

PROGRAMMING MANUAL

Mobile dot-matrix printer

MPD2

All reserves the right to make changes without notice to the specification and materials contained herein and shall not be responsible for any damage (including consequential) caused by reliance on the materials presented, including but not limited to typographical, arithmetic, or listing errors.

You can refer to our website: www.hpert.com for more information.

REVISION RECORDS

REV.	DATE	DESCRIPTION	Drawn	Checked	Approved
1.0	2013.08.05	——	Lin liting	Lin yang	Ren xiaowei

Content

1. Overview.....	5
1.1 Key terms-----	5
1.2 Command Notation -----	5
2. Miscellaneous Commands.....	6
3. Commands	8
Miscellaneous function commands.....	8
1 ESC @ -----	8
2 LF -----	8
3 CR-----	8
4 ESC J n-----	9
5 ESC d n -----	9
6 HT-----	9
Character commands	10
1 ESC ! n -----	10
2 GS ! n-----	11
3 ESC M n -----	12
ESC – n-----	12
5 ESC E n -----	13
6 ESC G n-----	14
7 GS B n-----	14
8 ESC V n-----	15
Print position commands	15
1 ESC \$ nL nH -----	15
2 ESC D n1 n2...nk NULL -----	16
3 ESC 2-----	17
4 ESC 3 n -----	17
5 ESC SP n-----	18
6 ESC a n-----	18
7 GS L nL nH-----	19
8.GS W nL nH -----	19
Bit-Image commands.....	20
1 ESC * m nL nH d1...dk-----	20
2 GS * x y d1...dk-----	24
3 GS / n-----	26
4 FS p n-----	27
5 FS q n [xL xH yL yH d1...dk]1...[xL xH yL yH d1...dk]n -----	27
User-defined character commands	29

1 ESC % <i>n</i>	29
2 ESC & y c1 c2 [x1 d1...d(y * x1)]...[xk d1...d(y * xk)]	30
3 ESC ?	32
Kanji character commands.....	32
1 FS &	32
2 FS 2 c1 c2 d1...dk.....	32
A Bar code.....	34
<i>A.1 Bar code length ASCII.....</i>	<i>34</i>
B. Pre-printing specifications	34

1. Overview

1.1 Key terms

Real-time commands: These commands are acted on immediately upon being received by the printer ;

Page mode: Under this mode, the printer stores all data in a specified memory and thinks of this as a virtual page. The page is printed when the printer receives print command either FF or ESC FF;

Standard mode: Standard mode is the default mode of printer, namely line mode. Under this mode, the printer prints data and feeds paper upon print line buffer full (data is enough for one print line) or receiving print command like LF;

HRI character: Barcode note character. Human Readable Interface;

NV: Non-volatile memory in which data stored does not loss when powered off. NV: Non-volatile;

RAM : Random Access Memory;

ASB: Auto Send Back

DPI: Print dots per inch (one inch equals to 25.4mm). It is used to identify the resolution of a printer.

Example, 203DPI means 203 print dots per inch. DPI: Dot Per Inch

1.2 Command Notation

[Name] The name of the command.

[Format] The code sequence.

[]k indicates the contents in brackets [] should be repeated k times.

[Range] Gives the allowable ranges, if any, for the command parameters.

[Default] Gives the default values, if any, for the arguments.

[Description] Describes the function of the command.

" – " in the table indicates 0 or 1.

[Notes] Provides important information on setting and using the printer command, if necessary.

[Reference] Gives references, if any.

2. Miscellaneous Commands

The command set of Mobile thermal printer (MPD 2) is compatible with ESC/POS.

Function type:

Chapter	Command	Description
Print commands		
1	<u>ESC @</u>	Initialize printer
2	<u>LF</u>	Print and line feed
3	<u>CR</u>	Print and carriage return
4	<u>ESC J n</u>	Print and feed paper
5	<u>ESC d n</u>	Print and feed n lines
6	<u>HT</u>	Horizontal tab
Character commands		
1	<u>ESC ! n</u>	Select print mode(s)
2	<u>GS ! n</u>	Select character size
3	<u>ESC M n</u>	Select character font
4	<u>ESC – n</u>	Turn underline mode on/off
5	<u>ESC E n</u>	Turn emphasized mode on/off
6	<u>ESC G n</u>	Turn double-strike mode on/off (ESC E)
7	<u>GS B n</u>	Turn white/black reverse printing mode on/off
8	<u>ESC V n</u>	Turn 90° clockwise rotation mode on/off
Print position commands		
1	<u>ESC \$ nL nH</u>	Set absolute print positin
2	<u>ESC D n1 n2...nk NULL</u>	Set horizontal tab position
3	<u>ESC \ nL nH</u>	Set absolute print position
4	<u>ESC 2</u>	Select default line spacing
5	<u>ESC 3 n</u>	Set line spacing
6	<u>ESC SP n</u>	Set character spacing
7	<u>ESC a n</u>	Select justification
8	<u>.GS W nL nH</u>	Set printing area width
Bit image commands		
1	<u>ESC * m nL nH d1...dk</u>	Select bit-image mode
2	<u>GS * x y d1...dk</u>	Define downloaded bit image
3	<u>GS / n</u>	Print downloaded bit image
4	<u>FS p n</u>	Print bit image in NV memory
5	<u>FS q n [xL xH yL yH d1...dk]1...[xL xH yL yH</u>	Print NV bit image

	d1...dk]n	
User-defined character commands		
1	<u>ESC</u> % <u>n</u>	Select/cancel user-defined character set
2	<u>ESC</u> & y c1 c2 ...	Define user-defined characters
3	<u>ESC</u> ?	Cancel user-defined characters
Kanji character commands		
1	<u>FS</u> &	Select Kanji character mode
2	<u>FS</u> 2 c1 c2 d1...dk	Define user-defined Kanji characters

3. Commands

Miscellaneous function commands

1 ESC @

[Name] Initialize printer

[Format]	ASCII	ESC	@
	Hex	1B	40
	Decimal	27	64

[Description] Initialize the printer. All settings, including character font and line spacing settings, are canceled. The data in the print buffer is cleared and the printer mode is reset to the mode that was in effect when the power was turned on. The DIP switch settings are not checked again, the data in the receive buffer is not cleared, and any macro definitions are not cleared.

[Program example]

```
char SendStr[3];
SendStr[0] = 0x1B ;
SendStr[1] = 0x40;
PrtSendData(SendStr, 2);
```

2 LF

[Name] Print and line feed

[Format]	ASCII	LF
	Hex	0A
	Decimal	10

[Description] LF prints the data in the printer buffer and feeds one line.

[Note] This command sets the print position to the beginning of the line.

[Reference] **CR**

[Progame example]

```
char SendStr[2];
SendStr[0]= 0x0A; //C language'\n' feed line
PrtSendData( SendStr, 1);
```

3 CR

[Name] Print and carriage return

[Format]	ASCII	CR
	Hex	0D
	Decimal	13

[Description] When auto line feed is disabled, this command is ignored.

[Note] This command sets the print position to the beginning of the line after being printed.

[Reference] **LF**

[Program example]

```
char SendStr[2];
SendStr[0] = 0x0D;
PrtSendData( SendStr, 1);
```


4 ESC J n

[Name] Print and feed paper

[Format]	ASCII	ESC	J	n
	Hex	1B	4A	n
	Decimal	27	74	n

[Rang] $0 \leq n \leq 255$

[Description] Prints the data in the print buffer and feeds the paper $n \times$ (vertical motion unit).

[note] • This command sets the print position to the beginning of the line after being printed.

•Vertical motion unit space is 0.125mm.

[Reference] **ESC d**

```
[ Program example] char SendStr[4];
                    SendStr[0] = 0x1B;
                    SendStr[1] = 0x4A;
                    SendStr[2] = 0x08;
                    PrtSendData( SendStr, 3);// feeding1mm
```

5 ESC d n

[Name] Print and feed n lines

[Format]	ASCII	ESC	d	n
	Hex	1B	64	n
	Decimal	27	100	n

[Range] $0 \leq n \leq 255$

][Description] Prints the data in the buffer and feeds n lines

[Note] • This command sets the print position to the beginning of the line after being printed.

[Reference] **ESC J**

```
[ Progam example] char SendStr[4];
                    SendStr[0] = 0x1B;
                    SendStr[1] = 0x64;
                    SendStr[2] = 0x02;
                    PrtSendData( SendStr, 3);//feeding 2 lines
```

6 HT

[Name] Horizontal tab

[Format]	ASCII	HT
	Hex	09
	Decimal	9

[Description] HT moves the print start position to the next horizontal tab

[Note] • Set horizontal tab position by ESC D command ESCD command set horizontal tab position.

- This command is ignored unless the next horizontal tab position has been set.
- Set Horizontal tab default to 8 character width (9th, 17th,25thlist) of

character A (12 × 24).

[Reference] **ESC D**

[Program example] `char NextPos = 9;`
`PrtSendData("commodity Name",6);`
`PrtSendData(&NextPos,1);`
`PrtSendData("unti price",4);`
`PrtSendData(&NextPos,1);`
`PrtSendData("quantity",4);`
`PrtSendData(&NextPos,1);`
`PrtSendData("amount",4);`

Character commands

1 ESC ! n

[Name] select print mode

[Format]	ASCII	ESC	!	n
Hex		1B	21	<i>n</i>
Decimal		27	33	<i>n</i>

[Range] $0 \leq n \leq 255$

[Description] Select printing mode by defined parameter n . Parameter n defined as follow:

Bit	Value	Name
0	0	Western character Font A (8× 24), Chinese character Font A (16× 16)
	1	Western character Font B (6 × 12), Chinese character Font B (12× 12)
1	—	Undefined
2	—	Undefined
3	0	Emphasized mode not selected
	1	Emphasized mode selected
4	0	Double-height mode not selected
	1	Double-height mode selected
5	0	Double-width mode selected
	1	Double-width mode not selected
6	—	Undefined
7	0	Underline mode not selected
	1	Underline mode selected

[Note]

- When select double-height and double –width mode, prints character that is forth as much.
- When underline mode is turned on, 90° clockwise-rotated characters and white/black reverse characters cannot be underlines.
- There are some characters with double height or higher height in a line, all characters

align at reference axis

- **ESC M** sets character font. The last command received is valid.
- **ESC E** selects emphasized mode or not. The last command received is valid.
- **ESC –** selects underline mode or not. The last command received is valid.
- **GS!** sets character size. The last command received is valid.
- This command is effective for all characters (except for HRI characters)

[Default] $n = 0$

[Reference] **ESC -, ESC E, GS!, ESC M**

```
[Program example ] char SendStr[4];
                    SendStr[0] = 0x1B;
                    SendStr[1] = 0x21;
                    SendStr[2] = 0x28;// 00101000 double-width emphasized mode
                    PrtSendData( SendStr, 3);
```

2 GS ! n

[Name] Select character size

[Format]	ASCII	GS	!	n
	Hex	1D	21	n
	Decimal	29	33	n

[Range] $0 \leq n \leq 255$ ($1 \leq$ vertical number of times normal font size ≤ 8 , $1 \leq$ horizontal number of times normal font size ≤ 8)

[Description] The number 0~3 to select character height, The number 4~7 to select character width, as follow:

0	1	2	3	Height	4	5	6	7	Width
0	0	0	0	1 times	0	0	0	0	1 times
1	0	0	0	2 times	1	0	0	0	2 times

- [Note]
- This is command is effective for all characters (except for HRI characters).
 - If n is outside of the specified range, this command is ignored.
 - Feeding paper in vertical direction and then all characters are 90° clockwise rotated. Vertical direction and horizontal direction reverse, and that means the priority of this command is lower than SC V. When these two commands are valid, character rotates first and then enlarges.
 - Enlarge characters in the same line with different size, all characters align at reference axis.
 - **ESC !** sets character size. The last command received sets current mode.

[Default] $n = 0$

[Reference] **ESC !, ESC @**

```
[Program example] char SendStr[4];
                    SendStr[0] = 0x1D;
                    SendStr[1] = 0x21;
                    SendStr[2] = 0x01;// 00000001 double height
                    PrtSendData( SendStr, 3);
```

3 ESC M n

[Name] Select character font

[Format]	ASCII	ESC	M	<i>n</i>
	Hex	1B	4D	<i>n</i>
	Decimal	27	77	<i>n</i>

[Rang] $n = 0, 1, 16, 17, 18, 19$

[Description] Selects the character font.

n	Function
0	Western character font (8× 16)
1	Western character font (6× 12)
16	Simplified chinese character font (16×16)
17	Simplified chinese character font (12×12)
18	BIG5 chinese character font (24×24)
19	BIG5 chinese character font (16×16)

[Note] • **ESC !** can also select character font types. However the setting of the last received command is effective.

- When you use this command to set the font, can be set separately chinese font and western font, and independently of each other.

[Reference] **ESC !, ESC @**

```
[Program example] char SendStr[8];  
SendStr[0]=0x1B;  
SendStr[1]=0x4d;  
SendStr[2]=0x00;// Western character font ( $12 \times 24$ )  
SendStr[0]=0x1B;  
SendStr[1]=0x4D;  
SendStr[2]=0x10;// Simplified chinese character font ( $16 \times 16$ )  
PrtSendData( SendStr, 6);// Chinese character font for ( $16 \times 16$ ) ,  
Western is ( $12 \times 24$ ) after printing
```

ESC – n

[Name] Turn underline mode on or off

[Format]	ASCII	ESC	–	<i>n</i>
	Hex	1B	2D	<i>n</i>
	Decimal	27	45	<i>n</i>

[Rang] $0 \leq n \leq 2$

[Description] Base on following n value, turn underline mode on or off

n Decimal	Definition
0	Turn underline mode off
1	Turn underline on (one-dot width)
2	Turn underline on (two-dot width)

[Note] • When underline mode is on, 90 clockwise rotated characters and white/black reverse characters cannot be underlined.

- Character size selection is not effective for underline width
- **ESC !** turns underline on or off. The last command received is valid.
- This command is valid for all English and Chinese characters.

[Default] $n = 0$

[Reference] **ESC !**

[Program example] `char SendStr[16];`

`SendStr[0] = 0x1B;`

`SendStr[1] = 0x2D; // uniline underline`

`SendStr[2] = 0x01;`

`SendStr[3] = 't';`

`SendStr[4] = 'e';`

`SendStr[5] = 's';`

`SendStr[6] = 't';`

`SendStr[7] = 0x0A;`

`SendStr[8] = 0x1B;`

`SendStr[9] = 0x2D;`

`SendStr[10] = 0x00;`

`SendStr[11] = 't';`

`SendStr[12] = 'e';`

`SendStr[13] = 's';`

`SendStr[14] = 't';`

`SendStr[15] = 0x0A;`

`PrtSendData( SendStr, 16);`

5 ESC E n

[Name] Turn emphasized mode on / off

[Format]	ASCII	ESC	E	n
	Hex	1B	45	n
	Decimal	27	69	n

[Range] $0 \leq n \leq 255$

[Description] Turn emphasized mode on/ off

When the LSB (least significant bit) of n is 1, emphasize mode is turned on;

When LSB is 0, emphasized mode is turned off.

- [Note]
- LSB of n is allowed to be used.
 - **ESC !** turns emphasized mode on / off . The last command received is valid.

[Default] $n = 0$

[Reference] **ESC !**, **ESC G**

[Program example] `char SendStr[3];`

`SendStr[0] = 0x1B;`

`SendStr[1] = 0x45;`

`SendStr[2] = 0x01; // emphasized mode`

`PrtSendData(SendStr,3);`

6 ESC G *n*

[Name] Turn double-strike mode on/off

[Format]	ASCII	ESC	G	<i>n</i>
	Hex	1B	47	<i>n</i>
	Decimal	27	71	<i>n</i>

[Range] $0 \leq n \leq 255$

[Description] turn double-strike mode on/off

When the LSB (least significant bit) of *n* is 1, double-strike mode is turned on;

When LSB is 0, double-strike mode is turned off.

- [Note]
- Only LSB of *n* is allowed to be used.
 - Output of printer appear the same in Double –strike mode and emphasized mode.

[Default] *n* = 0

[Reference] **ESC E, ESC !**

[Program example]

```
char SendStr[3];
SendStr[0] = 0x1B;
SendStr[1] = 0x47;
SendStr[2] = 0x01;// turn double-strike mode on
PrtSendData( SendStr, 3);
```

7 GS B *n*

[Name] Turn white/black reverse printing mode on/off

[Format]	ASCII	GS	B	<i>n</i>
	Hex	1D	42	<i>n</i>
	Decimal	29	66	<i>n</i>

[Range] $0 \leq n \leq 255$

[Description] Turn white/black reverse printing mode on/off.

When the LSB (least significant bit) of *n* is 0, white/ black reverse printing mode is turned off;

When LSB is 1, white/black reverse printing mode is turned on

Only LSB of *n* is allowed to be used.

- This command is effective for all characters (except for HRI characters)
- In white/black reverse printing mode, characters are printed in white on a black background.
- This command is not effective for bitmap, user-defined bitmap, bar code, barcode graphic character and the space skipped by HT \$ and ESC\.
- White /black reverse mode prior to underline mode. When select white/black reverse mode, underline mode is prohibited but not canceled even turn underline mode on.

[Default] *n* = 0

[Program example]

```
char SendStr[3];
SendStr[0] = 0x1D;
SendStr[1] = 'B';
SendStr[2] = 1;// white/black reverse printing
```

```
PrtSendData( SendStr, 3);
```

8 ESC V *n*

[Name] Turn 90° clockwise rotation mode on/off

[Format] ASCII ESC V *n*
 Hex 1B 56 *n*
 Decimal 27 86 *n*

[Range] $0 \leq n \leq 3$

[Description] Turn 90° clockwise rotation mode on/off

n Decimal	Definition
0	Turn 90° clockwise rotation mode off
1	turn 90° clockwise rotation mode
2	turn 180° clockwise rotation mode
3	turn 270° clockwise rotation mode

[Note] • When turn underline mode, 90°clockwise-rotated characters, underline mode is canceled.

- Under 90° clockwise rotation mode, the direction of characters with double-width and double-width enlargement reverse to that of characters with double-width and double-width enlargement under general mode.

[Default] *n* = 0

[Reference] **ESC !**, **ESC –**

[Program example] char SendStr[3];
 SendStr[0] = 0x1B;
 SendStr[1] = 0x56;
 SendStr[2] = 0x02;// 180° rotation
 PrtSendData(SendStr, 3);

Print position commands

1 ESC \$ *nL nH*

[Name] Set absolute print position

[Format] ASCII ESC \$ *nL nH*
 Hex 1B 24 *nL nH*
 Decimal 27 36 *nL nH*

[Range] $0 \leq nL \leq 255$

$0 \leq nH \leq 255$

[Description] Sets the print starting position from the beginning of the line. From the beginning of the line to the printer starting position is about N horizontal motion units.

nL and nH are the low order and high order of N with double bytes and signless integer.

[Note] • If set print position exceeds printable area (N>384) , the value of printable

area is N=384.

[Reference] **ESC **

[Program example] char SendStr[4];
 SendStr[0] = 0x1B;
 SendStr[1] = 0x24;
 SendStr[2] = 0x18;//3×8=24
 SendStr[3] = 0x00;
 PrtSendData(SendStr, 4); Set the absolute position 3 MM from the
 left margin (24 horizontal motion unit)
 PrtSendData (Start printing from left margin 3 mm \n, 22)

2 ESC D n1 n2...nk NULL

[Name] Set horizontal tab positions

[Format]	ASCII	ESC	D	<i>n1...nk NULL</i>
	Hex	1B	44	<i>n1...nk 00</i>
	Decimal	27	68	<i>n1...nk 0</i>

[Range] $1 \leq n \leq 255$ $0 \leq k \leq 8$

[Description] Set horizontal tap position.

n columns from the beginning of a line.

k indicates the total number of horizontal tab position to be set.

- [Note]
- Tap position as the value storage, the value is n characters width, and measures from the beginning of line. Character width includes default character width between characters.
 - Characters enlargement (ESC ! GS !) is not effective for this command.
 - This command cancels any previous horizontal tab settings.
 - Set n=8, through sending HT, print position is moved to the ninth list.
 - Set 8 tap position (k=8). Date more than 8 tap position will be processed as general data.
 - Transfer [n]k according to ascending order, put NULL code 0 at the end.
 - In this command, $n_k > n_{(k-1)}$, if $n_k \leq n_{(k-1)}$, data after $n_{(k-1)}$ processes as the general date processing after setting finished.
 - **ESC D NULL** cancels horizontal tap position.
 - Previous horizontal tap position stay the same even change of character width.

[Default] Default tap position is character A (12 × 24) and character spacing 8 (list9, 17,25,.....).

[Reference] **HT**

[Program example] char SendStr[7];
 char NextPos = 0x09;
 SendStr[0] = 0x1B;
 SendStr[1] = 0x44;
 SendStr[2] = 0x0B;// 10 character spacing from first list
 SendStr[3] = 0x11;// 16 character spacing from first list


```

SendStr[4] = 0x17;// 22 character spacing   from first list
SendStr[5] = 0x1D;// 28 character spacing   from first list
SendStr[6] = 0x00; // End
PrtSendData(SendStr,7)
PrtSendData("Name",4);
PrtSendData(&NextPos,1);
PrtSendData("Chinses",4);
PrtSendData(&NextPos,1);
PrtSendData("Mathematics",4);
PrtSendData(&NextPos,1);
PrtSendData("Foreign language",4);
PrtSendData(&NextPos,1);
PrtSendData("Amount",4);

```

3 ESC 2

[Name] Select default line spacing

[Format]	ASCII	ESC	2
	Hex	1B	32
	Decimal	27	50

[Description] Set default line spacing to 1mm (8 vertical motion units)

[Note] • This command is effective for line spacing between bit image and character.

[Reference] **ESC 3**

```

[Program example] char SendStr[4];
                  SendStr[0] = 0x1B;
                  SendStr[1] = 0x32;
                  PrtSendData(SendStr,2);

```

4 ESC 3 n

[Name] Set line spacing

[Format]	ASCII	ESC	3	<i>n</i>
	Hex	1B	33	<i>n</i>
	Decimal	27	51	<i>n</i>

[Range] $0 \leq n \leq 255$

[Description] Sets the line spacing to $n \times$ (vertical motion unit)

[Note] • This command is effective for line spacing between image and characters.

[Default] $n = 8$

[Reference] **ESC 2**

```

[Program example] char SendStr[4];
                  SendStr[0] = 0x1B;
                  SendStr[1] = 0x33;
                  SendStr[2] = 0x10;
                  PrtSendData(SendStr,3);// set the line spacing to vertical motion 16
                  units ( 2mm).

```

5 ESC SP *n*

[Name] Set character spacing

[Format]

ASCII	ESC	SP	<i>n</i>
Hex	1B	20	<i>n</i>
Decimal	27	32	<i>n</i>

[Range] $0 \leq n \leq 255$

[Description] Sets the right-side character spacing to $n \times$ (horizontal motion unit)

[Note]

- Under double-width mode, right-side character spacing is twice as much as that of normal characters spacing.
- This command is effective for all characters. (except for HRI characters.)

[Default] $n = 0$

[Program example]

```
char SendStr[4];
SendStr[0] = 0x1B;
SendStr[1] = 0x20;
SendStr[2] = 8;
```

6 ESC a *n*

[Name] Select justification

[Format]

ASCII	ESC	a	<i>n</i>
Hex	1B	61	<i>n</i>
Decimal	27	97	<i>n</i>

[Range] $0 \leq n \leq 2$

[Description] Aligns all the data in one line to position specified by *n*.
n value and definition

<i>n</i>	definition
0	Left justification
1	Center
2	Right justification

[Note]

- This command is enable only when processesd at the beginning of a line.
- This command is effective in the printing area.
- This command execute the blank area justification by **HT**, **ESC \$** or **ESC **.

[Default] $n = 0$

[Program example]

```
char SendStr[4];
SendStr[0] = 0x1B;
SendStr[1] = 0x61;
SendStr[2] = 0x01;
PrtSendData(SendStr,3);// Set horizontal justification to center.
```

7 GS L nL nH

[Name] Set left margin

[Format] ASCII GS L nL nH
 Hex 1D 4C nL nH
 Decimal 29 76 nL nH

[Range] $0 \leq nL \leq 255$ $0 \leq nH \leq 255$

[Description] Set left margin to N horizontal motion units. nL is low-order byte and nH is high-order byte, of the integer with no mark and double bytes. $N = nL + nH \times 256$, left margin is width between left side of printing area

[Note] • This command is enabled only when processed at the beginning of a line.
 • Max left margin is 336. If it exceeds 336, will regards left margin as 336.

[Default] nL = 0, nH = 0

[Program example] char SendStr[4];
 SendStr[0] = 0x1D;
 SendStr[1] = 0x4C;
 SendStr[2] = 0x10;
 SendStr[3] = 0x00;
 PrtSendData(SendStr,4); // Set left margin to 16 horizontal motion units (2mm).

8. GS W nL nH

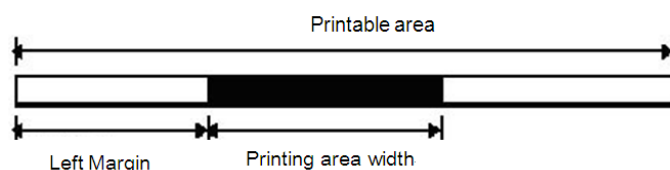
[Name] Set printing area width

[Format] ASCII GS W nL nH
 Hex 1D 57 nL nH
 Decimal 29 87 nL nH

[Range] $0 \leq nL \leq 255$
 $0 \leq nH \leq 255$

[Description] Set printing area width using nL and nH.

• Set printing area width to $[(nL + nH \times 256) \times 0.125\text{mm}]$ from the beginning of a line.



[Note] • In standard mode, this command is enabled only when processed at the beginning of a line.
 • This command is ineffective in page mode, all the command data is managed as normal characters.
 • This command does not affect printing in page mode.
 • If the command sets the value of left margin + printing area width more than the printable area, the printing area width is the printable area width-left margin.

[Default] related to the actual printable width, different printer types set different.

Bit-Image commands

1 ESC * m nL nH d1...dk

[Name] Select bit-image mode

[Format] ASCII ESC * *m nL nH d1...dk*
 Hex 1B 2A *m nL nH d1...dk*
 Decimal 27 42 *m nL nH d1...dk*

[Range] *m* = 0, 1, 32, 33
 $0 \leq nL \leq 255$
 $0 \leq nH \leq 1$
 $0 \leq d \leq 255$

[Description] Printing height is 8 dots or 24 dots, width should not exceeds printable area of black & white bitmap. Parameter definition as follow :
 Use m to select bitmap mode, nL and nH set the dots of horizontal direction of bitmap.

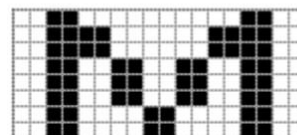
m	Vertical dots (Heights)	Double width mode
0	8	double width
1	8	Single width
32	24	double width
33	24	Single width

nL nH refer to the low – order and high-order byte of the integer with no mark and double bytes , means the dots in the horizontal direction in bitmap. Max value of single width is 384 and max value of double width is 192.

d1...dk indicates bitmap data, details as following figure

[Program example] example 1. m=0 (8 dots, double width) d1 indicates the first and the second list data, dk indicates the $(2k-1)^{th}$ and $2k^{th}$ list data, bn indicates the n^{th} of byte

d1	d2	d3	d4	d5	d6	d7	d8	d9	
0	1	0	0	0	0	0	1	0	b7
0	1	1	0	0	0	1	1	0	b6
0	1	1	0	0	0	1	1	0	b5
0	1	0	1	0	1	0	1	0	b4
0	1	0	1	0	1	0	1	0	b3
0	1	0	1	0	1	0	1	0	b2
0	1	0	0	1	0	0	1	0	b1
0	1	0	0	1	0	0	1	0	b0



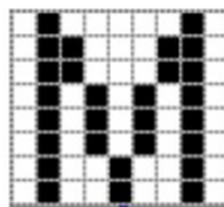
Amplification print figure.

Print Image showed, procession code as follow:

```
char SendStr[15];
SendStr[0] = 0x1B;
SendStr[1] = 0x2A;
SendStr[2] = 0x00; //m=0 ( height 8 dots, double width)
SendStr[3] = 0x09; // ( image width 9 dotds)
SendStr[4] = 0x00;
SendStr[5] = 0x00; // (image dot line bitmap)
SendStr[6] = 0xFF;
SendStr[7] = 0x60;
SendStr[8] = 0x1C;
SendStr[9] = 0x03;
SendStr[10] = 0x1C;
SendStr[11] = 0x60;
SendStr[12] = 0xFF;
SendStr[13] = 0x00;
SendStr[14] = 0x0a; // paper feed
PrtSendData(SendStr,15); // print image
```

Example2: m=1 (8 dots, single width) d1 indicates the first list data, dk indicates the kth list data, bn indicates the nth of byte

d1	d2	d3	d4	d5	d6	d7	d8	d9	
0	1	0	0	0	0	0	1	0	b7
0	1	1	0	0	0	1	1	0	b6
0	1	1	0	0	0	1	1	0	b5
0	1	0	1	0	1	0	1	0	b4
0	1	0	1	0	1	0	1	0	b3
0	1	0	1	0	1	0	1	0	b2
0	1	0	0	1	0	0	1	0	b1
0	1	0	0	1	0	0	1	0	b0



Amplification print figure.

Print Image showed, procession code as follow:

```
char SendStr[15];
SendStr[0] = 0x1B;
SendStr[1] = 0x2A;
SendStr[2] = 0x01; //m=1 (height 8 dots, no enlargement )
SendStr[3] = 0x09; // image width 9 dots
```

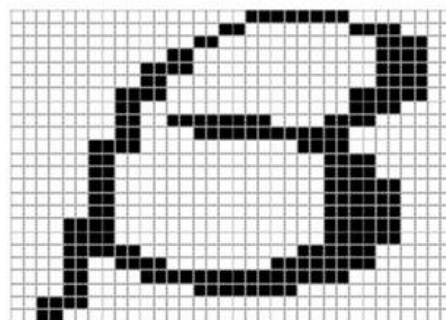
```

SendStr[4] = 0x00;
SendStr[5] = 0x00; // image dot line bitmap
SendStr[6] = 0xFF;
SendStr[7] = 0x60;
SendStr[8] = 0x1C;
SendStr[9] = 0x03;
SendStr[10] = 0x1C;
SendStr[11] = 0x60;
SendStr[12] = 0xFF;
SendStr[13] = 0x00;
SendStr[14] = 0x0a; // paper feed
PrtSendData(SendStr,15); // print image

```

Example3: m=32 (24 dots, double width) d1,d2,d3 indicate the first, second and third list data, dk indicates the kth list data, bn indicates the nth of byte

	d4d7	d49
d1	0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0	b7
	0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0	b6
	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0	b5
	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1	b4
	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1	b3
	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1	b2
	0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1	b1
	0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1	b0
d2	0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 1 0	b7
	0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 0	b6
	0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0	b5
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0	b4
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0	b3
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0	b2
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0	b1
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0	b0
d3	0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0	b7
	0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0	b6
	0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0	b5
	0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0	b4
	0 0 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0	b3
	0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0	b2
	0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	b1
	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	b0



Amplification print figure

Print Image showed, procession code as follow:

```

char SendStr[64];
SendStr[0] = 0x1B;
SendStr[1] = 0x2A;
SendStr[2] = 0x20;//m=32 ( Height 24 dots, double width)
SendStr[3] = 0x11;// 17dots image width 17 dots

```

```

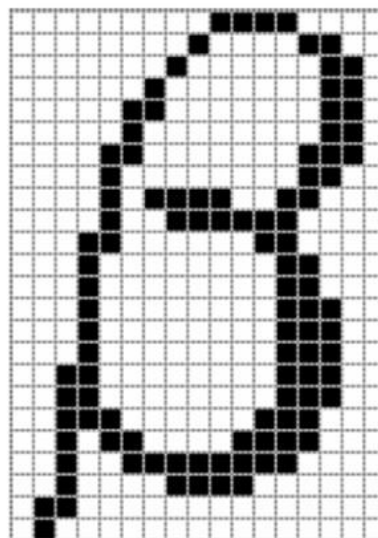
SendStr[4] = 0x00;
// image data
SendStr[5] = 0x00; SendStr[6] = 0x00; SendStr[7] = 0x00;//1
SendStr[8] = 0x00; SendStr[9] = 0x00; SendStr[10] = 0x03;//2
SendStr[11] = 0x00; SendStr[12] = 0x00; SendStr[13] = 0xFE;//3
SendStr[14] = 0x00; SendStr[15] = 0x3F; SendStr[16] = 0xE0;//4
SendStr[17] = 0x03; SendStr[18] = 0xE0; SendStr[19] = 0x30;//5
SendStr[20] = 0x0E; SendStr[21] = 0x00; SendStr[22] = 0x18;//6

SendStr[23] = 0x11; SendStr[24] = 0x00; SendStr[25] = 0x08;//7
SendStr[26] = 0x20; SendStr[27] = 0xC0; SendStr[28] = 0x0C;//8
SendStr[29] = 0x40; SendStr[30] = 0xC0; SendStr[31] = 0x0C;//9
SendStr[32] = 0x80; SendStr[33] = 0xC0; SendStr[34] = 0x0C;//10
SendStr[35] = 0x80; SendStr[36] = 0x40; SendStr[37] = 0x1C;//11
SendStr[38] = 0x80; SendStr[39] = 0x60; SendStr[40] = 0x1C;//12
SendStr[41] = 0x80; SendStr[42] = 0xFF; SendStr[43] = 0xF8;//13
SendStr[44] = 0x43; SendStr[45] = 0x9F; SendStr[46] = 0xF0;//14
SendStr[47] = 0x7F; SendStr[48] = 0x07; SendStr[49] = 0xC0;//15
SendStr[50] = 0x3E; SendStr[51] = 0x00; SendStr[52] = 0x00;//16
SendStr[53] = 0x00; SendStr[54] = 0x00; SendStr[55] = 0x00;//17
PrtSendData(SendStr,56);//print image

```

m=33 (24 dot, singel width) d1、 d2、 d3 indicats the first list of priting data, so bn indicats the n^{th} of

	d4d7	d49
d1	0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0	b7
	0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0	b6
	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0	b5
	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0	b4
	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0	b3
	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0	b2
	0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0	b1
	0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0	b0
d2	0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 0 0 0	b7
	0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 0 0 0	b6
	0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0	b5
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0	b4
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0	b3
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0	b2
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0	b1
	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0	b0
d3	0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0	b7
	0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0	b6
	0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0	b5
	0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 0	b4
	0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0	b3
	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	b2
	0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	b1
	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	b0



Amplification print figure

Print Image showed, procession code as follow:

```
char SendStr[64];
SendStr[0] = 0x1B;
SendStr[1] = 0x2A;
SendStr[2] = 0x21; // m=33 ( height 24 dots, no enlargement)
SendStr[3] = 0x11; // 17dots image width 17 dots
SendStr[4] = 0x00;
// image data

SendStr[5] = 0x00; SendStr[6] = 0x00; SendStr[7] = 0x00; //1
SendStr[8] = 0x00; SendStr[9] = 0x00; SendStr[10] = 0x03; //2
SendStr[11] = 0x00; SendStr[12] = 0x00; SendStr[13] = 0xFE; //3
SendStr[14] = 0x00; SendStr[15] = 0x3F; SendStr[16] = 0xE0; //4
SendStr[17] = 0x03; SendStr[18] = 0xE0; SendStr[19] = 0x30; //5
SendStr[20] = 0x0E; SendStr[21] = 0x00; SendStr[22] = 0x18; //6
SendStr[23] = 0x11; SendStr[24] = 0x00; SendStr[25] = 0x08; //7
SendStr[26] = 0x20; SendStr[27] = 0xC0; SendStr[28] = 0x0C; //8
SendStr[29] = 0x40; SendStr[30] = 0xC0; SendStr[31] = 0x0C; //9
SendStr[32] = 0x80; SendStr[33] = 0xC0; SendStr[34] = 0x0C; //10
SendStr[35] = 0x80; SendStr[36] = 0x40; SendStr[37] = 0x1C; //11
SendStr[38] = 0x80; SendStr[39] = 0x60; SendStr[40] = 0x1C; //12
SendStr[41] = 0x80; SendStr[42] = 0xFF; SendStr[43] = 0xF8; //13
SendStr[44] = 0x43; SendStr[45] = 0x9F; SendStr[46] = 0xF0; //14
SendStr[47] = 0x7F; SendStr[48] = 0x07; SendStr[49] = 0xC0; //15
SendStr[50] = 0x3E; SendStr[51] = 0x00; SendStr[52] = 0x00; //16
SendStr[53] = 0x00; SendStr[54] = 0x00; SendStr[55] = 0x00; //17
PrtSendData(SendStr,56); // Print image
```

[Note] • If m exceeds set range, the unexpected results may appear.

- If bit image data exceeds set line dots, the exceeding data will be ignored.
- Printer returns to normal date procession mode after printing bit image.
- The print position is (emphasized mode, double strike mode, underline, character size, or white/black reverse printing mode) not effective for this command.

2 GS * x y d1...dk

[Name]	Define downloaded bit image			
[Format]	ASCII	GS	*	<i>x y d1...dk</i>
	Hex	1D	2A	<i>x y d1...dk</i>
	Decimal	29	42	<i>x y d1...dk</i>
[Range]	$1 \leq x \leq 255$			
	$1 \leq y \leq 48$			
	$x * y \leq 576$			
	$0 \leq d \leq 255$			
	$k = x * y * 8$			
[Description]	Using x and y dots to define downloaded bit image.			

[Note]

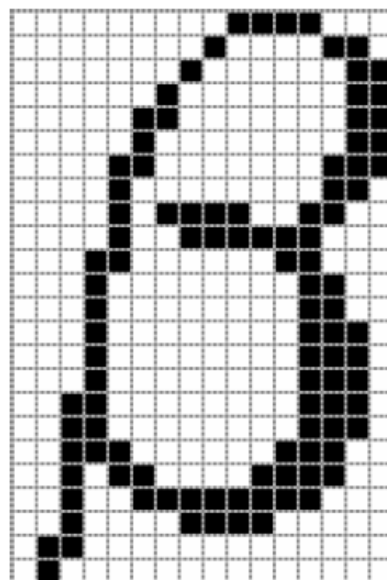
- $x*8$ dots in the horizontal direction
- $y*8$ dots in the vertical direction
- If $x * y$ exceeds specified range, this command is prohibited.
- d indicates bit image data. Data d set printing 0, no printing 1.
- Bit image specified by this command print through the command **GS/n**.
- Clear the downloaded bit image under following cases;
 1. Execute **ESC@**.
 2. The printer is reset and the poser is turned off.

[Program example]

The relation between downloaded bit image and printing date showed as follow:

If $x=2$, $y=3$, $d1.....dk$ in left picture, bitmap right picture

d4d7															d45						
d1	{	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	b7			
		0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	b6		
		0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	b5		
		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	b4	
		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	b3	
		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	b2
		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	b1
		0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	1	1	b0
d2	{	0	0	0	0	1	0	0	0	1	1	1	0	0	1	1	0	0	0	b7	
		0	0	0	0	1	0	0	0	1	1	1	1	1	1	0	0	0	0	b6	
		0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	b5	
		0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	b4	
		0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	b3	
		0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	b2	
		0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	b1	
		0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	b0	
dy y=3	{	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	b7	
		0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	b6	
		0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	b5	
		0	0	1	0	1	1	0	0	0	0	1	1	1	1	1	1	0	0	b4	
		0	0	1	0	0	1	1	1	1	1	1	1	1	1	0	0	0	b3		
		0	0	1	0	0	0	0	1	1	1	1	1	0	0	0	0	0	b2		
		0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b1	
		0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	b0	



Print Image showed, proccession code as follow:

```
char SendStr[64];
SendStr[0] = 0x1D;
SendStr[1] = 0x2A;
SendStr[2] = 2;//x=2 ( Width 16 dots)
SendStr[3] = 3;//y=3 ( Height 24 dots)
// Image data
SendStr[4] = 0x00; SendStr[5] = 0x00; SendStr[6] = 0x00;//1
SendStr[7] = 0x00; SendStr[8] = 0x00; SendStr[9] = 0x03;//2
SendStr[10] = 0x00; SendStr[11] = 0x00; SendStr[12] = 0xFE;//3
SendStr[13] = 0x00; SendStr[14] = 0x3F; SendStr[15] = 0xE0;//4
```

```

SendStr[16] = 0x03; SendStr[17] = 0xE0; SendStr[18] = 0x30;//5
SendStr[19] = 0x0E; SendStr[20] = 0x00; SendStr[21] = 0x18;//6
SendStr[22] = 0x11; SendStr[23] = 0x00; SendStr[24] = 0x08;//7
SendStr[25] = 0x20; SendStr[26] = 0xC0; SendStr[27] = 0x0C;//8
SendStr[28] = 0x40; SendStr[29] = 0xC0; SendStr[30] = 0x0C;//9
SendStr[31] = 0x80; SendStr[32] = 0xC0; SendStr[33] = 0x0C;//10
SendStr[34] = 0x80; SendStr[35] = 0x40; SendStr[36] = 0x1C;//11
SendStr[37] = 0x80; SendStr[38] = 0x60; SendStr[39] = 0x1C;//12

```

```

SendStr[40] = 0x80; SendStr[41] = 0xFF; SendStr[42] = 0xF8;//13
SendStr[43] = 0x43; SendStr[44] = 0x9F; SendStr[45] = 0xF0;//14
SendStr[46] = 0x7F; SendStr[47] = 0x07; SendStr[48] = 0xC0;//15
SendStr[49] = 0x3E; SendStr[50] = 0x00; SendStr[51] = 0x00;//16
PrtSendData(SendStr,52);/ defined image

```

```

SendStr[0] = 0x1D;
SendStr[1] = 0x2F;
SendStr[2] = 0x00;
PrtSendData(SendStr,3);// print image

```

3 GS / n

[Name] Print a downloaded bit image

[Format] ASCII GS / n
Hex 1D 2F n
Decimal 29 47 n

[Range] $0 \leq n \leq 3$

[Description] Print a downloaded bit image using the mode specified by n.
The definition of n:

n	Enlargement
0	Normal
1	Double width
2	Double height
3	Double width double height

- [Note]
- If bit image data is not defined, this command is ignored
 - The print position is (emphasized mode, double strike mode, underline, character size, or white/black reverse printing mode) not effective for this command.
 - If the downloaded bit image exceeds printable area, the exceeding data won't be printed.
 - If print area specified by **GS L** is smaller than image width, it will minish left margin to print bit image.
 - If bit image height exceeds 64 dots, this command is ignored. If printing with double height, bit image height shall not be more than 32 dots.

[Reference] **GS ***

[Program example] Please refer to **3.4.2 GS *** program example

4 FS p n

[Name] Print bit image in NV memory

[Format] ASCII FS p n
 Hex 1C 70 n
 Decimal 28 112 n

[Range] $0 \leq n \leq 7$

[Description] This command prints the pre-stored non-volatile memory in the printer 2 values in the bitmap. Printer non-volatile memory bitmap on a PC via a special tool to generate and write the bitmap width of up to 128, the maximum height of 64.

n is the number of the NV bit image .

- [Note]
- This command is not effective when the specified NV bit image has not been defined.
 - NV bit image must be 2 value bit image.
 - This command is not affected by print modes (emphasized, double-strike, underline, character size, white/black reverse printing, or 90° rotated characters, etc.), except upside-down printing mode.
 - The printing area width is extended to the right in NV bit image mode up to one line vertically. In this case, printing does not exceed the printable area.
 - Need to use special tools to upload print bitmaps, see [three, MPT-II printer tool]. In this way upload bitmap is not lost, unless the re-upload to another overlay bitmap.

[Reference] **ESC *, GS /**

[Program example] `char SendStr[4];
 SendStr[0] = 0x1C;
 SendStr[1] = 0x50;
 SendStr[2] = 0x01;
 PrtSendData(SendStr,3); // Print number 1 of the NV bit image`

5 FS q n [xL xH yL yH d1...dk]1...[xL xH yL yH d1...dk]n

[Name] Define NV bit image

[Format] ASCII FS q n [xL xH yL yH d1...dk]...[xL xH yL yH d1...dk]
 Hex 1C 71 n [xL xH yL yH d1...dk]...[xL xH yL yH d1...dk]
 Decimal 28 113 n [xL xH yL yH d1...dk]...[xL xH yL yH d1...dk]

[Range] $1 \leq n \leq 255$

$0 \leq xL \leq 255$

$1 \leq (xL + xH \times 256) \leq 1023$

$1 \leq (yL + yH \times 256) \leq 800$

$0 \leq d \leq 255$

$k = (xL + xH \times 256) \times (yL + yH \times 256) \times 8$

Total defined data area = 64K bytes

[Description] Define the NV bit image specified by n.

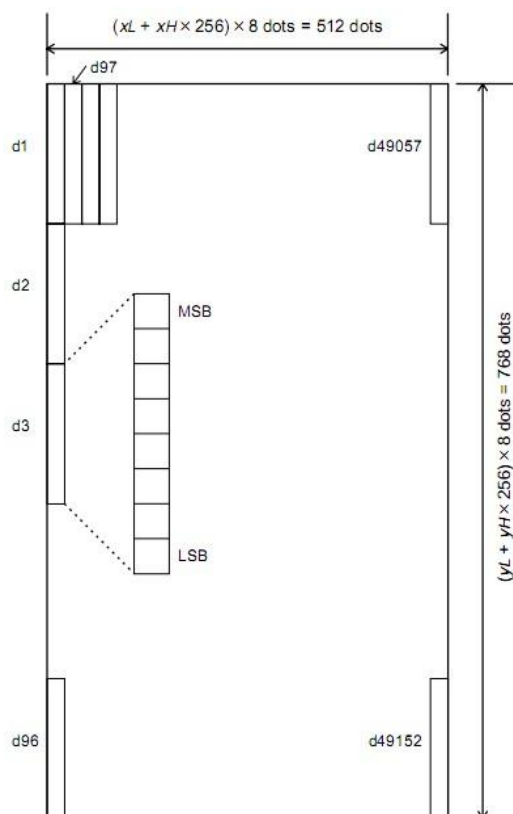
- n specifies the number of the defined NV bit image.
- xL, xH specifies $(xL + xH \times 256) \times 8$ dots in the horizontal direction for the NV bit image you are defining.

[Notes]

- yL, yH specifies $(yL + yH \times 256) \times 8$ dots in the vertical direction for the NV bit image you are defining.
- Frequent write command executions may damage the NV memory. Therefore, it is recommended to write the NV memory 10 times or less a day.
- This command cancels all NV bit images that have already been defined by this command. The printer cannot redefine only one of several data definitions previously defined. In this case, all data needs to be sent again.
- During processing of this command, the printer is BUSY when writing data to the user NV memory and stops receiving data. Therefore it is prohibited to transmit the data, including real-time commands, during the execution of this command.
- NV bit image is a bit image defined in non-volatile memory by **FS q** and printed by **FS p**.
 - In standard mode, this command is effective only when processed at the beginning of the line.
 - This command is effective when 7 bytes <FS yH> of the command are processed normally.
 - When the amount of data exceeds the capacity left in the range defined by xL, xH, yL, yH, the printer processes xL, xH, yL, yH out of the defined range.
 - In the first group of NV bit images, when any of the parameters xL, xH, yL, yH is out of the definition range, this command is disabled.
 - In groups of NV bit images other than the first one, when the printer encounters xL, xH, yL, yH out of the defined range, it stops processing this command and starts writing into the NV images. At this time, NV bit images that haven't been defined are disabled (undefined), but any NV bit images before that are enabled.
 - The d indicates the definition data. In data (d) a 1 bit specifies a dot to be printed and a 0 bit specifies a dot not to be printed.
 - This command defines n as the number of a NV bit image. Numbers rise in order from NV bit image 01H. Therefore, the first data group [xL xH yL yH d1...dk] is NV bit image 01H, and the last data group [xL xH yL yH d1...dk] is NV bit image n. The total agrees with the number of NV bit images specified by the command **FS p**.
 - The definition data for an NV bit image consists of [xL xH yL yH d1...dk]. Therefore, when only one NV bit image is defined n=1, the printer processes a data group [xL xH yL yH d1...dk] once. The printer uses $((data: (xL + xH \times 256) \times (yL + yH \times 256) \times 8) + [header :4])$ bytes of NV memory.
 - The definition area in this printer is a maximum of 64K bytes. This command can define several NV bit images, but cannot define bit image data whose total capacity [bit image data + header] exceeds 40K bytes.
 - The printer is busy immediately before writing into NV memory
 - The printer does not transmit ASB status or perform status detection during processing of this command even when ASB is specified.
 - When this command is received during macro definition, the printer ends macro definition, and begins performing this command.
 - Once an NV bit image is defined, it is not erased by performing **ESC @**, reset, and power off.
 - This command performs only definition of an NV bit image and does not perform printing. Printing of the NV bit image is performed by the **FS p** command.
 - NV bit image of each piece of space in NV memory is equal to the size

of the NV bit image data plus 4 bytes.

[Example] When $xL = 64$, $xH = 0$, $yL = 96$, $yH = 0$



User-defined character commands

1 ESC % n

[Name] Select /cancel user-defined character set

[Format] ASCII ESC % n
Hex 1B 25 n
Decimal 27 37 n

[Range] $0 \leq n \leq 255$

[Description] Select /cancel user-defined character set

When $n \text{ LSB} = 0$, cancel user-defined character set.

When $n \text{ LSB} = 1$, select user-defined character set.

[Note] • When user-defined character set is canceled, it selects internal character set.

• $n \text{ N}$ is only enable when $n = \text{LSB}$

[Default] $n = 0$

[Reference] ESC &, ESC ?

[Program example]

```
char SendStr[4];
SendStr[0] = 0x1B;
SendStr[1] = 0x25;
SendStr[2] = 0x01;
```

```
PrtSendData(SendStr,3);// select user-defined character set
```

2 ESC & y c1 c2 [x1 d1...d(y * x1)]...[xk d1...d(y * xk)]

[Name] Define user-defined characters

[Format] ASCII ESC & y c1 c2 [x1 d1...d(y * x1)]...[xk d1...d(y * xk)]
 Hex 1B 26 y c1 c2 [x1 d1...d(y * x1)]...[xk d1...d(y * xk)]
 Decimal 27 38 y c1 c2 [x1 d1...d(y * x1)]...[xk d1...d(y * xk)]

[Range] y = 3 font A (12 * 24)
 y = 2 font B (8 * 16)
 $32 \leq c1 \leq c2 \leq 126$
 x = 12 font A (12 * 24)
 x = 8 font B (8 * 16)
 $0 \leq d1 \dots d(y * xk) \leq 255$

[Description] Define user-defined character

y specifies the number of bytes in the vertical direction.
 define character A font y=3
 define character B font y=2

- c1 specifies beginning character code, c2 specifies end character code.
- x specifies the number of dots in the horizontal direction.

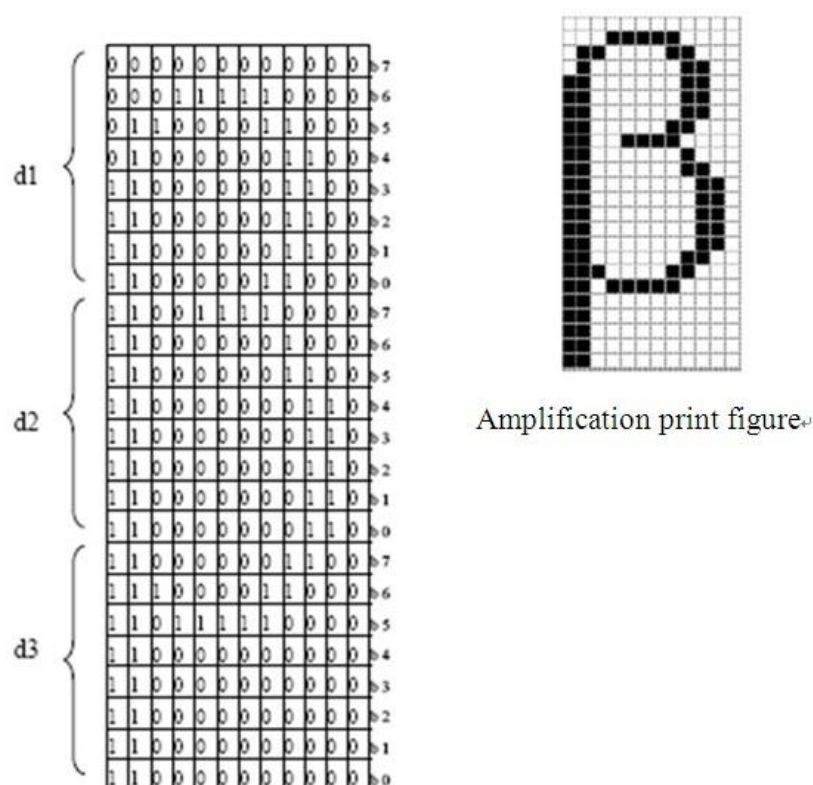
define character A font x=12,
 define character B font y=12

- $k=c2-c1+1$

[Note]

- From 0x20 to 0x7E ASCII (95 characters), the range of defined character code.
- Can be defined several continuously character code. when it need only a character, c1 should equal to c2.
- d is dot line data of character
- Define user-defined character data is (y * x) byte.
- Set 1 for print and 0 for not printed.
- This command is effect for user-defined character mode with different character definition. **ESC !** sets character font.
- User-defined character and bitmap is not more than 32K.
- User-defined characters will be cleared under following cases:
 1. Execute **ESC @**
 2. Execute **ESC ?**
 3. Printer is reset or the power is off.

[Program example] y=3, x=12 specifies character B (code 0x42)as character 12*24



```

char SendStr[42];
SendStr[0] = 0x1B;
SendStr[1] = 0x26;
SendStr[2] = 0x03; // y=3 character height 24
SendStr[3] = 0x42; // character 'B' (code 0x42) redefined
SendStr[4] = 0x42;
SendStr[5] = 0x0C; //x=12 character width 12
SendStr[6] = 0x0F; SendStr[7] = 0xFF; SendStr[8] = 0xFF; //1
SendStr[9] = 0x3F; SendStr[10] = 0xFF; SendStr[11] = 0xFF; //2
SendStr[12] = 0x20; SendStr[13] = 0x00; SendStr[14] = 0x40; //3
SendStr[15] = 0x40; SendStr[16] = 0x00; SendStr[17] = 0x20; //4
SendStr[18] = 0x40; SendStr[19] = 0x80; SendStr[20] = 0x20; //5
SendStr[21] = 0x40; SendStr[22] = 0x80; SendStr[23] = 0x20; //6
SendStr[24] = 0x40; SendStr[25] = 0x80; SendStr[26] = 0x20; //7
SendStr[27] = 0x61; SendStr[28] = 0x80; SendStr[29] = 0x60; //8
SendStr[30] = 0x3F; SendStr[31] = 0x60; SendStr[32] = 0xC0; //9
SendStr[33] = 0x1E; SendStr[34] = 0x3F; SendStr[35] = 0x80; //10
SendStr[36] = 0x00; SendStr[37] = 0x1F; SendStr[38] = 0x00; //11
SendStr[39] = 0x00; SendStr[40] = 0x00; SendStr[41] = 0x00; //12
PrtSendData(SendStr,42); // select user-deifned character set
[Default] internal character set
[Reference] ESC %, ESC ?, FS 2

```

3 ESC ?

[Name] Cancel user-defined characters

[Format]	ASCII	ESC	?	<i>n</i>
	Hex	1B	3F	<i>n</i>
	Decimal	27	63	<i>n</i>

[Range] $32 \leq n \leq 126$

[Description] Cancel user-defined characters

[Note]

- Cancel the user-defined characters defined for the character code *n*. After the user-defined characters are canceled, the internal character set is printed.
- If there is no specified character code for the user-defined character, the printer will ignore this command.

[Reference] **ESC &, ESC %**

[Program example]

```
char SendStr[4];
SendStr[0] = 0x1B;
SendStr[1] = 0x3F;
SendStr[2] = 0x42;
PrtSendData(SendStr,3);// Clear characters defined by 0x42 .
```

Kanji character commands**1 FS &**

[Name] Select Kanji character mode

[Format]	ASCII	FS	&
	Hex	1C	26
	Decimal	28	38

[Description] Selects Kanji character mode.

[Notes] • Kanji character mode is selected when the power is turned on.

[Reference] FS., FS C

[Program example]

```
char SendStr[4];
SendStr[0] = 0x1C;
SendStr[1] = 0x26;
PrtSendData(SendStr,2);// Select Kanji character mode
```

2 FS 2 c1 c2 d1...dk

[Name] Define user-defined Kanji characters

[Format]	ASCII	FS	2	<i>c1 c2 d1...dk</i>
	Hex	1C	32	<i>c1 c2 d1...dk</i>
	Decimal	28	50	<i>c1 c2 d1...dk</i>

[Range] *c1* and *c2* indicate character codes for the defined characters.

$c1 = FEH A1H \leq c2 \leq FEH$

$0 \leq d \leq 255$

$k = 72$

[Description] Defines user-defined Kanji characters for the character codes specified by

c1 and c2.

- [Note]
- c1 and c2 indicate character codes for the defined characters. c1 specifies for the first byte, and c2 for the second byte.
 - d indicates the dot data. Set a corresponding bit to 1 to print a dot or to 0 to not print a dot.

Define data format is the same as ESC & command .

- When the user selects the font A, the definition should be 24 characters dot matrix characters, select font B, the definition should be 16 characters dot matrix characters

[Default] All spaces.

[Reference] **ESC &**

SendStr[0] = 0x1C;

SendStr[1] = 0x2E;

PrtSendData(SendStr,2); //Turn Kanji characters mode off

A Bar code

UPC-A: UPC-A should accord with UCC standard (<http://www.uccnet.org>)

UPC-E: UPC-E should accord with UCC standard (<http://www.uccnet.org>)

ENA8: ENA8 should accord with EAN standard (<http://www.ean-int.org>)

ENA13: ENA13 should accord with EAN standard (<http://www.ean-int.org>)

CODE39: also called 39 code, CODE39start and stop bit character should be '*', among start and stop bit should not include character'*' .

ITF: also called INTERLEAVED 25, INTERLEAVED 2 of 5, data bit only can be even

CODABAR: also called Codebar, start and stop bit should be among A,B,C,D.

CODE93: CODE93 start and stop bit character should be '*', and among start and stop bit should not included '*',

A.1 Bar code length ASCII

Barcode system	Length	ASCII f
UPC-A	12	0~9
UPC-E	8	0~9
EAN8	8	0~9
EAN13	13	0~9
CODE39	No limit	0~9 A~Z - . SP \$ / + % *
INTERLEAVED 25	even number	0~9
CODABAR	No limit	0~9 - : / % . A~D
CODE93	No limit	0~9 A~Z - . SP \$ / + % *

B. Pre-printing specifications

If user wants to locate receipt by detecting mark preprinted, testing mark should meet the testing print regulation. Otherwise it may cause printer can't recognize the detecting mark.

Printing position: Testing mark should be printed on right margin of thermo sensitive side,

Width range: width $\geq 7\text{mm}$

Height range: $4\text{mm} \leq \text{height} \leq 6\text{mm}$

The reflection rate to IR is less than <10% (reflection rate to the detecting mark on paper >65%)

Hps indicates the distance from down edge of detecting mark from upper edge.

$0\text{mm} \leq \text{Hps} \leq 1\text{mm}$

