Introduction

zyPrtLib Micro-printer Driver Libaray

Micro-printer Serial Product

Rev 1.00 Date: 2011/05/04 **Product Data Sheet**

Document Information

TYPE	CONTENT
Key words	zyPrtLib, driver library
Abstract	Guangzhou ZLGMCU Technology Co., Ltd. developed several types of Micro-Thermal Printer. They are fully functional, and can support more than thirty common ESC/POS instructions, enabling customers to complete their product development in a short time and make their products more competitive in the market. This document provides the zyPrtLib driver library which encapsulates the ECS/POS commands for application development.

Revision History

Version	Rev. Date	Modifications
V1.00	2011-05-04	Original version

Sales Information

Guangzhou ZLGMCU Technology Co., Ltd.

Address: F1 Room, 15 Floor, Everbright BANK Building, 689 Tianhe Northern Road, Guangzhou, CHINA
TEL: +86-20-38730916, 38730917, 38730972, 38730976, 38730977
FAX: +86-20-38730925
Website: www.zlgmcu.com

Guangzhou Sales Office
Address: Room 203 & 204, XinSaiGE Electronic Building, Tianhe District, Guangzhou, CHINA
TEL: +86-20-87578634, 87569917
FAX: +86-20-87578842

Nanjing Sales Office
Address: Room 2006, Zhujiang Building, 280 Zhujiang Road, Nanjing, CHINA
TEL: +86-25-68123901, 68123902
FAX: +86-25-68123900

Beijing Sales Office
Address: Room 712, Yingwang Centre, 113 Zhichun Road, Haiding District, Beijing, CHINA
TEL: +86-10-62536178, 62536179 82628073
FAX: +86-10-82614433

Chongqing Sales Office
Address: Room 1661, Saige electronics market, Daxiyang International Building, 2 Keyuanyi Road, Shiqiao, Chongqing, CHINA
TEL: +86-23-68796438, 68796439
FAX: +86-23-68796439

Hangzhou Sales Office
Address: Room 502, Jiangnan Electronics Building, 217 Tianmushan Road, Hangzhou, CHINA
TEL: +86-571-28139611, 28139612, 28139613, +86-571-28139615, 28139616, 28139618
FAX: +86-571-88009204

Chengdu Sales Office
Address: Room 401, Shumatongrengang Building, 1 Scoend Southern Yihuan Road, Chengdu, CHINA
TEL: +86-28-85439836, 85437446
FAX: +86-28-85437896

Shenzhen Sales Office
Address: Room D, Floor 4, C Side, Dianzikeji Building, 2070 ShenNanZhong Road, Shenzhen, CHINA
TEL: +86-755-83781788 (line 5)
FAX: +86-755-83793285

Wuhan Sales Office
Address: Room 12128, Huazhong Computer and electronics market, 158 LuoYu Road, GuangFouTun, HongShan District, Wuhan, CHINA
TEL: +86-27-87168497, 87168297, 87168397
FAX: +86-27-87163755

Shanghai Sales Office
Address: Room 7E, Eastern side, Kejijingcheng Building, 668 Beijingdong Road, Shanghai, CHINA
TEL: +86-21-53083452, 53083453, 53083496
FAX: +86-21-53083491

XiAn Sales Office
Address: Room 1201, Pacific Building, 54 Changanbei Road, XiAn, CHINA
TEL: +86-29-87881296, 83063000, 87881295
FAX: +86-29-87880865

Technical Supports

Guangzhou ZHIYUAN Electronics Co., Ltd.

Address: Floor 2, Building No.3 Huangzhou Industrial Estate, Chebei Road, Tianhe District, Guangzhou, CHINA, Post code: 510660

TEL: +86-20-22644249, 28872524, 22644399, 28872342, 28872349, 28872569, 28872573

FAX: +86-20 38601859

Website: www.embedtools.com www.embedcontrol.com www.ecardsys.com

Technical Supports for Intelligent Serial Display Module

TEL: +86-20-28267818

E-mail: ztm@zlgmcu.com

Sales Contact

TEL: +86-20-22644249, 22644399, 22644372, 22644261, 28872524,
+86-20-28872342, 28872349, 28872569, 28872573, 38601786

Repair and rework

TEL: +86-20-22644245

Content

Chapter 1: Scope of Application

zyPrtLib micro-printer driver library is available to any Micro-thermal printer series products developed by Guangzhou ZLGMCU Technology Co., Ltd. It allows user to program different printing functions easily without knowing the complex knowledge on ESC/POS protocols, enabling users to complete their product development in a short time and make their products more competitive in the market.

There are 5 versions of the zyPrtLib based on different operation systems:

- Windows XP (VC++) version;
- Windows CE (Embedded VC++) version;
- Linux version;
- Cortex-M0(C) version;
- C51(C) version.

This document introduces the C51(C) version zyPrtLib micro-printer driver library.

Chapter 2: Product Specifications

zyPrtLib micro-printer driver library contains six files located in the zyPrtLib directory under the project. Figure 2-1 shows the software architecture: application layer, hardware base layer and printer state message handler interface:

- Application layer implements the printing function implementations. It is a user-oriented programming interface;
- Hardware base layer is not user-oriented, it contains the driver program related to the hardware directly, such as UART data transceiver, delay function and so on;
- Printer state message handler interface is a user-oriented programming interface. Since it is optional, user should configure it before use. If printer returns state message, it will perform the presetting processing code.

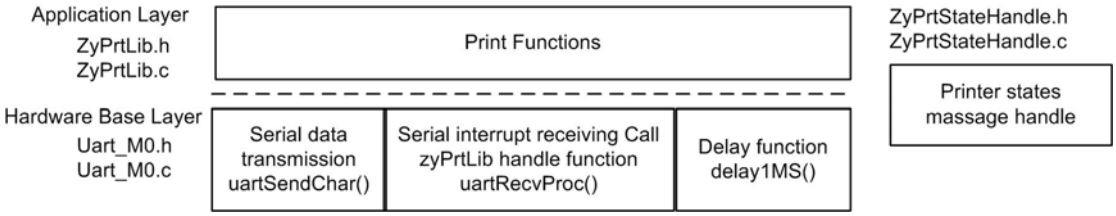


Figure 2-1: Software Architecture

Chapter 3: Driver Library Portability

The porting of zyPrtLib is very convenient. It can be done by just modifying the following configurations of hardware interface and base layer driver.

- Serial port initial function;
- Serial port transmission function;
- Serial receiving interrupt entry function;
- Delay function;
- Flow control interface definition.

The functions above can be found in the file Uart_C51.c and file Uart_C51.h.

An example of porting zyPrtLib to NXP P89V51 51 MCU is given in the following sections.

3.1 Porting serial port initial function

The transportation of serial port initial function to P89V51 is shown as Program list 3-1:

Program list 3-1: Porting of serial port initial function

```
/*******************************************************************************************
** Function name:           uartInit
** Descriptions:            Initialize the serial port
** input parameters:        ulBaud：Baud rate
** Returned value:          none
*******************************************************************************************/
void uartInit (unsigned long ulBaud)
{
    /*
     * The followings are customizable.
     */
unsigned char Tmp;
    TMOD = 0x20;                                    /* Set T1 to mode 2                     */
    PCON = 0x00;                                    /* Set baud rate to no frequency multiplying   */
    SCON = 0x50;                                    /* Data length is 8 bits, and T1 controls baud
                                                       rate                                 */
    Tmp  = 256 – (unsigned long )11059200/12/32/ulBaud;
    TH1  = Tmp;                                     /* Baud rate setting, use a 11.0592MHz crystal
                                                       oscillator.                          */
    TL1  = Tmp;
    EA   = 1;
    ES   = 1;
```

```
    TR1    = 1;
}
```

Notes: *When the main operating frequency of the system isn't 11.0592MHz, the program above should be modified.*

3.2 Porting serial port transmission function

The transportation of serial port transmission function to P89V51 is given as Program list 3-2:

Program list 3-2: Porting of serial port transmission function

```
/*******************************************************************************************
** Function name:           uartSendChar
** Descriptions:            Serial port transmission function
** input parameters:        ucData: the transmitted bytes
** Returned value:          none
*******************************************************************************************/
void uartSendChar (unsigned char ucData)
{
    /*
     * The followings are customizable.
     */
    SBUF = ucData;
    while (!TI);
    TI = 0;
}
```

Notes: *TI can only be cleared by software.*

3.3 Porting serial receiving interrupt entry function

The transportation of serial port transmission function to P89V51 is given as Program list 3-3:

Program list 3-3: Porting of serial receiving interrupt entry function

```
/*******************************************************************************************
**Function name:           uartRecvISR
** Descriptions:            Serial receiving interrupt entry function
** input parameters:        none
** Returned value:          none
*******************************************************************************************/
void uartRecvISR (void) interrupt 4 using 1
{
    /*
     * The followings are customizable.
     */
```

```
    if (RI) {
        RI = 0;
        uartRecvProc(SBUF);                    /* Call the receiving processing function
                                                  of zyPrtLib                          */
    }
}
```

Notes: *RI can only be cleared by software.*

3.4 Porting delay function

The transportation of delay function to P89V51 is shown as Program list 3-4:

Program list 3-4: Porting of delay function

```
/*******************************************************************************************
** Function name:          delay1MS
** Descriptions:           1MS delay function
** input parameters:       none
** Returned value:         none
*******************************************************************************************/
void delay1MS (void)
{
    /*
     * The followings are customizable.
     */
    unsigned char i;

    for (i = 0; i < 114; i++);                 /* Use a 11.0592MHz crystal oscillator. */
}
```

3.5 Porting flow control interface definition

The transportation of flow control interface definition is listed as Program list 3-5:

Program list 3-5: Porting of flow control interface definition

```
/*******************************************************************************************
** Hardware flow control BUSY pin definition. It is user-defined.
*******************************************************************************************/
sbit busypin = P1^3;                           /* User code                            */

#define BUSY_PIN                               /* It is not required on driver library template 51*/
#define BUSY_PIN_DIR                           /* It is not required on driver library template 51*/
#define BUSY_PIN_DATA                          /* It is not required on driver library template 51*/
#define BUSY_PIN_IN()                          /* It is not required on driver library template 51*/
#define BUSY_PIN_STATE()        (busypin)      /* Define the BUSY pin in driver library template
                                                  to P1_3.                             */
```

Chapter 4: Driver Library Function

4.1 Function list

The zyPrtLib driver library functions are listed in Table 4-1.

Table 4-1: Driver library function list

Function	Explanation	See
uartSendStr	Transmit the text	Table 6-1
Functions for print and feed paper		
print	Print and feed paper	Table 6-2
printAndFeedNDotLine	Print and feed paper for n dot	Table 6-3
printAndBackFeedNDotLine	Print and back feed paper for n dot	Table 6-4
printAndFeedNFontLine	Print and feed paper for n line	Table 6-5
printAndBackFeedNFontLine	Print and back feed paper for n line	Table 6-6
Functions for printing items settings		
lineSpacingSet	Set the line space to n dot	Table 6-7
lineSpacingDefault	Set the line space to a default value	Table 6-8
leftMarginSet	Set the left margin	Table 6-9
rightMarginSet	Set the right margin	Table 6-10
abscissaSet	Set the absolute print position	Table 6-11
fontTypeSet	Set the font types	Table 6-12
horizontalAlign	Set the print alignment	Table 6-13
fontGrayscaleSet	Set the print grayscale	Table 6-14
printSpeedSet	Set the print speed	Table 6-15
fontSizeSet	Set the font size	Table 6-16
selectKanjiMode	Select Kanji character mode	Table 6-17
cancelKanjiMode	Cancel Kanji character mode	Table 6-18
selectInternationalChar	Select an international character set	Table 6-19
selectCharCodePage	Select character code page	Table 6-20
Functions for bit-image printing		
picturePrintV	Select bit-image mode	Table 6-21
picturePrintH	Print raster bit image	Table 6-22
Functions for table printing		
horizontalTabSet	Set horizontal tab positions	Table 6-23
horizontalTab	Horizontal tab	Table 6-24
formPrintV	Print the vertical table	Table 6-25
Functions for one dimension bar code setting/printing		
barcodeHeightSet	Set the height of one-dimension bar code	Table 6-26

Function	Explanation	See
barcodeWidthSet	Set the width of one-dimension bar code	Table 6-27
barcodeHriPosSet	Select print position of one-dimension HRI	Table 6-28
barcodeHriFontSet	Select font for one-dimension bar code	Table 6-29
barcodePrint	Print one-dimension bar code	Table 6-30
Miscellaneous		
printerInit	Initialize the printer	Table 6-31
printerClearBuffer	Clear the printer buffer (real-time)	Table 6-32
printerCutPaper	Feed paper and cut paper	Table 6-33
printerStateRT	Query the states of printer (real-time)	Table 6-34
printerStateAutoUpload	Set/cancel the printer states automatic back	Table 6-35
printerBaudRateSet	Set printer baud rate	Table 6-36
printerFlowCtrlSet	Set flow control mode	Table 6-37
printerEnterLowPowerMode	Enter low power mode	Table 6-38
printerExitLowPowerMode	Exit low power mode	Table 6-39
Functions for message handler		
printerOverHeatHandler	Printer over temperature handler entry	Table 6-40
printerOverHeatResumeHandler	Printer over temperature resume handler entry	Table 6-41
printerOfPaperHandler	Printer paper end handler entry	Table 6-42
printerOfPaperResumeHandler	Printer paper end resume handler entry	Table 6-43
printerOverVoltageHandler	Printer over voltage handler entry	Table 6-44
printerOverVoltageResumeHandler	Printer over voltage resume handler entry	Table 6-45
printerAxoOpenHandler	Printer platen open handler entry	Table 6-46
printerAxoCloseHandler	Printer platen close handler entry	Table 6-47
printerCutterOnHandler	Pinter cutter down handler entry	Table 6-48
printerCutterOffHandler	Pinter cutter resume handler entry	Table 6-49

Chapter 5: How to Use

5.1 Introduction

In this section, we are going to give an example of CRP report list printing to explain how to use the zyPrtLib driver library, as Figure 5-1 shows.

Figure 5-1: CRP report list printing

5.2 Programming steps

The program flow is shown in Figure 5-2. And the example code is listed in the following sections.

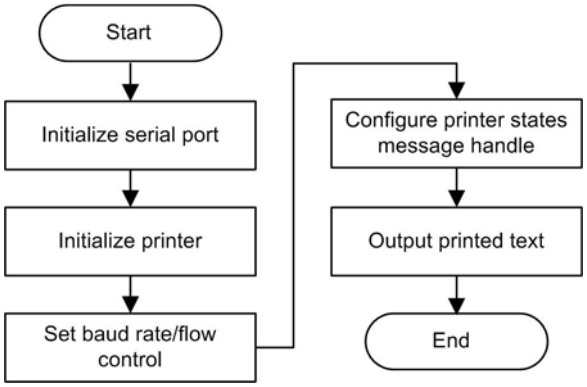


Figure 5-2: Program flow

1. Initialize the serial port

At first, user should initialize the serial port of the main control MCU. Notice that the communication baud rate should be complied with the default settings on power up.

For detailed information about the default settings of the communication baud rate, please refer to the relevant data sheet of the micro-printer. In additional, function printerBaudRateSet (see Table 6-36) can be applied to modify the default communication baud rate of the printer if required. Program list 5-1 gives the related example code.

Program list 5-1: Serial port initialization

```
int main (void)
{
      .....
      uartInit(9600);                          /* Initialize the serial port, assuming that the
                                                  default communication baud rate is 9600MB/s*/
  pinterBaudRateSet(115200);                   /* Send a command to modify the default baud
                                                  rate of the printer to 115200MB/s
            */
      uartInit(115200);                        /* Modify the baud rate to 115200MB/s      */
      ......
}
```

2. Initialize the printer

And then, call the function printerInit() to reset the printing parameters of the printer. Initialization to all the parameters will be performed as soon as printer is power on, except the communication baud rate. Program list 5-2 shows the related example code.

Program list 5-2: Printer initialization

```
int main (void)
{
      .....
      printerInit();                           /* Reset printer parameters                */
      .....
}
```

3. Set the flow control

For large data printing, the data sent to the printer may be lost during the printing because of data overflow. In this case, flow control is required. zyPrtLib supports hardware flow control and software flow control. After initialed the printer, call the function printerFlowCtrlSet (see Table 6-37) to set flow control mode. Program list 5-3 explains how to set the software flow control.

Program list 5-3: Flow control mode setting

```
int main (void)
{
```

```
    ......
    printerFlowCtrlSet(FLOW_CTRL_SOFTWARE);
    ......
}
```

4. Add exception handler code

In the process of printing, some exceptions may occur on the printer, such as overheat or paper end. Printer will call exception handler entry functions to deal with them, when returning related state message. User can program the exception handler entry functions in the file ZyPrtStateHandler.c as required. For more information about exception handler entry functions, please refer to Table 6-40~Table 6-49.

（1）Program the printer state back

User can use function printerStateRT (see Table 6-34) to query the printer state.

And function printerStateAutoUploa (see Table 6-35) can be applied to set the printer states automatic back, since this function is disabled by default. Program list 5-4 gives the example code.

Program list 5-4: Printer states automatic back settings

```
int main (void)
{
    ……
    printerStateAutoUpload(PRINTER_STATE_OFPAPER);   /* Paper end state automatic back should be set
                                                        after the initialization of printer          */
    ......
}
```

（2）Add exception handler code

Program list 5-5 takes the paper end LED indication as an example.

Program list 5-5: Exception handler code adding

```
void printerOfPaperHandler (void)
{
    /*
     * The followings are customizable.
     */
     LedOn();                                    /* Turn on the LED for paper end indication   */
}
......
void printerOfPaperResumeHandler (void)
{
    /*
     * The followings are customizable.
     */
     LedOff();                                   /* Turn off the LED for paper end indication  */
}
```

Notes: *In the exception handler entry function, the blocking code can not be executed.*

5. Output printing text

Now, it is ready to print the CRP report. Program list 5-6 shows the complete code for printing, including the code shown in 1~4 above.

Program list 5-6: Printing the CRP report

```
/*******************************************************************************************
** Function name:          main
** Descriptions:           Print CRP report
** input parameters:       none
** output parameters:      none
** Returned value:         none
*******************************************************************************************/
int    main (void)
{
    /*
     * Set horizontal tab positions settings, unit: 1mm (8 dots), valid value: 0~47
     */
     unsigned char ucTab1[] = {24};                    /* Store the settings of the horizontal tab 1 */
     unsigned char ucTab2[] = {20, 36};                /* Store the settings of the horizontal tab 2 */

     /*
     * Initialization
     */
    uartInit(9600);                                    /* Initialize the serial port, assuming that
                                                          the default baud rate is 9600MB/s      */

     printerInit();                                    /* Initialize the printer                 */
pinterBaudRateSet(115200);                             /* Send a command to modify the default
                                                          baud rate of the printer to 115200MB/s*/
     uartInit(115200);                                 /* Modify the baud rate to 115200MB/s  */
     printerFlowCtrlSet(FLOW_CTRL_SOFTWARE);           /* Set software flow control           */
     printerStateAutoUpload(PRINTER_STATE_OFPAPER);    /* Set printer states automatic back   */

     /*
     * The following is the content of the CRP report.
     */

     /*
      * Print “Printing Example”
      */
    horizontalAlign (ALIGN_CENTER);                    /* Set the print alignment to center      */
    fontTypeSet(FONT_TYPE_DOUBLE_HEIGHT | FONT_TYPE_DOUBLE_WIDTH);
                                                       /* Set the font type to double height and
```

```
                                                    double width.                   */
uartSendStr("Printing Example");
printAndFeedNDotLine(60);                           /* Print and feed paper for 60 lines
 */

 horizontalAlign (ALIGN_LEFT);                      /* Set the print alignment to left  */
/*
 * Print “CRP report    Sample Number: 50”
 */
abscissaSet (72);                                   /* Set the absolute print position  */
fontTypeSet(FONT_TYPE_NORMAL);                      /* Set font type to normal          */
uartSendStr("CRP report    Sample Number: 50");
print();                                            /* Print and feed one line          */

/*
 * Print “Name:                Gender:”
 */
horizontalTabSet(ucTab1, 1);                        /* Set horizontal tab positions     */
uartSendStr("Name:");
horizontalTab();                                    /* Shift to next tab position       */
uartSendStr("Gender:");
print();                                            /* Print and feed one line          */

/*
 * Print “Age:             Department: Emergency Department”
 */
uartSendStr("Age:       ");
horizontalTab();                                    /* Shift to next tab position       */

uartSendStr("Department: Emergency Department ");
print();                                            /* Print and feed one line          */

/*
 * Print “Doctor:              Bed: 909”
 */
uartSendStr("Doctor:");
horizontalTab();                                    /* Shift to next tab position       */
uartSendStr("Bed: 909");
print();                                            /* Print and feed one line          /*
 * Print “Outpatient server number: 0111      Specimens type: blood serum”
 */
uartSendStr("Outpatient server number: 0111");
horizontalTab();                                    /* Shift to next tab position       */
uartSendStr("Specimens type: blood serum ");
```

```
print();                                        /* Print and feed one line          */

/*
 * Print “Inspector : John       Checker: John”
 */
uartSendStr("Inspector : John ");
horizontalTab();                                /* Shift to next tab position       */
uartSendStr("Checker: John ");
print();                                        /* Print and feed one line          */

/*
 * Print “Inspection date: 2002-11-15”
 */
uartSendStr("Inspection date: 2002-11-15");
print();                                        /* Print and feed one line          */

/*
 * Print “Check date: 2009-11-19”
 */
uartSendStr("Check date: 2009-11-19");
print();                                        /* Print and feed one line          */

/*
 * Print “----------------”
 */
uartSendStr("----------------");
print();                                        /* Print and feed one line          */

/*
 * Print “Inspection item   Result   Unit   Medical reference range”
 */
fontTypeSet(FONT_TYPE_EMPHASIZE);               /* Set the font to bold             */
horizontalTabSet(ucTab2, 2);                    /* Set the absolute print position  */
uartSendStr("Inspection item ");
horizontalTab();                                /* Shift to next tab position       */
uartSendStr("Result");
horizontalTab();                                /* Shift to next tab position       */
uartSendStr("Unit");
print();                                        /* Print and feed one line          */

/*
 * Print “CRPT   1.59   ml/L   3-200”
 */
fontTypeSet(FONT_TYPE_NORMAL);                  /* Set the font to normal           */
```

```
    uartSendStr("CRPT ");
    horizontalTab();                          /* Shift to next tab position      */
    uartSendStr("1.59");
    horizontalTab();                          /* Shift to next tab position      */
    uartSendStr("ml/L");
    print();                                  /* Print and feed one line         */

    /*
    *Printing end
    */
    while(1);
}
```

Chapter 6: Functions Description

The functions of zyPrtLib driver library are listed in Table 6-1 toTable 6-49.

Notes: *For the functions in zyPrtLib.h, conditional compile can be applied to remove any unwanted code from the target code. So that the size of the target code can keep small.*

Table 6-1: uartSendStr

Function Name	Transmit the text
Function Prototype	void uartSendStr (char *pucStr)
Parameter	pucStr: Character string (end with '\0')
Return Value	None
Model Supported	All the models
Description	It is used for text message transmission. If the buffer within the printer is full, the image data already stored in the print buffer is printed, and a line feed is executed.

Table 6-2: print

Function Name	Print and feed paper
Function Prototype	void print (void)
Parameter	None
Return Value	None
Model Supported	All the models
Description	This function prints all the content in the printer buffer, and feeds one line based on the current line spacing. After printing, the print position moves to the beginning of the next line.

Table 6-3: printAndFeedNDotLine

Function Name	Print and feed paper for n dots
Function Prototype	void printAndFeedNDotLine (unsigned char ucDotLineNum)
Parameter	ucDotLineNum: 0~255 dots
Return Value	None
Model Supported	All the models
Description	This function prints all the content in the printer buffer and feeds paper for n dots. However, when the printer buffer is empty, it only feeds paper for n lines but not print. After printing, the print position moves to the beginning of the next line.

Table 6-4: printAndBackFeedNDotLine

Function Name	Print and back feed paper for n dots
Function Prototype	void printAndBackFeedNDotLine (unsigned char ucDotLineNum)
Parameter	ucDotLineNum: 0~255 dots
Return Value	None
Model Supported	All the models
Description	This function prints all the content in the printer buffer and feeds paper back for n dots. However, when the printer buffer is empty, it only feeds paper back for n dots but not print. After printing, the print position moves to the beginning of the next line.

Table 6-5: printAndFeedNFontLine

Function Name	Print and feed paper for n lines
Function Prototype	void printAndFeedNFontLine (unsigned char ucFontLineNum)
Parameter	ucFontLineNum: 0~255 lines
Return Value	None
Model Supported	All the models
Description	This function prints all the content in the printer buffer and feeds paper for n lines. However, when the printer buffer is empty, it only feeds paper for n lines but not print. The line space is set by lineSpacingSet or lineSpacingDefault. After printing, the print position moves to the beginning of the next line.

Table 6-6: printAndBackFeedNFontLine

Function Name	Print and feed paper back for n lines
Function Prototype	void printAndBackFeedNFontLine (unsigned char ucFontLineNum)
Parameter	ucFontLineNum: 0~255 lines
Return Value	None
Model Supported	All the models
Description	This function prints all the content in the printer buffer and feeds paper back for n lines. However, when the printer buffer is empty, it only feeds paper back for n lines but not print. The line space is set by lineSpacingSet or lineSpacingDefault. After printing, the print position moves to the beginning of the line.

Table 6-7: lineSpacingSet

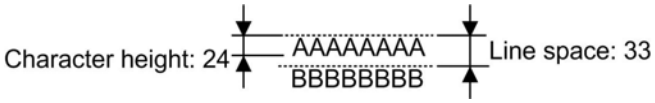
Function Name	Set the line space to n dots
Function Prototype	void lineSpacingSet (unsigned char ucDotLineNum)
Parameter	ucDotLineNum: 0~255 dots
Return Value	None
Model Supported	All the models
Description	This function sets the line space to n dots, as follows:  If the maximum character height exceeds the specified line space in a line, the line spacing will be automatically set to that maximum height. If lineSpacingDefault() is executed, printerInit() is executed, printer is reset or printer is turned off, the line space will be reset to a default value 33 dots.

Table 6-8: lineSpacingDefault

Function Name	Set the line space to a default value 33 dots
Function Prototype	void lineSpacingDefault (void)
Parameter	None
Return Value	None
Model Supported	All the models
Description	This function sets the line space to 33 dots. For more details in line space settings, please refer to lineSpacingSet(). If the maximum character height exceeds the specified line space in a line, the line spacing will be automatically set to that maximum height. The line space can be set by lineSpacingSet().

Table 6-9: leftMarginSet

Function Name	Set the left margin
Function Prototype	void leftMarginSet (unsigned char ucLeftMargin)
Parameter	ucLeftMargin: the left margin value (Unit: 8 dots) For ZYTP58 and MTP58: 0≤ ucLeftMargin ≤47, and 0≤(left margin + right margin)≤47 For ZYTP80 and MTP80: 0≤ ucLeftMargin ≤71, and 0≤(left margin + right margin)≤71
Return Value	None
Model Supported	All the models
Description	This function sets the left margin (unit: 8 dots) to make sure the content printed not exceed the left margin position. The left margin position is the left edge position of the printing range. Following is an example of left margin setting.

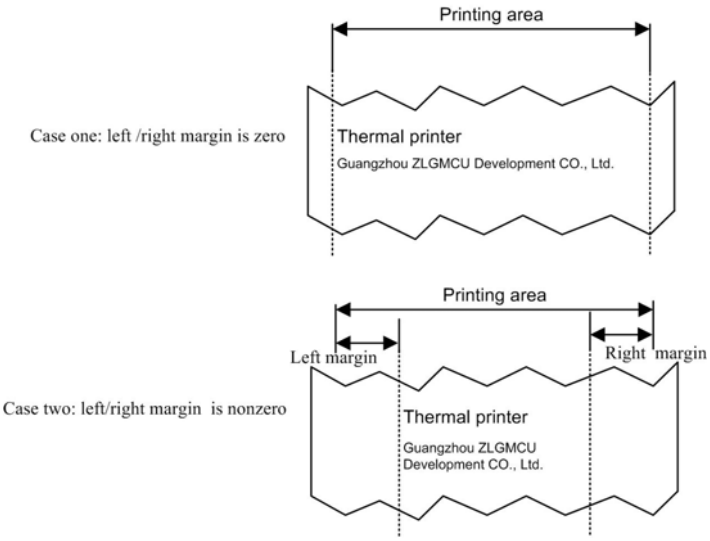
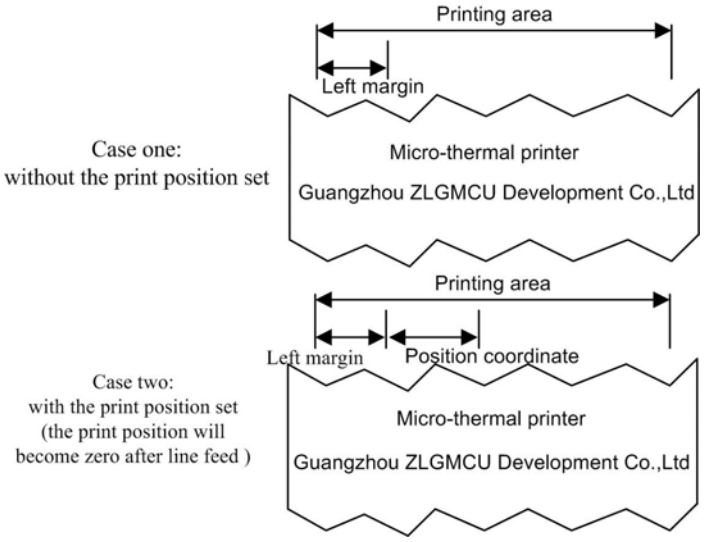
	 If lineSpacingDefault() is executed, printerInit() is executed, printer is reset or printer is turned off, the left margin setting will be reset to 0.

Table 6-10: rightMarginSet

Function Name	Set the right margin
Function Prototype	void rightMarginSet (unsigned char ucRightMargin)
Parameter	ucRightMargin: the right margin value (Unit: 8 dots) For ZYTP58 and MTP58: 0≤ ucRightMargin ≤47, and 0≤(left margin + right margin)≤47 For ZYTP80 and MTP80: 0≤ ucRightMargin ≤71, and 0≤(left margin + right margin)≤71
Return Value	None
Model Supported	All the models
Description	This function sets the right margin (unit: 8 dots) to make sure the content printed not exceed the right margin position. The right margin position is the right edge position of the printing range. For more details in margin setting, please refer to leftMarginSet(). If lineSpacingDefault() is executed, printerInit() is executed, printer is reset or printer is turned off, the right margin setting will be reset to 0.

Table 6-11: abscissaSet

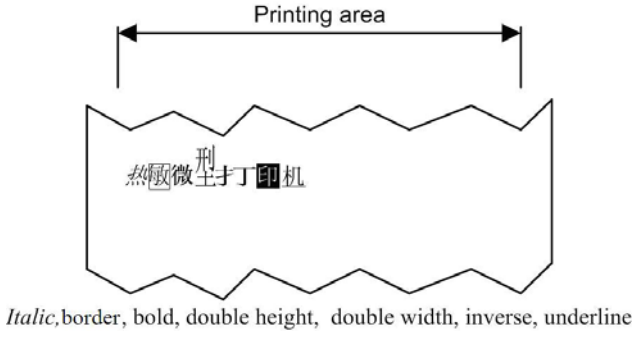
Function Name	Set the absolute print position
Function Prototype	void abscissaSet (unsigned short ucXNum)
Parameter	ucXNum: absolute print position, 0~65535 dots
Return Value	None
Model Supported	All the models
Description	This function shifts the print position to a location in a distance of (ucXNum) dots from the starting position for printing, as following shows:



When a left margin is set, the position of the left margin is the beginning of the line.

This function only affects one line. The print position is the starting position again after line feeding.

Table 6-12: fontTypeSet

Function Name	Set the font types
Function Prototype	void fontTypeSet (unsigned char ucFontType)
Parameter	ucFontType: font types with combination applicable (use “OR”) FONT_TYPE_NORMAL normal FONT_TYPE_ITALIC italic FONT_TYPE_BORDER border FONT_TYPE_EMPHASIZE bold FONT_TYPE_DOUBLE_HEIGHT double height FONT_TYPE_DOUBLE_WIDTH double width FONT_TYPE_REVERSE_VEDIO inverse FONT_TYPE_UNDERLINE underline
Return Value	None
Model Supported	All the models
Description	This function sets the font types, including italic, border, bold, double width, double height, inverse or underline, as following shows. It is valid for both Chinese font and English font.  *Italic*,border, bold, double height, double width, inverse, underline

	All the font types can be used in combination. The font types settings are effective until printerInit() is executed, printer is reset or printer is turned off

Table 6-13: horizontalAlign

Function Name	Set the print alignment
Function Prototype	void horizontalAlign(unsigned char align)
Parameter	Align: print alignment for printing ALIGN_LEFT left ALIGN_CENTER center ALIGN_RIGHT right
Return Value	None
Model Supported	All the models
Description	This function aligns all the content in a line. The settings by horizontalAlign() are effective until printerInit() is executed, printer is reset or printer is turned off.

Table 6-14: fontGrayscaleSet

Function Name	Set the font grayscale
Function Prototype	void fontGrayscaleSet (unsigned char ucGrayscale)
Parameter	ucGrayscale: font grayscale, range: 1~8
Return Value	None
Model Supported	All the models
Description	This function sets the font grayscale. And there are 8 levels of grayscale to satisfy different colors depth requirements for different thermal paper, in which ”1” is the lightest and “8” is the darkest. For ZYTPxx-xx4xx and MTPxx-xx4xx, the smaller the gray value is, the faster print speed is. However, since the low gray value may cause the step motor out of step, user should adjust the gray value according to the actual situation. For ZYTPxx -xx5xx and MTPxx -xx5xx, the gray value doesn’t affect the print speed. The settings by fontGrayscaleSet() are effective until printerInit() is executed, printer is reset or printer is turned off.

Table 6-15: printSpeedSet

Function Name	Set the print speed
Function Prototype	void printSpeedSet (unsigned char ucPrintSpeed)
Parameter	ucPrintSpeed: print speed PRINT_SPEED_LOW low speed PRINT_SPEED_MIDIUM moderate speed PRINT_SPEED_HIGHT high speed
Return Value	None
Model Supported	ZYTP80, MTP80, ZYTP58-xx5xx and MTP58-xx5xx

Description	This function sets the print speed. For ZYTP80/MTP80, the maximum speed can only reach the moderate speed (n≤1) when the serial communication baud rate is below 9600bps. The settings by fontGrayscaleSet() are effective until printerInit() is executed, printer is reset or printer is turned off

Table 6-16: fontSizeSet

Function Name	Set the font size
Function Prototype	void fontSizeSet (unsigned char ucFontSize)
Parameter	ucFontSize: font size FONT_SIZE_24 24×24 FONT_SIZE_16 16×16 FONT_SIZE_12 12×12
Return Value	None
Model Supported	ZYTP80, MTP80, ZYTP58-xx5xx and MTP58-xx5xx
Description	This function is valid for both Chinese and English fonts, but only available for the products with multiple fonts supported. The settings by fontSizeSet() are effective until printerInit() is executed, printer is reset or printer is turned off

Table 6-17 selectKanjiMode

Function Name	Select Kanji character mode
Function Prototype	void selectKanjiMode (void)
Parameter	None
Return Value	None
Model Supported	ZYTP80, MTP80, ZYTP58-xx5xx and MTP58-xx5xx
Description	This function can be used only for the Japanese, Simplified Chinese, And Traditional Chinese Models.

Table 6-18 cancelKanjiMode

Function Name	Cancel Kanji character mode
Function Prototype	void cancelKanjiMode (void)
Parameter	None
Return Value	None
Model Supported	ZYTP80, MTP80, ZYTP58-xx5xx and MTP58-xx5xx
Description	This function can be used only for the Japanese, Simplified Chinese, And Traditional Chinese Models.

Table 6-19 selectInternationalChar

Function Name	Select an international character set
Function Prototype	void selectInternationalChar (unsigned char ucCountry)
Parameter	ucCountry: Country 0 U.S.A

	1 France 2 Germany 3 U.K. 4 Denmark I 5 Sweden 6 Italy 7 Spain 8 Japan 9 Norway 10 Denmark II 11 Spain II 12 Latin America 13 Korean 14 Slovenia / Croatia 15 Chinese
Return Value	None
Model Supported	ZYTP80, MTP80, ZYTP58-xx5xx and MTP58-xx5xx
Description	Select an international character set

Table 6-20 selectCharCodePage

Function Name	Select character code table
Function Prototype	void selectCharCodePage (unsigned char ucPageNum)
Parameter	ucPageNum: Character code page 0 PC437(U.S.A.,Standard Europe) 1 Katakana 2 PC850(Multilingual) 3 PC860(Portuguese) 4 PC863(Canadian-French) 5 PC865(Nordic) 6 Simplified Kanji, Hirakana 7 Simplified Kanji 8 Simplified Kanji 16 WPC1252 17 PC866(Cyrillic #2) 18 PC852(Latin 2) 19 PC858(Euro) 254 Page 254 255 Page 255
Return Value	None
Model Supported	ZYTP80, MTP80, ZYTP58-xx5xx and MTP58-xx5xx
Description	The characters of each page are the same for alphanumeric parts (ASCII code: Hexadecimal = 20H to 7FH / Decimal = 32 to 127 20H to 7FH), and different for the escape character parts (ASCII code: Hexadecimal = 80H to

	FFH / Decimal = 128 to 255 80H to FFH).

Table 6-21: picturePrintV

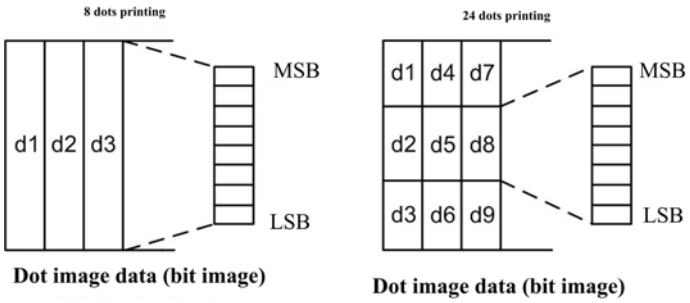
Function Name	Select bit-image mode
Function Prototype	void picturePrintV (unsigned char ucType, unsigned short usHSize, unsigned char * pucDataBuf)
Parameter	cType: bit-image modes as follows: DOT_8_SINGLE_DENSITY 8 dots single density DOT_8_DOUBLE_DENSITY 8 dots double density DOT_24_SINGLE_DENSITY 24 dots single density DOT_24_DOUBLE_DENSITY 24 dots double density usHSize: horizontal dots For ZYTP58 and MTP58: 0 ⩽ usHSize ⩽ 384 For ZYTP80 and MTP80: 0 ⩽ usHSize ⩽ 576 pucDataBuf: starting address of the image buffer
Return Value	None
Model Supported	All the models
Description	pucDataBuf specifies a bit printed to 1 and not printed to 0. If the bit image exceeds one line of print area, the excess part will be ignored. The print result is as follows:  The bit image is only stored in the print buffer and is not printed. Only when the print function is received, the printing starts. After printing, no data exists in the print buffer. If the image printed is too high, it should be divided into smaller images with a height of 8 dots (DOT_8_SINGLE_DENSITY or DOT_8_DOUBLE_DENSITY) or 24 dots (DOT_24_SINGLE_DENSITY or DOT_24_DOUBLE_DENSITY) and printed respectively. After storing the image data, additional information can also be filled into the print buffer to print with the image. Both function printAndFeedNDotLine(24) and function print() can be applied for printing, but print() would cause the line feeding, making the image for several lines discontinuously.

Table 6-22: picturePrintH

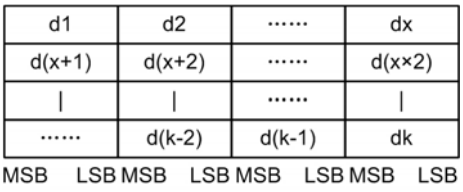
Function Name	Print raster bit image
Function Prototype	void picturePrintH (unsigned char ucType, unsigned short usHSize, unsigned short usVSize, unsigned char * pucDataBuf)
Parameter	cType: raster bit image modes as follows: PICTURE_TYPE_NORMAL normal PICTURE_TYPE_DOUBLE_WIDTH double width PICTURE_TYPE_DOUBLE_HEIGHT double height PICTURE_TYPE_QUADRUPLE double width & double height usHSize: horizontal bytes For ZYTP58 and MTP58: 0 ⩽ usHSize ⩽ 384 For ZYTP80 and MTP80: 0 ⩽ usHSize ⩽ 576 usVSize: vertical dots, range: 0~65535 pucDataBuf: starting address of the image buffer
Return Value	None
Model Supported	All the models
Description	pucDataBuf specifies a bit printed to 1 and not printed to 0. If the bit image exceeds one line of print area, the excess part will be ignored. The bit image is only stored in the print buffer and is not printed. This function executes paper feed for amount needed for printing the bit image regardless of the setting by lineSpaceSet() or lineSpaceSetDefault(). After printing the bit image, this function sets the print position to the beginning of the line, and clears up the buffer. The printing result is as follows:  When this function is executed, the data is transmitted and printed synchronously. So no other printing function is required.

Table 6-23: horizontalTabSet

Function Name	Set horizontal tab positions
Function Prototype	void horizontalTabSet (unsigned char * pucNBuf, unsigned char ucDataNum)
Parameter	pucNBuf: horizontal tab positions, unit: 8 dots For ZYTP58 and MTP58: 0 ⩽ pucNBuf[k] ⩽ 48 For ZYTP80 and MTP80: 0 ⩽ pucNBuf[k] ⩽ 72 pucNBuf[k] ⩾ pucNBuf[k-1] ucDataNum: the number of data, range: 0~16
Return Value	None

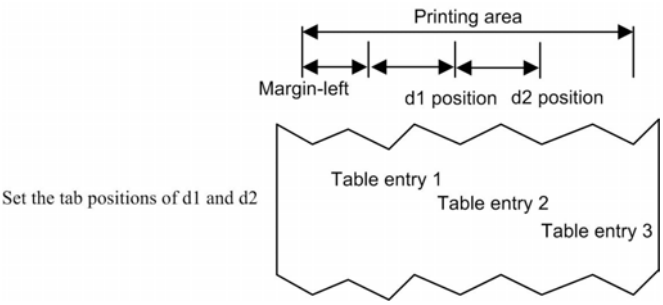
Model Supported	All the models
Description	The tab position is shown as follows:  A maximum of 16 tab positions can be set. When this function is used, any previous horizontal tab settings will be canceled. When pucNBut[k] is less than or equal to pucNBut[k-1], horizontal tab setting is finished, and the following data will be processed as normal data. The tab position can be switched by horizontalTab() function. When the left margin is changed, the tab position is also changed. Horizontal tab position settings are effective until printerInit() is executed, the printer is reset, or the power is turned off.

Table 6-24: horizontalTab

Function Name	Horizontal tab
Function Prototype	void horizontalTab (void)
Parameter	None
Return Value	None
Model Supported	All the models
Description	Horizontal tab position is set by horizontalTab(). If no horizontal tab position is set (it is default setting), this function will be used as print(). If the horizontal tab position exceeds the print area, printing position will be moved to the starting position of next line (Considering as a line is full, print the data and feed one line).

Table 6-25: formPrintV

Function Name	Print the vertical table
Function Prototype	void formPrintV (unsigned char ucVLineNum, unsigned char *pucVLinePos, unsigned char ucItemNum, formItem *psItem)
Parameter	ucVLineNum: the number of vertical line, range: 0~17 pucVLinePos: the positions of vertical lines, unit: 8 dots For ZYTP58 and MTP58: 0 ≤ pucVLinePos [k] ≤ 48 For ZYTP80 and MTP80: 0 ≤ pucVLinePos [k] ≤ 72 pucVLinePos [k] ≥ pucVLinePos [k-1] ucItemNum: the number of table item, range: 0~16 psItem: the setting of item, including position, font type and text. The structure is defined as following: typedef struct __formItem { unsigned char ucPos; /* Set the position of table item */ /* The valid range is the same with pucVLinePos */ unsigned char ucFontType; /* Set the font type */ unsigned char ucDataLen; /* Set the data length of the table item, range: 0~20 */ unsigned char *pucDataBuf; /* Set the data in the table item */ }formItem; The followings are the font types avaialbe in the table item: VF_FONT_TYPE_NORMAL normal VF_FONT_TYPE_EMPHASIZE bold VF_FONT_TYPE_UNDERLINE underline VF_FONT_TYPE_REVERSE_VEDIO inverse
Return Value	None
Model Supported	ZYTPxx-xxxCx and MTPxx-xxxCx

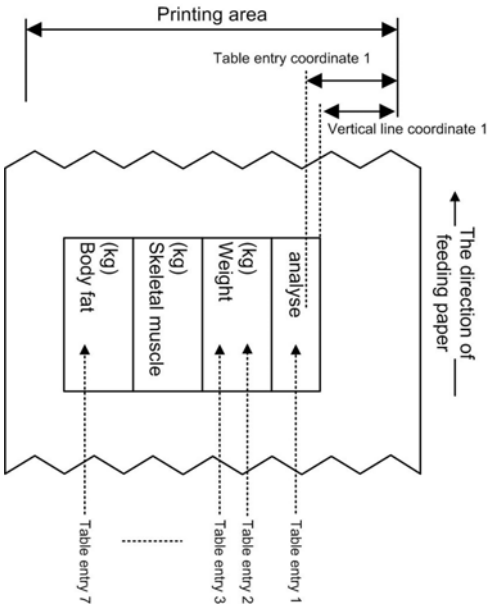
Description	The following diagram is an example for explaining how to print the vertical table:  The reference 0 is located at the right side of the paper in the direction of paper feeding. Each table item contains maximum 10 Chinese characters or 20 English characters If no table border is required, m will be zero.

Table 6-26: barcodeHeightSet

Function Name	Set the height of one-dimension bar code
Function Prototype	void barcodeHeightSet(unsigned char ucHeight)
Parameter	UcHeigh: the height of one-dimension bar code, range: 0~255 dots
Return Value	None
Model Supported	ZYTPxx-xxxBx, ZYTPxx-xxxEx, MTPxx-xxxBx and MTPxx-xxxEx
Description	barcodeHeightSet() sets the height of one-dimension bar code to ucHeight dots, as following shows:  The height is 50 dots  The height is 100 dots The settings by barcodeHeightSet() are effective until printerInit() is executed, printer is reset or printer is turned off.

Table 6-27: barcodeWidthSet

Function Name	Set the width of one-dimension bar code
Function Prototype	void barcodeWidthSet(unsigned char ucWidth)
Parameter	ucWidth: the width of a bar in one-dimension bar code, range: 1~6 dots
Return Value	None
Model Supported	ZYTPxx-xxxBx, ZYTPxx-xxxEx, MTPxx-xxxBx and MTPxx-xxxEx

Description	barcodeWidthSet() sets the width of a bar in one-dimension bar code to ucWidth dots, as following shows:  The width is 3 dots  The width is 4 dots The settings by this function are effective until printerInit() is executed, printer is reset or printer is turned off.

Table 6-28: barcodeHriPosSet

Function Name	Select print position of one-dimension HRI
Function Prototype	void barcodeHriPosSet (unsigned char ucHriPos)
Parameter	ucHriPos: position of one-dimension HRI BARCODE_HRI_POS_NONE none BARCODE_HRI_POS_TOP top BARCODE_HRI_POS_BOTTOM bottom BARCODE_HRI_POS_TOP_BOTTOM top+bottom
Return Value	None
Model Supported	ZYTPxx-xxxBx, ZYTPxx-xxxEx, MTPxx-xxxBx and MTPxx-xxxEx
Description	The font size of HRI to a module with multiple fonts supported, such as xxTPxx-xx5Bx, can be set by barcodeHriFontSet(). The settings by this function are effective until printerInit() is executed, printer is reset or printer is turned off.

Table 6-29: barcodeHriFontSet

Function Name	Select the font size of one-dimension HRI
Function Prototype	void barcodeHriFontSet(unsigned char ucHriFont)
Parameter	ucHriFont: one-dimension HRI BARCODE_HRI_FONT_24 12×24 BARCODE_HRI_FONT_16 8×16 BARCODE_HRI_FONT_12 6×12
Return Value	None
Model Supported	ZYTPxx-xxxBx, ZYTPxx-xxxEx, MTPxx-xxxBx and MTPxx-xxxEx
Description	The settings by this function are effective until printerInit() is executed, printer is reset or printer is turned off.

Table 6-30: barcodePrint

Function Name	Select the font size of one-dimension HRI
Function Prototype	void barcodeHriFontSet(unsigned char ucHriFont)
Parameter	void barcodePrint(unsigned char ucBarcodeSys, unsigned char *pucCodeBuf, unsigned char ucCodeLen); ucBarcodeSys: Bar code system Bar Code System — Number BARCODE_SYS_UPCA — 0 BARCODE_SYS_UPCE — 1 BARCODE_SYS_EAN13 — 2 BARCODE_SYS_EAN8 — 3 BARCODE_SYS_CODE39 — 4 BARCODE_SYS_ITF25 — 5 BARCODE_SYS_CODABAR — 6 BARCODE_SYS_CODE93 — 7 BARCODE_SYS_CODE128 — 8 BARCODE_SYS_EAN128 — 9 pucCodeBuf: the starting address of bar code data ucCodeLen: the length of bar code data
Return Value	None
Model Supported	ZYTPxx-xxxBx, ZYTPxx-xxxEx, MTPxx-xxxBx and MTPxx-xxxEx
Description	The relationship among the bar code system, bar code data and bar code length is shown is Table 6-50. **Notes for UPCA (Number = 0) process:** If the length of input data is any of 11 or 12 bytes, the parity bit will be added automatically for error correcting. The start character, central separating character and stop character will also be added automatically. **Notes for UPCE (Number = 1) process** If the data length is 6 bytes, the system character (NSC) 0 will be added automatically. If the data length is any of 7,8,11 or 12 bytes, the first data (d1) is processed as number system character (NSC) so 0 must be specified. If the length of input data is any of 6, 7, 8, 11 or 12 bytes, the parity bit will be added automatically for error correcting. If the length of input data is any of 6, 7, 8, 11 or 12 bytes, only the shortened 6 bits of bar code HRI will be printed, in which the system character (NSC) and parity code is not included.

Following is the relationship between data transferred and data printed:

Data transferred										Data printed					
d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d1	d2	d3	d4	d5	d6
0~9	0~9	0	0	0	-	-	0~9	0~9	0~9	d2	d3	d9	d10	d11	0
0~9	0~9	1	0	0	-	-	0~9	0~9	0~9	d2	d3	d9	d10	d11	1
0~9	0~9	2	0	0	-	-	0~9	0~9	0~9	d2	d3	d9	d10	d11	2
0~9	0~9	3~9	0	0	-	-	-	0~9	0~9	d2	d3	d4	d10	d11	3
0~9	0~9	0~9	1~9	0	-	-	-	-	0~9	d2	d3	d4	d5	d11	4
0~9	0~9	0~9	0~9	1~9	-	-	-	-	5~9	d2	d3	d4	d5	d6	d11

When 1≤d6≤9, be sure to specify (5≤d11≤9).

The start character and stop character are added automatically.

Notes for JAN13/EAN13 (Number = 2) process

If the length of input data is any of 11 or 12 bytes, the parity bit will be added automatically for error correcting.

Start character, central separating character and stop character will be added automatically.

Notes for JAN8/EAN8 (Number = 3) process

If the length of input data is any of 7 or 8 bytes, the parity bit will be added automatically for error correcting.

Start character, central separating character and stop character will be added automatically.

Notes for CODE39 (Number = 4) process

When the first bar code d1 is not “*”, the printer adds a first character (*) automatically.

When the last bar code dn is not “*”, the printer adds a last character (*) automatically.

When “*” is processed during bar code data processing, the printer processes “*” as a stop character. The printer prints data preceding “*” and finishes function processing. Therefore, data following “*” are processed as normal data.

Parity bit are not calculated and added.

Notes for ITF 25 (Number = 5) process

The start character and stop character will be added automatically

Parity bit are not calculated and added.

Notes for CODABAR (NW-7) (Number = 6) process

Since the start character and stop character are not added automatically, user should add them manually. Its valid range is “A” ~ “D” or “a” ~ “d”.

Parity bit is not calculated and added.

Notes for CODE93 (Number = 7) process

Start character and stop character are added automatically.

Parity codes (2 bits) are calculated and added automatically.

For the bar code HRI printing, no HRI character will be used as start character or stop character.

For the bar code HRI printing, space character will be used as the control character.

Notes for CODE128 (Number = 8) process

Bar code system can identify data intelligently and perform the minimum length encoding without setting the character set (including the start character set) or switching the character set.

The function characters FNC1 to FNC4 can be inputted by using C1H to C4H.

Parity bit is calculated and added automatically.

For bar code HRI printing, space character will be used as control character or FNC1 ~ FNC4.

Notes for EAN128 (Number =9) process

Basic structure:

Start character set	FNC1	AI	Data part	Parity bit A	Parity bit B	Stop character
Added automatically		(d1 … dk)			Added automatically	

Connect structure:

Start character set	FNC1	AI	Data part	Parity bit A	FNC1	AI	Data part	Parity bit A	Parity bit B	Stop character
Added automatically		(d1 … dk)							Added automatically	

Bar code system can identify data intelligently and perform the minimum length encoding without setting the character set (including the start character set) or switching the character set.

The function characters FNC1 to FNC4 can be inputted by using C1H to C4H.

When inputting data, AI should not be added in “()”, since the bar code system will do it automatically. Otherwise error may occur. For example: GS k 74 18 "019501234567890*" is correct, in which 01 is AI. While GS k 74 18 "(01)9501234567890*" is wrong.

When linking two data together, FNC1 (C1H “Decimal = 193”) should be inserted between them. For example: GS k 74 18 "019501234567890*" 193 "029501234567890*.

For bar code HRI printing, the space character is used as control character, but FNC1 ~ FNC4 are removed.

Table 6-31: printerInit

Function Name	Initialize the printer
Function Prototype	void printerInit (void)
Parameter	None
Return Value	None
Model Supported	All the models
Description	This function initializes the printer: 1.Clears the data in the print buffer; 2. Resets the printer modes to the modes that were in effect when the power was turned on.

Table 6-32: printerClearBuffer

Function Name	Clear up the printer buffer (real time)
Function Prototype	void printerClearBuffer (void)
Parameter	None
Return Value	None
Model Supported	All the models
Description	When receiving this command, the printer clears up the buffer immediately.

Table 6-33: printerCutPaper

Function Name	Feed paper and cut paper
Function Prototype	void printerCutPaper(unsigned char ucDotLineNum, unsigned char type)
Parameter	ucDotLineNum: the number of lines for paper feeding before cut Type: cutting mode CUT_TYPE_HALF half cut CUT_TYPE_ALL full cut
Return Value	None
Model Supported	ZYTPxx-xxxxC and MTPxx-xxxxC
Description	This function feeds paper for n lines, and then cuts paper.

Table 6-34: printerStateRT

Function Name	Query printer state (real time)
Function Prototype	unsigned char printerStateRT(unsigned char *pucState, unsigned short usTimeOut)
Parameter	pucState: the state value returned by printer, each bit is independent. RINTER_STATE_NORMAL normal PRINTER_STATE_OVERVOLATGE over voltage PRINTER_STATE_AXOOPEN platen open PRINTER_STATE_OFPAPER paper end PRINTER_STATE_OVERHEAT over heat PRINTER_STATE_CUTTERON cutter down usTimeOut: response is over time (unit: 1ms).
Return Value	FAIL or SUCCESS
Model Supported	All the models

Description	This function queries the state of the printer. Notice that zyPrtLib may process the state value returned automatically, such as calling the entry functions for overheat or paper end.

Table 6-35: printerStateAutoUpload

Function Name	Set/Cancel the printer states automatic back
Function Prototype	void printerStateAutoUpload(unsigned char key)
Parameter	Key: options for state returned, it is multiple selected. RINTER_STATE_NORMAL normal PRINTER_STATE_OVERVOLATGE over voltage PRINTER_STATE_AXOOPEN platen open PRINTER_STATE_OFPAPER paper end PRINTER_STATE_OVERHEAT over heat PRINTER_STATE_CUTTERON cutter down
Return Value	None
Model Supported	All the models
Description	This function sets/cancels printer state automatic back. Notice that all the options are disabled by default, so user should configure them before the first printing after printer was power on or exited from low power mode. Otherwise unpredicted result may occur. The settings by this function are effective until printerInit() is executed, printer is reset or printer is turned off.

Table 6-36: printerBaudRateSet

Function Name	Set printer baud rate
Function Prototype	void printerBaudRateSet (unsigned long ulBaudRate)
Parameter	ulBaudRate: baud rate, such as 9600, valid range: 110~115200
Return Value	None
Model Supported	All the models
Description	This function sets the printer baud rate. Notice that user should configure the baud rate before the first printing after printer was power on or exited from low power mode. Otherwise unpredicted result may occur. The settings by this function are effective until printer is reset or printer is turned off.

Table 6-37: printerFlowCtrlSet

Function Name	Set flow control mode
Function Prototype	void printerFlowCtrlSet(unsigned char Mode)
Parameter	Mode: flow control modes FLOW_CTRL_NONE 0 /* None */ FLOW_CTRL_SOFTWARE 49 /* Software flow control */ FLOW_CTRL_HARDWARE 48 /* Hardware flow control */
Return Value	None
Model Supported	All the models

Description	This function sets the printer flow control mode. Notice that the flow control modes are disabled by default, so user should configure them before the first printing after printer was power on or exited from low power mode. Otherwise unpredicted result may occur. The settings by this function are effective until printerInit() is executed, printer is reset or printer is turned off.

Table 6-38: printerEnterLowPowerMode

Function Name	Enter low power mode
Function Prototype	unsigned char printerEnterLowPowerMode (unsigned short usTimeOut)
Parameter	usTimeOut: response is over time (unit: 1ms).
Return Value	FAIL or SUCCESS
Model Supported	All the models except ZYTPxx-xxxxC and MTPxx-xxxxC
Description	None

Table 6-39: printerExitLowPowerMode

Function Name	Exit low power mode
Function Prototype	unsigned char printerExitrLowPowerMode (unsigned short usTimeOut)
Parameter	usTimeOut: response is over time (unit: 1ms).
Return Value	FAIL or SUCCESS
Model Supported	All the models except ZYTPxx-xxxxC and MTPxx-xxxxC
Description	This function is used for exiting the low power mode. When entering the low power mode, an appropriate delay should be inserted before exiting.

Table 6-40: printerOverHeatHandler

Function Name	Printer over temperature handler entry
Function Prototype	unsigned char printerExitLowPowerMode (unsigned short usTimeOut)
Parameter	None
Return Value	None
Description	printerOverHeatHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-41: printerOverHeatResumeHandler

Function Name	Printer over temperature resume handler entry
Function Prototype	void printerOverHeatResumeHandler (void)
Parameter	None
Return Value	None
Description	printerOverHeatResumeHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-42: printerOfPaperHandler

Function Name	Printer paper end handler entry
Function Prototype	void printerOfPaperHandler (void)
Parameter	None
Return Value	None
Description	printerOfPaperHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-43: printerOfPaperResumeHandler

Function Name	Printer paper end resume handler entry
Function Prototype	void printerOfPaperResumeHandler (void)
Parameter	None
Return Value	None
Description	printerOfPaperResumeHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-44: printerOverVoltageHandler

Function Name	Printer over voltage handler entry
Function Prototype	void printerOverVoltageHandler (void)
Parameter	None
Return Value	None
Description	printerOverVoltageHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-45: printerOverVoltageResumeHandler

Function Name	Printer over voltage resume handler entry
Function Prototype	void printerOverVoltageResumeHandler (void)
Parameter	None
Return Value	None
Description	printerOverVoltageResumeHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-46: printerAxoOpenHandler

Function Name	Printer platen open handler entry
Function Prototype	void printerAxoOpenHandler (void)
Parameter	None
Return Value	None
Description	printerAxoOpenHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-47: printerAxoCloseHandler

Function Name	Printer platen close handler entry
Function Prototype	void printerAxoCloseHandler (void)
Parameter	None
Return Value	None
Description	printerAxoCloseHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-48: printerCutterOnHandler

Function Name	Pinter cutter down handler entry
Function Prototype	void printerCutterOnHandler (void)
Parameter	None
Return Value	None
Description	printerCutterOnHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-49: printerCutterOffHandler

Function Name	Pinter cutter resume handler entry
Function Prototype	void printerCutterOffHandler (void)
Parameter	None
Return Value	None
Description	printerCutterOffHandler() is called in serial receiving interrupt function. Notice that blocking code mustn't be used in function programming and the code handler time should be less than the serial data receiving time.

Table 6-50: The relationships among bar code system, bar code data and bar code data length

Number	Encode system	Bar code data (SP for space)			
		Data length	n	Character set	Data(d)
0	UPC-A	Fixed	n=11,12	0~9	48≤d≤57
1	UPC-E	Fixed	6≤d≤8 n=11,12	0~9	48≤d≤57 (d1=48 when k=7,8,11,12)
2	JAN13(EAN13)	Fixed	n=12,13	0~9	48≤d≤57
3	JAN8(EAN8)	Fixed	n=7,8	0~9	48≤d≤57
4	CODE39	Variable	1≤n≤255	0~9, A~Z SP,$,%,*, +,-,.,/	48≤d≤57 65≤d≤90 d=32,36,37,42,43,45,46,47
5	ITF (Interleaved 2 of 5)	Variable	2≤n≤255 (even)	0~9	48≤d≤57
6	CODABAR (NW-7)	Variable	1≤n≤255	0~9, A~D, a~d $,+,-,.,/,:	48≤d≤57 65≤d≤68 97≤d≤100 d=36,43,45,46,47,58 (65≤d1≤68 65≤dk≤68 97≤d1≤100 97≤dk≤100)
7	CODE93	Variable	1≤n≤255	00H~7FH	0≤d≤127
8	CODE128	Variable	1≤n≤255	00H~7FH C1H~C4H(FNC)	0≤d≤127 D=193,194,195,196
9	UCC/EAN128	Variable	1≤n≤255	00H~7FH C1H~C4H(FNC)	0≤d≤127 D=193,194,195,196

Chapter 7: Rights & Statements

The software or document provided by Guangzhou ZLGMCU Technology Co., Ltd (ZLGMCU hereafter) is intended to provide for you (Customer), and is limited and only for the Product licensed or sale by ZLGMCU.

This software or document is owned by ZLGMCU and/or its suppliers, and protected by applicable copyright law. All rights reserved. Anyone who performs any material breach may face relevant criminal sanction according to applicable law, and should bear corresponding civil liabilities caused by the infringement of the terms and conditions specified in this License. ZLGMCU reserves the right of modifying the document or software without notice the Customer, and has no liability for any affects occurring in use.

This software or document is provided in “as is”. No warranty is made (explicitly, implicitly or legally). Such warranties are including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose to use this document. In no event shall ZLGMCU be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this software or document.

Company name:	Guangzhou ZLGMCU Technology Co., Ltd.
Address:	Floor 2, No.7 Building, Huangzhou Industrial Estate Guangzhou, CHINA
Post code:	510660
Website:	www.zlgmcu.com
Sales:	+86-20-2264-4249
Tech. Support:	+86-20-2264-4361
Facsimile:	+86-20-3860-1859
Sales Email:	80c51mcu@zlgmcu.com
Tech. Sup. Email:	printer@zlgmcu.com