

# POSTEK PPLE

API Manual

Version 2.04

POSTEK ELECTRONICS CO., LTD.

**2010**

## ● Description of API functions file

Name: WINPSK.dll

version number: 9.0.0.8

Copyright: ©2003–2010 by Postek Electronics CO., LTD. All rights reserved.

## ● Purpose

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 Windows9X, NT, 2000, XP and many more.

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

## ● Abbreviation comparison

PPL I: The first set of Printer Program Language of the Postek Electronics CO., LTD.

API: Application Program Interface.

Dots: Pixel is a kind dimensional unit used in computer science technology. Originally means the minimum unit of the TV imaging. The minimum imaging unit for a printer: Dots are equal to one inch divided by the maximum resolution of the printer.

- For the 203 DPI printer, 1dot = 25.4mm/203 = 0.125mm (1dot = 1000 / 203 = 5mil);
- For the 300 DPI printer, 1dot = 25.4mm/300 = 0.085mm (1dot = 1000 / 300 = 3mil).

TrueType Font: based on the Windows operating system, can be loaded or unloaded.

-All installed TrueType Font can be used by this function.

## ● Notice

### 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:

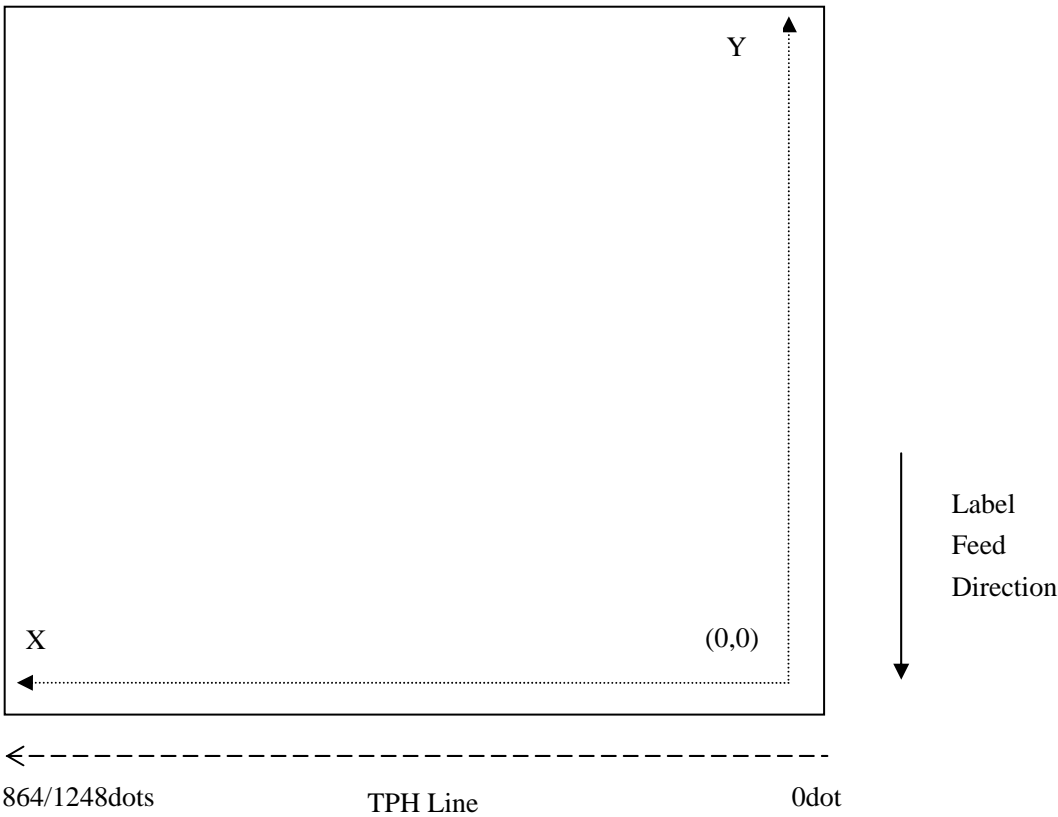
<u>character</u>	<u>Enter into Data Field</u>
"	\ "
\	\\
0x00 - 0x7F	\x00 - \x7F

\* 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 "enter" symbol.

- Coordinate system

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



- Functions

Function name	Description
<a href="#"><u>OpenPort</u></a>	Open the communication port.
<a href="#"><u>ClosePort</u></a>	Close the communication port that opened by OpenPort function.
<a href="#"><u>SetPCComPort</u></a>	Set the baudrate speed of the serial port.
<a href="#"><u>GetErrState</u></a>	Check if there's any error after using other functions in CDFPSK.DLL.
<a href="#"><u>PTK_GetInfo</u></a>	Obtain the edition information of this API functions.
<a href="#"><u>PTK_DrawText</u></a>	Print text letters, the content can be constant, counter value, variable or combined string.
<a href="#"><u>PTK_DrawTextTrueTypeW</u></a>	Print TrueType Font, the width and height of the characters can be adjusted.
<a href="#"><u>PTK_DrawBarcode</u></a>	Print bar code.
<a href="#"><u>PTK_DrawBar2D_Pdf417</u></a>	Print 2D bar code
<a href="#"><u>PTK_DefineCounter</u></a>	Define counter variable.
<a href="#"><u>PTK_SetDarkness</u></a>	Set the darkness of the TPH.
<a href="#"><u>PTK_SoftFontList</u></a>	Print the soft font list stored in the RAM or flash memory.
<a href="#"><u>PTK_SoftFontDel</u></a>	Delete one or all of the soft fonts stored in the RAM or flash memory.
<a href="#"><u>PTK_FormEnd</u></a>	End form store, it must be used together with the PTK_FormDownload.
<a href="#"><u>PTK_FormList</u></a>	Print the form name list stored in the printer.
<a href="#"><u>PTK_FormDel</u></a>	Delete one or all of the forms stored in the printer.
<a href="#"><u>PTK_ExecForm</u></a>	Execute the designated form.
<a href="#"><u>PTK_FormDownload</u></a>	Store a form to the printer, it must be used together with the PTK_FormEnd.
<a href="#"><u>PTK_DrawPcxGraphics</u></a>	Print the designated graphics.
<a href="#"><u>PTK_PcxGraphicsList</u></a>	Print the graphics name list stored in the RAM or flash memory.
<a href="#"><u>PTK_PcxGraphicsDel</u></a>	Delete one or all of the graphics stored in the printer.
<a href="#"><u>PTK_PcxGraphicsDownload</u></a>	Store a PCX graphic to the printer.
<a href="#"><u>PTK_PrintPCX</u></a>	Print a PCX graphic.
<a href="#"><u>PTK_DrawBinGraphics</u></a>	Print binary graphics.
<a href="#"><u>PTK_DisableCircumgyrate</u></a>	Disable the function of circumgyration.
<a href="#"><u>PTK_EnableCircumgyrate</u></a>	Set the function of circumgyration.
<a href="#"><u>PTK_DrawLineXor</u></a>	Draw straight-line (If two straight-lines intersects, use an exclusive OR operation).
<a href="#"><u>PTK_DrawLineOr</u></a>	Draw straight-line (If two straight-lines intersects, use an OR operation).
<a href="#"><u>PTK_DrawDiagonal</u></a>	Draw diagonal.

<a href="#"><u>PTK_DrawWhiteLine</u></a>	Draw white straight-line.
<a href="#"><u>PTK_ClearBuffer</u></a>	Clear the contents in printer buffer.
<a href="#"><u>PTK_SetPrinterState</u></a>	Set the printer's working state.
<a href="#"><u>PTK_PrintLabel</u></a>	Order the printer to execute printing work.
<a href="#"><u>PTK_PrintLabelAuto</u></a>	Auto-execute the printing work.
<a href="#"><u>PTK_SetLabelHeight</u></a>	Set label's height and the height of media gap/black line/ hole.
<a href="#"><u>PTK_SetLabelWidth</u></a>	Set label's width.
<a href="#"><u>PTK_SetCoordinateOrigin</u></a>	Set/change the coordinate reference point.
<a href="#"><u>PTK_SetPrintSpeed</u></a>	Set printing speed.
<a href="#"><u>PTK_PrintConfiguation</u></a>	The current setting for printer/printer's working state.
<a href="#"><u>PTK_DisableErrorReport</u></a>	Cancel the error feedback.
<a href="#"><u>PTK_EnableErrorReport</u></a>	Set the error feedback.
<a href="#"><u>PTK_DefineVariable</u></a>	Define variable.
<a href="#"><u>PTK_DrawRectangle</u></a>	Draw rectangle..
<a href="#"><u>PTK_SetDirection</u></a>	Set label printing direction.
<a href="#"><u>PTK_EnableFLASH</u></a>	Select the flash memory.
<a href="#"><u>PTK_DisableFLASH</u></a>	Cancel Select the flash memory.
<a href="#"><u>PTK_Download</u></a>	Download variable or counter variable.
<a href="#"><u>*PTK_Reset</u></a>	Reset the printer.
<a href="#"><u>*PTK_BackFeed</u></a>	Require the printer feedback the error report at once.
<a href="#"><u>PTK_CutPage</u></a>	Set cutter's working cycle (namely the cutter will start to cut labels once how many labels have been printed)

The function with sign \* cannot be used for the moment.

#### ● Expounding of the Function

# OpenPort

## Description:

**This function is used to open the communications port.**

**The OpenPort function must be executed correctly prior to being able to use the other functions in this directory.**

## Syntax:

```
int OpenPort(LPTSTR xxxx);
```

## Parameter:

xxxx: the name of the current printer used by windows.

## Return value:

0 -> OK;

Other returns: Please refer to chapter: **WINPSK.dll** erroneous return value specification.

## Example:

```
int return = OpenPort("POSTEK G-2108"); // means open the  
communication port currently chosen by G-2108.
```

# ClosePort

## Description:

**This function is used to close the communication port which opened by the OpenPort function.**

**it suggests that call the ClosePort to close the communication port after completing all the operation operated by other functions. Otherwise, your program will always take up the opened communication port until the program be closed.**

## Syntax:

```
int ClosePort(void);
```

Parameters: none

## Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
ClosePort( );
```

## **SetPCComPort**

### **Description:**

**This function is used to set the baudrate speed of the serial port in PC.**

**It is only available when use the serial port.**

### **Note:**

**Be sure to corresponded with the serial baudrate setting in your printer (by adjusting the DIP's 7,8PIN on-off, please refer to the user's manual).**

### **Syntax:**

```
int SetPCComPort(DWORD BaudRate, BOOL HandShake);
```

### **Parameters:**

BaudRate: set serial port baudrate, values:  
9600, 19200, 38400, 57600;

HandShake: whether uses HandShaking or not;  
TRUE: enable HandShaking,  
FALSE: disable HandShaking.

### **Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

### **Example:**

```
SetPCComPort ( 9600, TRUE);
```

## **GetErrState**

### **Description:**

This function is used to check the correctness/validity of the other functions in CDFPSK.DLL.

**Please refer to the section of CDFPSK.dll description of returning errors for error codes.  
This function must use before the Closeport().**

**Syntax:**

```
int GetErrState(void);
```

**Parameters:** none

**Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

**Example:**

```
int state = 0;
OpenPort(1);
...
state = GetErrState();
...
ClosePort();
```

## **PTK\_GetInfo**

**Description:**

**This function is used to get the edition information of this API DLL.**

**Syntax:**

```
int PTK_GetInfo(void)
```

**Parameters:** none

**Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

**Example:**

```
PTK_GetInfo(void)
```



## PTK\_DrawText

### Description:

This function is used to print text, the content can be constant, counter value, variable or combined string.

### Syntax:

```
int PTK_DrawText ( unsigned int px, unsigned int py,  
                  unsigned int pdirec, unsigned int pFont,  
                  unsigned int pHorizontal, unsigned int pVertical,  
                  char ptext, LPTSTR pstr );
```

### Parameters:

px: X coordinate in dots.

py: Y coordinate in dots.

pdirec: select printing Direction. 0—No rotation; 1—90° rotation; 2—180° rotation; 3—270° rotation.

pFont: Selects internal fonts or softfonts, 1—5: internal fonts; A—Z: downloaded soft fonts.

**a: 24\*24 simple Chinese fonts in printer.**

Value	Description
1	Foreign language fonts1
2	Foreign language fonts 2
3	Foreign language fonts 3
4	Foreign language fonts 4
5	Foreign language fonts 5
a	24*24 dot matrix Chinese font
A~Z	Soft fonts

pHorizontal: Horizontal multiplier expands the text horizontally. Value range: 1-24

pVertical: Vertical multiplier expands the text vertically. Value range: 1-24  
The acceptable values for both pHorizontal and pVertical are from 1 to 24.

ptext: Choosing 'N' prints normal text (i.e. black characters on a white background)

Choosing 'R' prints reversed text (i.e. white characters on a black background)

pstr: A character string (length is 1 to 100), using "DATA", Cn and Vn parameters

"DATA": A data fixed string, it must begin and end with " " Example: "POSTEK Printer".

Cn: A counter value. Refer to C command.

Vn: A variable string. Refer to V command.

Example: "data1" CnVn "data2".

#### Returns:

0-> OK;

Other returns: Please make reference to chapter: CDFPSK.DLL erroneous return value specification.

#### Example:

```
PTK_DrawText (50,30,0,2,1,1,'N','\123456789\");
```

```
PTK_DrawText (50,30,0,2,1,1,'N',C1);
```

```
PTK_DrawText (50,30,0,2,1,1,'N',V3);
```

```
PTK_DrawText (50,30,0,2,1,1,'N',' "Printer" C2V1 "is ok. " ");
```

## PTK\_DrawBarcode

#### Description:

This function is used to **print a bar code**.

#### Syntax:

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

#### Parameters:

Px: X coordinate in dots.

py: Y coordinate in dots.

pdirec: select printing Direction. 0—No rotation; 1—90° rotation; 2—180° rotation; 3—270° rotation.

pCode: Bar code selection

P4 Value	Bar Code 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:** Narrow bar width in dots.

**pHorizontal:** Wide bar width in dots.

**pVertical:** Bar code height in dots.

**ptext:** N-no print the readable characters under barcode.

B- print the readable characters under barcode

**pstr:** A combined character string (length is 1 to 100) using “DATA” ,

Cn, Vn.

“DATA” : A data field string, it must begin and ended with “” .

Example: “POSTEK Printer”.

Cn: A counter value. Refer to C command.

Vn: A variable string. Refer to V command.

Example: “data1” CnVn “data2” .

**Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_DrawBarcode (50,30,0,'1A',1,1,10,'N',"123456");
PTK_DrawBarcode (50,30,0,'1A',1,1,10,'N',C2);
PTK_DrawBarcode (50,30,0,'1A',1,1,10,'N',V1);
PTK_DrawBarcode (50,30,0,'1A',1,1,10,'N',C1" is "V2);
```

## PTK\_DrawBar2D\_Pdf417

### Description:

This function is used to print a PDF417 barcode.

Syntax:

```
int 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)

Parameter:

unsigned int	x;	X coordinates;
unsigned int	y;	Y coordinates;

Note: 1 dot = 0.125 mm.

unsigned int	w;	Maximum printing width, in dots;
unsigned int	v;	Maximum printing height, in dots;
unsigned int	s;	correction degrees, range: 0~8.
unsigned int	c;	compression degrees, range: 0 or 1.
unsigned int	px;	Module width, range: 2~9 dots.
unsigned int	py;	Module height, range: 4~99 dots.
unsigned int	r;	Max row count.
unsigned int	l;	Max column count.
unsigned int	t;	Truncation flag, '0' means normal and '1' means truncated.

unsigned int	o;	print direction positioning, '0' means 0° , '1' means 90° , '2' means 180° , '3' means 270°
--------------	----	---

LPCTSTR pstr; A character string (length is 1 to 100), using "DATA", Cn and Vn parameters.

"DATA": A data fixed string, it must begin and end with " " Example: "POSTEK Printer".

Cn: A counter value. Refer to C command.  
Vn: A variable string. Refer to V command.  
Example: **“data1” CnVn “data2” .**

Return value: 0 -> OK.  
Reference Error.txt file.

Example: unsigned int x, y, w, v, s, c, px, py, r, l, t, o;  
LPCTSTR pstr = "POSTEKINFO";  
x=10;y=10;w=400;v=300;s=0;c=0;px=3;py=7;r=10;l=2;t=0;o=0;  
PTK\_DrawBar2D\_Pdf417 (x, y, w, v, s, c, px, py, r, l, t, o, pstr);

## **PTK\_DrawTextTrueTypeW**

### **Note:**

**Be sure to install the printer driver of POSTEK G-2108 or POSTEK G-3106 prior to using PTK\_DrawTextTrueTypeW()**

### **Description:**

**This function is used to print a group TrueType Font characters. And the character width and height can be adjusted.**

### **Syntax:**

```
int PTK_DrawTextTrueTypeW( int x, int y,  
                           int FHeight, int FWidth,  
                           LPCTSTR FType, int Fspin,  
                           int FWeight, BOOL FItalic,  
                           BOOL FUnline, BOOL FStrikeOut,  
                           LPCTSTR id_name, LPCTSTR data )
```

### **Parameters:**

Px: X coordinate in dots.

py: Y coordinate in dots.

FHeight: Font type height in dots.

FWidth: Font type width in dots.

**\* If you want to print the normal scale font, set FWidth to 0.**

FType: Font type name.

Fspin: Font rotation. 1—No rotation; 2—90° rotation; 3—180° rotation;  
4—270° rotation.

Fweight: Font thickness.

0 and 400 -> 400 standard,  
 100 -> extra thin, 200 -> tiny thin,  
 300 -> thin, 500 -> middling,  
 600 -> half thick, 700 -> thick,  
 800 -> extra thick, 900 -> bolder.

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

**Funline:** Add base line, 0 -> FALSE, 1 -> TRUE.

**FstrikeOut:** Add strikethrough, 0 -> FALSE, 1 -> TRUE.

**id\_name:** Identify the name. True type characters will be transferred to PCX data and store to the printer as the name of id\_name. users can Call id\_name to print this true type characters by PTK\_DrawPcxGraphics( )for several times before powering off.

**data:** The contents of string.

#### Returns:

0-> OK;  
 Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

#### Example:

Print Chinese fonts for 3mm heigh:

For the 203 DPI models printer, FHeight is  $3 / 0.125 = 24\text{dots}$ ;

For the 300 DPI models printer, FHeight is  $3 / 0.08 = 38\text{dots}$  (round).

```
PTK_DrawTextTrueTypeW (30, 35, 24, 0, "宋体", 4, 400, 0, 0, 0, "A1", "机要绝密");
```

## PTK\_DefineCounter

#### Description:

This function is used to **define a counter variable**.

#### Syntax:

```
int PTK_DefineCounter ( unsigned int id, unsigned int maxNum,
                        char ptext, LPTSTR pstr, LPTSTR pMsg );
```

#### Parameters:

**id:** Counter ID, acceptable values are from 0 to 9.

**maxNum:** Maximum digit number, acceptable values are from 1 to 40.

**ptext :** Justification code. L for left justification, R for right justification, C for centralization and N for no justification.

**Pstr:** Counter change rule: "+" or "-" sign followed by a single digit of 1

- 9, then followed by a change symbol (i.e. D – decimal base, B – binary system, O – octonary number system, H – hexadecimal system, X – user defined pattern, to a maximum of 64 characters. )

Step values:

“+1” = Increases each time by 1, according to Decimal base computation. Example: 1234, 1235, 1236, ...

“+3D” = Increases each time by 3, according to Decimal base computation. Example: 1234, 1235, 1236, ...

“-1B” = Decreases each time by 1, according to Binary computation. Example: 1111, 1110, 1101, ...

“-4O” = Decreases each time by 4, according to Octonary number system computation. Example: 1234, 1230, 1224, ...

“-6H” = Decreases each time by 1, according to hexadecimal base computation. Example: 1234, 122E, 1228, ...

“+3X” = Increase each time by 3, according to a user-defined pattern. Example: In user-defined rule: TE2DOKLU046MNY37, the starting value is “T062”, followed by T062, T06K, T060, ...

pMsg: A text string that will be sent to LCD or KDU.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_DefineCounter (0, 6, ' N' , " +1" , "\n Enter\n" Code:");
```

## **PTK\_SetDarkness**

### **Description:**

This function is used to **set darkness of the TPH.**

Syntax:

```
int PTK_SetDarkness (unsigned int id);
```

Parameters:

id: Acceptable values are from 0 to 20, the default value is 10.

This value is not in deed the temperature of the TPH, it is a relative value. The lightest print darkness is achieved with a value of 0 and the greatest print

darkness is achieved with a value of 20.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_SetDarkness (10);
```

## **PTK\_SoftFontList**

### **Description:**

This function will cause the printer to print a list of all soft fonts that are stored in RAM or FLASH memory.

Syntax:

```
int PTK_SoftFontList (void);
```

Parameters: none

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_SoftFontList ();
```

## **PTK\_SoftFontDel**

### **Description:**

This function causes the printer to remove one or all, soft fonts stored in RAM and/or Flash memory.

Syntax:

```
int PTK_SoftFontDel (char pid);
```



Parameters:

pid: Soft font ID, A—Z,\*

If pid = '\*' ,all fonts will be deleted from RAM or flash memory.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_SoftFontDel ('A');
```

## **PTK\_FormEnd**

**Description:**

This function is used to end a form stored in the printer's memory. e, it must be used together with the PTK\_FormDownload.

Syntax:

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

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_FormDownload ("Form1") ;  
...  
PTK_FormEnd ( );
```

## **PTK\_FormList**

**Description:**

This function causes the printer to print a list of forms that have been stored.

Syntax:

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

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_FormList ( );
```

## **PTK\_FormDel**

**Description:**

**This function causes the printer to delete forms currently stored.**

Syntax:

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

Parameters:

pid: Form name with a maximum of 16 characters.

If pid = “\*”, all forms will be deleted from RAM or flash memory.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_FormDel (“FORMNAME”);
```

## **PTK\_ExecForm**

**Description:**

**This function is used to execute the designated form.**

Syntax:

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

**Parameters:**

pid: Form name with a maximum of 16 characters.

**Returns:**

0-> OK;

Other returns: Please make reference to chapter: CDFPSK.DLL erroneous return value specification.

**Example:**

```
PTK_ExecForm ("FORM1" );
```

## **PTK\_FormDownload**

**Description:**

**This function is used to store a form to the printer , it must be used together with the PTK\_FormEnd.**

**If this function is used after EnableFLASH ( ) , the form will be save to FLASH memory.**

**If this function is used under default state or after DisableFLASH ( ) , the form will be saved to RAM.**

**Syntax:**

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

**Parameters:**

pid: User-defined form name with a maximum of 16 characters. Use this form prior to execute this function after storing the form to the printer

**Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

**Example:**

```
PTK_FormDownload ("FORMNAME" );
```

## **PTK\_DrawPcxGraphics**

**Description:**

**This function is used to print the designated graphics.**

**Note: The graphics must store in the printer by using PTK\_PcxGraphicsDownload before it to be printed.**

Syntax:

```
int PTK_DrawPcxGraphics (unsigned int px, unsigned int py, LPTSTR gname);
```

Parameters:

px: X coordinate in dots.

py: Y coordinate in dots.

game: Graphics name with a maximum of 16 characters, it must be user-defined name in the PTK\_PcxGraphicsDownload ( ) .

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_DrawPcxGraphics (100, 50, "PCX1" );
```

## **PTK\_PcxGraphicsList**

**Description:**

**This function causes the printer to print the list of graphics that stored to RAM or flash memory from host.**

Syntax:

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

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_PcxGraphicsList ( );
```

## **PTK\_PcxGraphicsDel**

### **Description:**

**This function causes the printer to delete graphics currently stored in RAM or flash memory.**

### **Syntax:**

```
int PTK_PcxGraphicsDel (LPTSTR pid);
```

### **Parameters:**

pid: Graphics name with a maximum of 16 characters.

if pid = “\*”, all graphics will be deleted from RAM or flash memory.

### **Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

### **Example:**

```
PTK_PcxGraphicsDel ( “PCX2” );
```

## **PTK\_PcxGraphicsDownload**

### **Description:**

**This function causes the printer to store a PCX graphics to printer.**

### **Syntax:**

```
int PTK_PcxGraphicsDownload (char* pcxname, char* pcxpath);
```

### **Parameters:**

pcxname: User-defined graphics name with a maximum of 16 characters. graphics can only be printed by using this name in PTK\_DrawPcxGraphics () after the graphics being stored to the printer

pcxpath: The path of the PCX graphics in memory.

### **Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_PcxGraphicsDownload ( "PCXA", "c:\\test1111.pcx");
```

**PTK\_PrintPCX**

**Description:**

**This function is used to print a PCK graphics.**

Using this function is equal to use `PTK_PcxGraphicsDownload` ( ) and `PTK_DrawPcxGraphics` ( ) at the same time.

Syntax:

```
int PTK_PrintPCX (unsigned int px, unsigned int py, char* filename);
```

Parameters:

px: X coordinate in dots.

py: Y coordinate in dots.

filename: PCX Graphics name with a maximum of 16 characters, include the path of the file.

Format: "XXXXXXXXX.XXX" or "X:\\\\XXX\\\\XXX.PCX".

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK PrintPCX(10, 100, "c:\\phone.pcx");
```

## PTK DrawBinGraphics

**Description:**

**This function causes the printer to print binary graphics.**

**Binary graphics is non-compressed graphics data. Each bit represents a dot., 0 represent print. a value of “0” prints a dot; a value of “1” does not print a dot.**

Syntax:

[illegible]

UCHAR\* Gdata );

#### Parameters:

px: X coordinate in dots.

py: Y coordinate in dots.

pbyte: Bytes for one line data. If 8 cannot divide the bits of one line data, then the bytes equal to the result add 1. Example: the bytes of one line data is 2(14 bits data),

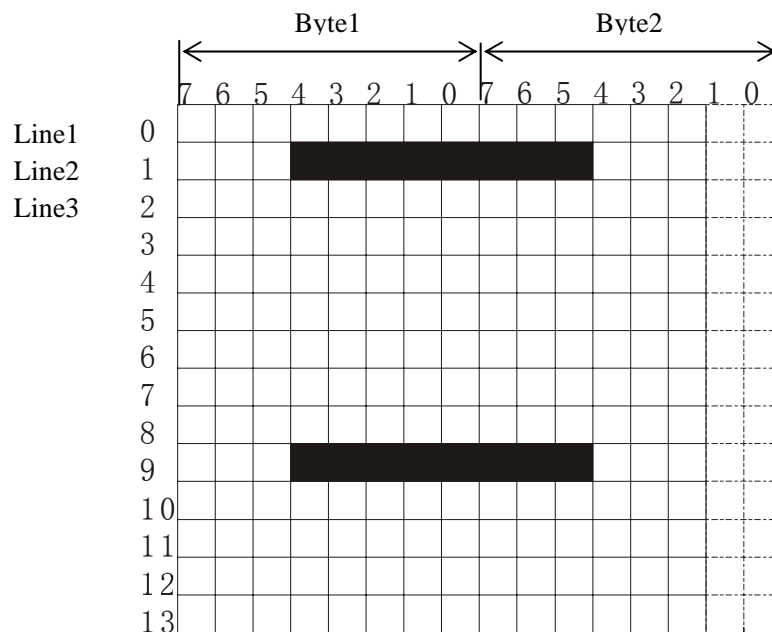
Ph: Height of graphic, in dots.

Gdata([···raster data···]): Binary graphic data; data size = pbyte \* pH (Bytes)。

Binary data transmission sequence is left to right, up to down, example:

data transmission sequence : Line1' s Byte1(0xff), Line1' s Byte2(0xff) , Line2' s Byte1(0xe0),Line2' s Byte2(0x1f), Line3' s Byte1(0xff), Line3' s Byte2(0xff), ...

The part of the broken line is non-graphic area, and then the corresponding bit is 0.



#### Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value

specification.

Example:

```
char buf[] = {0xff, 0xff, 0xe0, 0x1f, 0xff, 0xff...};  
PTK_DrawBinGraphics (20, 30, 4, 14, buf);
```

## **PTK\_DisableCircumgyrate**

### **Description:**

**This function causes the printer to disable the feed back function.**

Syntax:

```
int PTK_DisableCircumgyrate(void);
```

Parameters: none.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_DisableCircumgyrate( );
```

## **PTK\_EnableCircumgyrate**

### **Description:**

**This function causes the printer to set the feed back function.**

Syntax:

```
int PTK_EnableCircumgyrate (unsigned int distance);
```

Parameters:

distance: Circumgyrate distance, in dots.

Returns:

0-> OK;



Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_EnableCircumgyrate (140);
```

## **PTK\_DrawLineXor**

### **Description:**

**This function causes the p rinter to draw straight-line (If two straight-lines intersects, use an exclusive or operation).**

Syntax:

```
int PTK_DrawLineXor (unsigned int  px, unsigned  int  py,  
                    unsigned int  pbyte, unsigned int  pH);
```

Parameters:

px: X coordinate in dots.  
py: Y coordinate in dots.  
pbyte: Horizontal length in dots.  
pH: Vertical length in dots.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_DrawLineXor (100,20,5,110);
```

## **PTK\_DrawLineOr**

### **Description:**

**This function causes the p rinter to draw straight-line (If two straight-lines intersects, use Or operation).**

Syntax:

```
int PTK_DrawLineXor (unsigned int  px, unsigned  int  py,
```

```
unsigned int  pbyte, unsigned int  pH);
```

**Parameters:**

px: X coordinate in dots.  
py: Y coordinate in dots.  
pbyte: Horizontal length in dots.  
pH: Vertical length in dots.

**Returns:**

0-> OK;  
Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

**Example:**

```
PTK_DrawLineOr (100,20,5,110);
```

## **PTK\_DrawDiagonal**

**Description:**

**This function is to draw diagonal.**

**Syntax:**

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

**Parameters:**

px: X coordinate in dots.  
py: Y coordinate in dots.  
thickness: set Thickness in dots.  
pEx: End X coordinate in dots.  
pEy: End Y coordinate in dots.

**Returns:**

0-> OK;  
Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

**Example:**

```
PTK_DrawDiagonal (50, 30, 10, 100, 80);
```

## **PTK\_DrawWhiteLine**

### **Description:**

**This function is used to draw a whit line.**

### **Syntax:**

```
int PTK_DrawWhiteLine (unsigned int  px, unsigned int  py,  
                      unsigned int  plength, unsigned int  pH);
```

### **Parameters:**

px: X coordinate in dots.  
py: Y coordinate in dots.  
plength: Horizontal length in dots.  
pH: Vertical length in dots.

### **Returns:**

0-> OK;  
Other returns: Please make reference to chapter: CDFPSK. DLL erroneous return value specification.

### **Example:**

```
PTK_DrawWhiteLine (100, 20, 5, 110);
```

## **PTK\_ClearBuffer**

### **Description:**

**This function is used to clear the contents in printer buffer. It suggest that use this function to clear the contents in printer buffer before sending a new label to the printer.**

**Please do not use this function during editing the FORM.**

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

Parameters: none.

### **Returns:**

0-> OK;  
Other returns: Please refer to chapter: CDFPSK. DLL erroneous return value specification.

### **Example:**

```
PTK_ClearBuffer ( );
```

## **PTK\_SetPrinterState**

### **Description:**

**This function is used to set the printer's working state.**

### **Syntax:**

```
int PTK_SetPrinterState (char state);
```

### **Parameters:**

#### **State:**

D: Enable direct thermal printing (without ribbon).

P: Enable continuous printout. (default)

L: After printing a label the printer will stop, requiring user input to print the next label. Input determined by: 1. By pressing the “feed” button for each label to be printed. 2. Will continue automatically after previously printed label is removed (with peeler kit installed)

C: Enable Cutting mode. (Only with cutter kit installed)

N: Enable Peeler mode. (Only with peeler kit installed)

### **Notes:**

1. The cutter and dispenser cannot be enabled at the same time.
2. Once the options are incorrectly selected, the READY light in front panel will flicker. Please refer to the trouble-shooting section to correct the errors.

### **Returns:**

0-> OK;

Other returns: Please make reference to chapter: CDFPSK. DLL erroneous return value specification.

### **Example:**

```
PTK_SetPrinterState ('D');
```

## **PTK\_PrintLabel**

### **Description:**

**This function is used to order the printer to execute the print work.**

### **Notes:**

**Please don't use this function during editing the FORM, use the PTK\_PrintLabelAuto () instead.**

Syntax:

```
int PTK_PrintLabel (unsigned int  number, unsigned int  cpnumber);
```

Parameters:

number: Number of label sets, 1—65535.

cpnumber: Number of copies per label, 1—65535.

If the cpnumber have no set value , it will default 1.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_PrintLabel (2,3);
```

## **PTK\_PrintLabelAuto**

**Description:**

**This function causes the printer to execute to printing job automatically.** It suggests that use this function when variables or counter values exist. **The printer prints the form, as soon as all variable data have been input.**

**Notes: Only use in FORM.**

Syntax:

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

Parameters:

number: Number of label, 1—65535.

cpnumber: Number of copies for per label, 1—65535.

If the cpnumber haven't been set, it will default 1.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_PrintLabelAuto (2,3);
```

## **PTK\_SetLabelHeight**

### **Description:**

**This function is used to set the label's height and the height of media gap/black line/bore hole.**

### **Syntax:**

```
int PTK_SetLabelHeight (unsigned int lheight, unsigned int gapH);
```

### **Parameters:**

lheight: label height in dots. Value range: 0-65535.

gapH: the height of media gap/black line/bore hole in dots. Value range: 0-65535.

The value of gapH is related to the position of labels.

Gap mode: By default, set gapH to gap length. The printer is in Gap mode and parameters are set with the media AutoSense.

Black Line Mode: Set gapH to B plus black line thickness in dots.

Continuous Media Mode: Set gapH to 0 (zero). The transmissive (gap) sensor will be used to detect the end of media.

### **Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

### **Example:**

```
PTK_SetLabelHeight (160, 24);
```

## **PTK\_SetLabelWidth**

### **Description:**

**This function is used to set the label's width.**

### **Syntax:**

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

### **Parameters:**

lwidth: label width in dots .

Returns:

0-> OK;

Other returns: Please make reference to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_SetLabelWidth (250);
```

## **PTK\_SetCoordinateOrigin**

**Description:**

**This function is used to set/change the coordinate origin point.**

Syntax:

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

Parameters:

px: X coordinate moved distance in dots.

Py: Y coordinate moved distance in dots.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_SetCoordinateOrigin (12,23);
```

## **PTK\_SetPrintSpeed**

**Description:**

**This function is used to set the print speed.**

Syntax:

```
int PTK_SetPrintSpeed (unsigned int px);
```

Parameters:

px: Value range :0 - 6, or 10 - 80.

p1 value	Speed	PPLB(compatible)
10	1.0 ips (25 mm/s)	0 or 1
15	1.5 ips (37 mm/s)	
20	2.0 ips (50 mm/s)	2
25	2.5 ips (63 mm/s)	
30	3.0 ips (75 mm/s)	3
35	3.5 ips (83 mm/s)	
40	4.0 ips (100 mm/s)	4
50	5.0 ips (125 mm/s)	5
60	6.0 ips (150 mm/s)	6
70	7.0 ips (175 mm/s)	
80	8.0 ips (200 mm/s)	

Returns:

0-> OK;

Other returns: Please make reference to chapter: CDFPSK. DLL erroneous return value specification.

Example:

```
PTK_SetPrintSpeed (5);
```

## **PTK\_PrintConfiguration**

### **Description:**

**This function is used to print the current printer configuration.**

Syntax:

```
int PTK_PrintConfiguration ( );
```

Parameters: none.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK. DLL erroneous return value specification.

Example:

```
PTK_PrintConfiguration ( );
```



## **PTK\_DisableErrorReport**

### **Description:**

**This function is used to cancel the error feedback.**

### **Syntax:**

```
int PTK_DisableErrorReport(void);
```

Parameters: none.

### **Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

### **Example:**

```
PTK_DisableErrorReport( );
```

## **PTK\_EnableErrorReport**

### **Description:**

**This function is used to set the error feedback.**

The printer sends its feedback to the PC through the RS232 port.

If an error occurs, the printer will send a NACK (15H), followed by the error number, to the host.

If no errors occur, the printer will echo ACK (06H) after each P command.

Error Code	Description
0x00	No Error
0x01	Object Exceeded Label Border
0x02	Bar Code Data Length Error
0x03	Insufficient Memory to Store Data
0x04	Memory Configuration Error
0x05	RS-232 Interface Error
0x06	Paper or Ribbon Empty
0x07	Duplicate Name: Form, Graphic or Soft Font

0x08	Name Not Found: Form, Graphic or Soft Font
0x09	Not in Data Entry Mode
0x0a	Print Head Up (Open)
0x0b	Pause Mode or Paused in Peel mode
0x0c	Does not fit in area specified
0x0d	Data length too long
0x0c	PDF-417 coded data too large to fit in bar code
0x0d	
0x0e	

Syntax:

```
int PTK_ EnableErrorReport (void );
```

Parameters: none.

Returns:

0-> OK;

Other returns: Please refer to chapter:CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_ EnableErrorReport ( );
```

## **PTK\_DefineVariable**

### **Description:**

**This function is used to define variable in the FORM.**

Syntax:

```
int PTK_DefineVariable (unsigned int pid, unsigned int pmax,  
char porder, LPTSTR pmsg);
```

Parameters:

pid: Variable ID number, value range: 00—99;

pmax: Maximum number of characters, value range: 1—99.

If you use KDU, the length should limited to under 16 characters.

porder: Field Justification; L-left justification, R- right justification, C-center, N-no justification.

pmsg: remind contents; Will be sent to KDU or displayed on LCD display of the printer.

Returns:

0-> OK;

Other returns: Please make reference to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_DefineVariable (0, 16, L, "\ " Enter Title: \ " ");
```

## **PTK\_DrawRectangle**

**Description:**

**This function is used to draw rectangle.**

Syntax:

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

Parameters:

px: Horizontal start position (X) in dots.

py: Vertical start position (Y) in dots.

thickness: Line thickness in dots.

pEx: Horizontal end position (X) in dots.

pEy: Vertical end position (Y) in dots.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_DrawRectangle (50, 120, 5, 250, 150);
```

## **PTK\_SetDirection**

**Description:**

**This function is used to set print orientation for all graphics, text, bar codes, lines**

and **rectangles**.

this function will change the direction of contents, such as text, barcode, straight line, rectangle.

Syntax:

```
int PTK_SetDirection (char direct);
```

Parameters:

direct: Orientation; Acceptable values are B or T. The default value is T.

B: Print from bottom right corner.

T: Print from top left corner.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_SetDirection ( 'B' );
```

## **PTK\_EnableFLASH**

**Description:**

**This function is used to enable Flash memory.**

**the data sent to the printer will be stored to the flash memory when use this function.**

Syntax:

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

Parameters: none

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_EnableFLASH ( );
```

## **PTK\_DisableFLASH**

**This function is used to disable Flash memory.**

**The data sent to the printer will be stored to the SDRAM when use this function.**

Syntax:

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

Parameters: none

Returns:

0-> OK;

Other returns: Please make reference to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_DisableFLASH ( );
```

## **PTK\_Download**

Description:

**This function is used to download variable or counter variable to the printer.**

**Please refer to the command “?” of the PPL I.**

Syntax:

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

Parameters: none

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_Download( );
```

## **PTK\_Reset**

### **Description:**

This function is used to reset the printer.

**This function is equal to the power off, then power on, thus reinitializing the printer.**

The reset function is not available during the download of PCX graphics, soft fonts or while the printer is in dump mode.

The reset function cannot be used within a stored form.

The reset function can be sent to the printer during all other printing operations.

The printer will ignore all commands sent while the reset command is executing, up to 2 seconds.

### **Syntax:**

```
int PTK_Reset(void);
```

Parameters: none

### **Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

### **Example:**

```
PTK_Reset( );
```

## **PTK\_BackFeed**

### **Description:**

This function is used to get printer error and status reports immediately.

Users can get the current error state by this function. The printer will report 4 bytes back to the host by the following format:

0x0d 0x0a	: <CR><LF>
0xXX XX	: Error/Status code

### **Syntax:**

```
int PTK_BackFeed (void);
```

Parameters: none

### **Returns:**

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_BackFeed ( );
```

## **PTK\_CutPage**

Description:

This function is used to **set cutter's working circle** (namely the cutter will start to cut labels once how many labels have been printed)

Syntax:

```
int PTK_CutPage (UINT page);
```

Parameters:

page: pages, value range: 1-999, the default value is 1.

Returns:

0-> OK;

Other returns: Please refer to chapter: CDFPSK.DLL erroneous return value specification.

Example:

```
PTK_CutPage(1);
```

**WINPSK.dll    Erroneous return value specification**

- 3001 :    PTK\_GetInfo    execution error;
- 3002 :    PTK\_DrawText    execution error;
- 3003 :    PTK\_DrawText    parameter error;
- 3004 :    PTK\_DrawText or PTK\_DrawBarcode    pdirec parameter error;
- 3005 :    PTK\_DrawText    pFont parameter error;
- 3006 :    PTK\_DrawText    pHorizontal or pVertical parameter error;
- 3007 :    NULL;
- 3008 :    PTK\_DrawBarcode    execution error;
- 3009 :    PTK\_DrawBarcode    parameter error;
- 3010 :    PTK\_DefineCounter    execution error;
- 3011 :    PTK\_DefineCounter    parameter error;
- 3012 :    PTK\_SetDarkness    execution error;
- 3013 :    PTK\_DS    execution error;
- 3014 :    PTK\_SoftFontList    execution error;
- 3015 :    PTK\_SoftFontDel    parameter error;
- 3016 :    PTK\_SoftFontDel    execution error;
- 3017 :    PTK\_FormEnd    execution error;
- 3018 :    PTK\_FormList    execution error;
- 3019 :    PTK\_FormDel    distribution memory error;
- 3020 :    PTK\_FormDel    parameter error;
- 3021 :    PTK\_FormDel    execution error;
- 3022 :    PTK\_ExecForm    distribution memory error;
- 3023 :    PTK\_ExecForm    parameter error;
- 3024 :    PTK\_ExecForm    execution error;
- 3025 :    PTK\_FormDownload    distribution memory error;
- 3026 :    PTK\_FormDownload    parameter error;
- 3027 :    PTK\_FormDownload    execution error;
- 3028 :    PTK\_DrawPcxGraphics    distribution memory error;
- 3029 :    PTK\_DrawPcxGraphics    parameter error;
- 3030 :    PTK\_DrawPcxGraphics    execution error;
- 3031 :    PTK\_PcxGraphicsList    execution error;
- 3032 :    PTK\_PcxGraphicsDel    distribution memory error;
- 3033 :    PTK\_PcxGraphicsDel    parameter error;
- 3034 :    PTK\_PcxGraphicsDel    execution error;
- 3035 :    PTK\_PcxGraphicsDownload    distribution memory error;
- 3036 :    PTK\_PcxGraphicsDownload    open file error;
- 3037 :    PTK\_PcxGraphicsDownload    execution error;
- 3038 :    PTK\_DrawBinGraphics    execution error;
- 3039 :    (ditto);
- 3040 :    NULL;



-3041 : PTK\_DisableCircumgyrate execution error;  
-3042 : PTK\_EnableCircumgyrate execution error;  
-3043 : PTK\_DrawLineXor execution error;  
-3044 : PTK\_DrawLineOr execution error;  
-3045 : PTK\_DrawDiagonal execution error;  
-3046 : PTK\_DrawWhiteLine execution error;  
-3047 : PTK\_ClearBuffer execution error;  
-3048 : PTK\_SetPrinterState execution error;  
-3049 : PTK\_SetPrinterState parameter error;  
-3050 : PTK\_PrintLabel execution error;  
-3051 : PTK\_PrintLabel parameter error;  
-3052 : PTK\_PrintLabelAuto execution error;  
-3053 : PTK\_PrintLabelAuto parameter error;  
-3054 : PTK\_SetLabelHeight execution error;  
-3055 : PTK\_SetLabelWidth execution error;  
-3056 : PTK\_SetCoordinateOrigin execution error;  
-3057 : PTK\_SetPrintSpeed execution error;  
-3058 : PTK\_SetPrintSpeed parameter error;  
-3059 : PTK\_PrintConfiguration execution error;  
-3060 : PTK\_DisableErrorReport execution error;  
-3061 : PTK\_EnableErrorReport execution error;  
-3062 : PTK\_DefineVariable execution error;  
-3063 : PTK\_DefineVariable parameter error;  
-3064 : PTK\_DefineVariable parameter error;  
-3065 : PTK\_DrawRectangle execution error;  
-3066 : PTK\_Y execution error;  
-3067 : PTK\_Y parameter error;  
-3068 : PTK\_SetDirection execution error;  
-3069 : PTK\_SetDirection parameter error;  
-3070 : PTK\_EnableFLASH execution error;  
-3071 : PTK\_DisableFLASH execution error;  
-3072 : PTK\_Download execution error;  
-3073 : PTK\_Reset execution error;  
-3074 : PTK\_BackFeed execution error;  
-3075 : distribution save PCX HEAD file framework memory error;  
-3076 : distribution save PCX data memory error;  
-3077 : distribution save contemporary file path memory error;  
-3078 : build PCX file error;  
-3079 : save PCX data error;  
-3080 : PTK\_PrintPCX execution error;  
-3081 : build PrinterDC fail, deal with the error;  
-3082 : distribution save bitmap memory error;  
  
-3100 : SetCommPort change the serial port error;

-3101 :     PTK\_CutPage execution error;

-3200

to

-3400 :     the port have not opened or have closed.