers,

In subdirectory of “Driver”, different directory correspond to different target system, list as follows:

Directory	Target Windows
98	Windows 98/Me
x86	Windows 2000/XP/2003/Vista, 32 bit version
AMD64	Windows XP/2003/Vista, AMD 64 bit version
IA64	Windows 2000/XP/2003/Vista IA 64 bit version

32 bit or 64 bit version system can judge by “PROCESSOR_ARCHITECTURE” of environment parameter.

- c. FpDrv.dll -> C:\MyAppDir(The directory of EXE to use scanner, modified base on actual);
- FpDrv_S503.dll -> C:\MyAppDir;

RunUpdate.exe -> C:\MyAppDir;

UpdateDev.dll -> C:\MyAppDir;

6.1.2 Register Dirver Information

Use RunUpdate to register the driver and hardware information.

Usages: RunUpdate [/a] [/d] [ModalCode] [Inf File Path]

/a - Install

/d - Uninstall

ModalCode - Scanner code, for S500E is 84

Inf File Path - Inf file path

For example, to install s500e driver in cmd line, run “RunUpdate /a 84 c:\windows\inf\hds500e.inf” .

This step can be ignored, if ignored, the system will discover a new hardware and popup a setup wizard dialog.

In 64 bit system, the driver need Digital Certificate. If you not use RunUpdate.Exe to install, must run AddCert.Exe as follow:

AddCert "Hongda Company"

6.1.3 Update Driver

If install fail, remove the device in “Device management machine” and re-install driver.

6.1.4 Silent Install

Another simple way is use silent mode to install and uninstall driver.

a. Run S500Ev1.5.exe in command line with ‘/r’ parameter. It is record

mode to create a setup.iss file.

Example: S500Ev1.5.exe /r

- b. It will display an usual setup screen. Can cancel SKD option while install.

Do not select “restart computer” lastly, otherwise batch file can not be run.

- c. After install, move setup.iss file in c:\windows to the directory same as S500Ev1.5.exe.

- d. Use the command to silent install: S500Ev1.5.Exe /s /f1"Setup.iss"

The uninstallation can also use a similar method.

For more information of parameters reference to ‘InstallShield’ help document.

6.2 Uninstall Driver

If installed driver with RunUpdate.exe, also need to uninstall driver with it.

For example, use “RunUpdate /d 84” to clean registered information.

The driver may be used by other programs, should judge before uninstall it with Fp_GetExeRecord.

6.3 Compatible of Old Version

The old version driver of the scanner(S500/S503 etc.) use Kernel Streaming model, and this model isn't suitable for fingerprint scanner. S500E's driver use new model to transfer data. Although drive a model dissimilarity, however our company provided an uniform API interface, as long as the software usage's unifying API interface development can keep

compatibly.

The old version SDK provided 4.2.1 - 4.2.14 functions, S500E's interface continue use and expand them. The software developed with old version SDK not need do any modification, only update driver and then can use S500E.

The update driver method is, one by one install old version SDK, software, S500E's driver, then rename old version fpdrv.dll to fpdrv_s503.dll and copy the new version fpdrv.dll in S500E's SDK to here. If you confirm no longer uses the old version scanner, can only install S500E's driver and replace old version fpdrv.dll if it exists.

If you use S500E in FPGuard 2.0(Hongda Fingerprint Security System), install sequence is S503's driver in FPGuard CD, S500E's driver, FPGuard software.