
MPT-II



Portable thermal printer

Developing manual (V1.0)

1. Product introduction	1
2. Communication Interface	2
2.1 Infrared ray: IR interface	2
2.1.1 RAW-IR	3
2.1.2 VIR (Developing)	4
2.1.3 IRDA IrCOMM	4
2.2 Bluetooth interface	4
2.2.1 Matched pair	5
2.2.2 Printing through Bluetooth interface	5
2.3 RS232 interface	5
2.4 WiFi wireless communication (developing)	7
4. Printing Control Commands	8
4.1 Basic control commands	9
4.1.1 ESC @	9
4.1.2 FF	10
4.1.3 LF	10
4.1.4 CR	10
4.1.5 ESC J n	11
4.1.6 ESC d n	11
4.1.7 HT	11
4.2 Character parament setting commands	12
4.2.1 ESC ! n	12
4.2.2 GS ! n	13
4.2.3 ESC - n	14
4.2.4 ESC E n	14
4.2.5 ESC G n	15
4.2.6 GS B n	15
4.2.7 ESC V n	16
4.3 Print typeset parameter setting commands	16
4.3.1 ESC \$ nL nH	16
4.3.2 ESC D n1 n2...nk NULL	17
4.3.3 ESC 2	18
4.3.4 ESC 3 n	18
4.3.5 ESC SP n	19
4.3.6 ESC a n	19
4.3.7 GS L nL nH	20
4.4 Graphics/image print commands	20
4.4.1 ESC * m nL nH d1...dk	20
4.4.2 GS * x y d1...dk	25
4.4.3 GS / n	27
4.5 Barcode print commands	27
4.5.1 GS h n	27
4.5.2 GS w n	28
4.5.3 GS k	28
4.5.4 GS H n	29
4.6 user-defined character commands	31

4.6.1 ESC % n	31
4.6.2 ESC & y c1 c2 [x1 d1...d(y * x1)]...[xk d1...d(y * xk)]	32
4.6.3 ESC ?	34
B.1 Barcode	34
B.2 Barcode length character set	34
C.Pre-printing black mark specifications	35

1.Product introduction

MPT-II series portable thermal printer is based on the platform of ARM, it having ascendant function, support infrared ray (IR), blue-tooth, WIFI wireless communication that is applicable for the receipt printer, barcode labels printer. MPT-III can be used in the government affair, police affair, the logistic for tobacco and post , transportation, and so on.

There are three models for MPT-II serials : MPT-II/IR,MPT-II/BL, MPT-II/WIFI

MPT-II/IR: support infrared ray(IR)

MPT-II/BL: support bluetooth

MPT-II/WIFI: support WIFI wireless communication

Specification:

Model	MPT-II/IR	MPT-II/BL	MPT-II/WIFI
Print method	Line thermal printer		
Paper width	58mm		
print width	48mm		
Weight (g)	210		
Diameter of paper roller	≤30mm		
Dot per lie	384 dots/line		
internal characters	ASCII (16×8、24×12)、GBK(16×16、24×24)		
Print speed	50~70mm/s		
Life cycle time	50km(printing density under 12.5%)/100 million pulses		
text and graphic Support	English, figure, symbol, Chinese, graph, curve, icon pre-stored, barcode (CODE39、EAN13、EAN8、 CODABAR、 CODE128、 CODE93、 ITF)		
Dot pitch (resolution)	203DPI/8 dots/mm		
interface	IRDA/VIR/RS232	Bluetooth V1.1 CLASS /RS232	WIFI/RS232
Power	1500mAh 7.4V rechargeable Lithium battery		
Recharge adapter	Stand by recharging / AC adapter		
Black mark test	Supported		
Paper feeding	Easy paper loading		
commands	ESC/POS compatible with order sets		
Paper cut	By hands		
operate setting	Temperature -10°C~50°C ; Humidity 20%~85%		
Recharge Setting	Temperature 5°C~40°C; Humidity 20%~85%		
Storage Setting	Temperature -20°C~70°C ; Humidity 5%~95%		
Standard accessories	Long-term preservable paper, AC adapter, li-ion battery		
Optional accessories	Serial interface cable / leather coat		

2. Communication Interface

MPT-II portable thermal printer has 4 interfaces that can be connect with the host for transmitting the data. That is IrDA(IrCOMM protocols) , RAWIR, VIR Bluetooth, and RS232 serial interface. Thereinto, Bluetooth is wireless data communication interface; IrDA is compatible with the international IrCOMM protocols established by the international IR data communication association. RAW-IR is compatible with the physics criterion issued by the international IrCOMM protocols. VIR is based on the RAW-IR and integrate the VIR protocol. RS232 is lineate-asynchronism series port .

MPT—II supports one or more interfaces and please confirm the host device whether it at least support one of the above interfaces.

2.1 Infrared ray: IR interface

Infrared Ray is the wireless data communication technology with infrared light as communication data carrier that explored and developed by the IrDA association.

Infrared ray interface as the wireless data transmitting port, because of its having lower power, manure technology, easy using, is widely applicable to portable devices, such as all the palm computer with WINCE, laptop, mobile cell, palm computer, data collector, measure device and etc.

MPT-II , its hardware meets the regulations of IrDA1.1 physical layer.

Because IR interface depend on the infrared light as data carrier, there can not having any remora between the data sender and the receiver device; their infrared port should aim at and close each other.

When MPT-II/IR communicates with the host devices that hardware also meets the regulation of IrDA 1.1 physical layer, make the IR interfaces (from both MPT-II and the host device) right towards to each other (angle shall not be more than 30°), without any obstacles between them, the distance shall be less than 0.5M.

MPT-II/IR can use RAW-IR to communicates with the host device,also can adopt the Vir protocol or IrCOMM protocol (IrDA).

When using RAW-IR to communicates with the host device, IR transceiver just change the serial interface data signal into IR signal or change light signal into serial interface data signal according to the certain coding regulation. Under this situation, this IR interface is called RAW-IR. When you use RAW-IR, it is not compatible with IrDA because of the software do not use IrDA protocol during sending and receiving data.

VIR is safer and more reliable IR communication way, that is, it adds VIR protocol based on RAW-IR. In case the host device can not support IrCOMM protocol, advise to use VIR protocol.

Compared with VIR protocol, IrCOMM protocol is based on the hardware of RAW-IR ,also IrCOMM is more popular way which is made by international IR Communication association . IrCOMM is one of subset of IrDA protocol that is wireless data communication protocol developed by IrDA association, based on the hardware of RAW-IR for ensuring data transmission's stability , reliability, easy using. All portable devices supporting IrDA protocol inn (such as wince, pocket pc, palm, IR cell phones etc.)

can support IrCOMM. As if the host device can support IrDA protocol inn, who drive the printer to use the operation system (such as WINCE , PALM OS). So that , for developer, IrCOMM interface is the virtual serial port. As if the host device has no IrDA protocol inn and wants to realize IrCOMM data transmission, the developer has to write IrDA protocol inn. Whether the host device can support IrCOMM, please refer to the developing documents of the host device or make a consultation with the supplier of the host device.

Due to the RAW-IR interface ,VIR interface and IrCOMM interface of MPT-II printer use same hardware source, so they can't work at the same time. The default mode of MPT-II is IrCOMM. If you want to change the current IR interface mode, please use the special tool of MPT-II——CONFIG modification setting. Pls refer to the details 【Chapter-3: MPT-II printer tool software】

As to the detailed IrDA protocol, please refer to IrDA official website: <http://www.irda.org/>

2.1.1 RAW-IR

RAW-IR interface is that the IR transceiver is directly subjoined with the input and output port of RS232 serial interface. So the IR transceiver just simply put the asynchronism serial data signal of RS232 change to the IR light signal or light signal change to serial data signal as per the certain coding regulation. As to the RAW-IR interface , the developer for host application can operate the RAW-IR of the host as per the operation of RS232 interface, but the small different.

Before using the RAW-IR interface of the host device, must to know the port code of the RAW-IR, the User can learn RAW-IR interface port code from the developing manual of the host device and its supplier. As to the host device that partially can use RAW-IR interface, the way of opening RAW-IR is different from the way of opening serial port, please refer to the developing manual of the host device.

RAW-IR interface is a little different from standard RS232 serial interface. RAW-IR interface adopt the way of wireless infrared connection , so that the serial TXD signal and RXD signal is valid, and other operation on other RS232 pin of the host device have no effect to RAW-IR interface.

Activate the printer through RAW-IR interface when the MPT-II is under the dormancy condition

The baud rate of mobile thermal printer MPT-II/IR is: 9600bps, 19200bps, 38400bps, 57600bps, 115200bps. Default baud rate of RAW-IR of the printer is 9600bps. If user wants to change baud rate, please refer to MPT-II_CONFIG modification setting. Please refer to the details 【Chapter-3: MPT-II printer tool software】

Remark: Not all the host with RAW-IR can work under these baud rates, before change its baud rate, pls make sure the RAW-IR of your current host device can work under this baud rate. When programming the host device ,the Setting of RAW-IR interface as follow:

Data Bit : 8 phase

Stop bit: 1 phase

Even-odd check: No

Flow control: No

Compared with other general IR source, RAW-IR has enough ability of anti-disturbance but can not close to the IR sources from these IR devices (such as laptop, cell phone with IR). And the printer can not shield these disturbances.

2.1.2 VIR (Developing)

For MPT-II doing better communication with a good many hand terminals without IrDA protocol inn, so develop the VIR protocols on the base of printing data communication protocol that meets the regulation of IrDA physical layer.

Wake up through VIR protocol when the MPT-II is under the dormancy condition.

For IR communication device without integration of IrDA protocol inn, VIR is the good protocol instead of IrDA protocol with low power loss, easy realization and avoiding messy code during data transmission. With regards to details of VIR definition and how to realize VIR, please refer to addenda D. if you need any technical support, please contact to the supplier.

2.1.3 IRDA IrCOMM

There are many IrDA protocols, and IrCOMM is one of IR communication protocols recommended by IrDA Association.

When using IrCOMM protocols, the printer can totally avoid the interference from other IR source.

MPT-II supports the IrCOMM protocol.

Although regard IrCOMM as virtual serial interface, and there is no need to set baud rate, the real baud rate of data transmission will adapt automatically when IrCOMM is working . IrCOMM is responsible for verification and data buffer etc, so there is no sense for other IrCOMM serial interface setting.

User should know COM port number and how to open it. User can learn this info from the host device manual or its suppliers.

Wake the printer up through IrCOMM interface.

If users want to develop the IrCOMM protocols, please refer to IrDA official website: <http://www.irda.org/>

2.2 Bluetooth interface

Bluetooth is one of wireless technology and support short distance communication (within 10m). It can transfer the data among many devices, such as mobile phone, PDA, printer, laptop, wireless earphone etc. Bluetooth standard is IEEE802, working at 2.4 GHz, bandwidth 1Mb/s. For more details of Bluetooth, please refer to Bluetooth official website: <http://www.bluetooth.org>.

MPT-II/BL support wireless data transmission interface, meet Bluetooth 1.1 regulations, power class 2.

MPT-II/BL is only sub-device, and user need to drive the printer by Bluetooth host devices (such as PDA, mobile phone, laptop).and other Bluetooth sub-device(such as bluetooth earphone) can't drive the printer to print by Bluetooth.

MPT-II/BL default device Name is MPT-III/BL. User can change the device Name according to your requirements. Please refer to 【Chapter 3: MPT-II printing tools software】 .

The default password of Bluetooth connecting is PRT. User can change the password according to requirements. Please refer to 【Chapter 3: MPT-II printing tools software】 .

2.2.1 Matched pair

Before printing, MPT—II/BL need to pair with the host devices that drives the mobile thermal printer. Pairing process is originated by the host devices.

Usual Pairing processes as follow:

1. Turn on the printer
2. The host device search exterior Bluetooth device
3. If there are some exterior Bluetooth devices, please select **SPB**
4. Enter password “**0000**”.
5. Finish pairing.

Detailed pairing methods, please refer to the Bluetooth function introduction of the host devices

When pairing, mobile thermal printer MPT-III must be turn on.

Note: when pairing, do not turn on many printers at the same time , otherwise it can't judge which one succeeds in pairing.

After finished pairing, other the host device (upper machine) still can pair with this mobile thermal printer. one printer can be most allowed to pair with 8 units upper machines (the host devices) . If the ninth device will be paired with the printer, the earliest ones will be deleted from printer paring list automatically. If this earliest device will be allowed to drive the printer again, it need to pair with the printer again.

2.2.2 Printing through Bluetooth interface

After pairing, the upper machine with virtual Bluetooth interface (such as smart phone , pocket PC, palm, laptop) can drive mobile thermal printer MPT-II/BL to print through such virtual Bluetooth interface. If there is no virtual Bluetooth interface on upper machine, please make a consultation with the suppliers of upper machine.

2.3 RS232 interface

RS232 is common data communication interface. All models of MPT-II printer have the RS232 serial interface.

The RS232 interface specification for MPT-II printer:

Date transmission: serial

Synchronous mode: asynchronism

Hand-shaking signal: None

Flow control: hardware flow control / software flow control/ no optional

Baud rate(option): 9600bps、19200bps、38400bps、57600bps、115200bps.

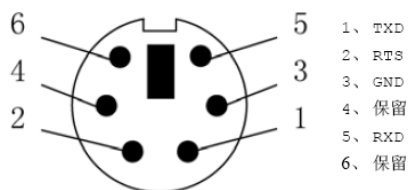
Date byte : 8 phase

Verification mode: NO

Stop bit: 1 phase

Socket pin definition (printer): 6 pin mini-din, as described in the following picture:

1. TXD
2. RTS
3. DND
4. Reserved pins
5. RXD
6. Reserved pins



TXD:the printer send data

RTS: The printer request data (when the printer is setted as the hardware flow control mode, RTS take effect, otherwise it is inutility, about the setting way for the hardware flow control mode, please refer to 【Chapter 3: MPT-II printer tool software】

GND: LAND

RXD:The Printer receive the data

Note: Reserved pins have other usages. Please do not use the reserved pins if you want to make the connection by yourself, otherwise the printer can't work well or even cause loss to printer.

Buffer- area capacity of printer is about 3K bytes. When printing data is less than 3K byte, there is no need to use flow control, connection method as follow:

Printer	Host devices
RXD	TXD
GND	GND

When print date is more than 3K byte, need use the flow control. When connection through Bluetooth and IrCOMM, there is no need to consider the flow control. When using VIR protocols, users need the flow control mode to avoid the overflow from data buffer, according to VIR protocols. Not use flow control in RAW-IR.

When printer is on hardware flow control, connection between host device and printer as follow:

Printer	Host device
TXD	RXD
RTS	CTS

RXD	TXD
GND	GND

During communication, The host device can monitor the signal intensity of CTS. When the signal intensity is strong, the host device can send data to printer. When the signal intensity is low, it means the printer is busy and need to stop sending data. Until the signal intensity is strong, start sending data again.

When use software flow control with XON/XOFF method, the connection method as follow:

Printer	Host devices
TXD	RXD
RXD	TXD
GND	GND

When using flow control, the host devices need to judge whether the print buffer is full by check the RXD data of itself. Detailed method as follow: Starting printing, the host device send the data to the printer, and monitor data receiving of serial interface at the same time. When receiving XOFF (0x13), stop sending data to the printer; When receiving XON(0x11), send the data to printer again. It recycles until finishing sending the printing data.

2.4 WiFi wireless communication (developing)

4. Printing Control Commands

MPT-II printer control adopt the ESC/POS compatible commands. Send the printing control commands and data to the printer through communication interface for controlling the printer

The printing commands list as below , classify as per the function

Chapter	Command	Description
Printer control command		
4.1.1	<u>ESC @</u>	Initialize printer
4.1.2	<u>FF</u>	Print and return to standard mode (in page mode)
4.1.3	<u>LF</u>	Print and line feed
4.1.4	<u>CR</u>	Print and carriage return
4.1.5	<u>ESC J n</u>	Print and feed paper by n unit vertical motion
4.1.6	<u>ESC d n</u>	Print and feed n lines
4.1.7	<u>HT</u>	Horizontal tab
4.1.8	<u>FS U nL nH...</u>	Print Unicode character
Character parameter setting command		
4.2.1	<u>ESC ! n</u>	Select print mode(s)
4.2.2	<u>GS ! n</u>	Select character size
4.2.3	<u>ESC - n</u>	Turn underline mode on/off
4.2.4	<u>ESC E n</u>	Turn emphasized mode on/off
4.2.5	<u>ESC G n</u>	Turn double-strike mode on/off (ESC E)
4.2.6	<u>GS B n</u>	Turn white/black reverse printing mode on/off
4.2.7	<u>ESC V n</u>	Turn 90 ° clockwise rotation mode on/off
Print typeset parameter setting command		
4.3.1	<u>ESC \$ nL nH</u>	Set absolute print position
4.3.2	<u>ESC D n1 n2...nk</u> <u>NULL</u>	Set horizontal tab position
4.3.3	<u>ESC \ nL nH</u>	Set the relative printing position
4.3.4	<u>ESC 2</u>	Select default line spacing
4.3.5	<u>ESC 3 n</u>	Set line spacing
4.3.6	<u>ESC SP n</u>	Set character spacing
4.3.7	<u>ESC a n</u>	Select justification
Graph/image image print command		
4.4.1	<u>ESC * m nL nH</u> <u>d1...dk</u>	Select bit-image mode

4.4.2	<u>GS * x y d1...dk</u>	Define downloaded bit image
4.4.3	<u>GS / n</u>	Print downloaded bit image
Bar code command		
4.5.1	<u>GS h n</u>	Set bar code height
4.5.2	<u>GS w n</u>	Set bar code width
4.5.3	<u>GS H n</u>	Select printing position of HRI characters
4.5.4	<u>GS k</u>	Print bar code
User-defined character command		
4.6.1	<u>ESC % n</u>	Select/cancel user-defined character set
4.6.2	<u>ESC & y c1 c2 ...</u>	Define user-defined characters
4.6.3	<u>ESC ?</u>	Cancel user-defined characters

This Chapter detailedly introduce the printing command of printer, The format show of description as below:

【COMMAND】 + **【parameter】**

【COMMAND】 is command part, composed by ESC and command character, there is a little of single byte command have not ESC.

【parameter】 is Parameter part, showed by italic, the parameter is not number character, is the value of character

All the example was writed by C language, thereinto, the PrtSendData is dummy function, User can write it as per the fact condition of host device. The define of the function as below :

```
PrtSendData(char *buf, int len)
```

Description: send data to the printer

char *buf: the finger of the data to be printed

int len: data length . units:byte。

4.1 Basic control commands

4.1.1 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);

4.1.2 FF

[Name] Print and return to standard mode (in page mode)

[Format] ASCII FF
 Hex OC
 Decimal 12

[Description] FF prints the data in the printer buffer collectively and returns to standard mode.

[Note]

- The buffer data is deleted after being printed.
- This command sets the print position to the beginning of the line.
- After printing data in the print buffer, the printer will stop when meeting the dark mark if there is dark marks on paper. If no dark mark on the paper, it will stop after feeding 0.5m paper. Please refer to the Addenda C preprint dark-mark regulation

[Program example] char SendStr[2];
SendStr[0] = 0x0C;
PrtSendData(SendStr, 1);

4.1.3 LF

[Name] Print and line feed

[Format] ASCII LF
 Hex 0A
 Decimal 10

[Description] 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]='\\n'; //C language '\\n' feed line
PrtSendData(SendStr, 1);

4.1.4 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]

[Reference] **LF**

4.1.5 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] = 'J';
SendStr[2] = 8;
PrtSendData(SendStr, 3);// feeding1mm

4.1.6 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] = 'd';
SendStr[2] = 2;
PrtSendData(SendStr, 3);//feeding 2 lines

4.1.7 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 the position of the horizontal tab by ESC D command
- 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);
```

4.2 Character parament setting commands

4.2.1 ESC ! *n*

[Name] Select print mode

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

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

[Description] Select printing mode by the value of parameter *n*

Parameter *n* defined as follow:

Bit	Value	Name
0	—	Undefined
1	—	Undefined
2	—	Undefined
3	0	Bold mode not selected
	1	Body 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 bold 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] = '!';

SendStr[2] = 0x28;// 00101000 double-width bold mode

PrtSendData(SendStr, 3);

4.2.2 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 !**

[Program example] char SendStr[4];

SendStr[0] = 0x1D;

SendStr[1] = '!';

```
SendStr[2] = 0x01;// 00000001 double height
PrtSendData( SendStr, 3);
```

4.2.3 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)

[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[3];

```
SendStr[1] = ' ';
SendStr[2] = 1;// uniline underline
PrtSendData( SendStr, 3);
```

4.2.4 ESC E *n*

[Name] Turn bold mode on / off

[Format]	ASCII	ESC	E	<i>n</i>
	Hex	1B	45	<i>n</i>
	Decimal	27	69	<i>n</i>

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

[Description] Turn bold mode on/ off

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

When LSB is 0, bold mode is turned off.

[Note]

- LSB of *n* is allowed to be used.
- ESC ! turns bold 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] = 'E';
SendStr[2] = 1;// bold mode
PrtSendData(SendStr,3);

4.2.5 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]

- LSB of *n* is allowed to be used.
- Output of printer appear the same in Double –strike mode and bold mode.

[Default] *n* = 0

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

[Program example] char SendStr[3];

```
SendStr[0] = 0x1B;  
SendStr[0] = 0x1B;  
SendStr[1] = '-';  
SendStr[2] = 1;// single underline  
PrtSendData( SendStr, 3);
```

4.2.6 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 on;

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

[Note]

- 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] $n = 0$

[Example] char SendStr[3];

SendStr[0] = 0x1D;

SendStr[1] = 'B';

SendStr[2] = 1;// white/black reverse printing

PrtSendData(SendStr, 3);

4.2.7 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] = 'V';

SendStr[2] = 2;// 180°rotation

PrtSendData(SendStr, 3);

4.3 Print typeset parameter setting commands

4.3.1 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.

$N = nL + nH \leq 256$

[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] = '\$';

SendStr[2] = 24; //3×8=24

PrtSendData(SendStr, 3); //Set the absolute position 3 MM from the left margin (24 horizontal motion unit)

PrtSendData ("Start printing from left margin 3 mm \n, 22")

4.3.2 ESC D n_1 $n_2 \dots n_k$ NULL

[Name] Set horizontal tab positions

[Format]	ASCII	ESC	D	$n_1 \dots n_k$ NULL
	Hex	1B	44	$n_1 \dots n_k$ 00
	Decimal	27	68	$n_1 \dots n_k$ 0

[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 ×) and character spacing 8 (list9,

17,25,.....). 24

[Reference] **HT**

```
[Program example] char SendStr[16];
char NextPos = 9;
SendStr[0] = 0x1B;
SendStr[1] = 'D';
SendStr[2] = 11;// 10 charcter spacing from first list
SendStr[3] = 17;// 16 charcter spacing from first list
SendStr[4] = 23;// 22 charcter spacing from first list
SendStr[5] = 29;// 28 charcter spacing from first list
SendStr[6] = 0; // ending
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);
```

4.3.3 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] = '2';
PrtSendData(SendStr,2);
```

4.3.4 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] = '3';

SendStr[2] = 16;

PrtSendData(SendStr,3);// set the line spacing to vertical motion 16 units (2mm).

4.3.5 ESC SP n

[Name] Set character spacing

[Format]	ASCII	ESC	SP	n
	Hex	1B	20	n
	Decimal	27	32	n

[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;

4.3.6 ESC a n

[Name] Select justification

[Format]	ASCII	ESC	a	n
	Hex	1B	61	n
	Decimal	27	97	n

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

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

n value and definition

n	Definition
0	Flush left
1	Center
2	Flush right

[Note]

- This command is enable only when processed 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] = 'a';

SendStr[2] = 1;

PrtSendData(SendStr,3);// Set horizontal justification to center.

4.3.7 GS L nL nH

[Name] Set left margin

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

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

[Description] Set left margin to N horizontal dot pitch. nL and nH separately is low-order byte and nH is high-order byte of the integer with no mark and double bytes. $N=nL+nH*256$, left margin is the left distance and the width of the 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] = 0x1B;

SendStr[1] = 'L';

SendStr[2] = 16;

SendStr[3] = 0;

PrtSendData(SendStr,4);// Set left margin to 16 horizontal motion units (2mm).

4.4 Graphics/image print commands

4.4.1 ESC * m nL nH d1...dk

[Name] 打印黑白位图 Select white and black bitmap mode

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

[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. as the following :

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

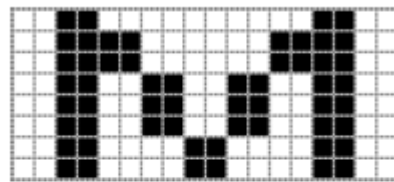
nL nH refer to the low – order and high-order byte of the integer with no mark and double bytes , means the dots in the horizontal direction in bitmap. Max value of single width is 384 and max value of double width is 192.

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 2kth 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



打印放大图

Print Image as the above , procession code as follow:

```

char SendStr[16];
SendStr[0] = 0x1B;
SendStr[1] = '*';
SendStr[2] = 0;//m=0( height 8 dots, double width)
SendStr[3] = 9;// ( image width 9 dots)
SendStr[4] = 0;
SendStr[5] = 0;// (image dot line bitmap)
SendStr[6] = 0xFF;
SendStr[7] = 0x60;

```

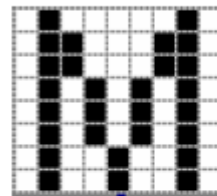
```

SendStr[8] = 0x1C;
SendStr[9] = 0x03;
SendStr[10] = 0x1C;
SendStr[11] = 0x60;
SendStr[12] = 0xFF;
SendStr[13] = 0;
PrtSendData(SendStr,14);// print image

```

Example2: m=1 (8 dots, single width) d1 indicates the first list data, dk indicates the k th 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



打印放大图

Print Image as the above, procession code as follow:

```

char SendStr[16];
SendStr[0] = 0x1B;
SendStr[1] = '*';
SendStr[2] = 1;//m=1(height 8 dots, no enlargement )
SendStr[3] = 9;// image width 9 dots
SendStr[4] = 0;
SendStr[5] = 0;// image dot line bitmap
SendStr[6] = 0xFF;
SendStr[7] = 0x60;
SendStr[8] = 0x1C;
SendStr[9] = 0x03;

```

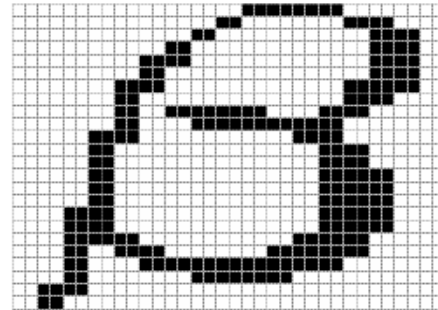
```

SendStr[10] = 0x1C;
SendStr[11] = 0x60;
SendStr[12] = 0xFF;
SendStr[13] = 0;
PrtSendData(SendStr,14);// print image

```

Example3: m=32 (24 dots, double width) d1,d2,d3 indicates the 1st ,2nd row dot data, and the like , bn indicates the n th place of byte

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



打印放大图

Print Image as the above, proccession code as follow:

```

char SendStr[64];
SendStr[0] = 0x1B;
SendStr[1] = '*';
SendStr[2] = 32;//m=32( Height 24 dots, double width)
SendStr[3] = 17;//image width 17 dots
SendStr[4] = 0;
// 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

```

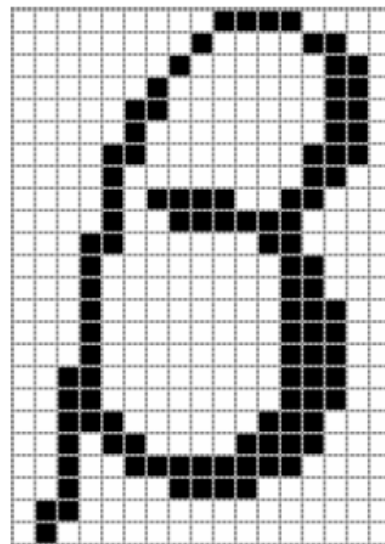
```

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 indicates the nth of (bn, n=1, 2, 3.....; nth, 1st, 2nd, 3rd, 4th)

	d4	d7		d49											
d1	0	0	0	0	0	0	0	1	1	1	0	0	0	b7	
	0	0	0	0	0	0	0	1	0	0	0	1	1	0	b6
	0	0	0	0	0	0	1	0	0	0	0	0	1	1	b5
	0	0	0	0	0	1	0	0	0	0	0	0	1	1	b4
	0	0	0	0	1	0	0	0	0	0	0	0	1	1	b3
	0	0	0	0	1	0	0	0	0	0	0	0	1	1	b2
	0	0	0	1	1	0	0	0	0	0	0	1	1	1	b1
	0	0	0	1	0	1	0	0	0	0	0	1	1	1	b0
d2	0	0	0	1	0	0	1	1	1	0	0	1	1	0	b7
	0	0	0	1	0	0	1	1	1	1	1	0	0	0	b6
	0	0	0	1	1	0	0	0	0	0	1	1	0	0	b5
	0	0	0	1	0	0	0	0	0	0	1	1	0	0	b4
	0	0	0	1	0	0	0	0	0	0	1	1	0	0	b3
	0	0	0	1	0	0	0	0	0	0	1	1	1	0	b2
	0	0	0	1	0	0	0	0	0	0	1	1	1	0	b1
	0	0	0	1	0	0	0	0	0	0	1	1	1	0	b0
d3	0	0	1	1	0	0	0	0	0	0	1	1	1	0	b7
	0	0	1	1	0	0	0	0	0	0	1	1	1	0	b6
	0	0	1	1	1	0	0	0	0	0	1	1	1	0	b5
	0	0	1	0	1	1	0	0	0	1	1	1	1	0	b4
	0	0	1	0	0	1	1	1	1	1	1	0	0	0	b3
	0	0	1	0	0	0	1	1	1	1	0	0	0	0	b2
	0	1	1	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	b0



打印放大图

If print the above Image show, procession code as the following:

```

char SendStr[64];
SendStr[0] = 0x1B;
SendStr[1] = '*';
SendStr[2] = 33;// m=33 ( height 24 dots, no enlargement)
SendStr[3] = 17;// image width is 17 dots
SendStr[4] = 0;
// 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 (bold mode, double strike mode, underline, character size, or white/black reverse printing mode) not effective for this command.

4.4.2 GS * x y d1...dk

[Name] Define downloaded bit image

[Format]	ASCII	GS	*	x y d1...dk
	Hex	1D	2A	x y d1...dk
	Decimal	29	42	x y d1...dk

[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.

- $x*8$ dots in the horizontal direction
- $y*8$ dots in the vertical direction

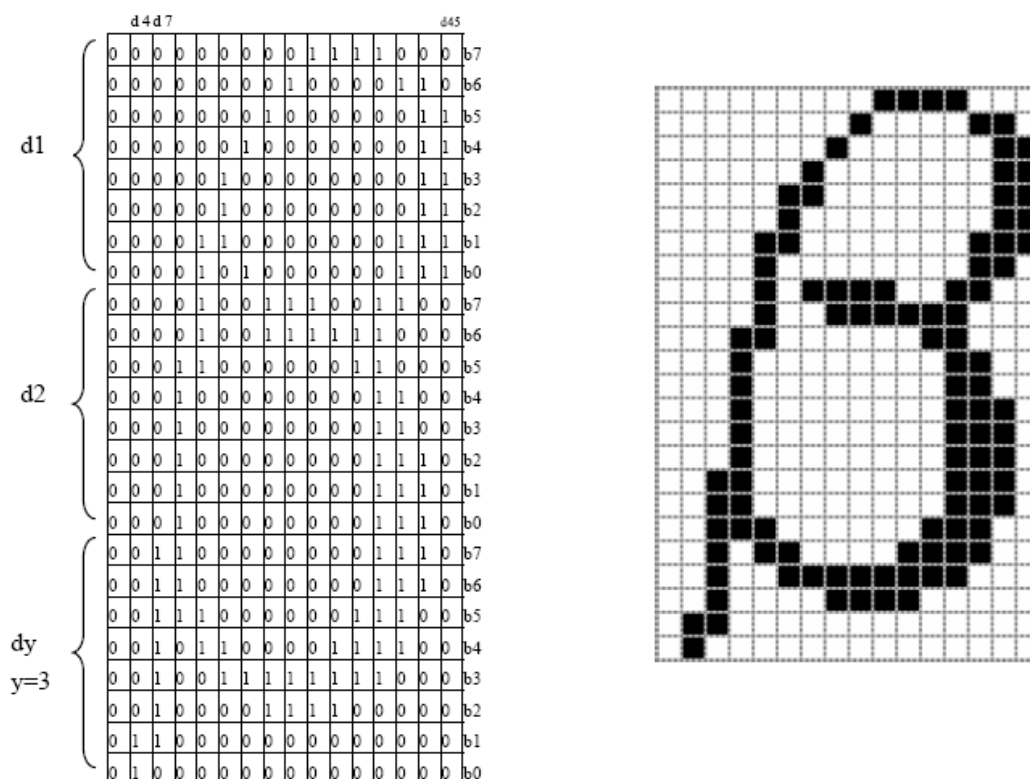
[Note]

- If $x * y$ exceeds specified range, this command is prohibited.
- d indicates bit image data. Data d set printing o, 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 $x=2, y=3, d_1, \dots, d_k$, as following left picture, bitmap right picture



If print the above Image show, procession code as the above:

```

char SendStr[64];
SendStr[0] = 0x1D;
SendStr[1] = '*';
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

```

4.4.3 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.

n The definition of n as the following :

n	Enlargement mode
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 (bold 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 GSL 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 4.4.2 GS program example

4.5 Barcode print commands

4.5.1 GS h n

[Name] Set bar code height

[Format] ASCII GS h n
 Hex 1D 68 n
 Decimal 29 104 n

[Range] $1 \leq n \leq 40$

[Description] Select the height of a bar code.
n specified the number of dots in the vertical direction.
 [Note] •If $n > 40$, bar code height will be set 40.
 [Default] $n = 36$
 [Reference] **GS k**
 [Program example] Please refer to 4.5.5 GS k example.

4.5.2 GS w *n*

[Name] Set bar code width
 [Format] ASCII GS w *n*
 Hex 1D 77 *n*
 Decimal 29 119 *n*

[Range] $1 \leq n \leq 4$
 [Description] Select the width of a bar code.

n definition as the following :

N	Module width for multilevel bar code (mm)	Binary level bar code	
		Thin element width (mm)	Thick element width (mm)
1	0.125	0.125	0.25
2	0.25	0.25	0.50
3	0.375	0.375	0.75
4	0.50	0.50	1.0

- Multilevel bar code as follows:
 UPC-A, UPC-E, EAN13, EAN8, CODE93
- Binary level bar code as follows;
 CODE39, CODABAR

[Default] $n = 2$
 [Reference] **GS k**
 [Program example] Please refer to 4.5.5 GS k program example

4.5.3 GS H *n*

[Name] Select printing position of HRI characters
 [Format] ASCII GS H *n*
 Hex 1D 48 *n*
 Decimal 29 72 *n*

[Range] $0 \leq n \leq 2$
 [Description] Select the printing position for HRI characters when printing a bar code.



n select printing position, as the following :

<i>n</i>	select printing position
0	not printed
1	above the bar code
2	below the bar code

[Note]

- HRI characters are printed using the font specified by **GS f** .
- Print mode (bold mode, double strike mode, underline, character size, or white/black reverse printing mode etc.) is not effect for this command.

[Default] *n* = 0

[Reference] **GS f**, **GS k**

[Program example] Please refer to 4.5.5. GS k program example

4.5.4 GS k

[Name] print bar code

[Format] there having two formats in this command:

Format1 ($0 \leq m \leq 8$)

ASCII GS k *m d1...dk NUL*

Hex 1D 6B *m d1...dk 00*

Decimal 29 107 *m d1...dk 0*

Format 2($65 \leq m \leq 73$)

ASCII GS k *m n d1...dn*

Hex 1D 6B *m n d1...dn*

Decimal 29 107 *m n d1...dn*

[Range] $0 \leq m \leq 8$ (k and d specified by bar code system)

$65 \leq m \leq 73$ (n and d specified by bar code system)

n indicates the number of the bar code data

[Description] Select a bar code system and print the bar code.

m specified as a bar code system as follows:

<i>m</i>	Bar code system
0, 65	UPC-A
1, 66	UPC-E
2, 67	EAN8
3, 68	EAN13
4, 69	CODE39
5, 70	INTERLEAVED 25(ITF)

6, 71	CODABAR
7, 72	CODE93
8, 73	CODE128

[Note]

- when using format 1 order, if defined the data length of barcode in the barcode types, so the value of K(the barcode data length received by printer) should is equal to the defined the data length, otherwise , the command is not valid . please refer to the data length of barcode as 【appendix B】

- The barcode data characters received by the printer should include into the characters set defined by the barcode type. If the character of barcode data characters is out of the character set, the command is not valid. About the character sets of barcode , please refer to the appendix A.

- when using format 2 order, if defined the data length of barcode in the barcode types, so the value of n should is equal to the defined the data length, otherwise , the command is not valid . please refer to the data length of barcode as 【appendix B】

- The data length of the code of INTERLEAVED 25(ITF) is even number, the value of K(the barcode length received by the printer) is even number, if using format 1 to print ITF barcode. If it is odd number, the lasted one byte data will be neglected. If using format 2 to print ITF barcode, so that the value of n is even number, otherwise, the lasted one byte data will be neglected, if it is odd number.

- If horizontal direction size exceeds print area, the exceeding part will be ignored.

- Print mode (bold mode, double strike mode, underline, character size, or white/black reverse printing mode etc.) is not effect for this command.

- Print barcode should meet the barcode regulation , otherwise it may cause no scanning.

- The printer don't calculate the check code, if the barcode need the check code, the barcode data must include the check code, the printer can not check whether the check code is correct or not , if the user calculate wrong the check code, it will can not scan the barcode.

- If the reading character be set to print, the invisible character of CODE93 and CODE128 can not be printed, here can be replaced with '□'

- CODE39 not include the extend of CODE39 (EXTERN CODE 39)

- CODE93 not include the extend of CODE93 (EXTERN CODE 93)

- The head of the barcode data string of CODE128 must be the selected character in coding set(CODE A, CODE B, or CODE C), also can shift coding set in the inner of one barcode. Using character '{' and one character to compose for defining the special function through sending two time '{'define ASCII character '{'. Please refer to the following picture:

ASCII	HEX	Function
{A	7B, 41	Selected coding set A
{B	7B, 42	Selected coding set B

{C	7B, 43	Selected coding set C
{S	7B, 53	SHIFT
{1	7B, 31	FNC1
{2	7B, 32	FNC2
{3	7B, 33	FNC3
{4	7B, 34	FNC4

```
[Example] char SendStr[16];
SendStr[0] = 0x1D;
SendStr[1] = 'h';
SendStr[2] = 40;
PrtSendData(SendStr,3);// set the height of barcode as 40 vertical dot pitch(5mm)
SendStr[0] = 0x1D;
SendStr[1] = 'w';
SendStr[2] = 2;
PrtSendData(SendStr,3);//set the barcode width as 2
SendStr[0] = 0x1D;
SendStr[1] = 'H';
SendStr[2] = 2;
PrtSendData(SendStr,3);// set the reading character position of barcode as the below of
barcode
SendStr[0] = 0x1D;
SendStr[1] = 'f';
SendStr[2] = 0;
PrtSendData(SendStr,3);// set the reading character of barcode as 12x24 lattice character
SendStr[0] = 0x1D;
SendStr[1] = 'k';
SendStr[2] = 4;
SendStr[3] = '*'; SendStr[4] = 'T'; SendStr[5] = 'E' ;
SendStr[6] = 'S'; SendStr[7] = 'T'; SendStr[8] = '8' ;
SendStr[9] = '0'; SendStr[10] = '5'; SendStr[11] = '2' ;
SendStr[12] = '*'; SendStr[13] = 0;
PrtSendData(SendStr,14);// use formar1 to print CODE39 code“TEST8052”
```

4.6 user-defined character commands

4.6.1 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 LSB = 0, cancel user-defined character set.

When n LSB =1, select user-defined character set.

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

• n the lowest valid bit only is useful

[Default] $n = 0$

[Reference] **ESC &, ESC ?**

[Program example] char SendStr[4];

SendStr[0] = 0x1B;

SendStr[1] = '%';

SendStr[2] = 1;

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

4.6.2 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 ($12 \square 24$)

$y = 2$ font B ($8 \square 16$)

$32 \leq c1 \leq c2 \leq 126$

$x = 12$ font A ($12 \square 24$)

$x = 8$ font B ($8 \square 16$)

$0 \leq d1 \dots d(y \square xk) \leq 255$

[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]

• The range of defining character coding : ASCII code from 0x20 to 0x7E (95 characters)

• The sequence character coding that can be defined many character. When only need one character, order $c1=c2$.

• d is dot line data of character

• The character data of defined by user is $(y * x)$ byte

• Set the corresponding bit of print dot as 1 or the corresponding bit of non print dot as 0

• The command can define the different user-defined character mode for each type font. Use ESC! to set font.

• The total data of user-defiend character and download bitmap

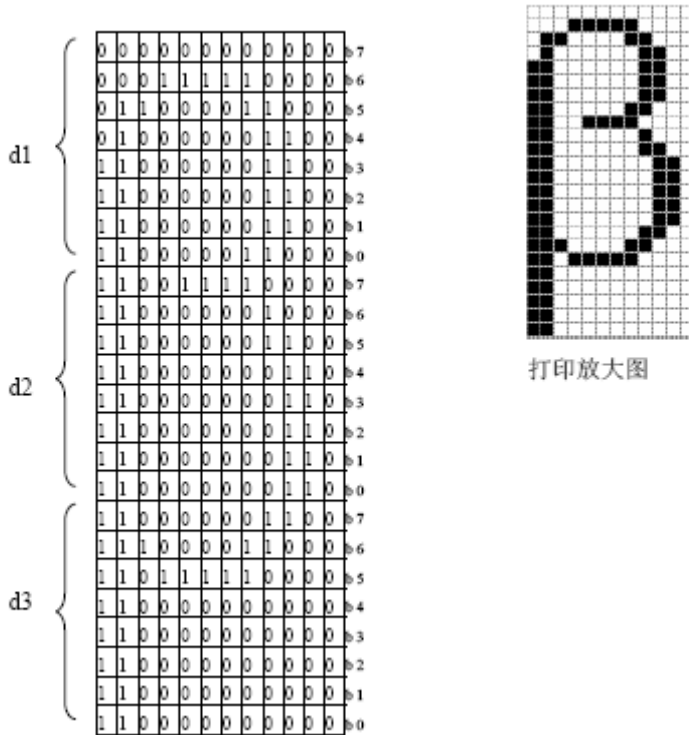
• User-defined characters will be cleared under following cases:

1. Execute **ESC @**

2. Execute **ESC ?**

3. Printer is reset or the power is off.

[Example] y=3,x=12 put character "B"(coding 0x42) defined as the following character with 12x24 dots



```

char SendStr[16];
SendStr[0] = 0x1B;
SendStr[1] = '&';
SendStr[2] = 3; //y=3 character height 24
SendStr[3] = 0x42; // charact , 'B' (code 0x42) redefined
SendStr[4] = x042;
SendStr[5] = 12; //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

```

4.6.3 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 user-defined character for the specified character code, the printer will neglect this command.

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

[Program example] char SendStr[4];

SendStr[0] = 0x1B;

SendStr[1] = '?';

SendStr[2] = 0x42;

PrtSendData(SendStr,3);//Clear characters defined by 0x42 .

4.7 Curve print command

4.7.1 GS ‘

[Name] Print *n* line segment in the same parallel line

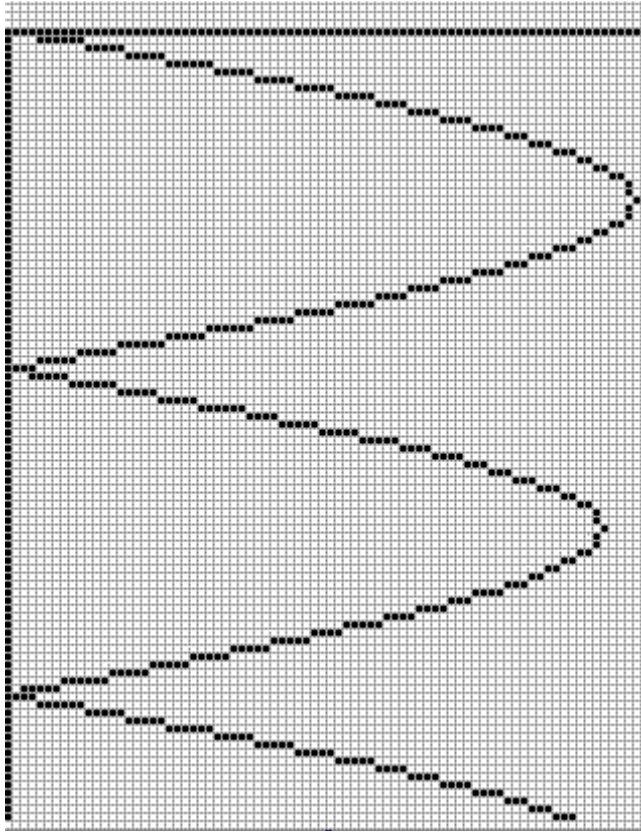
[Format] ASCII code **GS ‘** *n xlsL xlsH xleL xleH ... xnsL xnsH xneL xneH*

Hex code: **1D 27** *n xlsL xlsH xleL xleH ... xnsL xnsH xneL xneH*

decimal code: **29 39** *n xlsL xlsH xleL xleH ... xnsL xnsH xneL xneH*

[Range] $0 \leq n \leq 8$

[Discription] Print amplification show as below: each curve is by many level line (dots's length is one linesegment). This command as Print *n* line segment in the same parallel line , continuous use *n* this command can print the users need.



x_{ksL}

The low place of the starting point of Horizontal coordinates of the Kth line segment

x_{ksH}

The high place of the starting point of Horizontal coordinates of the Kth line segment

x_{keL}

The low place of the end point of Horizontal coordinates of the Kth line segment

x_{keH}

The high place of the end point of Horizontal coordinates of the Kth line segment;

The coordinates count begin from the most left of the printing area , mini value is 0,

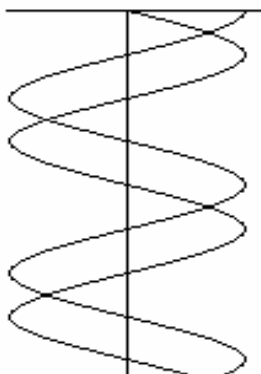
max is 383, that is , the max value of $x_{keL} + x_{keH} * 256$ is 383.

The data of line segment can not accord to the sequence;

[NOTE] • when print one dot, $x_{keL}=x_{ksL}$, $x_{keH}=x_{ksH}$.

[Reference] **GS** “

[Example] print the continuum Curve of Sin,Cos function , the print result as below:



Printing outline

```

SendStr[0] = 0x1D;
SendStr[1] = 0x27;
SendStr[2] = 3; // three line segment: X coordinate, Sin function and Cos function curv
SendStr[3] = 180; // X coordinate place is 180
SendStr[4] = 0;
SendStr[5] = 180;
SendStr[6] = 0;
for(i=1;i<1200;i++){
y1 = sin(i/180*3.1416)*(380-30)/2+180; // count the coordinate point of Sin function
y2 = cos(i/180*3.1416)*(380-30)/2+180; // count the coordinate point of Cos function

if(i==1) { y1s = y1; y2s = y2; }
PrtSendData( SendStr, 7 );
PrtSendData( &y1s, 2 ); // the Sin function Curve is located the starting point of this
line segment
PrtSendData( &y1, 2 ); // the Sin function Curve is located the end point of this line
segment

PrtSendData( &y2s, 2 ); // the cos function Curve is located the starting point of this
line segment

PrtSendData( &y2, 2 ); //the cos function Curve is located the end point of this line
segment

y1s = y1; // the Sin function Curve is located the starting point of this line segment
when setting of next line printing
y2s = y2; // the Cos function Curve is located the starting point of this line segment
when setting of next line printing

}

```

4.7.2 GS “

[Name] print the character of the curve

[Format] ASCII code GS “ *n xL xH c1 c2 ... NULL*

Hex code :1D 22 *n xL xH c1 c2 ... 00*

decimal code: 29 34 *n xL xH c2 ... 0*

[range] $0 \leq n \leq 1$

[description] this command print the characters of the printing curve as per current font, the command will circumgyrate the letter 90 degree(character string will circumgyrate clockwise)

n character code ;

xL xH is the level coordinate of character;

c1 c2 ... NULL is the character with zero as ending

[Note] • the command is available only when the command appear between two ESC

' command .

- After receiving the command, it will print the character with 90 degree circumgyrated from the current line.
- When the level point line have character, if print other character , need set the character code as other value, only limited to 0 and 1
- Per level point line most have tow character

[Reference] **GS** ‘

[example] print the continuum Curve of Sin,Cos function , the print result as below:

```
char SendStr1[8], SendStr2[16];
```

```
int i;
```

```
short y1,y2,y1s,y2s;
```

```
// print Y axes line (one line segment)
```

```
SendStr1[0] = 0x1D;
```

```
SendStr1[1] = 0x27;
```

```
SendStr1[2] = 1; // one line segment
```

```
SendStr1[3] = 30; // the start point is 30
```

```
SendStr1[4] = 0;
```

```
SendStr1[5] = 104; // the end point is 360
```

```
SendStr1[6] = 1;
```

```
// the origin character of the print coordinate "0" .
```

```
PrtSendData(SendStr,7);
```

```
SendStr2[0] = 0x1D;
```

```
SendStr2[1] = 0x22;
```

```
SendStr2[2] = 0; // character code 0
```

```
SendStr2[3] = 150; // the level coordinate is 150
```

```
SendStr2[4] = 0;
```

```
SendStr2[5] = '0'; //character'0'
```

```
SendStr2[6] = 0;
```

```
PrtSendData(SendStr2,7);
```

```
// print the curve line
```

```
SendStr1[0] = 0x1D;
```

```
SendStr1[1] = 0x27;
```

```
SendStr1[2] = 3; // three line segment: X coordinate, Sin function and Cos function curv
```

```
SendStr1[3] = 180; // X coordinate place is 180
```

```
SendStr1[4] = 0;
```

```
SendStr1[5] = 180;
```

```
SendStr1[6] = 0;
```

```
// character
```

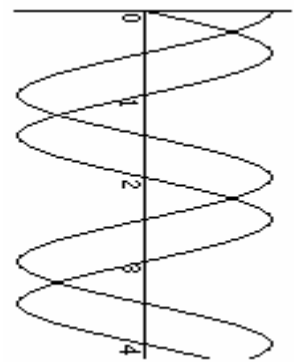
```
SendStr2[0] = 0x1D;
```

```
SendStr2[1] = 0x22;
```

```
SendStr2[2] = 0;
```

```
SendStr2[3] = 150;
```

```
SendStr2[4] = 0;
```



Printing outline

```

SendStr2[5] = 0;
SendStr2[6] = 0;
for(i=1;i<1200;i++){
y1 = sin(i/180*3.1416)*(380-30)/2+180; // count the coordinate point of
Sin function

y2 = cos(i/180*3.1416)*(380-30)/2+180; // count the coordinate point of
Cos function
if(i==1) { y1s = y1; y2s = y2; }
PrtSendData( SendStr1, 7 );
PrtSendData( &y1s, 2 ); // the Sin function Curve is located the starting point
of this line segment
PrtSendData( &y1, 2 ); // the Sin function Curve is located the end point of
this line segment

PrtSendData( &y2s, 2 ); // the cos function Curve is located the starting point
of this line segment

PrtSendData( &y2, 2 ); // the cos function Curve is located the end point of
this line segment

y1s = y1; // the Sin function Curve is located the starting point of this line
segment when setting of next line printing

y2s = y2; // the Cos function Curve is located the starting point of this line segment
when setting of next line printing

if(i%180==0) SendStr2[5] = '0'+i/180; // print the gauge of X axes
PrtSendData( SendStr1, 7 );

```

B.1 Barcode

UPC-A: UPC-A code should be compatible with the regulation of UUC association(<http://www.uccnet.org>)

UPC-E: UPC-E code should be compatible with the regulation of UUC association(<http://www.uccnet.org>)

ENA8:ENA code should be compatible with the regulation of EAN association(<http://www.ean-int.org>)

ENA13: ENA13 code should be compatible with the regulation of EAN association(<http://www.ean-int.org>)

CODE39: the start bit character and ending bit character of code39 must be '*', and they can not include character '*', the data include or non include the check code, the check code has fixed arithmetic.

ITF: named INTERLEAVED25, INTERLEAVED 2 of 5, the data length only is even

number, the data include or non include the check code, the check code has fixed arithmetic.

CODEBAR: the start bit and ending bit must is one of four character A,B,C,D, There aren't necessarily the same between the start bit character and the ending character

CODE93: the start bit character and ending bit character of code39 must be '*' , and they can not include character '*', the last two character of code93 must include two character check code, the check code has fixed arithmetic.

B.2 Barcode length character set

Barcode type	Length	character set (ASCII)
UPC-A	12	0~9
UPC-E	8	0~9
EAN8	8	0~9
EAN13	13	0~9
CODE39	without limit	0~9 A~Z - . SP \$ / + % *
INTERLEAVED 25	even number	0~9
CODABAR	without limit	0~9 - : / % . A~D
CODE93	without limit	0~9 A~Z - . SP \$ / + % *

C.Pre-printing black mark 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 thermosensitive 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 $< 10\%$ (reflection rate to the detecting mark on paper $> 65\%$)

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

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

