SZFP TECHNOLOGY LIMITED
WWW.SZFPTECH.COM

FP3550 Application Development Guide

Version：　V1.0.7

Directory

1 Outline

1.1 Introduction

This guide is an introduction document for POS application development based on the finance secure payment module

The finance secure module usually support magnetic cards, contactless smart cards, fast thermal printer , fingerprint and so on .

1.2 Revision of history

Date	Version	Remarks
2018.02.08	V1.0.2	First draft
2018.03.01	V1.0.3	add some interfaces and remove unused interface
2018.11.06	V1.0.7	add decode interface

2 Class Fingerprint （In the Java layer）

2.1 Constructor and method summary

Constructor:	Fingerprint ();
	int Lib_FpTestConnection()
	int Lib_FpSetParam(int nParamIndex, int nParamValue)
	Lib_FpGetParam(int nParamIndex, int[] pnParamValue)
	int Lib_FpGetDeviceInfo(String[] szDevInfo)
	int Lib_FpSetIDNote(int nTmplNo, String pstrNote)
	int Lib_FpGetIDNote(int nTmplNo, String[] pstrNote)
	int Lib_FpSetModuleSN(String pstrModuleSN)
	int Lib_FpGetModuleSN(String[] pstrModuleSN)
	int Lib_FpGetImage()
	int Lib_FpFingerDetect(int[] pnDetectResult)
	int Lib_FpUpImage(int nType, byte[] pFpData, int[] pnImgWidth, int[] pnImgHeight)
	int Lib_FpDownImage(byte[] pData, int nWidth, int nHeight)
	int Lib_FpSLEDControl(int nState)
	int Lib_FpStoreChar(int nTmplNo, int nRamBufferID, int[] pnDupTmplNo)
	int Lib_FpLoadChar(int nTmplNo, int nRamBufferID)
	int Lib_FpUpChar(int nRamBufferID, byte[] pbyTemplate)
	int Lib_FpDownChar(int nRamBufferID, byte[]

	pbyTemplate)
	int Lib_FpDelChar(int nSTmplNo, int nETmplNo)
	int Lib_FpGetEmptyID(int nSTmplNo, int nETmplNo, int[] pnEmptyID)
	int Lib_FpGetStatus(int nTmplNo, int[] pnStatus)
	int Lib_FpGetBrokenID(int nSTmplNo, int nETmplNo, int[] pnCount, int[] pnFirstID)
	int Lib_FpGetEnrollCount(int nSTmplNo, int nETmplNo, int[] pnEnrollCount)
	int Lib_FpGenerate(int nRamBufferID)
	int Lib_FpMerge(int nRamBufferID, int nMergeCount)
	int Lib_FpMatch(int nRamBufferID0, int nRamBufferID1)
	int Lib_FpSearch(int nRamBufferID, int nStartID, int nSearchCount, int[] pnTmplNo, int[] pnLearnResult)
	int Lib_FpVerify(int nTmplNo, int nRamBufferID, int[] pnLearnResult)

2.2 Method detail

2.2.1 FpOpen

Prototype:	int Lib_FpOpen();	
Function:	Open and reset the module	
Parameters:	none	
Return value:	0x00	success
	others	failed
	-1001	Send failed
	-1002	Recv failed
Note:		

2.2.2 FpClose

Prototype:	int Lib_FpClose();
Function:	Close the module
Parameters:	none

Return value:	0x00	success
	others	failed
	-1001	Send failed
	-1002	Recv failed
Note:		

2.2.3 FpTestConnection

Prototype:	**int Lib_FpTestConnection()**	
Function:	Check the connection status of the target and the host. The host needs to first send this command to detect with the target Connection Status. If unsuccessful, then the connection with the target module can be considered abnormal, or not working properly or Potter The rate is set incorrectly.	
Parameters:	none	
Return value:	0x00	success
	others	failed

2.2.4 FpSetParam

Prototype:	**int Lib_FpSetParam(int nParamIndex, int nParamValue)**	
Function:	According to the specified Parameter Type, set the device parameters (Device ID, Security Level, Baudrate, Duplication Check, Auto Learn) and return the result.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.5 FpGetParam

Prototype:	**Lib_FpGetParam(int nParamIndex, int[] pnParamValue)**
Function:	According to the specified parameter type (Parameter Type), access to the following parameters: Device ID, Security Level, Baudrate, Duplication Check, Auto Learn Refer to CMD_SET_PARAM above for parameter type codes.

Parameter:		
Return value:	0	Success (Registered ID)
	others	failed

2.2.6 FpGetDeviceInfo

Prototype:	**int Lib_FpGetDeviceInfo(String[] szDevInfo)**	
Function:	Gets the Device Information of the Target, the firmware name and version number of the device, and is fixed to "SEO_HTO20_FPC1011 Vx.x". X.x indicates the firmware version number (F / W Version)	
Parameter:		
Return value:	0	Success (Registered ID)
	others	failed

2.2.7 FpSetIDNote

Prototype:	**int Lib_FpSetIDNote(int nTmplNo, String pstrNote)**	
Function:	The ID Note is received from the Host and stored in the specified number.	
Return value:	0	success
	others	failed

2.2.8 FpGetIDNote

Prototype:	**int Lib_FpGetIDNote(int nTmplNo, String[] pstrNote)**	
Function:	Sends the ID Note in the specified number to the Host	
Parameter:		
Return value:	0	success
	others	failed
Note:		

2.2.9 FpSetModuleSN

Prototype:	**int Lib_FpSetModuleSN(String pstrModuleSN)**	
Function:	Receive the Module SN from the Host and save it in the module.	
Parameter:		
Return value:	0	success

	others	failed
Note:		

2.2.10 FpGetModuleSN

Prototype:	**int Lib_FpGetModuleSN(String[] pstrModuleSN)**	
Function:	The Module SN stored in the module is sent to the Host.	
Parameter:		
Return value:	0	Success
	others	failed
Note:		

2.2.11 FpGetImage

Prototype:	**int Lib_FpGetImage()**	
Function:	Fingerprint images are collected from the Collector and stored in the ImageBuffer.	
Parameter:		
Return value:	0	success
	others	failed
Note:		

2.2.12 FpFingerDetect

Prototype:	**int Lib_FpFingerDetect(int[] pnDetectResult)**	
Function:	Check the fingerprint input status at the time of receipt of this finger and return the result.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.13 FpUpImage

Prototype:	**int Lib_FpUpImage(int nType, byte[] pFpData, int[] pnImgWidth, int[] pnImgHeight)**
Function:	According to the specified Image Type (image type specifies

	whether the full-resolution or low-resolution fingerprint image Image), and sends the image stored in the ImageBuffer to the Host. ① If the Image Type is 0, it is Full Mode, and the whole picture is sent (optical: 242 * 266, FPC1011: 202 * 258) (2) If the Image Type is 1, it is Quarter Mode, and 1/4 image is sent (4 dots are 1 dots).	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.14 FpDownImage

Prototype:	**int Lib_FpDownImage(byte[] pData, int nWidth, int nHeight)**	
Function:	The image data received from the Host is stored in the ImageBuffer. Host to 496 bytes as the unit will be Figure Such as sending to Target. In this case, the image data number is transmitted at the same time.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.15 FpSLEDControl

Prototype:	**int Lib_FpSLEDControl(int nState)**	
Function:	Control the collector lamp	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.16 FpStoreChar

Prototype:	**int Lib_FpStoreChar(int nTmplNo, int nRamBufferID, int[] pnDupTmplNo)**
Function:	Saves the template in the specified Ram Buffer to the designated number fingerprint library of this module.

Parameters:		
Return value:	0x00	success
	others	failed

2.2.17 FpLoadChar

Prototype:	**int Lib_FpLoadChar(int nTmplNo, int nRamBufferID)**	
Function:	Store the fingerprint template in the specified number in the specified Ram Buffer.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.18 FpUpChar

Prototype:	**int Lib_FpUpChar(int nRamBufferID, byte[] pbyTemplate)**	
Function:	Send the Template in the specified Ram Buffer to the Host.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.19 FpDownChar

Prototype:	**int Lib_FpDownChar(int nRamBufferID, byte[] pbyTemplate)**	
Function:	Received from the Host Template is not stored in the specified Ram Buffer.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.20 FpDelChar

Prototype:	**int Lib_FpDelChar(int nSTmplNo, int nETmplNo)**
Function:	Delete the Template registered within the specified deletion range (from the beginning Template number to the ending

	Template number).	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.21 FpGetEmptyID

Prototype:	**int Lib_FpGetEmptyID(int nSTmplNo, int nETmplNo, int[] pnEmptyID)**	
Function:	Gets the specified range (from the beginning Template number to the ending Template number) that can be registered (Template is not registered) of the first Template number.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.22 FpGetStatus

Prototype:	**int Lib_FpGetStatus(int nTmplNo, int[] pnStatus)**	
Function:	Gets the registration status of the Template in the specified number.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.23 FpGetBrokenID

Prototype:	**int Lib_FpGetBrokenID(int nSTmplNo, int nETmplNo, int[] pnCount, int[] pnFirstID)**
Function:	Check the corrupted condition of all registered templates in the specified range (from the beginning Template number to the ending number). Write operation in the Flash, there may be due to power failure and other reasons leading to damage to the template. At any time (for example, the initial start of Target), the HOST checks the damage of the template.

Parameters:		
Return value:	0x00	success
	others	failed

2.2.24 FpGetEnrollCount

Prototype:	**int Lib_FpGetEnrollCount(int nSTmplNo, int nETmplNo, int[] pnEnrollCount)**	
Function:	Gets the total number of registered fingerprints within the specified range (from the beginning Template number to the ending Template number).	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.25 FpGenerate

Prototype:	**int Lib_FpGenerate(int nRamBufferID)**	
Function:	Template is generated from the fingerprint image in ImageBuffer and stored in the specified Ram Buffer.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.26 FpMerge

Prototype:	**int Lib_FpMerge(int nRamBufferID, int nMergeCount)**	
Function:	Syntheses the Template in the Ram Buffer and generates a Template and stores it in the specified Ram Buffer The number can be 2 or 3. If 2: then the synthesis Ram Buffer0 and Ram Buffer1 the Template. If 3, then the synthesis Ram Buffer0, Ram Buffer1 and Ram Buffer2 Template.	
Parameters:		
Return value:	0x00	success
	others	failed

2.2.27 FpMatch

Prototype：	int Lib_FpMatch(int nRamBufferID0, int nRamBufferID1)	
Function：	The two templates in the specified Ram Buffer are compared against each other.	
Parameters：		
Return value：	0x00	success
	others	failed

2.2.28 FpSearch

Prototype：	int Lib_FpSearch(int nRamBufferID, int nStartID, int nSearchCount, int[] pnTmplNo, int[] pnLearnResult)	
Function：	Specify the Template in the Ram Buffer with the specified search range (start Template number ~ end Template number) of all registered in the module database fingerprint template between 1: N ratio Pair and return the result.	
Parameters：		
Return value：	0x00	success
	others	failed

2.2.29 FpVerify

Prototype：	int Lib_FpVerify(int nTmplNo, int nRamBufferID, int[] pnLearnResult)	
Function：	Specify a 1: 1 match between the template in the Ram Buffer and the template with the specified number in the database. The result is returned.	
Parameters：		
Return value：	0x00	success
	others	failed

3 Class MCR

3.1 Constructor and method summary

Constructor:	**Mcr();**
Method Summary:	**int Lib_McrOpen();**
	int Lib_McrClose();
	int Lib_McrCheck();
	int Lib_McrRead(BYTE key_no, BYTE mode, BYTE *track1, BYTE *track2, BYTE *track3)

3.2 Method detail

3.2.1 Open

Prototype:	**int Lib_McrOpen();**	
Function:	Open the card reader	
Parameter:	None	
Return:	0	success
	others	fail
Note:	Read the magnetic card data by using interrupt way, once open the card reader, the magnetic head can read data as long as there is swipe, even the read function not called. Therefore the magnetic card reader will be better closed when there no need to use the magnetic card reader.	

3.2.2 Close

Prototype:	**int Lib_McrClose();**	
Function:	Close the card reader	
Parameter:	None	
Return:	0	success
	others	fail
Note:		

3.2.3 Check

Prototype:	**int Lib_McrCheck();**
Function:	Check there is a swipe or not.

Parameter:	None	
Return:	0	Swiped card
	Other	No card swiped
Note:	This function will return immediately no matter there is a swipe or not.	

3.2.4 Read

Prototype:	**int Lib_McrRead(BYTE key_no, BYTE mode, BYTE *track1, BYTE *track2, BYTE *track3);**	
Function:	Read the 1, 2, 3 track data in the buffer	
Parameters:	**keyNo[in]**	No use
	mode[in]	NO use
	track1[out]	The buffer of track1 data
	track2[out]	The buffer of track2 data
	track3[out]	The buffer of track3 data
Return Value:	0	Brush card error

	(>0)	bit0 = 1 read track1 data ok
		bit1 = 1 read track2 data ok
		bit2 = 1 read track3 data ok

Note:	Corrdinate with Lib_MCRCheck function. If no need some track data, the corresponding track buffer pointer can be set to NULL, then the track data will not be output. Generally the magnetic card ‘s data is according to the ISO7811 standard: track1 : 79 bytes track2 : 40 bytes track3 : 107 bytes This function also supports the magnetic card not conform to the ISO7811 standard.	

4 Class PICC

4.1 Constructor and method summary

Constructor:	**Picc();**
Method Summary:	**int Lib_PiccOpen();**
	int Lib_PiccClose();
	Int Lib_PiccCheck(BYTE mode,BYTE

	*CardType,BYTE *SerialNo,,BYTE *ATS,BYTE *ATSlen);
	int Lib_PiccCommand(APDU_SEND *ApduSend, APDU_RESP *ApduResp);

4.2 Method detail

4.2.1 Open

Prototype:	int Lib_PiccOpen();	
Function:	Power on and reset contactless card module, check the status of module.	
Parameters:	None	
Return value:	0	Success
	Other values	Fail,refer to the error code below
Note:	When device power on, the contactless module will be in close status default, before the transaction, this function will be called once; If this function not be called, other functions will be failed. This function will fail when the module is not installed or the module is fault.	

4.2.2 Close

Prototype:	int Lib_PiccClose();	
Function:	Close contactless card module, so that the module is turned off	
Parameters:	None	
Return value:	0x00	Success
	other values	Fail,refer to the error code below
Note:	After calling this function, contactless card module will be in a closed state, the module is no longer radiate the carrier .and only the function Lib_PiccOpen () will be effective after call this function.	

4.2.3 Check

Prototype:	**Int Lib_PiccCheck(BYTE mode,BYTE *CardType,BYTE *SerialNo,BYTE *ATS,BYTE *ATSlen);**	
Function:	Search the card according to the specified mode, select and active the card when searched.	
Parameters:	**mode[in]**	No use
	cardType [out]	card type byte buffer(2 bytes) CardType [0]: 'A' - A-type card searched 'B' - B-type card searched CardType [1]: 'C'
	serialNo [out]	card serial number information. This information includes the length of the serial number and the content of serial number; the length of B-type cards and M1 type card's serial number is 4 bytes; the length of A-type card's serial number is usually 4 bytes, 7 bytes or 10 bytes. SerialNo [0] indicates the length of the serial number, SerialNo [1 ~ 10] to save the serial number (left justified).
	ATS[out]	ATS
	ATSlen[out]	The length of ATS
Return value:	0	Success
	Other values	Fail,refer to the error code below
Note:	After search for a A- type CPU card, the Lib_PiccReset () function must be called, then the card can be exchanged data.	

4.2.4 Command exchange

Prototype:	**int Lib_PiccCommand(APDU_SEND *ApduSend, APDU_RESP *ApduResp);**
Function:	Send APDU formatted data to the card, and receive a response from the card.

Parameters:	apduSend[in]	According to the define of" APDU_SEND" structure in "Commnents" ,send the bytearray in order
	apduResp[out]	According to the define of" APDU_RESP" structure in "Commnents" ,recv the bytearray in order
Return value:	0	Success
	Other values	Fail,refer to the error code below
Note:	only Lib_PiccCheck () wase called successfully, we can call this function; otherwise it will be failed.	

5 Class SCAN

5.1 Constructor and method summary

Constructor:	**scan();**
Method Summary:	**int Lib_ScanOpen();**
	int Lib_ScanClose();
	int Lib_ScanRead(short time_out, BYTE *data);

5.2 Method detail

5.2.1 Open

Prototype:	**int Lib_ScanOpen();**	
Function:	Open the scan head , turn on the light	
Parameter:	None	
Return:	0	success
	others	fail
Note:		

5.2.2 Close

Prototype:	**int Lib_ScanClose();**
Function:	Turn off the power
Parameter:	None

Return:	0	success
	others	fail
Note:		

5.2.3 read

Prototype:	int Lib_ScanRead(short time_out, BYTE *data);	
Function:	Read the 1, 2, 3 track data in the buffer	
Parameters:	Time_out[in]	Timeout
	data[out]	Get the data
Return Value:	0	Brush card error
	others	fail
Note:		

5.2.4 decode

Prototype:	String Lib_ScanDecode(ImageScanner mScanner, Image barcode);	
Function:	Decode the data from Camera preview.	
Parameters:	mScanner [in]	Object for the class "net.sourceforge.zbar.ImageScanner".
	barcode [in]	Object for the class "net.sourceforge.zbar. Image".
Return Value:	null	Failed:No result.
	others string	Success:Result data.
Note:		

6 Class System

6.1 Constructor and method summary

Constructor:	int Sys();
Method Summary:	int Lib_Beep();
	int Lib_ReadSN (byte [] serialNo);
	int Lib_WriteSN (byte [] serialNo);
	int Lib_GetVersion(byte[] buf);
	int Lib_ReadChipID (byte[] buf, int len);
	int Lib_SetAPNFromList(Context context, APNInfo apnInfo);
	List<APNInfo> Lib_GetAPNList(Context context);
	int Lib_AppendAPNList(Context context, APNInfo apnInfo)
	Int Lib_ClearAPNList(Context context)

6.2 Method detail

6.2.1 Beep

Prototype:	int Lib_Beep();	
Function:	Sound beep in fixed-frequency	
Parameter:	none	
Return:	0	success
	-1001	Send failed
	-1002	Recv failed
	Other	failed
Note:	None	

6.2.2 Read SN

Prototype:	int Lib_ReadSN (byte [] SN);	
Function:	Read the SN of the secure module	
Parameters:	serialNo[out]	The buffer address to store serial number, pre-allocate 32 bytes of space
Return Value:	0	success
	-1001	Send failed
	-1002	Recv failed
	Other	failed
Note:		

6.2.3 Write SN

Prototype:	int Lib_WriteSN (byte [] SN);	
Function:	Write the SN to the secure module	
Parameters:	serialNo [in]	The sn to be written, up to 32bytes.
Return Value:	0	Success
	-1001	Send failed
	-1002	Recv failed
	other	Failed
Note:		

6.2.4 Get version

Prototype:	int Lib_GetVersion(byte[] buf);

Function:	Get the version information of secure module and SO,	
Parameters:	buf [out]	The size of buffer should be large than 12,to save version info. * BUF[0~2]:secure module's version(1.0.0) * BUF[3~5]:Security Boot's Version(1.0.0) * BUF[6~8]:App Compile Time(18.02.28) * BUF[9~11]:Lib Version(1.0.0)
Return Value:	0	Success
	-1001	Send failed
	-1002	Recv failed
	other	Failed
Note:		

6.2.5 Read Security Chip ID

Prototype:	int Lib_ReadChipID (byte[] buf, int len);	
Function:		
Parameters:	buf[out]	Chip ID buf, 16bytes
	len[in]	16
Return Value:	0	Success
	-1001	Send failed
	-1002	Recv failed
	other	Failed
Note:		

6.2.6 get the module id

Prototype:	int INFO_GetSysteminfo(byte bid,byte[] baBuf);	
Function:		
Parameters:	bid[in]	0:Get the ID of the contactless smart cards module 1:Get the ID of the clock module 2:Get the ID of the fingerprint module 3:Get the ID of the info module 4:Get the ID of the keyboard module 5:Get the ID of the KMS module 6:Get the ID of the magnetic cards module 7:Get the ID of the printer module 8:Get the ID of the smart cards module 9:Get the ID of the scan module 10:Get the ID of the Secure Data Storage module
	baBuf[out]	The size of buffer should be large than 3,to save module id.
Return Value:	>=0	Success
	other	Failed
Note:		

6.2.7 Get APN list

Prototype:	List<APNInfo> Lib_GetAPNList(Context context)

Function:	Get apn list	
Parameters:	context [in]	Acticity's context
Return Value:		
Note:	The ApnInfo class is included in the Sys class. eg: Sys.ApnInfo = new ApnInfo(...);	

6.2.8 Set Apn From Apn List

Prototype:	int Lib_SetAPNFromList(Context context, APNInfo apnInfo)	
Function:	Set apn	
Parameters:	context [in]	Acticity's context
	apnInfo [in]	Apn info need to set
Return Value:	0	success
Note:	Throws Exception if Sim Operator is failed.	

6.2.9 Add apn to list

Prototype:	int Lib_AppendAPNList(Context context, APNInfo apnInfo)	
Function:	add apn	
Parameters:	context [in]	Acticity's context
	apnInfo [in]	Apn info need to add
Return Value:	0	success
	others	failed
Note:	Throws Exception if Sim Operator is failed.	

6.2.10 Clear apn list

Prototype:	int Lib_ClearAPNList(Context context)	
Function:	clear apn list	
Parameters:	context [in]	Acticity's context
Return Value:	0	success
Note:		

7 Class Printer

7.1 Constructor and method summary

Constructor:	Print();
Method Summary:	int Lib_PrnInit();
	int Lib_PrnSetSpace(byte x, byte y);
	int Lib_PrnSetFont(byte asciiFontHeight, byte

	extendFontHeight, byte zoom);
	int Lib_PrnGetFont(byte asciiFontHeight[], byte extendFontHeight[], byte[] zoom);
	int Lib_PrnStep(int pixel);
	int Lib_PrnStr(byte[] str);
	int Lib_PrnLogo(byte logo[]);
	int Lib_PrnStart();
	int Lib_PrnSetLeftIndent(int x);
	int Lib_PrnSetGray(byte nLevel);
	int Lib_PrnCheckStatus();
	int Lib_PaperConsumptionGetCM(byte[] paperCLen);
	int Lib_PaperConsumptionResetCM();

7.2 Method detail

7.2.1 Initialization

Prototype:	int Lib_PrnInit();	
Function:	Restore printer's default settings and clear the contents of the print buffer	
Parameters:	None	
Return value:	0	Success
	(-4007)	No font library.
Note:		

7.2.2 Set Space

Prototype:	int Lib_PrnSetSpace(byte x, byte y);	
Function:	Set the line spacing and column spacing	
Parameters:	x[in]	Column spacing [dots]. The default spacing is 0, maximum is 255.
	y[in]	Line spacing [dots]. The default spacing is 0, maximum 255
Return value:	0	Success
	others	failed
Note:	① The setting will be effective until the next call Lib_PrnInit();	

7.2.3 Set font

Prototype:	**int Lib_PrnSetFont(byte asciiFontHeight, byte extendFontHeight, byte zoom);**	
Function:	Set the print font and zoom parameters	
Parameters:	**asciiFontHeight[in]**	ASCII character height: PRN_FONT_SMALL (8X16) PRN_FONT_BIG (12X24), the default value is PRN_FONT_BIG Others are illegal values.
	extendFontHeight[in]	Extended character height. PRN_FONT_SMALL (16X16) PRN_FONT_BIG (24X24), the default value is PRN_FONT_BIG Others are illegal values.
	zoom[in]	Font zoom parameters, the default value is 0, which means no zoom in or zoom out. ① PRN_ASCII_X_ENLARGE: ASCII characters enlarge doubled in the X direction; ② PRN_ASCII_Y_ENLARGE: ASCII character enlarge doubled in the Y direction; ③ PRN_EXT_X_ENLARGE: Extended characters enlarge doubled in the X direction; ④ PRN_EXT_Y_ENLARGE: Extended characters enlarge doubuled in the Y direction.
Return value:	0	Success
	(-4009)	Parameters error
Note:	①The setting will be effective until the next call Prn_Init(); ②the following Value private final int PRN_FONT_SMALL = 16 private final int PRN_FONT_BIG = 24 private final int PRN_ASCII_X_ENLARGE = 0x01 private final int PRN_ASCII_Y_ENLARGE = 0x02 private final int PRN_EXT_X_ENLARGE = 0x10 private final int PRN_EXT_Y_ENLARGE = 0x20	

7.2.4 Get font

Prototype:	**int Lib_PrnGetFont(byte asciiFontHeight[], byte extendFontHeight[], byte[] zoom);**
Function:	Get the current font and zoom parameters

Parameters:	asciiFontHeight[out]	ASCII character height: PRN_FONT_SMALL (8X16); PRN_FONT_BIG (12X24);
	extendFontHeight[out]	Extended character height: PRN_FONT_SMALL (16X16); PRN_FONT_BIG (24X24);
	zoom[out]	Font zoom parameters, the default value is 0, which means no zoom in or zoom out. ① PRN_ASCII_X_ENLARGE: ASCII characters enlarge doubled in the X direction; ② PRN_ASCII_Y_ENLARGE: ASCII character enlarge doubled in the Y direction; ③ PRN_EXT_X_ENLARGE: Extended characters enlarge doubled in the X direction; ④ PRN_EXT_Y_ENLARGE: Extended characters enlarge doubuled in the Y direction.
Return value:	0	Success
	others	failed
Note:		

7.2.5 Offset in buffer

Prototype:	int Lib_PrnStep(int pixel);	
Function:	Move to the specified pixel in buffer.	
Parameters:	pixel[in]	The number of pixels to be offset, which could be positive or negative.
Return value:	0	Success
	others	failed
Note:	When users print to note the following points: 1, the pixels to be offset can be positive or negative. Positive to move forward; negative to move backward. 2, assuming that there are already 100 pixels in buffer, then the legitimate range for “pixel” is [-100,4900], out of the range will no action;	

7.2.6 Send string data to print buffer

Prototype:	int Lib_PrnStr(byte[] str);	
Function:	Sent the data to be printed to print buffer. (The system will automatically converts the string to dot matrix data and stored within the print buffer)	
Parameters:	str[in]	The data to be printed
Return value:	0	Success
	(-4008)	The buffer overflow
Note:	1) support variable parameters, please refer to printf () function of standard C; 2) support '\ n' [line], '\ f' [feed] control characters in the buffer; 3) wrap automatically; 4) maximum 2047 bytes for once. 5) Not support the paramenters with "% f" format, otherwise it will cause system crashes, which is due to arm-elf-gcc compiler.	

7.2.7 Send logo dot data to print buffer

Prototype:	int Lib_PrnLogo(byte logo[]);	
Function:	Send the logo dot data to the print buffer	
Parameters:	logo [in]	The logo data to be printed
Return value:	0	Success
	(-4003)	Logo size error
	(-4004)	Decompression error
	(-4008)	The buffer overflow
Note:	The logo to be printed must be in BMP format, the maximum size is 384 * 400.	

7.2.8 Start printing

Prototype:	int Lib_PrnStart();	
Function:	Printer starting to print.	
Parameter:	None	
Return value:	0	Success
	-1001	Send failed
	-1002	Recv failed
	(-4001)	Printer is busy
	(-4002)	Printer is lack of paper
	(-4003)	Print data format error
	(-4004)	Printer is broken
	(-4005)	Printer is too hot
	(-4006)	Print is unfinished
	(-4007)	No print font library

	(-4008)	Print buffer overflow
	（-4011）	Low battery
	Others	Other errors
Note:	① After calling this function, it will return until completed ② After finished, this function will return the state of printer; ③ After calling this function, the print buffer will be cleared;	

7.2.9 Set the left margin

Prototype:	**int Lib_PrnSetLeftIndent(int x);**	
Function:	Set the left margin printing characters	
Parameters:	**x[in]**	Point blank left margin, range: 0 to 336
Return value:	0	Success
	others	failed
Note:	The default boundary is 0	

7.2.10 Set gray

Prototype:	**int Lib_PrnSetGray(byte nLevel);**	
Function:	Set gray	
Parameters:	**nLevel[in]**	The range is: 0 to 7, if nLevel >=7, nLevel = 7; The default value is 1;
Return value:	0	Success
	others	failed
Note:		

7.2.11 Check print status

Prototype:	**int Lib_PrnCheckStatus();**
Function:	Check the current status of printer.
Parameters:	None
Return value:	Refer to **Lib_PrnStart's** return value.
Note:	

7.2.12 Get paper print length

Prototype:	**int Lib_PaperConsumptionGetCM(byte[] paperCLen);**
Function:	Get length of paper printed.

Parameters:	paperCLen [out]	
Return value:	0	Success
	others	failed
Note:		

7.2.12 Reset paper print length

Prototype:	int Lib_PaperConsumptionResetCM ();	
Function:	Reset length of paper printed.	
Parameters:	None	
Return value:	0	Success
	others	failed
Note:		