

CPCL 指令集编程手册

撰写人：张紫玲

撰写日期：2023.4.4

审核人：

审核日期：

变更记录

版本	变更内容	修改人	时间
V1.0	初版	张紫玲	2023.4.4

目录

目录.....	3
前言.....	5
1 格式介绍.....	6
1-1 控制指令 ”!”.....	6
1-2 打印指令 PRINT.....	7
1-3 终止命令 END.....	8
1-4 忽略指令 ABORT.....	8
1-5 设置页面宽度 PAGE-WIDTH.....	9
1-6 打印前走纸距离 PREFEED.....	10
1-7 打印后走至距离 POSTFEED.....	10
1-8 对齐命令.....	11
1-9 增量指令COUNT.....	13
1-10 注释指令 <;>.....	14
2 文字打印指令.....	15
2-1 文本打印指令TEXT.....	15
2-2 放大指令SETMAG.....	17
2-3 文字串联命令(CONCAT和VCONCAT).....	18
2-4 多行文本打印指令 MULTILINE.....	19
2-5 设置字符间距指令 SETSP.....	20
2-6 设置字符加粗指令 SETBOLD.....	21
2-7 设置字符下划线指令 UNDERLINE.....	22
3 条码打印指令.....	23
3-1 条码指令 BARCODE.....	23
3-2 条码标识符 BARCODE-TEXT.....	25
4 二维码条码指令.....	26
4-1 二维码 PDF417.....	26
4-2 二维码 QRCODE.....	28
4-3 二维码 DATAMATRIX.....	30

5 基本图形与位图打印指令31

5-1 线条指令 LINE 31

5-2 反白线条 INVERSE-LINE 32

5-3 矩形指令 BOX 33

5-4 位图指令 34

前言

本文为精臣智慧标识科技有限公司开发的CPCL指令集编程手册，旨在引导使用者如何正确、有效的使用标准CPCL指令集，该指令集一般适用于便携机。CPCL指令集在打印快递面单、处罚单、缴费单等等有一定的优势。通过指令的编排可以批量生成固定格式的面单。目前支持CPCL指令集的打印机可以通过蓝牙或者USB发送CPCL指令进行打印。

发送CPCL指令集需保证格式正确、内容合法。其中参数范围需要根据不同打印机的规格进行调整，打印机规格主要决定了指令集中需要输入参数的上下限。若输入不合法参数，机器会自动调整参数到合法范围内，为保证打印效果请详细阅读此文档并输入合法的指令。

内部人员使用须知

CPCL指令集的开发并未迭代到一个非常成熟的阶段。经过初步的测试可以保证绝大部分内容可以正常打印和使用。但更希望在很多内部工作人员作为使用端，可以发现并反馈更多的问题。在使用前确保自己的使用方式，如在使用过程中发现指令集中的问题请私聊相关开发工作人员或者提交缺陷单。

1 格式介绍

1-1 控制指令 ”!”

该指令表示开启一次标签打印，该格式必须遵守**感叹号**开头，且为英文感叹号，后续每一个参数以一个**空格**隔开。

格式：

```
<!> {offset} <200> <200> {height} {total page}
```

解释：

<!>：作为控制会话的起始字符，也是CPCL指令集格式的起始标记

{offset}：标签的横向偏移(以点为单位)

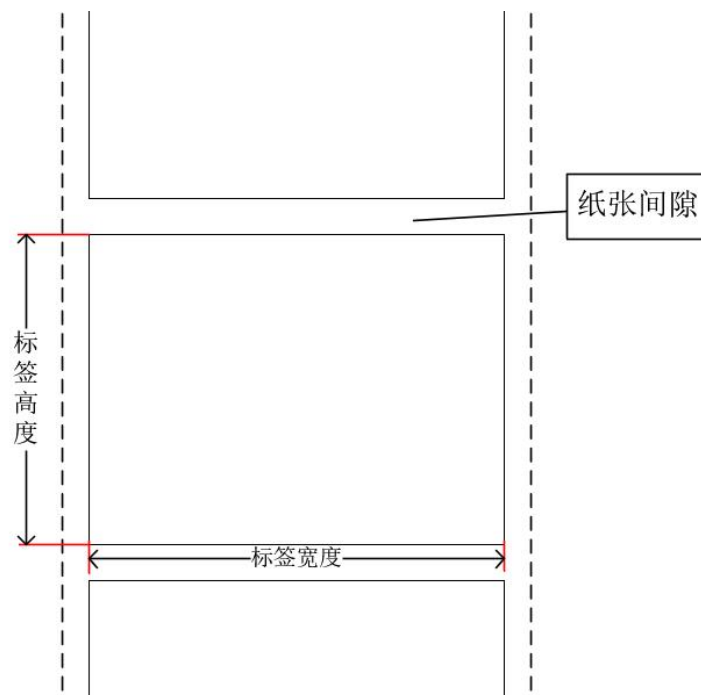
<200>：横向分辨率 (以点为单位) } 这两项一般与打印
<200>：纵向分辨率 (以点为单位) } 头规格保持一致

{height}：标签的最大高度/需进行打印的高度 (单位：点)

{total page}：需要打印的总份数

备注：

纸张高度宽度以及纸张间隙如下图所示。



指令示例

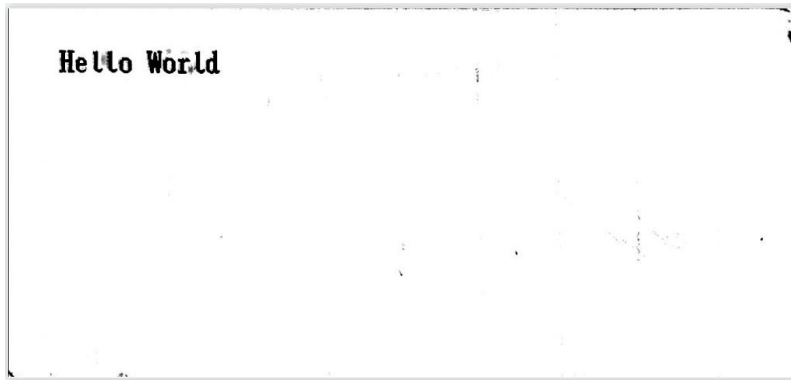
输入:

```
! 0 200 200 200 1
```

```
TEXT 24 0 30 40 Hello World
```

```
PRINT
```

输出:



1-2 打印指令 PRINT

PRINT 命令作为整个打印过程的结束命令，将会启动打印。在任何情况下（行式打印模式除外），这项命令都必须是最最后一条命令。执行 **PRINT** 命令时，打印机将从控制会话中退出。确保使用回车和换行字符结束此项及所有命令。

格式:

```
{command}
```

示例:

```
PRINT
```

1-3 终止命令 **END**

结束指令并执行END之前的指令，在执行END时会处理前面的指令并打印出来

格式：

{command}

示例：

END

指令示例：

执行END之前的BOX指令并打印该页

! 0 200 200 240 1

BOX 0 0 200 200 10

BOX 50 50 220 220 10

END

1-4 忽略指令 **ABORT**

ABORT指令即忽略该指令之前输入的指令，并退出指令集模式。

格式

{command}

示例：

ABORT

指令示例：

结果：并不会打印任何样张

! 0 200 200 240 1

BOX 0 0 200 200 10

BOX 50 50 220 220 10

ABORT

1-5 设置页面宽度 PAGE-WIDTH

打印机会默认页面宽度为打印机的完整宽度(即打印头宽度)。打印页面最大高度由打印机可用内存决定。如果需要打印的标签页宽小于打印机完整宽度，则用户可以使用该指令来指定当前最大可打印的页面宽度，该页面宽度对于打印机的完整宽度来说是居中的。

注意事项：此命令应该在打印内容指令开始前发出。

格式：

{command} {width}

解释：

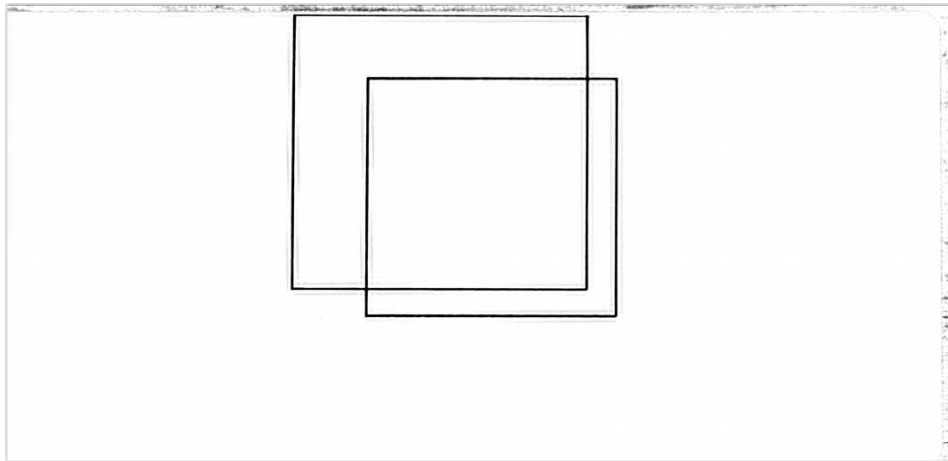
{command}：设置页宽的指令字 PAGE-WIDTH (或者PW)

{width}：页面的单位宽度。

示例：

```
! 0 200 200 240 1
PAGE-WIDTH 240
BOX 0 0 200 200 2
BOX 50 50 220 220 2
PRINT
```

打印结果：



1-6 打印前走纸距离 PREFEED

该命令控制打印机在打印之前将介质(纸张/标签/标牌...)向前移动指定长度。

格式:

{command} {length}

解释:

{command}: PREFEED

{length}: 打印机在打印之前将介质向前移动的单位长度。

示例:

下列指令将打印机设置为在打印之前向前移动40点的长度

输入:

```
! 0 200 200 210 1
PREFEED 40
TEXT 16 0 0 20 PREFEED EXAMPLE
PRINT
```

1-7 打印后走至距离 POSTFEED

POSTFED指令控制打印机在打印之后将介质(纸张/标签/标牌...)向前移动指定长度。

格式:

{command} {length}

其中:

{command}: POSTFEED

{length}: 打印机在打印之后将介质向前移动的单位长度

示例:

下列指令将打印机设置为在打印之后向前移动40点。

输入:

```
! 0 200 200 210 1
TEXT 16 0 0 20 POSTFEED EXAMPLE
FORM
POSTFEED 40
PRINT
```

1-8 对齐命令

使用对齐命令可以控制字段的对齐方式。默认情况下，打印机将左对齐所有字段。对齐命令将对所有后续字段保持有效，直至指定了其他对齐命令。

格式：

{command}

解释：

{command}：

CENTER：居中对齐所有后续字段

(居中对齐是已字段x坐标与最右边界居中)

LEFT：左对齐所有后续字段

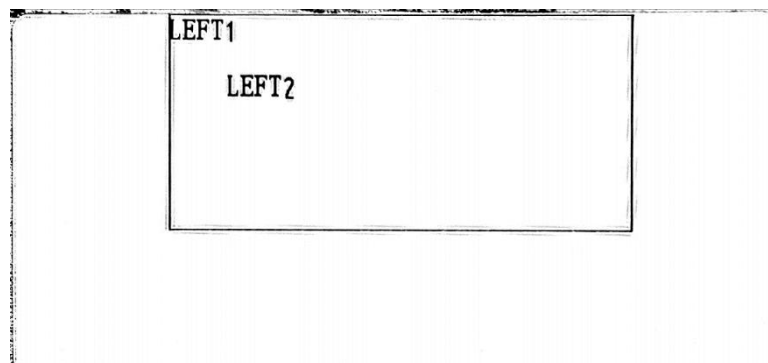
RIGHT：右对齐所有后续字段

指令示例：

居左：

```
! 0 200 200 210 1
PAGE-WIDTH 400
BOX 0 0 398 198 2
LEFT
T 24 0 0 10 LEFT1
T 24 0 50 60 LEFT2
PRINT
```

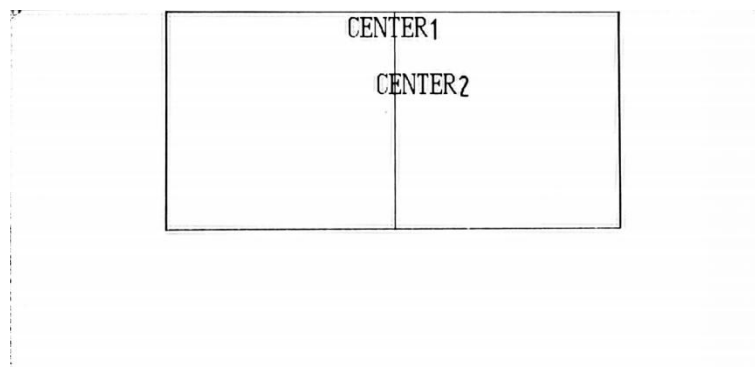
结果：



居中：

```
! 0 200 200 200 1
PAGE-WIDTH 400
LINE 200 0 200 198 1
BOX 0 0 398 198 2
CENTER
T 24 0 0 10 CENTER1
T 24 0 50 60 CENTER2
PRINT
```

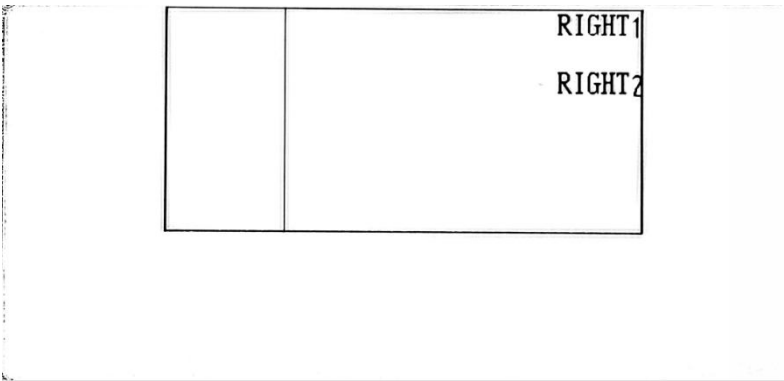
结果：



居右:

```
! 0 200 200 200 1
PAGE-WIDTH 400
LINE 100 0 100 198 1
BOX 0 0 398 198 2
RIGHT
T 24 0 0 10 RIGHT1
T 24 0 50 60 RIGHT2
PRINT
```

结果:



1-9 增量指令COUNT

COUNT命令可以用于打印多份标签，其中条码中编码的数字文本域或数字数据将针对每个标签依次递增或者递减。TEXT/BARCODE命令字符串必须包含此数字数据，将其作为字符串的最后若干字符。数字数据部分最多可以包含20个字符，且可以以‘-’符号作为前缀。增加或减少数字数据时不能以‘0’为增量或减量。前导零将予以保留。一个标签文件中最多可使用三个COUNT命令。

递增/递减的数字数据包含在TEXT或BARCODE命令中，后面紧跟COUNT命令。

格式：

{command} {numeric value}

解释：

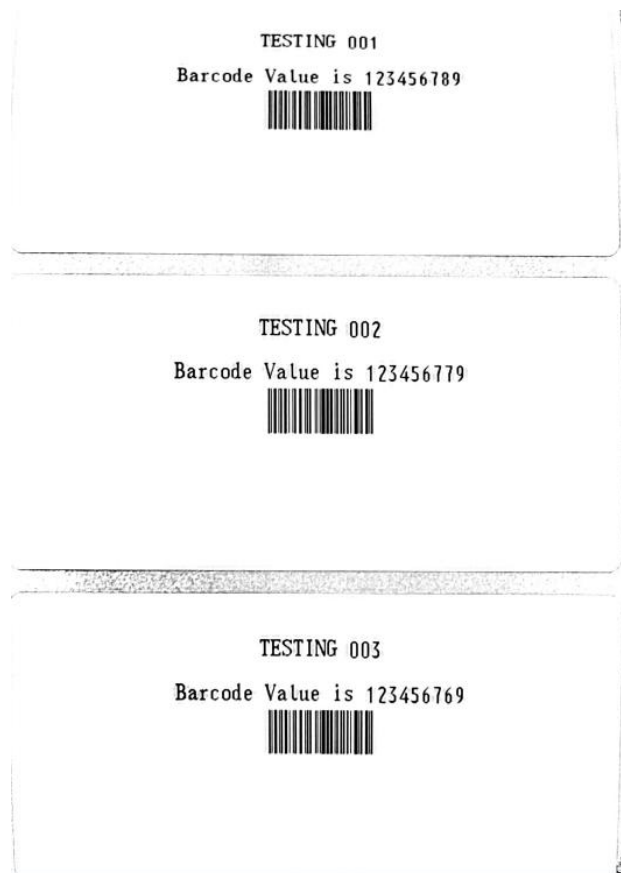
{command}：COUNT

{numeric value}：任何整数值都不能超过20个字符。如果希望减小TEXT/BARCODE，则可以在值前添加‘-’符号。输出结果中将保留前导零。

示例：

结果：

```
! 0 200 200 210 3
CENTER
TEXT 4 0 0 50 TESTING 001
COUNT 1
TEXT 7 0 0 100 Barcode Value is 123456789
COUNT -10
BARCODE 128 1 1 50 0 130 123456789
COUNT -10
PRINT
```



1-10 注释指令 <;>

注释可以添加在每一条命令上方，一般添加在PRINT指令上方即可。

在文件中添加注释时，需要将”;;”字符置入第一列，以此作为注释行的起始部分。”;;”字符与行末尾的所有其他文本都将被忽略。但带有首尾部分例如：CONCAT与ENDCONCAT以及BARCODE与ENDQR之间不可加入注释。

注释示例：

```
! 0 200 200 200 1
;;Center
CENTER
;;Print the words “HELLO WORLD”
TEXT 24 0 20 30 HELLO WORLD
;;Print the label
PRINT
```

2 文字打印指令

2-1 文本打印指令TEXT

TEXT指令用于打印文本。这项指令以及这项指令其他的控制指令可以调整文字的**方向**、文字的**起始坐标**、**对齐方式**、**文本间距**、以及**文本行间距**。

格式：

{command} {font} {size} {x} {y} {data}

解释：

{command}：可选择以下几项

command	预期打印效果
TEXT(或T)	横向打印文本。
VTEXT(或VT)	逆时针旋转90度，纵向打印文本。
TEXT90(或T90)	逆时针旋转90度，纵向打印文本。
TEXT180(或T180)	逆时针旋转180度，反向打印文本。
TEXT270(或T270)	逆时针旋转270度，纵向打印文本。
TR	水平打印反白文本。
TR90	逆时针旋转90度打印反白文本
TR180	逆时针旋转180度打印反白文本。
TR270	逆时针旋转270度打印反白文本。

{font}：字体名称/编号 具体字体支持情况根据机器有所变化

{size}：字体的大小标识（目前未使用）

{x}：文本的起始横坐标(单位：点) 注意：起始坐标是以文本左上角坐标为基准

{y}：文本的起始纵坐标

{data}：需要打印的文本

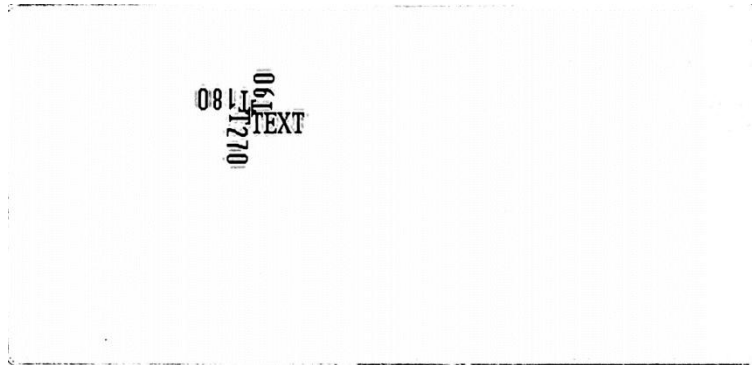
注意事项：

当使用不同角度打印文本的时候，注意坐标可能需要改变。(例如文本旋转90度，文本的左上角纵坐标需要增加(下移)，否则后续的文本打印在标签的上方，因纵坐标过小则导致文本超出页面显示范围)。

指令示例：

```
! 0 200 200 210 1
TEXT 24 0 200 100 TEXT
TEXT90 24 0 200 100 T90
TEXT180 24 0 200 100 T180
TEXT270 24 0 200 100 T270
PRINT
```

打印结果：



2-2 放大指令SETMAG

该指令可以将默认字体放大指定的放大倍数。

格式:

{command} {wide} {high}

解释:

{command}: SETMAG

{wide}: 字体的宽度放大倍数。有效放大倍数为 1~16

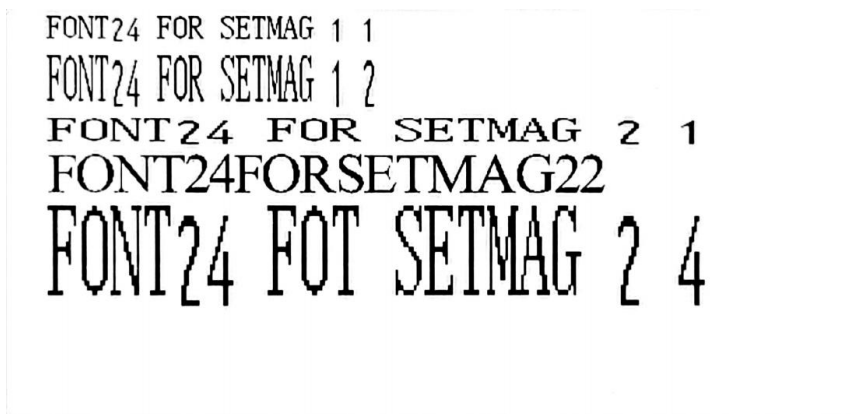
{high}: 字体的高度放大倍数。有效放大倍数为 1~16

指令示例

输入:

```
! 0 200 200 240 1
SETMAG 1 1
TEXT 24 0 20 10 FONT24 FOR SETMAG 1 1
SETMAG 1 2
TEXT 24 0 20 40 FONT24 FOR SETMAG 1 2
SETMAG 2 1
TEXT 24 0 20 80 FONT24 FOR SETMAG 2 1
SETMAG 2 2
TEXT 24 0 20 110 FONT24 FOR SETMAG 2 2
SETMAG 2 4
TEXT 24 0 20 155 FONT24 FOT SETMAG 2 4
SETMAG 0 0
PRINT
```

打印结果:



FONT24 FOR SETMAG 1 1
FONT24 FOR SETMAG 1 2
FONT24 FOR SETMAG 2 1
FONT24FORSETMAG22
FONT24 FOT SETMAG 2 4

2-3 文字串联命令(CONCAT和VCONCAT)

使用文本串联，可以为字符串分配不同的字符样式，在同一文本行上使用同一的间距打印。

格式：

```
{start command} {x} {y}
{font} {size} {data}
.....
{font} {size} {data}
{end command}
```

解释：

{start command}： CONCAT： 水平方向的文字关联 VCONCAT： 垂直方向的文字关联

{x}： 文本区域开始的水平坐标

{y}： 文本区域开始的垂直坐标

{font}： 字体的名字或者编号

{size}： 字体字号(大小) 该参数参考上述TEXT指令中的size

{offset}： 文本相对起始位置的偏移，用于对齐单个文本。

{data}： 要打印的文本

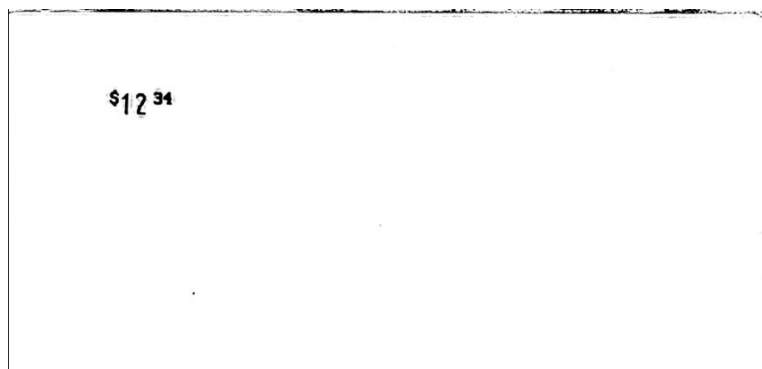
{end command}： ENDCONCAT 用于终止文字串联功能。

注意：如果开始了文字串联功能，未收到终止文字串联功能，是无法正常使用文字串联功能的。

指令示例：

```
! 0 200 200 210 1
CONCAT 75 75
16 0 5 $
24 0 0 12
16 0 5 34
END CONCAT
PRINT
```

打印结果：



2-4 多行文本打印指令 MULTILINE

使用MULTILINE(ML)，可以以相同的字体和指定行高打印多行文字。

格式：

```
{command} {height}
{text} {font} {size} {x} {y}
{content}
...
{content}
<ENDMULTILINE>
```

解释：

{command}：使用指令MULTILINE或ML(缩写) 开始打印多行文本。

{height}：每行文本的高度(即下一行文字的起始纵坐标相对上一行文字的起始纵坐标的偏移)

{text}：文本指令 使用<TEXT/VTEXT>

{font}：字体名称/编号

{size}：字体的大小

{x}：文本区域的起始横坐标

{y}：文本区域的起始纵坐标

{content}：要打印的文本

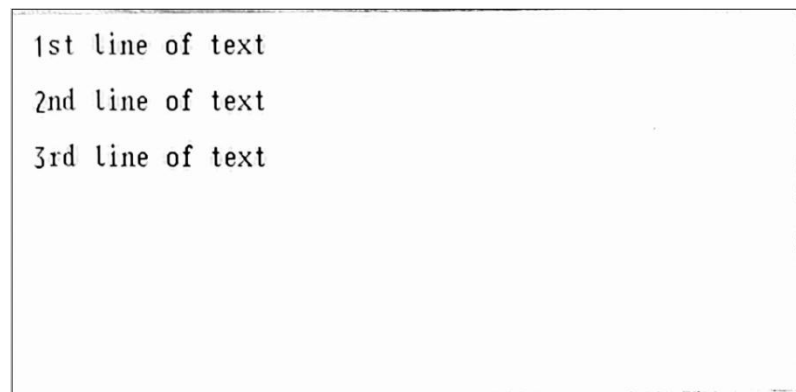
<ENDMULTILINE>或者ENDML：终止多行打印指令

注意：如果开始多行打印文本，未收到多行打印的终止指令，则无法正常使用该指令。

指令示例：

打印结果：

```
! 0 200 200 210 1
ML 47
TEXT 24 0 10 20
1st line of text
2nd line of text
3rd line of text
ENDML
PRINT
```



```
1st line of text
2nd line of text
3rd line of text
```

2-5 设置字符间距指令 SETSP

该指令用于更改文本字符之间的间距。

格式:

{command} {space}

解释:

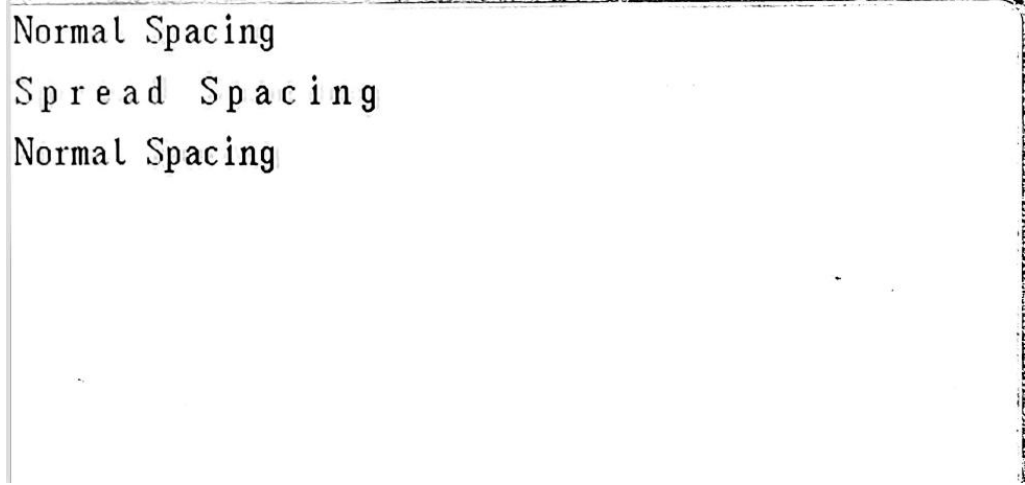
{command}: 指令填入SETSP

{spacing}: 字符与字符之间的间隔大小

示例:

```
! 0 200 200 210 1
T 24 0 0 10 Normal Spacing
SETSP 5
T 24 0 0 50 Spread Spacing
SETSP 0
T 24 0 0 90 Normal Spacing
PRINT
```

打印结果:



```
Normal Spacing
Spread Spacing
Normal Spacing
```

2-6 设置字符加粗指令 SETBOLD

该指令可以使文本加粗、稍微加宽。SETBOLD指令会使用一个参数来设置加粗的程度。

格式:

```
{command} {value}
```

解释:

{command}: SETBOLD

{value}: 一个介于0-5的偏移量, 用于标注加粗的点数

示例:

```
! 0 200 200 300 1
SETBOLD 2
T 24 0 10 10 abcdefghABCDEFGH
T 24 0 10 70 0123456789
T 16 0 10 140 中国梦, 中国心
T 24 0 10 180 中国梦, 中国心
SETBOLD 0
PRINT
```

打印结果:



2-7 设置字符下划线指令 UNDERLINE

该命令用于给文本加下划线。仅当所使用的字体支持下划线时，此命令才会起作用。如果所使用的字体不支持UNDERLINE，则会忽略此命令。

格式：

{command} {mode}

解释：

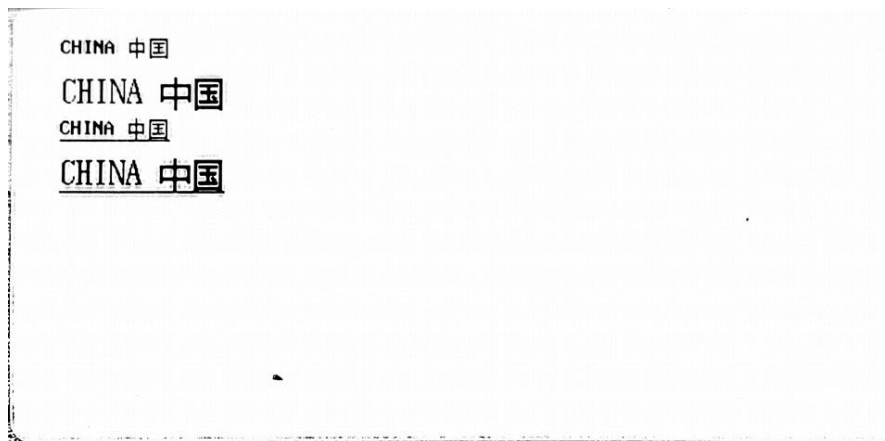
{command}： UNDERLINE

{mode}： ON - 打开字符下划线 OFF - 关闭字符下划线

示例：

```
! 0 200 200 200 1
UNDERLINE OFF
TEXT 16 0 20 30 CHINA 中国
TEXT 24 0 20 60 CHINA 中国
UNDERLINE ON
TEXT 16 0 20 90 CHINA 中国
TEXT 24 0 20 120 CHINA 中国
PRINT
```

打印结果：



3 条码打印指令

条码可以对很多需要处理的对象进行自动标识、分类和处理。条码目前广泛用于各种物品。从商品条码到快递单号。

3-1 条码指令 **BARCODE**

BARCODE指令能够以指定的宽度和高度纵向和横向打印条码。

格式：

{command} {type} {width} {ratio} {height} {x} {y} {data}

解释：

{command}：

BARCODE(或B)：横向打印条码。

VBARCODE(或VB)：纵向打印条码。

{type}：条码类型，从下表中选择

条码	Type类型字
CODE128	128、128A、128B、128C
UPC-A	UPCA、UPCA2、UPCA5
UPC-E	UPCE、UPCE2、UPCE5
EAN13	EAN13、EAN132、EAN135
EAN8	EAN8、EAN82、EAN85
CODE39	39、39C、F39、F39C
CODE93	93
Interleaved 2 of 5	I2OF5
Codabar	CODABAR、CODABAR16

{width}：窄条的单位宽度。

{ratio}：宽条和窄条的比率。

0 = 1.5:1	20 = 2.0:1	26 = 2.6:1
1 = 2.0:1	21 = 2.1:1	27 = 2.7:1
2 = 2.5:1	22 = 2.2:1	28 = 2.8:1
3 = 3.0:1	23 = 2.3:1	29 = 2.9:1
4 = 3.5:1	24 = 2.4:1	30 = 3.0:1
	25 = 2.5:1	

{height}：条码的单位高度(单位：点)

{x}：条码开始的横坐标(单位：点)

{y}：条码开始的纵坐标(单位：点)

{data}：需要编码的数据

一维码数据格式：

在输入需要编码的数据的时候，需要注意一维码类型，不同的一维码类型需要输入不同的格式。

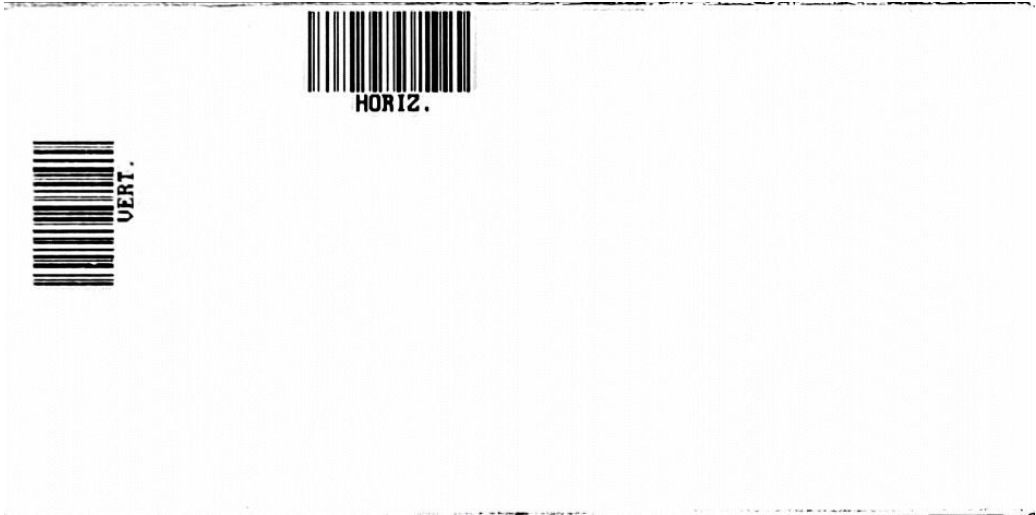
以下表格列出了不同的一维码的可编码数据的格式要求。

种类	格式	校验	支持的字符集
EAN8	7或8位	校验码	支持0~9数字字符
EAN13	13位	校验码	支持0~9数字字符
UPC-A	11或12位	校验码	支持0~9数字字符
UPC-E	11或12位	校验码	支持0~9数字字符
39码	可变长，最多43	自校验	支持数字、字母字符以及少量符号
93码	可变长	校验码	支持数字、字母字符以及少量符号
128	可变长	校验码	支持全ASCII码
CODABAR	首尾须为字母	自校验	支持ABCD字母、数字以及少量符号
ITF	可变长	自校验	支持0~9数字字符

示例：

```
! 0 200 200 210 1
BARCODE 128 1 1 50 180 10 HORIZ.
TEXT 16 0 210 60 HORIZ.
VBARCODE 128 1 1 50 10 180 VERT.
VTEXT 16 0 60 140 VERT.
PRINT
```

结果：



3-2 条码标识符 **BARCODE-TEXT**

该指令用于创建条码时所用的编码数据打印在指定区域用于标记条码的内容。这项指令避免了单独使用文本指令来标记条码。文本位于条码下方的中间位置。

使用BARCODE-TEXT(或BT) OFF可以禁用该功能。

格式:

```
{command} {font number} {font size} {offset}
```

解释:

{command}: BARCODE-TEXT(或BT)

{font number}: 注释条码时要使用的字体号

{font size}: 注释条码时要使用的字体大小。

{offset}: 文本距离条码的单位偏移量。

示例:

```
! 0 200 200 400 1
CENTER
BARCODE-TEXT 16 0 5
BARCODE 128 1 1 50 0 20 123456789
VBARCODE 128 1 1 50 40 300 112233445
BARCODE-TEXT OFF
PRINT
```

结果:



4 二维码条码指令

二维码相当于条码类型中“便携式数据库”。二维码可以编码数字、文本、标点符号、语音、视频等等。第一代二维条码是一维码的扩展。可以使用多个Code39堆叠式放在一起，这一类二维码叫做堆叠式二维码，例如下面即将介绍的类型：PDF417。

Qrcode为框架式二维码，三个定位图形为Qrcode框架，其他部分为数据内容。Qrcode可以编码数字、字母数字字符、中文、日文。一个符号中最多可以支持7089个字符的编码。Qrcode具有强大的纠错能力，即使二维码遭到一定程度的损毁和遮挡，也可以对数据进行纠错，不影响正常的扫描。

Datamatrix为矩阵式二维码。在几种二维码中，Datamatrix的尺寸是最小的，可以在25*25(mm)的面积上编码30个数字，特别适用于小零件的标识或者直接印刷在实体上。比如贵重物品的快递单有特意的编排的序列号吗，可以直接使用datamatrix进行印刷识别。

4-1 二维码 PDF417

格式：

```
{command} {type} {x} {y} {XD n} {YD n} {C n} {S n}  
{data}  
<ENDPDF>
```

解释：

{command}：

BARCODE(或B)：水平打印条码。

VBARCODE(VB)：垂直打印条码。

{type}：PDF-417

{x}：二维码起始横坐标

{y}：二维码起始纵坐标

{XD n}：最窄元素点的单位宽度。范围介于1-32之间，默认值为2。

{YD n}：最窄元素点的单位高度。范围介于1-32之间，默认值为6。

{C n}：要使用的列数。数据列不包括起始/终止字符和左/右指示符。范围介于1-30之间，默认值为3。

{S n}：纠错等级，只是要检测或要纠正的最大错误量。纠错等级分为0~8一共9个等级，默认值为1。

<ENDPDF>：终止PDF-417二维码指令。

注意：如果不能按照以上正常格式进行输入，无法打印出正确的PDF-417二维码。

示例：

```
! 0 200 200 210 1
B PDF-417 10 20 XD 3 YD 12 C 3 S 2
PDF Data
ABCDE12345
ENDPDF
T 16 0 10 120 PDF Data
T 16 0 10 170 ABCDE12345
PRINT
```

结果:



4-2 二维码 Qrcode

格式:

```
{command} {type} {x} {y} {M n} {U n}  
{data}  
<ENDQR>
```

解释:

{command}:

BARCODE(或B): 水平打印二维码

VBARCODE(或VB): 垂直打印二维码

{type}: QR

{x}: 二维码起始横坐标

{y}: 二维码起始纵坐标

{M n}: QR Code规范编号。选项为1或2.这里未做处理,默认使用最新模式,且对大小进行自适应,即根据数据量自动调整二维码尺寸。

{U n}: 模块的单位宽度/高度(QRcode模组默认为正方形),范围是1-32。默认值为6.

{data}: 提供生成QR Code所需要的信息。有格式要求如下:

该部分除了输入需要编码的数据内容外,还包含模式选择字符。输入数据的类型可以由打印机自行判断,也可以使用手动指定相应的模式,模式选择符号和实际数据之间有一个分隔符(逗号)。

格式为 <Error Correction Level><Data Input Mode>,<Character Mode1><Data Character String 1> <Character Mode n><Data Character String n>

纠错等级选项:

H – 极高可靠性级别

Q – 高可靠性级别

M – 标准级别

L – 高密度级别

注意: 当选择数据编码模式为自动模式时,分隔符后续的编码数据不需要进行模式指定且无法设置0x80至0x9f和0xe0至0xff的二进制代码。如果为手动选择数据类型,则需要遵守字符串模式和字符串的格式规定。

字符模式符号:

N – 数字模式

A – 字母数字模式

B – 二进制模式,二进制模式包含由2字节BCD代码表示数据字符的数量

K – Kanji 因为目前暂无Kanji编码,为方便使用该部分修改为中文编码

不同的数据字段应带有队形的字符模式符号，且由逗号分隔。

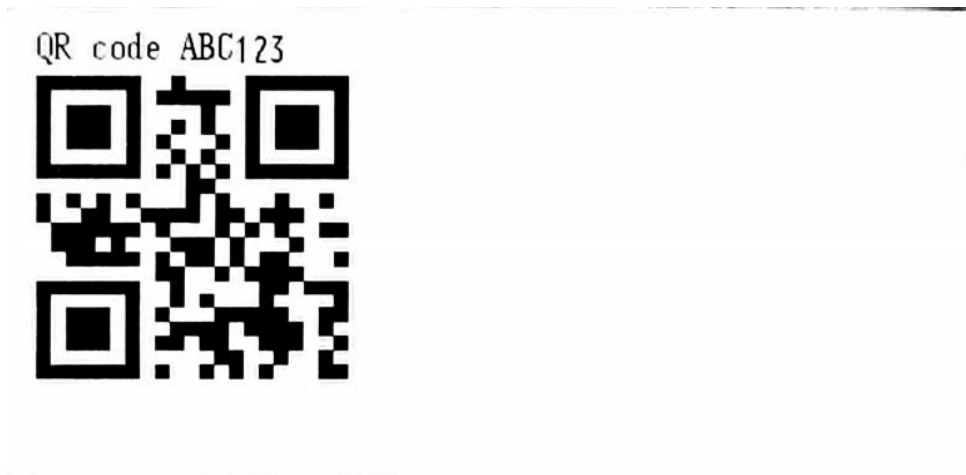
<ENDQR>: 终止QR Code代码

示例:

方便起见，一般使用自动选择数据类型模式即可。

```
! 0 200 200 500 1  
B QR 10 50 M 2 U 10  
MA,QR code ABC123  
ENDQR  
T 24 0 10 20 QR code ABC123  
PRINT
```

结果:



4-3 二维码 Datamatrix

格式:

```
{command} {type} {x} {y} {H n}  
{data}  
<ENDDATAMATRIX>
```

解释:

{command}:

 BARCODE(或B): 水平打印Datamatrix二维码

 VBARCODE(或VB): 垂直打印Datamatrix二维码

{type}:DATAMATRIX

{x}: 二维码的起始横坐标

{y}: 二维码的起始纵坐标

{H n}: 二维码模组的高度/宽度

{data}: 需要编码的数据

<ENDDATAMATRIX>: 表示Datamatrix指令结束

示例:

```
! 0 200 200 600 1  
B DATAMATRIX 20 40 H 5  
NIIMBOT  
More simplier,More better  
ENDDATAMATRIX  
PRINT
```

结果:



5 基本图形与位图打印指令

5-1 线条指令 LINE

使用LINE质量零可以绘制任意长度宽度和方向的线条

格式:

{command} { x_0 } { y_0 } { x_1 } { y_1 } {width}

解释:

{command}: LINE(或L) 绘制线条的指令字

{ x_0 }: 线条起始点的横坐标

{ y_0 }: 线条起始点的纵坐标

{ x_1 }: 线条终止点的横坐标

水平线条的右上角

垂直线条的左下角

{ y_1 }: 线条终止点的纵坐标

水平线条的右上角

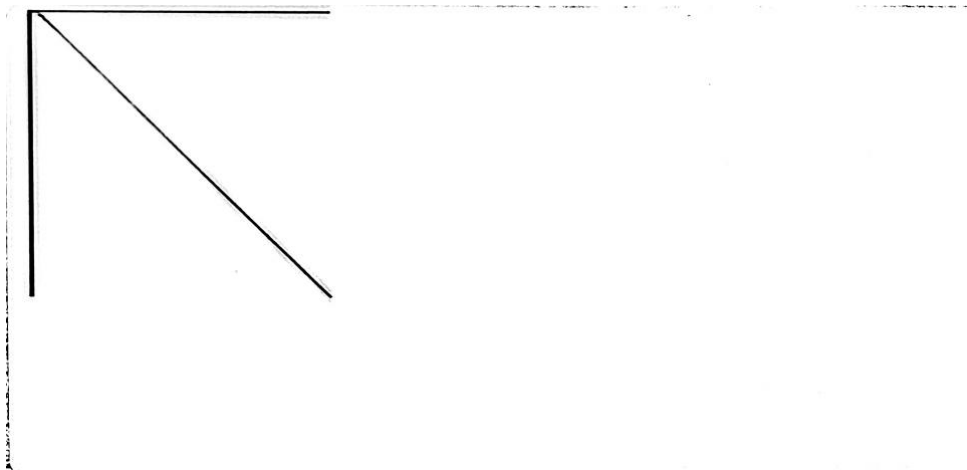
垂直线条的左下角

{width}: 线条的单位宽度

示例:

```
! 0 200 200 210 1
LINE 0 0 200 0 1
LINE 0 0 200 200 2
LINE 0 0 0 200 3
PRINT
```

结果:



5-2 反白线条 INVERSE-LINE

INVERSE-LINE命令的语法与LINE命令相同。位于INVERSE-LINE命令所定义的区域内将其中的对象黑色的区域将重绘为白色，白色的区域重绘为黑色。这些对象包含文本、条码/或图形。INVERSE-LINE对在其之后创建的对象不起作用，即使这些对象位于反白线段的覆盖区域也不会起作用。

格式：

{command} {x₀} {y₀} {x₁} {y₁} {width}

解释：

{command}：从下面选择一项

INVERSE-LINE(或IL)：在现有字段上方打印一个线条以反转图像。

{x₀}：左上角的x坐标

{y₀}：左上角的y坐标

{x₁}：水平轴的右上角x坐标 或者 垂直轴的左下角x坐标

{y₁}：水平轴的右上角y坐标 或者 垂直轴的左下角y坐标

{width}：反转线条的单位宽度 (单位:点)

示例：

```
! 0 200 200 210 1
CENTER
TEXT 24 0 0 45 SAVE
TEXT 24 0 0 95 MORE
INVERSE-LINE 0 45 145 45 45
INVERSE-LINE 0 95 145 95 45
PRINT
```

结果：



5-3 矩形指令 BOX

用户可以使用BOX指令来生成具有指定宽度的矩形

格式:

`{command} {x0} {y0} {x1} {y1} {width}`

解释:

`{command}`: BOX

`{x0}`: 左上角的横坐标

`{y0}`: 左上角的纵坐标

`{x1}`: 右下角的横坐标

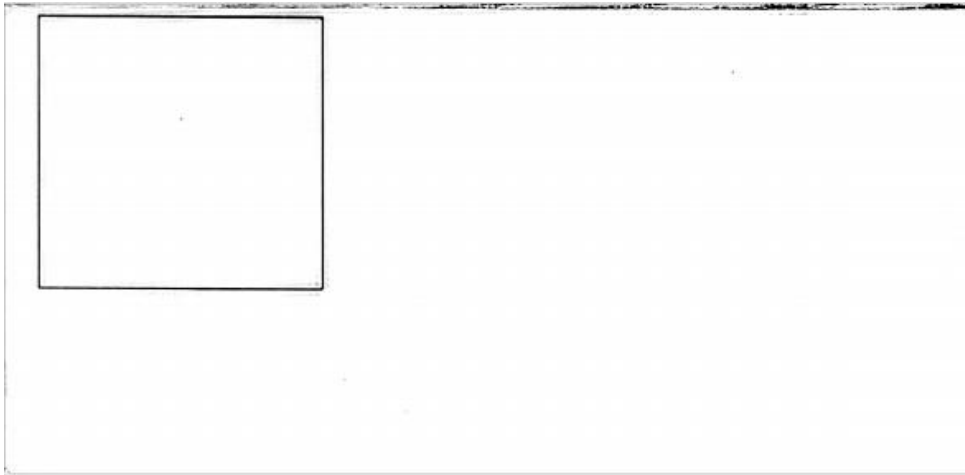
`{y1}`: 右下角的纵坐标

`{width}`: 矩形框线条的单位宽度

示例:

```
! 0 200 200 210 1
BOX 10 10 200 200 1
PRINT
```

结果:



5-4 位图指令

可以使用位图指令打印位映射图形。扩展图形数据使用ASCII十六进制字符进行表示。通过对十六进制数据的等效二进制字符使用COMPRESSED-GRAPHICS命令，可以将数据大小缩减一半。如果使用COMPRESSED-GRAPHICS(以下简称CG)，对于每8位图形数据，将会发送一个8位字符。如果使用EG，将使用两个字符(16位)来传输8位图形数据，因此EG的效率会减半，但是由于该数据是字符数据，因此比二进制数据更容易处理和传输。

格式:

$$\{\text{command}\} \quad \{\text{width}\} \quad \{\text{height}\} \quad \{x\} \quad \{y\} \quad \{\text{data}\}$$

解释:

`{command}`: 有如下几种选择

EXPANDED-GRAPHICS(或EG): 横向打印扩展图形

VEXPANDED-GRAPHICS(或VEG): 纵向打印扩展图形

COMPRESSED-GRAPHICS(或CG): 横向打印压缩图形

VCOMPRESSED-GRAPHICS(或VCG): 纵向打印压缩图形

{width}: 图像的宽度(以字节为单位)。

{height}: 图像的高度(以点为单位)。

$\{x\}$: 图像起始位置的横坐标

{y}: 图像起始位置的纵坐标

{data}: 图像数据

示例:

输入:

! 0 200 200 210 1

EG 2 16 90 45 F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0

F0F0F0F0F0F0F0F00F0F0F0F0F0F0F0F

PRINT

结果:

