
POSTEK CDFPSK

API Function Manual

Version 3.1.0

2023

Description of API functions file

Name: CDFPSK_API Function Manual.

Version number: 3.1.0, published in October 2023.

Copyright: ©2023 by Postek Electronics CO., LTD. All rights reserved.

Trademarks

POSTEK is a registered trademark by Postek Electronics Co., Ltd. In this manual, POSTEK specifically refers to Shenzhen Postek Electronics Co., Ltd. Other trademarks or companies mentioned in this manual are the property of their respective owners.

Disclaimer

This API functions provide a series of commands/functions for users of Postek Electronics CO., LTD. All the functions offer convenience for them when they compile the application programs based on the operating system of Windows7, Windows 8, Windows 10 and many more.

This API functions only supports products of Postek Electronics CO., LTD.

Postek Electronics Co., Ltd.

Address: Wisdom Plaza, Block B, Tower 2, 18th Floor Qiaoxiang Road, Nanshan District, Shenzhen, Guang Dong 518052

Web site: <http://www.postekchina.com>

Tel: +86-755-83240988 400-893-3288

Email: Sales - Overseas@postek.com.cn

Technical Support - Tech@postek.com.cn

Table of Content

Description of API functions file	0
Trademarks	0
Disclaimer.....	0
Table of Content	1
Notice	2
Ports	2
CDFPSK.ini configuration file description.....	2
String	2
Dot Size	3
Printer Coordinate System.....	3
Functions	4
Log Functions	4
Port Operations	4
Printer Setup	5
Label Setup	5
Print Text.....	6
Print Graphics	6
Print A Binary Graphic	6
Printing Lines.....	7
Print Barcode	7
Print 2D Barcode	7
Print Forms and Related	8
RFID Tag Read/Write Related.....	8
HF Tag Read/Write Related	9
Detailed Description of Functions	10
Code Conversion	错误!未定义书签。
Port Operation.....	14
Printer settings	24
Label setup.....	50
Print text	57
Print Graphic.....	62
Print bitmap	70
Printing Lines.....	78
Print Barcode	81
Print 2D Barcode	85
Print Forms and Related	92
RFID Label Read/Write Related.....	100
HF label read/write related	112
Appendix	127
Table - Printer Status Code Analysis	127
Table - National or regional frequency bands.....	127
CDFPSK.dll error return value analysis	128

Notice

Ports

1. Before sending commands to the printer, you must first open a set of ports, otherwise all API calls to the printer operations are invalid;
2. Only one set of ports can be opened, and the current port must be closed before opening other ports.

CDFPSK.ini configuration file description

The CDFPSK.ini file configuration function is placed under the Dll path and servers the following purpose:

- logMode=0 Disable logging function.
- logMode=1 Enable the logging function and generate logs in the DLL path.

Note: By default, it is disabled.

- tph=2 Define the printer resolution as 200 dots.
- tph=3 Define the printer resolution as 300 dots.

Note: By default, it is 300 dots and is currently used for adding text below the Code128 Auto barcode, centering, and right alignment.

The following is applicable only supports the MultiByte version of the DLL:

textEncode=0 The input strings for functions are in GBK encoding.

textEncode=1 The input strings for functions are in UTF-8 encoding.

Note: By default, it is UTF-8 encoding.

String

The quotation mark character (") designates the beginning and ending of a string.

The back slash character (\) designates that the character following is a literal and will encode into the data field.

Refer to the following examples:

Character	Enter into Data Field
“	\“
\	\\
0x00 – 0x7F	\x00 - \x7F

-
- All print commands and alpha character command, parameters are case sensitive.
 - <CR> is “13” of the USASCII decimal system, or “0DH” of the hexadecimal system, namely “carriage return” symbol.

Dot Size

Pixels are a unit of measurement in computer science and technology, originally referring to the smallest unit in the imaging of television images. In the field of printers, it represents the printer's minimum print imaging unit. 1 dot is equal to one inch divided by the printer's maximum resolution.

A dot is the smallest unit of printing on a printer. The dot sizes for printers with different resolutions are as follows:

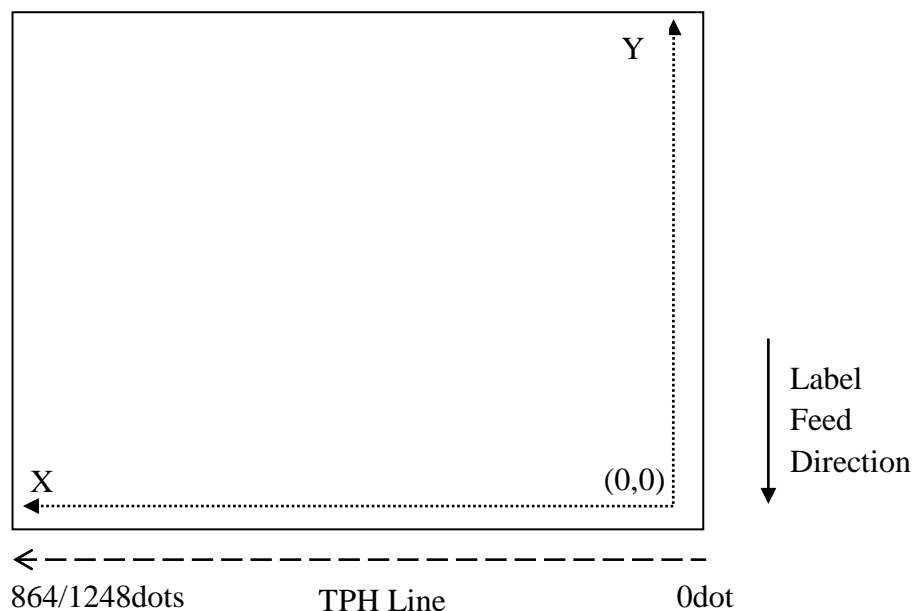
- For 203 DPI printers, $1\text{dot} = 25.4\text{mm}/203 \approx 0.125\text{mm}$ (1dot \approx 5mil)
- For 300 DPI printers, $1\text{dot} = 25.4\text{mm}/300 \approx 0.085\text{mm}$ (1dot \approx 3mil)
- For 600 DPI printers, $1\text{dot} = 25.4\text{mm}/600 \approx 0.042\text{mm}$ (1dot \approx 2mil)

In this manual, API functions are represented in integer conversions:

- For a 203 DPI printer, $1\text{ mm} \approx 8\text{dots}$;
- For a 300 DPI printer, $1\text{ mm} \approx 12\text{dots}$;
- For a 600 DPI printer, $1\text{ mm} \approx 24\text{dots}$.

Printer Coordinate System

The barcode label printer's coordinates system is depicted in following figure:



Functions

Log Functions

Function Name	Description
PTK_GetErrorInfo	Used for error code parsing
PTK_OpenLogMode	Enable logging and specify the log generation path
PTK_CloseLogMode	Disable logging functionality

Note: If both the CDFPSK.ini configuration file and API functions are used to enable or disable the logging function simultaneously, the API function call will take precedence.

Port Operations

Function Name	Description
PTK_GetAllPrinterUSBInfo	Used to retrieve USB port information for all currently connected printers via USB
PTK_OpenUSBPort	Open a USB port
PTK_CloseUSBPort	Close the opened USB port
PTK_OpenSerialPort	Open a serial port
PTK_CloseSerialPort	Close the opened serial port
PTK_OpenPrinter	Open a printer driver
PTK_ClosePrinter	Close the opened printer driver
PTK_OpenParallelPort	Open a parallel port
PTK_CloseParallelPort	Close the opened parallel port
PTK_OpenTextPort	Create a file to which the data sent to the printer will be stored
PTK_CloseTextPort	Close the created file
PTK_Connect	Connect to a printer using a network port
PTK_Connect_Timer	Connect to a printer using a network port with the option to set a connection timeout
PTK_CloseConnect	Disconnect the network connection to the printer
PTK_OpenUSBPort_Buff	Create a buffer where the content to be sent to the printer will be written, and open a USB port
PTK_Connect_Timer_Buff	Create a buffer where the content to be sent to the printer will be written, and open a network port
PTK_OpenPrinter_Buff	Create a buffer where the content to be sent to the printer will be written, and open a driver port
PTK_CloseBuffPort	Release the buffer and close the open port
PTK_WriteBuffToPrinter	Send the content of the buffer to the open port

PTK_SendFile	Send the content of the file to the printer
------------------------------	---

Printer Setup

Function Name	Description
PTK_SetCharSets	Set character encoding
PTK_PrintConfiguration	Command the printer to print configuration
PTK_MediaDetect	Command the printer to perform media sensor calibration
PTK_FeedMedia	Command the printer to feed a label
PTK_UserFeed	Command the printer to feed a fixed length of label
PTK_UserBackFeed	Command the printer to feedback a fixed length of label
PTK_EnableFLASH	Enable flash memory
PTK_DisableFLASH	Disable flash memory
PTK_CutPage	Set the cutting frequency, which takes effect after a restart
PTK_CutPageEx	Set the cutting frequency, which becomes ineffective after a restart
PTK_SetCoordinateOrigin	Set the origin point for printing coordinates
PTK_GetUtilityInfo	Retrieve common configuration information stored in the printer's flash memory
PTK_SetUtilityInfoProc	Configure the printer's common information
PTK_SetUtilityInfo	Send common printer configuration information to save in the printer's flash memory
PTK_GetAllPrinterInfo	Retrieve all configuration information saved in the printer's flash memory
PTK_SetAllPrinterInfo	Set all printer information and save it to flash memory
PTK_ErrorReport_USBInterrupt	Real-time retrieval of the current printer status through USB interrupts

Label Setup

Function Name	Description
PTK_ClearBuffer	Clear the printer's buffer content
PTK_SetPrintSpeed	Set the current label printing speed, which is not saved after a restart
PTK_SetDarkness	Set the current label printing darkness or intensity, which is not saved after a restart
PTK_SetDirection	Set the current label printing direction, which is not saved after a restart
PTK_SetLabelHeight	Set the current label height, gap/black mark/perforation height, and alignment offset, which is not saved after a restart

PTK_SetLabelWidth	Set the current label width, which is not saved after a restart
PTK_PrintLabel	Command the printer to start executing the print job
PTK_PrintLabelFeedback	Command the printer to start executing the print job, and after completion, read the current printer status

Print Text

Function Name	Description
PTK_DrawText	Print text
PTK_DrawTextEx	Print text, serial number, and variables (please use in conjunction with print form functions)
PTK_DrawText_TrueType	Call Windows' font library and print a line of TrueType font
PTK_RenameDownloadFont	Match the fonts downloaded to the printer with the font IDs A to Z used in PTK_DrawText_TrueType

Print Graphics

Function Name	Description
PTK_ChangeIMGtoPCX	Convert the image to PCX format and generate a PCX image with the same name in the same path
PTK_PcxGraphicsList	Print a list of graphic names stored in the printer's RAM or flash memory
PTK_PcxGraphicsDel	Delete stored graphics in the printer
PTK_AnyGraphicsDownload	Download graphics to the printer from a specified file path
PTK_DrawPcxGraphics	Print graphics saved in the printer
PTK_AnyGraphicsPrint	Directly print a graph from a specified file path
PTK_AnyGraphicsDownloadFromMemory	Store graphics in the printer using graphics data
PTK_AnyGraphicsPrintFromMemory	Print graphics using graphics data
PTK_AnyGraphicsPrint_Base64	Print graphics using base64 graphics data

Print A Binary Graphic

Function Name	Description
PTK_BinGraphicsList	Print a list of graphic names stored in the printer's RAM or flash memory
PTK_BinGraphicsDel	Delete binary graphics stored in the printer
PTK_BinGraphicsDownload	Store a binary format graphic in the printer

PTK_RecallBinGraphics	Print binary format graphics stored in the printer
PTK_DrawBinGraphics	Directly define and print binary format graphics

Printing Lines

Function Name	Description
PTK_DrawRectangle	Draw an empty rectangle on the label.
PTK_DrawLineXor	Draw a straight line, and if it intersects, perform an XOR operation
PTK_DrawLineOr	Draw a straight line, and if it intersects, perform an OR operation
PTK_DrawDiagonal	Draw a diagonal line, and if it intersects, perform an OR operation
PTK_DrawWhiteLine	Draw a white line

Print Barcode

Function Name	Description
PTK_DrawBarcode	Print multiple types of one-dimensional barcodes with constant string content
PTK_DrawBarcodeEx	Print multiple types of one-dimensional barcodes with content that can be constant, serial numbers, or variable strings

Print 2D Barcode

Function Name	Description
PTK_DrawBar2D_QR	Print a QR code
PTK_DrawBar2D_QREx	Print a QR code as an image and save it as an image in the printer for printing, supporting older firmware
PTK_DrawBar2D_HANXIN	Print a HanXin code
PTK_DrawBar2D_Pdf417	Print a PDF417 code
PTK_DrawBar2D_Pdf417Ex	Print a PDF417 code as an image, supporting older firmware
PTK_DrawBar2D_MaxiCode	Print a MaxiCode code
PTK_DrawBar2D_DATAMATRIX	Print a DataMatrix code

Print Forms and Related

Function Name	Description
PTK_GetStorageList	Retrieve the names of forms, fonts, or graphics stored in flash memory
PTK_FormList	Print a list of form names stored in the printer's RAM or flash memory
PTK_FormDel	Delete forms stored in the printer
PTK_FormDownload	Notify the printer to start storing a form in the printer
PTK_FormEnd	Notify the printer that the form storage is complete
PTK_ExecForm	Execute a form, which is equivalent to running all the API functions in the form's content once
PTK_DefineVariable	Define a variable in the printer
PTK_DefineCounter	Define a serial number in the printer
PTK_Download	Instruct the printer to assign an initial value to a variable or serial number
PTK_DownloadInitVar	Initialize variables or serial numbers in the order of their definitions
PTK_PrintLabelAuto	Command the printer to start printing labels using a form, and incorporate the use of serial numbers or variables
PTK_FormPrinting	Print a form

RFID Tag Read/Write Related

Function Name	Description
PTK_RFIDCalibrate	UHF Tag and HF Tag Calibration
PTK_RWRFIDLabel	Write data to a UHF RFID tag
PTK_RWRFIDLabelEx	Write data to a UHF RFID tag (without clearing existing data)
PTK_SetRFLabelPWAndLockRFLabel	Set UHF RFID tag password and lock the tag
PTK_EncodeRFIDPC	Write the PC value or EPC header for an RFID tag
PTK_SetRFID	Configure UHF RFID tag printing parameters
PTK_ReadRFIDLabelData	Read UHF RFID tag data
PTK_ReadRFIDLabelDataEx	Read UHF RFID tag data
PTK_RFIDEndPrintLabel	Print a label and then retrieve the printed RFID tag data
PTK_RFIDEndPrintLabelFeedback	Print a label and then retrieve the printed RFID tag data along with the printer's status
PTK_ReadRFIDSetting	After printing the UHF RFID tag, return the data settings
PTK_PrintAndCallback	Print UHF or HF RFID tags and provide data feedback
PTK_SetReadRFIDForwardS	Set the speed at which the label advances to the optimal

peed	Read/write position when reading RFID data
PTK_SetReadRFIDBackSpeed	Set the speed at which the label retracts to the print line when reading RFID data

HF Tag Read/Write Related

Function Name	Description
PTK_RFIDCalibrate	Calibration of UHF RFID tags and HF tags
PTK_RWHFLabel	Write HF tag data
PTK_SetHFRFID	Configure HF tag information
PTK_ReadHFLabelData	Read HF tag data
PTK_ReadHFLabeUID	Read the UID of a HF tag
PTK_ReadHFRFIDSetting	After printing a HF RFID tag, return the data settings
PTK_ReadHFTagDataPrintAuto	During the printing process, first read the data content of a specified block of a HF tag, and then print it on the label
PTK_ReadHFTagUIDPrintAuto	During the printing process, first read the UID of a HF tag, and then print it on the label
PTK_SetHFAFI	Set the value of the AFI for a HF tag
PTK_SetHFDSFID	Set the value of the DSFID for a HF tag
PTK_SetHFEAS	Set the EAS data
PTK_HFDecrypt	Decrypt HF tags (only supported for ISO 14443A protocol)
PTK_LockHFLabel	Lock HF tags (only supported for ISO 14443A protocol)
PTK_LockHFIdentifier	Lock ISO 15693 tag AFI/DSFID
PTK_LockHFBlock	Lock ISO 15693/NTAG blocks
PTK_SetHFKey	Set a key
PTK_SetHFCRCCommand	Set, modify, or lock the CRC password
PTK_SetHFPrivateCommand	Set or modify a private mode password
PTK_LockHFUser	Set user area lock
PTK_SetHFCFG10	Set CFG function parameter 0x10
PTK_SetHFCFG80	Set CFG function parameter 0x80

Detailed Description of Functions

PTK_GetErrorInfo

Description

Used for parsing error codes.

Syntax

```
int _stdcall PTK_GetErrorInfo(int error_n, LPTSTR errorInfo, DWORD infoSize);
```

Parameters

error_n:	The return values for all API functions in this dynamic library
errorInfo:	Used for storing parsed strings
infoSize:	The allocated space size for errorInfo

Returns

0:	OK
-1:	Failed to parse the error code; the error code is not found
4:	InfoSize is too small; errorInfo is insufficient to store the parsed information

Example

```
int nErrorcode = 0;
TCHAR buff[128] = { 0 };
PTK_OpenLogMode(_T("./log.txt"));
nErrorcode = PTK_OpenUSBPort(255);
PTK_GetErrorInfo(nErrorcode, buff, sizeof(buff));
MessageBox(buff);
PTK_PrintLabel(1, 1);    //print a label
PTK_CloseUSBPort ();
PTK_CloseLogMode();
```

PTK_OpenLogMode

Description

Used to enable the logging function and specify the log generation path.

Note:

- If this function is used, and the log mode in CDFPSK.ini is also enabled, the log will be recorded in the path specified by this API function. However, if the log mode in CDFPSK.ini is not disabled, CDFPSK_log.txt will still be generated in the DLL's location.
- If this function is not used, and the log mode in CDFPSK.ini is enabled, the log will be recorded in CDFPSK_log.txt located in the same directory as the DLL.

Syntax

```
int _stdcall PTK_OpenLogMode(LPTSTR filePath);
```

Parameters

filePath: Specify the path for generating the log.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenLogMode(_T("./log.txt"));
PTK_OpenUSBPort(255);
PTK_PrintLabel(1, 1);    // Print a label
PTK_CloseUSBPort ();
PTK_CloseLogMode();
```

[PTK_CloseLogMode](#)

Description

Used to disable the logging function.

Syntax

```
int _stdcall PTK_CloseLogMode(void);
```

Parameters

None

Returns

0 -> OK

Example

```
PTK_OpenLogMode(_T("./log.txt"));
PTK_OpenUSBPort(255);
PTK_PrintLabel(1, 1);    // Print a label
PTK_CloseUSBPort ();
PTK_CloseLogMode();
```

[PTK_GetLastError](#)

Description

Used to retrieve the latest error code from Windows library functions.

Syntax

```
int _stdcall PTK_GetLastError(void);
```

Parameters

None

Returns

The return value of GetLastError().

Reference documentation:

[https://docs.microsoft.com/en-us/previous-versions/aa911366\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/aa911366(v=msdn.10))

<https://docs.microsoft.com/en-us/previous-versions/aa914935%28v%3dmsdn.10%29>

Example

```
int errCode = PTK_GetLastError();
```

PTK_WSAGetLastError

Description

Used to retrieve the latest error code from Windows network-related library functions.

Syntax

```
int _stdcall PTK_WSAGetLastError(void);
```

Parameters

None

Returns

The return value of WSAGetLastError().

Reference documentation:

[https://docs.microsoft.com/en-us/previous-versions/aa915624\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/aa915624(v=msdn.10))

<https://docs.microsoft.com/en-us/previous-versions/aa924071%28v%3dmsdn.10%29>

Example

```
int errCode = PTK_WSAGetLastError();
```

Encoding Conversion

PTK_UTF8toGBK

Description

Convert a UTF-8 string to a GBK string.

Syntax

```
int _stdcall PTK_UTF8toGBK(char *utf8Str, char *gbkStr, int gbkStrCount);
```

Parameters

utf8Str: UTF-8 string content to be converted.
gbkStr: Store the converted GBK string in a buffer.
gbkStrCount: The number of elements that gbkStr can store.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
char str[16] = "Postek2020"; //Chinese windows system input string for GBK encoding
char utf8[48] = { 0 };
char gbk[48] = { 0 };
PTK_GBKtoUTF8(str, utf8, 48);
PTK_UTF8toGBK(utf8, gbk, 48);
```

PTK_GBKtoUTF8

Description

Convert a GBK string to a UTF-8 string.

Syntax

```
int _stdcall PTK_GBKtoUTF8(char* gbkStr, char *utf8Str, int utf8StrCount);
```

Parameters

gbkStr: Content of the GBK string to be converted.
utf8Str: Store the converted UTF-8 string in a buffer.
utf8StrCount: The number of elements that utf8Str can store.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
char str[16] = "Postek2020"; //Chinese windows system input string for GBK encoding
char utf8[48] = { 0 };
PTK_GBKtoUTF8(str, utf8, 48);
```

Port Operations

PTK_GetAllPrinterUSBInfo

Description

Used to retrieve information about the USB ports of all printers connected via USB.

Note: This function supports USB communication and requires a USB connection with the printer, which should be powered on. It does not require opening a port group before calling.

Syntax

```
int _stdcall PTK_GetAllPrinterUSBInfo(LPTSTR USBInfo, DWORD infoSize);
```

Parameters

USBInfo: Used to store USB port information.
infoSize: The size of the USBInfo space.

Returns

0 -> OK

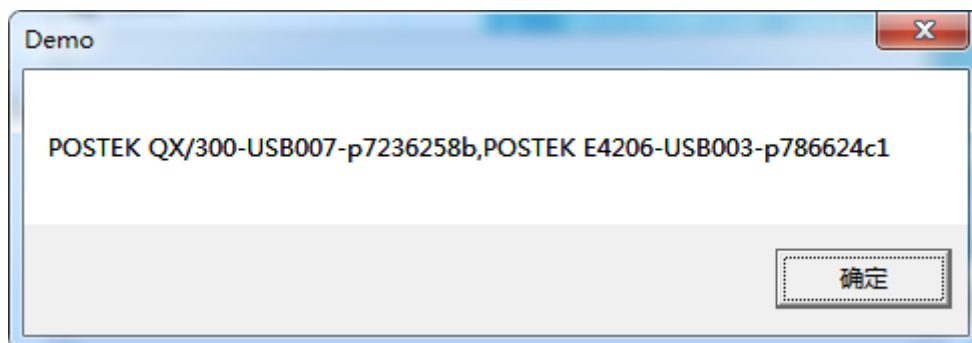
For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR buff[1024] = { 0 };  
PTK_GetAllPrinterUSBInfo(buff, sizeof(buff));  
MessageBox(buff);
```

Example of USBInfo return:

Printer name - USB port number - USB identification number



PTK_OpenUSBPort

Description

Open a USB port.

Syntax

```
int _stdcall PTK_OpenUSBPort(unsigned int port);
```

Parameters

port: USB port number. Value range 1~255.

Note:

- You can use **PTK_GetAllPrinterUSBInfo** to view the USB port information of connected printers.
- Entering 255 will automatically open a connected printer's USB port. If multiple printers are already connected, it will prioritize opening the USB port of the printer with an earlier USB identifier.

Returns

0 -> OK

For other return values, please call **PTK_GetErrorInfo** to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_PrintLabel(1, 1);    // Print a label  
PTK_CloseUSBPort ();
```

[*PTK_CloseUSBPort*](#)

Description

Close the opened USB port.

Syntax

```
int _stdcall PTK_CloseUSBPort(void);
```

Parameters

None

Returns

0 -> OK

Example

```
PTK_OpenUSBPort(255);  
PTK_PrintLabel(1, 1);    // Print a label  
PTK_CloseUSBPort ();
```

[*PTK_OpenSerialPort*](#)

Description

Open a serial port.

Syntax

```
int _stdcall PTK_OpenSerialPort(unsigned int port, unsigned int bRate);
```

Parameters

port: Serial port number, value range 1~255.
bRate: The printer's serial port baud rate, the value of 9600, 19200, 38400, 57600.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenSerialPort(28, 57600);  
PTK_PrintLabel(1, 1);    // Print a label  
PTK_CloseSerialPort();
```

PTK_CloseSerialPort

Description

Close the opened serial port

Syntax

```
int _stdcall PTK_CloseSerialPort(void);
```

Parameters

None

Returns

0 -> OK

Example

```
PTK_OpenSerialPort(28, 57600);  
PTK_PrintLabel(1, 1);    // Print a label  
PTK_CloseSerialPort();
```

PTK_OpenPrinter

Description

Open a printer driver.

Note: The driver port only supports sending data to the printer and cannot be used to read data from the printer.

Syntax

```
int _stdcall PTK_OpenPrinter(LPTSTR printerName);
```

Parameters

printerName: Printer name

Note: You can use **PTK_GetAllPrinterUSBInfo** to view the printer names or check the printer driver names in 'Devices and Printers' on your PC.

Returns

0 -> OK

For other return values, please call **PTK_GetErrorInfo** to parse.

Example

```
PTK_OpenPrinter(_T("POSTEK C168/300s"));
PTK_PrintLabel(1, 1);     // Print a label
PTK_ClosePrinter();
```

PTK_ClosePrinter

Description

Close the opened printer driver.

Syntax

```
int _stdcall PTK_ClosePrinter(void);
```

Parameters

None

Returns

0 -> OK

Example

```
PTK_OpenPrinter(_T("POSTEK C168/300s"));
PTK_PrintLabel(1, 1);     // Print a label
PTK_ClosePrinter();
```

PTK_OpenParallelPort

Description

Open a parallel port.

Syntax

```
int _stdcall PTK_OpenParallelPort(unsigned int port);
```

Parameters

port: Parallel port number, value range 1 to 3.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenParallelPort(1)
PTK_PrintLabel(1, 1);    //Print a label
PTK_CloseParallelPort();
```

[PTK_CloseParallelPort](#)

Description

Close the opened parallel port.

Syntax

```
int _stdcall PTK_CloseParallelPort(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenParallelPort(1)
PTK_PrintLabel(1, 1);    // Print a label
PTK_CloseParallelPort();
```

[PTK_OpenTextPort](#)

Description

Create a file where the data to be sent to the printer will be saved.

Note: Each time it is called, the file will be recreated, overwriting the content from the previous session.

Syntax

```
int _stdcall PTK_OpenTextPort(LPTSTR fn);
```

Parameters

fn: The path where the file is to be created.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenTextPort(_T("command.txt"));
PTK_AnyGraphicsPrint(0, 0, _T("P1"), _T("Koala.jpg"), 1, 0, 0, 0);
PTK_PrintLabel(1, 1);
PTK_CloseTextPort();
```

```
PTK_OpenUSBPort(255);
PTK_SendFile(_T("command.txt"));
PTK_CloseUSBPort();
```

PTK_CloseTextPort

Description

Closes the opened file.

Syntax

```
int _stdcall PTK_CloseTextPort(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenTextPort(_T("command.txt"));
PTK_AnyGraphicsPrint(0, 0, _T("P1"), _T("Koala.jpg"), 1, 0, 0, 0);
PTK_PrintLabel(1, 1);
PTK_CloseTextPort();
```

```
PTK_OpenUSBPort(255);
PTK_SendFile(_T("command.txt"));
PTK_CloseUSBPort();
```

PTK_Connect

Description

Connect to a printer using a network port.

Syntax

```
int _stdcall PTK_Connect(LPTSTR IPAddr, unsigned int netPort);
```

Parameters

IPAddr: IP address of the printer.
netPort: The printer's network port (typically 9100).

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_Connect(_T("199.9.10.245"), 9100);  
PTK_PrintLabel(1, 1);    //Print a label  
PTK_CloseConnect();
```

PTK_Connect_Timer

Description

Connect to a printer using a network port, and you can set the connection timeout.

Syntax

```
int _stdcall PTK_Connect_Timer(LPTSTR IPAddr, unsigned int netPort, unsigned int time_sec);
```

Parameters

IPAddr: IP address of the printer.
netPort: The printer's network port (typically 9100).
time_sec: Set the maximum connection waiting time, in seconds.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_Connect_Timer(_T("199.9.10.245"), 9100, 3); //Returns after 3 seconds of no successful  
connection
```

```
PTK_PrintLabel(1, 1);    //Print a label  
PTK_CloseConnect();
```

PTK_CloseConnect

Description

Disconnect from the printer's network connection.

Syntax

```
int _stdcall PTK_CloseConnect(void);
```

Parameters

None

Returns

0 -> OK

Example

```
PTK_Connect_Timer(_T("199.9.10.245"), 9100);  
PTK_PrintLabel(1, 1);    //Print a label  
PTK_CloseConnect();
```

PTK_OpenUSBPort_Buff

Description

Create a buffer where the content to be sent to the printer will be stored, and open a USB port.

Note: Calling PTK_WriteBuffToPrinter will send the content of the buffer to the open port all at once.

Syntax

```
int _stdcall PTK_OpenUSBPort_Buff(unsigned int portNum);
```

Parameters

port: USB port number. Value range 1 to 255.

Note: You can use PTK_GetAllPrinterUSBInfo to view information about printers connected to USB ports. If you input 255, it will automatically open the USB port of a connected printer. If multiple printers are connected, it will prioritize opening the USB port of the printer with an earlier USB identification number.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUsbPort_Buff(255);  
PTK_DrawText(100, 50, 0, 6, 8, 8, 'N', _T("POSTEK 2023"));  
PTK_PrintLabel(1, 1);  
PTK_WriteBuffToPrinter();  
PTK_CloseBuffPort();
```

PTK_Connect_Timer_Buff

Description

Create a buffer where the content to be sent to the printer will be stored in the buffer, and open a network port.

Note: Calling PTK_WriteBuffToPrinter will send the content of the buffer all at once to the open port.

Syntax

```
int _stdcall PTK_Connect_Timer_Buff(LPTSTR IPAddr, unsigned int netPort, unsigned int time_sec);
```

Parameters

IPAddr:	IP address of the printer.
netPort:	Network port for the printer (typically 9100).
time_sec:	Set the maximum connection waiting time, in seconds.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_Connect_Timer_Buff(_T("199.9.10.233"), 9100, 10);  
PTK_DrawText(100, 50, 0, 6, 8, 8, 'N', _T("POSTEK 2023"));  
PTK_PrintLabel(1, 1);  
PTK_WriteBuffToPrinter();  
PTK_CloseBuffPort();
```

PTK_OpenPrinter_Buff

Description

Create a buffer where the content to be sent to the printer will be stored in the buffer, and open a driver port.

Note: Calling PTK_WriteBuffToPrinter will send the content of the buffer all at once to the open port.

Syntax

```
int _stdcall PTK_OpenPrinter_Buff(LPTSTR printerName);
```

Parameters

printerName:	Printer name.
--------------	---------------

Note: You can use PTK_GetAllPrinterUSBInfo to view the printer names, or open your PC's 'Devices and Printers' to check the printer driver names.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenPrinter_Buff(_T("POSTEK TX6"));
PTK_DrawText(100, 50, 0, 6, 8, 8, 'N', _T("POSTEK 2023"));
PTK_PrintLabel(1, 1);
PTK_WriteBuffToPrinter();
PTK_CloseBuffPort();
```

PTK_CloseBuffPort

Description

Release the buffer and close the opened port.

Syntax

```
int _stdcall PTK_CloseBuffPort(void);
```

Parameters

None

Returns

0 -> OK

Example

```
PTK_OpenPrinter_Buff(_T("POSTEK TX6"));
PTK_DrawText(100, 50, 0, 6, 8, 8, 'N', _T("POSTEK 2023"));
PTK_PrintLabel(1, 1);
PTK_WriteBuffToPrinter();
PTK_CloseBuffPort();
```

PTK_WriteBuffToPrinter

Description

Send the contents of the buffer to the open port.

Note: In the buffer, please refrain from calling API functions that involve reading data, as doing so may result in port blockage

Syntax

```
int _stdcall PTK_WriteBuffToPrinter(void);
```

Parameters

None

Returns

0 -> OK

Example

```
PTK_OpenPrinter_Buff(_T("POSTEK TX6"));
PTK_DrawText(100, 50, 0, 6, 8, 8, 'N', _T("POSTEK 2023"));
PTK_PrintLabel(1, 1);
PTK_WriteBuffToPrinter();
PTK_CloseBuffPort();
```

PTK_SendFile

Description

Send file content to the printer.

Syntax

```
int _stdcall PTK_SendFile(LPTSTR filePath);
```

Parameters

filePath: The path to the file you want to send.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenTextPort(_T("command.txt"));
PTK_AnyGraphicsPrint(0, 0, _T("P1"), _T("Koala.jpg"), 1, 0, 0, 0);
PTK_PrintLabel(1, 1);
PTK_CloseTextPort();
```

```
PTK_OpenUSBPort(255);
PTK_SendFile(_T("command.txt"));
PTK_CloseUSBPort();
```

Printer settings

PTK_SetCharSets

Description

The function PTK_SetCharSets is used to set character sets.

Syntax

int PTK_SetCharSets (unsigned int BitValue, UCHAR CharSets, LPTSTR CountryCode);

Parameters

BitValue:	Data bit value; 8 represents 8-bit code, 7 represents 7-bit code.
CharSets:	Character set.

Bit Value (CharSets =8)	Code page	Country/Region
0	CP437	English - USA
1	CP850	Latin1 - Western European
2	CP852	Latin2 - Central European
3	CP860	Portuguese, Spanish, and Italian.
4	CP863	French Canadian
5	CP865	Nordic
6	CP857	Turkish
7	CP861	Icelandic
8	CP862	Hebrew
9	CP855	Cyrillic
10	CP866	Cyrillic CIS 1
11	CP737	Greek
12	CP851	Greek 1
13	CP869	Greek 2
A	CP1252/ Windows1252	The single-byte character encoding of the Latin alphabet, used by default in older components of Microsoft Windows for English and many European languages, including Spanish, French, German, and Portuguese, is the most commonly used single-byte character encoding in the world.
B	CP1250/ Windows1250	Code page used under Microsoft Windows for representing text in Central and Eastern European languages that use Latin script, such as Polish, Czech, Slovak, Hungarian, Slovenian, Bosnian, Croatian, Serbian (Latin script), Romanian (before the 1993 spelling reform) and Albanian.
C	CP1251/ Windows1251	The code page used under Microsoft Windows covers languages that use the Cyrillic alphabet, such as Russian, Bulgarian, Serbian Cyrillic and other languages. It is the most widely used code for Bulgarian, Serbian and Macedonian.
D	CP1253/ Windows1253	Code page used under Microsoft Windows for writing Greek, now mostly replaced by Unicode.
E	CP1254/ Windows1254	Code page used under Microsoft Windows for writing Turkish, now mostly replaced by Unicode.
F	CP1255/ Windows1255	The code pages used under Microsoft Windows for writing Hebrew have mostly been replaced by Unicode.
G	CP936/ GBK	Chinese, a character set commonly used within China, can write Simplified Chinese, Traditional Chinese, some Japanese, Korean and Russian under Chinese DOS.
U	CP65001/ UTF-8	UTF-8, Unicode character set

CountryCode: Country code for programmable keypad (KDU) when having a keypad connection.

Note: 001, American Standard Keyboard.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
Int return = PTK_SetCharSets (8,'U','001');
```

PTK_PrintConfiguration

Description

Command the printer to print a self-test page.

Syntax

```
int _stdcall PTK_PrintConfiguration(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_PrintConfiguration();  
PTK_CloseUSBPort ();
```

PTK_MediaDetect

Description

Command the printer to perform media calibration.

Syntax

```
int _stdcall PTK_MediaDetect(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_MediaDetect();  
PTK_CloseUSBPort ();
```

PTK_FeedMedia

Description

Command the printer to feed one label.

Syntax

```
int _stdcall PTK_FeedMedia(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_FeedMedia();  
PTK_CloseUSBPort ();
```

PTK_UserFeed

Description

Command the printer to feed a fixed length of label.

Syntax

```
int _stdcall PTK_UserFeed(unsigned int feedLen);
```

Parameters

feedLen: Length of label feed, unit dot

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_UserFeed(360);  
PTK_CloseUSBPort ();
```

PTK_UserBackFeed

Description

Command the printer to feedback the label to a fixed length.

Syntax

```
int _stdcall PTK_UserBackFeed(unsigned int feedLen);
```

Parameter

feedLen: Length of label feedback, unit dot.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_UserBackFeed(360);  
PTK_CloseUSBPort ();
```

PTK_EnableFLASH

Description

Command the printer to open the flash storage for use with storage-related API functions. Data will be stored in the printer's flash memory.

Note: It is used to store forms, graphics, and for specific details, please refer to the relevant API documentation.

Syntax

```
int _stdcall PTK_EnableFLASH(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

Example 1: This function is used to store a form.

```
PTK_OpenUSBPort(255);
    /* Store a form */
PTK_EnableFLASH();           //Enable FLASH storage
PTK_FormDel(_T("F1"));       // Delete forms with the same name to avoid renaming
PTK_FormDownload(_T("F1"));  // Command the printer to begin storing the contents of the form
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("1234567"), FALSE); // Print a text
PTK_FormEnd();               //Tell the printer that form storage is over
PTK_DisableFLASH();          //Disable FLASH storage
PTK_CloseUSBPort();
```

Example 2: This function is used to store a graphic.

```
PTK_OpenUSBPort(255);
PTK_EnableFLASH();           // Enable FLASH storage
PTK_PcxGraphicsDel(_T("P1"));
PTK_AnyGraphicsDownload(_T("P1"), _T("koala.jpg"), 1, 0, 0, 0);
PTK_DisableFLASH();          //Disable FLASH storage
PTK_CloseUSBPort();
```

Note: You can view the list of forms and graphics stored in flash by using the PTK_GetStorageList function.

PTK_DisableFLASH

Description

To close flash storage

Syntax

```
int _stdcall PTK_DisableFLASH(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

Example 1: This function is used to store a form.

```
PTK_OpenUSBPort(255);
    /* Storing a form */
```

```
PTK_EnableFLASH();           // Enable FLASH storage
PTK_FormDel(_T("F1"));       // Delete forms with the same name to avoid renaming
PTK_FormDownload(_T("F1"));  // Command the printer to begin storing the contents of the form
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("1234567"), FALSE); // Print a text
PTK_FormEnd();               //Tell the printer that form storage is over
PTK_DisableFLASH();          //Disable FLASH storage
PTK_CloseUSBPort();
```

Example 2: This function is used to store a graphic.

```
PTK_OpenUSBPort(255);
PTK_EnableFLASH();           // Enable FLASH storage
PTK_PcxGraphicsDel(_T("P1"));
PTK_AnyGraphicsDownload(_T("P1"), _T("koala.jpg"), 1, 0, 0, 0);
PTK_DisableFLASH();          //Disable FLASH storage
PTK_CloseUSBPort();
```

Note: Forms and graphics stored in FLASH can be viewed by calling PTK_GetStorageList.

PTK_CutPage

Description

Set label cutting frequency, effective after restart.

Syntax

```
int _stdcall PTK_CutPage(unsigned int page)
```

Parameters

Page: Set label cutting frequency, how many sheets per cut.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_CutPage(3);
PTK_PrintLabel(6, 1); //Print six labels
PTK_CloseUSBPort();
```

PTK_CutPageEx

Description

Set label cutting frequency, not effective after restart.

Syntax

```
int _stdcall PTK_CutPageEx(unsigned int page)
```

Parameters

Page: Set label cutting frequency, how many sheets per cut.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_CutPageEx(3);
PTK_PrintLabel(6, 1); // Print 6 labels, if cutting mode is enabled, it will cut every three labels.
PTK_CloseUSBPort();
```

PTK_SetCoordinateOrigin

Description

Set the printing origin point.

Syntax

```
int _stdcall PTK_SetCoordinateOrigin(unsigned int px, unsigned int py);
```

Parameters

Px: Horizontal axis coordinate.

Py: Vertical axis coordinate.

Note: Refer to the barcode label printer's coordinate system. When the label printing is in reverse, set the coordinate origin in reverse as well. Refer to PTK_SetDirection for more information.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_SetCoordinateOrigin(100,100);
PTK_DrawTextEx(0, 0, 0, 6, 3, 3, 'N', _T("ABC123"), FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_GetUtilityInfo

Description

Retrieve common settings information saved in the printer's flash memory.

Note: Supports reading only through serial and USB interfaces.

Syntax

```
int _stdcall PTK_GetUtilityInfo(unsigned int infoNum, LPTSTR data, unsigned int dataSize);
```

Parameters

infoNum: The number of the information to be retrieved.

data: The buffer to store the retrieved information.

dataSize: The size of the 'data' space.

infoNum	Printer information represented by infoNum	data returns the length	Data parsing
1	LANGUAGE	1	“0”: CHINESE “1”: ENGLISH
2	PRINT MODE	1	“0”: Thermal “1”: Thermal transfer
3	SENSOR TYPE	1	“0”: TRANSMISSIVE “1”: LOWER REFLECTIVE “2”: UPPER REFLECTIVE Note: DIP switch setting is the standard for small printeres.
4	PRINT DARKNESS	2	“00”: Subject to PTK_SetDarkness setting “01”: Darkness 1 “02”: Darkness 2 “20”: Darkness 20
5	PRINT SPEED	2	“00”: Subject to PTK_SetPrintSpeed setting “01”:Speed 1 ips “02”:Speed 2 ips “03”:Speed 2.5 ips “04”:Speed 3 ips “05”:Speed 3.5 ips “06”:Speed 4 ips “07”:Speed 5 ips “08”:Speed 6 ips “09”:Speed 7 ips “10”:Speed 8 ips Note: Please set the speed according to the maximum speed supported by the model.
6	PRINT DIRECTION	1	“0”: forward, “1”: reverse direction
7	FIRMWARE VERSION	4	Firmware Version: 7.58
8	FIRMWARE ID	11	Firmware ID: 00.8083.909

9	CMD TYPE	1	"0":PPLE "1":PPLZ
11	CUTTER MODE	1	"0": DISABLE "1": ENABLE Note: DIP switch setting is the standard for small printeres. Cutters and peelers cannot be equipped at the same time
12	CUT FREQUENCY	2	"01": Cut once for every 1 print "02": Cut once for every 2 print "99": Cut once for every 99 print
13	BLADE POSITION	2	"00": Blade position offset 0 "01" Blade position offset 1 "99" Blade position offset 99 Note:This parameter is configured by default, without the guidance of professional and technical personnel, it is best not to change the value.Unit dot.
14	PEELER MODE	1	"0": DISABLE "2": ENABLE Note: DIP switch setting is the standard for small printeres. Cutters and peelers cannot be equipped at the same time.
15	BAUD RATE	1	"0": 9600 "1": 19200 "2": 38400 "3": 57600 "4": 115200 Note: DIP switch setting is the standard for small printeres.
16	PARITY BIT	1	"0": Untested
17	DATA BITS	1	"0": 8 Data Bit
18	IP ADDRESS	Maximum 15 digits	IP Address String :"192.168.1.2"
19	SUBNET MASK	Maximum 15 digits	Mask string: "255.255.255.0"
20	GATEWAY	Maximum 15 digits	Gateway String: "192.168.1.1"
21	NETWORK PORT	4	Network port string: "9100"
22	Horizontal Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: Pixels.
23	Vertical Offset	4	Unit dot

			e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: pixels.
24	Tear-off Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: Pixels.
25	Positioning Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: Pixels.
26	Cutting Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: Pixels.
27	Peeler Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: Pixels.
28	Date Time	14 year: 4 bits month: 2 bits date: 2 bits hour: 2 bits minute: 2 bits second: 2 bits	Year, month, day, hour, minute and second For example, on January 22, 2020, 13:54:37 hours. Enter "20200122135437"
29	Total Length Printed	6	Unit: meters If "000050" means 50 meters have been printed.
30	Tear-off Mode Switch	1	"0":Off "1":On Note: DIP switch setting is the standard for small printeres.
31	Current status of the printer	3	See Table - Printer Status Code Analysis

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
enum
{
    POSOriginalInfo = 0,    // The original information contains all the printer information.
    POSLanguage = 1,       // Printer Language: C/E
    POSThermal = 2,        // Printing method: Thermal/Thermal transfer
    POSSensorMode = 3,     // Sensor type:  TRANSMISSIVE /
                           LOWER REFLECTIVE/ UPPER REFLECTIVE

    POSDark = 4,           // Print Darkness: 0-20
    POSSpeed = 5,          //Print Speed: 1-8
    POSDirect = 6,         // Print Direction
    POSVersion = 7,        // Print Firmware Version
    POSPrintNum = 8,       // Printer Code
    POSCMDType = 9,       // Printer Command Type
    POSDPI = 10,           // Printer Resolutions
    POSCUTMode = 11,      // Cutter Mode
    POSCUTPage = 12,      // Cutter Frequency
    POSCUTPos = 13,       // Blade Position
    POSPEELMode = 14,     // Peeling Mode
    POSCOMSpeed = 15,     // Serial Port Rate
    POSCOMParity = 16,    // Serial Port Check Bit
    POSCOMLength = 17,    // Serial Port Data Bits
    POSIP = 18,           // Printer ID
    POSIPMask = 19,       // Printer Mask
    POSGateWay = 20,      // Printer Gateway
    POSPort = 21,         // Printer Network Port
    POSOffsetX = 22,      // Printer Horizontal Offset
    POSOffsetY = 23,      // Vertical Offset
    POSOffsetPaneel = 24, // Tear-off Offset
    POSOffsetTph = 25,    // Positioning Offset
    POSOffsetCut = 26,    // Cutting Offset
    POSOffsetPeel = 27,   // Peeling Offset
    POSDate = 28,         // Printer Current Time
    POSPrintSum = 29,     // Total length of current printer prints
    POSPTearMode = 30,    // Tear-off Mode
    POSStatus = 31,       // Printer Status
};

TCHAR buff[1024] = { 0 };

PTK_OpenUSBPort(255);
PTK_GetUtilityInfo(POSVersion, buff, sizeof(buff));
```

```
MessageBox(buff);  
PTK_CloseUSBPort();
```

PTK_SetUtilityInfoProc

Description

Configure common printer settings.

Note:

- **The information configured by this function will be saved to flash and retained after a restart.**
- **You need to combine PTK_GetUtilityInfo and PTK_SetUtilityInfo for usage.**
- **To take effect, you need to call PTK_SetUtilityInfo.**

Syntax

```
int _stdcall PTK_SetUtilityInfoProc(LPTSTR _G1Info, unsigned int infoNum, LPTSTR info);
```

Parameters

_G1Info: When calling PTK_GetUtilityInfo, you obtain the original information of the printer.

infoNum: The format of the information index to be set is as follows.

Note: Not all information can be set.

info: The format of the printer information to be configured is as follows.

Note: If the length of the input string is incorrect, the configuration sent to the printer will fail.

infoNum	Printer information represented by infoNum	data returns the length	info
1	LANGUAGE	1	"0": CHINESE "1": ENGLISH
2	PRINT MODE	1	"0": Thermal "1": Thermal transfer
3	SENSOR TYPE	1	"0": TRANSMISSIVE "1": LOWER REFLECTIVE "2": UPPER REFLECTIVE Note: DIP switch setting is the standard for small printers.
4	PRINT DARKNESS	2	"00": Subject to PTK_SetDarkness setting "01": Darkness 1 "02": Darkness 2 "20": Darkness 20
5	PRINT SPEED	2	"00": Subject to PTK_SetPrintSpeed setting "01": Speed 1 ips "02": Speed 2 ips "03": Speed 2.5 ips "04": Speed 3 ips "05": Speed 3.5 ips "06": Speed 4 ips

			"07": Speed 5 ips "08": Speed 6 ips "09": Speed 7 ips "10": Speed 8 ips Note: Please set the speed according to the maximum speed supported by the model.
6	PRINT DIRECTION	1	"0": forward, "1": reverse direction
9	CMD TYPE	1	"0": PPPE "1": PPLZ
11	CUT MODE	1	"0": DISABLE "1": ENABLE Note: DIP switch setting is the standard for small printers. Cutters and strippers cannot be equipped at the same time
12	CUT FREQUENCY	2	"01": Cut once for every 1 print "02": Cut once for every 2 print "99": Cut once for every 99 print
13	BLADE POSITION	2	"00": Blade position offset 0 "01" Blade position offset 1 "99" Blade position offset 99 Note: This parameter is configured by default, without the guidance of professional and technical personnel, it is best not to change the value. Unit dot.
14	PEELER MODE	1	"0": DISABLE "2": ENABLE Note: DIP switch setting is the standard for small printers. Cutters and peelers cannot be equipped at the same time.
15	BAUD RATE	1	"0": 9600 "1": 19200 "2": 38400 "3": 57600 "4": 115200 Note: DIP switch setting is the standard for small printers.
18	IP ADDRESS	Maximum 15 digits	IP Address String :"192.168.1.2"
19	SUBNET MASK	Maximum 15 digits	Mask string: "255.255.255.0"
20	GATEWAY	Maximum 15 digits	Gateway String: "192.168.1.1"
21	NETWORK PORT	4	Network port string: "9100"
22	Horizontal Offset	4	Unit dot

			e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: pixels.
23	Vertical Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: pixels.
24	Tear-off Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: pixels.
25	Positioning Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: pixels.
26	Cutting Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: pixels.
27	Peeler Offset	4	Unit dot e.g. "0012", 1mm on a 300DPI machine. e.g. "0024", 3mm on a machine with a resolution of 203DPI. Note: For conversion, please refer to Dots: pixels.
28	Date Time	14 year: 4 bits month: 2 bits date: 2 bits hour: 2 bits minute: 2 bits second: 2 bits	Year, month, day, hour, minute and second For example, on January 22, 2020, 13:54:37 hours. Enter "20200122135437"
30	Tear-off Mode Switch	1	"0": Off "1": On Note: DIP switch setting is the standard for

			small printeres.
--	--	--	------------------

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

enum

```
{
    POSLanguage = 1,          // Printer Language: C/E
    POSThermal = 2,          // Printing method: Thermal/Thermal transfer
    POSSensorMode = 3,       // SENSOR TYPE: TRANSMISSIVE/LOWERREFLECTIVE/
                             UPPER REFLECTIVE

    POSDark = 4,             // Print Darkness: 0-20
    POSSpeed = 5,            // Print Speed: 1-8
    POSDirect = 6,           // Print Direction
    POSCMDType = 9,          // Printer Command Type
    POSCUTMode = 11,         // Cutter Mode
    POSCUTPage = 12,         // Cutter Frequency
    POSCUTPos = 13,          // Blade Position
    POSPEELMode = 14,        // Peeler Mode
    POSCOMSpeed = 15,        // Serial Port Rate
    P POSIP = 18,            // Printer ID
    POSIPMask = 19,          // Printer Mask
    POSGateWay = 20,         // Printer Gateway
    POSPort = 21,            // Printer Network Port
    POSOffsetX = 22,         // Printer Horizontal Offset
    POSOffsetY = 23,         // Vertical Offset
    POSOffsetPaneel = 24,    // Tear-off Offset
    POSOffsetTph = 25,       // Positioning Offset
    POSOffsetCut = 26,       // Cutting Offset
    POSOffsetPeel = 27,      // Peeler Offset
    POSDate = 28,            // Printer Current Time
    POSPTearMode = 30,       // Tear-off Mode
};
```

```
TCHAR buff[1024] = { 0 };
```

```
PTK_OpenUSBPort(255);
```

```
PTK_GetUtilityInfo(0, buff, sizeof(buff));
```

```
PTK_SetUtilityInfoProc(buff, POSDark, _T("08"));    // Set the print darkness to 8
```

```
PTK_SetUtilityInfoProc(buff, POSSpeed, _T("05"));    // Set the print speed to 3.5ips
```

```
PTK_SetUtilityInfoProc(buff, POSDirect, _T("1"));    // Set the print direction to reverse
```

```
PTK_SetUtilityInfo(buff);
```

```
PTK_CloseUSBPort();
```

PTK_SetUtilityInfo

Description

Save common printer configuration information to the printer's flash memory.

Note: This function will wait for the printer to finish saving before returning.

Syntax

```
int _stdcall PTK_SetUtilityInfo(LPTSTR _G1Info);
```

Parameters

_G1Info: The printer configuration information obtained after using PTK_GetUtilityInfo and PTK_SetUtilityInfoProc.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
enum
{
    POSLanguage = 1,          // Printer Language C/E
    POSThermal = 2,           // Printing method: Thermal/Thermal transfer
    POSSensorMode = 3,        // SENSOR TYPE: TRANSMISSIVE /
                                LOWER REFLECTIVE/ UPPER REFLECTIVE

    POSDark = 4,              // Print Darkness: 0-20
    POSSpeed = 5,             // Print Speed: 1-8
    POSDirect = 6,            // Print Direction
    POSCMDType = 9,           // Printer Command Type
    POSCUTMode = 11,          // Cutter Mode
    POSCUTPage = 12,          // Cutter Frequency
    POSCUTPos = 13,           // Blade Position
    POSPEELMode = 14,         // Peeler Mode
    POSCOMSpeed = 15,         // Serial Port Rate
    P POSIP = 18,             // Printer ID
    POSIPMask = 19,           // Printer Mask
    POSGateWay = 20,          // Printer Gateway
    POSPort = 21,             // Printer Network Port
    POSOffsetX = 22,          // Printer Horizontal Offset
    POSOffsetY = 23,          // Vertical Offset
    POSOffsetPaneel = 24,     // Tear-off Offset
    POSOffsetTph = 25,        // Position Offset
    POSOffsetCut = 26,        // Cutting Offset
}
```

```

        POSOffsetPeel = 27,      // Peeler Offset
        POSDate = 28,          // Printer Current Time
        POSPTearMode = 30,     // Tear-off Mode
    };

```

```

TCHAR buff[1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_GetUtilityInfo(0, buff, sizeof(buff));
PTK_SetUtilityInfoProc(buff, POSDark, _T("08"));    // Set the print darkness to 8
PTK_SetUtilityInfoProc(buff, POSSpeed, _T("05"));  // Set the print speed to 3.5ips
PTK_SetUtilityInfoProc(buff, POSDirect, _T("1"));   // Set the print direction to reverse
PTK_SetUtilityInfo(buff);
PTK_CloseUSBPort();

```

PTK_GetAllPrinterInfo

Description

Retrieve all stored settings from the printer's flash memory.

Note: This function is currently only supported on printers with firmware versions 7.58 and above, and can be used to read printer settings via serial ports, USB connections, and network connections

Syntax

```
int _stdcall PTK_GetAllPrinterInfo(int infoNum, BOOL fileflag, LPTSTR data, DWORD dataSize)
```

Parameters

infoNum: The serial number of the printer information you want to retrieve.

fileflag: This parameter specifies whether to create a file in the current path to save the information. "FALSE" indicates "no," and "TRUE" indicates "yes."

Note: Selecting "TRUE" will generate a "PrinterConfig.txt" file in the DLL's directory to save the information.

data: The buffer for storing the printer configuration information.

dataSize: The size of the 'data' buffer where the printer configuration information is stored.

infoNum	Printer information represented by infoNum	Data parsing
0	All Information	Return data format: Refer to the example data provided. START indicates the beginning of the data, \r\n (line break) is the spacer, CRC is the CRC check code of all the data, and END\r\n indicates the end of the data;
1	Printer Name	Printer Model Name
2	LANGUAGE	"0": CHINESE

		“1”: ENGLISH
3	BAUD RATE	“0”: 9600 “1”: 19200 “2”: 38400 “3”: 57600 “4”: 115200 Note: DIP switch setting is the standard for small machines.
4	PARITY BIT	“N”: untested
5	DATA BITS	“8”: 8 Data Bit
6	Serial port stop bit	"1": 1 stop bit
7	FIRMWARE VERSION	Firmware Version: 7.58
8	FIRMWARE ID	Firmware ID: 00.8083.909
9	CMD TYPE	“0”: PPLE “1”: PPLZ
10	HF labeling protocol types	"0": None "1": ISO 15693 protocol "2": ISO 14443A protocol
11	Total printer FLASH capacity	Unit:MB For example, "32" means the capacity of FLASH is 32MB.
12	Total printer RAM capacity	Unit:MB For example, "32" means RAM capacity is 32MB.
13	Number of labels printed	Unit: sheets For example, "7425" means that 7425 labels have been printed.
14	Printed Label Length	Unit: meters For example, "125" means the length of the printed label is 125 meters.
15	Ribbon Detection Switch	"0" : OFF "1" : ON
16	Tear-off Mode Switch	"0" : OFF "1" : ON
17	Tear-off Offset	Unit:dot e.g. "12" means 12dot offset of the tearing position. Note: For conversion, please refer to Dots: pixels.
18	Cutter Mode Switch	"0" : OFF "1" : ON
19	Cutting Offset	Unit:dot e.g. "12" means 12dot offset of the tearing position. Note: For conversion, please refer to Dots: pixels.
20	CUT FREQUENCY	Unit:/sheet For example, "3" means cut once every three prints.

21	BLADE POSITION	Unit : ms For example, "25" means that the cutter stop position is offset by 25ms.
22	Peeler Mode Switch	"0" : OFF "1" : ON
23	Peeling stop position offset	Unit:dot e.g. "12" means 12dot offset of the tearing position. Note: For conversion, please refer to Dots: pixels.
24	Print Darkness	Range 0~20, the more the value is played, the blacker the printing effect is. Such as "8" on behalf of the print blackness of 8
25	Print Speed	Unit: ips, accurate to 1 decimal place For example, "2.5" means the printing speed is 2.5 ips. "3.0" means the print speed is 3.0 ips.
26	Print Mode	"0": Thermal "1": Thermal Transfer
27	Sensor Type	"0": TRANSMISSIVE "1": LOWER REFLECTIVE "2": UPPER REFLECTIVE
28	Maximum Fees length for media calibration	Unit mm e.g. "200" means that 200mm label will be taken away during media calibration.
29	Print Direction	"0":forward, i.e., the coordinate system of the barcode label printer "1": Reverse direction
30	Vertical Offset	Unit dot e.g. "12", 1mm on a machine with a resolution of 300 DPI e.g. "24", 3mm on a 203DPI machine. Note: For conversion, please refer to Dots: pixels.
31	Horizontal Offset	Unit dot e.g. "12", 1mm on a machine with a resolution of 300 DPI e.g. "24", 3mm on a 203DPI machine. Note: For conversion, please refer to Dots: pixels.
32	Positioning Offset	Unit dot e.g. "12", 1mm on a machine with a resolution of 300 DPI e.g. "24", 3mm on a 203DPI machine. Note: For conversion, please refer to Dots: pixels.
33	RFID Function Switch	"0": OFF "1": ON
34	RFID Write Power	Unit db

		For example, "25" means that the power is 25db.
35	RFID Read Power	Unit db For example, "18" means write power is 18db.
36	RFID Offset	Unit dot e.g. "12", 1mm on a machine with a resolution of 300 DPI e.g. "24", 3mm on a 203DPI machine. Note: For conversion, please refer to Dots: pixels.
37	RFID read/write failure retry count	Unit: times Such as "3", represents the RFID tag reading and writing failure and then retry up to 3 times.
38	RFID Password Function Switch	"0" : OFF "1" : ON
39	RFID R6 Permanent Lock Switch	"0" : OFF "1" : ON
40	RFID Frequency Band Selection	Returns the country or region abbreviation Region Name, see appendix for details Table - Country or Region Bands Defaults to "PRC".
41	Wireless Type	"none": None "bluetooth": Bluetooth "wifi": Wifi
42	MAC Address	Hexadecimal address, space separated For example, "00 1B E7 0B 58 45".
43	DHCP Switch	"0" : OFF "1" : ON
44	IP Address	The IP address of the printer, e.g. "199.9.10.245".
45	Subnet Mask	The subnet mask of the printer, e.g. "255.255.255.0".
46	Gateway	The printer's gateway to the network, e.g., "199.9.10.1".
47	Network Port	Network port of the printer, e.g. "9100".
48	Current status of the printer	See Table - Printer Status Code Analysis
49	Graphic Resolution Conversion	"N": Not turned on "S": 600 points -> 300 points, the graph will be reduced to a quarter of its original size "B": 300 dots -> 600 dots, the graphic will be enlarged to four times of the original Remarks: Not support the machine whose print head resolution is 200 dots.
50	Overall Scaling	"N": Not turned on "S": The whole label content is reduced to one quarter of the original, only support

		300-dot printer "B": The whole label content is enlarged to four times the original, only support 600-dot printer
51	USBID	USB serial number of the printer, unique for each printer

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR buff[1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_GetAllPrinterInfo(0, FALSE, buff, sizeof(buff));
MessageBox(buff);
PTK_CloseUSBPort();
```

PTK_SetAllPrinterInfo

Description

Set all printer information, save to FLASH (corresponds to PTK_GetPrinterInfo).

Note: This function is currently only supported on printers with firmware version 7.58 or higher.

Syntax

```
int _stdcall PTK_SetAllPrinterInfo(int infoNum, LPTSTR printerInfo);
```

Parameters

infoNum: The serial number of the printer information to be obtained, some fixed values are not allowed to be set.

printerInfo: The value to be set, in the following format.

infoNum	Printer information represented by infoNum	PrinterInfo Value
2	LANGUAGE	"0": CHINESE "1": ENGLISH
3	BAUD RATE	"0": 9600 "1": 19200 "2": 38400 "3": 57600 "4": 115200 Note: DIP switch setting is the standard for small machines.
9	CMD TYPE	"0": PPLE

		“1”: PPLZ
10	HF labeling protocol types	"0": None "1": ISO 15693 protocol "2": ISO 14443A protocol
13	Number of labels printed	Unit: sheets For example, "7425" means that 7425 labels have been printed.
14	Printed Label Length	Unit: meters For example, "125" means the length of the printed label is 125 meters.
15	Ribbon Detection Switch	"0": OFF "1": ON
16	Tear-off Mode Switch	"0": OFF "1": ON
17	Tear-off Offset	Unit:dot e.g. "12" means 12dot offset of the tearing position. Note: For conversion, please refer to Dots: pixels.
18	Cutter Mode Switch	"0": OFF "1": ON
19	Cutting Offset	Unit:dot e.g. "12" means 12dot offset of the tearing position. Note: For conversion, please refer to Dots: pixels.
20	CUT FREQUENCY	Unit:/sheet For example, "3" means cut once every three prints.
21	BLADE POSITION	Unit:ms For example, "25" means that the cutter stop position is offset by 25ms.
22	Peeler Mode Switch	"0": OFF "1": ON
23	Peeling stop position offset	Unit:dot e.g. "12" means 12dot offset of the tearing position. Note: For conversion, please refer to Dots: pixels.
24	Print Darkness	Range 0~20, the more the value is played, the blacker the printing effect is. Such as "8" on behalf of the print blackness of 8
25	Print Speed	Unit: ips, accurate to 1 decimal place For example, "2.5" means the printing speed is 2.5 ips. "3.0" means the print speed is 3.0 ips.
26	Print Mode	"0": Thermal "1": Thermal Transfer
27	Sensor Type	“0”: TRANSMISSIVE

		"1": LOWER REFLECTIVE "2": UPPER REFLECTIVE
28	Maximum Fees length for media calibration	Unit mm e.g. "200" means that 200mm label will be taken away during media calibration.
29	Print Direction	"0": Forward, i.e., the coordinate system of the barcode label printer "1": Reverse direction
30	Vertical Offset	Unit dot e.g. "12", 1mm on a machine with a resolution of 300 DPI e.g. "24", 3mm on a 203DPI machine. Note: For conversion, please refer to Dots: pixels.
31	Horizontal Offset	Unit dot e.g. "12", 1mm on a machine with a resolution of 300 DPI e.g. "24", 3mm on a 203DPI machine. Note: For conversion, please refer to Dots: pixels.
32	Positioning Offset	Unit dot e.g. "12", 1mm on a machine with a resolution of 300 DPI e.g. "24", 3mm on a 203DPI machine. Note: For conversion, please refer to Dots: pixels.
33	RFID Function Switch	"0": OFF "1": ON
34	RFID Write Power	Unit db For example, "25" means that the power is 25db.
35	RFID Read Power	Unit db For example, "18" means write power is 18db.
36	RFID Offset	Unit dot e.g. "12", 1mm on a machine with a resolution of 300 DPI e.g. "24", 3mm on a 203DPI machine. Note: For conversion, please refer to Dots: pixels.
37	RFID read/write failure retry count	Unit: times Such as "3", represents the RFID tag reading and writing failure and then retry up to 3 times.
38	RFID Password Function Switch	"0": OFF "1": ON
39	RFID R6 Permanent Lock Switch	"0": OFF "1": ON
40	RFID Frequency Band Selection	Returns the country or region abbreviation Region Name, see appendix for details

		Table - Country or Region Bands Defaults to "PRC".
41	Wireless Type	"none":None "bluetooth":Bluetooth "wifi":Wifi
43	DHCP Switch	"0": OFF "1": ON
44	IP Address	The IP address of the printer, e.g. "199.9.10.245".
45	Subnet Mask	The subnet mask of the printer, e.g. "255.255.255.0".
46	Gateway	The printer's gateway to the network, e.g., "199.9.10.1".
47	Network Port	Network port of the printer, e.g. "9100".
49	Graphic Resolution Conversion	"N": Not turned on "S": 600 points -> 300 points, the graph will be reduced to a quarter of its original size "B": 300 dots -> 600 dots, the graphic will be enlarged to four times of the original Remarks:Not support the machine whose print head resolution is 200 dots.
50	Overall Scaling	"N": Not turned on "S": The whole label content is reduced to one quarter of the original, only support 300-dot printer "B": The whole label content is enlarged to four times the original, only support 600-dot printer

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_SetAllPrinterInfo(2,_T("C"));    // Setting the printer language to Chinese
PTK_CloseUSBPort();
```

PTK_ErrorReport_USBInterrupt

Description

Real-time access to the current status of the printer through the USB interrupt, see Table - Printer Status Code Analysis.

Syntax

```
int _stdcall PTK_ErrorReport_USBInterrupt(int* status);
```

Parameters

status: Integer buffer for status code, see Table - Printer Status Code Parsing for value parsing.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
int status = 0;
PTK_OpenUSBPort(255);
PTK_ErrorReport_USBInterrupt(&status);
PTK_CloseUSBPort ();
```

Label setup

PTK_ClearBuffer

Description

Clear the buffered contents of the printer.

Note:

- Before sending a new job to the printer, it is recommended that you use this command to clear the printer's cache of existing data to prevent the printer from having residual data that can lead to the failure of the API function.
- Please do not use this function in the process of storing the form, otherwise the form will fail to build.

Syntax

```
int _stdcall PTK_ClearBuffer(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_DrawTextEx(100, 100, 0, 6, 3, 3, 'N', _T("ABC123"), FALSE);
PTK_PrintLabel(1, 1);
```

PTK_CloseUSBPort();

PTK_SetPrintSpeed

Description

Set the current label printing speed, without saving after a restart.

Note:

- **The maximum printing speed of different printer models may vary (please refer to the respective user manual). If the set value is greater than the printer's maximum printing speed, the setting will be invalid.**
- **This API setting will be ineffective if the printer's printing speed has been set to a value other than 0 through flash settings, such as the LCD screen.**
- **The PTK_EnableFLASH function is only used for saving forms and graphics, and is not relevant to this API.**

Syntax

int _stdcall PTK_SetPrintSpeed(unsigned int speed);

Parameters

Speed: Specifies the desired print speed. The valid value ranges and their corresponding speeds depend on your printer's firmware format and version. Please refer to the appropriate range based on your printer's firmware.

p1	Speed
2	2.0 ips (50.80 mm/s)
/	2.5 ips (63.50 mm/s)
3	3.0 ips (76.20 mm/s)
/	3.5 ips (88.90 mm/s)
4	4.0 ips (101.60 mm/s)
5	5.0 ips (127.00 mm/s)
6	6.0 ips (152.40 mm/s)
7	7.0 ips (177.80 mm/s)
8	8.0 ips (203.20 mm/s)
9	9.0 ips (228.60 mm/s)
10	10.0 ips (254.00 mm/s)
11	11.0 ips (279.40 mm/s)
12	12.0 ips (304.80 mm/s)
13	13.0 ips (330.20 mm/s)
14	14.0 ips (355.60 mm/s)
15	15.0 ips (381.00 mm/s)
16	16.0 ips (406.40 mm/s)
17	17.0 ips (431.80 mm/s)
18	18.0 ips (457.20 mm/s)
19	19.0 ips (482.60 mm/s)
20	20.0 ips (508.00 mm/s)

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetPrintSpeed(3);
PTK_DrawTextEx(100, 100, 0, 6, 3, 3, 'N', _T("ABC123"), FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_SetDarkness

Description

To set the current label's print darkness without saving after a restart.

Note:

- If you don't explicitly set the print darkness using the PTK_Darkness function, the default darkness level for most printers is typically 10.
- If you've set the print darkness for the printer through flash and it's not 0, this API setting is invalid.
- PTK_EnableFLASH function is only used for saving forms and graphics; it is not effective for this API.

Syntax

```
int _stdcall PTK_SetDarkness(unsigned int dark);
```

Parameters

dark: The label printing darkness, with a range of 0 to 20. A higher value will result in darker and more pronounced printing.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetDarkness(12);
PTK_DrawTextEx(100, 100, 0, 6, 3, 3, 'N', _T("ABC123"), FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_SetDirection

Description

If the printer direction has been set using flash, the direction set by this API function will take precedence until a restart.

Note:

- **If the printer direction is set through FLASH, the direction set by this API function will prevail before power down.**
- **PTK_EnableFLASH function is only used for saving forms and graphics; it is not effective for this API.**

Syntax

```
int _stdcall PTK_SetDirection(TCHAR direct);
```

Parameters

direct: Direction: B (Bottom) or T (Top).
B -> Printing starts from the bottom right corner of the label.
T -> Printing starts from the top left corner of the label, referring to the coordinate system of the barcode label printer.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_ClearBuffer();  
PTK_SetDirection('B');  
PTK_DrawTextEx(100, 100, 0, 6, 3, 3, 'N', _T("ABC123"), FALSE);  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

PTK_SetLabelHeight

Description

Set the current label height, gap/black mark/notch height, and label offset, without saving after a restart.

Note: If not set, the printer will use the calibrated label height. PTK_EnableFLASH function is only used to save forms, graphics, this API is invalid.

Syntax

```
int _stdcall PTK_SetLabelHeight(unsigned int lheight, unsigned int gapH, int gapOffset, BOOL bFlag);
```

Parameters

lheight: The label height in dots, with a range of 0 to 65535.

gapH:	<p>The height of the gap, black mark, or hole between labels, measured in dots. The range for this value is 0 to 65535.</p> <p>The value of gapH depends on the label's positioning mode:</p> <p>Gap Mode: In this mode, gapH is set to the height of the gap. Perforation positioning belongs to the special case of gap mode, also belongs to the gap mode.</p> <p>Black Line Mode: In this mode, gapH is set to the height of the black line.</p> <p>Continuous Mode: In this mode, gapH is set to 0. In this case, the media sensor only checks if there's label left.</p>
apOffset:	<p>The value of the locOffset parameter represents the offset of label gap/black line/punch hole positioning in dots. It is specified in dots as well.</p>
bFlag:	<p>The gapOffset parameter specifies whether the gap offset is effective, TRUE - valid; FALSE - invalid</p>

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetLabelHeight(160, 24, 0, FALSE);
PTK_DrawTextEx(100, 100, 0, 6, 3, 3, 'N', _T("ABC123"), FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_SetLabelWidth

Description

Set the current label width, does not save after restart.

Note: PTK_EnableFLASH function is only used to save forms, graphics, this API is invalid.

Syntax

```
int _stdcall PTK_SetLabelWidth(unsigned int lwidth);
```

Parameters

lwidth: The width of the label in dots.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetLabelWidth(1228);
PTK_DrawTextEx(100, 100, 0, 6, 3, 3, 'N', _T("ABC123"), FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_PrintLabel

Description

Command the printer to start printing label content.

Note: After setting up the print content, you need to call this function to start printing.

Syntax

```
int _stdcall PTK_PrintLabel(unsigned int number, unsigned int cnumber);
```

Parameters

number: Number of labels to print, ranging from 1 to 65535.

cnumber: Number of copies for each label, ranging from 1 to 65535.

Note: The cnumber parameter can be used for serialization, as shown in the example.

Returns

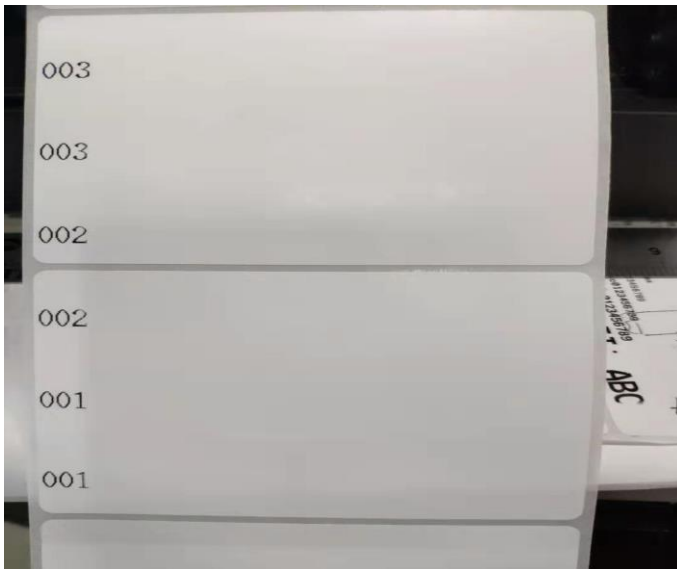
0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetLabelHeight(200, 0, 0, FALSE); // Set labels as continuous label with a height of 200dots
PTK_DefineCounter(0, 6, 'N', _T("+1"), _T("\nEnter\Code:")); // Define serial number C0
PTK_DrawTextEx(20, 20, 0, 6, 2, 2, 'N', _T("C0"), TRUE); // Print C0
PTK_Download();
PTK_DownloadInitVar(_T("001"));        // Initialize C0
PTK_PrintLabel(3, 2);                   // Start printing labels, print 3 sheets, copy each sheet 2 times
PTK_CloseUSBPort();
```

Printing results:



PTK_PrintLabelFeedback

Description

Command the printer to start printing label content and, after printing, read the current printer status.

Note: This function only supports firmware versions V7.60 and above. Note that it cannot be used simultaneously with PTK_PrintLabel.

Syntax

```
int _stdcall PTK_PrintLabelFeedback(LPTSTR data, DWORD dataSize);
```

Parameters

data: Buffer to store the printer status code.

Note: The returned format is "W1xxxx," where "xxxx" is the printer's status code.

dataSize: The "data" variable is the space allocated for storing the printer status code.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR buff[1024] = { 0 };  
PTK_OpenUSBPort(255);  
PTK_PrintLabelFeedback(buff, sizeof(buff));  
MessageBox(buff);  
PTK_CloseUSBPort();
```

Print text

PTK_DrawText

Description

To print a line of text

Note: Finally, call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawText(unsigned int px, unsigned int py, unsigned int pdirec, unsigned int pFont, unsigned int pHorizontal, unsigned int pVertical, TCHAR pColor, LPCTSTR pstr);
```

Parameters

px: Set the X coordinate in dots.

py: Set the Y coordinate in dots.

pdirec: Select the print direction of the text.
0 - No rotation; 1 - Rotate 90°; 2 - Rotate 180°; 3 - Rotate 270°.

pFont: Select the built-in font or soft font.
1-5: These are the 5 built-in Western dot matrix fonts in the printer.
6: This represents the single built-in Chinese font in the printer.
'A'-'Z': This represents the downloaded soft font (downloaded via utility software).

Note: When using the Windows dynamic library with functions like PTK_DrawText_TrueType or PTK_DrawText_TrueTypeEx, you can directly use the fonts located in the C:\Windows\Fonts directory without the need for downloading them separately.

Value	Description
1	Foreign language fonts1
2	Foreign language fonts2
3	Foreign language fonts3
4	Foreign language fonts4
5	Foreign language fonts5
6	24*24 dot matrix Chinese
'A'~'Z'	Truetype fonts

pHorizontal: When pFont is set to an internal font (1 to 6), pHorizontal represents the point enlargement factor. You can choose from 1 to 24. When pFont selects a TrueType font (A to Z), pHorizontal represents the font's width, measured in pixels (no size limit).

pVertical: When pFont is set to an internal font (1 to 6), pVertical is used to set the dot matrix vertical magnification factor. You can choose from 1 to 24. When pFont is set to a TrueType font (A to Z), pVertical is used to set the font height, measured in pixel(no size limit).

pColor: Select 'N' corresponding to the ASCII value 78 to print regular text (e.g., black text on a white background). Select 'R,' corresponding to the ASCII

value 82, to print inverted text (e.g., white text on a black background).

pstr: A string with a length of 1 to 100 characters.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_DrawText(20, 20, 0, 6, 2, 2, 'N', _T("Print test text 6, ABC123"));
PTK_RenameDownloadFont(1, 'A', _T("arial")); // You need to download the arial font through
utility before you can use it, and name arial as the printer's A font.
```

```
PTK_DrawText(20, 120, 0, 'A', 48, 48, 'N', _T("TrueType"));
PTK_PrintLabel(1, 1); // Start printing labels
PTK_CloseUSBPort();
```

PTK_DrawTextEx

Description

Printing a line of text, a serial number, and variables.

Note: Please use it in conjunction with the form-related functions. Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawTextEx(unsigned int px, unsigned int py, unsigned int pdirec, unsigned int
pFont, unsigned int pHorizontal, unsigned int pVertical, TCHAR pColor, LPCTSTR pstr, BOOL
Variable);
```

Parameters

px: Set the X coordinate in dots.
py: Set the Y coordinate in dots.
pdirec: Select the print direction of the text.
0 - No rotation; 1 - Rotate 90°; 2 - Rotate 180°; 3 - Rotate 270°.

pFont: Selects the built-in font or soft font.
1-5: These are the 5 built-in Western dot matrix fonts in the printer.
6: This represents the single built-in Chinese font in the printer.

Note: When using the Windows dynamic library with functions like PTK_DrawText_TrueType or PTK_DrawText_TrueTypeEx, you can directly use the fonts located in the C:\Windows\Fonts directory without the need for downloading them separately.

Value	Description
1	Foreign language fonts1
2	Foreign language fonts2
3	Foreign language fonts3
4	Foreign language fonts4
5	Foreign language fonts5
6	24*24 dot matrix Chinese
'A'~'Z'	Truetype fonts

pHorizontal: When pFont is set to an internal font (1 to 6), pHorizontal represents the point enlargement factor. You can choose from 1 to 24. When pFont selects a TrueType font (A to Z), pHorizontal represents the font's width, measured in pixels (no size limit).

pVertical: When pFont is set to an internal font (1 to 6), pVertical is used to set the dot matrix vertical magnification factor. You can choose from 1 to 24. When pFont is set to a TrueType font (A to Z), pVertical is used to set the font height, measured in pixel(no size limit).

pColor: Select 'N' corresponding to the ASCII value 78 to print regular text (e.g., black text on a white background). Select 'R,' corresponding to the ASCII value 82, to print inverted text (e.g., white text on a black background).

pstr: A string with a length of 1 to 100 characters. Users can freely combine it into a composite string using "DATA", Cn, Vn.
 "DATA": A constant string that must be enclosed in double quotation marks, such as "Postek Printer."
 Cn: Numerical value of a serial number that must have been previously defined.
 Vn: Variable string that must have been previously defined.
 For example: "text1" Cn "text2" Vn.
 R: Indicating that the printed content is the current UHF RFID label TID (64-bit) data. Please note that this function is only supported RFID printers.

Variable: TRUE indicates that the current string contains variable operations and that constant strings should be enclosed in double quotation marks (" ").
 For example:
 PTK_DrawTextEx (50,30,0,2,1,1,'N',"123456789\C0",TRUE);
 FALSE indicates that the current string does not contain variable operations, and there's no need to enclose constant strings in double quotation marks.
 For example: the following two command have the same effect;
 PTK_DrawTextEx (50,30,0,2,1,1,'N',"123456789",FALSE);
 PTK_DrawText(50,30,0,2,1,1 ,'N'," 123456789") .

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetLabelHeight(500, 0, 0, FALSE); // Set labels to continuous label with a height of 500dots
PTK_DefineCounter(0, 6, 'N', _T("+1"), _T("\nEnter\Code:")); // Define serial number C0
PTK_DefineCounter(1, 6, 'N', _T("+3"), _T("\nEnter\Code:")); // Define serial number C1
PTK_DrawTextEx(20, 20, 0, 6, 2, 2, 'N', _T("\Test 0 serial number \"C0\"), TRUE); // Print C0
PTK_DrawTextEx(20, 120, 0, 6, 2, 2, 'N', _T("\Test 1 serial number \"C1\"), TRUE); // Print C1
PTK_DrawTextEx(20, 220, 0, 6, 2, 2, 'N', _T("Test 2 plain text"), FALSE); // Print Plain
TextPTK_Download();
PTK_DownloadInitVar(_T("111")); // Initialize C0
PTK_DownloadInitVar(_T("222")); // Initialize C1
PTK_PrintLabel(3, 1); // Start printing labels
PTK_CloseUSBPort();
```

PTK_DrawText_TrueType

Description

Call windows font library to print a line of TrueType font.

Note: At the end, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawText_TrueType(unsigned int x, unsigned int y, unsigned int FHeight,
unsigned int FWidth,LPCTSTR FType, unsigned int Fspin, unsigned int FWeight, BOOL
FItalic,BOOL FUnline, BOOL FStrikeOut, LPCTSTR data);
```

Parameters

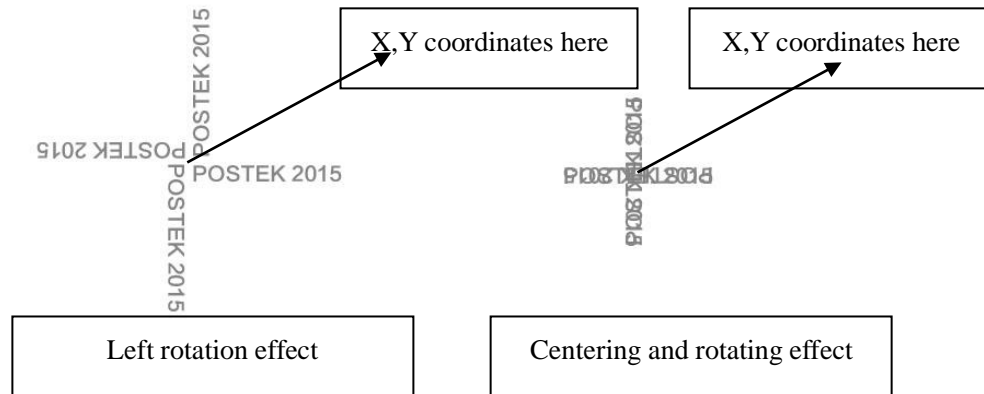
x: Set the X coordinate, measured in dots.
y: Set the Y coordinate, measured in dots.
FHeight: Height of the font in dots;
FWidth: Width of the font in dots;

Note: If you want to print normal scale font, you need to set FWidth to 0.

FType: Font name;

**Note: TrueType Font is based on the Windows operating system, can be loaded or unloaded.
All installed TrueType Font can be used by this function.**

Fspin: The angle of rotation of the font:
1->0 degrees to the left, 2->90 degrees to the left,
3->180 degrees to the left, 4->270 degrees to the left
5->0 degrees center, 6->90 degrees center,
7->180 degrees center, 8->270 degrees center



Fweight: Font thickness:
 0 and 400 -> 400 standard,
 100 -> very fine, 200 -> extremely fine,
 300 -> Fine, 500 -> Medium,
 600 -> semi-bold, 700 -> bold,
 800 -> extra bold, 900 -> bold.

Fitalic: Italic, 0 -> FALSE, 1 -> TRUE.

Funline: Text underlined, 0 -> FALSE, 1 -> TRUE.

FstrikeOut: Text with strikethrough, 0 -> FALSE, 1 -> TRUE.

data: Content of the string.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_DrawText_TrueType(50, 50, 48, 0, _T("Arial"), 1, 400, FALSE, FALSE, FALSE, _T("POSTEK
Arial"));
PTK_DrawText_TrueType(50, 150, 48, 0, _T("Calibri"), 1, 400, FALSE, FALSE, FALSE,
_T("POSTEK Calibri "));
PTK_PrintLabel(1, 1); // Start printing labels
PTK_CloseUSBPort();
```

[PTK_RenameDownloadFont](#)

Description

Match the font downloaded to the printer with the font ID A~Z used by PTK_DrawText.

Note: The name of the font downloaded from the printer can be obtained by calling PTK_GetStorageList.

Syntax

```
int _stdcall PTK_RenameDownloadFont(unsigned int StoreType, TCHAR Fontname, LPTSTR DownloadFontName);
```

Parameters

StoreType: Where the downloaded fonts are stored in the printer, 0: SDRAM, 1: FLASH.

Note: The fonts downloaded to the printer's SDRAM are erased after the printer is powered off, while the fonts downloaded to FLASH are still saved after the printer is powered off.

Fontname: Rename the ID of the downloaded font, value range: A-Z.

DownloadFontName: The name of the downloaded font in the printer.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
```

```
PTK_ClearBuffer();
```

```
PTK_DrawText(20, 20, 0, 6, 2, 2, 'N', _T("Print test text 6,ABC123"));
```

```
PTK_RenameDownloadFont(1, 'A', _T("arial")); // You need to download the arial font through  
                                              utility before you can use it, and name arial as  
                                              the printer's A font
```

```
PTK_DrawText(20, 120, 0, 'A', 48, 48, 'N', _T("TrueType"));
```

```
PTK_PrintLabel(1, 1); // Start printing labels
```

```
PTK_CloseUSBPort();
```

Print Graphic

[*PTK_ChangeIMGtoPCX*](#)

Description

Convert the image to PCX format and generate a PCX image with the same name in the same path.

Note: This API call does not require opening a port.

Syntax

```
int _stdcall PTK_ChangeIMGtoPCX(LPTSTR filePath);
```

Parameters

filePath: The path of the image to be converted.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_ChangeIMGtoPCX(_T("Koala.jpg"));
```

PTK_PcxGraphicsList

Description

To print the list of graphic names stored in the printer's RAM or FLASH storage.

Note: This function does not require calling PTK_PrintLabel to print.

Syntax

```
int _stdcall PTK_PcxGraphicsList(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_PcxGraphicsList();  
PTK_CloseUSBPort();
```

PTK_PcxGraphicsDel

Description

Delete graphics stored in the printer.

Note: This API call is invalid if the image is not present within the printer.

Syntax

```
int _stdcall PTK_PcxGraphicsDel(LPTSTR pcxname);
```

Parameters

pcxname:	The internal storage graphic name within the printer, with a maximum length of 16 characters.
----------	---

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_PcxGraphicsDel(_T("P1"));  
PTK_CloseUSBPort();
```

PTK_AnyGraphicsDownload

Description

Please provide the graphic path for downloading to the printer.

Syntax

```
int _stdcall PTK_AnyGraphicsDownload(LPTSTR pcxname, LPTSTR filePath, float ratio,  
    unsigned int width, unsigned int height, unsigned int iDire);
```

Note: If you do not enable flash storage, the downloaded images will not be saved after power down.

Parameters

pcxname:	This function allows you to set the name for a graphic stored in the internal memory of the printer. The graphic name can have a maximum length of 16 characters. This name is used as an identifier to access and print the stored graphic using the PTK_DrawPcxGraphics function.
filePath:	This parameter represents the file path of the graphic you want to download to the printer. The file path can be a local file path or an HTTP URL. Supported graphic formats include bmp, jpg, png, tif, ico, and pcx. The function will download and store this graphic in the printer's internal memory for later use.

Note: When specifying a URL path for remote file download, you should include either "http://" or "https://" at the beginning of the URL to indicate the protocol used for downloading the graphic from a remote server.

ratio:	The scaling factor. It only takes effect when this parameter is set to 0, in which case the width and height parameters will be used.
width:	The "width" parameter specifies the width of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original width.
height:	The "height" parameter specifies the height of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original height.
iDire:	Rotation angle 0 - 0° 1 - 90° 2 - 180° 3 - 270° 4 - Vertical mirror flip 5 - Horizontal mirror flip

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
// PTK_EnableFLASH(); //Enable FLASH storage
PTK_PcxGraphicsDel(_T("P1"));
PTK_AnyGraphicsDownload(_T("P1"), _T("koala.jpg"), 1, 0, 0, 0);
PTK_AnyGraphicsDownload(_T("P2"), _T("http://www.postek.com.cn/cn/skins/images/logo.jpg"),
1, 0, 0, 0);
//PTK_DisableFLASH(); //Disable FLASH storage
PTK_CloseUSBPort();
```

PTK_DrawPcxGraphics

Description

Print the graphic stored in the printer.

Note: You can use PTK_GetStorageList or PTK_PcxGraphicsList to retrieve the names of graphics stored in the printer. Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawPcxGraphics(unsigned int px, unsigned int py, LPTSTR pcxname);
```

Parameters

px:	Set the X coordinate, measured in dots.
py:	Set the Y coordinate, measured in dots.
pcxname:	The name of the graphic stored within the printer; maximum length is 16 characters.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
// PTK_EnableFLASH(); // Enable FLASH storage
PTK_PcxGraphicsDel(_T("P1"));
PTK_AnyGraphicsDownload(_T("P1"), _T("koala.jpg"), 0.5, 0, 0, 0);
// PTK_DisableFLASH(); //Disable FLASH storage
PTK_DrawPcxGraphics(0, 0, _T("P1"));
PTK_DrawPcxGraphics(0, 500, _T("P1"));
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_AnyGraphicsPrint

Description

Print a graphic directly from the file path.

Note: Finally, you need to call PTK_PrintLabel to start the printing process.

Syntax

```
int _stdcall PTK_AnyGraphicsPrint(unsigned int px, unsigned int py, LPTSTR pcxname, LPTSTR  
filePath, float ratio, unsigned int width, unsigned int height, unsigned int iDire);
```

Parameters

px:	Set the X coordinate, measured in dots.
py:	Set the Y coordinate, measured in dots.
pcxname:	The name of the graphic stored within the printer; maximum length is 16 characters.
filePath:	This parameter represents the file path of the graphic you want to download to the printer. The file path can be a local file path or an HTTP URL. Supported graphic formats include bmp, jpg, png, tif, ico, and pcx. The function will download and store this graphic in the printer's internal memory for later use.

Note: When specifying a URL path for remote file download, you should include either "http://" or "https://" at the beginning of the url to indicate the protocol used for downloading the graphic from a remote server.

ratio:	The scaling factor. It only takes effect when this parameter is set to 0, in which case the width and height parameters will be used.
width:	The "width" parameter specifies the width of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original width.
height:	The "height" parameter specifies the height of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original height.
iDire:	Rotation Angle 0 - 0 degree 1 - 90 degrees 2 - 180 degrees 3 - 270 degrees 4 - Vertical mirror flip 5 - Horizontal mirror flip

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
```

```
PTK_SetDirection('B');
// PTK_EnableFLASH(); //Enable FLASH storage
PTK_AnyGraphicsPrint(0, 0, _T("P1"), _T("Koala.jpg"),0.5, 0, 0, 0); // Integrated functions for
deleting, downloading, and printing in three steps.
```

```
PTK_AnyGraphicsPrint(0, 400, _T("P2"),
_T("http://www.postek.com.cn/cn/skins/images/logo.jpg"),1, 0, 0, 0);
```

```
// PTK_DisableFLASH(); // Disable FLASH storage
PTK_DrawPcxGraphics(0, 500, _T("P1")); // Print graphic P1
PTK_DrawPcxGraphics(600, 0, _T("P2")); // Print graphic P2
PTK_PrintLabel(1, 1); //Two P1s and two P2s printed.
PTK_CloseUSBPort();
```

PTK_AnyGraphicsDownloadFromMemory

Description

To store a graphic on a printer using graphic data.

Note: The function requires accurate graphic type, size, and graphic data input; otherwise, the storage will fail.

Syntax

```
int _stdcall PTK_AnyGraphicsDownloadFromMemory(LPTSTR pcxname, unsigned int imageType,
unsigned int imageSize,float ratio, unsigned int width, unsigned int height, unsigned int iDire,
unsigned char* imageBuffer);
```

Parameters

pcxname:	The name of the graphic stored within the printer; maximum length is 16 characters.
imageType:	The supported graphic formats currently include: 1 – bmp; 3 – jpg; 4 – png; 5 – ico; 6 – tif; 8 – pcx.
imageSize:	The size of the graphic file in bytes.
ratio:	The scaling factor. It only takes effect when this parameter is set to 0, in which case the width and height parameters will be used.
width:	The "width" parameter specifies the width of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original.
width height:	The "height" parameter specifies the height of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original height.
iDire:	Rotation Angle 0 - 0 degree 1 - 90 degrees 2 - 180 degrees 3 - 270 degrees 4 - Vertical mirror flip 5 - Horizontal Mirror Flip

imageBuffer: The graphic data to be printed.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
unsigned char imagebuff[1024 * 1024] = { 0 }; // Define it globally
int imagesize = 0;
FILE * fp = NULL;
char file_name[256] = "Koala.jpg";

PTK_OpenUSBPort(255);
/* Read image data into memory */
fopen_s(&fp, file_name, "rb");
imagesize = fread(imagebuff, sizeof(unsigned char), sizeof(imagebuff) - 1, fp);
fclose(fp);
/*Store the graphic into the printer*/
PTK_PcxGraphicsDel(_T("P1"));
PTK_AnyGraphicsDownloadFromMemory(_T("P1"), 3, imagesize, 1, 0, 0, 0, imagebuff);
PTK_DrawPcxGraphics(0, 0, _T("P1"));
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_AnyGraphicsPrintFromMemory

Description

Print the graphic using the graphic data.

Note: The function requires accurate graphic type, size, and graphic data input; otherwise, the storage will fail. Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_AnyGraphicsPrintFromMemory(int px, int py, LPTSTR pcxname, unsigned int
imageType, unsigned int imageSize, float ratio, unsigned int width, unsigned int height, unsigned int
iDire, unsigned char* imageBuffer);
```

Parameters

px:	Set the X coordinate, measured in dots.
py:	Set the Y coordinate, measured in dots.
pcxname:	The name of the graphic stored within the printer; maximum length is 16 characters.
imageType:	The supported graphic formats currently include: 1 – bmp; 3 – jpg; 4 – png; 5 – ico; 6 – tif; 8 – pcx.
imageSize:	The size of the graphic file in bytes.
ratio:	The scaling factor. It only takes effect when this parameter is set to 0, in

	which case the width and height parameters will be used.
width:	The "width" parameter specifies the width of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original.
height:	The "height" parameter specifies the height of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original height.
iDire:	Rotation Angle: 0 - 0 degree 1 - 90 degrees 2 - 180 degrees 3 - 270 degrees 4 - Vertical mirror flip 5 - Horizontal Mirror Flip
imageBuffer:	The graphic data to be printed.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```

unsigned char imagebuff[1024 * 1024] = { 0 }; // Define it globally
int imagesize = 0;
FILE * fp = NULL;
char file_name[256] = "Koala.jpg";

PTK_OpenUSBPort(255);
/* Read image data into memory */
fopen_s(&fp, file_name, "rb");
imagesize = fread(imagebuff, sizeof(unsigned char), sizeof(imagebuff) - 1, fp);
fclose(fp);
/* Print graphic */
PTK_AnyGraphicsPrintFromMemory(0, 0, _T("P1"), 3, imagesize, 1, 0, 0, 0, imagebuff);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();

```

[*PTK_AnyGraphicsPrint_Base64*](#)

Description

Print graphic via Base64 graphic data.

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

int _stdcall PTK_AnyGraphicsPrintFromMemory(int px, int py, unsigned int imageType, float ratio,

unsigned int width, unsigned int height, unsigned int iDire, LPCTSTR imageBuffer);

Parameters

px:	Set the X coordinate, measured in dots.
py:	Set the Y coordinate, measured in dots.
imageType:	The supported graphic formats currently include: 1 – bmp; 3 – jpg; 4 – png; 5 – ico; 6 – tif; 8 – pcx.
ratio:	The scaling factor. It only takes effect when this parameter is set to 0, in which case the width and height parameters will be used.
width:	The "width" parameter specifies the width of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original
height:	The "height" parameter specifies the height of the graphic after scaling, measured in dots. If you set it to 0, the graphic will retain its original height.
iDire:	Rotation Angle 0- 0 degree 1- 90 degrees 2- 180 degrees 3- 270 degrees 4- Vertical mirror flip 5- Horizontal Mirror Flip
imageBuffer:	The base64 graphic data to be printed.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_AnyGraphicsPrint_Base64(10,10,3,1,0,0,0,image_base64);  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

Print bitmap

PTK_BinGraphicsList

Description

Print a list of graphic names stored in the printer's RAM or FLASH memory

Note: Functions the same as PTK_PcxGraphicsList

Syntax

```
int _stdcall PTK_BinGraphicsList(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_BinGraphicsList();  
PTK_CloseUSBPort();
```

PTK_BinGraphicsDel

Description

Delete bin graphics stored in the printer.

Syntax

```
int _stdcall PTK_BinGraphicsDel(LPTSTR binname);
```

Parameters

binname: The name of the graphic stored internally in the printer, maximum length 16 characters.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_BinGraphicsDel(_T("bo64X64"));  
PTK_PrintLabel(1, 1);
```

PTK_BinGraphicsDownload

Description

Store a binary format graphic in the printer.

Syntax

```
int _stdcall PTK_BinGraphicsDownload(LPTSTR binname, unsigned int pbyte, unsigned int pH,  
unsigned char * Gdata);
```

Parameters

binname: The name of the graphic stored internally in the printer, the maximum length of 16

characters.

pbyte: Number of bytes in one row of data. If the number of bits in one row is not a multiple of 8, round up to the nearest byte.

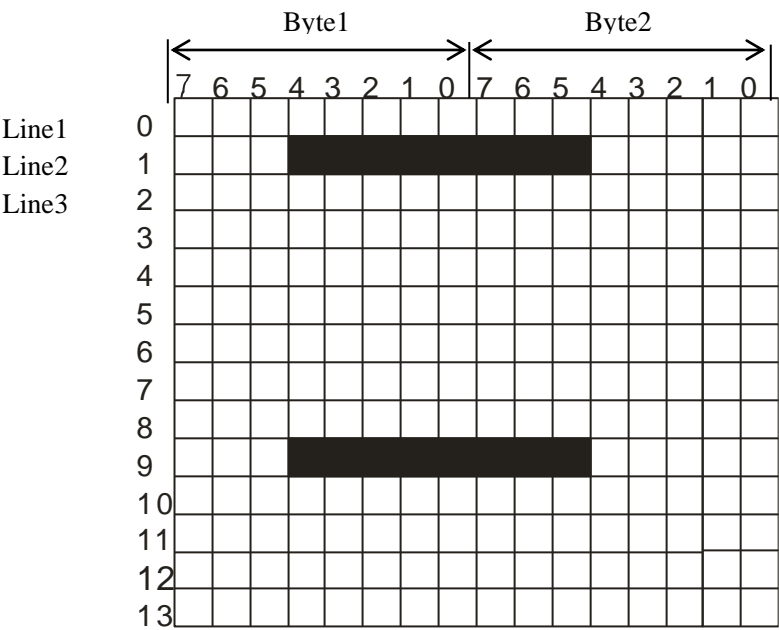
pH: Graphic height, measured in dots.

Gdata: Binary graphic data where each bit's value of 1 indicates printing content, and 0 indicates no printing

Note: The dot matrix data format is: row by row, with the high bit first, in negative logic.

Example:

The data transmission sequence is as follows: Line1's Byte1(0x00), Line1's Byte2(0x00), Line2's Byte1(0x1f), Line2's Byte2(0xe0), Line3's Byte1(0x00), Line3's Byte2(0x00), and so on.
The dashed portions represent non-graphical areas, and their corresponding bit values are 0.



Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
unsigned char font_bo[] =
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x20, 0x00, 0x00, 0x06, 0x04, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00,
    0x07, 0x87, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x07, 0xC3, 0xC0, 0x00,
    0x00, 0x1F, 0x00, 0x00, 0x07, 0x81, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00,
    0x07, 0x80, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x78, 0x00,
    0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00,
```

0x07, 0x80, 0x31, 0x80, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x03, 0xC0,
0x00, 0x1C, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00, 0x1C, 0x07, 0xFF,
0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x07, 0x80, 0x00, 0x00, 0x00, 0x1C, 0x00, 0xC0, 0x07, 0x80, 0x18, 0x00,
0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x1C, 0x30, 0xFF,
0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x1C, 0x78, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x1F, 0xFF, 0xFC, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x0F, 0xFF, 0xFE, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1C, 0x00, 0xFF,
0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x38, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x81, 0xA0, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x06, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0xC0, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x00, 0x01, 0xE0, 0xC0, 0x00, 0x1C, 0x00, 0x00, 0x00, 0x01, 0xE1, 0xE0,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x01, 0xE3, 0xF0, 0x00, 0x1C, 0x7F, 0xFF,
0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x1C, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC,
0x00, 0x1C, 0x00, 0x80, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x60,
0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x38, 0x00, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x1C, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x0F,
0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x0F, 0x80, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x07, 0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x07,
0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x03, 0x80, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x03, 0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x01,
0x00, 0x01, 0xC0, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, 0xFF, 0xC0, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x3F, 0xC0, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x00, 0x1F, 0xC0, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x07, 0x80, 0x00,
0x00, 0x20, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

};/* The Chinese character "博" in a 64x64 dot matrix is as follows*/

```
PTK_OpenUSBPort(255);  
//PTK_EnableFLASH(); //Enable FLASH storage  
PTK_BinGraphicsDel(_T("bo64X64"));  
PTK_BinGraphicsDownload(_T("bo64X64"), 8, 64, font_bo);  
//PTK_DisableFLASH(); //Disable FLASH storage
```

PTK_CloseUSBPort();

PTK_RecallBinGraphics

Description

Print the bin graphics stored in the printer.

Note: The name of the graphics in the printer can be obtained by calling PTK_GetStorageList or PTK_BinGraphicsList. Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_RecallBinGraphics(unsigned int px, unsigned int py, LPTSTR binname);
```

Parameters

px: Set the X coordinate, measured in dots.
py: Set the Y coordinate, measured in dots.
binname: The printer's internal storage of the name of the graphics, the maximum length of 16 characters.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
unsigned char font_bo[] =  
{  
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
    0x00, 0x20, 0x00, 0x00, 0x06, 0x04, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00,  
    0x07, 0x87, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x07, 0xC3, 0xC0, 0x00,  
    0x00, 0x1F, 0x00, 0x00, 0x07, 0x81, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00,  
    0x07, 0x80, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x78, 0x00,  
    0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00,  
    0x07, 0x80, 0x31, 0x80, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x03, 0xC0,  
    0x00, 0x1C, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00, 0x1C, 0x07, 0xFF,  
    0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00,  
    0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00,  
    0x07, 0x80, 0x00, 0x00, 0x00, 0x1C, 0x00, 0xC0, 0x07, 0x80, 0x18, 0x00,  
    0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x1C, 0x30, 0xFF,  
    0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x1C, 0x78, 0xE0, 0x07, 0x80, 0x3C, 0x00,  
    0x1F, 0xFF, 0xFC, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x0F, 0xFF, 0xFE, 0xE0,  
    0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,  
    0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,  
    0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFC, 0x00,
```

```

0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1C, 0x00, 0xFF,
0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x38, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x81, 0xA0, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x06, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0xC0, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x00, 0x01, 0xE0, 0xC0, 0x00, 0x1C, 0x00, 0x00, 0x00, 0x01, 0xE1, 0xE0,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x01, 0xE3, 0xF0, 0x00, 0x1C, 0x7F, 0xFF,
0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x1C, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC,
0x00, 0x1C, 0x00, 0x80, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x60,
0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x38, 0x00, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x1C, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x0F,
0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x0F, 0x80, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x07, 0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x07,
0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x03, 0x80, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x03, 0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x01,
0x00, 0x01, 0xC0, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, 0xFF, 0xC0, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x3F, 0xC0, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x00, 0x1F, 0xC0, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x07, 0x80, 0x00,
0x00, 0x20, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};/*The Chinese character "博" in a 64x64 dot matrix is as follows */

```

```

PTK_OpenUSBPort(255);
PTK_BinGraphicsDel(_T("bo64X64"));
PTK_BinGraphicsDownload(_T("bo64X64"), 8, 64, font_bo);
PTK_RecallBinGraphics(50, 50, _T("bo64X64"));
PTK_RecallBinGraphics(50, 150, _T("bo64X64"));
PTK_RecallBinGraphics(150, 50, _T("bo64X64"));
PTK_RecallBinGraphics(150, 150, _T("bo64X64"));
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();

```

PTK_DrawBinGraphics

Description

Directly define and print a bin graphic.

Note: Finally, you need to call **PTK_PrintLabel** to start the printing process.

Syntax

```
int _stdcall PTK_DrawBinGraphics(unsigned int px, unsigned int py, unsigned int pbyte, unsigned int pH, unsigned char* Gdata);
```

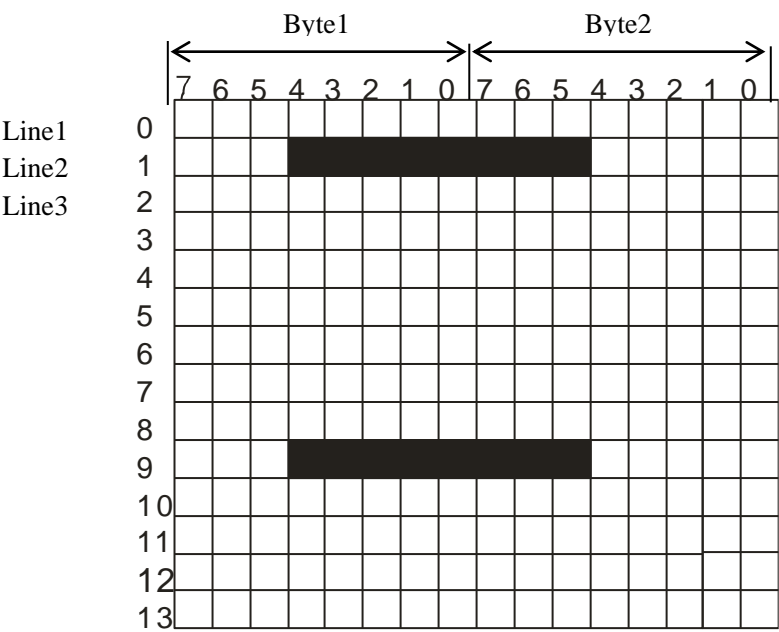
Parameters

- px: Set the X coordinate, measured in dots.
- py: Set the Y coordinate, measured in dots.
- pbyte: The number of bytes in one line of data. If the number of bits in one line is not a multiple of 8, round up to the nearest whole byte.
- pH: Graphic height, measured in dots.
- Gdata: Binary graphic data where a bit with a value of 1 indicates printing, and 0 indicates no printing.

Note: The dot matrix value rule is: line by line, high bit first, inverse code.

Example:

The data transmission sequence is as follows: Line1's Byte1(0x00), Line1's Byte2(0x00), Line2's Byte1(0x1f), Line2's Byte2(0xe0), Line3's Byte1(0x00), Line3's Byte2(0x00), and so on. The dashed portions represent non-graphical areas, and their corresponding bit values are 0.



Returns

0 -> OK
For other return values, please call PTK_GetErrorInfo to parse.

Example

```
unsigned char font_bo[] =  
{  
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x20, 0x00, 0x00, 0x06, 0x04, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00,
0x07, 0x87, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x07, 0xC3, 0xC0, 0x00,
0x00, 0x1F, 0x00, 0x00, 0x07, 0x81, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x07, 0x80, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x78, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x38, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x07, 0x80, 0x31, 0x80, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x03, 0xC0,
0x00, 0x1C, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0, 0x00, 0x1C, 0x07, 0xFF,
0xFF, 0xFF, 0xFF, 0xF0, 0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x07, 0x80, 0x00, 0x00, 0x00, 0x1C, 0x00, 0xC0, 0x07, 0x80, 0x18, 0x00,
0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x1C, 0x30, 0xFF,
0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x1C, 0x78, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x1F, 0xFF, 0xFC, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x0F, 0xFF, 0xFE, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFC, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x80, 0x3C, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xFF, 0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1C, 0x00, 0xFF,
0xFF, 0xFF, 0xFC, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x3C, 0x00,
0x00, 0x1C, 0x00, 0xE0, 0x07, 0x80, 0x38, 0x00, 0x00, 0x1C, 0x00, 0xE0,
0x07, 0x81, 0xA0, 0x00, 0x00, 0x1C, 0x00, 0xE0, 0x06, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0xC0, 0x00, 0x01, 0xF0, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x00, 0x01, 0xE0, 0xC0, 0x00, 0x1C, 0x00, 0x00, 0x00, 0x01, 0xE1, 0xE0,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x01, 0xE3, 0xF0, 0x00, 0x1C, 0x7F, 0xFF,
0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x1C, 0x3F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC,
0x00, 0x1C, 0x00, 0x80, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x60,
0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x38, 0x00, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x1C, 0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x0F,
0x00, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x0F, 0x80, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x07, 0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x07,
0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x03, 0x80, 0x01, 0xE0, 0x00,
0x00, 0x1C, 0x00, 0x03, 0x80, 0x01, 0xE0, 0x00, 0x00, 0x1C, 0x00, 0x01,
0x00, 0x01, 0xC0, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00, 0xFF, 0xC0, 0x00,
0x00, 0x1C, 0x00, 0x00, 0x00, 0x3F, 0xC0, 0x00, 0x00, 0x1C, 0x00, 0x00,
0x00, 0x1F, 0xC0, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x07, 0x80, 0x00,
0x00, 0x20, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

};/* The Chinese character "博" in a 64x64 dot matrix is as follows */

```
PTK_OpenUSBPort(255);
PTK_DrawBinGraphics(50, 50, 8, 64, font_bo);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

Printing Lines

PTK_DrawRectangle

Description

Draw an empty rectangle on a label

Note: At the end, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawRectangle(unsigned int px, unsigned int py, unsigned int thickness, unsigned
int pEx,unsigned int pEy);
```

Parameters

px:	Starting point's X coordinate, measured in dots.
py:	Starting point's Y coordinate, measured in dots.
thickness:	Border thickness, measured in dots.
pEx:	Ending point's X coordinate, measured in dots.
pEy:	Ending point's Y coordinate, measured in dots.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DrawRectangle(50, 50, 24, 200, 200);
PTK_PrintLabel(1, 1); // Print a label
PTK_CloseUSBPort();
```

PTK_DrawLineXor

Description

Draw a straight line, and perform XOR processing if they intersect.

Note: It can also be used to draw a solid rectangle. Finally, you need to call PTK_PrintLabel to begin printing.

Syntax

```
int _stdcall PTK_DrawLineXor(unsigned int px, unsigned int py,unsigned int pL, unsigned int pH);
```

Parameters

px: Starting point's X coordinate, measured in dots.
py: Starting point's Y coordinate, measured in dots.
pL: Set the horizontal length of the line in dots.
pH: Set the vertical height of the line in dots.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_DrawLineXor(300, 300, 500, 12);  
PTK_DrawLineXor(400, 200, 12, 500);  
PTK_PrintLabel(1, 1); // Print a label  
PTK_CloseUSBPort();
```

PTK_DrawLineOr

Description

Draw a line, perform a logical OR operation if they intersect.

Note: Can also be used to draw a solid rectangle. Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawLineOr(unsigned int px, unsigned int py, unsigned int pL, unsigned int pH);
```

Parameters

px: Starting point's X coordinate, measured in dots.
py: Starting point's Y coordinate, measured in dots.
pL: Set the horizontal length of the line in dots.
pH: Set the vertical height of the line in dots.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_DrawLineOr (300, 300, 500, 12);  
PTK_DrawLineOr(400, 200, 12, 500);  
PTK_PrintLabel(1, 1); // Print a label  
PTK_CloseUSBPort();
```

PTK_DrawDiagonal

Description

Draw a diagonal line with XOR processing where it intersects.

Note: Finally, you need to call PTK_PrintLabel to begin printing.

Syntax

```
int _stdcall PTK_DrawDiagonal(unsigned int px, unsigned int py, unsigned int thickness, unsigned int pEx, unsigned int pEy);
```

Parameters

px:	Starting point's X coordinate, measured in dots.
py:	Starting point's Y coordinate, measured in dots.
thickness:	Border thickness, measured in dots.
pEx:	Ending point's X coordinate, measured in dots.
pEy:	Ending point's Y coordinate, measured in dots.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_DrawDiagonal(50, 50, 12, 300, 300);  
PTK_DrawDiagonal(300, 50, 12, 50, 300);  
PTK_PrintLabel(1, 1); // Print a label  
PTK_CloseUSBPort();
```

PTK_DrawWhiteLine

Description

Draw a white straight line.

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawWhiteLine(unsigned int px, unsigned int py, unsigned int pL, unsigned int pH);
```

Parameters

px:	Starting point's X coordinate, measured in dots.
py:	Starting point's Y coordinate, measured in dots.
pL:	Set the horizontal length of the line in dots.

pH: Set the vertical height of the line in dots.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DrawLineOr(50, 50, 500, 500);
PTK_DrawWhiteLine(100, 100, 300, 12);
PTK_DrawWhiteLine(100, 200, 300, 12);
PTK_DrawWhiteLine(100, 300, 300, 12);
PTK_PrintLabel(1, 1); // Print a label
PTK_CloseUSBPort();
```

Print 1D Barcode

PTK_DrawBarcode

Description

Print multiple types of 1D barcodes with constant strings as content.

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawBarcode(unsigned int px, unsigned int py, unsigned int pdirec, LPTSTR
pCode, unsigned int NarrowWidth, unsigned int pHorizontal, unsigned int pVertical, TCHAR ptext,
LPTSTR pstr);
```

Parameters

px: Set the X coordinate, measured in dots.

py: Set the Y coordinate, measured in dots.

pdirec: Select the barcode printing orientation:
0 - no rotation; 1 - rotate 90°; 2 - rotate 180°; 3 - rotate 270°.

pCode: Select the barcode type to print. (Different barcode types have character limits or restrictions on the number of characters, please refer to the specific standards.)

pCode	Barcode Type
0	Code 128 UCC (shipping container code)
1	Code 128 AUTO
1A	Code 128 subset A
1B	Code 128 subset B
1C	Code 128 subset C

1E	UCC/EAN
2	Interleaved 2 of 5
2C	Interleaved 2 of 5 with check sum digit
2D	Interleaved 2 of 5 with human readable check digit
2G	German Postcode
2M	Matrix 2 of 5
2U	UPC Interleaved 2 of 5
3	Code 3 of 9
3C	Code 3 of 9 with check sum digit
3E	Extended Code 3 of 9
3F	Extended Code 3 of 9 with check sum digit
9	Code93
E30	EAN-13
E32	EAN-13 2 digit add-on
E35	EAN-13 5 digit add-on
E80	EAN-8
E82	EAN-8 2 digit add-on
E-85	EAN-8 5 digit add-on
K	Codabar
P	Postnet
UA0	UPC-A
UA2	UPC-A 2 digit add-on
UA5	UPC-A 5 digit add-on
UE0	UPC-E
UE2	UPC-E 2 digit add-on
UE5	UPC-E 5 digit add-on

NarrowWidth: Set the width of the narrow element in the barcode, measured in dots.

pHorizontal: Set the width of the wide element in the barcode, measured in dots.

pVertical: Set the barcode height , measured in dots.

ptext: Select 'N', corresponding to ASCII value 78, to not print human-readable text below the barcode.
Select 'B' to print human-readable text below the barcode, left-aligned.
When the barcode type is selected as Code 128 AUTO:
Select 'C' to print human-readable text below the barcode, centered.
Select 'R' to print human-readable text below the barcode, right-aligned.

Note: If you are using a 200 DPI printer, you need to set "tph=2" in the CDFPSK.ini to use the center and right-align functions correctly.

pstr: A string with a length of 1 to 100 characters.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DrawBarcode(100, 50, 0, _T("1"), 3, 0, 100, 'N', _T("POSTEK2020"));
PTK_DrawBarcode(100, 200, 0, _T("1"), 3, 0, 100, 'B', _T("POSTEK2020"));
PTK_DrawBarcode(100, 350, 0, _T("1"), 3, 0, 100, 'C', _T("POSTEK2020")); // Currently, only
Code 128 Auto supports center and right alignment for display
```

```
PTK_DrawBarcode(100, 500, 0, _T("1"), 3, 0, 100, 'R', _T("POSTEK2020"));
PTK_PrintLabel(1, 1); // Print a label
PTK_CloseUSBPort();
```

PTK_DrawBarcodeEx

Description

Print multiple types of 1D barcodes, and the printing content can be constants, serial numbers, or variable strings.

Note: At the end, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawBarcodeEx(unsigned int px, unsigned int py, unsigned int pdirec, LPTSTR
pCode, unsigned int NarrowWidth, unsigned int pHorizontal, unsigned int pVertical, TCHAR ptext,
LPTSTR pstr, BOOL Variable);
```

Parameters

px: Set the X coordinate, measured in dots.

py: Set the Y coordinate, measured in dots.

pdirec: Select the barcode printing orientation:
0 - no rotation; 1 - rotate 90°; 2 - rotate 180°; 3 - rotate 270°.

pCode: Select the barcode type to print. (Different barcode types have character limits or restrictions on the number of characters, please refer to the specific standards.)

pCode	Barcode Type
0	Code 128 UCC (shipping container code)
1	Code 128 AUTO
1A	Code 128 subset A
1B	Code 128 subset B
1C	Code 128 subset C
1E	UCC/EAN
2	Interleaved 2 of 5
2C	Interleaved 2 of 5 with check sum digit
2D	Interleaved 2 of 5 with human readable check digit
2G	German Postcode
2M	Matrix 2 of 5
2U	UPC Interleaved 2 of 5
3	Code 3 of 9
3C	Code 3 of 9 with check sum digit

3E	Extended Code 3 of 9
3F	Extended Code 3 of 9 with check sum digit
9	Code93
E30	EAN-13
E32	EAN-13 2 digit add-on
E35	EAN-13 5 digit add-on
E80	EAN-8
E82	EAN-8 2 digit add-on
E-85	EAN-8 5 digit add-on
K	Codabar
P	Postnet
UA0	UPC-A
UA2	UPC-A 2 digit add-on
UA5	UPC-A 5 digit add-on
UE0	UPC-E
UE2	UPC-E 2 digit add-on
UE5	UPC-E 5 digit add-on

NarrowWidth: Set the width of the narrow element in the barcode, measured in dots.

pHorizontal: Set the width of the wide element in the barcode, measured in dots.

pVertical: Set the barcode height , measured in dots.

ptext: Select 'N', corresponding to ASCII value 78, to not print human-readable text below the barcode.
Select 'B' to print human-readable text below the barcode, left-aligned.
When the barcode type is selected as Code 128 AUTO:
Select 'C' to print human-readable text below the barcode, centered.
Select 'R' to print human-readable text below the barcode, right-aligned.

Note: If you are using a 200 DPI printer, you need to set "tph=2" in the CDFPSK.ini to use the center and right-align functions correctly.

pstr: A string with a length of 1 to 100 characters. You can combine "DATA," Cn, Vn into a composite string.
"DATA": Constant string, which must be enclosed in double quotes, like "POSTEK Printer."
Cn: Serial number value, which must have been previously defined.
Vn: Variable string, which must have been previously defined.
Such as: "text1" Cn "text2" Vn.
R: This indicates that the printed content in this case is the current UHF RFID label's TID (64-bit) data. Please note that this functionality is only supported by RFID printers.

Variable: "True" means that the current string contains variable operations, and you should use “\” to enclose constant strings for printing.
For example:
PTKDrawBarcodeEx (50,30,0,"1A",1,1,10,'N',"123456",true);
False indicates that the current string does not contain variable operations, and there is no need to include “\”.

For example: PTK_DrawBarcode(100, 100, 0, _T("1"), 3, 0, 100, 'N', _T("POSTEK2020")); and PTK_DrawBarcodeEx(100, 100, 0, _T("1"), 3, 0, 100, 'N', _T("POSTEK2020"), FALSE); produce the same result.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DefineCounter(0, 6, 'N', _T("+1"), _T("\nEnter\Code:")); //To define a serial number C0
PTK_DrawBarcodeEx(100, 100, 0, _T("1"), 3, 0, 100, 'C', _T("\Test Serial:\C0"), TRUE);
PTK_DrawBarcodeEx(100, 300, 0, _T("1"), 3, 0, 100, 'C', _T("Test Serial:Text"), FALSE);
PTK_Download();
PTK_DownloadInitVar(_T("2023")); // To initialize a variable or serial number C0
PTK_PrintLabel(3, 1); // Print three labels
PTK_CloseUSBPort();
```

Print 2D Barcode

PTK_DrawBar2D_QR

Description

Print a QR code

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawBar2D_QR(unsigned int x, unsigned int y, unsigned int w, unsigned int v,
unsigned int o, unsigned int r, unsigned int m, unsigned int g, unsigned int s, LPTSTR pstr);
```

Parameters

- | | |
|----|--|
| x: | Starting point's X coordinate, measured in dots. |
| y: | Starting point's Y coordinate, measured in dots. |
| w: | Reserved parameter, please enter 0. |
| v: | The QR code version (Version) corresponds to the QR code graphic size (size), with version numbers ranging from 1 to 40. Version 1 represents a 21x21 matrix, and for each increment in version number, the matrix size increases by 4 modules (Module). Therefore, version 40 represents a 177x177 matrix.
0: Automatic matching (default)
1: 21x21
2: 25x25
...
40: 177x177 |
| o: | Setting the rotation direction, range: 0 to 3. |

	(0 - 0°, 1 - 90°, 2 - 180°, 3 - 270°)
r:	Setting the zoom factor, in dots, with a range of values from 1 to 99. (1 - magnification 1 times, 2 - magnification 2 times, 3 - magnification 3 times ...)
m:	Reserved parameter, please enter 0.
g:	Selecting the QR code error correction level, with a range of values from 0 to 3. 0 corresponds to 'L' level 1 corresponds to 'M' level 2 corresponds to 'Q' level 3 corresponds to 'H' level
s:	Select the QR code mask pattern, with a range of values from 0 to 8. 0 - Mask pattern 000 1 - Mask pattern 001 2 - Mask pattern 010 3 - Mask pattern 011 4 - Mask pattern 100 5 - Mask pattern 101 6 - Mask pattern 110 7 - Mask pattern 111 8 - Automatically select mask pattern
pstr:	Barcode content string.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DrawBar2D_QR(300, 100, 0, 0, 0, 10, 0, 0, 8, _T("Postek Electronics POSTEK2023"));
PTK_PrintLabel(1, 1); // Print a label
PTK_CloseUSBPort();
```

[*PTK_DrawBar2D_QREx*](#)

Description

Printing a QR code in image format and saving it to the printer for printing is supported, even for older firmware versions.

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawBar2D_QREx(unsigned int x, unsigned int y, unsigned int o, unsigned int r,
unsigned int g, unsigned int v, unsigned int s, LPTSTR binname, LPTSTR pstr);
```

Parameters

x:	Starting point's X coordinate, measured in dots.
y:	Starting point's Y coordinate, measured in dots.
o:	Setting the rotation direction, range: 0 to 3. (0 - 0°, 1 - 90°, 2 - 180°, 3 - 270°)
r:	Setting the zoom factor, in dots, with a range of values from 1 to 99. (1 - magnification 1 times, 2 - magnification 2 times, 3 - magnification 3 times ...)
g:	Selecting the QR code error correction level, with a range of values from 0 to 3. 0 corresponds to 'L' level 1 corresponds to 'M' level 2 corresponds to 'Q' level 3 corresponds to 'H' level
v:	The QR code version (Version) corresponds to the QR code graphic size (size), with version numbers ranging from 1 to 40. Version 1 represents a 21x21 matrix, and for each increment in version number, the matrix size increases by 4 modules (Module). Therefore, version 40 represents a 177x177 matrix. 0: Automatic matching (default) 1: 21x21 2: 25x25 ... 40: 177x177
s:	Select the QR code mask pattern, with a range of values from 0 to 8. 0 - Mask pattern 000 1 - Mask pattern 001 2 - Mask pattern 010 3 - Mask pattern 011 4 - Mask pattern 100 5 - Mask pattern 101 6 - Mask pattern 110 7 - Mask pattern 111 8 - Automatically select mask pattern
binname:	The name of the graphic stored in the printer's memory, with a maximum length of 16 characters.

Note: You can use this name to recall the 2D code and print it again using **PTK_RecallBinGraphics**. However, please note that this functionality is not available if the printer does not have FLASH storage enabled. If you have FLASH storage enabled, the graphic will still be available after a power cycle.

pstr: Barcode content string.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
```

```
PTK_DrawBar2D_QREx(100, 100, 0, 10, 0, 0, 8, "QR1", _T("Postek Electronics POSTEK2023"));
PTK_RecallBinGraphics(400, 100, _T("QR1"));
PTK_RecallBinGraphics(100, 400, _T("QR1"));
PTK_RecallBinGraphics(400, 400, _T("QR1"));
PTK_PrintLabel(1, 1); // Print a label
PTK_CloseUSBPort();
```

[*PTK_DrawBar2D_HANXIN*](#)

Description

Print HanXin Code.

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawBar2D_HANXIN(unsigned int x, unsigned int y, unsigned int w, unsigned
int v, unsigned int o, unsigned int r, unsigned int m, unsigned int g, unsigned int s, LPTSTR pstr);
```

Parameters

x:	Starting point's X coordinate, measured in dots.
y:	Starting point's Y coordinate, measured in dots.
w:	Reserved parameter, please enter 0.
v:	Reserved parameter, please enter 0.
o:	Setting the rotation direction, range: 0 to 3. (0 - 0°, 1 - 90°, 2 - 180°, 3 - 270°)
r:	Setting the zoom factor, in dots, with a range of values from 1 to 30. (0 - magnification 1 times, 1 - magnification 2 times, 2 - magnification 3 times ...)
m :	HanXin Code encoding mode selection, with a range of values from 0 to 6. 0 - Numeric Mode 1 - Text Mode 2 - Binary Mode 3 - Common Chinese Characters 1 Area Mode Encoding 4 - Common Chinese Characters 2 Area Mode Encoding 5 - GB 18030 Double-Byte Area Mode 6 - GB 18030 Four-Byte Mode Encodin
g :	Selecting the HanXin Code error correction level, with a range of values from 0 to 3. 0 - Corresponds to 'L1' level 1 - Corresponds to 'L2' level 2 - Corresponds to 'L3' level 3 - Corresponds to 'L4' level
s :	Select the HanXin code mask pattern, with a range of values from 0 to 3. 0 - Mask pattern 00 1 - Mask pattern 01 2 - Mask pattern 02

3 - Mask pattern 03
pstr: Barcode content string.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_DrawBar2D_HANXIN(50, 50, 0, 0, 0, 8, 5, 3, 2, _T("http://www.postekchina.com"));  
PTK_PrintLabel(1, 1); // Print a label  
PTK_CloseUSBPort();
```

[*PTK_DrawBar2D_Pdf417*](#)

Description

Print a PDF417 barcode.

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

int _stdcall PTK_DrawBar2D_Pdf417(unsigned int x, unsigned int y, unsigned int w, unsigned int v, unsigned int s, unsigned int c, unsigned int px, unsigned int py, unsigned int r, unsigned int l, unsigned int t, unsigned int o, LPTSTR pstr);

Parameters

x: Starting point's X coordinate, measured in dots.
y: Starting point's Y coordinate, measured in dots.
w: Reserved parameter, please enter 0.
v: Reserved parameter, please enter 0.
s: Selecting error correction level, with a range of values from 0 to 8.
c: Reserved parameter, please enter 0.
px: Module width, range: 2~9 dots.
py: Module height, range: 4~99 dots.
r: Mmaximum line number, range: 3~90.
l: Maximum column number, range: 1~30.
t: Truncation Flag
0 - Do not truncate, 1 - Truncate.
o: Setting the rotation direction, range: 0 to 3.
(0 - 0°, 1 - 90°, 2 - 180°, 3 - 270°)
pstr: Barcode content string.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example:

```
PTK_OpenUSBPort(255);
PTK_DrawBar2D_Pdf417(50, 50, 0, 0, 0, 0, 3, 7, 10, 2, 0, 0, _T("POSTEKINFO"));
PTK_PrintLabel(1, 1); // Print a label
PTK_CloseUSBPort();
```

PTK_DrawBar2D_Pdf417Ex

Description

Printing a PDF417 barcode as a graphic.

Note: This function is supported by firmware that cannot call PTK_DrawBar2D_Pdf417 to print. Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawBar2D_Pdf417Ex(unsigned int x, unsigned int y, unsigned int w, unsigned int v, unsigned int s, unsigned int c, unsigned int px, unsigned int py, unsigned int r, unsigned int l, unsigned int t, unsigned int o, LPTSTR pstr);
```

Parameters

x:	Starting point's X coordinate, measured in dots.
y:	Starting point's Y coordinate, measured in dots.
w:	Reserved parameter, please enter 0.
v:	Reserved parameter, please enter 0.
s:	Selecting error correction level, with a range of values from 0 to 8.
c:	Reserved parameter, please enter 0.
px:	Module width, range: 2~9 dots.
py:	Module height, range: 4~99 dots.
l:	Maximum column number, range: 1~30.
t:	Truncation Flag. 0 - Do not truncate, 1 - Truncate.
o:	Setting the rotation direction, range: 0 to 3. (0 - 0°, 1 - 90°, 2 - 180°, 3 - 270°)
pstr:	Barcode content string.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DrawBar2D_Pdf417Ex(50, 50, 0, 0, 0, 0, 3, 7, 10, 2, 0, 0, _T("POSTEKINFO"));
PTK_PrintLabel(1, 1); // Print a label
PTK_CloseUSBPort();
```

PTK_DrawBar2D_MaxiCode

Description

Print MaxiCode barcode.

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawBar2D_MaxiCode(unsigned int x, unsigned int y, unsigned int m, unsigned int u, LPTSTR pstr);
```

Parameters

x: Starting point's X coordinate, measured in dots.
y: Starting point's Y coordinate, measured in dots.
m: Mode [2 - 4].
u: If or not in UPS format, value 0 or 1.
pstr: Barcode content string.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_DrawBar2D_MaxiCode(50, 50, 4, 0, _T("1Z000A7&dajc_iaj-3=+~#^$5"));  
PTK_PrintLabel(1, 1);    // Print a label  
PTK_CloseUSBPort();
```

PTK_DrawBar2D_DATAMATRIX

Description

Print a DataMatrix barcode

Note: Finally, you need to call PTK_PrintLabel to start printing.

Syntax

```
int _stdcall PTK_DrawBar2D_DATAMATRIX(unsigned int x, unsigned int y, unsigned int w, unsigned int v, unsigned int o, unsigned int m, LPTSTR pstr);
```

Parameters

x: Starting point's X coordinate, measured in dots.
y: Starting point's Y coordinate, measured in dots.
w: Reserved parameter, please enter 0.
v: Reserved parameter, please enter 0.
o: Setting the rotation direction, range: 0 to 3.

m: (0 - 0°, 1 - 90°, 2 - 180°, 3 - 270°)
Setting the zoom factor, in dots.
pstr: Barcode content string.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example:

```
PTK_OpenUSBPort(255);  
PTK_DrawBar2D_DATAMATRIX(50, 50, 0, 0, 0, 20, _T("POSTEK2023"));  
PTK_PrintLabel(1, 1); // Print a label  
PTK_CloseUSBPort();
```

Print Forms and Related

PTK_GetStorageList

Description

Get the names of forms, fonts, or graphics stored in flash.

Note: It can only retrieve names from FLASH, not from RAM. This function only supports USB reading.

Syntax

```
int _stdcall PTK_GetStorageList(LPTSTR listBuff, DWORD listBuffSize, int TempType);
```

Parameters

listBuff: Storage name buffer.
listBuffSize: Size of the listBuff space.
TempType: The type of name list to be retrieved.
0 - Forms
1 - Fonts
2 - Graphics

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR buff[1024] = { 0 };  
PTK_OpenUSBPort(255);  
PTK_GetStorageList(buff, sizeof(buff), 0);  
MessageBox(buff);
```

```
PTK_CloseUSBPort();
```

PTK_FormList

Description

Print a list of form names stored in the printer's RAM or FLASH memory.

Note: This function does not require calling PTK_PrintLabel to print.

Syntax

```
int _stdcall PTK_FormList(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_FormList();  
PTK_CloseUSBPort();
```

PTK_FormDel

Description

Delete the form stored in the printer.

Syntax

```
int _stdcall PTK_FormDel(LPTSTR pid);
```

Parameters

pid: Form name stored in the printer, not exceeding 16 characters.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_FormDel(_T("F1"));  
PTK_CloseUSBPort();
```

PTK_FormDownload

Description

Notify the printer to start storing a form in the printer.

Note: Please use it in conjunction with PTK_FormEnd.

Syntax

```
int _stdcall PTK_FormDownload(LPTSTR pid);
```

Parameters

pid: Name the form stored in the printer, not exceeding 16 characters.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
    /* Store a form */  
//PTK_EnableFLASH(); //Enable FLASH storage  
PTK_FormDel(_T("F1")); //Delete a form with the same name to avoid naming conflicts  
PTK_FormDownload(_T("F1")); //Command the printer to start storing form content  
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("1234567"), FALSE); // Print a line of text  
PTK_FormEnd(); //Notify the printer that form storage is complete  
//PTK_DisableFLASH(); //Disable FLASH storage  
PTK_CloseUSBPort();
```

PTK_FormEnd

Description

Notify the printer that the form storage process has ended.

Note: Please use it in conjunction with PTK_FormDownload.

Syntax

```
int _stdcall PTK_FormEnd(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
    /* Store a form */
//PTK_EnableFLASH(); //Enable FLASH storage
PTK_FormDel(_T("F1")); //Delete a form with the same name to avoid naming conflicts
PTK_FormDownload(_T("F1")); //Commands the printer to begin storing the contents of the form
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("1234567"), FALSE); //Print a line of text
PTK_FormEnd(); //Notify the printer that form storage is complete
//PTK_DisableFLASH(); //Disable FLASH storage
PTK_CloseUSBPort();
```

PTK_ExecForm

Description

Running a form is equivalent to executing all the API functions in the form's content once.

Notes:

- **The form needs to be stored in the printer, and you can check the form names through PTK_GetStorageList and PTK_FormList.**
- **If the form does not include a call to PTK_PrintLabelAuto for automatic printing, you will need to use PTK_PrintLabel to initiate label printing.**

Syntax

```
int _stdcall PTK_ExecForm(LPTSTR pid);
```

Parameters

pid: The names of forms stored in the printer should not exceed 16 characters.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
    /* Store a form */
//PTK_EnableFLASH(); //Enable FLASH storage
PTK_FormDel(_T("F1")); //Delete a form with the same name to avoid naming conflicts
PTK_FormDownload(_T("F1")); //Commands the printer to begin storing the contents of the form
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("1234567"), FALSE); //Print a line of text
PTK_FormEnd(); //Notify the printer that form storage is complete
//PTK_DisableFLASH(); //Disable FLASH storage
PTK_ExecForm(_T("F1"));
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_DefineVariable

Description

Define a variable in the printer.

Syntax

```
int _stdcall PTK_DefineVariable(unsigned int id, unsigned int maxNum, TCHAR ptext, LPTSTR hintMsg);
```

Parameters

id:	Variable ID number, with a range of 00 to 99.
maxNum:	Maximum character count, with a range of 1 to 99.
ptext:	Alignment options. L (Left align), R (Right align), C (Center align), N (No alignment).
hintMsg:	Prompt content, which will be displayed on the KDU (Keyboard Display Unit) or the printer's LCD

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR v0[16] = _T("1234");
PTK_OpenUSBPort(255);
PTK_DefineVariable(0, 4, 'L', _T(""));
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("\Test variable\\"V0"), TRUE);
PTK_Download();
PTK_DownloadInitVar(v0); //Assigning a value to a printer variable
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_DefineCounter

Description

Defining a serial number in the printer

Syntax

```
int _stdcall PTK_DefineCounter(unsigned int id, unsigned int maxNum, TCHAR ptext, LPTSTR prule, LPTSTR hintMsg);
```

Parameters

id:	Serial number ID number, with a range of 0 to 9.
maxNum:	Maximum number of digits for the serial number, with a range of 1 to 40.
ptext:	Alignment options.

prule:	<p>L (Left align), R (Right align), C (Center align), N (No alignment).</p> <p>Pattern of change for the serial number; composed of a "+" or "-" followed by a number and a change indicator (D - Decimal, B - Binary, O - Octal, H - Hexadecimal):</p> <p>“+1” = Increases by 1 each time, defaulting to decimal calculation, for example: 1234, 1235, 1236, ...</p> <p>“+3D” = Increases by 3 each time, calculated in decimal, as above.</p> <p>“-1B” = Decreases by 1 each time, calculated in binary, for example: 1111, 1110, 1101, ...</p> <p>“-4O” = Decreases by 4 each time, calculated in octal, for example: 1234, 1230, 1224, ...</p> <p>“-6H” = Decreases by 6 each time, calculated in hexadecimal, for example: 1234, 122E, 1228, ...</p>
hintMsg:	<p>Prompt content, which will be displayed on the KDU (Keyboard Display Unit) or the printer's LCD</p>

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DefineCounter(0, 6, 'L', _T("+3"), _T(""));
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("\Test serial number\\"C0"), TRUE);
PTK_Download();
PTK_DownloadInitVar(_T("0001")); //Assigning an initial value to the printer's serial number
PTK_PrintLabel(3, 1);
PTK_CloseUSBPort();
```

PTK_Download

Description

Notify the printer to start assigning initial values to variables or serial numbers.

Syntax

```
int _stdcall PTK_Download(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DefineCounter(0, 6, 'L', _T("+3"), _T(""));
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("\Test serial number \"C0\"), TRUE);
PTK_Download();
PTK_DownloadInitVar(_T("0001")); //Assigning an initial value to the printer's serial number
PTK_PrintLabel(3, 1);
PTK_CloseUSBPort();
```

PTK_DownloadInitVar

Description

Initialize variables or serial numbers in the order they were defined

Syntax

```
int _stdcall PTK_DownloadInitVar(LPTSTR pstr);
```

Parameters

pstr: The string to set as the initial value

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_DefineCounter(0, 6, 'L', _T("+3"), _T(""));
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("\Test serial number \"C0\"), TRUE);
PTK_Download();
PTK_DownloadInitVar(_T("0001")); // Assigning an initial value to the printer's serial number
PTK_PrintLabel(3, 1);
PTK_CloseUSBPort();
```

PTK_PrintLabelAuto

Description

Command the printer to start printing labels, for use within a form, in conjunction with serial numbers or variables.

Note: This function should be used in combination with PTK_Download; the printer will only start printing if serial numbers or variables are defined within the form.

Syntax

```
int _stdcall PTK_PrintLabelAuto(unsigned int number, unsigned int cpnumber);
```

Parameters

number: The number of labels to print, with a range of 1 to 65535.
cpnumber: The number of copies for each label, with a range of 1 to 65535.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
    /* Store a form */
PTK_FormDel(_T("F1"));
PTK_FormDownload(_T("F1"));
PTK_DefineVariable(0, 6, 'N', _T(""));
PTK_DefineCounter(0, 6, 'N', _T("+1"), _T(""));
PTK_DrawTextEx(100, 100, 0, 6, 2, 2, 'N', _T("V0\" Test variable\""), TRUE);
PTK_DrawTextEx(100, 200, 0, 6, 2, 2, 'N', _T("C0\" Test serial number\""), TRUE);
PTK_PrintLabelAuto(3, 1); //Used within a form
PTK_FormEnd();
    /* Run a form */
PTK_ExecForm(_T("F1"));
PTK_Download();
PTK_DownloadInitVar(_T("1234"));
PTK_DownloadInitVar(_T("1234"));
PTK_CloseUSBPort();
```

PTK_FormPrinting

Description

Print the form.

Syntax

```
int _stdcall PTK_FormPrinting(LPTSTR FormName, LPTSTR FormVariable,unsigned int
Quantity,unsigned int Copies);
```

Parameters

FormName: Form name, not exceeding 16 characters.
FormVariable: Form variables, with each variable separated by “\r\n”. If a variable contains “\r\n”, please use the format \\r\\n.
number: The number of labels to print, with a range of 1 to 65535.
cpnumber: The number of copies for each label, with a range of 1 to 65535.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_FormPrinting(_T("Test"), _T("1\r\n2\r\n3\r\n4\r\n5\r\n6\r\n7\r\n"), 1, 1);  
PTK_CloseUSBPort();
```

RFID Label Read/Write Related

PTK_RFIDCalibrate

Description

UHF and HF RFID tag detection calibration.

Syntax

```
int _stdcall PTK_RFIDCalibrate(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_RFIDCalibrate();  
PTK_CloseUSBPort();
```

PTK_RWRFIDLabel

Description

Write RFID tag data

Syntax

```
int _stdcall PTK_RWRFIDLabel(unsigned int nRWMode, unsigned int nWForm, unsigned int  
nStartBlock, unsigned int nWDataNum, unsigned int nWArea, LPTSTR pstr);
```

Note: Finally, you need to call PTK_PrintLabel to start printing.

Parameters

nRWMode: RFID operation mode.
 0 - (Reserved: no function); 1 - Write RFID.

nWForm: RFID write format.
0 - HEX (hexadecimal); 1 – ASCII.
nStartBlock: Write start block.
Note: If writing is performed on the EPC area, unit: words (2 bytes).
nWDataNum: Number of bytes to be written.
nWArea: Write region.

0 - Reserved Area; 1 - EPC; 3 – USER.

pstr: A constant string. (The format is limited by parameter P2).

Note: If writing is done in HEX format, nWDataNum is the number of words in the written data, not the number of words in the string.

Example: The HEX string "313233343536" corresponds to data 0x31, 0x32, 0x33, 0x34, 0x35, 0x36 with a data length of 6 bytes.

The length of the written data must be in units of words (2 bytes), with the effective data length being a multiple of 2 bytes.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example:

```
PTK_OpenUSBPort(255);
```

```
//PTK_RWRFIDLabel(1, 0, 2, 6, 1, "313233343536"); // Write in HEX format
```

```
PTK_RWRFIDLabel(1, 1, 2, 6, 1, "123456"); // Write in ASCII format, same effect as above
```

```
PTK_PrintLabel(1, 1);
```

```
PTK_CloseUSBPort();
```

[PTK_RWRFIDLabelEx](#)

Description

Write RFID tag data (not cleared)

Syntax

```
int _stdcall PTK_RWRFIDLabelEx(unsigned int nRWMode, unsigned int nWForm, unsigned int nStartBlock, unsigned int nWDataNum, unsigned int nWArea, LPTSTR pstr);
```

Note: At the end, you need to call PTK_PrintLabel to start printing.

Parameters

nRWMode: RFID operation mode.
0 - (Reserved: no function); 1 - Write RFID.
nWForm: RFID write format.
0 - HEX (hexadecimal); 1 – ASCII.
nStartBlock: Write start block.

Note: If writing is performed on the EPC area, unit: words (2 bytes).

nWDataNum: Number of bytes to be written.

nWArea: Write region.
0 - Reserved Area; 1 - EPC; 3 – USER.

pstr: A constant string. (The format is limited by parameter P2).

Note: If writing is done in HEX format, nWDataNum is the number of words in the written data, not the number of words in the string.

Example: The HEX string "313233343536" corresponds to data 0x31, 0x32, 0x33, 0x34, 0x35, 0x36 with a data length of 6 bytes.

The length of the written data must be in units of words (2 bytes), with the effective data length being a multiple of 2 bytes.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
//PTK_RWRFIDLabel(1, 0, 2, 6, 1, "313233343536"); // Write in HEX format  
PTK_RWRFIDLabelEx(1, 1, 2, 6, 1, "123456"); // Write in ASCII format, same effect as above  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

PTK_SetRFLabelPWAndLockRFLabel

Description

Set RFID tag password and lock RFID tag.

Syntax

```
int _stdcall PTK_SetRFLabelPWAndLockRFLabel(unsigned int nOperationMode, unsigned int  
OperationnArea, LPTSTR pstr);
```

Parameters

nOperationMode: Operation mode
0 - Unlock; 1 - Lock; 2 - Full unlock;
3 - Full lock; 4 - Password write.

OperationnArea: Operation area
0 - Kill password area; 1 - Access password area;
2 - EPC; 3 - TID; 4 - USER.

pstr: A constant string. (Format restricted to 8 characters in HEX).

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_SetRFLLabelPWAndLockRFLLabel(1,1,"73BE115B");
PTK_CloseUSBPort();
```

PTK_EncodeRFIDPC

Description

Encode the PC value of RFID or the encoding header of the national standard.

Syntax

```
int _stdcall PTK_EncodeRFIDPC(char* PCValue);
```

Parameters

PCValue: The PC value of RFID or the encoding header according to national standards, with data format in hexadecimal.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_RWRFLLabel(1, 1, 2, 6, 1, "123456"); //Write in ASCII format
PTK_EncodeRFIDPC("A040");
PTK_CloseUSBPort();
```

PTK_SetRFID

Description

Set RFID tag printing parameters.

Note:Valid for a single print. If you want to use it, you should call it before each print. It's not saved to FLASH.

Syntax

```
int _stdcall PTK_SetRFID(unsigned int nReservationParameters, unsigned int nReadWriteLocation,
unsigned int ReadWriteArea, unsigned int nMaxErrNum, unsigned int nErrProcessingMethod);
```

Parameters

nReservationParameters:	Reserved parameter, default input is 0.
nReadWriteLocation:	RFID read/write position, range: 0-999, default is 0. Unit is mm.
ReadWriteArea:	Reserved parameter, default value is 0.
nMaxErrNum:	Number of retries for read/write errors, range: 0-9, default is 1.
nErrProcessingMethod:	Reserved parameter, default input is 0.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_SetRFID(0, 0, 0, 3, 0);
PTK_RWRFIDLabel(1, 1, 2, 6, 1, "123456");
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_ReadRFIDLabelData

Description

Read RFID tag data.

Note: It supports reading via serial port, USB, and network.

Syntax

```
int _stdcall PTK_ReadRFIDLabelData(unsigned int nDataBlock, unsigned int nRFPower, unsigned
int bFeed, LPTSTR data, DWORD dataSize);
```

Parameters

nDataBlock:	Select the data area: 0 - TID, 1 - EPC, 2 - TID+EPC, 3 - USER.
nRFPower:	Reserved. Please enter 0.
bFeed:	Whether to advance one label after reading: TRUE: Advancing one label is effective. FALSE: Advancing one label is not effective.
data:	Buffer to store RFID tag data.
dataSize:	The space size of the buffer.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR data [1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_ReadRFIDLabelData(1, 23, 1, data, sizeof(data));
MessageBox(data);
PTK_CloseUSBPort();
```

Returns the data example:

TID :

E200341401260100016630C7
EPC :
12345692
TID+EPC:
TID: E200341401260100016630C7, EPC:12345692
USER:
450000000001200000058000000800024
The returned data example includes error codes (ERROR+Error Area+Error Code).
ERROR+TID+EPC0003
Error code 0003: Printing Void 0 means that TID cannot be read.
Error code 0004: Printing Void 1 means that the write operation failed.
Error code 0005: Printing Void 2 means that a duplicate TID was read.
Error code 0006: Printing Void 3 means that multiple new labels were found during re-inventory.

PTK_ReadRFIDLabelDataEx

Description

Read RFID tag data.

Note: It supports reading via serial port, USB, and network.

Syntax

```
int _stdcall PTK_ReadRFIDLabelData(unsigned int nDataBlock, unsigned int nRFPower, unsigned int bFeed, LPTSTR AccessCode, LPTSTR data, DWORD dataSize);
```

Parameters

nDataBlock:	Selects the data reading area. 0 - TID 1 - EPC 2 - TID+EPC 3 - USER 4 - TID+USER 5 - RESERVED 6 - TID+RESERVED
nRFPower:	Reserved. Please enter 0.
bFeed:	Whether to advance one label after reading: TRUE: Advancing one label is effective. FALSE: Advancing one label is not effective.
AccessCode:	Access password (format restricted to 4-byte HEX characters), with the default access password being 00000000.
data:	Buffer to store RFID tag data.
dataSize:	The space size of the buffer.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR data [1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_ReadRFIDLabelData(1, 23, 1, data, sizeof(data));
MessageBox(data);
PTK_CloseUSBPort();
```

Returns the data example:

```
TID :
    E200341401260100016630C7
EPC :
    12345692
TID+EPC:
    TID: E200341401260100016630C7, EPC:12345692
USER:
    450000000001200000058000000800024
```

The returned data example includes error codes (ERROR+Error Area+Error Code).

ERROR+TID+EPC0003

Error code 0003: Printing Void 0 means that TID cannot be read.

Error code 0004: Printing Void 1 means that the write operation failed.

Error code 0005: Printing Void 2 means that a duplicate TID was read.

Error code 0006: Printing Void 3 means that multiple new labels were found during re-inventory.

PTK_RFIDEndPrintLabel

Description

Print a label and then return the printed RFID tag data.

Note: This function includes the functionality of PTK_PrintLabel, so you don't need to call PTK_PrintLabel again to start printing.

Syntax

```
int _stdcall PTK_RFIDEndPrintLabel(unsigned int block, LPTSTR data, DWORD dataSize);
```

Parameters

block:	Selects the data reading area. 0 - TID , 1- EPC , 2 - TID+EPC, 3 - USER
data:	This is used to store and retrieve RFID tag data information.
dataSize:	Specify the size of the data space.

Returns

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR data[1024] = { 0 };
TCHAR status[1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_DrawText(20, 20, 0, 6, 2, 2, 'N', _T("313233343536")); // Print text
PTK_RWRFIDLabel(1, 0, 2, 6, 1, _T("313233343536")); // Write RFID EPC data
PTK_RFIDEndPrintLabel(1, data, sizeof(data)); // Print a label
MessageBox(data);
PTK_CloseUSBPort();
```

Returns the data example:

```
TID :
    E200341401260100016630C7
EPC :
    12345692
TID+EPC:
    TID: E200341401260100016630C7, EPC:12345692
USER:
    450000000001200000058000000800024
```

The returned data example includes error codes (ERROR+Error Area+Error Code).

ERROR+TID+EPC0003

Error code 0003: Printing Void 0 means that TID cannot be read.

Error code 0004: Printing Void 1 means that the write operation failed.

Error code 0005: Printing Void 2 means that a duplicate TID was read.

Error code 0006: Printing Void 3 means that multiple new labels were found during re-inventory.

PTK RFIDEndPrintLabelFeedBack

Description

Print a label and then return the printed RFID tag data and the printer's status.

Note: This function contains the function of PTK_PrintLabel, there is no need to call PTK_PrintLabel to start printing. This function is only supported in firmware versions V7.61 and above.

Syntax

```
int _stdcall PTK_RFIDEndPrintLabelFeedBack(unsigned int block, LPTSTR data, DWORD
dataSize,LPTSTR printerStatus, DWORD statusSize);
```

Parameters

block: Selects the data reading area.

	0 - TID , 1 - EPC , 2 - TID+EPC, 3 - USER
data:	This is used to store and retrieve RFID tag data information.
dataSize:	Specify the size of the data space.
printerStatus:	The function returns the current printer status in the format W1XXXX, where XXXX is the printer's status code.
status:	Space size of printerStatus.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR data[1024] = { 0 };
TCHAR status[1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_DrawText(20, 20, 0, 6, 2, 2, 'N', _T("313233343536")); // Print text
PTK_RWRFIDLabel(1, 0, 2, 6, 1, _T("313233343536")); // Write RFID EPC data
PTK_RFIDEndPrintLabelFeedBack(1, data, sizeof(data), status, sizeof(status));
MessageBox(data);
MessageBox(status);
PTK_CloseUSBPort();
```

Returns the data example:

```
TID :
    E200341401260100016630C7
EPC :
    12345692
TID+EPC:
    TID: E200341401260100016630C7, EPC:12345692
USER:
    45000000001200000058000000800024
```

The returned data example includes error codes (ERROR+Error Area+Error Code).

ERROR+TID+EPC0003

Error code 0003: Printing Void 0 means that TID cannot be read.

Error code 0004: Printing Void 1 means that the write operation failed.

Error code 0005: Printing Void 2 means that a duplicate TID was read.

Error code 0006: Printing Void 3 means that multiple new labels were found during re-inventory.

PTK_ReadRFIDSetting

Description

Set the return data settings after printing a UHF RFID tag.

Note: This function is supported in firmware versions V10.0 and above.

Syntax

int _stdcall PTK_ReadRFIDSetting(unsigned int nRMode, unsigned int nStartBlock, unsigned int nRBlockLength, LPTSTR AccessCode);

Parameters

nRMode: Selects the data reading area.
0 - TID
1 - EPC
2 - TID+EPC
3 - USER
4 - TID+USER
5 - RESERVED
6 - TID+RESERVED

Note: If you choose to read TID (nRMode=0), you should enter 0 for both nStartBlock and nRBlockLength.

nStartBlock: Read start address (unit: word). Read the start address (in words).
nRBlockLength: Read length (unit: byte). Read length (in bytes).
AccessCode: Access password (formatted as a 4-byte hexadecimal string), with the default access password being 00000000.

The data return format includes feedback information in the following formats:

(1) Read success feedback f Successful read feedback information format:

ZONE1,x1,y1+DATA1*ZONE2,x2,y2+DATA2*..... ZONE_n,x_n,y_n+DATA_n

(2) Failed read feedback information format.

ZONE1,x1,y1+DATA1*ZONE2,x2,y2+DATA2*..... ERROR+ZONE_m,x_m,y_m+ERRORTYPE

ZONE_n represents the nth read command data area [TID; EPC; TID+TIDDATA+EPC (TIDDATA represents the tag's TID data); USER; TID+TIDDATA+USER; RESERVED; TID+TIDDATA+RESERVED].

x_n represents the number of blocks parameter for the nth read command. When nRMode is 0 (TID), this field is not present.

y_n represents the start block parameter for the nth read command. When nRMode is 0 (TID), this field is not present.

DATA_n represents the data read by the nth read command in hexadecimal format.

ERRORTYPE represents an error code, including various categories as listed, each with a specific meaning.

0003---> Unable to scan RFID tags.

0004 ---> RFID tag write operation failed.

0005---> Able to read the TID of previously written tags but unable to read the TID of new tags.

0006 ---> Detected at least two new TIDs.

0009 ---> TID type mismatch.
0010 ---> General RFID feedback error.
0203---> Unable to scan RFID tags, attempt to print again.
0204---> RFID tag write operation failed, attempt to print again.
0205 ---> Able to read the TID of previously written tags but unable to read the TID of new tags, attempt to print again.
0206 ---> Detected at least two new TIDs, attempt to print again.
0209 ---> TID type mismatch, attempt to print again.
0210 ---> General RFID feedback error, attempt to print again.
When RFID reading fails and there's an error, the LCD screen will display "RFID Feedback Error."

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR data[1024] = { 0 };
TCHAR status[1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_DrawText(20, 20, 0, 6, 2, 2, 'N', _T("313233343536")); // Print a text
PTK_RWRFIDLabel(1, 0, 2, 6, 1, _T("313233343536")); // Write RFID EPC data
PTK_ReadRFIDSetting(2,0,5, _T("00000000"));
PTK_PrintAndCallback(data, status);
MessageBox(data);
MessageBox(status);
PTK_CloseUSBPort();
```

PTK_PrintAndCallback

Description

Print UHF or HF RFID label and feedback data. This function is used to print UHF or HF RFID tags and receive feedback data.

Note: This function contains the function of PTK_PrintLabel, there is no need to call PTK_PrintLabel to start printing. This function is supported in firmware versions V10.0 and above.

Syntax

```
int _stdcall PTK_PrintAndCallback(LPTSTR data, LPTSTR printerStatus).
```

Parameters

ata: Used to store the RFID label data information.
printerStatus: The current state of the printer, the return value format for the W1XXXX,

XXXX for the printer status code.

Note: To understand the feedback data for UHF RFID, refer to the PTK_ReadRFIDSetting function. For HF RFID, refer to the PTK_ReadHFRFIDSetting function.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR data[1024] = { 0 };
TCHAR status[1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_DrawText(20, 20, 0, 6, 2, 2, 'N', _T("313233343536"));    // Print a text
PTK_RWRFIDLabel(1, 0, 2, 6, 1, _T("313233343536"));          // Write RFID EPC data
PTK_ReadRFIDSetting(2,0,5);
PTK_PrintAndCallback(data, status);
MessageBox(data);
MessageBox(status);
PTK_CloseUSBPort();
```

PTK_SetReadRFIDForwardSpeed

Description

Setting the speed at which the label advances to the optimal read/write position when reading RFID data.

Note: This function is only supported in firmware versions V7.60 and above.

Syntax

```
int _stdcall PTK_SetReadRFIDForwardSpeed(unsigned int speed).
```

Parameters

speed: The speed at which the label advances to the optimal read/write position, measured in inches per second (ips).

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example:

```
PTK_OpenUSBPort(255);
PTK_SetReadRFIDForwardSpeed(4);
PTK_CloseUSBPort();
```

PTK_SetReadRFIDBackSpeed

Description

S Setting the speed at which the label retracts to the print line when reading RFID data.

Note: This function is only supported in firmware versions V7.60 and above.

Syntax

int _stdcall PTK_SetReadRFIDBackSpeed(unsigned int speed).

Parameters

speed: The speed at which the label retracts to the print line, measured in inches per second (ips).

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_SetReadRFIDBackSpeed(4);
PTK_CloseUSBPort();
```

HF RFID tag read/write-related functions.

PTK_RWHFLabel

Description

This function is used to write data to HF RFID tags.

Syntax

int _stdcall PTK_RWHFLabel(TCHAR nRWMode, unsigned int nStartBlock, unsigned int nBlockNum, LPTSTR pstr, BOOL Variable);

Parameters

nRWMode: RFID operation mode, reserved parameter, default is "W".

nStartBlock: The start block for writing.

Note:

- For the ISO 15693 protocol, the range is from 0 to the maximum supported block, and each block is 4 bytes in size.
- For the ISO 14443A protocol, there are two limitations:
 - The manufacturer block (Block 0) cannot be written to. Control blocks (e.g., the first 32 sectors: $4 * n - 1$; sector 32 to start: $128 + 16 * n - 1$, n is 1/2/3/4.....) cannot be used as the starting block. Each block is 16 bytes in size.

-
- **For the NTAG protocol, the range is from 4 (index 0) to the maximum supported block, and each block is 4 bytes in size.**

nBlockNum: This parameter specifies the number of blocks to be written.

pstr: This is a string with a length of 1-100 characters. You can combine "DATA", Cn, and Vn to create a composite string:
"DATA": Represents a constant string enclosed in double quotes, e.g., "POSTEK Printer."
Cn: Represents a serial number value, which must have been defined previously.
Vn: Represents a variable string, which must have been defined previously.
For example: "text1" Cn "text2" Vn.

Variable: When set to TRUE, it indicates that the string contains variable operations, so you should add \" around constant strings.
When set to FALSE, it indicates that the string does not contain variable operations, and \" are not required.
For example: PTK_RWHFLabel('W', 5, 1, _T("1234"), FALSE), and PTK_RWHFLabel('W', 5, 1, _T("\"1234\""), TRUE); have the same effect;

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

General Printing.

```
PTK_OpenUSBPort(255);  
PTK_RWHFLabel('W', 5, 1, _T("1234"), FALSE);  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

Serial Number Printing.

```
PTK_OpenUSBPort(255);  
PTK_ClearBuffer();  
PTK_DefineCounter(0, 6, 'N', _T("+1"), _T("")); //Define serial number C0  
PTK_RWHFLabel('W', 5, 1, "C0", TRUE);  
PTK_Download();  
PTK_DownloadInitVar(_T("1111")); // Initialize C0  
PTK_PrintLabel(3, 1); // Start printing labels  
PTK_CloseUSBPort();
```

PTK_SetHFRFID

Description

Set HF Tag Information.

Syntax

```
int _stdcall PTK_SetHFRFID(TCHAR pWForm, int nProtocolType, int nMaxErrNumd);
```

Parameters

WForm:	Data type for read and write. A - ASCII; H - Hexadecimal format; Default is ASCII format.
nProtocol:	Protocol type. 0 - Recognize tag type during detection and calibration. 1 - ISO 15693 protocol. 2 - ISO 14443 protocol. Default value is saved in FLASH and can be set through the printer screen or the PTK_SetAllPrinterInfo function.
nMaxErrNumd:	The number of retries for read and write operations (excluding the first attempt). After a maximum number of failures, it will print a "VOID" label. If there is a display screen, it will show an error message. Default value: 1.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_SetHFRFID('A', 1, 3);  
PTK_RWHFLabel('W', 5, 1, _T("1234"), FALSE);  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

PTK_ReadHFLabelData

Description

Read HF RFID tag data.

Note: Supports reading via serial port, USB, and network.

Syntax

```
int _stdcall PTK_ReadHFLabelData(unsigned int nStartBlock, unsigned int nBlockNum, TCHAR  
pFeed, LPTSTR data, DWORD dataSize);
```

Parameters

nStartBlock:	Start block for reading tags
nBlockNum:	Number of blocks to read from nStartBlock onwards.
pFeed:	Whether to advance one label after reading the data: Y: Advances one label after reading the information. N: No feeding action after reading the information.
data:	Buffer to store HF tag data.
dataSize:	The size of the space of data

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_ReadHFLabelData(5, 1, 'Y', buff, sizeof(buff));  
MessageBox(buff);  
PTK_CloseUSBPort();
```

PTK_ReadHFLabeUID

Description

Read HF Tag UID

Note: Supports reading via serial port, USB, and network.

Syntax

```
int _stdcall PTK_ReadHFLabeUID(TCHAR pFeed, LPTSTR data,DWORD dataSize);
```

Parameter

pFeed:	Whether to advance one label after reading the data: Y: Advances one label after reading the information. N: No feeding action after reading the information.
data:	Buffer to store HF tag UID.
dataSize:	The size of the data space.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_ReadHFLabeUID('Y', buff, sizeof(buff));  
MessageBox(buff);
```

PTK_CloseUSBPort();

PTK_ReadHFRFIDSetting

Description

Setting the data format for HF RFID tag return after printing.

Note: This function is supported in firmware versions V10.0 and above.

Syntax

```
int _stdcall PTK_ReadHFRFIDSetting(unsigned int nRMode, unsigned int nStartBlock, unsigned int nRBlockLength);
```

Parameters

nRMode: Select data area.
0 - UID , 1- BLOCK , 2 - UID+BLOCK

Note: When selecting to read UID (nRMode=0), set nStartBlock and nRBlockLength to 0.

nStartBlock: The starting block for reading.

Note: Note that the allowable range depends on the protocol being used.

- For the 15693 protocol, the range is from 0 to the maximum supported block, and each block is 4 bytes in size.
- For the 14443A protocol, there are two limitations:
 - The manufacturer block (Block 0) cannot be written to. Control blocks (e.g., the first 32 sectors: $4 * n - 1$; sector 32 to start: $128 + 16 * n - 1$, n is 1/2/3/4.....) cannot be used as the starting block. Each block is 16 bytes in size.
 - For the NTAG protocol, the range is from 4 (index 0) to the maximum supported block, and each block is 4 bytes in size.

nRBlockLength: The number of blocks to read, with a range of 1-1024.

Data return format description:

(1) The format for successful data retrieval:

ZONE1,x1,y1+DATA1*ZONE2,x2,y2+DATA2*.....

ZONEn,xn,yn+DATAn

(2) The format for failed data retrieval:

ZONE1,x1,y1+DATA1*ZONE2,x2,y2+DATA2*.....

ERROR+ZONEm,xm,ym+ERRORTYPE

ZONEn represents the data area for the nth read command [UID; BLOCK; UID+BLOCK].

xn represents the starting block parameter for the nth read command. This field is not present when nRMode is set to 0 (UID).

yn represents the number of blocks to be read for the nth read command. This field is not present when nRMode is set to 0 (UID).

DATAn represents the data retrieved by the nth read command in hexadecimal format.

ERRORTYPE represents an error code, including various categories as listed, each with a specific meaning.

0003---> Unable to scan HF RFID tags.

0004 ---> HF RFID tag write operation failed.

0005---> Able to read the UID of previously written tags but unable to read the UID of new tags.

0006 ---> Detected at least two new UIDs.

0009 ---> UID type mismatch.

0010 ---> General HF RFID feedback error.

0203---> Unable to scan RFID HF tags, attempt to print again.

0204---> RFID HF tag write operation failed, attempt to print again.

0205 ---> Able to read the UID of previously written tags but unable to read the UID of new tags, attempt to print again.

0206 ---> Detected at least two new UIDs, attempt to print again.

0209 ---> UID type mismatch, attempt to print again.

0210 ---> General RFID HF feedback error, attempt to print again.

When RFID reading fails and there's an error, the LCD screen will display "RFID Feedback Error."

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
TCHAR data[1024] = { 0 };
TCHAR status[1024] = { 0 };
PTK_OpenUSBPort(255);
PTK_DrawText(20, 20, 0, 6, 2, 2, 'N', _T("313233343536"));    //Print a text
PTK_RWHFLabel('W', 5, 1, _T("1234"), 0);
PTK_ReadHFRFIDSetting(2, 0, 5);
PTK_PrintAndCallback(data, status);
MessageBox(data);
MessageBox(status);
PTK_CloseUSBPort();
```

PTK_ReadHFTagDataPrintAuto

Description

During the printing process, first read the data content of specified blocks in HF RFID tags, and then print it on the label.

Note: This function should be used in conjunction with PTK_DrawTextEx.

Syntax

```
int _stdcall PTK_ReadHFTagDataPrintAuto(unsigned int nStartBlock, unsigned int nBlockNum);
```

Parameters

nStartBlock: Starting block for reading tags.
nBlockNum: The number of blocks to read

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_ClearBuffer();  
PTK_ReadHFTagDataPrintAuto(5, 1);  
PTK_DrawTextEx(50, 30, 0, 2, 1, 1, _T('N'), _T("R"), TRUE);  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

PTK_ReadHFTagUIDPrintAuto

Description

During the printing process, first read the UID of the HF RFID tag and then print it on the label.

Note: This function should be used in conjunction with PTK_DrawTextEx.

Syntax

```
int _stdcall PTK_ReadHFTagUIDPrintAuto(void);
```

Parameters

None

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_ClearBuffer();  
PTK_ReadHFTagUIDPrintAuto();  
PTK_DrawTextEx(50, 30, 0, 2, 1, 1, _T('N'), _T("R"), TRUE);  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

PTK_SetHFAFI

Description

To set the value of the HF RFID tag AFI.

Syntax

```
int _stdcall PTK_SetHFAFI(int nAFIValue);
```

Parameters

nAFIValue: AFI value to be set.

Returns

0 -> OK

For other return values, please call `PTK_GetErrorInfo` to parse.

Note: If the write operation fails, the tag will print a VOID label, and the screen will display RFID read/write error.

Example

```
PTK_OpenUSBPort(255);
```

PTK_ClearBuffer();

PTK_SetHFAFI(25);

PTK_PrintLabel(1, 1);

PTK_CloseUSBPort();

PTK_SetHFDSFID

Description

To set the value of the HF RFID tag DSFID.

Syntax

```
int __stdcall PTK_SetHFDSFID(int nDSFIDValue);
```

Parameters

nDSFIDValue: DSFID value to be set.

Returns

0 -> OK

For other return values, please call `PTK_GetErrorInfo` to parse.

Note: If the write operation fails, the tag will print a VOID label, and the screen will display RFID read/write error.

Example

```
PTK_OpenUSBPort(255);
```

PTK_ClearBuffer();

PTK_SetHFDSFID(15);

PTK_PrintLabel(1, 1);

PTK_CloseUSBPort();

PTK_SetHFEAS

Description

To set the value of the HF RFID tag EAS.

Syntax

```
int _stdcall PTK_SetHFEAS(TCHAR EAS);
```

Parameters

EAS: EAS value to be set, which can be "E" or "R".

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Note: If the write operation fails, the tag will print a VOID label, and the screen will display RFID read/write error.

Example

```
PTK_OpenUSBPort(255);  
PTK_ClearBuffer();  
PTK_SetHFEAS('E');  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

PTK_HFDecrypt

Description

To decrypt HF RFID tags. operating at 13.56 MHz and conforming to the ISO 14443A protocol, you can choose either keyA or keyB for decryption. Here's how it works:

Note: For HF 13.56MHz band and conform to the ISO 14443A protocol, you can choose either keyA or keyB for decryption.

Syntax

```
int _stdcall PTK_HFDecrypt(int key, unsigned int nStartBlock, unsigned int nBlockNum, char* VerifyPassword);
```

Parameters

key:	Choose the decryption key. 1 - keyA 2 - keyB
nStartBlock:	Specify the verification start block.
nBlockNum:	Specify the number of blocks to be verified.
password:	Provide the decryption password in hexadecimal format. The default is FFFFFFFFFFFFFFFF.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Note: If the write operation fails, the tag will print a VOID label, and the screen will display RFID read/write error.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_HFDecrypt(1,1,4,"222222222222");
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_LockHFLabel

Description

Lock HF RFID Tag.

Note: Electronic label for HF 13.56M band, and in line with ISO 14443A protocol tags. For HF 13.56MHz band and conform to the ISO 14443A protocol, you can choose either keyA or keyB for locking.

Syntax

```
int _stdcall PTK_LockHFLabel(int nStartBlock, int nBlockNum, char* keyA, char* keyB, char* nControlByte);
```

Parameters

nStartBlock: Start block for locking tags

Note: There is a limitation in the setting range of ISO 14443A protocol, i.e., the block of 3*n (n is 0/1/2/3/...) can not be used as the starting block.

nBlockNum: Set the number of blocks to be locked.

keyA: Set the keyA password for locking.

keyB: Set the keyB password for locking.

nControlByte: Set the control word in hexadecimal format. If set to NULL, the default value is FF078069.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Note: If the write operation fails, the tag will print a VOID label, and the screen will display RFID read/write error.

Example

1. Encode HF RFID Data and Lock

For the first time, use the default password FFFFFFFFFF. So, you decrypt with this password.

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_HFDecrypt(1,1,1,"FFFFFFFFFFFF");
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);
PTK_LockHFLabel(1,1, _T("FFFFFFFFFFFF"), _T("33333333333333"),"FF078069");
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

2. Unlock and Re-Encode Locked HF RFID Tags.

If an HF RFID tag has been locked in the first step with a keyA password of 222222222222, you can use this password for decryption. Successful decryption is required for writing tag data. You can decrypt using keyA or keyB, but typically, keyA decryption is sufficient.

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_HFDecrypt(1,1,1,"222222222222");
PTK_RWHFLabel('W', 5, 1, _T("2222222266666666"), FALSE);
PTK_LockHFLabel(1,1, _T("FFFFFFFFFFFF"), _T("33333333333333"), _T("FF078069"));
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_LockHFIdentifier

Description

Lock AFI and DSFID on ISO 15693 tag.

Syntax

```
int _stdcall PTK_LockHFIdentifier(TCHAR Identifier);
```

Parameters

Identifier: Choose the identifier you want to lock.
L - Lock AFI, U - Lock DSFID.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);
PTK_LockHFIdentifier('L');
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_LockHFBlock

Description

Lock ISO 15693/NTAG tag data blocks.

Syntax

```
int _stdcall PTK_LockHFBlock(unsigned int nStartBlock, unsigned int nBlockNum);
```

Parameters

nStartBlock:	Specify the starting block to be locked.
nBlockNum:	Specify the number of blocks to be locked.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_ClearBuffer();  
PTK_LockHFBlock(1,5);  
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

PTK_SetHFKey

Description

Set the key.

Syntax

```
int _stdcall PTK_SetHFKey((int lockType, char* keyA, char* keyB, char* keyFx);
```

Parameters

lockType:	Lock setting. 1 - lock after modifying the key; 0 - no locking.
keyA	- KeyA value
keyB	- keyB value
keyFx	- keyFx value.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetHFKey(1,"00112233","44556677","88990000");
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_SetHFCRCCommand**Description**

To set, verify, or lock a CRC password.

Syntax

```
int _stdcall PTK_SetHFCRCCommand((int lockType, char* oldCRCCommand, char*
newCRCCommand);
```

Parameters

lockType:	Lock setting. 1 - Indicate that after successfully verifying and changing the CRC password; 0 - Indicate that it should not be locked.
oldCRCCommand:	Old CRC Password: This is the current CRC password in hexadecimal format, used for verification.
newCRCCommand:	New CRC Password: This is the new CRC password in hexadecimal format, used for both verification and modification.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetHFCRCCommand(0,"11111111","66666666");
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_SetHFPrivateCommand**Description**

To set, verify, or lock a private mode password.

Syntax

```
int _stdcall PTK_SetHFPrivateCommand((int lockType, char* oldPrivateCommand, char* newPrivateCommand);
```

Parameters

lockType:	Lock setting. 1 - Indicate that after successfully verifying and changing the private mode password 0 - Indicate that it should not be locked.
oldPrivateCommand:	Old Private Mode Password: This is the current private mode password in hexadecimal format, used for verification.
newPrivateCommand:	New Private Mode Password: This is the new private mode password in hexadecimal format, used for both verification and modification.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);  
PTK_ClearBuffer();  
PTK_SetHFPrivateCommand(0,"11111111","66666666");  
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);  
PTK_PrintLabel(1, 1);  
PTK_CloseUSBPort();
```

[*PTK_LockHFUser*](#)

Description

To lock the user area.

Syntax

```
int _stdcall PTK_LockHFUser((int lockType, int nStartBlock, int nBlockNum);
```

Parameters

lockType:	Lock setting. 1 - enable locking; 0 - disable locking.
nStartBlock:	Starting Block for Locking
nBlockNum:	Number of blocks to lock.

Returns

0 -> OK

For other return values, please call `PTK_GetErrorInfo` to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_LockHFUser(0,1,2);
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_SetHFCFG10

Description

Configure CFG Set 0x10 function parameters.

Syntax

```
int _stdcall PTK_SetHFCFG10((char* CFG_Set_0x10);
```

Parameters

CFG_Set_0x10: CFG Set 0x10 function parameters in Hexadecimal format.

Returns

0 -> OK

For other return values, please call `PTK_GetErrorInfo` to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetHFCFG10("1111111166666666");
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

PTK_SetHFCFG80

Description

Configure CFG Set 0x80 function parameters.

Syntax

```
int __stdcall PTK_SetHFCFG80((char* CFG_Set_0x80);
```

Parameters

CFG_Set_0x80: CFG Set 0x80 function parameters in Hexadecimal format.

Returns

0 -> OK

For other return values, please call PTK_GetErrorInfo to parse.

Example

```
PTK_OpenUSBPort(255);
PTK_ClearBuffer();
PTK_SetHFCFG80("1111111166666666");
PTK_RWHFLabel('W', 5, 1, "1111111166666666", FALSE);
PTK_PrintLabel(1, 1);
PTK_CloseUSBPort();
```

Appendix

Table - Printer Status Code Analysis

Error/Status code	Error/Status Description
000	No Error
001	Syntax Error
004	Print head is heating up (only in industrial grade printers)
082	Ribbon Error
083	Media Error
086	Cutter Error
087	Print Head Open
088	Pause
108	The RF command (Setting the method and content area for RFID write data) failed to execute, and the wrong parameter was entered.
109	RFID label failed to write data and has reached the retry count
110	Failed to write data without exceeding the retry count
111	RFID label calibration failure
112	The RI command (Setting the way RFID reads data and the content area) failed to execute, wrong parameters were entered
116	Failure to read RFID label data

Table - National or regional frequency bands

Region Name	Country or Region	Serial Interface Region Code
NA	North America	1
NA2	North America	13
NA3	North America	14

IN	India	4
JP	Japan	5
PRC	China	6
EU3	Europe	8
KR2	Korea	9
AU	Australia	11
NZ	New Zealand	12
MY	Malaysia	16
ID	Indonesia	17
PH	Philippines	18
TW	Taiwan	19
MO	Macao	20
RU	Russia	21
SG	Singapore	22
VN	Vietnam	25
TH	Thailand	26
AR	Argentina	27
HK	Hong Kong	28
BD	Bangladesh	29

CDFPSK.dll error return value analysis

For other return values, please call PTK_GetErrorInfo to parse.