

YKPOS. DLL

interface description

[illegible]

Orders to record

I. Connection port	- 3 -
1.1 Open and close.....	- 3 -
1.2 Inspection status	- 4 -
1.3 the initialization	- 6 -
Two, standard printing instructions	- 6 -
2.1 Print control commands.....	- 6 -
2.2 Set the print format	- 8 -
Third, print pictures	- 10 -
3.1 Needle printer prints pictures.....	- 10 -
3.2 Thermal printing pictures.....	- 11 -
Four, print one-dimensional code/bar code	- 11 -
4.1 print pdf417	- 12 -
4.2 Printing QR code	- 14 -
Five, the coffers.....	- 14 -
6, call reference example (C++ pseudo code, for reference only)....	- 15 -

I. Connection port

1.1 Open and close

The function prototype	<code>int __stdcall YkOpenDevice(int iport,int baud);</code>
Parameters that	Int iport -- device communication connection number (see details) Int baud -- baud rate (only for serial ports, other interfaces pass 0)
The return value	0 -- success; -1 -- Failure;
Detailed instructions	<p>This interface is used to open a communication port Make sure you have successfully opened the communication port before calling another interface</p> <p>Communication port definition:</p> <pre>// serial port #define COM1 1 #define COM2 2 #define COM3 3 #define COM4 4 #define COM5 5 #define COM6 6 #define COM7 7 #define COM8 8 #define COM9 9 #define COM10 10 // parallel port #define LPT111 #define LPT212 // USB port #define USB 13 //(Wired or WiFi) network printer #define NET 14</pre>

The function prototype	<code>int __stdcall YkCloseDevice();</code>
Parameters that	There is no
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to close the communication port and can be called before the program exits.

1.2 Inspection status

The function prototype	<code>int __stdcall YkGetStatus(unsigned char n);</code>
Parameters that	Unsigned char n (n) unsigned char n (n) unsigned char n
The return value	>0 -- Successful, indicating the corresponding state resolution is described in detail; -1 -- Failure;
Detailed instructions	<p>This interface is used to get the printer status or return the printer status if communication is normal 1 to 4 correspond to different states, which can be obtained as needed Parsing status reference code:</p> <pre> unsigned char status[4]; status[0] = YkGetStatus(1); status[1] = YkGetStatus(2); status[2] = YkGetStatus(3); status[3] = YkGetStatus(4); if(status[0] & 0x08) { // offline } else { // online } if(status[1] & 0x20) { // When the printer runs out of paper, stop printing } else { // normal } if(status[1] & 0x08) { // Feed the paper through the feed key } else { // Do not feed through the feed key } if(status[1] & 0x04) </pre>

	<pre> { // The nose lifts open } else { // Turn off the nose lift } if(status[2] & 0x08) { // Automatic paper cutting error occurred } else { // normal } if(status[2] & 0x04) { // There is a paper blocking or paper pulling error (requires a triple nozzle machine). } else { // normal } if(status[3] & 0x60) { // paper } else { // normal } if(status[3] & 0x0c) { // the paper will do } else { // normal } </pre>
--	--

1.3 the initialization

The function prototype	<code>int __stdcall YkInitPrinter();</code>
Parameters that	There is no
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to reset the printer's logic program to clear some of the Settings information

Two, standard printing instructions

2.1 Print control commands

The function prototype	<code>int __stdcall YkPrintStr(char *pstr);</code>
Parameters that	PSTR -- The string data to be printed
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to feed the string to be printed into the printer buffer (Note: it will be automatically printed when the line is full, and the part of the line below or beyond the line will be printed immediately by adding a newline character "\n" at the end of the line, otherwise it will be cached in the cache).

The function prototype	<code>int __stdcall YkPrintStr_utf8(char *pstr);</code>
Parameters that	PSTR -- string data to be printed, UTF-8 encoding
The return value	0 -- success; -1 -- Failure;
Detailed instructions	Ditto, this interface converts UTF-8 encoded data to GBK and sends it to the printer

The function prototype	<code>int __stdcall YkEnter();</code>
Parameters that	There is no
The return value	0 -- success; -1 -- Failure;
Detailed	This interface is used to print and enter, but does not move paper

instructions	
--------------	--

The function prototype	<code>int __stdcall YkFeedPaper();</code>
Parameters that	There is no
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to print and wrap the paper to the beginning of the next line

The function prototype	<code>int __stdcall YkPrnAndFeedLine (int n);</code>
Parameters that	N lines of characters on paper
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used for n lines of paper after printing

The function prototype	<code>int __stdcall YkPrnAndFeedPaper (int n);</code>
Parameters that	N =0~255 paper distance
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used for printing paper N dot line (Note: it can accurately walk paper 1 dot line =0.125mm)

The function prototype	<code>int __stdcall YkCutPaper(int m,int n);</code>
Parameters that	M = 66;N: The printer feeds the paper to (cutting position + N * 0.125mm) and cuts the paper, generally N =0
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to perform paper cutting actions, including feeding Also used for black mark cutting when opening black mark The normal direct call is as follows: <code>YkCutPaper (66, 0);</code>

The function prototype	<code>int __stdcall YkMoveBack (int n);</code>
Parameters	N The number of characters in the back paper

that	
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to return N lines of paper, only for note printing

The function prototype	<code>int __stdcall YkMoveBack_f(float n);</code>
Parameters that	N Return distance, unit mm, is a multiple of 0.125
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used for returning paper N mm, only for note printing

2.2 Set the print format

Format commands need to be called before sending print data to take effect, and set commands are cleared after initialization.

The function prototype	<code>Int __stdcall YkSetLineSpace (float fLineSpace = 3.75);</code>
Parameters that	FLINESPACE -- Units of millimeter 0.0 to 30.0
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the print line spacing, which is set to fLineSpace millimeters

The function prototype	<code>int __stdcall YkSetLeftMargin(int nL,int nH);</code>
Parameters that	NL NH Left Distance =(NL + NH x 256) x 0.125 mm
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the left margin for printing

The function prototype	<code>int __stdcall YkSetAlign(int n);</code>
Parameters that	N =0 left align; N = 1 center; n=2 Align right

The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the alignment when printing

The function prototype	<code>int __stdcall YkSetCharSize(int hsize,int vsize);</code>
Parameters that	HSIZE Horizontal Magnification VSIZE is placed vertically in the multiple, and the value range is 0~7
The return value	0 -- success; -1 -- Failure;
Detailed instructions	<p>This interface is used to set the character magnification HSIZE, VSIZE value and multiple corresponding relationship 0-----1 times (original size) 1-2 times 2-3 times 3-4 times 4-5 times 5-6 times 6-7 times 7-8 times</p>

The function prototype	<code>int __stdcall YkEnableUnderLine(int n=0);</code>
Parameters that	N =1 Enable, N =0 Disable
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to enable or disable character underlining

The function prototype	<code>int __stdcall YkEnableChineseUnderLine(int n);</code>
Parameters that	N =1 Enable, N =0 Disable
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to enable or disable the underlining function of Chinese characters

The function prototype	<code>int __stdcall YkSetEmphasized(int n);</code>
Parameters	N =1 Enable, N =0 Disable

that	
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to enable or disable weighted printing mode

The function prototype	<code>int __stdcall YkOverlap(int n);</code>
Parameters that	N =1 Enable, N =0 Disable
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to enable or disable overlapping print mode

The function prototype	<code>int __stdcall YkClockwiseD90(int n);</code>
Parameters that	N =1 Enable, N =0 Disable
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to enable or disable clockwise 90 degree rotation of character printing

The function prototype	<code>int __stdcall YkEnableUpsidedown(int n);</code>
Parameters that	N =1 Enable, N =0 Disable
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to enable or disable reverse printing mode

Third, print pictures

Print image requirements: the image must be a monochrome bitmap in BMP format.

3.1 Needle printer prints pictures

The function prototype	<code>int __stdcall YkPrintBitmapMatrix(char *szBmpFile);</code>
------------------------	--

Parameters that	SzBmpFile - Path to the image file
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used for needle printer to print pictures

3.2 Thermal printing pictures

The function prototype	<code>int __stdcall YkPrintRasterBmp(char *szBmpFile);</code>
Parameters that	SzBmpFile - Path to the image file
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used for printing raster bitmap by thermal printer

Other interfaces are not introduced for the moment

Four, print one-dimensional code/bar code

The function prototype	<code>int __stdcall YkSetHRIPos(int n=0);</code>
Parameters that	N =0 does not print, N =1 above the bar code, N =2 below the bar code, N =3 above and below the bar code
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the printing position of the HRI characters of the barcode

The function prototype	<code>int __stdcall YkSetHRICharStyle(int n);</code>
Parameters that	0 -- font A (12 × 24) 1 -- font B (9 × 17)
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the HRI character font of the bar code

The function prototype	<code>int __stdcall YkSetBarCodeHeight(float fHeight);</code>
------------------------	---

Parameters that	FHeight -- bar code height, in millimeters between 0.0 and 30.0
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the bar code height

The function prototype	<code>int __stdcall YkSetBarCodeWidth(float fWdith);</code>
Parameters that	FWdith -- bar code width setting, 0.250 ~ 0.750
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the bar code width

The function prototype	<code>int __stdcall YkPrintBarCode(int m,int n,char * barcode);</code>
Parameters that	M -- barcode type, n -- length of data to be printed, barcode -- data to be converted into barcode
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to print barcodes Each type of bar code supports a different character set, please refer to the user manual for selection.

4.1 print pdf417

The function prototype	<code>int __stdcall SetPDF417ModWidth(int ModWidth=3);</code>
Parameters that	ModWidth ranges from 2 to 8, default = 3
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the size and width of 2D barcode PDF417 module

The function prototype	<code>int __stdcall SetPDF417ModHeight(int ModHeight=3);</code>
Parameters that	ModHeight ranges from 2 to 8, default = 3
The return value	0 -- success; -1 -- Failure;

Detailed instructions	This interface is used to set the size and height of 2D barcode PDF417 module
-----------------------	---

The function prototype	<code>int __stdcall SetPDF417Level(int Level=1);</code>
Parameters that	Level ranges from 1 to 40, default = 1
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the error correction level of 2D barcode PDF417

The function prototype	<code>int __stdcall SetStdPDF417(int iStd=0);</code>
Parameters that	iStd =0, set the standard PDF417 barcode =1, set the truncated PDF417 barcode to 0 by default
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set or cancel the standard PDF417 barcode

The function prototype	<code>int __stdcall SetPDF417RowCol(int iRow=0,int iCol=0);</code>
Parameters that	IRow iCol default iCol = 0; iRow=0; $0 \leq \text{ICOL} \leq 30$; $\text{IRow} = 0 \text{ or } 3 \leq \text{iRow} \leq 90$; When ICOL =0, the number of barcode columns is automatically adjusted according to the effective printing range; If ICOL $\neq 0$, set the number of barcode columns as ICOL columns; When iRow=0, the number of barcode lines is automatically adjusted according to the effective print range; If iRow $\neq 0$, set the number of barcode lines to iRow;
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the number of PDF417 barcode lines and rows

The function prototype	<code>int __stdcall PrintPDF417Code(unsigned char *pCode,int iLen);</code>
Parameters that	PCODE, barcode data ILEN, barcode data length range $0 < \text{ILEN} < 256$
The return value	0 -- success; -1 -- Failure;
Detailed	This interface is used to print PDF417 barcode

instructions	
--------------	--

4.2 Printing QR code

The printing of QR code requires the corresponding model support.

The function prototype	<code>int __stdcall SetQRModSize(int ModWidth=3);</code>
Parameters that	ModWidth ranges from 2 to 16, default = 3
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the module size of the two-dimensional code

The function prototype	<code>int __stdcall SetQRLevel(int Level=48);</code>
Parameters that	Level ranges from 48 to 51, default = 48
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to set the error correction level of two-dimensional code

The function prototype	<code>int __stdcall PrintQRCode(unsigned char *pCode,int iLen);</code>
Parameters that	PCODE, QR code data ILEN, QR code data length range $0 < ILEN < 256$
The return value	0 -- success; -1 -- Failure;
Detailed instructions	This interface is used to print a QR code

Five, the coffers

The function prototype	<code>int __stdcall YkSetCashBoxDriveMode(int m,int t1,int t2);</code>
Parameters that	Using that pin, the output pulse is m=0, 2 pins m=1, 5 pins Pulse width is $T1 * 2ms$ $T2 * 2ms$, where $T1 < T2$ is required, generally $T1 = 150$ $T2 = 250$
The return value	0 -- success; -1 -- Failure;

Detailed instructions	This interface is used to open the money box The default method is: <code>YkSetCashBoxDriveMode(0, 150, 250);</code>
-----------------------	--

6, call reference example (C++ pseudo code, for reference only)

```
int main()
{
    if(usb)
    {
        YkOpenDevice(13, 0);
    }
    else if(com)
    {
        YkOpenDevice(1, 38400);
    }
    else if(lpt)
    {
        YkOpenDevice(11, 0);
    }

    unsigned char status[4];
    status[0] = YkGetStatus(1);
    // status[1] = YkGetStatus(2);
    // status[2] = YkGetStatus(3);
    status[3] = YkGetStatus(4);

    if (status[0] & 0x08)
    {
        Cout <<" Printer offline "<< endl;
    }

    if (status[3] & 0x60)
    {
        Cout <<" Printer paper runs out "<< endl;
    }

    if (status[3] & 0x0c)
    {
        Cout <<" The printer is running out of paper "<< endl;
    }
}
```

```
Char buff[] = "print test abc123";
YkPrintStr(buff);
YkPrnAndFeedLine(1);

YkSetEmphasized(1);
YkPrintStr(buff);
YkPrnAndFeedLine(1);

YkSetAlign(1);
YkPrintStr(buff);
YkPrnAndFeedLine(1);

YkSetCharSize(1, 1);
YkPrintStr(buff);
YkPrnAndFeedLine(1);

YkInitPrinter();
YkPrintStr(buff);
YkPrnAndFeedLine(1);

char szBmpFile[] = "111.bmp";
YkPrintRasterBmp(szBmpFile);

YkSetBarCodeHeight (20.25);
YkSetBarCodeWidth (0.25);
YkSetHRIPos(2);
YkSetHRICharStyle(0);
char barcode[] = "123456789ABC";
YkPrintBarCode(72, 12, barcode);

SetPDF417ModWidth(3);
SetPDF417ModHeight(3);
SetPDF417Level(1);
PrintPDF417Code((unsigned char *)barcode, 12);

SetQRModSize(6);
SetQRLevel(48);
PrintQRCode((unsigned char *)barcode, 12);

YkCutPaper(66, 0);

YkCloseDevice();
}
```