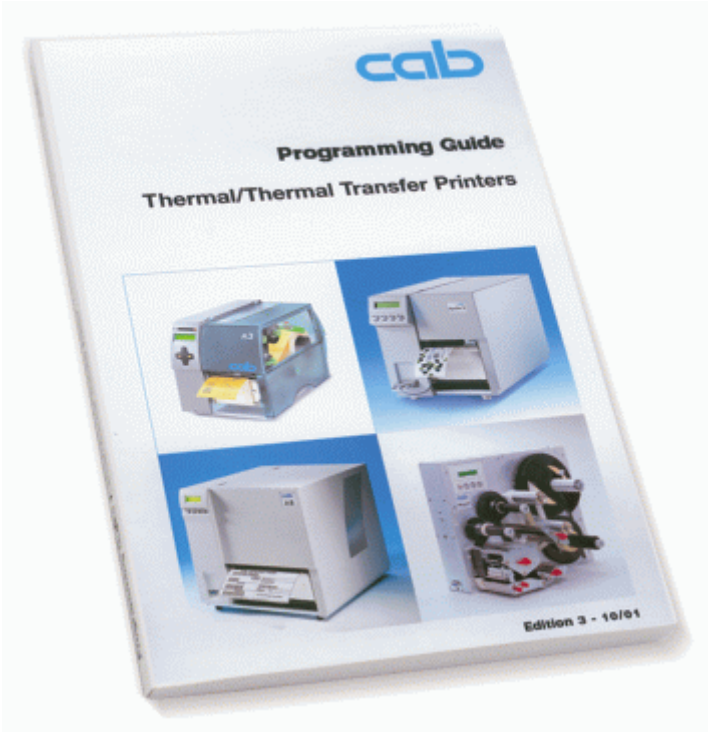

cab
Programming Guide
Thermal/Thermal Transfer Printers
Edition 3 - 10/01

cab 条码机程序手册

简介

cab 条码机可使用的三种型式指令：

- ESC 指令
- 小写英文字母的指令
- 大写英文字母的指令

ESC 指令

用来检测条码机状态、控制条码机、与内存管理，且 ESC 指令通常是会立刻执行，即使是条码机正在打印标签的状态；ESC 指令并非都用在打印卷标，还有其它功能，例如：

ESC ？ = 查询可用的内存状态
ESC c = 取消打印
ESC p0 = 结束条码机暂停状态

小写英文字母的指令

用来调整及设定条码机，与实际打印无关连，例如：

a = 条码机启动 ASCII 码打印模式
c = 条码机执行立刻裁切
f = 条码机执行进纸功能

大写英文字母的指令

用来编写标签格式，并依据固定结构与语法，例如：

J = 标签设计开始
S = 设定标签尺寸
H = 设定印字头工作温度、打印速度与打印方式
O = 设定打印方向
T = 卷标文字定义
B = 标签条码定义
G = 标签线条定义
I = 卷标图形定义
A = 标签打印数量

另外，cab 条码机还支持特殊功能指令，如 [DATE] 打印日期指令，与数据库连结指令，细节请参考后续解说。此程序手册所提供的范例是以公制毫米（mm）为单位，以 A3/300(dpi)打印机打印结果为图标，而标签软件编写都可以 Notepad、Wordpad 等文书编辑软件编写

ESC 指令 - ESCESC

该指令用在当下载图形或字型到条码机时，图形或字型可能会内含相同 ESC 指令，故以 ESCESC 替代单一 ESC（ASCII 27 或 Hex 1B 码）指令以避免无预期的错误反应；

语法： **ESCESC**

ESC 指令 - ESC！ESC！

该指令会强迫条码机重开机，功能如同手动开关条码机电源；

语法： **ESC！ESC！**

ESC 指令 - ESC*

该指令会透过 RS-485 联机启动所有连结的条码机，此时必须要所有的条码机皆有正确设定 RS-485 联机网络 ID 才可，最多可接 26 台条码机，RS-485 的网络地址代号为 A-Z；

语法： **ESC***

语法： **ESCA**

：

语法： **ESCZ**

ESC 指令 - ESC.

在传输图形或字型等二进制数据时，该指令可以二进制数据起始与结束值，此功能亦可透过 “cab cardmanager” 软件做数据传输的功能；

语法： **ESC.**

ESC 指令 - ESC:

该指令用在二进制数据的叙述，通常在下载数据到条码机时，需要先将数据转换，因此便需要起始顺序，再接二进制数据，最后再接 ESCend-of-data，而二进制数据不能包含任何 ESC 字符，否则会被系统错误解译；

该指令不能用于网络，下载二进制数据最好方式是使用 ESC. 指令；

语法： **ESC:**

ESC 指令 - ESC?

该指令会查询条码机内存缓冲区的使用情形，会以 0-9 数值回传状态；

语法： **ESC?**

数值		内存使用率
0	=	0-9%
1	=	10-19%
2	=	20-29%
3	=	30-39%
4	=	40-49%
5	=	50-59%
6	=	60-69%
7	=	70-79%
8	=	80-89%
9	=	90-99%

ESC 指令 - ESCc

该指令会取消条码机当时的打印动作，相当于按下条码机 LCD 控制面版上的 CANCEL 键；

语法： **ESCc**

ESC 指令 - ESCend-of-data

该指令用在二进制数据的叙述的结束，搭配 ESC：指令使用，但不能用于 RS-485 网络；

语法： **ESCend-of-data**

ESC 指令 - ESCf

该指令会执行条码机的进纸动作，相当于按下条码机 LCD 控制面版上的 FEED 键；

语法： **ESCf**

ESC 指令 - ESCp0

该指令会终止条码机的暂停模式，此时条码机 LCD 控制面版上的 PAUSE LED 指示灯会熄灭，并继续未完成的打印动作，或是结束条码机错误状态，如同按下 LCD 控制面版上的 PAUSE 键；

语法： **ESCp0**

ESC 指令 - ESCp1

该指令会使条码机立刻进入暂停模式，如同按下 LCD 控制面版上的 PAUSE 键，此时条码机 LCD 控制面版上的 PAUSE LED 指示灯会亮起，并暂停打印动作；

语法： **ESCp1**

ESC 指令 - ESCt

该指令会使条码机完全取消所有打印，并清除条码机缓冲区的数据，如同按下 LCD 控制面版上的 CANCEL 键，与 ESCc 指令不同处在于 ESCc 仅取消当时的打印动作，并未清除条码机缓冲区的数据；

语法： **ESCt**

ESC 指令 - ESCs

该指令会透过 RS-232 条码机连结，查询条码机状态并回传状态讯息，

语法： **ESCs**

回传讯息：**XYNNNNNNZ**

说明：**X** = 条码机的联机状态，

Y= 条码机与计算机为联机状态，N= 未联机

Y = 条码机错误代号

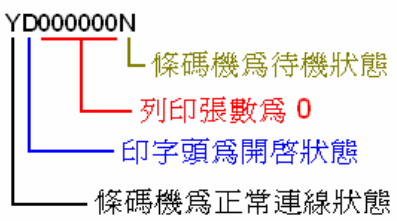
NNNNNN = 要打印的标签数量

Z = 条码机打印状态，Y=正在处理打印, N=条码机处于待机模式

其中 **Y** 的代号意义如下：

代号	说明	解决方式
-	无错误	
B	通讯协议错误	确定计算机与条码机的通讯协议设定皆为相同。
C	记忆卡错误	重新格式化记忆卡，再存入标签档案。
D	印字头为开启状态	关闭印字头。
E	同步化错误（无标签）	检查卷标纸与卷标感应器是否对准卷标与卷标之间的间距位置。
F	碳带用完	更换碳带。
H	印字头加热电压问题	关闭电源再重新开机，若还有相同问题，请联络专业人员服务。
O	内存不足	清除内存或重新开机。
P	标签纸用完	更换标签纸卷。
V	输入缓冲溢位	重新开机。
W	印字头过热	暂停打印一段时间。
X	外部 输入/输出 错误	检查外接装置。
Z	印字头损毁	清洁或更换印字头。
n	网络错误	检查网络设定与硬件安装。
u	USB错误	检查USB装置与连接。

範例：



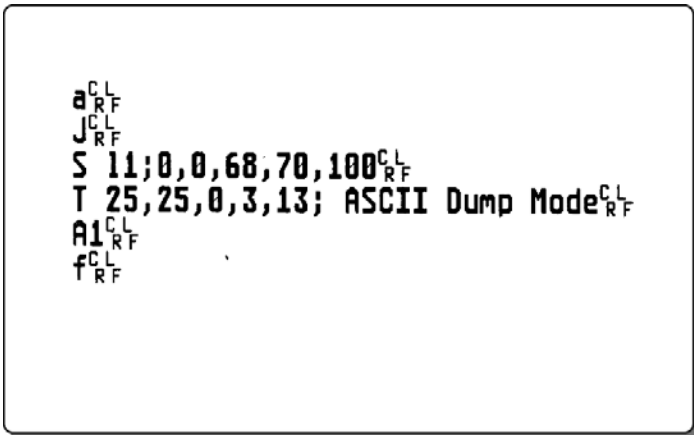
小写英文字母的指令

a - ASCII 码打印模式

语法： **a**

範例：

```
a
J
S 11,0,0,68,70,100
T 25,25,0,3,13;ASCII Dump Mode
A1
f
```



c - 立刻裁切

语法： **c**

c = 条码机立刻执行裁切动作，如条码机未装裁刀，则会在 LCD 上显示 “Protocol error” 的错误讯息；

e - 清除资料

语法： **e type;name**

e = 清除条码机内存数据（如图形或字型等），但此指令并不会清除条码机的记忆卡数据；

type= 清除档案的格式，如 BMP、FNT、GIF、IMG、MAC、PCX、PNG、TIF、TTF 等档案

name= 档案的名称，如使用 * 作为档名，则会删除所有相同格式的档案！

範例：

E FNT;*

说明：删除条码机内存内所有的 True Type 字型；

f - 条码机进纸

语法：**f**

f = 执行条码机进纸动作的指令，如同按下控制面版上的 FEED 键；

m - 设定长度单位

语法：**m t**

m = 设定长度单位指令

t = 单位选择，“m” 是公制(毫米，mm)，“i” 是英制（英吋，inches）

p - 暂停条码机

语法：**pn**

p = 条码机暂停指令

n = 0：取消暂停

1：启动暂停

启动条码机暂停时，相当于按下条码机的 PAUSE 键，此时 PAUSE 的 LED 指示灯会亮起，如当时有打印标签，则会立刻暂停打印动作；当取消暂停时，PAUSE 的 LED 指示灯会熄灭，如当时有被暂停打印标签，则会恢复打印动作。

大写英文字母的指令

A– 标签数量

A 指令是用来定义卷标打印数量，在条码机执行打印时，标签设计会储存在条码机的内部存储器缓冲区直到完成打印动作；

语法： **A n**

n = 标签数量，如未输入数目，则会无限打印；

其它有效参数

[NOPRINT] = 条码机回接收并处理卷标数据，但无打印动作，此目的是将卷标数据储存于条码机内存，亦可使用 [NO] 代替 [NOPRINT] 指令；

[?] = 条码机会在 LCD 面版上显示要求输入打印数量；

[REPEAT] = 在打印工作结束时，重复循环卷标数据，此功能常与 [?] 结合使用，亦可以 [R] 代替 [REPEAT] 指令

[$DBF] = 打印数据库的每笔记录，数据库的总记录数目就是卷标打印张数

範例：

```
J
S l1;0,0,68,71,100
T 25,10,0,5,8;LABEL PRINTER
A 550
```

说明：打印 550 张标签

範例：

```
J
S l1;0,0,68,71,100
T 25,10,0,5,8;LABEL PRINTER
A
```

说明：

A 指令后面不加任何指令，则会打印无限张数的标签，直到使用者输入取消讯号或按下 LCD 控制面版上的取消键（Cancel）为止

範例：

```
J
S l1;0,0,68,71,100
T 25,25,0,3,8;Suppress Printout
A [NOPRINT]
```

说明：

利用 [NOPRINT] 指令将卷标数据传送到条码机内存缓冲区

範例：

```
J
S l1;0,0,68,71,100
T 25,25,0,3,8;[?:Input?]
A [?,R]
```

说明：

此卷标程序会在条码机上的 LCD 控制面版上显示要求操作者输入卷标数据（[?:Input?]），然后会再显示要求操作者输入卷标打印张数，完成输入后才会开始打印卷标，所有数据输入可由 LCD 控制面版的按键输入，或由外接键盘输入。

範例：

```
m m
J
S l1;0,0,68,73,100
E DBF;CDPLAYER
T:IDX;25,225,0,3,5;[SER:100]
T0,40,0,3,6;>>[DBF:TYP,typ,NAME]<<
A [$DBF]
```

说明：

打印数据库 CDPLAYER.DBF 里所有的纪录，序列号指令（[SER:100]）会建立索引文件，从 100 开始起跳。

B - 条码

条码定义

各条码参数不尽相同，依条码类型而定，条码方向可为 0°、90°、180°、270°，高度与宽度皆可调整，亦可加入文字以利人工判读。

语法： **B [:name;] x,y,r,type[+options],size;text**

B	=	定义条码指令
[:name;]	=	定义数据内容于唯一的数据名
x	=	X 轴位置
y	=	Y 轴位置
r	=	旋转角度，0~359 度间隔 1°
type	=	条码类型
[+options]	=	附加参数
size	=	条码高度、宽度与比例
text	=	条码内容

解释如下

B

条码指令，指令后接参数设定与条码内容；

[:name;]

定义数据内容于唯一的数据名，供后续程序呼叫或其它用途(如数学运算)使用，最长 10 个字符，且不能使用特殊字符，其中数据名有字母大小之区别；

x

条码水平 X 轴起始位置，单位为 mm 或英吋；

y

条码垂直 Y 轴起始位置，单位为 mm 或英吋，最大坐标会依条码机类型而定；

r

旋转角度，可为 0°、90°、180°、270°；

type

条码类型，如以英文字母大写定义条码类型，则条码会附带人工判读字符，如以小写字母定义条码类型，则不会有人工判读字符的产生，人工判读字符的大小会依不同条码类型而有所不同，细节部分可参考后面范例；

cab 条码机能从条码名称里提取重点部分以判定条码类型，如指令行里为 EAN-13、EAN 13 或 EAN13 通通都会判定为 EAN 13 码；

[+options]

依据条码类型可有不同附加功能，不同条码有的不同附加功能会在各条码解说里详细叙述，有效的附加功能如下：

+MODxx

可在条码里加入检查码

MOD10 加入检查码 10

MOD11 加入检查码 11

MOD43 加入检查码 43

MOD16 加入检查码 16

+WSarea

白色空白区域-可打印（保留）白色区域，以便让打印之条码有更好的扫瞄判读性；

+BARS

在条码上下方打印直线的边线；

+XHRI

39 码：（附加人工可读的内容）添加起始与结束字符（＊）；

93 码：添加起始与结束字符（□）；

减小 UPC-A 与 UPC-E 的大小

+NOCHECK

隐藏专为 EAN-13 与 UPC-A 码的特别起始数字 21、24…29 设计的变量重量条码的检查计算码；

+ELx

错误层级，设定 PDF417 码的冗位，x 的有效值为 0~8；

DtatMatrix 码可打印成矩形或正方形，默认值为正方形，可用+RECT 指令打印成矩形。

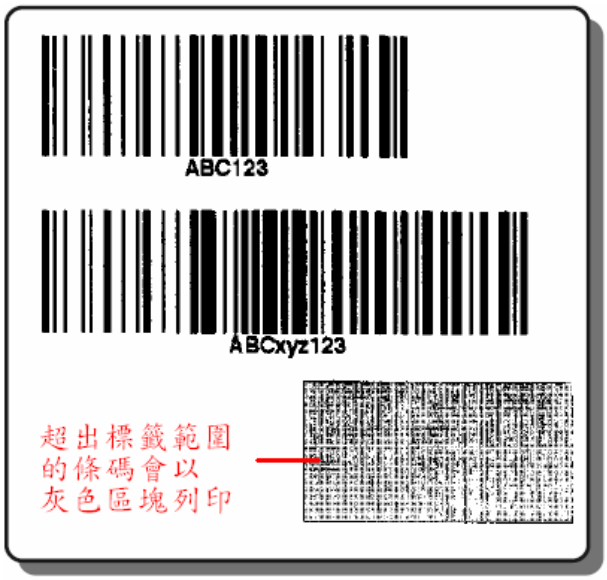
size

定义条码高度与宽度，EAN、JAN 及 UPC 码亦可以 SCx 定义标准码大小，高度计算在开启人为判读的字符时，包含该字符高度；

height - 高度

以预设单位-毫米(mm)或英吋(inches)定义条码高度，如条码大小（含

空白区域）超出卷标纸时，A 系列条码机会将该条码打印成灰色区块，如下图所示：



narrow element (ne) - 窄度

以毫米或英吋单位定义条码最小窄度，窄度大小会依条码机分辨率而不同；

ratio - 比例

比例为条码窄度与宽的比，如 3：1 表示宽的码条为小码条的三倍宽；

SCx

SC 为标准条码尺寸，EAN 与 UPC 码为统一尺寸；

text

依所选择的条码类型所包含条码的数据，不同条码有不同条码数据定义，例如，部分条码仅能使用数字，部分条码则有固定数据长度；

A 系列条码机会将超出标签纸的条码印呈灰色区块，可方便人为判别该条码为有问题的条码，以便免后续的读码问题。

B - 条码

条码总览表

不同条码会有不同的尺寸比例，条码名称为大写表示会有人人判读的文字行，此部分可参考该条码的规范；

Barcodename	Ratio	1D /2D code*	A-series	Apollo	Hermes	M-series
2 of 5 Interleaved	yes	1D	yes	yes	yes	yes
Add-On 2	no	1D	yes	yes	yes	yes
Add-On 5	no	1D	yes	yes	yes	yes
Aztec Code	no	2D	yes	-	-	yes
Codabar	yes	1D	yes	yes	yes	yes
Codablock F	no	stacked	yes	-	-	yes
Code 39	yes	1D	yes	yes	yes	yes
Code 93	no	1D	yes	yes	yes	yes
Code 128	no	1D	yes	yes	yes	yes
Data Matrix	no	2D	yes	yes	yes	yes
DBP (German Post code)	yes	1D	yes	yes	yes	yes
EAN 8	no	1D	yes	yes	yes	yes
EAN 13	no	1D	yes	yes	yes	yes
EAN 128	no	1D	yes	yes	yes	yes
FIM	no	1D	yes	yes	yes	yes
German Parcel	yes	1D	yes	yes	yes	yes
JAN 8	no	1D	yes	yes	yes	yes
JAN 13	no	1D	yes	yes	yes	yes
HIBC	yes	1D	yes	yes	yes	yes
MaxiCode	no	2D	yes	yes	yes	yes
Micro PDF	no	2D	yes	-	-	yes
MSI	yes	1D	yes	yes	yes	yes
PDF-417	no	2D	yes	yes	yes	yes
Plessey	yes	1D	yes	yes	yes	yes
Postnet	no	1D	yes	yes	yes	yes
QR -Code	no	2D	yes	-	-	yes

1D 表示一维条码，2D 表示二维条码

Barcodename	Ratio	1D /2D code*	A-series	Apollo	Hermes	M-series
RSS-14		1D	yes	-	-	yes
RSS-14 composite CC-A		1D+2D	yes	-	-	yes
RSS-14 composite CC-B		1D+2D	yes	-	-	yes
RSS-14 truncated		1D	yes	-	-	yes
RSS-14 truncated composite			yes	-	-	yes
RSS-14 truncated composite			yes	-	-	yes
RSS-14 stacked			yes	-	-	yes
RSS-14 stacked composite			yes	-	-	yes
RSS-14 stacked composite			yes	-	-	yes
RSS-14 stacked omnidirectional			yes	-	-	yes
RSS-14 stacked omnidirectional composite			yes	-	-	yes
RSS-14 stacked omnidirectional composite			yes	-	-	yes
RSS limited			yes	-	-	yes
RSS limited composite			yes	-	-	yes
RSS limited composite			yes	-	-	yes
RSS expanded			yes	-	-	yes
RSS expanded composite			yes	-	-	yes
RSS expanded composite			yes	-	-	yes
RSS expanded stacked			yes	-	-	yes
RSS expanded stacked half line			yes	-	-	yes
RSS expanded stacked composite (CC-A)			yes	-	-	yes
RSS expanded stacked composite (CC-B)			yes	-	-	yes
UCC 128	no	1D	yes	yes	yes	yes
UPC-E0	no	1D	yes	-	-	yes
UPC-A	no	1D	yes	yes	yes	yes
UPC-E	no	1D	yes	yes	yes	yes

此程序手册不包含 Aztek、Codablock、与 RSS 码！

B - 条码

每个条码都有个别条码数据与型式的定义，建议使用者先了解个别条码的定义规范，同时也建议先测试打印出来条码的可读性！

有效的检查码：
MOD 10 （仅数字数据）
MOD 10 （MSI 码用以计算差值（weighting 为 2/1 而非 3/1）
MOD 10 GP（2 of 5，weighting 3/1 + 1，仅用在德国包裹）
MOD 11 （仅数字数据）
MOD 16 （仅 Codabar）
MOD 43 （仅 39 码与 128 码）

Code 128 与 EAN/UCC-128 码会自动使用 103 检查码；
EAN-13、EAN-8、UPC-A、与 UPC-E0 会自动使用检查码 10；

POSTENT 会自动使用检查码 10（无 weighting）；
DBP 为 Deutsche Post AG 的 12 或 14 码条码，会自动使用检查码 10（weighting 4/9），可依要求加入空白与点。

B - 条码　　2 of 5 Interleaved

语法：　**B** [:name;] x,y,r,**2OF5INTERLEAVED**[+options],height,ne,ratio;text

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+MODxx

可加入检查码到条码里；

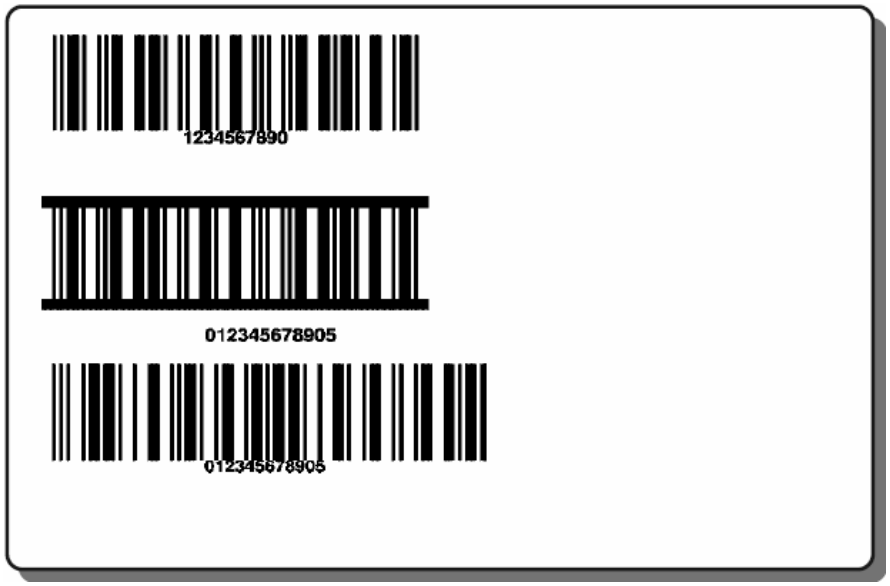
+BARS

在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

建议该条码使用固定长度，并设定条码扫描仪为相同的条码长度，以确保最佳的条码判读性。

範例：

```
J
S l1;0,0,68,71,100
B 5,5,0,2 OF 5 INTERLEAVED,10,.3,3;1234567890
B 5,20,0,2of5interleaved+BARS,10,.3,3;1234567890
B:Bar3;5,35,0,2OF5 INTERLEAVED+MOD10,10,.3,3;1234567890
A 1
```



B - 条码　　Add-On2

语法：　**B [:name;] x,y,r,ADDON2,[+options],height,ne;text**

[+options]=

+BARS

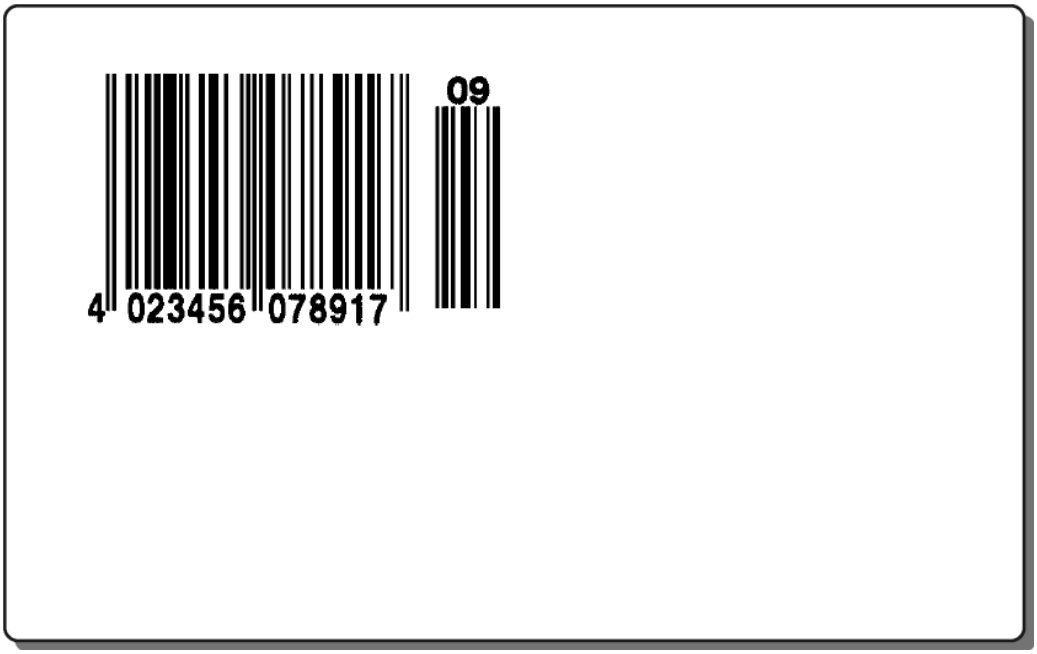
在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

SCx

设定条码大小，x 为 0~9 数值，实际条码大小会因条码机分辨率而不同；

範例：

```
J
S l1;0,0,68,71,100
B 10,5,0,EAN13  ,SC2;402345607891
B 45,5,0,ADDON2,SC2;09
A 1
```



B - 条码　　Add-On5

语法：　**B [:name;] x,y,r,ADDON2,[+options],height,ne;text**

[+options]=

+BARS

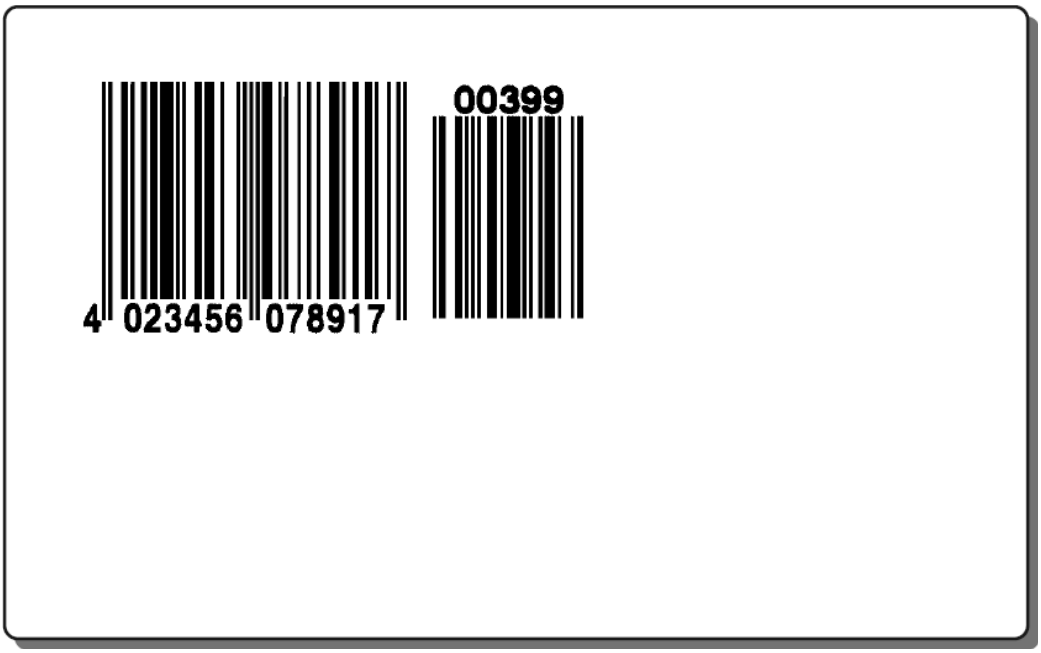
在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

SCx

设定条码大小，x 为 0~9 数值，实际条码大小会因条码机分辨率而不同；

範例：

```
J
S l1;0,0,68,71,100
B 10,5,0,EAN13, SC2;402345607891
B 45,5,0,ADDON5,SC2;00399
A 1
```



B - 条码　　Codabar

语法：　**B [:name;] x,y,r,CODABAR[+options],height,ne,ratio;text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+MODxx

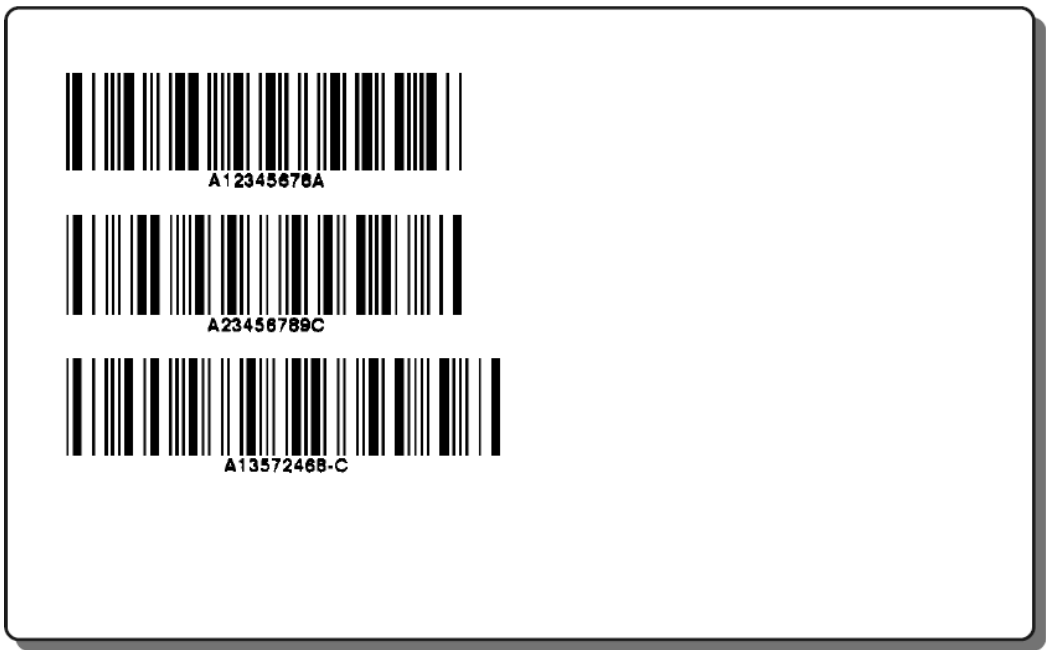
可加入检查码到条码里；

+BARS

在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

範例：

```
J
S l1;0,0,68,71,100
B 5, 5,0,CODABAR,       12,.3,3;A12345678A
B 5,20,0,CODABAR, 12,.3,3;A23456789C
B 5,35,0,CODABAR+MOD16,12,.3,3;A13572468C
A 1
```



B - 条码　　Code 39

语法：　**B [:name;] x,y,r,CODABAR[+options],height,ne,ratio;text**

[+options]=

+WSarea

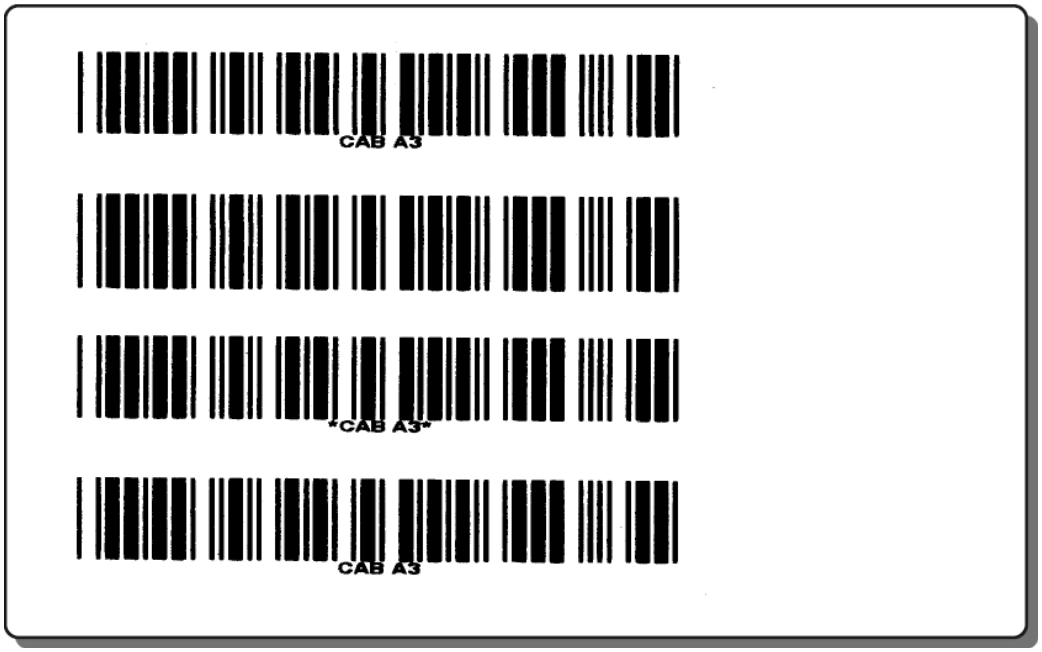
在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+XHRI

人工判读内容，并加入起始与结束字符；

範例：

```
J
S l1;0,0,68,71,100
B 5, 5,0,CODE39,10,0.3,3;CAB A3
B 5,20,0,code39,10, .3,3;CAB A3
B 5,35,0,CODE39+XHRI,10,0.3,3;CAB A3
B 5,50,0,CODE39,10,.3,3;cab A3
A 1
```



B - 条码　　Code 93

语法：　**B [:name;] x,y,r,CODE93,[+options],height,ne;text**

[+options]=

+BARS

在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

+XHRI

人工判读内容，并加入起始与结束字符（□）；

範例：

```
J
S l1;0,0,68,71,100
B 5,5,0,CODE93+XHRI,16,.28,3;ABC123
B 5,24,0,code93,16,.28,3;ABC123
B 5,44,0,CODE93+BARS,16,.28,3;ABC123
A 1
```

B - 条码　　Code 128

语法：　**B** [:name;] x,y,r,**CODE128**[+options],height,ne,ratio;[U:subcode]text

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+MODxx

可加入检查码到条码里；

+BARS

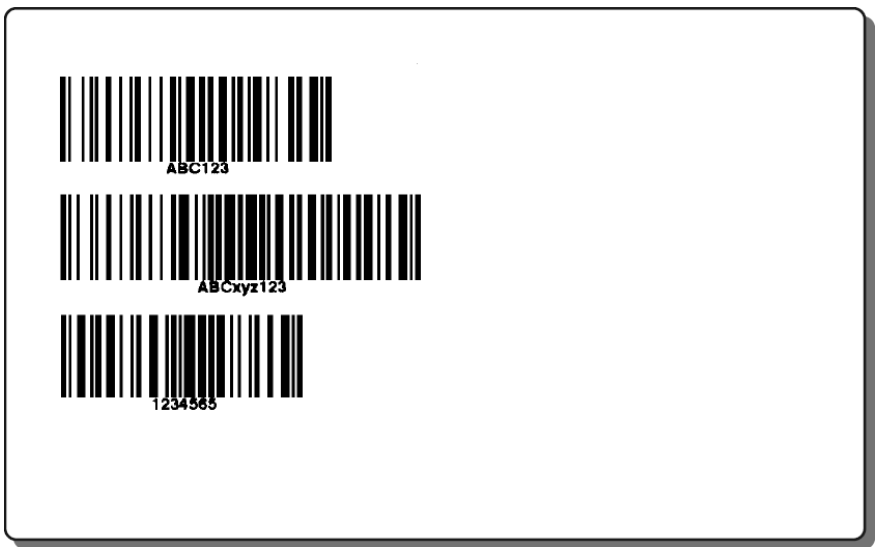
在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

[U:subcode]

可选用特定的条码子编码；

範例：

```
J
S l1;0,0,68,71,100
B 5, 5,0,CODE128, 12,.3;ABC123
B 5,20,0,CODE 128,12,.3;ABCxyz123
B 5,35,0,CODE128+MOD10, 12,.3;[U:CODEC]123456
A 1
```

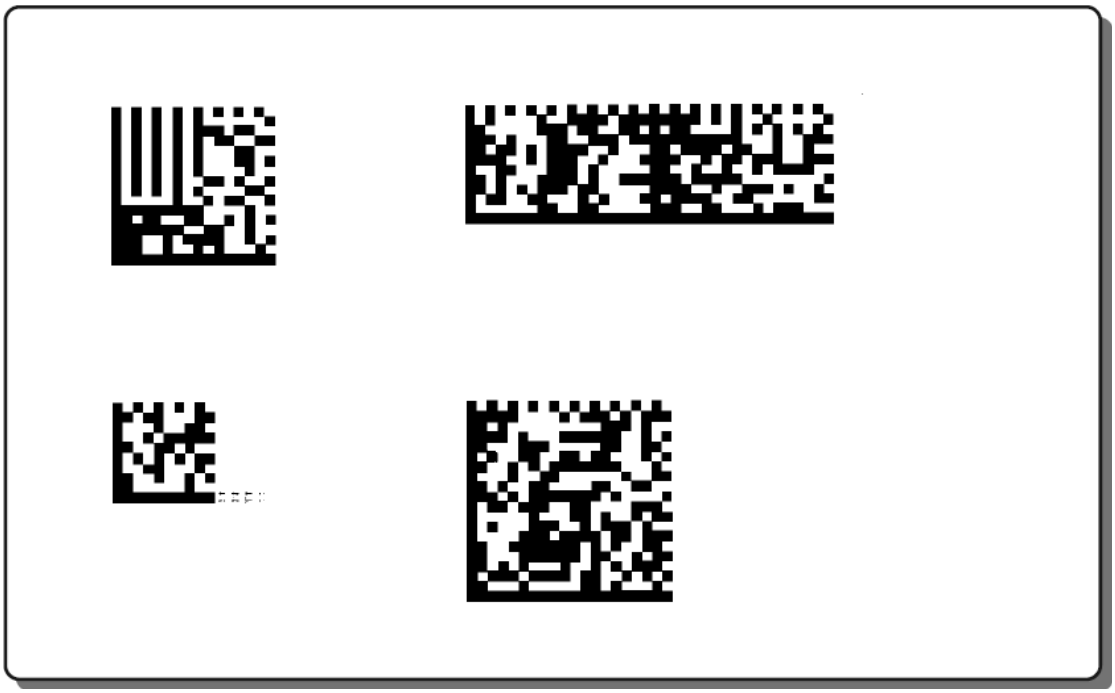


B - 条码　　Data Matrix

语法：　**B [:name;] x,y,r,DATAMATRIX [+RECT],height;text**

範例：

```
J
S l1;0,0,68,71,100
B 25, 5,0,DATAMATRIX,1;30Q324343430794<OQQ
B 60, 5,0,DATAMATRIX+RECT+WS2,1;cab Produkttechnik
B 25,35,0,DATAMATRIX,1;[U:PROG]
B 60,35,0,DATAMATRIX+WS2,1;[U:ANSI_AI]cabProdukttechnik
A 1
```



B - 条码　　DBP-German Post Identcode

语法：　**B [:name;] x,y,r,DBP[+options],height,ne,ratio;text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

範例：

```
J
S l1;0,0,68,71,100
B  5,10,0,DBP,10,.3;2134807501640
B 60,10,0,DBP,10,.3;56.310.243.031
A 1
```

B - 条码　　EAN-8 / JAN-8

语法：　**B [:name;] x,y,r,EAN8[+options],height,ne;text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+XHRI

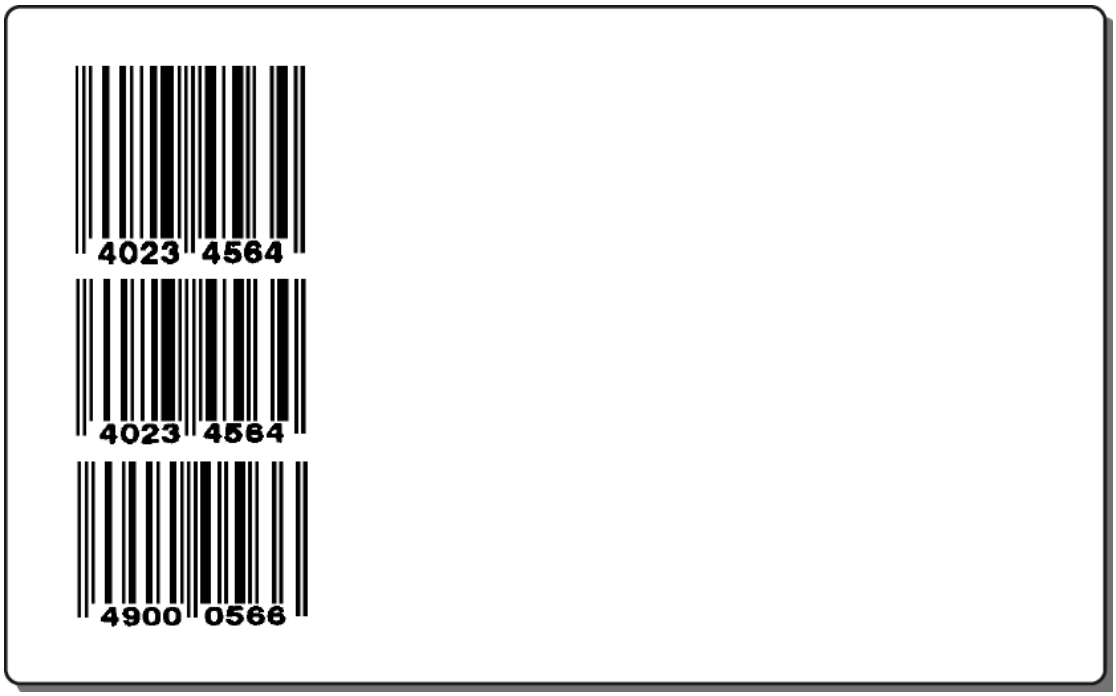
人工判读内容，并缩小条码大小（参见范例）；
高度与窄度可以 SC 参数设定；

SCx

设定条码大小，x 为 0~9 数值，实际条码大小会因条码机分辨率而不同；

範例：

```
J
S l1;0,0,68,71,100
B 10,  5,0,EAN8,    SC1;4023456
B 10,26,0,EAN8,16,.35;4023456
B 10,44,0,JAN8,16,.35;4900056
A 1
```



B - 条码　　EAN-13 / JAN-13

语法：　**B [:name;] x,y,r,EAN13[+options],height,ne;text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+XHRI

人工判读内容，并缩小条码大小（参见范例）；

+NOCHECK

隐藏 EAN-13 码的特别起始数字：21、24…29 之变量重量条码的检查计算码；

高度与窄度可以 SC 参数设定；

SCx

设定条码大小，x 为 0~9 数值，实际条码大小会因条码机分辨率而不同；

範例：

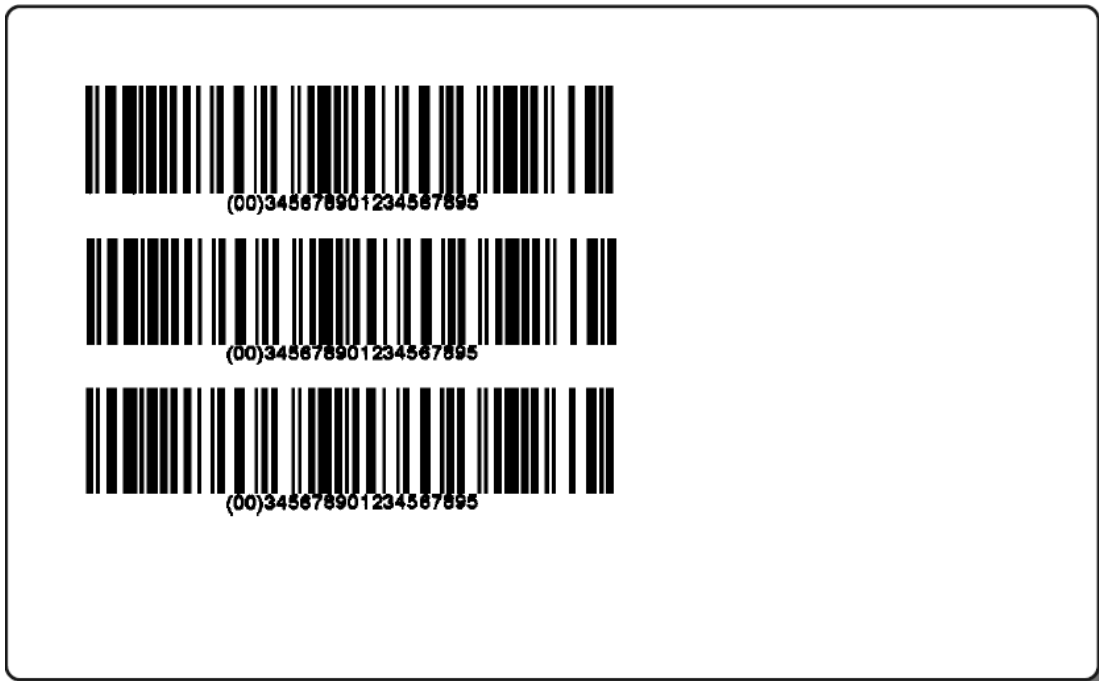
```
J
S l1;0,0,68,71,100
B 10, 5,0,EAN13,   SC1;402345607891
B 10,30,0,EAN13,16,.35;270072610950
B 10,48,0,JAN13,16,.35;490005607891
A 1
```

B - 条码　　EAN-128 / UCC 128

语法：　**B [:name;] x,y,r,EAN128[+options],height,ne;text**

範例：

```
J
S l1;0,0,68,71,100
B 5,  5,0,EAN128,12,.3;(00)345678901234567890
B 5,20,0,UCC128,12,.3;(00)345678901234567890
B 5,35,0,EAN128,     12,.3;(00)345678901234567890
A 1
```



B - 条码　　FIM

语法：　**B [:name;] x,y,r,FIM,[+options],height,ne;text**

[+options]=

+WSarea

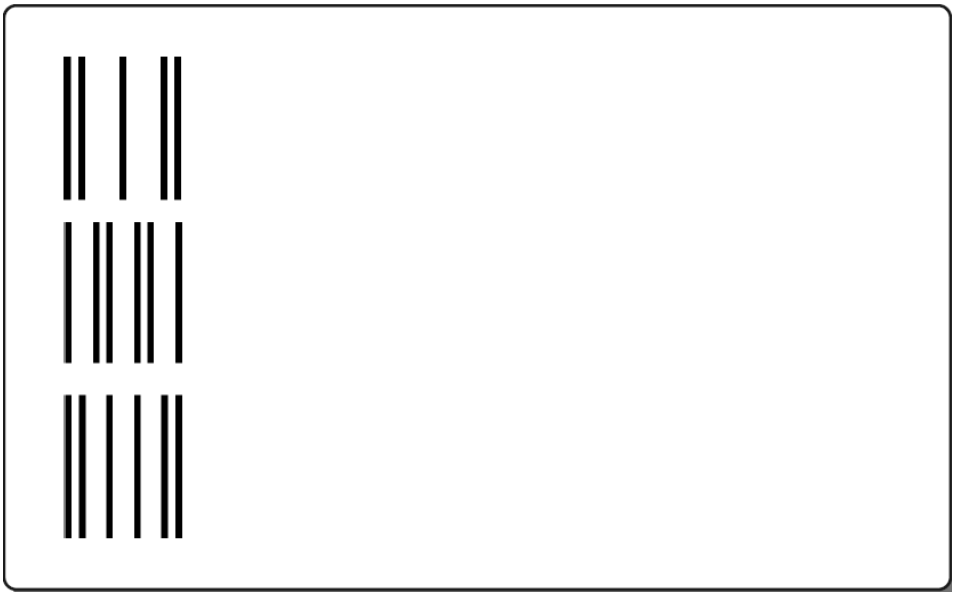
在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+BARS

在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

範例：

```
J
S l1;0,0,68,71,100
B 5, 5,0,FIM,16,.3,3;A
B 5,24,0,FIM,16,.3,3;B
B 5,44,0,FIM,  16,.3,3;C
A 1
```



B - 条码　　HIBC (Health Industry Barcode)

语法：　**B [:name;] x,y,r,HIBC[+options],height,width,ratio;text**

[+options]=

+WSarea

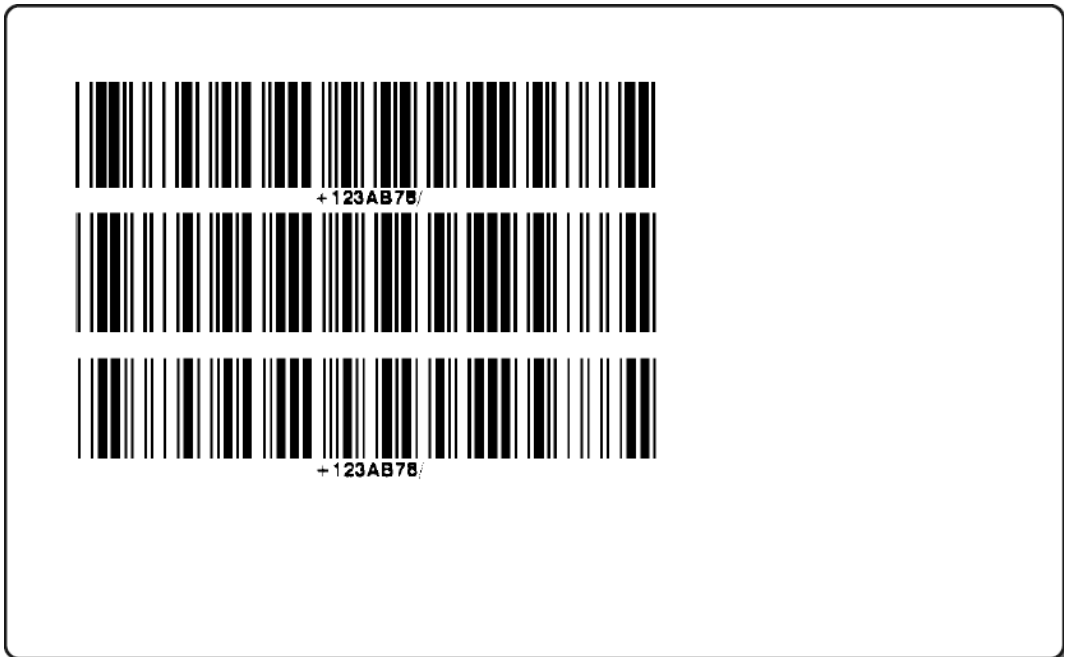
在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+BARS

在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

範例：

```
J
S l1;0,0,68,71,100
B 5, 5,0,HIBC,12,.3,3;+123AB78
B 5,18,0,hibc,12,.3,3;+123AB78
B 5,33,0,HIBC,   12,.3,3;+123AB78
A 1
```



B - 条码　　Maxicode

语法：　**B [:name;] x,y,r,MAXICODE[+MODE];[ZIPCODE],[COUNTRY]…**

範例：

```
J
S l1;0,0,68,71,100
B 25,5,0,Maxicode+MODE2;76131,260,999,Paket for cab
Produkttechnik GmbH
B 60, 5,0,Maxicode+ws2+mode4;MaxiCode (19 charcters)
B 25,35,0,Maxicode+MODE4;Paket for cab Produkttechnik GmbH
B 60,35,0,Maxicode+MODE6;Paket for cab Produkttechnik GmbH
A 1
```

B - 条码　　Micro PDF 417

语法：　**B [:name;] x,y,r,Micro[+COLSx],height,ne,ratio;text**

COLSx = 设定 PDF 码的数据行数，x=1-4

Micro PDF417 提供三种编码方式：

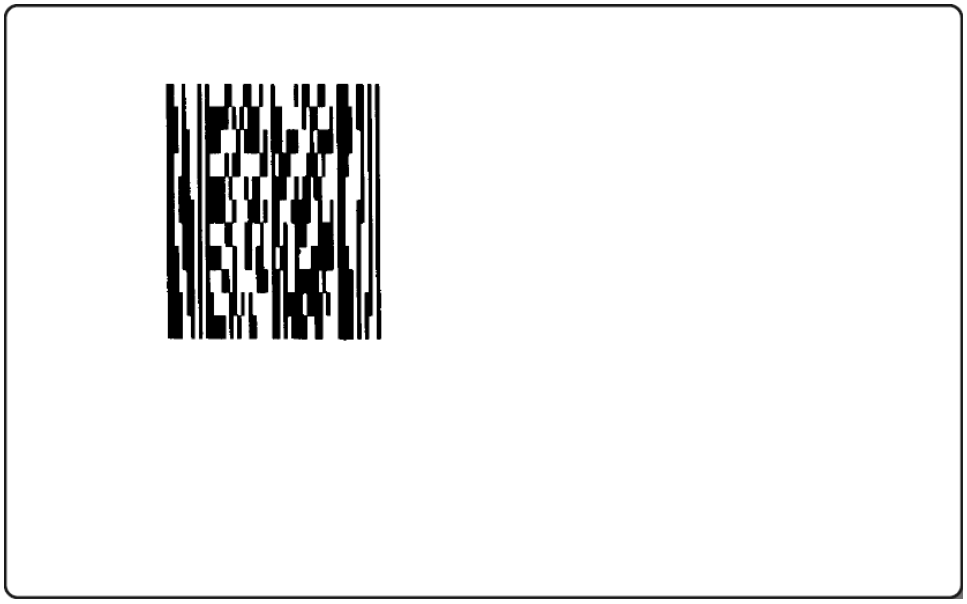
文字压缩模式 mode0：250 个字符，每 2 个数据字符压缩成一个码

位压缩模式 mode1：150 个字符，每 1.2 个数据字符压缩成一个码

数值压缩模式 mode2：366 个字符，每 2.93 个数据字符压缩成一个码

範例：

```
J
S 0,0,68,71,100
B 10,10,0,Micro+COLS2,3,.5;cab Produkttechnik
A 1
```



B - 条码　　MSI (MSI Plessey)

语法：　**B [:name;] x,y,r,MSI[+options],height,ne,ratio;text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+MODxx

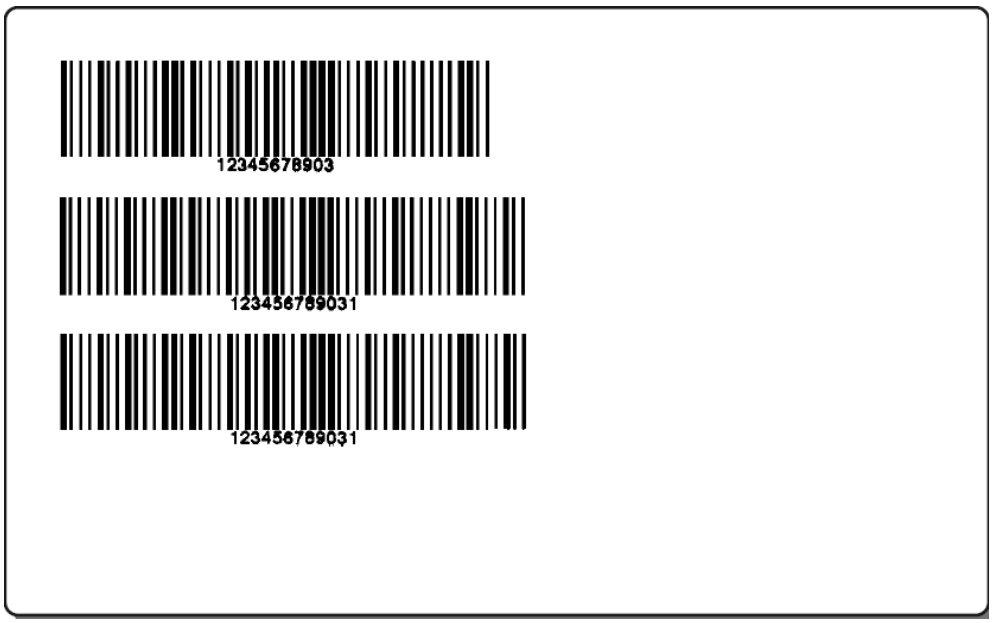
可加入检查码到条码里；

+BARS

在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

範例：

```
J
S l1;0,0,68,71,100
B 5, 5,0,MSI,12,          .3,2;1234567890
B 5,20,0,MSI+MOD10,12,.3,2;1234567890
B 5,35,0,MSI+MOD11,12,.3,2;1234567890
A 1
```



B - 条码　　PDF417

语法：　**B [:name;] x,y,r,PDF417[+WSarea],[+ELxx],height,ne,ratio;text**

[+options]=

+WSarea

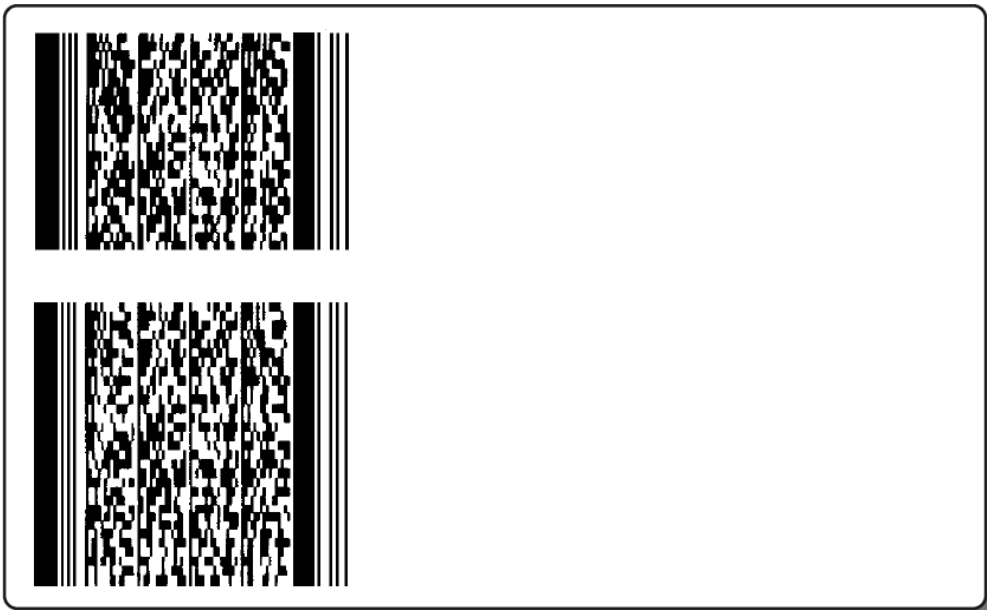
在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+ELxx

错误层级由此数值设定；

範例：

```
J
S l1;0,0,68,71,100
B 2, 5,0,PDF417+EL0,.1,.38,1;cab Produkttechnik
GmbH[U:13][U:10]Wilhelm Schickard Strasse[U:13][U:10]D-76131
Karlsruhe
B 2,35,0,PDF417+EL3,.1,.38,1;cab Produkttechnik
GmbH[U:13][U:10]Wilhelm Schickard Strasse [U:13][U:10]D-76131
Karlsruhe
A 1
```



B - 条码　　Plessey

语法：　**B [:name;] x,y,r,PDF417[+WSarea],[+ELxx],height,ne,ratio;text**

[+options]=

+WSarea

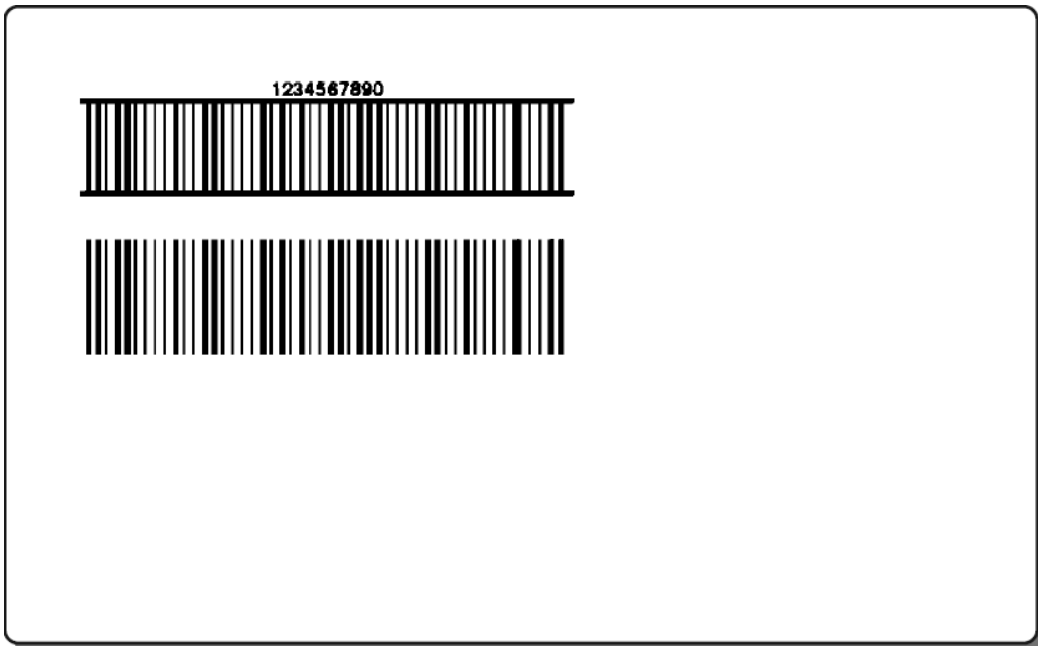
在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+BARS

在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

範例：

```
J
S l1;0,0,68,71,100
B 5,20,0,PLESSEY+BARS,12,.3,2;1234567890
B 5,35,0,plessey,     12,.3,2;1234567890
A 1
```



B - 条码　　Postnet

语法：　**B [:name;] x,y,r,POSTNET,[+options];text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

範例：

```
J
S l1;0,0,68,71,100
B 10, 5,0,postnet,20,.35;442120798
B 10,20,0,POSTNET,      20,.35;441361234
A 1
```

B - 条码　　QR-Code

语法：　**B [:name;] x,y,r,QRCODE[+ELx][+MODELx],size;text**

+ELx

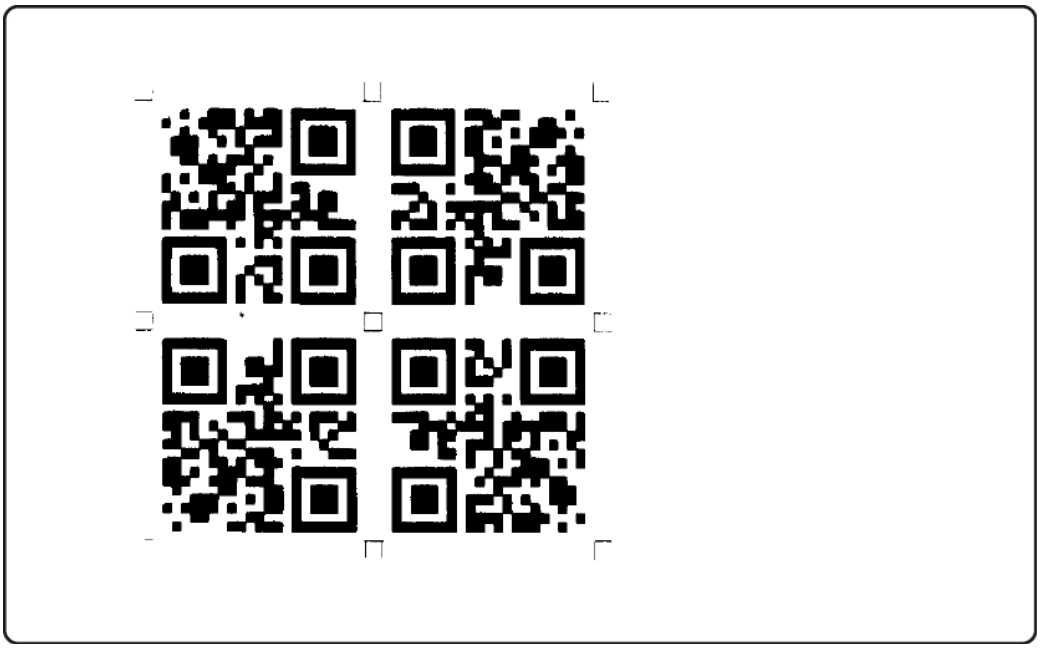
错误层级 – 有效数值 x：1-4, L, M, Q, H，默认值为 1

+MODELx

有效输入值 x 为 1 或 2，默认值为 1

範例：

```
J
S l1;0,0,68,71,104
B 52,32,0,QRCODE+ELL+MODEL2+WS2,1;Hello world!
B 52,28,90,QRCODE+ELL+MODEL2+WS2,1;Hello world!
B 48,28,180,QRCODE+ELL+MODEL2+WS2,1;Hello world!
B 48,32,270,QRCODE+ELL+MODEL2+WS2,1;Hello world!
G 0,0,0;L:104,3
G 0,65,0;L:104,3
H 150,-5,T
A 5
```



B - 条码　　UPC-A

语法：　**B [:name;] x,y,r,UPCA[+options],height,ne;text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+XHRI

人工判读内容，并缩小条码大小（参见范例）；

+NOCHECK

隐藏 UPC-A 码的特别起始数字：21、24…29 之变量重量条码的检查计算码；

高度与窄度可以 SC 参数设定；

SCx

设定条码大小，x 为 0~9 数值，实际条码大小会因条码机分辨率而不同；

範例：

```
m m
J
O R
S l1;0,0,68,71,100
B 10,5,0,UPC-A,20,.35;01234554321
B 10,30,0,UPCA+XHRI,SC1;01234554321
A 1
```

B - 条码　　UPC-E

语法：　**B [:name;] x,y,r,UPCE[+options],height,ne;text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+XHRI

人工判读内容，并缩小条码大小（参见范例）；

SCx

设定条码大小，x 为 0~9 数值，实际条码大小会因条码机分辨率而不同；

範例：

```
J
S l1;0,0,68,71,100
B 10,  5,0,UPC-E,20,.35;0123456
B 10,30,0,UPCE+XHRI,SC1;0123456
A 1
```

B - 条码　　UPC-E0

语法：　**B [:name;] x,y,r,UPCE0,height,ne;text**

[+options]=

+WSarea

在卷标周围打印（保留）空白标记，以确保该条码能被正确读取，此功能仅用在设计目的，在标签能正常打印与条码能正确判读后，应去掉此指令；

+BARS

在条码上下方打印直线的边线，让条码有更好的可读性，也能避免该条码的错误读取；

SCx

设定条码大小，x 为 0~9 数值，实际条码大小会因条码机分辨率而不同；

範例：

```
J
S l1;0,0,68,71,100
B 10, 5,0,UPCE0,20,.35;03210000678
B 10,30,0,UPCE0,        SC1;01230000088
A 1
```

C - 裁刀参数

语法： **C amount [,disp1[.disp2]]**

C	启动裁刀裁切指令
amount	每隔几张标签裁切一次
disp1	第一个裁切位移，以选定的长度单位作为依据
disp2	第一次裁切后的距离 disp2 再裁切一次， 此参数必为正数值！

长度单位是以毫米(mm)或英吋(inch)为单位

语法： **C e**

C	启动裁刀裁切指令
e	一直打印工作结束为止才裁切

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,9;cut after 2 labels
C2
A10
```

说明：

打印 10 张标签，每两张裁切一次。

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,9;cut after 2 labels
C5,0,2
A10
```

说明：

打印 10 张标签，每 5 张裁切一次，裁切位移为 0，第一次裁切后的 2 mm 再裁切一次，也就是第六张标签纸的前 2 mm 部分会被裁掉。

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,9;cut after 5 labels
C 5
A 100
R cut after 2 labels
C 2
A 60
```

说明：

先打印 100 张卷标时，每 5 张标签裁切一次，然后再印 60 张卷标，每 2 张卷标裁切一次。

E - 档案定义（延伸功能）

语法： **E EXT；name_type**

E = 延伸档案定义指令

EXT = 延伸档案类型

有效的档案类型：

DBF = Database 档案，与 [DBF] 指令一起使用

TMP = 暂存盘，如跳序号档案，
与 [RLOG] 及 [WLOG] 指令一起使用

LOG = 定义外部通讯协议档案（LOG 文件）之名称，
与 [RLOG] 及 [WLOG] 指令一起使用

SQL = 定义数据库服务器的地址，
与 cab Database Connector 软件一起使用

name_type = 即 DBF、TMP、或 LOG 的檔名

= SQL 的 IP 地址与端口值（IP:port）

範例：

```
E DBF;article
```

说明：连结 CF 记忆卡里的 article.DBF 檔，档名必为 8.3 格式（8 个字符文件名与 3 个延伸字符扩展名）

範例：

```
E TMP;SERNUM
```

说明：连结 CF 记忆卡里的 SERUM.TMP 序号暂存档，档名必为 8.3 格式

範例：

```
E LOG;PROTOCOL
```

说明：定义记忆卡内的 PROTOCOL.LOG 档案，档名必为 8.3 格式

範例：

```
E SQL;192.168.0.56:1001
```

说明：定义 SQL 的 IP 地址与端口值（1001）

F - 字型编号

语法： **F number;name**

F = 字型编号指令

number = 新的字型编号

name = 被字型编号取代的字型名称

範例：

```
F 4;Times New Roman
```

说明：使用 TrueType 或 Speedo 名字型；

範例：

```
F 40; Swiss 721 Bold Italic
```

说明：定义条码机内建字型 Swiss 721 Bold Italic 字型编号为 40；

範例：

```
J SAMPLE
H 66
S l1;0,0,68,71,100
F 10;Comix
T 0,15,0,10,pt20;SampleJ:c108]
T 10,25,0,3,pt12;label,
B 5,40,0,EAN-8,SC2;4376131
A 20
```

说明：定义下载到条码机内部存储器的字型（Comix）编号为 10，并打印两行文字，第一行文字是以 Comix 字型打印，第二行则是印条码；

G - 线条

语法： **G [:name;]x,y,r;ge:settings[,options]**

G	=	线条指令
[:name;]	=	定义数据内容于唯一的数据名，供后续程序呼叫或其它用途（如数学运算）使用，最长 10 个字符，且不能使用特殊字符，其中数据名有字母大小之区别；
x	=	x 轴坐标
y	=	y 轴坐标
r	=	旋转角度，0~359 度间隔 1°
ge	=	线条类型： L：直线 R：矩形 C：圆形 椭圆形可用圆形指令定义
settings	=	特殊图形设定
[+options]	=	附加参数
		,fill =
		,shade =
		,outline =

G -线条　C - 圆形

语法：　**G [:name;]x,y,r;C:radius1[,radius2[,width]][,options]**

G = 线条指令

[:name;] = 定义数据内容于唯一的数据名，供后续程序呼叫或其它用途（如数学运算）使用，最长 10 个字符，且不能使用特殊字符，其中数据名有字母大小之区别；

x = （水平）x 轴坐标

y = （垂直）y 轴坐标

圆形与椭圆形起始点在中心点

r = 旋转角度

在饼图案下无影响，在椭饼图案下则会有影响

C = 圆形

radius1 = 水平半径

radius2 = 垂直半径

width = 圆形线条的宽度

如未设定宽度，则会印出实心圆或实心椭圆

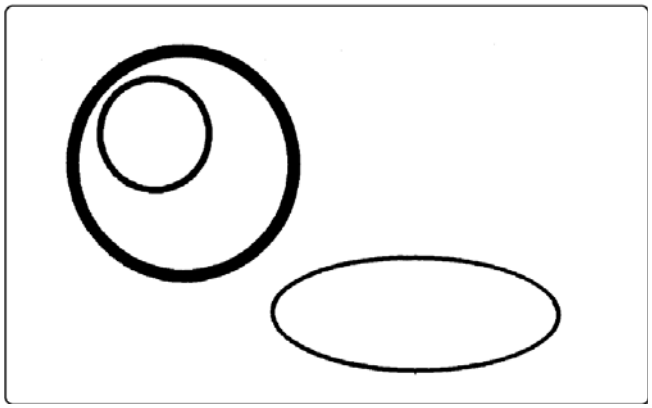
[+options] = **,fill** = 线条为实心模式（参见后面解说）

,shade = 线条为阴影模式（参见后面解说）

,outline = 线条有框线模式（参见后面解说）

範例：

```
J
S l1;0,0,68,71,100
G 65,50,0;C:25,10,.7
G 25,25,0;C:20,20,2
G 20,20,35;C:10,10,1
A 1
```



G -线条　L - 直线

语法：　**G [:name;]x,y,r;L:length,width[,start[,end]][,options]**

G = 线条指令

[:name;] = 定义数据内容于唯一的数据名，供后续程序呼叫或其它用途（如数学运算）使用，最长 10 个字符，且不能使用特殊字符，其中数据名有字母大小之区别；

x = （水平）x 轴坐标

y = （垂直）y 轴坐标

直线起始点在该直线起始位置的中心

r = 旋转角度，0~359 度间隔 1°

L = 直线

length = 长度

width = 宽度

start / end = 线条起点型态 /线条终点型态

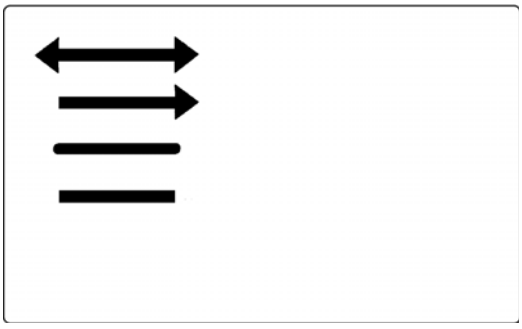
- **s** = 直角
- **r** = 圆弧
- **a** = 箭头

如未设定线条起点/终点型态，则会以直角型态打印

[+options] =
- **,fill** = 线条为实心模式（参见后面解说）
- **,shade** = 线条为阴影模式（参见后面解说）
- **,outline** = 线条有框线模式（参见后面解说）

範例：

```
J
S l1;0,0,68,71,100
G 5,5,0;L:24.5,2.5,a,a
G 5,15,0;L:24.5,2.5,s,a
G 5,25,0;L:24.5,2.5,r,r
G 5,35,0;L:24.5,2.5
A 1
```



G -线条　R - 矩形

语法：　**G [:name;]x,y,r;R:width,height[,hlt[,vlt]][,options]**

G = 线条指令

[:name;] = 定义数据内容于唯一的数据名，供后续程序呼叫或其它用途（如数学运算）使用，最长 10 个字符，且不能使用特殊字符，其中数据名有字母大小之区别；

x = （水平）x 轴坐标

y = （垂直）y 轴坐标

矩形起始点在左上角、矩形的外边

r = 旋转角度

R = 矩形

width = 矩形（水平）宽度

height = 矩形（垂直）长度

hlt = 水平线条厚度

vertw = 垂直线条厚度

如未设定宽度，则会印出实心矩形

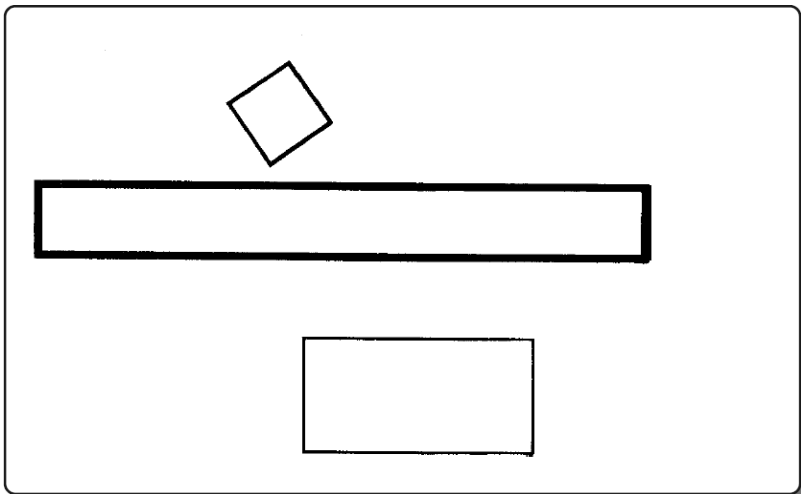
[+options] = **,fill** = 线条为实心模式（参见后面解说）

,shade = 线条为阴影模式（参见后面解说）

,outline = 线条有框线模式（参见后面解说）

範例：

```
J
S l1;0,0,68,71,100
G 35,45,0;R:30,15,.3,.3
G 0,25,0;R:80,10,1,1
G 25,15,35;R:10,10,.5,.5
A 1
```



G -线条　附加参数：Fill – 填满

语法：　**G [:name;]x,y,r;ge:settings[F:options]**

F　=　填满

options =　填满模式参数，有效参数如下：

0%、6%、12%、25%、38%、50%、100%（网点密度）

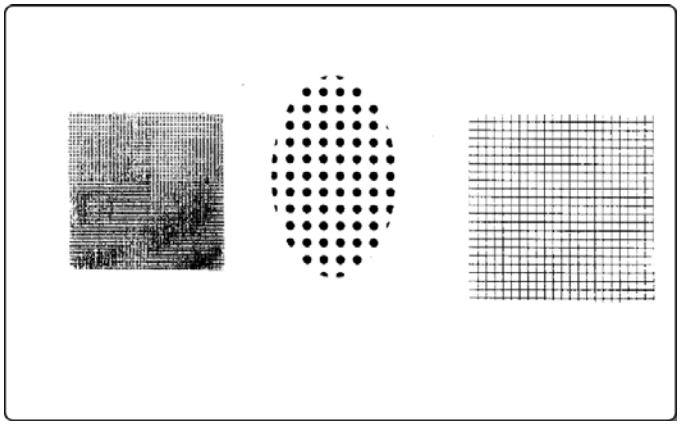
已定义模式有：

左(left)、右(right)、网点(dots)、网格(grid)、菱形(diamond)

图形 1~4(user1~4，下载图档为 32x32 dots)

範例：

```
J
S l1;0,0,68,71,100
G 70,20,0;R:30,30, 1,20[F:grid]
G 48,30,0;C:10,16,10,10[F:dots]
G 5,20,0;R:25,25, 1,20[F:25%]
A 1
```



G -线条　附加参数：Shade – 阴影

语法：　**G [:name;]x,y,r;ge:settings[S:%1[,%2[,direction]]]**

S　=　阴影

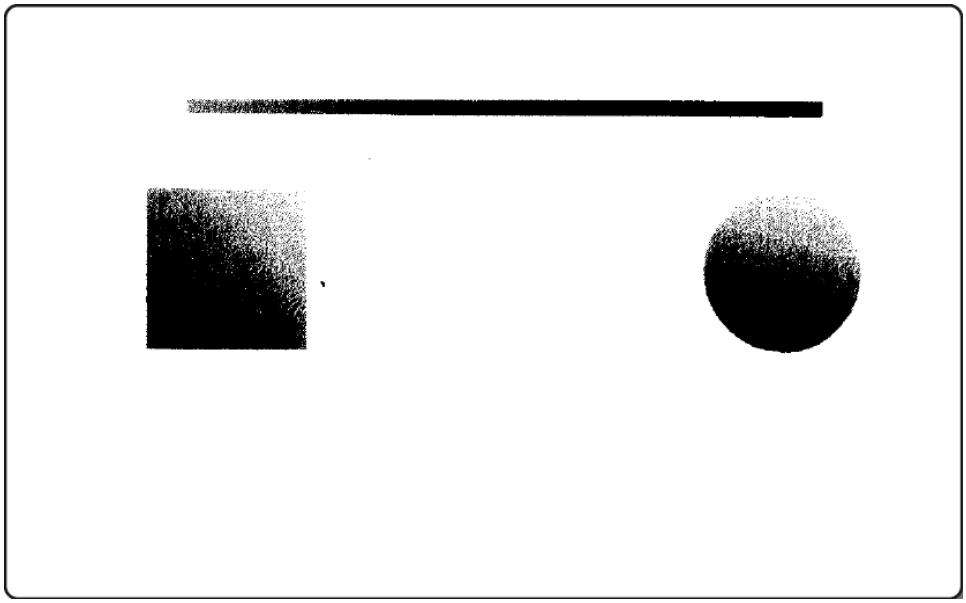
%1=黑色初始浓度值

%2=黑色结束浓度值

方向 direction=阴影角度

範例：

```
J
S l1;0,0,68,71,100
G  5,20,0;R:20,20, 1,20[S:60,10,45]
G 85,30,0;C:10,10,10,10[S:60,10,75]
G 10,10,0;L:80,2[S:30,90,0]
A 1
```



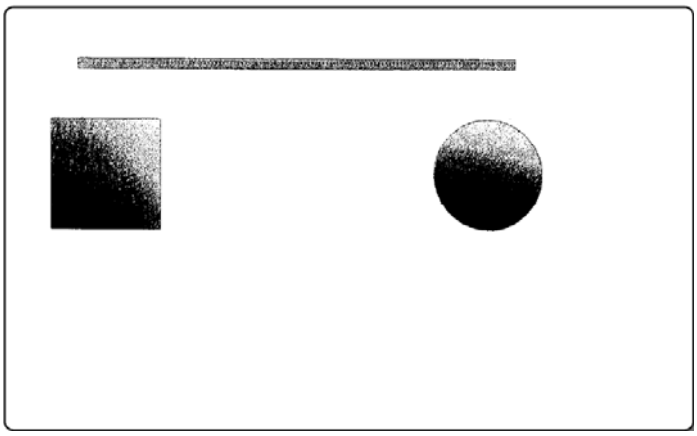
G - 线条　　附加参数：Outline – 外框

语法：　**G [:name;]x,y,r;ge:settings[shade options][O]**

O　　=　外框，用来呈现未填满物体的外围框线

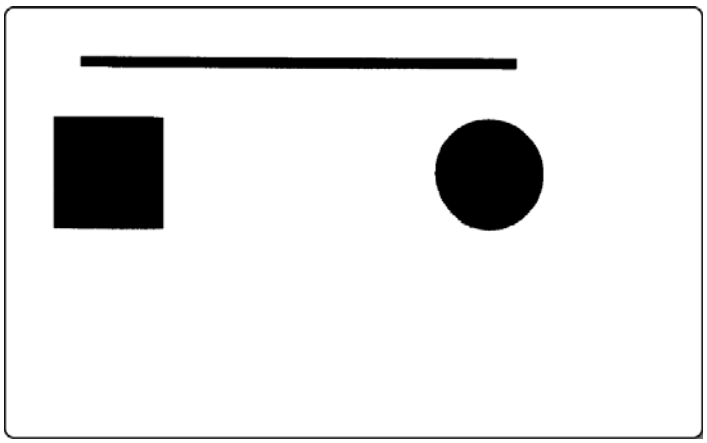
範例：

```
J
S l1;0,0,68,71,100
G 5,20,0;R:20,20,1,20[S:60,10,45][O]
G 85,30,0;C:10,10,10,10[S:60,10,75][O]
G 10,10,0;L:80,2[S:30][O]
A 1
```



範例：

```
J
S l1;0,0,68,71,100
G 5,20,0;R:20,20,1,20[O]
G 85,30,0;C:10,10,10,10[O]
G 10,10,0;L:80,2[O]
A 1
```



H - 打印温度、速度、与打印方式

语法：**H speed[,h][,t][,r][,Bb]**

H = 打印头工作温度等级与打印速度指令

speed = 打印速度，各机型的最快打印速度不同，请先参照操作手册，以便对打印速度能有正确设定，如设定错误，则会在条码机的 LCD 控制面版上显示错误设定值

h = 打印头工作温度等级（-20 到+10）

t = 打印型式，T 为热转打印，D 为热敏打印，默认值为热转打印 (T)

b = 回纸速度

範例：

```
H 150,0,D
```

说明：此设定为打印速度是 150mm/s，打印头工作温度等级为 0，以热敏方式打印。

I - 图片

语法：I **[:name;] x,y,r[mx,my];name**

I	=	图片指令
[:name;]	=	定义数据内容于唯一的数据名，供后续程序呼叫或其它用途（如数学运算）使用，最长 10 个字符，且不能使用特殊字符，其中数据名有字母大小之区别；
x	=	图片（水平）x 轴坐标
y	=	图片（垂直）y 轴坐标

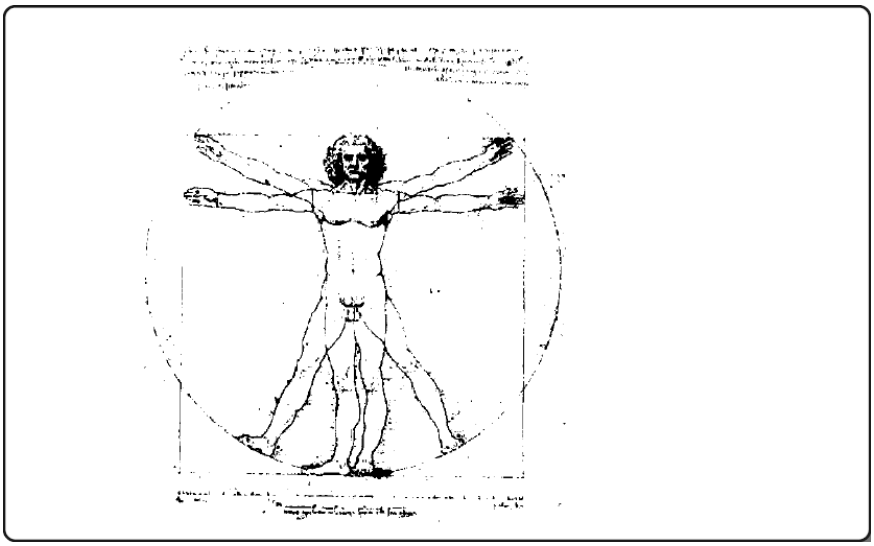
最大图片坐标依条码机机型不同而有所不同，请参照操作手册

r	=	旋转角度，为 0°、90°、180°、与 270°
mx	=	水平放大参数，数值为 1-10
my	=	垂直放大参数，数值为 1-10

範例：

```
J
S l1;0,0,68,71,100
I:IMAGE1;20,5,0,0,0;HUMAN
T 12,25,0,3,6;Todays date is: [DATE:+03,+02,+10]
A1
```

说明：打印已下载到条码机内部存储器的 ”HUMAN” 图片，亦可将图片储存在记忆卡，再从程序中呼叫出来打印



J - 打印温度、速度、与打印方式

语法：**J [comment]**

J = 打印开始指令

comment = 会显示在条码机 LCD 控制面版上的英文文字，可当作卷标名使用，最大显示长度为 15 个字符

範例：

```
J Adress label
```

说明：条码机的 LCD 上会显示“Adress label”，并开始执行 J 后面的打印指令

M - 记忆卡存取

语法：**M variation…**

M 指令为条码机在使用 CF 记忆卡时，对记忆卡做存取的动作

语法：**Mc [pathname]**

Mc = 记忆卡内容查询

pathname = /card/ 呼叫记忆卡储存的内容，此为默认值

= /iffs/ 呼叫条码机内部闪存档案系统内容

语法：**Md type;name**

Md = 删除记忆卡数据

type = LBL（label，卷标型式）

FNT（font，字型型式）

IMG（image，图片型式）

FMT（label format，标签格式）

name = 记忆卡内的文件名

“type”:FNT 删除所有 TTF 与 Speedo 字型

“type”:IMG 删除所有指定文件名的图片

範例：

```
M d IMG;logo
```

说明：删除所有记忆卡里的 “logo”图形文件，例如 logo.bmp 或 logo.pcx

语法：**Mf;name**

Mf = 格式化记忆卡

範例：

```
M f;MYDATA
```

说明：格式化记忆卡并将记忆卡命名为 “MYDATA”

语法：**Ml type;[pathname]name**

Ml = 读取记忆卡数据

type = LBL（label，卷标型式）
FNT（font，字型型式）
IMG（image，图片型式）
FMT（label format，标签格式）

name = 记忆卡内的文件名

pathname = /card/ 呼叫记忆卡储存的内容，此为默认值
= /iffs/ 呼叫条码机内部闪存档案系统内容

範例：

```
Ml LBL;TESTLBL
A2
```

说明：读取记忆卡的 TESTLBL 卷标文件，并打印 2 张标签

语法：**Mr**

Mr = 重复上一个档案内容

範例：

```
Ms LBL;LOOP
J
S l1;0,0,68,70,100
T:Text1;20,10,0,3,7;[?:SerialNo:]
A3
Mr
Ms LBL
```

说明：将“LOOP”标签文件储存于条码机的记忆卡，然后执行打印动作，条码机会先在 LCD 上显示“SerialNo”，并等待操作者输入序号后，会打印 3 张标签，然后再回到等待操作者输入序号的状态，直到操作者按下条码机操控面版的取消键（CANCEL）才会取消整个打印动作

语法：**Ms type;name**

Ms = 储存数据于记忆卡

type = LBL（label，卷标型式）
FNT（font，字型型式）
IMG（image，图片型式）
FMT（label format，标签格式）

name = 记忆卡内的文件名

pathname = /card/ 呼叫记忆卡储存的内容，此为默认值
= /iffs/ 呼叫条码机内部闪存档案系统内容

範例：

```
Ms LBL;ASERIES
J
S l1;0,0,36,38,89
T:Text1;20,10,0,3,pt25;cab printers
A5
Ms LBL
```

说明：将“ASERIES”标签文件储存于条码机的记忆卡，然后打印 5 张标签，如将标签文件名设定为 “DEFAULT.LBL”，则会在条码机开机后便会立刻打印该标签文件！

语法：**Mu IMG;logo**

Mu = 从记忆卡上传二进制数据文件内容

O - 设定打印参数

语法：**O [M,][R,] [N,][T,][S,][U,][p,]**

O = 打印参数指令

M = 标签映象打印

R = 180° 旋转标签方向

N = 整个标签打印反白

S = 单一卷标暂存，当上一标签完成打印时会处理下个标签

T = 启用撕纸模式，会在打印卷标后，

U = 封锁暂停后重打印的功能，以避免同一标签重复印两次

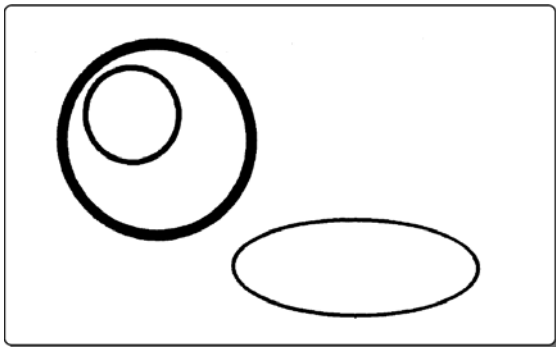
p = 打印模式的回纸（backfeed）功能设定

“**P**” 指令会设定回纸功能为智能型（smart），“**D**”指令则是启动回纸（always），此参数设定会先暂时覆盖条码机硬件的设定，等该标签打印工作完成后，条码机才会回复到原先回纸功能的设定模式。

O - 设定打印参数

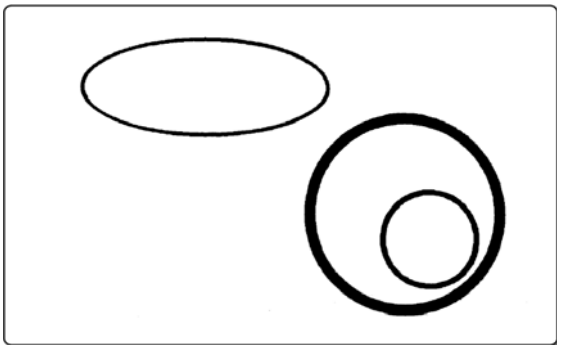
範例：

```
J
S l1;0,0,68,71,100
G 65,50,0;C:25,10,.7
G 25,25,0;C:20,20,2
G 20,20,35;C:10,10,1
A 1
```



範例：

```
J
O R
S l1;0,0,68,71,100
G 65,50,0;C:25,10,.7
G 25,25,0;C:20,20,2
G 20,20,35;C:10,10,1
A 1
```



说明：有使用跟无使用 O R（旋转卷标）指令的打印情形

P - 设定卷标剥离模式或贴标模式

语法：**P [disp]**

P	=	卷标剥离模式或贴标模式指令
disp	=	标签纸偏移位置，可用正数值与负数值

P 指令需放在 S 指令（定义卷标尺寸）后面！

S - 设定标签尺寸

语法：**S [ptype;]xo,yo,ho,dy,wd[,dx,col][;name]**

S = 设定卷标尺寸指令

ptype = 设定标签纸型式

e = 标签纸为连续纸，此参数会关闭条码机的标签纸传感器

以下 **l** 为 **L** 的小写

l0 = 卷标纸上方有反射标志供条码机的卷标纸传感器侦测

l1 = 标签纸上有间距供条码机的标签纸传感器侦测

l2 = 卷标纸下方有反射标志供条码机的卷标纸传感器侦测

xo = 标签起点水平 x 轴位移

yo = 标签起点垂直 y 轴位移

ho = 打印方向的标签高度

dy = 包含间距的标签高度

wd = 标签宽度

附加参数：当水平方向有多个标签纸时

dx = 水平方向的第一个卷标与第二个卷标之间的距离

col = 水平方向的卷标数目，默认值为 1

name = 显示在条码机 LCD 上的文字，

例如可用来显示所要求放入的卷标纸规格

範例：

```
S l1;0,0,50,52,100
```

说明：定义卷标为有间距的卷标纸，高为 50mm、含间距的高为 52mm，宽为 100mm，水平与垂直位移为 0

标签纸的最大宽度与最长高度设定会依不同条码机而有所不同，此部分设定请参照条码机使用手册！

T - 文字

语法：**T [:name;]x,y,r,font,size[,effects];text**

T = 文字指令

[:name;]= 定义数据内容于唯一的数据名，供后续程序呼叫或其它用途（如数学运算）使用，最长 10 个字符，且不能使用特殊字符，其中数据名有字母大小之区别；

x = 水平起点 x 轴坐标

y = 垂直起点 y 轴坐标

直线起始点在该直线起始位置的中心

r = 文字旋转角度，向量字型与 True type 字型皆可以间隔 1°。

旋转 360 度，Bitmap 自行则是可旋转四个方向（0°、90°、180°、270°）

font = 以自行编号定义字型，可用条码机内建字型或另外下载到条码机的字型，下列字型为条码机内建字型：

font nr.	Name	Type	Description
-1	_DEF1	Bitmap	Default-size 12x12 dots
-2	_DEF2	Bitmap	Default-size 16x16 dots
-3	_DEF3	Bitmap	Default-size 16x32 dots
-4	OCR_A_I	Bitmap	OCR-A Size I
-5	OCR_B	Bitmap	OCR-B
3	BX000003	Vector	Swiss 721™
5	BX000005	Vector	Swiss 721 Bold ™
596	BX000596	Vector	Monospace 821 ™

size = 设定字号，向量字型可设定为毫米（mm）或英吋（inches），亦可设为 “pt x”；

bitmap 字型则是以 mx（水平，1-10 倍）与 my（垂直，1-10 倍）放大或缩小字型

effects = 字型特效，不同字型的特效亦不同，有效的字型特效如下：

b = 粗体 **s** = 斜体 **z** = 往左斜体

n = 反白 **u** = 下标 **i** = 斜体

k = 字体空隙最佳化 **v** = 垂直打印文字

qn = 压缩字体，默认值为 100（不压缩），可设定值 10-10000

hn = 字体宽度，**h** 亦可用 **H**

mn = 水平文字间距，n 为毫米（mm）或英吋（inches）

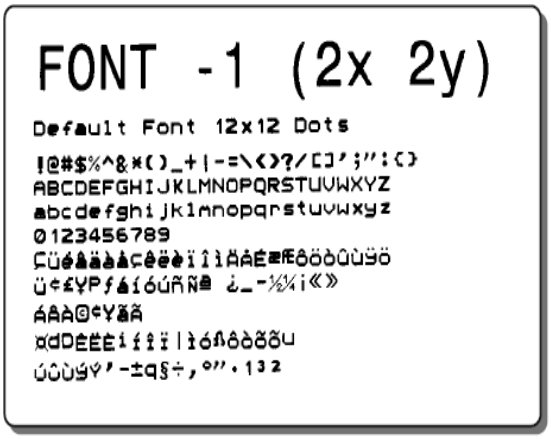
以下特效仅能用在条码机内建 bitmap 字型

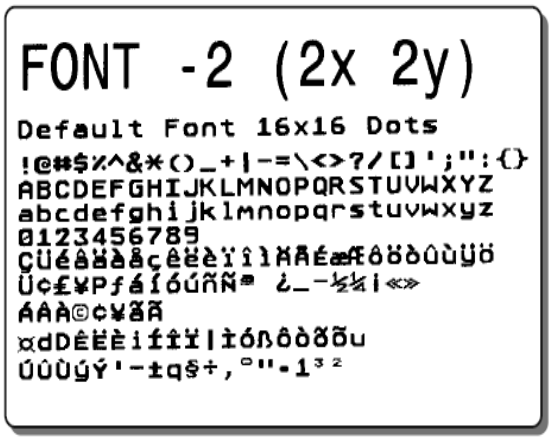
o = 外框（OCR 字型无效） **g** = 灰阶（OCR 字型无效）

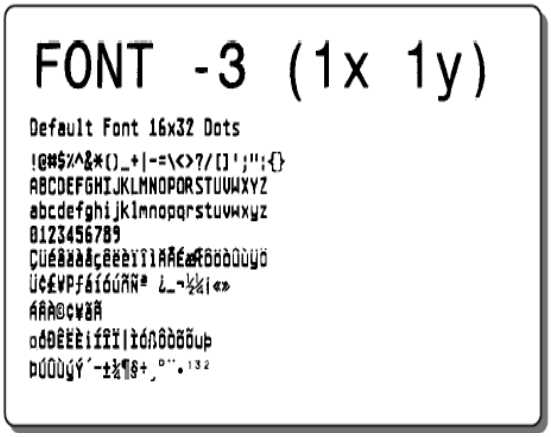
xn / yn = 水平 x 轴 / 垂直 y 轴放大倍数，n=1-10

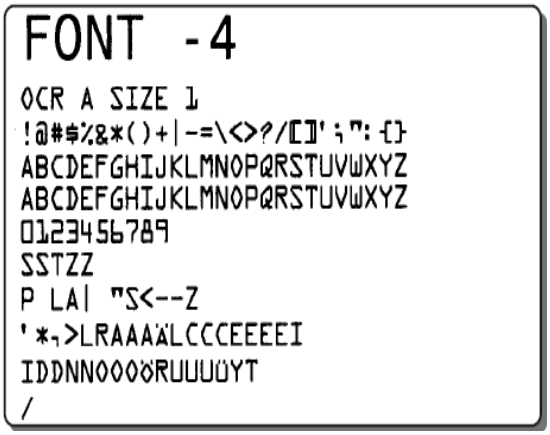
text = 文字内容

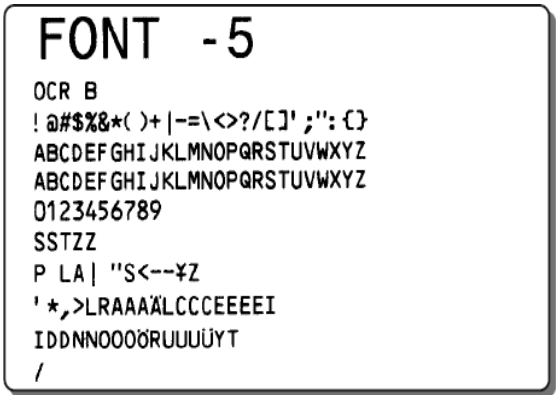
以下为条码机内建 Bitmap 字型实际打印情形，所有打印皆放大以便观看：





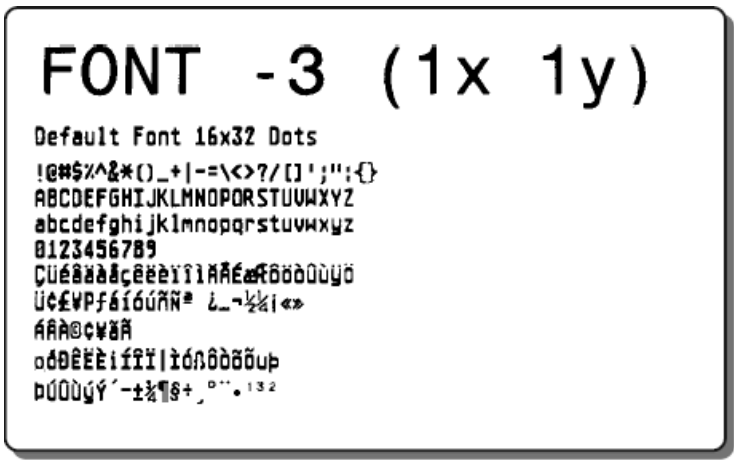


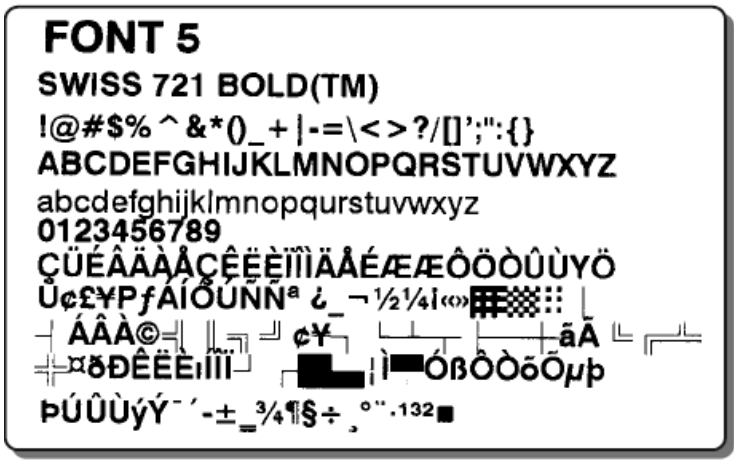


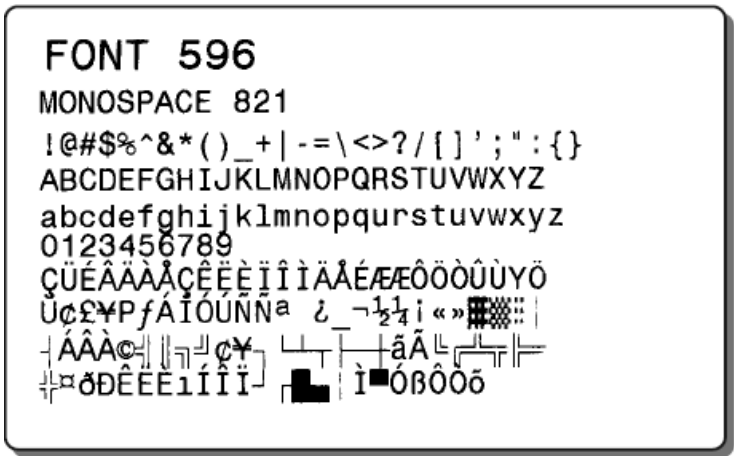


T - 文字

以下为条码机内建向量字型的打印效果：



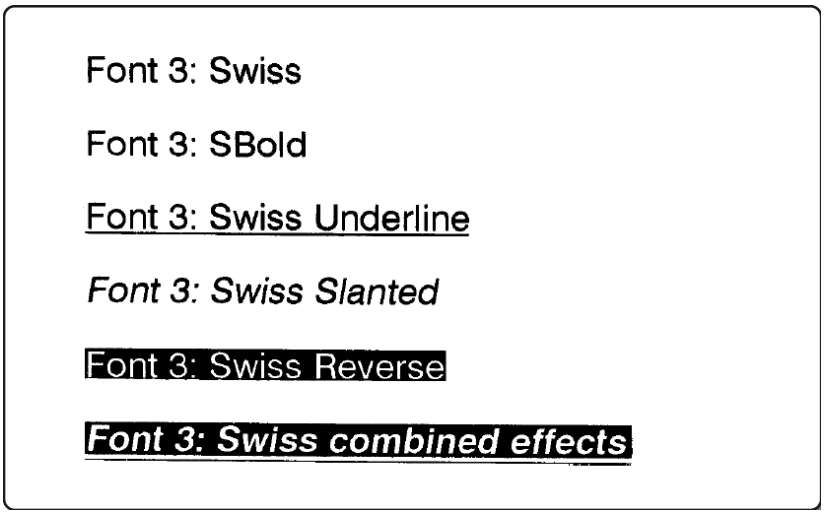




T - 文字

範例：

```
J
S 0,0,68,71,100
T 10,10,0,3,5;Font 3: Swiss
T 10,20,0,3,5;Font 3: S Bold
T 10,30,0,3,5,u;Font 3: Swiss Underline
T 10,40,0,3,5,s;Font 3: Swiss Slanted
T 10,50,0,3,5,n;Font 3: Swiss Reverse
T 10,60,0,5,5,s,u,n;Font 3: Swiss combined effects
A 1
```



X- 周边讯号设定同步化

语法：**X y[;ao]**

X = 同步化外接装置指令

y = 如应设定讯号时，打印坐标

ao = 16 进制半字节设定或重置讯号

X 指令需放在 **S**（卷标尺寸设定）指令之后

此指令之功能与设定会取决于条码机机型与周边装置，细节请参照操作手册

时间：小时：分：秒

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,9;It is [H12] [MIN][SEC]
A1
```

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,6;Actual time is [H024]:[MIN]:[SEC]
A1
```

说明：打印当时的时间，[H12] 为 12 小时制的小时，[H024] 为 24 小时制的小时，[MIN] 为分，[SEC] 为秒，该时间数据是从条码机取得

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,9;It is [H012]  o´clock
A1
```

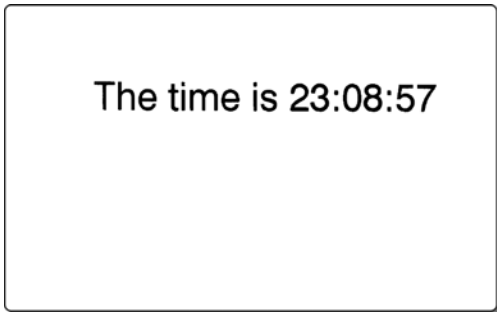
範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,9;The actual hour is [H024]
A1
```

说明：[H012] 为 12 小时制的小时，[H024] 为 24 小时制的小时，两个指令主要会在 1~9 小时补上 0，即打印出 01~09；

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,8;The time is [TIME]
A1
```

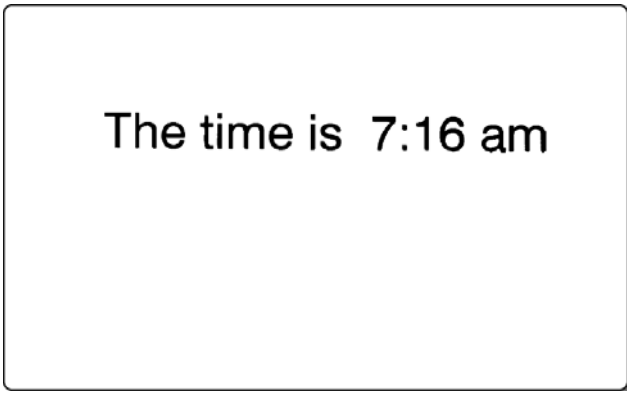


时间：am/pm

语法：**[XM]**

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,8;The time is [H12]:[MIN] [XM]
A1
```

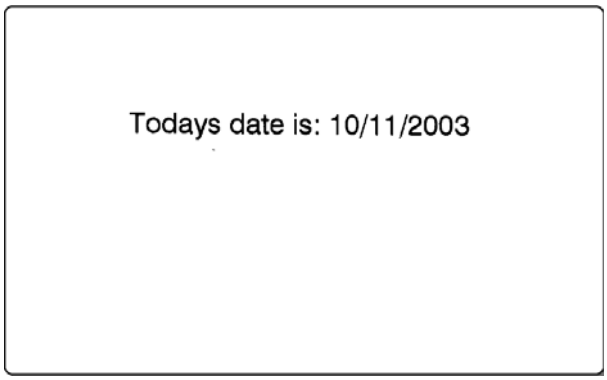


时间：日期

语法：**[DATE]**

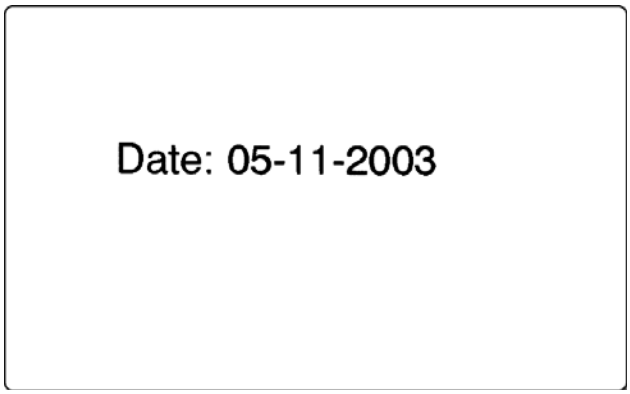
範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,5;Todays date is: [DATE]
A1
```



範例：

```
J
S l1;0,0,68,71,100
T 12,30,0,3,7;Date: [DAY02]-[MONTH02]-[YYYY]
A1
```



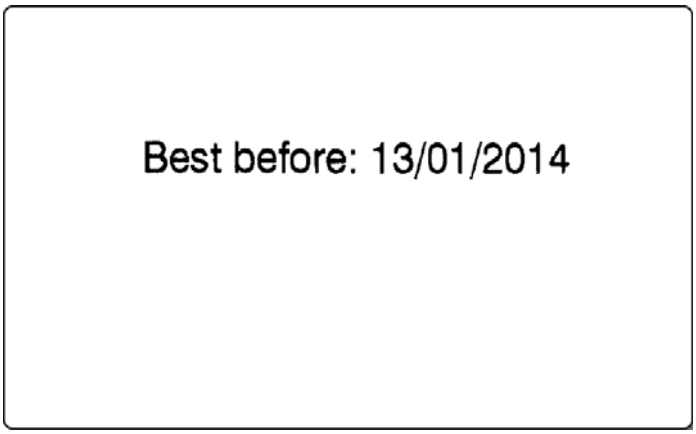
说明：打印当时日期：日/月/年，其中 [DAY02] 与 [MONTH02] 设定为显示 2 位数，不足 2 位数则补 0

时间：日期 – 加入一段时间

语法：**[DATE{:+DD{,+MM{+YY}}}]**

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,6;Best before: [DATE:+03,+02,+10]
A1
```



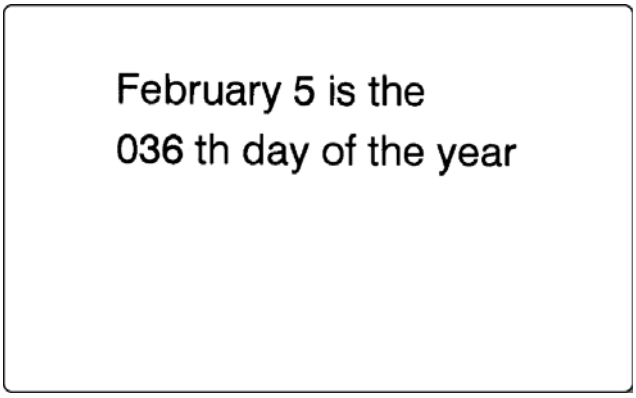
说明：当日日期原为 10/11/2003，加上 3(日)/2(月)/10(年)后，日期变为 13/01/2014，适合用在保存期限的日期使用

时间：打印一年的第几天

语法：**[DOFY{:+DD{,+MM{,+YY}}}]**

範例：

```
J
S l1;0,0,68,71,100
T 12,20,0,3,7;February 5 is the
T 12,30,0,3,7;[DOFY] th day of the year
A1
```



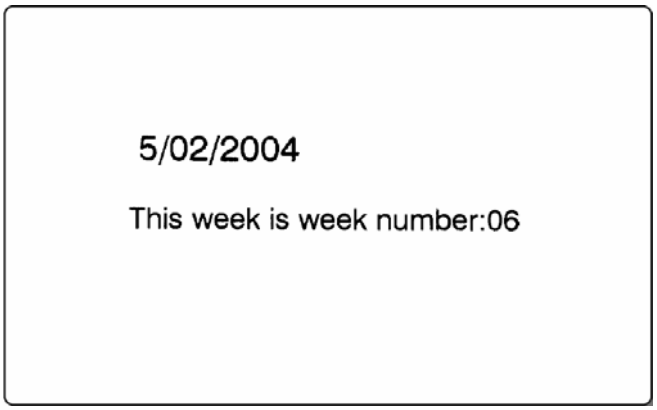
说明：当时日期为 2004 年 2 月 5 日，那天为一年的第 36 天，以三位数显示为 036；

时间：打印周数(1-53)

语法：**[WEEK{:+DD{,+MM{,+YY}}}]**

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,5;Date: [DATE]
T 12,35,0,3,5;This week is week number: [WEEK02]
A1
```



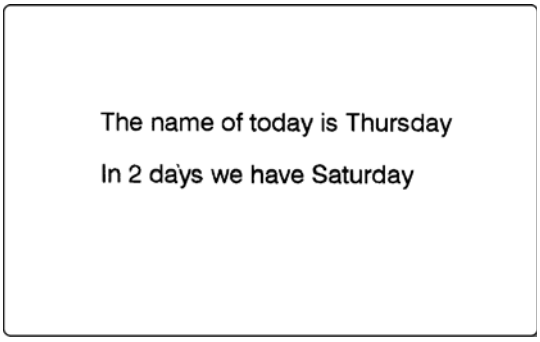
说明：打印当时日期，并打印当时日期为该年的第 6 周，[WEEK02] 会以 2 位数显示周数；

时间：打印完整或部分星期名

语法：**[wday{:+DD{,+MM{,+YY}}}]**

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,5;The name of today is [wday]
T 12,35,0,3,5;In 2 days we have [wday:+02,00,00]
A1
```



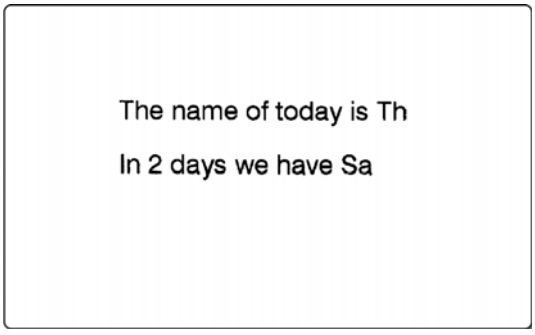
说明：显示当天为星期四（Thursday）及两天后为星期六（Saturday）

语法：**[wday2{:+DD{,+MM{,+YY}}}]**

语法：**[wday3{:+DD{,+MM{,+YY}}}]**

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,5;The name of today is [wday2]
T 12,35,0,3,5;In 2 days we have [wday2:+02,00,00]
A1
```

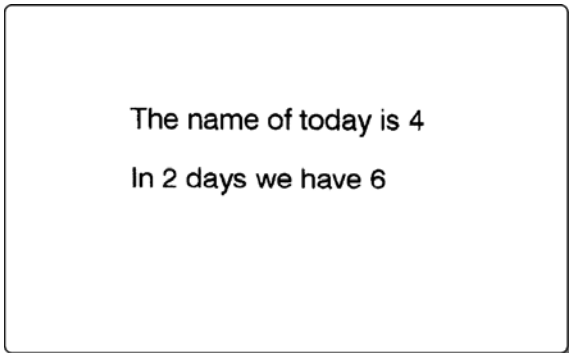


说明：[wday2] 以 2 个字符显示星期名，范例中日期为星期四（Thursday）则显示为 Th，星期六显示为 Sa，亦可以 [wday3] 显示 3 个字符的星期名，但无显示 4 个字符或更多字符的星期名；

时间：打印以数字表示星期名

範例：

```
J
S l1;0,0,68,71,100
T 12,25,0,3,5;The name of today is [WDAY]
T 12,35,0,3,5;In 2 days we have [WDAY:+02,00,00]
A1
```



说明：显示以 4 代表当天为星期四，及两天后为星期六，以 6 代表
0 = Sunday 星期日， 1 = Monday 星期一, 2 = Tuesday 星期二,
3 = Wednesday 星期三, 4 = Thursday 星期四, 5 = Friday 星期五,
6 = Saturday 星期六

时间：打印完整或部分月份名

语法：**[mon{:+DD{,+MM{,+YY}}}]**

範例：

```
J
S l1;0,0,68,71,100
T 10,30,0,3,10;[mon]
A1
```

说明：打印以三个字符显示当时月份为 Feb（二月）

语法：**[month{:+DD{,+MM{,+YY}}}]**

範例：

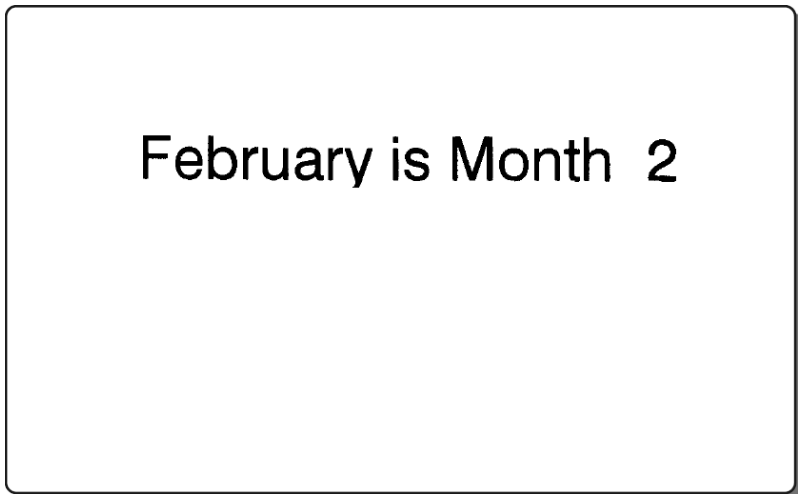
```
J
S l1;0,0,68,71,100
T 10,30,0,3,10;[month]
A1
```

说明：打印完整显示当时月份为 February（二月）

时间：打印以数字表示月份名

範例：

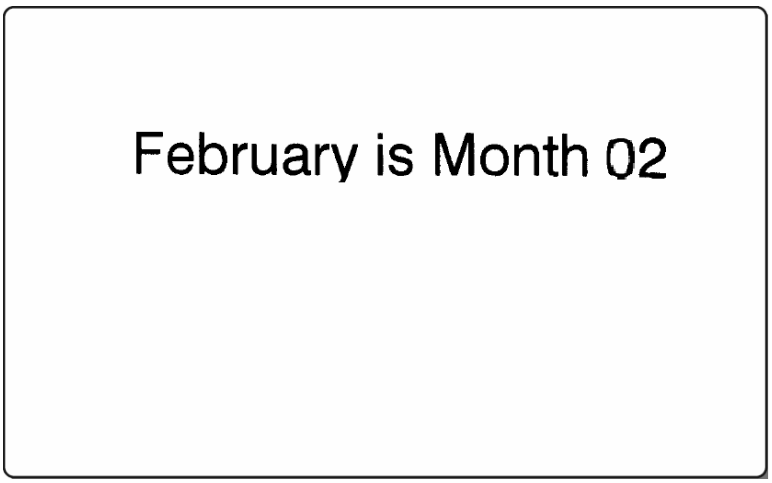
```
J
S l1;0,0,68,71,100
T 10,30,0,3,8;[month] is Month [MONTH]
A1
```



说明：打印当时月份及以 2 位数显示当时月份

範例：

```
J
S l1;0,0,68,71,100
T 10,30,0,3,8;[month] is Month [MONTH02]
A1
```



说明：打印当时月份及以 2 位数显示当时月份，月份数不足 2 位数补 0

时间：打印年份

语法：**[YY{:+DD{,+MM{,+YY}}}]**

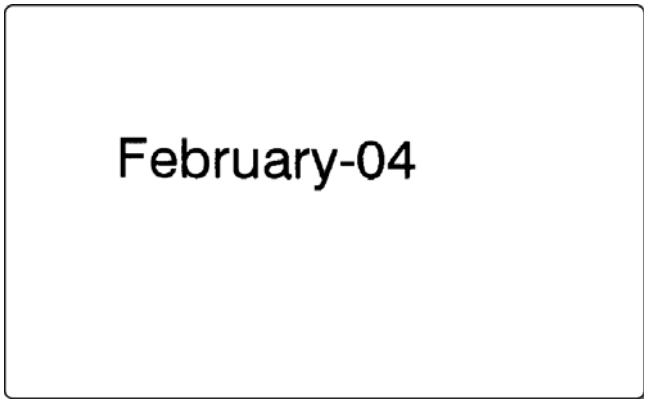
YY = 2 位数显示年份（0-99）

语法：**[YYYY{:+DD{,+MM{,+YY}}}]**

YYYY = 4 位数显示年份（1970-2069）

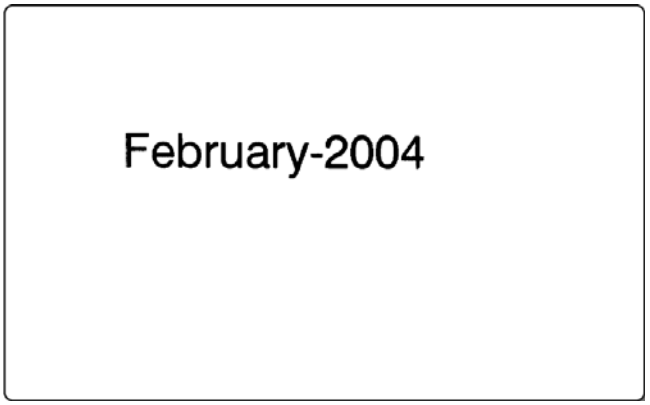
範例：

```
J
S l1;0,0,68,71,100
T 10,30,0,3,8;[month]-[YY]
A1
```



範例：

```
J
S l1;0,0,68,71,100
T 10,30,0,3,8;[month]-[YYYY]
A1
```

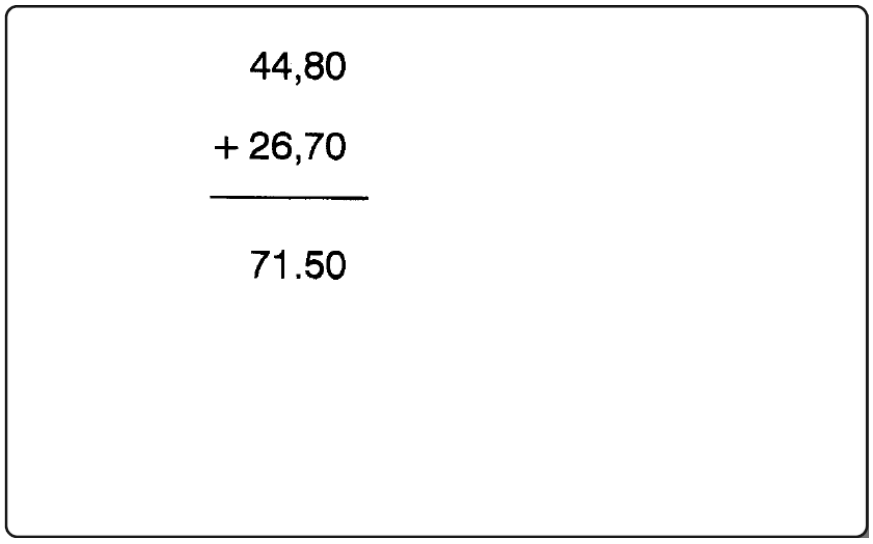


加法运算：

语法：**[+:op1,op2,・・・]**

範例：

```
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;44,80
T:var2;20,20,0,3,5;+
T:var2;25,20,0,3,5;26,70
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[+:var1,var2]
A1
```

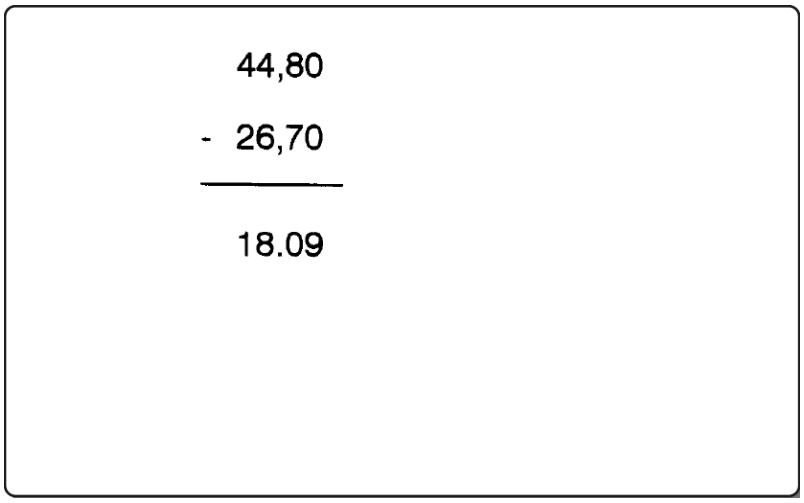


加法运算：

语法：**[-:op1,op2,・・・]**

範例：

```
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;44,80
T:var2;20,20,0,3,5;-
T:var2;25,20,0,3,5;26,70
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[-:var1,var2]
A1
```

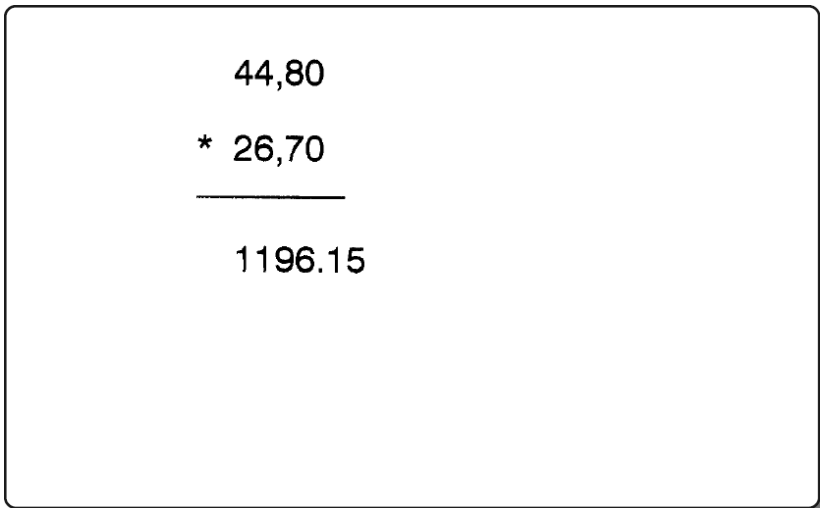


乘法运算：

语法：**[*:op1,op2,・・・]**

範例：

```
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;44,80
T:var2;20,20,0,3,5;*
T:var2;25,20,0,3,5;26,70
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[*:var1,var2]
A1
```

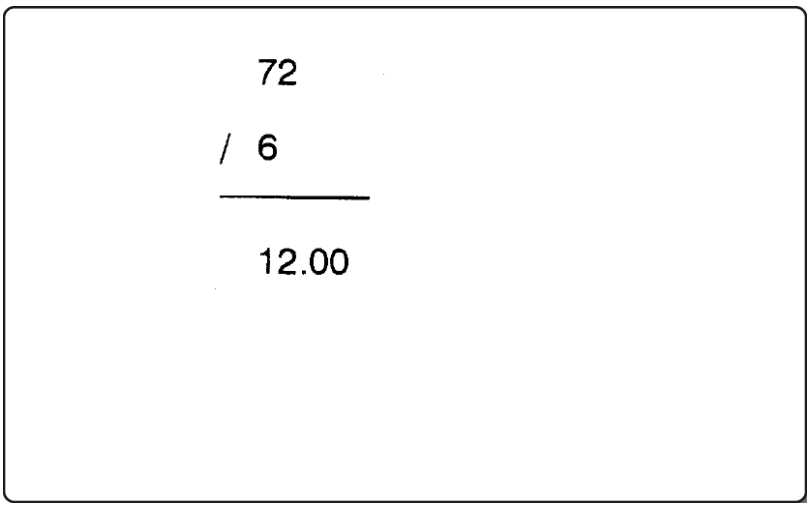


除法运算：

语法：**[/:op1,op2,・・・]**

範例：

```
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;72
T:var2;20,20,0,3,5;/
T:var2;25,20,0,3,5;6
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[/:var1,var2]
A1
```

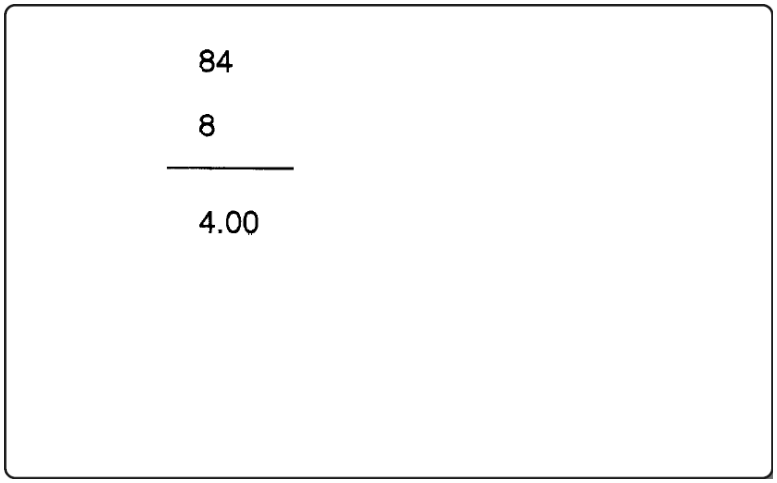


余数运算：

语法：**[%:op1,op2,・・・]**

範例：

```
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;84
T:var2;25,20,0,3,5;8
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[%:var1,var2]
A1
```



说明：打印 84 除以 8 的余数为 4

範例：

```
J
S l1;0,0,68,71,100
T:COUNT;5,10,,3,4;[SER:000000][I]
T:MODCALC;5,10,,3,4;[%:COUNT,15][I]
T:SHIFT; 5,10,,3,4;[+:MODCALC,1][D:2,0]
A 20
```

说明：此程序产生计数器，从 1 累加到 15 然后跳回 1，打印 20 张

金额格式 – [P:...]

语法：**[P:name,td{o}]**

P = 金额格式指令

name = 金额格式指令

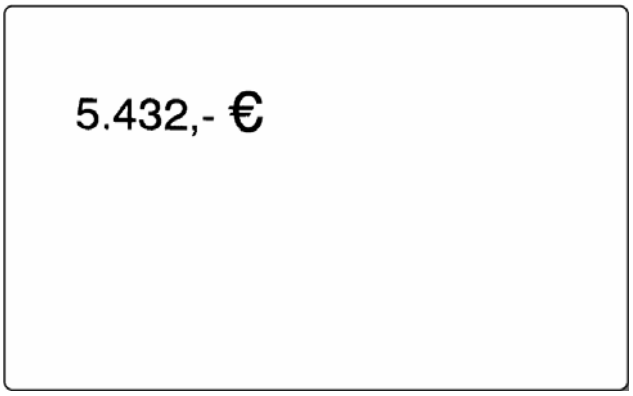
t = 千位数分隔号

d = 小数点与个位数分隔号

o = 附加补遗字符

範例：

```
J
S l1;0,0,68,71,100
T:Price1;10,20,0,3,8;[P:5432,.,-] [U:$20AC]
A 1
```



说明：打印金额为 5432 欧元的内容，其中 [U:$20AC] 为转换成欧元符号 €，千位数分隔号为“.”，小数点与个位数分隔号为“，”，金额与欧元符号间的区隔符号为“-”

小数点进位法 – [R:x]

语法：**[R:x]**

R = 小数点进位法指令

x = n：无进位法则（默认值）

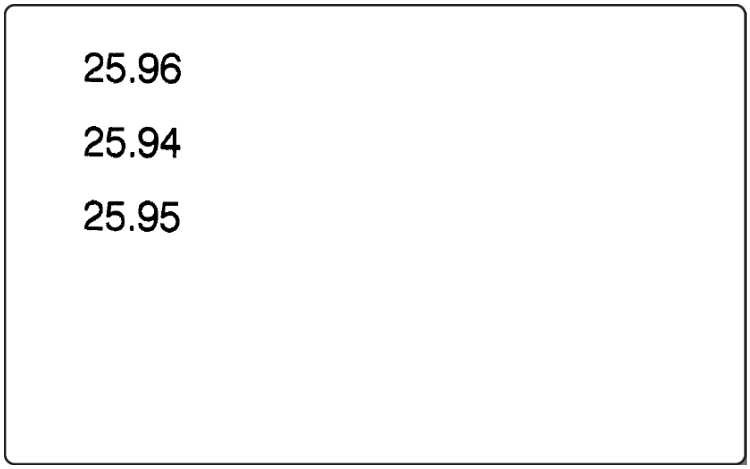
x = u：无条件进入

x = d：无条件舍去

x = m：四舍五入

範例：

```
J
S l1;0,0,68,71,100
T 10,10,0,3,6;[*:5.191,5] [R:u]
T 10,20,0,3,6;[*:5.1898,5] [R:d]
T 10,30,0,3,6;[*:5.1898,5] [R:m]
A 1
```



LCD 提示 – [? :...]

语法：**[？:x,y,z{,D}{,Lx}{,Mx}{,R}{,J}]**

？ = LCD 提示指令

x = 显示在条码机 LCD 上的文字

y = 预设等待输入的数据值，如无设定则为空白，或以前次输入的数据为主

z = 设定输入的频率

其它参数：

D = 删除前次输入的数据

Lx = 输入数据长度，x=1-200（字符）

Mx = 设定输入数据为下列定义：

x = 0 具小数点分隔符的数目
1 数值 0-9
2 小写英文字母
3 小写英文字母与数字
4 大写英文字母
5 大写英文字母与数字
6 大小写英文字母
7 大小写英文字母与数字
8 所有字符
9 符号与小数点分隔点

如在 M 指令后直接使用惊叹号“！”，则不能使用空格符！

R = 如无法从数据库找到输入的纪录，则会重复提示输入数据

J = 当条码机要求输入标签数目时重复提示，如搭配（A [？,R]）指令可当简单循环功能使用

範例：

```
[?:article number]
```

说明：在条码机 LCD 上显示要求输入料号（article number）

範例：

```
[?:article number,7733214]
```

说明：在条码机 LCD 上显示要求输入料号（article number），默认值为 7733214

範例：

[?:article,screw,3]

说明：在 LCD 上显示要求输入品名（article），默认值 screw，打印 3 张后跳回等待下一个输入品名的状态

範例：

[?:article no:,7733214,3,D]

说明：LCD 上显示要求输入料号，默认值为 7733214，每印三张标签后清除前次输入的数据（即 7733214），因此默认值只有在第一次有效

範例：

[?:article,screw,,L8]

说明：在 LCD 上显示要求输入品名（article），默认值 screw，最大输入数据长度为 8 字符/数值

範例：

[?:number,7733214,,M1111111]

说明：LCD 上显示要求输入号码，默认值为 7733214，并设定只能输入 7 位数数值

範例：

[?:artno?,,1,M1114444]

说明：LCD 上显示要求输入料号（artno?），无默认值，且设定只能输入 7 个字符，前三位为数值，后四位为大写英文字母

範例：

[?:article?,,1,M1111111,R,D]

说明：LCD 上显示要求输入品名（article?），无默认值，且设定只能输入 7 个数值，如无法从数据库里找到记录则会重复该提示，并删除前次输入的数据

範例：

```
[?:article,2200333,,,L6,M!11111]
```

说明：LCD 上显示要求输入品名，默认值为 2200333，且设定只能输入 5 个无空白的数值，即使默认值 2200333 为 7 位数，及长度限制为 6 位，实际操作时，也只会显示 5 位数（M！11111）

範例：

```
J simple loop
S l1;0,0,68,71,100
T 10,15,0,3,10;[SER:1]
T 10,30,0,3,10;[?:INPUT?]
T 10,45,0,3,10;[?:Second INPUT?,,,J]
A [?,R]
```

说明：此例为简易循环的应用，[SER:1] 跳序号，从 1 起跳，[?:INPUT?] 为仅一次提示输入，[?:Second INPUT?,,,J] 则会重复提示输入，直到按取消键（CANCEL）为止

置换 0 – [C:…]

语法：**[C:fill{,base}]**

C = 置换左边的 0

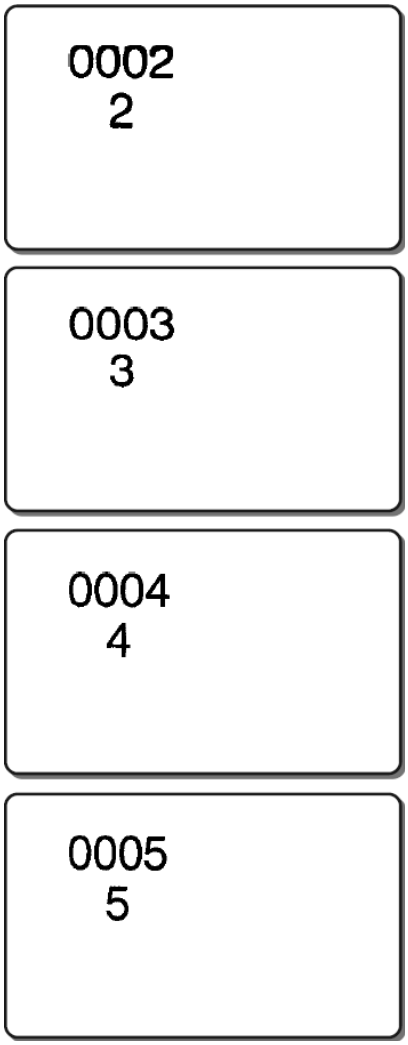
fill = 填满字符

base = 设定计数系统的附加参数

範例：

```
J
S l1;0,0,68,71,100
T:CNT; 10,15,0,3,10;[SER:1][I]
T:FIELD1;10,10,0,3,10;[+:1,CNT][C:0][D:4,0]
T:FIELD2;10,20,0,3,10;[+:1,CNT][C: ][D:4,0]
A 5
```

说明：打印 5 张卷标，卷标内有 2 组计数器，从 2 起跳，[D:4,0] 设定 4 位数值，[C:0] 指令设定要补 0，故第一个计数器有补 3 个 0，第二个计数器是 [C:]，故不补 0



位数设定 – [D:...]

语法：**[D:m,n]**

D = 设定显示位数

m = 总位数

n = 小数点后面的位数，默认值为 2 位

範例：

```
J
S l1;0,0,68,71,100
T:input;10,30,0,3,14;[*:10.79,4.16] [D:4,2]
A 1
```

数据库档案连结 – [DBF:...]

语法：**[DBF:key,keyvalue,entry]**

DBF	=	数据库档案连结指令
key	=	搜寻数据库的数据
keyvalue	=	数据库记录的数据值定义为英文字母及数字
entryfield	=	传送记录的数据值

範例：

```
[DBF:NUMBER,NUMBERTA,ARTICLE]
```

说明：搜寻数据库里的 NUMBERTA 数据表里的 NUMBER 数据值，并传送 ARTICLE 数值；

此指令需搭配 E 指令使用，一张卷标格式仅能使用一个数据库，此指令功能仅适用于小型数据库，大型数据库则建议使用 cab Database Connector 软件才有好的效能。

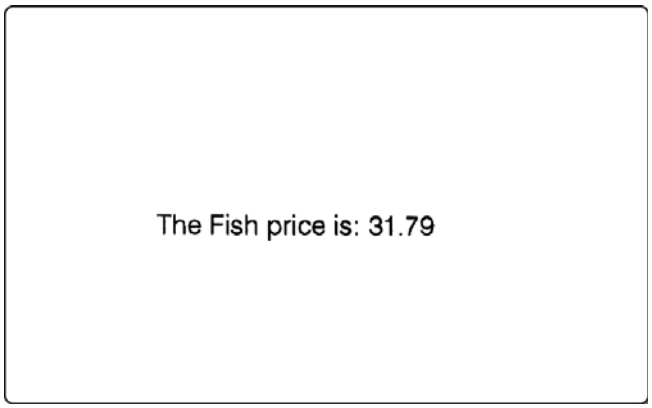
隐藏内容 – [I]

语法：**[I]**

範例：

```
J
S l1;0,0,68,71,100
T:WEIGHT;10,20,0,3,5;[?:Weight?][I]
T:PRICEUNIT;10,20,0,3,5;[I] 2.65
T:RESULT;10,40,0,3,4;The Fish price is: [*:WEIGHT,PRICEUNIT]
A 1
```

说明：此范例会在条码机 LCD 上显示要求分两次输入重量（Weight）与单价（PRICEUNIT），并将重量与单价相乘后打印出结果（RESULT），而重量与单价数据并不会打印在卷标上，下面打印结果是以鱼（Fish）的重量为 12 公斤、单价为 2.65/公斤，结果为 31.79：



对齐 – [J:...]

语法：**[J:ml]**

J = 对齐指令

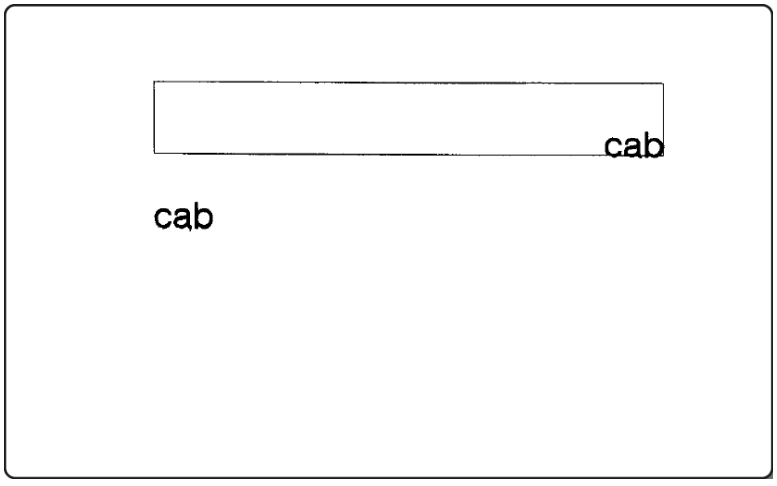
m = **l**：靠左（left）
c：置中（centered）
r：靠右（right）

l = 文字对齐位置的特别区域长度设定

範例：

```
J
S l1;0,0,68,71,100
G:AREA;10,10,0;R:70,10,.2,.2
T:NOADJUST;10,30,0,3,5;cab
T:ADJUST;10,20,0,3,5;cab[J:r70]
A 1
```

说明：在 NOADJUST 指令行里的 cab 不会做对齐的动作，而 ADJUST 指令行的 cab 会做靠右（长度 70 位）对齐



存取资料名 – [name]

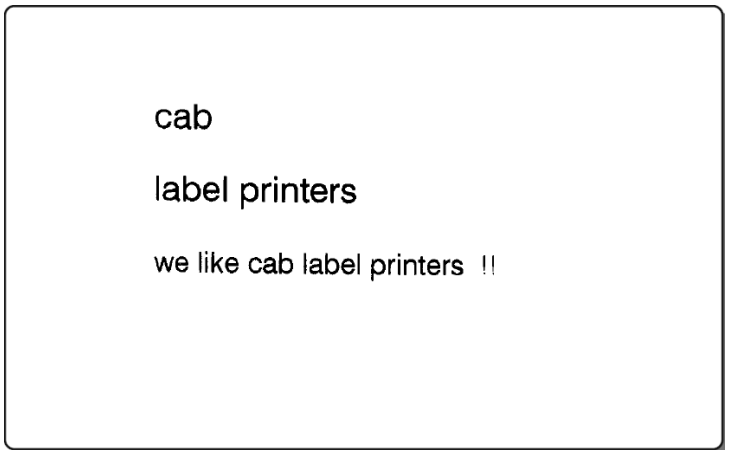
语法：**[name]**

name = 定义数据内容于唯一的数据名，供后续程序呼叫使用，其中数据名有字母大小之区别；

範例：

```
J
S l1;0,0,68,71,100
T:FIELD1;10,20,0,3,5;cab
T:FIELD2;10,30,0,3,5;label printers
T:FIELD3;10,40,0,3,4;we like [FIELD1] [FIELD2]  !!
A 1
```

说明：将 cab 定义为 FIELD1，再将 label printers 定义为 FIELD2，最后将 FIELD1 及 FIELD2 与 we like!!定义为 FIELD3 并印出 FIELD1、FIELD2 与 FIELD3：



复制字符串 – [name,m{,n}]

语法：**[name,m{,n}]**

name = 定义数据内容于唯一的数据名，供后续程序呼叫使用，其中数据名有字母大小之区别；

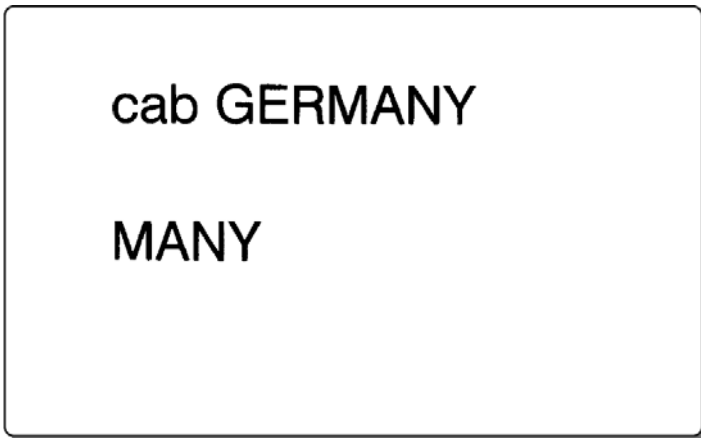
m = 要复制字符串的第一个开始复制字符位置

n = 要复制字符串的字符总数

範例：

```
J
S l1;0,0,68,71,100
T:ORIGINAL;10,20,0,3,8;cab GERMANY
T:CUTOFF;10,40,0,3,8;[ORIGINAL,8,4]
A 1
```

说明：将 cab GERMANY 定义为 ORIGINAL，再将 ORIGINAL 的数据内容的第 8 个字符开始复制，共复制 4 个字符，即复制 MANY，然后将 MANY 定义于 CUTOFF 数据名，并打印出 ORIGINAL 与 CUTOFF：



读取暂存档 – [RTMP…]

语法：**[RTMP]**

语法：**[RTMP:x]**

RTMP	=	读取 TMP 暂存档
x	=	设定数据重复次数

序列号 – [SER：…]

语法：**[SER:start{,incr{,freq}}]**

SER = 启用序列号指令

start = 序列号初值；

incr = 序列号增加值

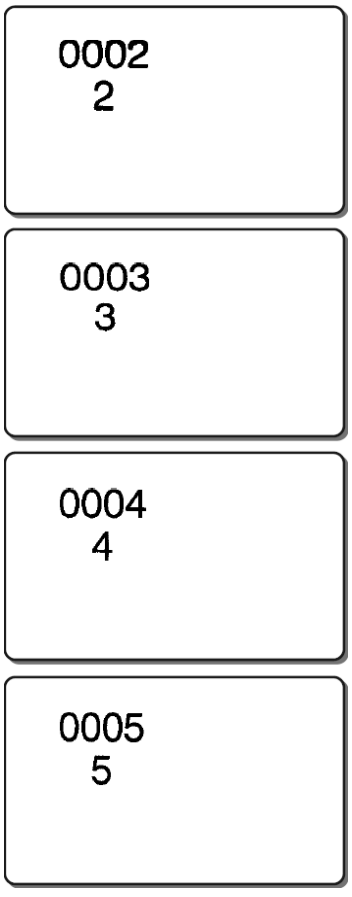
freq = 每隔几张跳一次序列号

如未设定 incr 与 freq，则条码机会自动以默认值 1 为设定；

範例：

```
J
S l1;0,0,68,71,100
T:CNT; 10,15,0,3,10;[SER:1][I]
T:FIELD1;10,10,0,3,10;[+:1,CNT][C:0][D:4,0]
T:FIELD2;10,20,0,3,10;[+:1,CNT][C: ][D:4,0]
A 4
```

说明：此例是序列号累加值为 1，但不打印序列号，而是将序列号定义为 CNT，并将 CNT 与 1 做加法运算，然后把运算值再定义为 DIELD1 与 FIELD2，再打印出 FIELD1 与 FIELD2；



Unicode 字符 – [U：x...]

语法：**[U:x]**

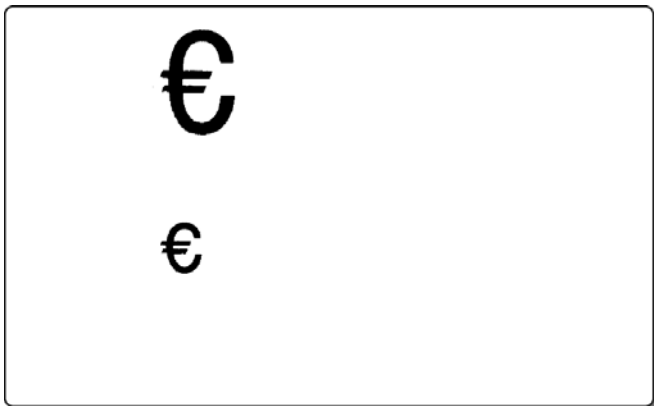
U = 选用 Unicode 字符

x = 以 $ 符号或 ASCII 句柄名表示 16 进制值，如 NUL、SOH、STX、ETX、EOT、ENQ、ACK、BEL、BS、HT、LF、VT、FF、CR、SO、SI、DLE、DC1、DC2、DC3、DC4、NAK、SYN、ETB、CAN、EM、SU、ESC、FS、GS、RS 与 US，或 128 码的句柄：FNC1、CODEA、CODEB、CODEC；

範例：

```
J
S l1;0,0,68,71,100
T 20,15,0,3,20;[U:$20AC]
T 20,40,0,596,10;[U:$20AC]
A1
```

说明：[U：$20AC]为欧元符号，另外，[U：FNC1]为设定 128 码的码 1 字符；



英文大小写转换 –[UPPER：…]、[LOWER：…]

语法：**[UPPER:Name]**

语法：**[LOWER:Name]**

UPPER = 将英文字母做大写转换；

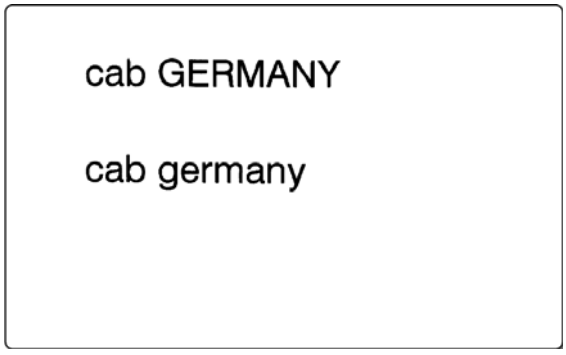
LOWER = 将英文字母做小写转换；

範例：

```
J
S l1;0,0,68,71,100
T:Input;10,20,0,3,8;cab Germany
T:UPPERCASE;10,40,0,3,8;[UPPER:Input]
A 1
```

範例：

```
J
S l1;0,0,68,71,100
T:Input;10,20,0,3,8;cab GERMANY
T:LOWERCASE;10,40,0,3,8;[LOWER:Input]
A 1
```



写入 LOG 檔 – [WLOG]

语法：**[WLOG]**

WLOG = 将数据写入记忆卡的 LOG 文件，可让打印卷标或记录使用

範例：

```
E LOG;EXAMPLE
T:VAL;    5,6,0,3,3;[SER:0001]
T:PRINT;5,6,0,3,3;Label [VAL] printed at [DATE] um [TIME].[WLOG][I]
```

说明：此范例是将 VAL 计数值写入 LOG 文件以便记录与打印卷标数量

写入暂存档 – [WTMP]

语法：**[WTMP]**

WTMP = 将数据写入条码机记忆卡内之已定义的暂存档；

範例：

```
E TMP;EXAMPLE
T:XVAL;10,10,,0,3,3;[RTMP,3][I]
T:SERNO;10,10,0,3,3;[+:XVAL,1][C:0][I][WTMP]
T:TESTFELD;10,10,0,3,3;Serial number is: [SERNO]
```

说明：定义暂存盘名为 EXAMPLE，并将暂存盘的数据定义为 XVAL，及 XVAL 的值会以暂存档 EXAMPLE 为依据；

cab 数据库连结指令

注意：欲使用数据库连结功能，必须安装支持的硬件。

cab Database Connector（软件）

该软件可让 cab A 系列条码机及网络卡透过 TCP/IP 传输，打印包含 SQL 兼容之数据库的卷标，可经由条码机的按键呼叫数据；
目前不使用 cab Database Connector 软件的方式是，必须将储存于记忆卡之固定格式的数据读取到条码机，此方式缺点是，该数据必须经过转换，当数据越多，速度就越慢，因此并不实际，且改变核心数据库时，则要更新条码机上的记忆卡；
而使用 cab Database Connector 软件的方式则不同，该软件可从网络上的任何地方呼叫现存的数据库，如打印新卷标时，在该数据库异动的同时，该软件也会立即更新数据；

使用数据库连结时所需配备：

- A 系列条码机
- 具备网络卡与支持 cab Database Connector 功能
- CF 记忆卡
- 输入装置（HL30 USB 扫瞄枪或 USB 键盘）
- cab Database Connector 软件

cab A 系列条码机都具 SQL 客户端连结功能，可直接经由 cab Database Connector 软件与 TCP/IP 网络与数据库服务器做连结，所有 ODBC 数据库或微软 OLEDB 接口都可支持；在 cab Database Connector 伺服端上，都可同时间取得数个 tables 与 fields，多个已定义的卷标可以由记忆卡上的内容表上选择；

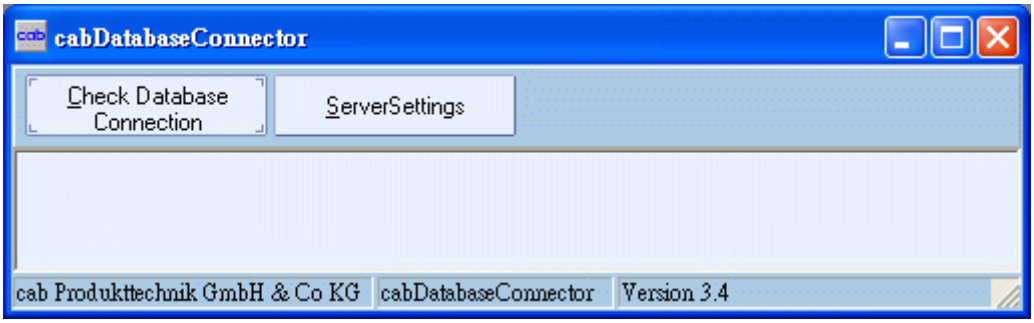
运作原理：
cab SQL 客户端透过 TCP/IP 网络连接 cab Database Connector 并送出 SQL 查询，cab Database Connector 接收到 SQL 查询后会经由 ADO（ActiveX DATA Object）送出该查询到数据库伺服端；
cab Database Connector 从数据库伺服端收到数据记录后，会经由 TCP/IP 方式送出该数据记录到 cab SQL 客户端，条码机的 SQL 客户端就会收到原先要求的数据记录；

支持的数据库：
MS Access, Ms SQL Server, Oracle, Dbase 与 ODBC 连结

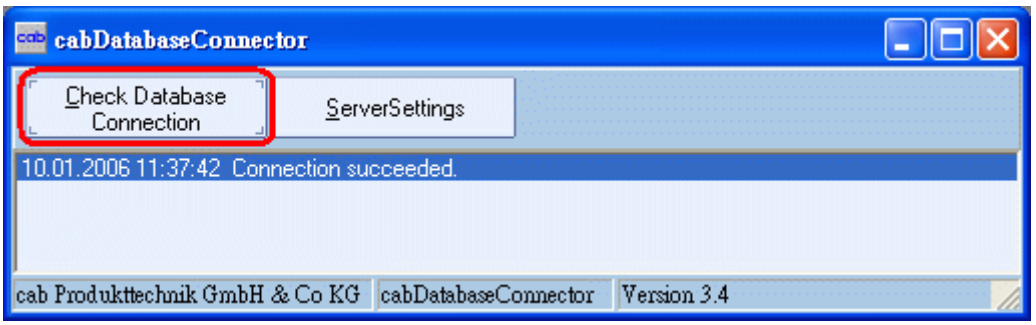
重要事项：必须安装 Jet40Sp3_Comp.exe 与 mdac_typ.exe，通常安装 Windows 2000 或 Office 2000 后，都会有这些档案，如使用 Windows 98/95 时，则需要另外安装这档案，可在微软网站（www.microsoft.com/data）下载。

cab Database Connector 与 A 系列条码机之 SQL 客户端功能
A 系列条码机可由 cab Database Connector 软件与内建的 SQL 客户端功能透过 TCP/IP 网络直接从数据库取得所需数据；当使用 A 系列条码机单机操作功能（即不外接计算机）时，可不需再记忆卡上储存与维护数据表；
使用者可连结所有具 ODBC 或微软 ADO 接口的数据库；
同时可连结数个数据表（table），且速度会比使用记忆卡连结数据还快上许多。

操作方式
步骤一
将 cabDatabaseConnector.exe 复制到计算机任意指定的目录下，然后执行该软件，cab Database Connector 并不需要任何 DLL 文件或其它程序，除非是系统本身所需的档案（如重要事项里所提到的档案）；
执行软件后会出现的窗口如下：

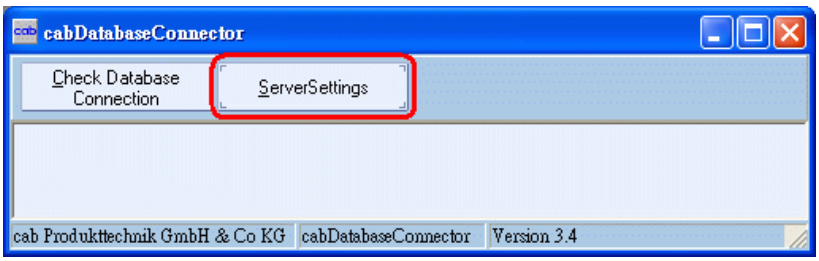


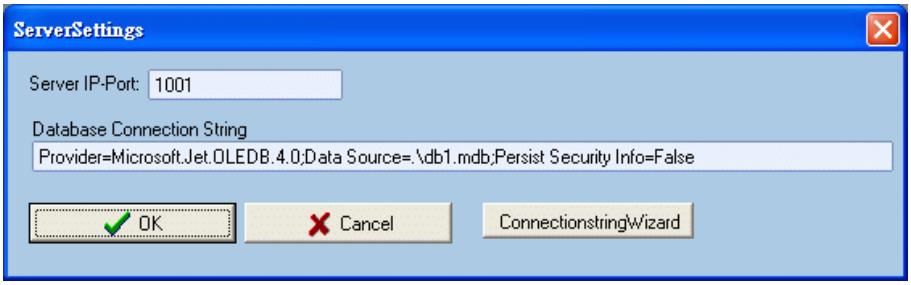
步骤二
先按下 Check Database Connection 检测计算机与条码机网络连结是否正常，正常则会出现连结成功的讯息：



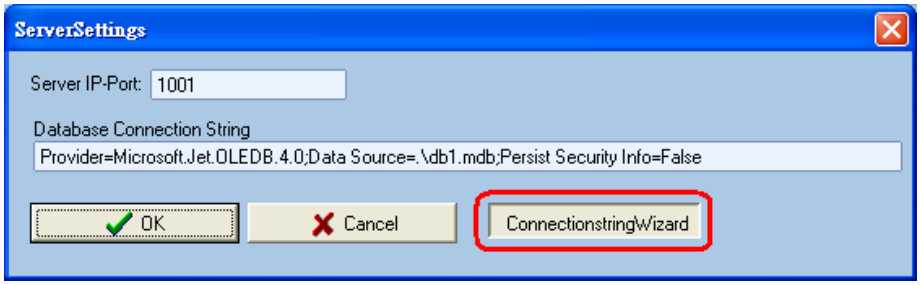
步骤三

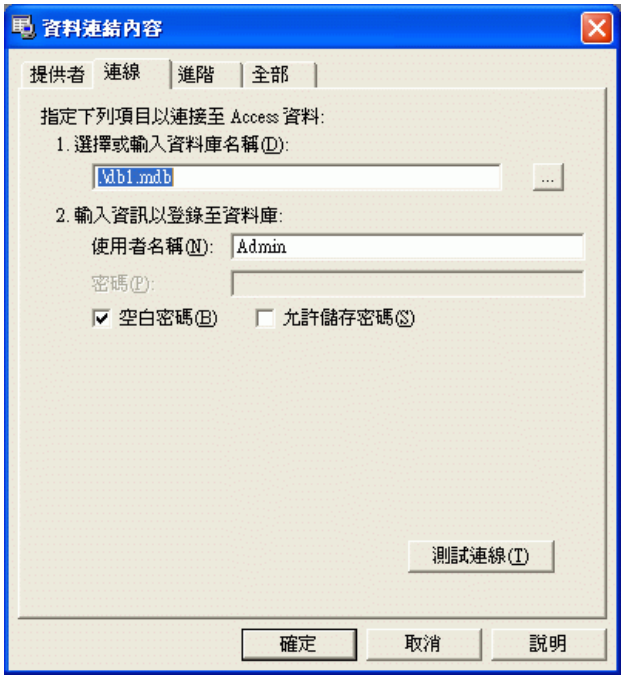
按下 ServerSettings 键，可做数据库连结的更改：





在 Database Connection String 下行里，键入完整的数据库连结路径，或可使用连结路径精灵（ConnectionstringWizard）做进阶设定：





进阶设定部分则需要对数据库有基本的了解；

连结路径范例：

MSAccess: Provider=Microsoft.Jet.OLEDB.4.0;Data-Source=<DatabasePath+MDB-Filename>

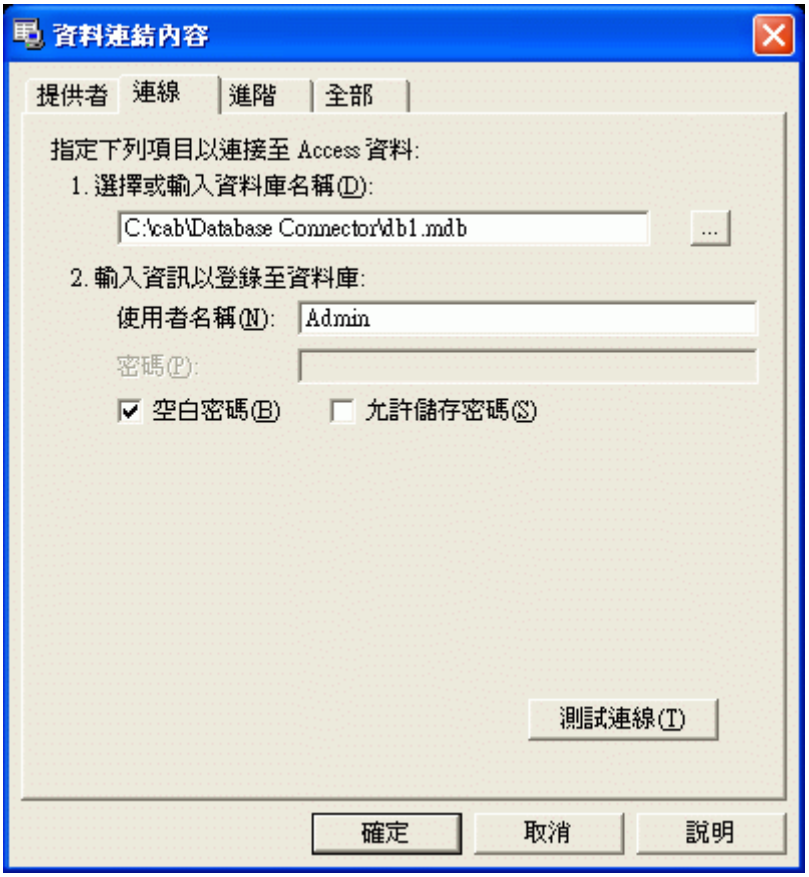
ODBC: in most cases simply type in the ODBC-Datasourcename

MSSQLServer: Provider=SQLOLEDB.1;Integrated Security=SSPI; Persist SecurityInfo=False;Initial Catalog=cab; Data Source=hostname

ORACLE: Provider=MSDAORA.1;User ID=User; Data Source=Prod;Persist Security Info=False

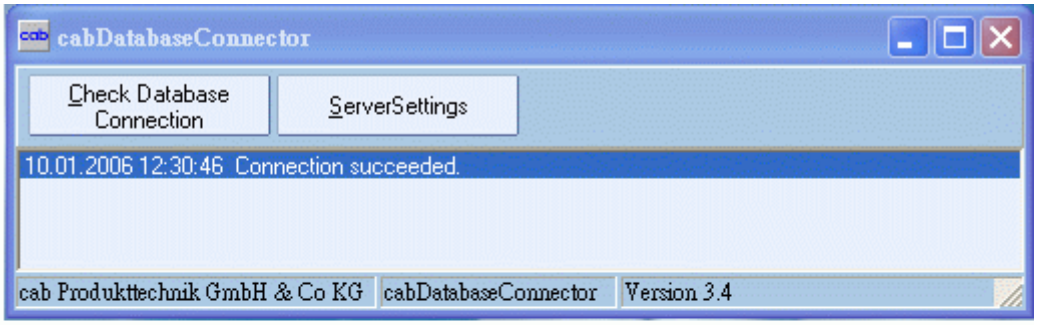
Dbase: DSN=ExampleDatasource;DBQ=<DatabasePath>; DefaultDir=<DatabasePath>;FIL=dBase IV

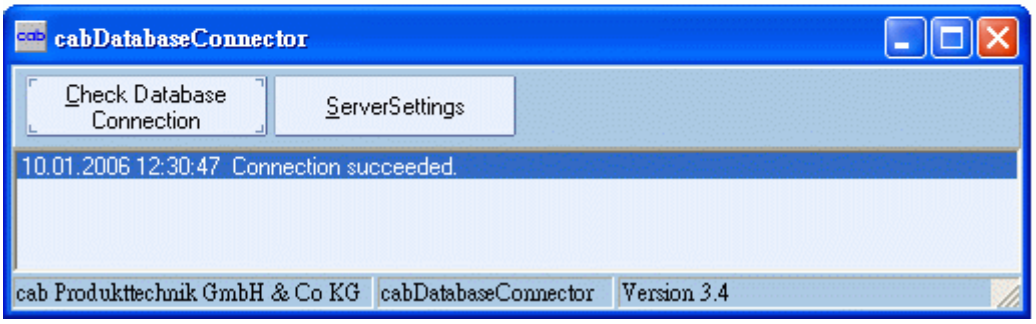
如以所附的范例档案（db1.mdb）为例，则指向 cab Database Connector 目录：



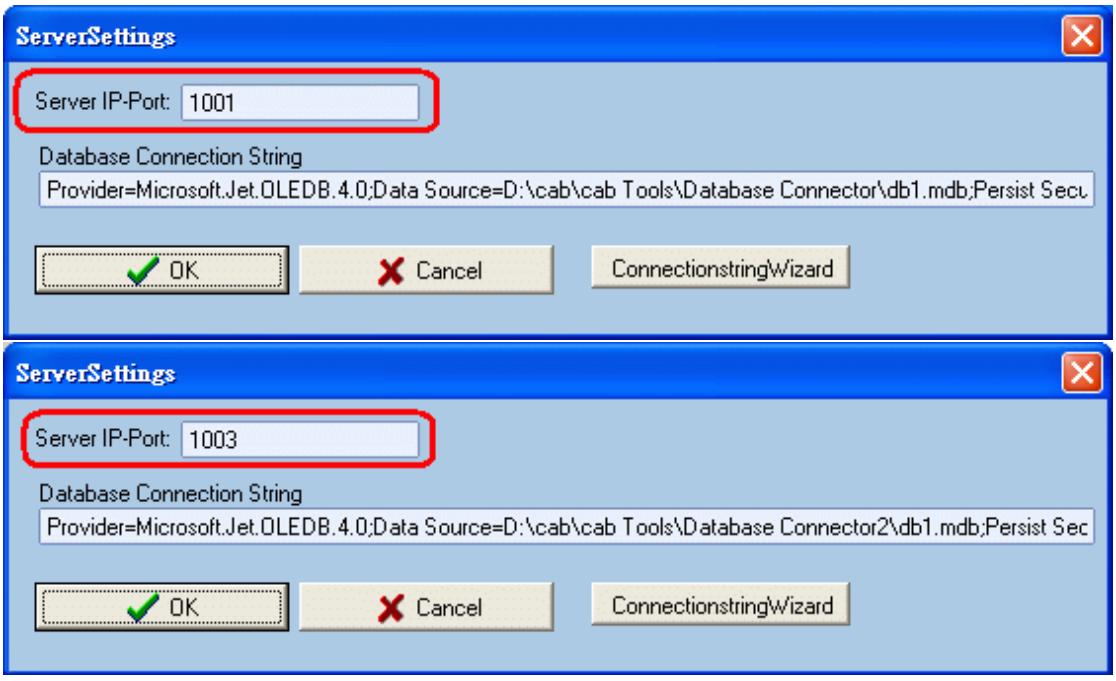
注意：

cab Database connector 可在一台计算机上开启数个窗口做网络连结：





欲使用多连结时，仅需更改 IP-Port 即可：



更改方式是开启 cab Database Connector 目录里的 cabDatabaseConnector.ini，

[IP]

Port=1001

1001

将 1001 改为其它可用的接口，如 1003 即可；

步骤四

将设计好的卷标文件储存于 A 系列条码机的记忆卡里，在该卷标文件里必须加入数据库连结指令才能正确连结数据库；

数据库连结指令：**E**

语法： **E SQL；<IP of cabDatabase connector >：Portnumber**

说明：

IP 地址是设定 cab Database Connector 所在的计算机地址，portnumber 则是要与 cab Database Connector 设定的 Port 相同；

范例： **E SQL**；192.168.0.80:1001

说明： 连接计算机的 IP 地址是 192.168.0.80，Port 为 1001

数据库存取功能

语法： [**SQL:Select** Field **from** Table **where** Searchvalue='{Fieldname}']

说明： SQL 指令用于 SQL 数据库存取数据

范例：

T 10,15,0,3,5;[SQL:SELECT PRODNAME FROM TA WHERE ARTICLE= '{ARTNR}']

SPLIT 指令

语法： [SPLIT:Field,Index]

范例： T 10,5,0,3,5;[SPLIT:RESULT,1]

下列为接下来的范例能成功连结数据库的必要条件：

- A 系列条码机有外接 USB 键盘
- 有安装 CF 记忆卡
- A 系列条码机需有安装网络卡且支持数据库连结功能
- cab Database Connector 有正确启动与设置
- 使用的数据库必须是有效的，范例使用的 table 名是 TA，field 名是 ARTICLE，其对应值是 “{ARTNR}”，PRODNAME 内容将会是由数据库提供
- 下面卷标范例须储存于记忆卡

范例：

行数	程序内容
1.	m m
2.	J
3.	S l1;0,0,68,70,100
4.	H 200
5.	E SQL;192.168.0.128:1001
6.	T:ARTNR;10,5,0,3,5;[?:Article Nummer,5560432,1,R,D]
7.	T 10,15,0,3,5;[SQL:SELECT PRODNAME FROM TA WHERE ARTICLE='{ARTNR}']
8.	A 3

说明：

行 1：设定长度单位（mm）

行 2：开始程序

行 3：设定卷标尺寸，此范例是卷标设定为宽度 100mm、高度 68mm

行 4：设定打印速度为 200 mm/s

行 5：设定连接计算机的 IP 为 192.168.0.128 及数据库连结的 port 为 1001

行 6：定义卷标文字内容，并会显示在条码机 LCD 操控面板上（T:ARTNR…）

行 7：设定 SQL 请求与同时定义位置及数据区字型

行 8：设定打印张数为 3 张

希爱比科技股份有限公司
台北县板桥市民生路一段 33 号 7 楼之一
Tel： +886 (0) 2 29509185
Fax： +886 (0) 2 29509183
http://www.cabasia.net
email: cabasia@cabgmbh.com