NOVEXX SOLUTIONS

PROGRAMMING MANUAL

Easy Plug

Edition 15 - 1/2024

Contents

What is Easy Plug?

- Easy Plug is a hardware-dependent *communications interface* for label printers and print & apply systems (called „printers“ afterwards) from NOVEXX Solutions. The data interface set at the printer is used for communication from the host to the printer. Easy Plug is designed for a 7 or 8-bit data format (ASCII).
- Easy Plug is a label-orientated (task) *printer control system*, i.e. the host computer is able to be assigned other tasks after one or more label formats have been transmitted. The printer produces the required number of labels independently.
- Easy Plug manages a label format and stores subsequent jobs in an integrated *spooler*. An external spooler is not required. Adding an additional spooler can impair the Easy Plug functions.
- Easy Plug communication is divided into individual *commands*. Command parameters can be single characters or texts of any length.

Notes about the command descriptions

VALIDITY

Easy Plug commands, which are described in chapter „Command reference“ of this manual, are valid for the current firmware versions according to the following table:

Printer	Firmware version
64-xx Gen. 3	6.75
ALX 92x	
ALX 73x (PMA)	
DPM, PEM	
XLP 50x	7.75
XLP 51x	MAR-3.0
XLP 60x	BEL-4.0
XPA 93x	1.02

PRINT JOB STRUCTURE

A print job consists of one or more Easy Plug commands, listed in a text file.
The simplest way to start a print job is to send the text file, containing the Easy Plug commands, to the printer. This can be done easily via copy-command, using the terminal window of your PCs operating system. Given the printer is in online mode, it starts printing after few seconds of interpretation time.

Structure		Commands (Example)
1.	Activate the printer	`#!A1`
2.	(Optional) Send the logo(s): If required, the logos must be sent to the printer before formatting begins.	`#G No Logos provided`
3.	Prepare the print series: Definition of material type and dimensions, gap offset and national character set.	`#IMS70.0/85.0`
4.	Start the label formatting	`#ERY`

Structure		Commands (Example)
5.	Label format: Font, bar code, logo, line, rectangle (type, size, direction of rotation, position).	`#J66.0#T15.0#M2/2` `#YT107/0///THERMO` `#J60.0#T20.5#M1/1` `#YT106/0///PRINTING-SYSTEM` `#J50.0#T20.5` `#YT104/0///The easy way` `#J45.0#T15.0` `#YT104/0///to create your la▶` `bels` `#J25.0#T18.5` `#YB1/0M/7/3///123456789012` `#J15.0#T11.0#M1/1` `#YT104/0///PRICE` `#J15.0#T37.0#M2/2` `#YT106/0///120,95` `#J28.0#T11.0#M1/1` `#YT103/1///90-degree-rotation` `#J7.0#T51.0` `#YT104/2///180-degree-rotation`
6.	Conclude the label format (print quantity).	`#Q1/`

Table 1: Print job example with print quantity 1.

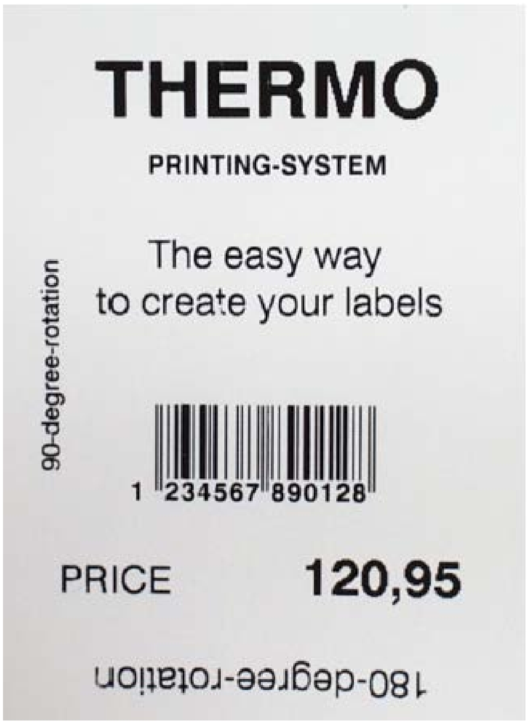


Fig. 1: Printout of the example format.

Related reference

Overview of command groups on page 135

DECLARATIONS

(blank)	Blank entry, no entry required, command parameter can be left out
int	Decimal value, integer, absolute (integer)
num	Decimal value, with following point positions, absolute (numerical)
name	Text string which names a variable – the following special characters may not be part of the string:

Character	Name	ANSI	Hex
		<32	<20
	blank	32	20
!	exclam	33	21
‘´	quotedbl	34	22
#	numbersign	35	23
,	comma	44	2c
.	period	46	2e
+	plus	43	2b
-	hyphen	45	2d
*	asterisk	42	2a
/	slash	47	2f

expression	Can contain konstant values, variables and functions
TEXT	Text string, alphanumerical
#	Commando character, designates a command

VARIABLE DATA FIELDS

If a variable data field (text field) is to be used, the variable data field is defined with the help of TEXT as follows:

```
TEXT = $n,c
```

$	Marker
n	Number of the variable data field [0...99]
,	Separator
c	Number of characters in the variable data field [0...255]

Example:

```
#YTz/0D///$n,c#G
```

INPUT FIELDS

Input fields are not filled with content, until the print job is started.
The input for each input field is queried only once, shortly after the print job was started. This input counts for all labels of the label amount designated in the print job.

```
TEXT = $<prompt>,default
```

$	Marker
<prompt>	Arbitrary text which shows up on the printer display together with the value default. This text allows to distinct several input fields. This text will not be printed.
,	Separator
default	Defines a default value for the respective field, which appears on the printer display together with prompt.

Example:

```
#YT101////$<Price:>,10,- Euro#G
```

For detailed information about input fields and the standalone functionality of the printer, please refer to the User Manual of the printer under „Standalone operation“.

Command reference

IMMEDIATE COMMANDS

#!An - Interface activation

The immediate command #!An activates Easy Plug (after the device has been switched on or after it received a passive command #!P1).
Following command #!A1, all incoming data is evaluated as Easy Plug commands.

After the device has been switched on, the active command must be sent once, otherwise the printer will not work.

Syntax

```
#!An
```

The command does not work in standalone mode.

Parameter	Value	Description
n	int	Set printer ID number [0...31].

#!CA - Clear All

The immediate command #!CA resets Easy Plug to a predefined initial status

- After a started label has been completed, an active printing process is stopped and broken off.
- The label format memory is erased.
- Material settings remain unchanged (width, length, reel/gap, material designation).
- The Easy Plug interface remains activated (that is, the effect of a preceding #!A1 command is maintained).

  But: During the act of erasing, for a short moment, no data can be transmitted (because the handshake lines are deactivated for an instance).

- Spooler (queue) is erased.
- The error number for the #!Xn response string is deleted, if the error is already acknowledged.

Syntax

```
#!CA
```

The command does not work in standalone mode.

#!CF - Clear Format

The immediate command #!CF breaks off the printing of the series currently being printed.

- The current printing process is stopped after a label which has already been started is completed.
- The label format memory is erased. Data then present in the spooler buffer is processed further.

Syntax

```
#!CF
```

‖ The command does not work in standalone mode. ‖

#!D - Trigger Single-Start

The immediate command #!D triggers exactly one Single-Start.

> Start signal input must be activated!
>
> The command #!D can not be used as start signal in applicator operation mode, if the applicator is PLC-controlled (64-xx with LTSI).

Syntax

```
#!D
```

‖ The command does not work in standalone mode. ‖

#!HP - print head pressure setting

‖ Only applicable for machines of the types XPA 93x and XLP 60x. ‖

The #HP command sets the print head contact pressure to the value x.

Syntax

```
#!HPx
```

‖ The command does not work in standalone mode. ‖

Parameter	Value	Description
x	num	Print head pressure [1.0..3.0] • 1.0 = low print head contact pressure (equals the setting „I“ of the adjustment knob at older machines) • 3.0 = high print head contact pressure (equals the setting „III“ of the adjust-ment knob at older machines)

Example

- The print head pressure setting is taken over from the setting in parameter Print > Head pressure:

```
#!A1
#IMN50/50
#ER
#T1#J1#YT109/0///HP Para#G
#Q2/
```

- The print head pressure setting is taken from the preceding HP command:

```
#!HP2.1
#ER
#T1#J1#YT109/0///HP2.1#G
#Q2/
```

- The print head pressure setting is taken over from the setting in parameter Print > Head pressure:

```
#ER
```

```
#T1#J1#YT109/0///HP Para#G
#Q1/
```

Related reference

#HP - print head pressure setting on page 39

#!H - Print Head temperature setting

The immediate command #!H sets the print head voltage and therefore the print head temperature to the required value n.

Syntax

```
#!Hn
```

The command does not work in standalone mode.

Parameter	Value	Description
n	int	Print head voltage [0...xxx] (value xxx depends on the printer type, see table below) 0 = lowest print head temperature xxx = highest print head temperature

Printer	xxx
64-xx, DPM, PEM, ALX 92x, ALX 73x	110
XLP 504/506 300 dpi	120
XLP 504 600 dpi	100

Table 2: Maximum values for the print head voltage depending on the printer type.

CAUTION!
The print head temperature (= n-value) directly affects the life of a print head. It counts: “The higher the temperature, the printhead is driven with, the lower is its life durance”. This counts even more if HV-values above 100% are driven. Therefore note:

► Always choose the lowest possible n-value necessary to produce an acceptable print result.

Related reference

#HV - Print head temperature setting on page 39
The #HV command sets the print head voltage and therefore the print head temperature to the required value n

#!Pn - Interface deactivation

The immediate command #!Pn switches Easy Plug into a passive state.
After receiving the command #!P1, all incoming commands are ignored with the exception of the active command #!A1.

‖ After #P1 the active command must be sent once, otherwise the printer will not work. ‖

Syntax

```
#!Pn
```

‖ The command does not work in standalone mode. ‖

Parameter	Value	Description
n	int	Set printer ID number [0...31]

#!PC - Setting start offset

The immediate command #!PC sets the start offset (equals DISPENSER PARA > Start offset or Options > Dispenser > Start offset).

Syntax

```
#!PC6004/<Wert>
```

‖ The command does not work in standalone mode. ‖

Parameter	Wert	Bedeutung
<Wert>	num	[0-999.9] mm ‖ As decimal separator has to be used a dot. ‖

#!PG - Reading out Parameters

The immediate command #!PG reads out parameter settings via the set interface.
The currently set parameter value is given back as ASCII text, which is terminated with <LF> (0x0a).
Additionally, there is the option of reading out all parameters together. Those can then be transferred to another printer (see command #PC).

‖ In case of a not available ID, only <LF> is returned.
Even those kind of parameters may be read out, which only appear under certain conditions in the menu. ‖

Syntax

```
#!PG<ParameterID>#G
```

‖ The command does not work in standalone mode. ‖

Parameter	Value	Description
<ParameterID>		Parameter ident number (ID). Each parameter in the menu has an own Parameter-ID.
		Alternatively, one of the following values can be applied:

Parameter	Value	Description
	-1	Printer specific settings (e.g. printhead resistance, sensor settings) are commented out (by a preceding "*"). By doing so, these settings are not transferred to another device (example 1).
	-2	Printer specific settings (e.g. printhead resistance, sensor settings) are not commented out. By doing so, these settings are transferred together with the other settings.
	-4	Additional output of parameter names and values as comments. Furthermore, some printer settings are written at the beginning of the output file (example 2).
	-8	Additional output of parameter type and setting range as comments (example 3). More information about parameter types can be found further below.
	-16	Additional output of service data as comments (example 4).
	-30	Combination of the values -16, -8, -4, -2 by addition: (-16)+(-8)+(-4)+(-2)=(-30). This activates all above mentioned types of comments (example 5).

Examples:

Example 1: Printer specific commands „commented out":

Command	Result

`#!PG-1#G`	`#!A1`
	`#PC1001/1`
	`#PC1002/8`
	`#PC1003/4.00`
	`#PC1004/4.00 *PC1024/30`
	
	(preceding "*")

Example 2: Additional comments:

Command:

```
#!PG-4#G
```

Result:

```
#!A1
#G Machine Setup for AP 5.4   300 Dpi    Version:
 R2.32P PE2.32C
#G Serial number        :  12345657890
#G MAC Address        :  00.0a.44.02.09.89
#G Creation date        :  16.06.2004 09:23
#G -------------------------------------------------
#G Printer Parameter Menu
#G -------------------------------------------------
#PC1001/1 #G  Infeed no. : Nr. 1
#PC1002/8 #G  Inf. change spd. : 8 Inch/s
#PC1003/4.00 #G  Print speed : 4 Inch/s
#PC1004/4.00 #G  Feed speed : 4 Inch/s
#PC1005/1 #G  Materialtype : Punched
#PC1006/35.00 #G  Materiallength : 35.0 mm
#PC1007/104.00 #G  Materialwidth : 104.0 mm
#PC1008/0.00 #G  Punch offset : 0.0 mm
...
```

Example 3: Additional comments (setting range):

Command	Result
`#!PG-8#G`	`#PC1505/4000`
	`#G  <i: 1024 - 65535 (1)>`
	`#PC1506/0`
	`#G  <d>`
	`#G   0 = Automatisch`
	`#G   1 = 10M Halbduplex`
	`#G   2 = 10M Vollduplex`
	`#G   3 = 100M Halbduplex`
	`#G   4 = 100M Vollduplex`
	`#PC1508/novexx`
	`#G  <s: 16>`
	`...`

Example 4: Additional service data as comments:

Command	Result
`#!PG-16#G`	…
	`#G -------------------------------------------------`
	`#G Power supply data`
	`#G -------------------------------------------------`
	`#G #G  Typ : HME PSupply 450`
	`#G #G  Version : H0.40 F1.32`
	`#G #G  Serien Nummer : 02510012`
	…

Example 5: All types of comments:

Command	Result
`#!PG-30#G`	`#!A1`
	`#G Machine Setup for AP 5.4   300 Dpi    Version:`
	` R2.32P PE2.32C`
	`#G Serial number :  12345657890`
	`#G MAC Adress :   00.0a.44.02.09.89`
	`#G Creation date :  16.06.2004 10:33`
	`#G -------------------------------------------------`
	`#G Printer Parameter Menu`
	`#G -------------------------------------------------`
	`#PC1001/1 #G  Einzug-Nr. : Nr: 1`
	`#G  <i: 1 - 4 (1)>`
	`#PC1002/8 #G  Einzugs. Geschw. :  8 Inch/s`
	`#G  <i: 4 - 10 (1)>`
	`#PC1003/4.00 #G  Druckgeschwind. :  4 Inch/s`
	`#G  <f: 2.0 - 6.0 (1.0)>`
	`#PC1004/4.00 #G  Vorschubgeschw. :  4 Inch/s`
	`#G  <f: 2.0 - 6.0 (1.0)>`

Parameter types

The printer provides 4 parameter types, whose setting range and assignment can be queried.

Parametertyp	Representation	Values
Integer Parameter (Integer value; must match the setting range)	`<i: 1024 - 65535 (1)>`	Setting range: [1024...65535] Unit interval: 1
Float Parameter (Value with decimal places; must match the setting range)	`<f: 0.0 - 999.9 (0.1)>`	Setting range: [0.0...999.9] Unit interval: 0.1
String Parameter	`<s: 16>`	Maximum string length: 16 characters
Diskreter Parameter (Integer value with a special meaning)	`<d>`	0 = RS232 1 = RS422 2 = RS485

Table 3: Overview parameter types

#!SP - Stop printing

The immediate command #!SP stops the printing of a print job.

- Labels being printed are completed
- Series is halted
- Spooler is no longer processed
- Printing process only restarts after the printer receives a start command #!SR

Syntax

```
#!SP
```

‖ The command does not work in standalone mode. ‖

#!SR - Start printing

The immediate command #!SR starts the printing of a halted print job.
The remaining labels of a format which has been halted by the stop command #!SP are printed.

Syntax

```
#!SR
```

‖ The command does not work in standalone mode. ‖

#!Xn - Status acknowledgement

The immediate command #!Xn requests a status report of the printer.
Request for a status acknowledgement using the selected interface. For transmission via Centronics, nibble mode is used.

Notes concerning status acknowledgement via serial interface:

- Acknowledgement at RS232 will only work, if the CTS pin is set to TRUE
- The acknowledgement is given with the same parameters as the receive channel (same Baud rate, parity, number of stop bits, data bits)
- Acknowledgement takes place without a handshake and inter-character delay
- A delay period can be set
- The status number is maintained in the acknowledgement string until the status changes or a reset or the command #!CF is performed

Syntax

```
#!Xn
```

The command does not work in standalone mode.

Parameter	Value	Description
n	int	Acknowledgement delay period [0..9]
	0	0 ms
	1	100 ms
	2	200 ms
	...	
	9	900 ms

The printer answers to the immediate comand with a return string that composes of the following characters:

```
SaaaaAbcdMqqqqqqFeeeeeeKxxxxxxxxxxxxxxxx
```

The acknowledgement string is always 40 characters long (in ASCII, decimal digits, without spaces).

Mark	Characters	Value	Description
S	aaaa	0000	No status message available
		xxxx	Last status number (as shown in operating panel display)
A	b	0	If status has already been requested with #!Xn
		1	If status is being requested for the first time
	c	0	If status has already been acknowledged at the operating panel
		1	If status is displayed, but has not yet been acknowledged
	d	0	If there is no label format stored
		1	If a format has been opened, but not closed with #Q

Mark	Characters	Value	Description
		2	If a valid format is being printed
M	qqqqqq	int	Number of labels still to be printed
F	eeeeee		Number of bytes still available in the spooler buffer
K	xxxxxxxxxxxxxxxx		Firmware version no. – same text string appears on the lower display line shortly after switching on the printer

Table 4: Breakdown of the return string.

Example

Command	Description
`#!X0`	Status request with 0 ms delay period. Example acknowledgement: `S0000A100M000000F065536KV6.37`

#!XC - Pharmacy Code acknowledgement

The immediate command #!XC returns the data of the last printed pharmacy bar code.

Syntax

```
#!XC
```

‖ The command does not work in standalone mode. ‖

The return value occurs in the form `xxxxxxxyyyyyyyy`, with:

xxxxxxx	7-digit CNK number
yyyyyyyy	8-digit sequential number

If no pharmacy bar code was printed since the last printer reset, the return value is `000000000000000`.

#!XMn - Diagnose dump / Read maschine status

The immediate command #!XMn reads out different machine states or Trigger Diagnose Dump and read out diagnostic data.

Syntax

```
#!XMn#G
```

‖ The command does not work in standalone mode. ‖

Return format: Value + linefeed

Parameter	Value	Description
n	-99	Diagnose dump is sent to the debug interface (which normally is Com1) and is stored in the flash memory.
	-100	Diagnose dump is sent to the active data interface and is not stored in the flash memory.

Parameter	Value	Description
	1004	*Return value*: 0 or 1 0 = Online 1 = Offline
	1007	*Return value*: number of print jobs, which are ready to print
	1008	*Return value*: 0 or 1 0 = Easy-Plug interpreter is *not* active 1 = Easy-Plug interpreter is active
	1009	*Return value*: rest print amount of the active print job (equals the value returned by #!Xn)
	1010	*Return value*: 0 or 1 0 = An error message is currently displayed 1 = No error message
	1011	*Return value*: last or currently displayed error status number. Additional information about which of the two cases occurred gives #!XM1010
	1013	On-/Offline state; *Return value*:: 0 = Offline 1 = Online 2 = Online stopped
	1015	Printer activity; *Return values*: 0 = Printer is idle 1 = Printer is busy (print job is being processed or printed)
	1201	*Return value*: Displayed text upper display line Only works with printers that use a two line text dispplay (e. g. 64-xx), not with a graphical display (as with XLP 50x)
	1202	*Return value*: Displayed text lower display line Only works with printers that use a two line text dispplay (e. g. 64-xx), not with a graphical display (as with XLP 50x)

Example

```
#!XM1004#G
```

Output: “1” followed by “linefeed” is sent back to the host, if the printer is offline.

B

#BOF - Spooler buffer off

The command #BOF is used to block the interface spooler buffer. The spooler buffer is blocked after receiving a format, and is only released after the required number of labels has been printed.

Syntax

```
#BOF
```

The command must stand *outside* of the command sequence #ER to #Q!

#BON - Spooler buffer on

This command is used to activate the interface spooler buffer. It releases the spooler buffer after it has been blocked (with #BOF). In this mode several formats can be sent to the spooler (even during the printing phase).

Syntax

```
#BON
```

The command must stand *outside* of the command sequence #ER to #Q!

#BR - Break print job

The command #BR stops the printer in the print job which follows the command #BR.

Syntax

```
#BR
```

The command must stand *outside* of the command sequence #ER to #Q!

Single job mode must be activated on the printer to be able to use this command properly (SYSTEM PARAMETER > Single-job mode = „On“; or System > Print > Single-job mode = „On“).

Examples

Example 1:

```
#G ---------------------------------------
#G Printer switches to stop mode.
#G ---------------------------------------
#!A1
#BR
#IMN100/20
#ERN0//
#T10#J5#YT108/0///TEXT
#Q3/
```

Example 2:

```
#G ---------------------------------------------
#G The first job is printed immediately.
#G ---------------------------------------------
#!A1
#IMN100/20
#ERN0//
#T10#J5#YT108/0///
#Q3/
#G -----------------------------------
#G Printer switches to stop mode
#G -----------------------------------
#BR
#!A1
#IMN100/20
#ERN0//
#T10#J5#YT108/0///
#Q3/
#BR
```

C

#CBF - Bar code Codablock F

The command #CBF prints bar codes ot the type *Codablock F*.

Syntax

```
#CBF/dw/s/m/c/r/vop/a/TEXT#G
```

Command must be placed between #ER and #Q!

The command must be terminated by #G!

Parameter	Value	Description
d	0	Bar code in normal writing direction
	1	Bar code rotation 90 degrees
	2	Bar code rotation 180 degrees
	3	Bar code rotation 270 degrees
w	W	The counting field TEXT is incremented/decremented without carry, that is, only the units position is increased/decreased.
s	int	Bar code width [1...30]; Default: 1
m	int	Height of a Codablock-row [1...100] mm; Default: 5 mm
c	int	No. of columns [4...62]; Default: 10
r	int	No. of rows [2...44]; Default: 0 0: Number of rows is calculated by codablock. 1: Improper value, interpretation causes an error message.
v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]
a	int	No. of labels with constant No. [1...255].

TEXT		Any alphanumerical text not exceeding the max. string length of 1024 characters. Consider the limitations of the bar code type. The text field may contain an input field.

Related reference

Input Fields on page 10
Printer-internal bar codes on page 144

#CFN - Code 49

The command #CFN defines bar code of the type Code 49 (ANSI/AIM-BC6-2000 „Uniform Symbology Specification Code 49“)

Syntax

```
#CFNm/dkx/h/s/vop/a/TEXT#G
```

Parameter	Value	Description
m	0	Alphanumeric Mode
	1	Append Mode
	2	Numeric Mode
	3	Group Alphanumeric Mode
	4	Alphanumeric Mode, Shift 1
	5	Alphanumeric Mode, Shift 2
	6	Reserved
	7	Automatic Mode (default setting). The printer determines starting mode and encodation method by analyzing TEXT. This is the recommend mode.
d	0	Bar code in normal writing direction
	1	Bar code rotated by 90 degrees
	2	Bar code rotated by 180 degrees
	3	Bar code rotated by 270 degrees
k	M	Bar code with plain copy line The plain copy line can extendet beyond the right edge of the code.
	O	Bar code without plain copy line (default setting)
x	J	Plain copy line below the bar code
	A	Plain copy line above the bar code
h	int	Row height Row height = (h + 1) * PRINT PARAMETERS > Bar code multip.
s	int	Bar code width factor [1..30]

v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]

a	int	No. of labels with constant No. [1...255].

TEXT		Any alphanumerical text. The stipulations for the respective bar code must be taken into consideration. Max. number of characters: 255. Admissible special characters see below.

Special characters in „TEXT“

Use of special characters:

- *Automatic Mode*: The special characters <FNC1>, <FNC2> and <FNC3> can be placed anywhere in the user data to insert the corresponding FNC code into the bar code.
- *All other modes*: The user is responsible for the correct application of these special characters. Detailed knowledge about the code 49 is necessary. Admissible special characters see table below.

Character	Abbrev.	Name
<	S1	Shift 1
>	S2	Shift 2
:	FNC1	Function 1
;	FNC2	Function 2
?	FNC3	Function 3
=	NS	Numeric Shift

Table 5: Admissible special characters, if „Automatic Mode“ is *not* selected.

Examples

Automatic mode:

```
#!A1#IMN100.0/60.0#N9#ERN0///0
#T10#J4.5#CFN/M0/6/4///<FNC1>12345<FNC3>ABCDE#G
#Q1#G
#ERN0///0
#T7.5#J4.5#CFN/M0/6/4///MULTIPLE ROWS IN CODE 49#G
#Q1#G
#ERN0///0
#T7.5#J4.5#CFN/M0/6/4/1/1/EXAMPLE 2#G
#Q2#G
```

Manual starting mode:

```
#!A1
#IMN100.0/100.0
#N9
#PO0
#ERN0///0
#T15#J50#CFN2/M0A/4/4///12345=>ABCDEF<S>S#G
#Q1#G
```

Related reference

Printer-internal bar codes on page 144

#CF - Clear file

The command #CF deletes a file store on the RAM disc or on an external memory medium.

Syntax

```
#CF/f
```

Parameter	Value	Meaning
f	file-name	Filename, which is supposed to be deleted. ‖ Always write file names in *capital letters*! ‖

Example

The file “FONT222.AFF” is to be deleted from RAM disk:

```
#CF/A:\FONTS\FONT222.AFF#G
```

Related reference

File operations on page 139
Description of the correct path specification for file operations.

#CG - Adjust intercharacter gap

The command #CG adjusts the intercharacter gap. The gap size has to be given in dots.

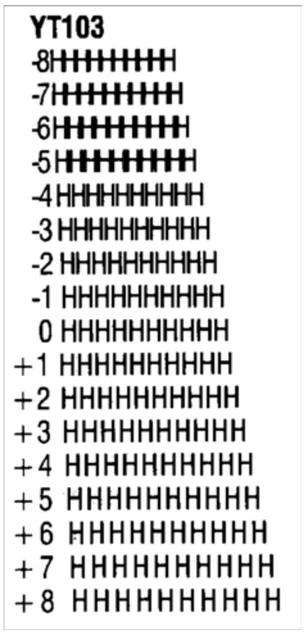
Syntax

```
#CGvn
```

Parameter	Value	Description
v	+	Enlarge the intercharacter gap compared with the default value
	-	Reduce the intercharacter gap compared with the default value
		‖ The default size of the character gap depends on the used font size ‖
n	int	Number of dots to adjust gap [-8...+8] ‖ 0: Sets the gap back to the default value ‖

Example

Command sequence	Printout

`#!A1#IMS82/180#ERN`
`#J175.0#T035#YT106/0///YT103`
`#J170.0#CG-8#T035#YT103/0///-8 HHHHHHHHHH#CG0`
`#J165.0#CG-7#T035#YT103/0///-7 HHHHHHHHHH#CG0`
`#J160.0#CG-6#T035#YT103/0///-6 HHHHHHHHHH#CG0`
`#J155.0#CG-5#T035#YT103/0///-5 HHHHHHHHHH#CG0`
`#J150.0#CG-4#T035#YT103/0///-4 HHHHHHHHHH#CG0`
`#J145.0#CG-3#T035#YT103/0///-3 HHHHHHHHHH#CG0`
`#J140.0#CG-2#T035#YT103/0///-2 HHHHHHHHHH#CG0`
`#J135.0#CG-1#T035#YT103/0///-1 HHHHHHHHHH#CG0`
`#J130.0#CGO#T035#YT103/0///0 HHHHHHHHHH#CG0`
`#J125.0#CG+1#T032#YT103/0///+1 HHHHHHHHHH#CG0`
`#J120.0#CG+2#T032#YT103/0///+2 HHHHHHHHHH#CG0`
`#J115.0#CG+3#T032#YT103/0///+3 HHHHHHHHHH#CG0`
`#J110.0#CG+4#T032#YT103/0///+4 HHHHHHHHHH#CG0`
`#J105.0#CG+5#T032#YT103/0///+5 HHHHHHHHHH#CG0`
`#J100.0#CG+6#T032#YT103/0///+6 HHHHHHHHHH#CG0`
`#J095.0#CG+7#T032#YT103/0///+7 HHHHHHHHHH#CG0`
`#J090.0#CG+8#T032#YT103/0///+8 HHHHHHHHHH#CG0`
`#Q1/`
	
	YT103
	-8HHHHHHHHHH
	-7HHHHHHHHHH
	-6HHHHHHHHHH
	-5HHHHHHHHHH
	-4HHHHHHHHHH
	-3HHHHHHHHHH
	-2HHHHHHHHHH
	-1 HHHHHHHHHH
	0 HHHHHHHHHH
	+1 HHHHHHHHHH
	+2 HHHHHHHHHH
	+3 HHHHHHHHHH
	+4 HHHHHHHHHH
	+5 HHHHHHHHHH
	+6 HHHHHHHHHH
	+7 HHHHHHHHHH
	+8 HHHHHHHHHH

#CIM - Cut

The command #CIM triggers a cut of the cutter or cutter-stacker, if available.

Syntax

```
#CIM
```

The command must stand *outside* of the command sequence #ER to #Q!

The command does not work in standalone mode.

Printer with RFID option: The #CIM command doesn´t work. Cuts have to be triggered using the #ER command.

Command placed after a print job: The label material is fed forward, is cut behind the last label and is fed backwards to the print position.

#CW - Cut width

The command #CW defines the cut width of the cutter.

Syntax

```
#CWab
```

Only applicable to printers with installed cutter option.

The command must be placed ahead of the #ER command.

Without the #CW command declared, the maximum cut width of the cutter is used (MAX_CUT_WIDTH, see table below).

The setting value indicates the cutting width in millimetres. However, deviations may occur depending on the cutting speed and material properties. It is recommended to determine the fine adjustment by testing.

Parameter	Value	Description
a	C	The printer uses the cut width set in PRINT PARAMETERS > Cut width or Options > Cutter > Cut width (depends on printer type).
b	int	Value for cut width; setting range: 0 - MAX_CUT_WIDTH

Printer	MAX_CUT_WIDTH
64-04	106
64-05	128
64-06	160
64-08	213
XLP 504/514 with 203 dpi	104
XLP 504/514 with 300/600 dpi	105
XLP 506 with 203 dpi	168
XLP 506/516 with 300 dpi	167
XLP 604	120

Table 6: The max. cut width for different printers.

D

#DC - Deleting all downloaded logos

The command #DC deletes all the logos present in the download memory and releases the entire memory capacity for use.

Syntax

```
#DCm
```

The command must stand *outside* of the command sequence #ER to #Q!

Parameter	Value	Description
m	blank	Deletes every logo on the internal RAM disk.
	A	
	C	Deletes every logo, which is stored on memory card in the folder `\logos`, if the file format is EPT. Logos which are not saved in file format EPT, but e.g. in JPG or BMP, are not deleted by this command.

#DF - Downloading a file

The command #DF downloads a file from a PC to the printer.

Syntax

```
#DF/s/f/i/b#G
```

Parameter	Value	Description
s	blank	Don´t overwrite file, if a file with the specified name already exists (default setting)
	N	
	O	Overwrite file
f		Filename, under which the downloaded data are saved. Always write file names in CAPITAL LETTERS!
i	int	File size in bytes [1... 2^{32}]
b		Binary file data To include the binary data of the file to be transmitted (see parameter b), you will need the help program „make_df.exe“. make_df.exe is contained on the Documentation-CD in folder „\utilities“.

Related reference

File operations on page 139
Description of the correct path specification for file operations.

Utility program „make_df.exe“ on page 142
The utility program „Make_DF.exe“ generates a complete #DF-command and writes it to a text file.

#DK - Downloading a logo

The command #DK is used to download a logo (to send a logo from the PC to the printer) under a reference number (0 to max. 255) which must be entered.

Syntax

```
#DKn/m/s/s.../s#G
```

The command must stand *outside* of the command sequence #ER to #Q!

The command #DK must be closed with #G.

Parameter	Value	Description
n	int	Logo reference number [0...255]
m	blank	The logo is copied on the internal RAM disk (default setting)
	A	The logo is copied on the internal RAM disk (default setting)
	C	The logo is copied on a memory card. The logo is copied to the `\logos` folder.
s	hex	Coding of a dot line in the logo matrix, hexadecimal with respectively 4 dots from left to right (000 to FFF). Assignment for hexadecimal coding: • 1 = Dot is printed • 0 = Dot is not printed Non-set dots at the end of the line can be left out. Every line in the logo matrix is coded by one or more parameters per line, from bottom (Line 1) to top. Admissible for hex coding are capital letters and numbers.

Example

```
#DK1/A/C03/E07/F0F/FFF#G
```

Logo number 1 with subsequent creation is sent and stored on the RAM disc.

```
Line 4: 1111 1111 1111 (/FFF)
Line 3: 1111 0000 1111 (/F0F)
Line 2: 1110 0000 0111 (/E07)
Line 1: 1100 0000 0011 (/C03)
```

Related reference

Logos on page 140
A logo is an image made of black and white points that is used as part of a label layout.

#DM - Downloading month names

The #DM command is for downloading any free definable name of a month.

Syntax

```
#DM^TEXT^TEXT
```

The command must stand *outside* of the command sequence #ER to #Q!

Parameter	Value	Description
	^	Control character for separating the individual month names.
TEXT		Name of a month, freely definable, maximum 20 characters. The TEXT after the first control character (^) is assigned to the month of January, the TEXT after the second control character (^) to February, etc. If less than 12 months have been defined, access to a missing month is ignored. The control character may not be used in the text.

Example

```
#G -----------------------------------------------------------------
#G The names of the months Jan to Dec are stored in the download buffer
#G (each with a closing space).
#G -----------------------------------------------------------------
#DM^Jan^Fev^Mar^Avr^Mai^Jui^Jul^Aou^Sep^Oct^Nov^Dec
```

#DO - Deleting one downloaded logo

The command #DO deletes one single logo from the RAM or an external memory medium.

Syntax

```
#DOn/m
```

The command must stand *outside* of the command sequence #ER to #Q!

Parameter	Value	Description
n	int	Reference number of the logo to be deleted [0...255]
m	leer	Deletes the logo on the internal RAM disk
	A	Deletes the logo on the internal RAM disk
	C	Deletes the logo on an external memory medium

Example

Logo number 5 is deleted and the corresponding memory capacity is made available again:

```
#DO5
```

E

#EMU - 300 dpi emulation

The #EMU command toggles the XLP 504 600 dpi between native (600 dpi) mode and 300 dpi emulation.

Syntax

```
#EMUn
```

‖ The command can be called as often as necessary between #ER and #Q. ‖

Parameter	Value	Description
n	0	*Native mode*: Print image is gererated with printhead resolution (600 dpi).
	1	*300 dpi emulation*: Print image is scaled so that the printout with the 600 dpi print head has the same size as if it was printed with a 300 dpi print head.

#ER - Start of label format

The #ER command marks the start of a label format (print job) and the definition of general information for this print job.

Syntax

```
#ERxrsftw/n/b/d/TEXT
```

Parameter	Value	Description
x	N	No change label
	Y	Change label [1] after series
r	int	No. of printed labels before cut (see figure below).
	0	No cut (default)
	blank	No cut (default)
	1	Each label / label row is cut
	2	Cut after every second label / label row
	x	Cut after every xth label / label row
s	D	Performs a dot check after the print job has been finished. ‖ Only with 64-xx, ALX 73x, ALX 92x, DPM, PEM ‖
f		Controls the ribbon saving function.
	C	Thermotransfer printing without ribbon saving
	F	Thermotransfer printing with ribbon saving
	T	Thermal printing without printhead lifting

1 A change label is somewhat longer than the preceding labels - this can be useful, for example when using a stacker. The longer change label then protrudes out of the label stack and makes it easier to sort the labels

Parameter	Value	Description
	U	Thermal printing with printhead lifting Only with 64-xx, ALX 73x, ALX 92x, DPM, PEM
	Z	Thermotransfer printing with turbo ribbon saving. Additional to normal ribbon saving, the label material is fed forward with a higher speed than print speed. This speed is set in PRINT PARAMETERS > Feed speed. Only with 64-xx, ALX 73x, ALX 92x, DPM, PEM
t	Pnum	[P0.0-P10.0] Delay time in seconds, until a new bit image is generated.
w	blank	After a stop by switching the printer offline, „Please wait…“ is displayed until the delay time (see above) is over. Only then, the printing process can be continued. The demanded delay time can not be overridden by quickly switching the printer offline and online again.
	V	A stop by switching the printer offline is executed immediately, without waiting for the delay time to be over. Thus, the demanded delay time can be overridden by quickly switching the printer offline and online again.

n	int	Number of labels in a label row. A label row is a line of labels rectangular to the feed direction. All labels in a row are printed at the same time (see fig. below)
	0	One label per row
	blank	
	1	One label per row
	2	Two labels per row
	x	x labels per row
		Incomplete label rows: If the print job provides less labes than are necessary to fill the last label row, the following happens: • Print job sent via data interface: The row is printed incomplete, starting the printout with the label at the material zero line side. • Print job started in standalone mode: The row is filled automatically and is printed completely.

b	num	Width of a label (including the distance to the neighbor label) Parameter b has only to be set for the use of multitrack label materials (n > 1). The number of tracks n multiplied by the label width b must not be larger than the material width, otherwise the overhanging track will not be printed.

d	num	Width of the double cut; Setting range: [0,00..5,00] mm If the double cut command is processed without specifying d, the value of the last processed print job or the value set by parameter menu will be used.

TEXT		Print job name (alphanumeric text with max. 255 characters string length).

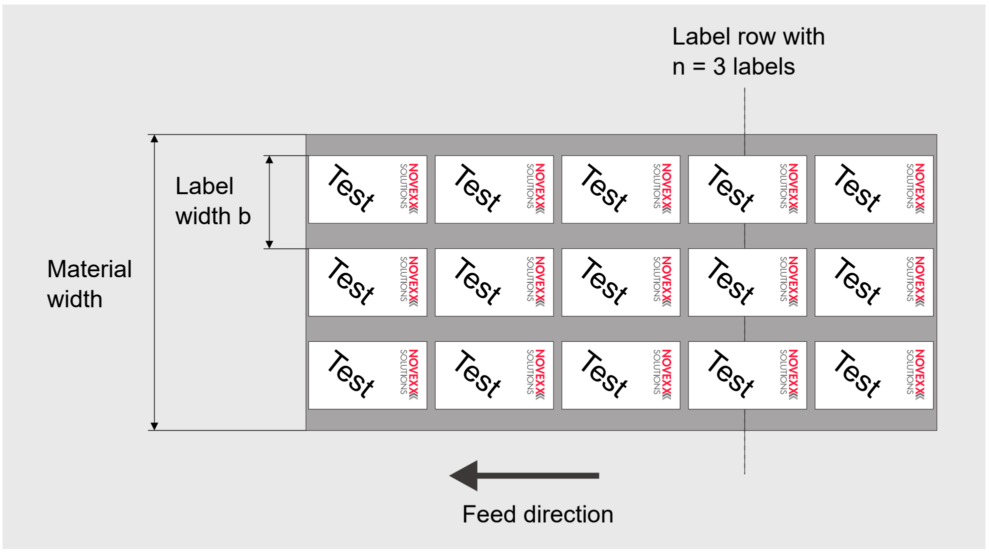


Fig. 2: Label material with 3 rows (n = 3).

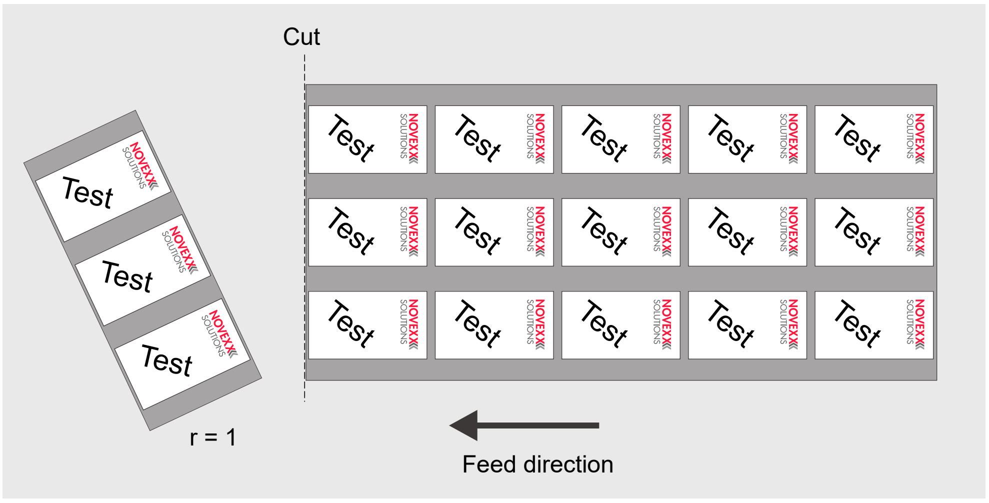


Fig. 3: Example of a setting with n = 3 and r = 1.

F

#FC - Material feed with cut

The #FC command feeds the material through by one label length followed by a cut.

```
#FC
```

The command must stand *outside* of the command sequence #ER to #Q!

The command does not work in standalone mode.

- *Gapped material*: feeds to the next gap or reflex mark
- *Endless material*: feeds by a defined label length

#FD - Field orientation and options

The #FD command defines the orientation and the appearance of a field.

Syntax

```
#FD/d/pz#G
```

Command must be placed between #ER and #Q!

Parameter	Value	Description
d		Rotation direction for *fixfonts*:
	0	normal writing direction
	1	Text rotated 90 degrees
	2	Text rotated 180 degrees
	3	Text rotated 270 degrees
	Dnum	[D0.0 ... D359.9] rotation direction for *scalable fonts*
p		Field position
	L	*Left justified*. The field is built up from the print position to the right.
	M	*Centered*. The field is built up from the print position to both sides
	R	*Right justified*. The field is built up from the print position to the left
z		Printout „normal“ or inverted
	P	Normal black printout
	A	“White printout”, that is the printing is left blank. Requires a dark background
	E	Printout with inverted bitimage (black is left blank; white is printed black)

#FF - Material feed

The #FF command triggers a material feeding by one label distance. The feeding length is up to the next gap, if any, or as long as the defined label length.

Syntax

```
#FF
```

The command must stand *outside* of the command sequence #ER to #Q!

The command does not work in standalone mode.

#FO - Importing an Easy Plug file

The #FO command is a placeholder of the therin specified file.

Syntax

```
#FO/f#G
```

Parameter	Value	Description
f	File name	Filename; must be according to the DOS name convention (drive, path, name, extension) File names are case sensitive. The file must already be copied to the printer RAM or the memory card.

How it works

By means of the #FO command, data traffic over the data interface can be reduced to variable data. The often more voluminous constant data can be stored in a file on memory card or on RAM-disk.

If the Easy-Plug interpreter hits a #FO command, it jumps to the specified file and interpretes the therein contained commands. After finishing this file, the rest of the initial command file is interpreted.

It does not work to store the constant data in an AUTOSTRT.FOR file and to send the variable data via data interface.

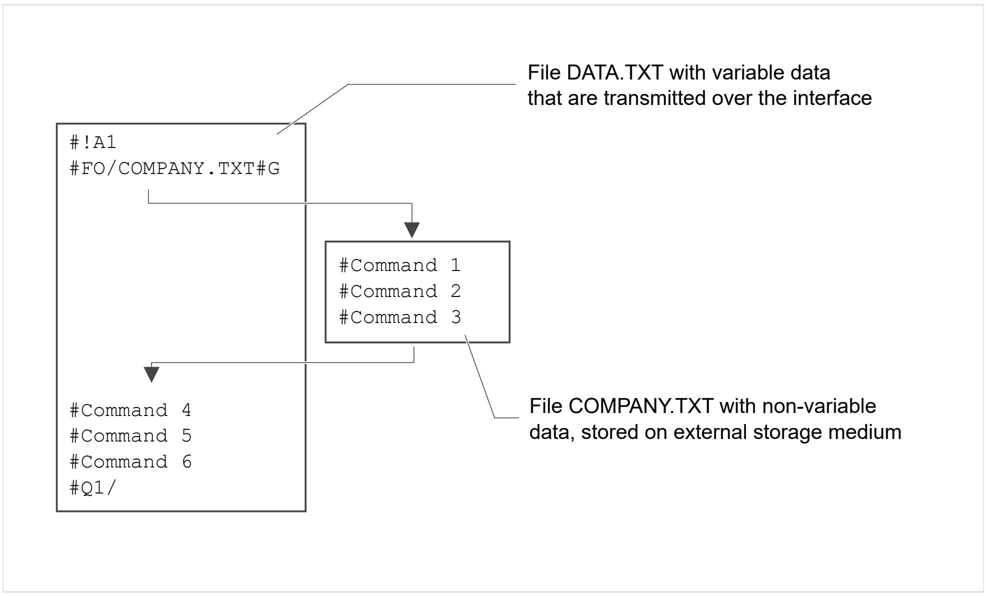

File DATA.TXT with variable data that are transmitted over the interface
#!A1
#FO/COMPANY.TXT#G
#Command 1
#Command 2
#Command 3
#Command 4
#Command 5
#Command 6
#Q1/
File COMPANY.TXT with non-variable data, stored on external storage medium

Fig. 4: Inserting commands from a file which is stored on a plug-in card into the initial format file.

Examples

Command:

```
#FO/C:\PFAD\...\VERZ_1\AFTER__8.FOR#G
```

Format file COMPANY.TXT on external memory medium:

```
#G------ Label size 50x70 mm ------------
#IMN70/50#G
-------------- Format command ----------------
#ERN0#G
#T5#J40#YT104/0///COMPANY
#T5#J30#YT104/0///located
#T5#J20#YT104/0///in GERMANY
#G---------- Feld 00 with 15 characters ----------
#T5#J20#YT104/0D///$00,15
#G------- Field 01 (bar code) with 12 characters ----
#T5#J02#YB1/0D/10/3///$01,12#G
```

File DATA.TXT is transmitted to the printer:

```
#!A1#G------------------------------------------ Activate printer
#FO/COMPANY.TXT#G---------------------------- Insert COMPANY.TXT
#YV00/Denmark#G------------------ Field 00 with content „Denmark“
#YV01/999333777001#G-------- Field 01 with content „999333777001“
#Q7/#G-------------------------------------------- Print amount 7
#YV00/USA#G-------------------------- Field 00 with content „USA“
#YV01/444197666001#G-------- Field 01 with content „444197666001“
#Q5/#G-------------------------------------------- Print amount 5
```

Related reference

File operations on page 139
Description of the correct path specification for file operations.

#DF - Downloading a file on page 28
The command #DF downloads a file from a PC to the printer.

#FW - Ribbon width

‖ Only applicable for machines of the types XPA 93x and XLP 60x. ‖

Defines the width of the ribbon material used for the print job.

Syntax

```
#FWn
```

‖ The command must stand *outside* of the command sequence #ER to #Q! ‖

Parameter	Value	Description
n	int	Ribbon width [30..107] mm

G

#G - End of command

The #G command can be used to close a preceding command or to mark a comment line.

Syntax

```
#G
```

Closing a preceding command

This command is used for closing a preceding command if the following applies:

- The last parameter of the preceding command was of type num, int or TEXT
- The preceding command is the last command of the format

#G should always be set for closing commands for

- Quantity details
- Downloading a logo

- Bar codes, for which the characters <20hex are permitted (e.g. Code 128, EAN 128)

Comment lines

Additionally, #G can mark a comment line in an Easy Plug file. A comment line, headed by #G may contain nearly every possible character combination.

Don´t start a comment line with slash or blank+slash:

- `#G/`
- `#G /`

You may use as many slashes as you want further behind: `#G L/ong live the Queen /////`

Example

Number of labels = 100 and the numerical value is closed with a blank command:

```
#Q100#G
```

H

#HP - print head pressure setting

> Only applicable for machines of the types XPA 93x and XLP 60x.

The #HP command sets the print head contact pressure to the value x.

Syntax

```
#HPx
```

> The command must stand *outside* of the command sequence #ER to #Q!

Parameter	Value	Description
x	num	Print head pressure [1.0..3.0] • 1.0 = low print head contact pressure (equals the setting „I“ of the adjustment knob at older machines) • 3.0 = high print head contact pressure (equals the setting „III“ of the adjust-ment knob at older machines)

Example

- The print head pressure setting is taken over from the setting in parameter Print > Head pressure:

```
#!A1
#IMN50/50
#ER
#T1#J1#YT109/0///HP Para#G
#Q2/
```

- The print head pressure setting is taken from the preceding HP command:

```
#HP2.1
#ER
#T1#J1#YT109/0///HP2.1#G
#Q2/
```

- The print head pressure setting is taken over from the setting in parameter Print > Head pressure:

```
#ER
#T1#J1#YT109/0///HP Para#G
#Q1/
```

Related reference

#!HP - print head pressure setting on page 12

#HV - Print head temperature setting

The #HV command sets the print head voltage and therefore the print head temperature to the required value n

Syntax

```
#HVn
```

The command must stand *outside* of the command sequence #ER to #Q!

CAUTION!

Hazard of reduced print head lifetime

The printhead temperature (= HV-value) directly affects the life of a printhead. It counts: “The higher the temperature, the printhead is driven with, the lower is its life durance”. This counts even more if HV-values above 100% are driven. Therefore note:

► Always choose the lowest possible HV-value necessary to produce an acceptable print result.

Parameter	Value	Description
n	int	Print head temperature [0...xxx] • 0 = lowest print head temperature • xxx = highest print head temperature (value depends on the printer type, see table below)

Printer	max. HV value
64-xx, DPM, PEM, ALX 92x, ALX 73x	110
XLP 504/506 300 dpi	120
XLP 504 600 dpi	100

Table 7: Settings to reach the highest possible print head temperature (max. HV value).

Related reference

#!H - Print Head temperature setting on page 13
The immediate command #!H sets the print head voltage and therefore the print head temperature to the required value n.

#IDM - Data Matrix Code

The #IDM command prints a 2-dim. „Data Matrix“ bar code.

Syntax

```
#IDMn/idgwrck/s/vop/a/TEXT#G
```

The command #IDM must be closed with #G.

Parameter	Value	Description
n	0	ASCII Encoding of ASCII data, double density numeric data and symbology control characters.
	1	C40 Encoding of ASCII data. Packs three alpha numerical data characters into two code words. Usage if data contains more upper case than lower case charac-ters.
	2	TEXT Encoding of ASCII data. Packs three alpha numerical data characters into two code words. Usage if data contains more lower case than upper case charac-ters.
	3	BASE256 Encoding of any 8 bit data. Set SYSTEM PARAMETER > Character filter resp. Printer Language > EasyPlug Setting > Character filter to “All characters“, if all characters from 0x00 to 0xff are supposed to be coded!
	4	reserved
	5	AUTO (default setting)
i	B	EAN/UCC mode with data designator put in parantheses. Data has to be sent in parantheses, but the parantheses will not be printed coded. Same handling as EAN 128 bar code.
	X	EAN/UCC mode with data designator not put in parantheses. Data has to be sent without parantheses. Same handling as EAN 128 bar code.
d	0	Bar code in normal write direction
	1	Bar code rotated by 90 degrees
	2	Bar code rotated by 180 degrees
	3	Bar code rotated by 270 degrees
g	D	Text field consists of one variable data field.
w	W	Counter field TEXT is incremented/decremented without a carry over, i.e. only the first unit position of the figure is increased/decreased.
r	Rn	n = Number of rows
c	Sn	n = Number of columns

		As a standard, the size (number of rows and columns) is calculated automatically in dependance of the input data (smallest possible matrix). Use the parameters r and c to set the size (see table below).
k	F	<FNC1> is used as data separator (--> Example)
	G	<GS> is used as data separator
		Only applicable for machines of type XPA 93x.

s	int	Number of printer dots for a Data Matrix Block [1...200]

v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]

a	int	No. of labels with constant No. [1...255].

TEXT		User data; can be any alphanumeric code with a maximum length of 1024 characters. The text field may contain an input field. The text field may also be a variable data field. Precondition: the D-flag must be set (see parameter „g“ above).

Rows	Columns	Rows	Columns
10	10	64	64
12	12	72	72
14	14	80	80
16	16	88	88
18	18	96	96
20	20	104	104
22	22	120	120
24	24	132	132
26	26	144	144
32	32	8	18
36	36	8	32
40	40	12	26

Rows	Columns		Rows	Columns
44	44		12	36
48	48		16	36
52	52		16	48

Table 8: Admissible combinations of rows (r) and columns (c). Example: #IDM/0R12S12/16/0/1/00000A89#G (data will be coded in a 12x12 matrix).

Optional control characters

‖ Not in EAN/UCC mode with BASE256 encoding! ‖

~X	Is used to represent character values from 0 to 26. Replace the X like in the following examples: • ~@ = 0 • ~A = 1 • ~B = 2 • ~C = 3
~1	Represents the character FNC1. If FNC1 appears in the first position (or in the fifth position of the first symbol of a Structured Append), it will indicate that the data conforms to the UCC/EAN Application Identifier standard format.
~2	Is used to represent Structured Append. Structured Append is used to link information from several symbols in a sequence. The ~2 must be followed by 3 additional bytes. The first 4 bits of the first byte identify the position of the particular symbol in the sequence. The last 4 bits identify the total number of symbols in the sequence. The second and third bytes are used as a file identifier and can have a value between 1 and 254 (up to 254*254=64516 identifiers). See Data Matrix Specification for more information about this (ISO 16022).
~3	Is only allowed in the first position of the symbol. It indicates that the data contains commands for the barcode reader.
~4	Not admissible
~5/~6	Are only allowed in the first position of the symbol. If ~5 is used the header [)> ASCII30 ASCII05 ASCII29 will be transmitted by the barcode reader before the data in the symbol and the trailer ASCII30 ASCII04 will be transmitted after the data. If a ~6 is used, the header [)> ASCII30 ASCII05 ASCII29 will be transmitted by the reader before the data and the trailer ASCII30 ASCII04 will be transmitted afterwards: `[)>ASCII30 ASCII05 ASCII 29 Daten ASCII30 ACII04`
~7NNNNNN	Specifies the Extended Channel to be used, where NNNNNN is a value between and 000000 - 999999. *Example*: `~7000010` means Extended Channel 10. Extended channel is used for using other character sets than ASCII.
~dNNN	Represents the ASCII character encoded by the 3 digits NNN. *Example*: `~d065` represents the character „A“.

Example: Different data separator characters

```
#IDM0/B0/12///
<FNC1>(01)08711744025670(17)181231(10)99999.E7L0185<FNC1>(21)00000D0A#G
```

```
--> <FNC1> is used as data separator (indicated by <FNC1> within barcode
 data)
```

```
#IDM0/B0G/12///
<FNC1>(01)08711744025670(17)181231(10)99999.E7L0185<GS>(21)00000D0A#G
--> <GS> is used as data separator (indicated by <GS> within barcode da►
ta)
```

The first <FNC1> is always a “FNC1” code as it defines that the subsequent data is coded regarding GS1 standard.

Related reference

Input Fields on page 10
Variable Data Fields on page 9
Printer-internal bar codes on page 144

#IM - Material information

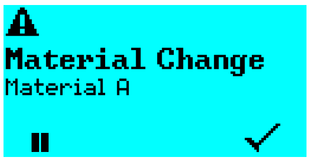
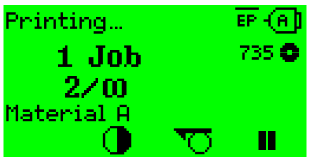
The #IM command defines the necessary material information and an (optional) material designation

Syntax

```
#IMxyb/l/c/tg/d/ef/h
```

The command must stand *outside* of the command sequence #ER to #Q!

If a print job does not contain any #IM-command, the material information defined in the previous print job will be used.

Parameter	Value	Description
x	N	For endless material without gaps
	S	For material with gaps
y	B	For *Batch-Mode*: Total surface of the label is printable; high output volume.
	E	For *Normal 1:1-Mode*: The initial zones of the label are not printable. All labels in a print job are cut (if programmed); high output volume.
	R	For *Real 1:1-Mode*: Total surface of labels is printable. All labels are issued (if programmed); reduced output volume.
b	num	Material width: [000.00...max. width] mm, depending on print head type.
l	num	Label length: [000.00...max. length] mm, depending on the configuration of the printer.. In the strict sense, this parameter determines the label pitch, that is the distance from the leading edge of the label to the leading edge of the next label. Find the maximum label length of the concerned printer on Info-Printout „Memory Status“ or under Info > System > Memory Data > Max. Labellength.
c	TEXT	Material designation with 16 alphanumerical characters. The character # must not be used. If the new material designation differs from the previous one, the printer shows the new designation on the display to inform the user that a material change may be necessary. The display must be acknowledged by the user by pressing the button.  Material Change Material A Fig. 5: Request to change material on an XPA 93x (here: c=“Material A”).  Printing... 1 Job 2/∞ Material A Fig. 6: The material designation “Material A” is also displayed during printing. Exception: If the similar message “Start next job” has been activated (System > Print Control > Single-job mode = “On”), only this is displayed, so there is no additional prompt to change material.

t	int	Tag type 0 = autodetect (default) ≠ 0 (compare table below „Supported transponder types“.)
g	C	Checks tags that are manufactured by „Alien Technologies“ for validity (read chapter below). For the purpose of eliminating these tags we check for A5A5 and if there is no A5A5 we invalidate these tags (cross out).
	D	(Default setting) Detect whether there is a tag to operate on. Applies to tags without UID/TID. Detected identification is treated internally as UID/TID. Do not use this option if you would like to operate on zero initialized EPC tags - they cannot be detected.
	N	None.

d	0	No RFID-operations.
	int	(d>0) Distance in print direction (x) from label edge to optimum of transponder antenna. It is recommended to set parameter y to R when parameter d is used. This parameter is mandatory for any RFID processing (even if distance is zero).

e	int	Number of retries with different tags in case of detect errors. Default defined by parameter RFID PARAMETER > Max Tag Stop. Setting range: [0...10].
f	V	Verify option (default: don't' verify) - Verify written data with an extra read after write and compare it. If written data and read data is different an error is reported.

h	int	Distance to middle of chip from label edge, to activate head lift for chip protection. With h option we define an explicit range of head lifting (see fig. below). With this setting, print data in the chip area is not printed.

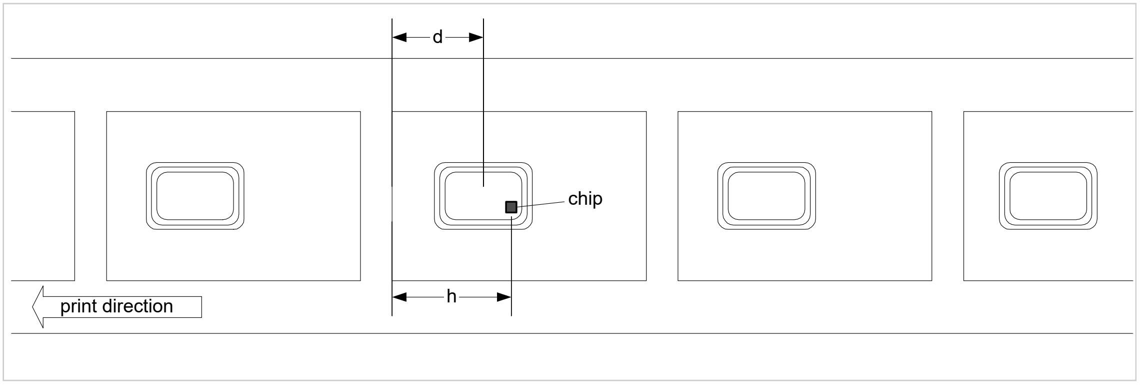


Fig. 7: Schematic figure of the parameters respectively the measures d and h.

ID	Manufacturer	HF/UHF	Label type
0	ISO 15963	HF	Autodetect ISO 15963 compliant
1	Infineon	HF	My-d
2	Philips	HF	NXP I-Code 1
3	Philips	HF	NXP I-Code SLI

ID	Manufacturer	HF/UHF	Label type
4	Tag-it	HF	HF ISO
7	Philips	HF	NXP I-Code EPC
8	Philips	HF	NXP I-Code UID
15		UHF	EPC Class 1 Gen 2 / ISO 18000-6C

Table 9: Supported transponder types.

Examples

Command	Description material type
`#IMS50/100/TYP1`	Gapped material, 50 mm wide, 100 mm long, TYPE 1
`#IMN20/50/`	Endless material, 20 mm wide, 50 mm long
`#IMSR98/165///70`	Gapped material, 98 mm wide, 165 mm long, with RFID transponder; 70 mm distance from label edge to middle of transponder antenna, autodetect transponder type
`#IMSR98/165//6/70/`	Gapped material, 98 mm wide, 165 mm long, with RFID transponder; 70 mm distance from label edge to middle of transponder antenna, transponder type “EPC class 1”

Transponder by Alien Technologies

On those tags there is an EPC written by the manufacturer. If this EPC begins with Hex A5A5 it is a "good" one (good performance/good quality). If it does not start with A5A5 it means it is NOT a "good" one. But anyway you may be able to write and read these tags.

J

#J - Vertical print position

The #J command determines the vertical print position with absolute value in mm. Zero position is always the bottom, left-hand corner of the label. Exception: command #R is active.

Syntax

```
#Jx
```

The command must stand *between* #ER and #Q!

Parameter	Value	Description
x	num	Vertical print position in mm in relation to the bottom left-hand corner of the label (000.00 mm) Text outside of the defined label area is not printed. Reference line of the text is the base line. A base line, which exceeds the defined label area, causes an error message, even if there are no descenders in the text.

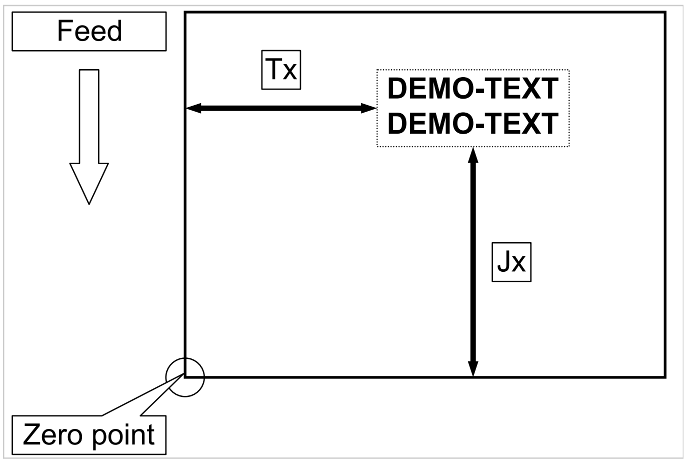


Fig. 8: Vertical (Jx) and horizontal (Tx) print position on the label.

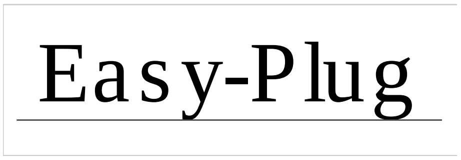


Fig. 9: Base line of a text. Even if the text has no descenders (as the y and the g in the example text), the base line lies a certain distance below the text.

Example

Text begins 5 mm from the bottom:

```
#J5
```

Text begins 20 mm from the bottom:

```
#J20
```

M

#M - Magnification factor

The #M command magnifies characters and logos in the X/Y axis by the defined factor.

Syntax

```
#Mx/y
```

The command must stand *between* #ER and #Q!

All characters which follow are magnified by the defined factor until #M1/1has been sent, the printer reset or a new label series started..

Parameter	Value	Description
x	int	Magnification of the font width (run direction) [1...16]
y	int	Magnification of the font height [1...16]

Examples

Characters are printed normally:

```
#M1/1
```

Characters are magnified by a factor of 2 in the X axis and by a factor of 4 in the Y axis:

```
#M2/4
```

Characters are magnified by a factor of 4 in the X axis and are printed normally in the Y axis:

```
#M4/1
```

#ME - Eject material

The #ME command triggers a cut and backwards feeding of the printer in order to eject the label material.

Syntax

```
#ME
```

se this command only for printers equipped with a cutter („Cutter 2000“).

The command must stand *outside* of the command sequence #ER to #Q!

The command does not work in standalone mode.

#MXC - Bar code Maxicode

The #MXC command prints the 2-dimensional "MaxiCode" bar code.

Syntax

```
#MXCz/dw/x/y/vop/a/TEXT#G
```

‖ The command must be closed with #G. ‖

Parameter	Value	Description
z	int	MaxiCode modes (2, 3, 4 and 6)
d	0	Normal write direction
	1	Bar code rotated by 90 degrees
	2	Bar code rotated by 180 degrees
	3	Bar code rotated by 270 degrees
w	W	Counter field TEXT is incremented/decremented without a carry over, i.e. only the first unit position of the figure is increased/decreased.
x	int	Barcode number for code-splitting onto more than one bar codes [1...8] (default setting: 1)
y	int	Amount of barcodes in case of code-splitting onto more than one bar codes [1...8] (default setting: 1)
v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]
a	int	No. of labels with constant No. [1...255].

TEXT		Depending on the mode (parameter „z“), contains TEXT the following information (see table below): ‖ The text field may contain an input field. ‖

z	TEXT (Syntax)	Data
2	`ppppppppp ccc sss MSG`	p: Numerical postcode (9 characters) c: ISO national code (3 characters) s: Service class (3 characters) MSG [2]: Code words (84 characters)
3	`pppppp ccc sss MSG`	p: Numerical postcode (6 characters) c: ISO national code (3 characters) s: Service class (3 characters) MSG [2]: Code words (84 characters)
‖ In Modes 2 and 3 a space must be left respectively between the postcode, national code, service class and user data. The number of characters in the postcode, national code and service class must be observed exactly. ‖		
4	`MSG`	MSG: Any alphanumerical code (93 characters)
6	`MSG`	

Table 10: Structure and length of the data string „TEXT“ depending on the mode „z“.

Simple example

The string “MSG” contains only alphanumerical characters.

```
#!A1
#IMS100/200

#ER

#T40#J10
#MXC4/0/1/1///NOVEXX Solutions Teststring#G
#Q1#G

#!P1
```

Fig. 10: Printout of the example.

2 Depending on the selection of alphanumerical characters (0x00 – 0xff) results a different amount of user data, because more or less switches between subsets are necessary.

Example with characters < 20h

‖ Important: Set parameter Character filter to "All characters"! ‖

The string "MSG" can also contain non-representable characters, i.e. characters with a hexadecimal code <20h. Such characters are, for example, the control characters <RS> (0x20) and <GS> (0x1E). Special editors make this invisible code visible by using placeholders:

```
#!A1
#IMS100/200

#ER
#T40#J10
#MXC4/0/1/1///[)>RS01GS96123GS840GS111GS1234567890GSUPSNGS123456GS222GS1GS1/
2GS10GSYGSTestGSTest CGSTeRSEOT#G
#Q1#G

#!P1
```

For such user data, it is recommended to use the alternative command #SMX, in which this data can be visibly displayed.

Fig. 11: Printout of the example.

Related reference

Input Fields on page 10

Printer-internal bar codes on page 144

#SMX - Maxicode definition on page 84

The #SMX command defines a "Maxicode" bar code. Printing of the bar code requires a subsequent #VW-command.

NOVEXX SOLUTIONS

N

#N - National character set

The #N command changes the fonts between different national character sets.

Syntax

```
#Nn
```

The command must stand *outside* of the command sequence #ER to #Q!

Parameter	Value	Description
n	0	USA
	1	UK
	2	France
	3	Germany
	4	Italy
	5	Sweden
	6	Spain
	7	Norway
	8	Spezial
	9	IBM-similar
	10	ANSI Codepage 1252 Latin 1 (equals ISO 8859-1 Latin 1)
	11	ANSI Codepage 1250 Central Europe
	12	ISO 8859-2 Latin 2
	13	UTF-8

Example

Printing of russian text with UTF- 8 coding:

```
#G -------------------------------------------------------------------
#G Prerequisite: The cyrillic font is stored on a memory card in folder
#G \fonts. Filename: „font900.xxx“.
#G -------------------------------------------------------------------
#!A1
#IM200/100
#N13
#ER
#SS900/OV/32/#G
#T01.0#J010.0
#VW/L/“Указания по тезнике безопасности при эксплуатации машины для
 печатания этикеток“
#Q1/
```

O

#OLVI - Initialize Online Verifier

Send an initialization string to the online verifier (OLV). Only valid for the OLV type RJS SV100.

Syntax

```
#OLVIn/INITTEXT
```

The command must stand *outside* of the command sequence #ER to #Q!

Parameter	Value	Description
n	0	INITTEXT will only then be sent to the OLV, if the content of the string has changed.
	1	INITTEXT is always sent to the OLV.
INITTEXT		SV100 commands, which are sent to the OLV before a print job is started. For admissible initialization commands refer to the SV100 manual.

#OLVD - Define OLV limits

The #OLVD command defines the limits for the separate readability criteria. Is only valid for RJS SV100 OLV

Syntax

```
#OLVD/a/b/c/d/e/f/g/h/i/j/k/u
```

The command must stand *between* #ER and #Q!

For every bar code may be defined a separate limit.

The measured value must in all cases be above or equal the set limit.

Parameter	Value	Description
a	P	Decodable (Pass)
	F	Not decodable (Fail)
b	int	Decodability [0...100]
c	int	Modulation [0...100]
d	int	Defects [0...100]
e	int	Edge contrast [0...100]
f	int	Rmin/Rmax [0...100]
g	int	Symbol contrast [0...100]

h	int	PCS [0...100]
i	int	R (white) [0...100]
j	int	R (black) [0...100]
k	int	Ratio [0...99]
u	int	ANSI symbol grade [0...40]

Example

Only „Decodability“ and „Defects“ is analysed:

```
#OLVD//45//20/////////
```

P

#PA - Offset print start

The #PA command offsets the beginning of the print range by the given value.

‖ Select ribbon saving mode to lift the print head while the material is being fed to the print range! ‖

Syntax

```
#PAa/b
```

‖ The command must be sent before the #IM command! ‖

Parameter	Value	Description
a	num	Offset in mm from the physical start of the label until the print start
b	num	Length of the range to be printed.

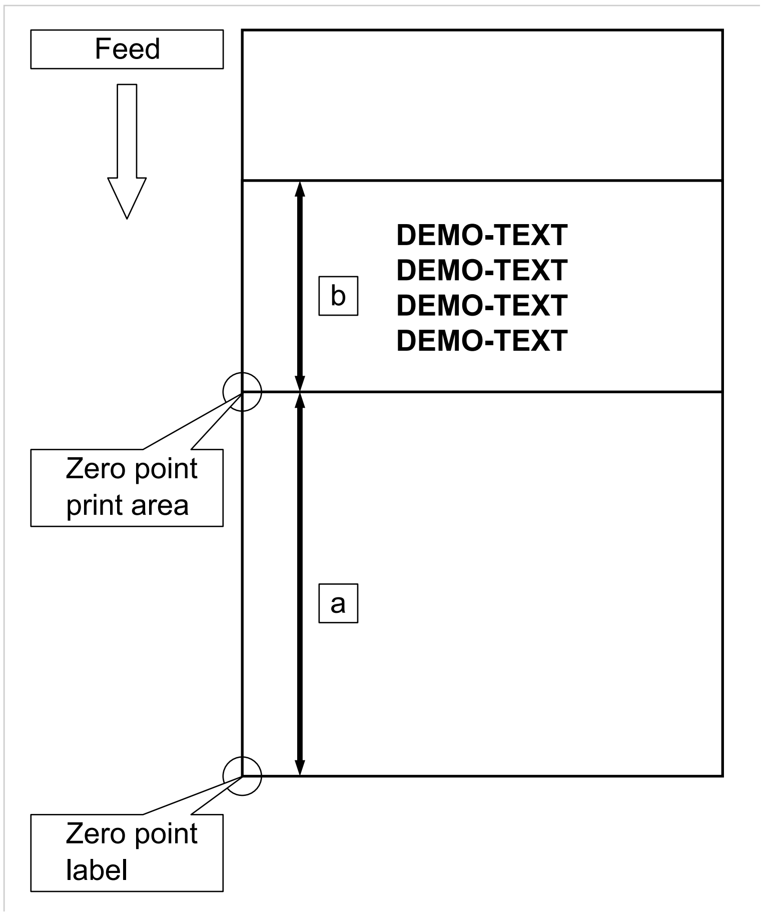


Fig. 12: Offsetting the zero point of the print area with the length „b“ by the distance „a“.

Example

Length of the print area 100 mm, offset 400 mm from where the label starts:

```
#PA400/100
```

Label width 100 mm, total length 600 mm:

```
#IMSB/100/600
```

#PC - Setting Parameter Values

The #PC command sets parameters in the printer menu to certain values.
The function is helpful, if…

- settings must be transferred from one printer to another.
- several printers are supposed to have the same settings..
- the settings of a printer are supposed to be recovered, e.g. after the CPU board was replaced.

It is adviseable to read out the parameter settings of the concerned device at first. This can be done one of the following ways:

- With the Easy-Plug command #!PG

- By aid of parameter SPECIAL FUNCTION > Store Parameters or Tools > Diagnostic > Store Parameters

Syntax

```
#PC<ParameterID>/<Value>#G
```

Many of the parameters need a system reset, before the changed setting becomes effective. This can be triggered at the printers operation panel or by an Easy Plug command.

Parameter	Value	Description
<ParameterID>		*Parameter Ident number* (ID) 999999 = Parameter-ID for special functions; must be applied in combination with one of the below listed values (see table). Each parameter in the menu has its own Parameter-ID. Finding out a Parameter-ID: Call SPECIAL FUNCTION > Store Parameters or Tools > Diagnostic > Store Parameters. The so generated text file (Setup file) contains the assignment Parameter-ID/Parameter. Example setup files can be found in the service manual of the printer in topic section „Advanced Applications“ (64-xx) or in the Appendix (XLP 50x).
<Wert>		*Parameter Value* The admissible values depend on the type of parameter. Assigned values can be integer numbers, floating point numbers or strings. The parameter values for the parameters IP address, Net mask and Gateway address can be specified in the format xxx.xxx.xxx.xxx or as plain integer.

Value	Function
-1	Reset the machine
1	Store the current parameter settings as customer default (equates to SPECIAL FUNCTION > Custom defaults = „Apply current“ or System > Custom defaults = „Apply current“)
2	Delete the stored customer default values (equates to SPECIAL FUNCTION > Custom defaults = „Delete“ or System > Custom defaults = „Delete“)
3	Restore factory settings with „custom defaults“ if available (equates to SPECIAL FUNCTION > Factory settings = „Custom defaults“ or System > Factory settings = „Custom defaults“), if not use „factory defaults“ (equates to SPECIAL FUNCTION > Factory settings = „Factory defaults“ or System > Factory settings = „Factory defaults“)

Table 11: Admissible values for special functions – must be applied together with Parameter-ID „999999“.

Example

Command sequence	Description
`#PC1508/viper#G`	Sets the FTP Server password to „viper“ (equates to INTERFACE PARA > >NETZWERK PARAM. > FTP Password = „viper“ or. Interface > Network > Services > FTP Password = „viper“)
`#PC999999/-1#G`	The printer will be started newly (reset)

#PDF - Bar code PDF 417

The #PDF command prints a bar code of the type PDF 417.

Syntax

```
#PDFn/td/s/l/z/w/h/TEXT#G
```

Parameter	Value	Description
n		Compression
	0	EXC-Mode (Extended alphanumeric Compaction Mode)
	1	Binary ASCII Plus Mode
t	leer	Bi-directional reading (default setting)
	T	Unidirectional reading
d	0	Bar code normal write direction
	1	Bar code rotated by 90 degrees
	2	Bar code rotated by 180 degrees
	3	Bar code rotated by 270 degrees
s	int	Security Level [0...8]
l	0	Bar code width is set automatically
	int	Bar code width in code words (number of columns) [1...30]: The width should be so selected that a height of 90 lines is not exceeded with regards to the area of the bar code defined in the user data. A maximum of 928 code words are permitted for one bar code. Of these the following are preassigned: • 1 word for the length input of the code • x words for the check total (Security Level) = 2 (1 + Security Level)
z	0	Number of lines is set automatically
	int	Number of lines [3...90]
w	int	Bar code width [1...16]
h	num	Height of a PDF bar code row in millimetres [1...100] mm
TEXT		User data: Permitted characters depending on the compression type. Max. string length: 1024 characters. ‖ The text field may contain an input field.. ‖

Related reference

Input Fields on page 10

Printer-internal bar codes on page 144

#PO - Gap offset

The #PO command is for determining the beginning of the label when irregularly formed labels are being printed.

Syntax

```
#POva
```

‖ The command must stand *outside* of the command sequence #ER to #Q! ‖

Parameter	Value	Description
v	+	Positive offset: label begins before the end of the gap
	-	Negative offset: label begins after the end of the gap
a	num	Offset in mm ‖ Mind the limitations of the concerned printer, see User Manual.. ‖

Examples

Label begins at the end of the gap:

```
#PO
```

Beginning of the label 5 mm after the end of the gap:

```
#PO+5
```

Beginning of the label 20 mm before the end of the gap:

```
#PO-20
```

#PR - Print speed

The #PR command sets the print speed and feed speed.

Syntax

```
#PRx/y/
```

Parameter	Value	Description
x	int	Print speed; The setting range depends on the printer type; setting interval: 1 inch/s
y	int	Feed speed; The setting range depends on the printer type; setting interval: 1 inch/s
‖ With DPM/ ALX 92x, decimal places may be programmed. The setting interval then is 0.2 Inch (e.g. #PR 8.6/8.6). ‖		

Examples

Both speeds = 8 inch/s (Default setting):

```
#PR
```

Print speed = 4 inch/s, Feed speed = Previously set value or, if no value has been set, default setting.:

```
#PR4/
```

Print speed = Previously set value or, if no value has been set, default setting. Feed speed = 6 inch/s:

```
#PR/6/
```

Print speed = 8 inch/s Feed speed = 10 inch/s:

```
#PR8/10/
```

Q

#Q - Print quantity

The #Q command terminates the print job and defines the number of labels to be printed. The print job is checked and stored in the memory.

The format cannot be altered or extended once it has been stored in the memory. Format commands are not evaluated until the memory is empty again. The memory is empty when the label series has been printed or broken off with #CF.

Syntax

```
#Qan/
```

Parameter	Value	Description
a	A	Standalone mode: The label amount is queried.
	X	Standalone mode: The label amount is *not* queried, but amount “n” is printed.
n	leer	Memory is emptied. Nothing is printed. In standalone mode is the label amount endless.
	0	
	int	Number of labels to be printed [1...2,2 Mrd]
	*	Label amount = endless
/		The command #Q must be closed with a slash (/) or #G.

Example

100 labels of the previous format are printed:

```
#Q100/
```

Special case: Multitrack printing with incomplete label rows

If the print job provides less labes than are necessary to fill the last label row, the following happens:

- Print job sent via data interface: The row is printed incomplete, starting the printout with the label at the material zero line side.
- Print job started in standalone mode: The row is filled automatically and is printed completely.

R

#R - X/Y Offset data blocks

The #R command offsets all subsequent data blocks in direction X/Y.
Logically or optically-related printouts (data blocks) can be shifted as a whole.

The reference point #Tx or #Jx (see #T and #J) for all subsequent positionings is shifted from the zero point by the entered x/y value.

Zero point is the bottom, left-hand corner of the label (see illustration), provided that this has not been shifted by the #PO command.

Syntax

```
#Rvx/vy
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Bedeutung
v	+	Offset in positive x
	-	Offset in negative x
x	num	Horizontal offset of the print position
v	+	Offset in positive y
	-	Offset in negative y
y	num	Vertical offset of the print position

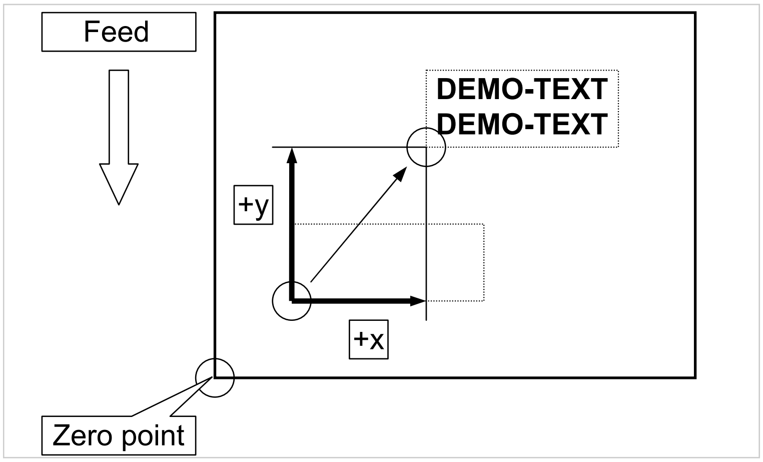


Fig. 13: Shifting a text block in x/y direction.

Example

Block is shifted to the right and upwards by resp. 5 mm:

```
#R5/5
```

Block is shifted to the left and downwards by resp. 5 mm:

```
#R-5/-5
```

#RFC - Special RFID commands

The #RFC command sends a command sequence to the RFID reader module to trigger special operations which are not mapped / part of the regular read/write/lock operations.

Syntax

```
#RFC<cmd>#G
```

The command must stand *between* #ER and #Q!

The commands are specific for a RFID reader module that must be installed and activated.

Parameter	Value	Description
<cmd>	int	Command code. ---------------- HF-Technology / FEIG reader --------- (*1) EAS features (only supported by „NXP I-Code“ chips) 16: Set EAS bit 17: Reset/clear EAS bit 18: Lock EAS bit permanently

#RFH - Request data - send to host

The #RFH command requests data of a RFID variable that has been read and assigned with #RFR command. That data is transmitted as response over selected EASYPLUG interface (same interface that is used for this request).

Syntax

```
#RFHi/m/n/HEAD
```

Only for printers with installed and activated RFID option.

The command must stand *between* #ER and #Q!

Parameter	Value	Description
i	int	Number of variable that has been created by read data command #RFR.
m	int	Number of bytes in response, filled if necessary with white space (0x20).
n	int	Number of bytes in HEAD
HEAD		Characters that precede response.

Example

Data on transponder (block size 4 assumed e.g. I-Code) shown in hex notation:

	MSB............LSB
Block address 3:	41 42 43 44
Block address 4:	33 31 32 33

Command sequence	Description
`#RFR1/0/I/L/2/3`	2 blocks from transponder starting at block 3 will be read and put into variable with id number 1. Text in variable 1: DCBA3210
`#RFH1/14/6/BLOCK:`	Request var 1 Response: BLOCK:DCBA3210

`#RFR3/2/B`
`#RFH3/20/4/EPC=`
	EPC in variable 3 Request var 3 Response: EPC=A5A5800F35609854

`#RFR3/1/B`
`#RFH3/20/4/UID=`
	UID in variable 3 Request var 3 Response: UID=E005000000000C5B

#RFL - Lock/unlock memory areas

The #RFL command enables or disables the write protection (simple lock) of various memory areas of EPC Gen 2 tags.
Before changing the write protection status for the first time, the access password must once have been written into the access password memory of the tag by means of a write command (#RFW or #SRF + #VW/T). For any attempt to change to protection status the same password must be provided as part of the #RFL command.

The #RFL command itself doesn’t write the access password into to tag!

The #RFL command doesn’t support the permalock option specified in the EPC Gen 2 standard. It implements only the write protection which can be reversed any time by issuing an unlock command with the right access password.

Syntax

```
#RFLa/b/c/TEXT#G
```

Only for printers with installed and activated RFID option.

The command must stand *between* #ER and #Q!

Parameter	Value	Description
a		Operation:
	1	Lock (write protect) memory area
	0	Unlock (unprotected) memory area
b		Memory area selection:
	0	User memory
	2	EPC
	3	Kill password

	4	Access password

c	B	Hex ASCII encoding: for the access password 8 characters must be provided.
	I	Raw data (binary data): for the access password 4 chars must be provided.

TEXT		AccessPassword: EPC Gen 2 specifies the access password with a size of 32 bit (4 bytes). Depending on the data encoding (option c), 4 or 8 characters must be provided.

#RFR - Read data

The #RFR command reads data of a RFID transponder and assigns it to a RFID variable. Content of variable can be requested with #!RF command.

Syntax

```
#RFRi/t/ab/s/n
```

Only for printers with installed and activated RFID option.

The command must stand *between* #ER and #Q!

Parameter	Value	Description
i	int	Number of variable that is created for read data.
t	0	BLOCK Parameter „s“ and „n“ required
	1	UID/TID Parameter “n” optional – for limiting size of data returned for read operations
	2	EPC
	3	KILL password
	4	ACCESS password
	5	MEM BANK
	6	AFI
	7	MEMBLOCK Parameter „s“ and „n“ required
	8	DSFID
a	I	ASCII data assumed on transponder (default) Each byte is coded as ASCII character.
	B	Binary data assumed: Each byte is coded as hexadecimal number in ASCII presentation.
b	L	Least significant byte of block first (default) : Bytes of each block on transponder are read beginning with the LSB first.
	M	Most significant byte of block first: Bytes of each block on transponder are read beginning with the MSB first.

s	int	If Data type = • BLOCK: Start block address • MEM BANK: Address/name of memory bank; (default: 0) • MEMBLOCK: Combination of memory bank identifier and address offset in the memory bank. Schema for address calculation:: `s = <EPC MemoryBankID> * 1000 + <BlockNumber>` EPC MemoryBankID: – 0: reserved (default setting) – 1: EPC – 2: TID – 3: USER – Other values are ignored • Other data type: parameter is ignored

n	int	If... • Datentyp = BLOCK, TID, MEM BANK oder MEMBLOCK: number of blocks to be read (default: 0). • Other data type: parameter is ignored For reading a TID field (t=1), the size parameter can be used to limit the size of the returned data. Giving no size information (n=0) will return the whole content of the TID memory bank, which may contain more information than expected (depending of RFID transponder/chip). Typical values for n are: • 2: Read 32Bit Vendor-ID/Chip-ID • 4: Read 32Bit Vendor-ID/Chip-ID + 32Bit Unique Serial Number

Example

Data on transponder (block size 4 assumed e.g. I-Code) shown in hex notation:

	MSB............LSB
Block address 3:	41 42 43 44
Block address 4:	30 31 32 33

Command sequence	Description
`#RFR1/0/I/L/2/3`	2 blocks from transponder starting at block 3 will be read and put into variable with id number 1. Text in variable 1: DCBA3210
`#RFR1//B/3/2`	Now binary data Text in variable 1: 4443424133323130
`#RFR1//M/3/2`	Now Ascii data reversed Text in variable 1: ABCD0123
`#RFR3/2/B`	EPC in variable 3
`#RFR3/1/B`	UID in variable 3

#RFW - Write typed data to RF tag

The #RFW command writes data of well-defined type onto a RFID label.

Syntax

```
#RFWt/abs/r///n/DATA#G
```

Only for printers with installed and activated RFID option.

The command must stand *between* #ER and #Q!

Parameter	Value	Description
t		Data type
	0	BLOCK
	1	UID/TID
	2	EPC
	3	KILL password
	4	ACCESS password

a	I	Data assumed in DATA as ASCII(default) Each byte in DATA is written to chip in identical form. There is no interpretation. This is especially designed for printable byte codes as ASCII characters.
	B	Data assumed in DATA in hex representation: Two consecutive bytes in DATA are interpreted as hexadecimal description for a byte code. This byte code is written on chip. This is especially designed for NON printable byte codes.
b	L	Least significant byte of block first (default) : Bytes of DATA are written on transponder beginning with the LSB first.
	M	Most significant byte of block first: Bytes of DATA are written on transponder beginning with the MSB first.
s	int	Address of start block. Default setting: 0 (= Block no.)

r	int	Number of read/write retries. Default defined by parameter RFID PARAMETER > Nr of CMD retries.

n	int	Number of characters in DATA.

DATA		Data to be written. (Data type BLOCK only) n must be a multiple of the block size, otherwise padding with 0 to a multiple of the block size.

Example

Write 12 bytes as an EPC 96 in hex representation:

```
#RFW2/B////24/01203D2A916E8B8719BAE03C
```

#RSS - GS1 DataBar & CC

The #RSS command prints a bar code field with a GS1 DataBar or a composite symbology bar code.

Syntax

```
#RSSzx/dw/s/vop/a/TEXT#G
```

The command must be terminated with #G!

Parameter	Value	Description
z	int	Bar code number [3] GS1 DataBar Expanded Stacked is applied automatically, if the following conditions are met: • Selected barcode type: GS1 DataBar Expanded (z = 6) • Width (x) is set to less than 22 symbol characters per row • Primary data exceeds the set width
x	Sn	n = [4...22] Symbol characters per row
d	0	Normal writing direction
	1	Bar code rotated by 90°
	2	Bar code rotated by 180°
	3	Bar code rotated by 270°
w	W	Counter field TEXT is incremented/decremented without a carry over, i.e. only the first unit position of the figure is increased/decreased.
s	int	Number of printer dots for the bar code module width.
v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]
a	int	No. of labels with constant No. [1...255].
TEXT		Primary data and optional secondary data (separated by a „|“); alphanumerical code according to the selected bar code type, up to a maximum of 1024 characters long.

Examples

- GS1 DataBar Omnidirectional

[3] See below, chapter „Additional Information“.

Data Output: 0109501101420021

```
#RSS1/0/3///0950110142002
```

- GS1 DataBar Stacked Omnidirectional

  Data Output: 0109501101420038

```
#RSS4/0/3///0950110142003
```

- GS1 DataBar Omnidirectional (POS and General Distribution)

  Data Output: 0109501101420045

```
#RSS1/0/6///0950110142004
```

- GS1 DataBar Expanded

  Data Output: 01095011014200522112345678

```
#RSS6/0/3///01095011014200522112345678
```

- GS1 DataBar Expanded Stacked

  Data Output: 1095011014200693922995<GS>3202000100 17100101422123<GS>2112345678

```
#RSS6S6/0/3///01095011014200693922995<FNC1>320200010017100101422123<FNC1>2112345
```

‖ CAUTION! No <cr><lf> in Easy Plug sequence or output. ‖

- GS1 DataBar Truncated

  Data Output: 0109501101420076

  ```
  #RSS2/0/3///0950110142007
  ```

- GS1 DataBar Limited

  Data Output: 0109501101420083

  ```
  #RSS5/0/3///0950110142008
  ```

- GS1 DataBar Stacked

  Data Output: 0109501101420090

  ```
  #RSS3/0/3///0950110142009
  ```

Related reference

Printer-internal bar codes on page 144

#RT - Read and print RFID data

The #RT command reads data of a RFID transponder and prints it on its label.

Syntax

```
#RTz/dbjk/t/ab/s/n
```

> Only for printers with installed and activated RFID option.
>
> The command must stand *between* #ER and #Q!

Parameter	Value	Description
z	int	Character set number [100...116] (for details see link below under „Additional Information“)

d	0	Bar code in normal write direction
	1	Bar code rotated by 90 degrees
	2	Bar code rotated by 180 degrees
	3	Bar code rotated by 270 degrees
b	A	Text printed inversely (white characters on black background) ‖ A dark background must first be applied to the print area (line or rectangle). ‖
j	L	Flush left (default): The write command (#T / #J) refers to the left edge of the text field. The field is built up to the right.
	M	Middle: The write command (#T / #J) refers to the centre of the text field. The field is built up on both sides of the print position.
	R	Flush right:: The write command (#T / #J) refers to the right edge of the text field. The field is built up to the left.
k	Snum	Fixed distance in mm between all characters in the read data string. The character S has to be succeeded without blank by the value.

t	0	BLOCK ‖ Parameters „s“ and „n“ required. ‖
	1	UID/TID ‖ Parameter “n” optional – for limiting size of data returned for read operations. ‖
	2	EPC
	3	KILL password
	4	ACCESS password
	5	MEM BANK
	6	AFI
	7	MEMBLOCK ‖ Parameters „s“ and „n“ required. ‖
	8	DSFID

a	I	ASCII data assumed on transponder (default) Each byte on chip is printed as ASCII character on the label.
	B	Binary data assumed: Each byte chip is printed as hexadecimal number in ASCII presentation on the label.
b	L	Least significant byte of block first (default) : Bytes of each block on transponder are read beginning with the LSB first.
	M	Most significant byte of block first: Bytes of each block on transponder are read beginning with the MSB first.

s	int	If Data type = • BLOCK: Start block address • MEM BANK: Address/name of memory bank; (default: 0) • MEMBLOCK: Combination of memory bank identifier and address offset in the memory bank. Schema for address calculation: `s = <EPC MemoryBankID> * 1000 + <BlockNumber>` EPC MemoryBankID: – 0: reserved (default setting) – 1: EPC – 2: TID – 3: USER – Other values are ignored • Other data type: parameter is ignored

n	int	If... • Data type = BLOCK, TID, MEM BANK or MEMBLOCK: number of blocks to be read (default: 0). • Other data type: parameter is ignored For reading a TID field (t=1), the size parameter can be used to limit the size of the returned data. Giving no size information (n=0) will return the whole content of the TID memory bank, which may contain more information than expected (depending of RFID transponder/chip). Typical values for n are: • 2: Read 32Bit Vendor-ID/Chip-ID • 4: Read 32Bit Vendor-ID/Chip-ID + 32Bit Unique Serial Number

Example

Data on transponder (block size 4 assumed e.g. I-Code) shown in hex notation:

	MSB............LSB
Block address 3:	41 42 43 44
Block address 4:	30 31 32 33

Command sequence	Description
`#RT101////3/2`	2 blocks from transponder starting at block 3 will be read and printed with font 101 in normal (from left to right) print direction. Text printed on label: DCBA3210
`#RT101///B/3/2`	Now binary data Text printed on label: 443424133323130
`#RT101///M/3/2`	Now Ascii data reversed Text printed on label: ABCD0123
`#RT105//2/B`	EPC printed in hex notation with font 105 in normal (from left to right) print direction

Related reference

Printer-internal fonts and line styles on page 148

#RTC - Setting the realtime clock

The #RTC command sets time and date of the realtime clock.

Syntax

```
#RTC/date#G
```

‖ The command must stand *outside* of the command sequence #ER to #Q! ‖

Parameter	Value	Description
date		Syntax: `dd.mm.yyyy hh:mm` (dd = day; mm = minute; yyyy = year; hh = hour; mm = minute)

Example

```
#RTC/01.02.2007 01:32#G
```

#RX - Select gap sensor

The #RX command selects one of the available label sensors at the printer.

Syntax

```
#RXn
```

‖ The command must stand *outside* of the command sequence #ER to #Q! ‖

Parameter	Value	Description
n	0	Transparency
	1	Reflex
	2	Full size
	6	Label sensor for short labels ‖ Option only for XLP 504 and XLP 514 ‖

S

#SB - Bar code definition

The #SB command defines of a bar code. Printing of the bar code requires a subsequent #VW-command.

Syntax

```
#SBz/kclbmre/h/s#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
z	int	Bar code number (for details refer to the links below in „Related Information“)

k	M	Bar code with plain-copy line
	O	Bar code without plain-copy line
c	C	A check digit according to Module 10 is calculated and printed. ‖ Works only with 2/5 interleaved bar codes and with ITF bar code! ‖
	N	No check digit calculation
l	H	Plain copy line justified
	I	Plain copy line left-justified
	K	Plain copy line centered
	L	Plain copy line right-justified ‖ Default setting depends on the barcode type. ‖
b	A	Position change of the plain-copy line. If the plain-copy line is normally printed below the bar code, it is shifted by this option to the top of the bar code, and the other way round.
m	B	EAN/UCC mode with brackets around the data designator. ‖ The data have to be sent in brackets! The brackets appear in the plain copy line but not as bar code. ‖
	X	EAN/UCC mode without brackets around the data designator. ‖ Data have to be sent without brackets! ‖
r	Pnum	Ratio of the bar code [2.0...3.0] ‖ The letter P (proportion) must stand immediately in front of the ratio (e.g. P2.5). A ratio without the letter P is invalid. A point (“.”) must be used as decimal separator. ‖
e	V	Bar code is verified. ‖ Only with connected and activated OLV. Only applicable for bar codes which are rotated by 0° or 180°. ‖

h	int	The *bar code height* calculates as follows: Bar code height = (h + 1) mm * PRINT PARAMETERS > Bar code multip. or Bar code height = (h + 1) mm * Print > Format > Bar code multip. ‖ The default setting for Bar code multip. is 1 This results in the following values for the bar code height: • 1 mm bar code height for h = 0 • 2 mm bar code height for h = 1 • 3 mm bar code height for h = 2 ‖

s	int	Bar code width factor [1...30] Dot

Related reference

Printer-internal bar codes on page 144
Relationship between bar code ratio and width factor on page 166

#SCF - Codablock F Definition

The #SCF command defines a “Codablock F” bar code. Printing of the bar code requires a subsequent #VW-command.

Syntax

```
#SCFs/m/c/r#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
s	int	Bar code width [1...30]; Default setting: 1
m	int	Height of a Codablock-row [1...100] mm; Default setting: 5 mm
c	int	No. of columns [4...62]; Default setting: 10
r	int	No. of rows [2...44]; Default setting: 0 ‖ 0: Number of rows is calculated by codablock 1: Improper value, interpretation causes an error message ‖

#SDM - Data Matrix definition

The #SDM command defines a “Data Matrix” bar code. Printing of the bar code requires a subsequent #VW-command.

Syntax

```
#SDMn/irck/s#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
n	int	Encoding methods ‖ For details refer to #IDM - Data Matrix Code ‖
	0	ASCII
	1	C40
	2	TEXT
	3	BASE256
	4	reserved
	5	AUTO (default setting)
i	B	EAN/UCC mode with data designator put in parantheses. Data has to be sent in parantheses, but the parantheses will not be printed coded. Same handling as EAN 128 bar code..
	X	EAN/UCC mode with data designator not put in parantheses. Data has to be sent without parantheses. Same handling as EAN 128 bar code.

r	Rn	n = Number of rows
c	Sn	n = Number of columns
		As a standard, the size (number of rows and columns) is calculated automatically in dependance of the input data (smallest possible matrix). Use the parameters r and c to set the size (see table below).
k	F	<FNC1> is used as data separator (--> Example)
	G	<GS> is used as data separator
		Only applicable for machines of type XPA 93x.

s	int	Number of printer dots for one Data Matrix block [1...200]

Rows	Column		Rows	Columns
10	10		64	64
12	12		72	72
14	14		80	80
16	16		88	88
18	18		96	96
20	20		104	104
22	22		120	120
24	24		132	132
26	26		144	144
32	32		8	18
36	36		8	32
40	40		12	26
44	44		12	36
48	48		16	36
52	52		16	48

Table 12: Admissible combinations of rows (r) and columns (c).

Example

EAN/UCC mode:

```
#ER
#SDM/B/14#G
#T10#J5
#T10#J5
#VW/L/"(01)34012345123457(10)12345 <FNC1>(17)101231"#G
#Q1/
```

Different data separators:

```
#SDM/X/14#G    --> <FNC1> is used as data separator
#SDM/XF/14#G   --> <FNC1> is used as data separator
#SDM/XG/14#G   --> <GS> is used as data separator
```

#SF - Fixfont definition

The #SF command defines the font for a not scalable text field (fixfont). Printing of the text field requires a subsequent #VW-command.

Syntax

```
#SFz/k/b#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
z	int	Font number (for details refer to „Related links“ below)
k	Snum	Fixed distance in mm between all characters in TEXT. Input in millimeters. The distance is measured from character beginning to character beginning. ‖ The character S has to be succeeded without blank by the value “num”. ‖
b	int	Micro spacing measured in printhead dots. The value is added to the standard character distance and allows fine adjustment of the text width. Value range [0…16].

Related reference

Printer-internal fonts and line styles on page 148

#SFN - Code 49 definition

The #SFN command defines a “Code 49” bar code. Printing of the bar code requires a subsequent #VW-command

Syntax

```
#SFNm/kx/s/h#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
m	0	Alphanumeric Mode
	1	Append Mode
	2	Numeric Mode
	3	Group Alphanumeric Mode
	4	Alphanumeric Mode, Shift 1
	5	Alphanumeric Mode, Shift 2
	6	Reserved
	7	Automatic Mode (default setting). The printer determines starting mode and encodation method by analyzing TEXT. This is the recommend mode.

k	M	Bar code with plain copy line ‖ The plain copy line can protrude from the right edge of the code ‖
	O	Bar code without plain copy line (default setting)
x	J	Plain copy line below the bar code
	A	Plain copy line above the bar code

h	int	Row height: Row height = (h + 1) * PRINT PARAMETERS > Bar code multip.

#SG - Graphics definition

The #SG command defines a graphic. Printing of the bar code requires a subsequent #VW-command.

Syntax

```
#SG#G
```

The command must stand *between* #ER and #Q!

Example

Command sequence	Description
`#SG#G`	Graphics definition
`#T82#J75`	Define print position
`#VW/L/"Grafik01.bmp" #G`	Print graphics

#SI - Write data to interface

The #SI command defines the trigger point for the context interface (Easy Plug interface)
This command is useful for many applications, for example logging of printed labels or soft “realtime” controls over the interface.

Writing data to the current Easy Plug interface is done by the #VW/I/expression command. The point of time for interface write (trigger point) can be defined by the #SI command.

Syntax

```
#SIr/n#G
```

Parameter	Value	Description
r	int	Logical point in time (trigger point) concerning print control that defines when to send data (see figure on next page). If a motion control error happens during label execution, possible later trigger operations are not executing. RFID: To transmit data, which were read from or written to the transponder, it is recom-mended to use trigger point r = 8. This makes sure that RFID operations and printing have been successfully completed.
	1	Start of a new Easy Plug format
	2	Restart of a previous Easy Plug format (#Q without #ER)
	3	Start of a label execution
	4	Start of RFID operation
	5	End of *successful* RFID operation(s); if read/write operations fail, the trigger doesn’t fire
	6	Start of the real print process
	7	End of the real print process (executed only, if label is printed without error)
	8	End of a label execution (executed only, if label is executed without error)

Parameter	Value	Description
	9	End of an Easy-Plug format (#Qn label printed)
	10	Any motion control error
	11	Any RFID error (tag read or tag write error); triggered together with the bad tag signal
	12	Dispensed label was taken off (64-xx dispenser, AP 5.x dispenser)
	13	End of the scan area of the online verifier (OLV)

n	int	Number of bytes to respond
	≠ 0	The return string (defined with #VW) is truncated to byte count n. If the respond string counts less than n bytes, it is filled with blanks (hex code 0x20).
	0	The size of the respond string is whatever is necessary to send the requested data (default).

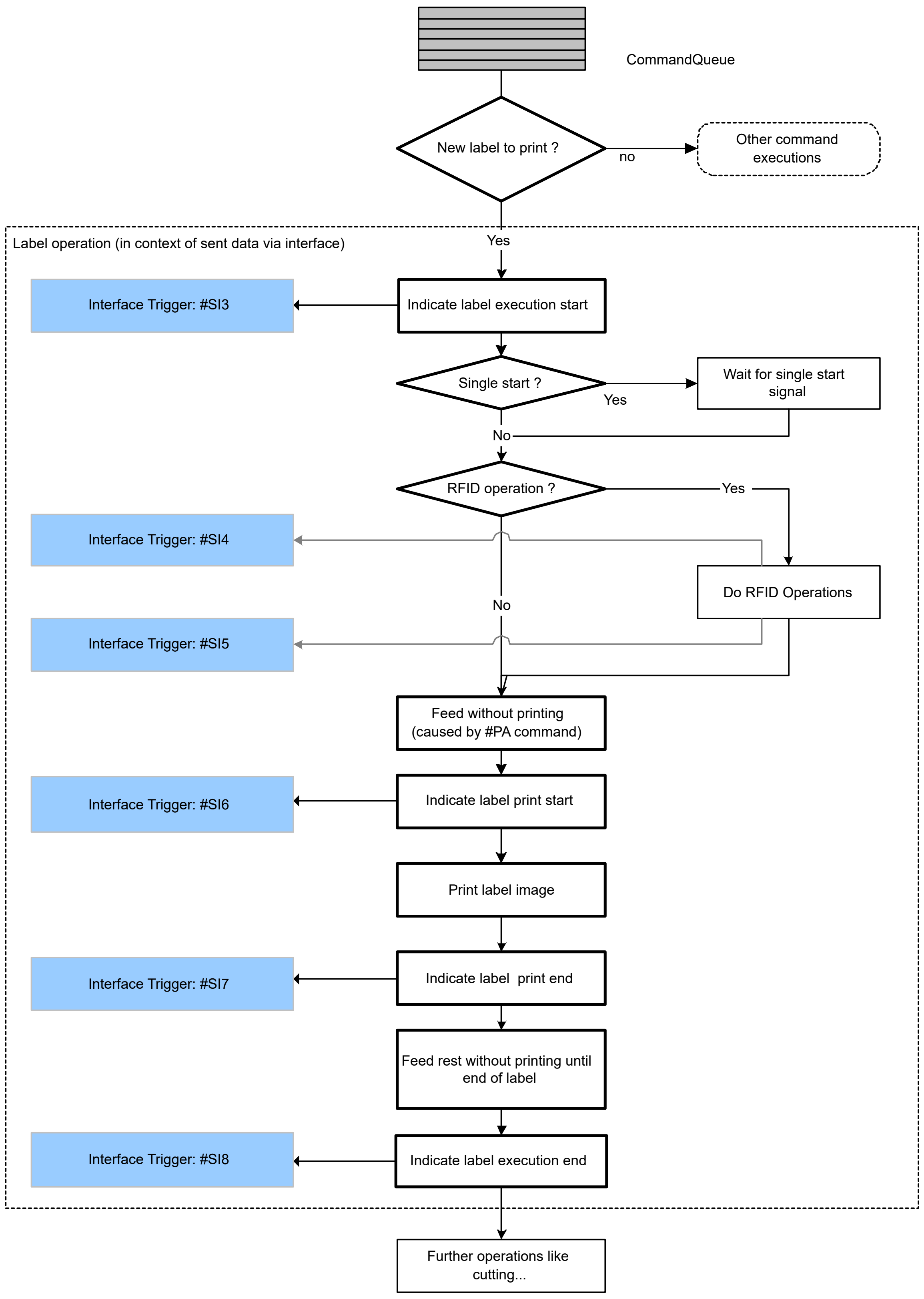


Fig. 14: Flowchart showing label operation trigger points #3 - #8.

Example

Simple example:

```
#G -----------------------------------------------------------------
#G Returns the printer counter value, if the label is printed without
#G error
#G -----------------------------------------------------------------
#!A1
#IMN100/100
#ER
#G ------------------------------------
#G Define and print counter as text
#G ------------------------------------
#VDT/Counter/1/1/0000#G
#SF109/
#FD0
#T10#J20
#VW/L/Counter#G
#G ------------------------------------
#G Return counter value to interface, as soon as a label is printed
#G without error.
#G ------------------------------------
#SI8/#G
#VW/I/chr(10) + chr(13) + "CNT: " + Counter#G
#Q3/
#G -----------------------------------------------------------------
#G Response at active interface:
#G CNT: 0000
#G CNT: 0001
#G CNT: 0002
#G -----------------------------------------------------------------
```

More complex example:

```
#G -----------------------------------------------------------------
#G Shows the usage of more different trigger events
#G -----------------------------------------------------------------
#!A1
#IMN100/100
#ER
#G ------------------------------------
#G Error code return on interface in case of any print related error.
#G ------------------------------------
#VDS/ErrorCode//M1011#G
#SI10/#G
#VW/I/chr(10) + chr(13) + " Error code: " + ErrorCode#G
#G ------------------------------------
#G Easy-PLug format related messages
#G ------------------------------------
#SI1/#G
#VW/I/chr(10)+chr(13)+chr(10)+chr(13) + "*** New Easy Plug format start
 *** "#G
#SI2/#G
#VW/I/chr(10)+chr(13)+chr(10)+chr(13) + "*** Easy Plug format restart
*** "#G
#SI9/#G #VW/I/chr(10) + chr(13) + "*** Easy Plug format finished *** "#G
#G ------------------------------------
#G Label related messages
#G ------------------------------------
#SI3/#G
#VW/I/chr(10)+chr(13) + "  * Label operation start"#G
#G ------------------------------------
#G Define and print counter as text
#G ------------------------------------
```

```
#VDT/Counter/1/1/0000#G
#SF109/
#FD0
#T10#J20
#VW/L/Counter#G
#G ------------------------------------
#G Return conuter value to interface, as soon a label is printed
#G without error.
#G ------------------------------------
#SI8/#G
#VW/I/chr(10) + chr(13) + " Printed Counter: " + Counter#G
#VDT/TextVar///Text1#G
#SF109/
#FD0
#T10#J30
#VW/L/TextVar#G
#SI8/#G
#VW/I/chr(10) + chr(13) + " Printed TextVar: " + TextVar#G
#G ------------------------------------
#G Label operation finished message.
#G ------------------------------------
#VDD/Time/AU//^h:^m.^s#G
#SI8/#G
#VW/I/chr(10)+chr(13) + "  * Label operation finished at: " +Time#G
#Q2/
#SV/TextVar/Neuer Text#G
#Q1/
#G ------------------------------------------------------------------
#G Response at active interface:
#G *** New EasyPlug format start ***
#G  * Label operation start
#G    Printed Counter: 0000
#G    Printed TextVar: Text1
#G  * Label operation finished at 13:07.07
#G  * Label operation start
#G    Printed Counter: 0001
#G    Printed TextVar: Text1
#G  * Label operation finished at 13:07.08
#G *** Easyplug format finished ***
#G
#G *** Easyplug format restart ***
#G  * Label operation start
#G    Printed Counter: 0002
#G    Printed TextVar: New text
#G  * Label operation finished at 13:07.09
#G *** Easyplug format finished ***
#G ----------------------------------------------------------------
```

#SMX - Maxicode definition

The #SMX command defines a “Maxicode” bar code. Printing of the bar code requires a subsequent #VW-command.

Syntax

```
#SMXz/x/y#G
```

Parameter	Value	Description
z	int	MaxiCode modes [2,3,4,6]
x	int	Symbol number for code-splitting [1...8]
y	int	Number of symbols for code-splitting [1...8]

z	TEXT (Syntax)	Data
2	`ppppppppp ccc sss MSG`	p: Numerical postcode (9 characters) c: ISO national code (3 characters) s: Service class (3 characters) MSG [4]: Code words (84 characters)
3	`pppppp ccc sss MSG`	p: Numerical postcode (6 characters) c: ISO national code (3 characters) s: Service class (3 characters) MSG [4]: Code words (84 characters)
In Modes 2 and 3 a space must be left respectively between the postcode, national code, service class and user data. The number of characters in the postcode, national code and service class must be observed exactly.		
4	`MSG`	MSG: Any alphanumerical code (93 characters)
6	`MSG`	

Table 13: Structure and length of the data string “TEXT” in the associated command #VDT, depending on the mode “z”.

Simple example

The string “MSG” contains only alphanumerical characters.

```
#!A1
#IMS100/200

#ER
#VDT/MaxicodeData////NOVEXX Solutions Teststring#G
#SMX4/
#T40#J45
#VW/L/MaxicodeData#G
#Q1#G

#!P1
```

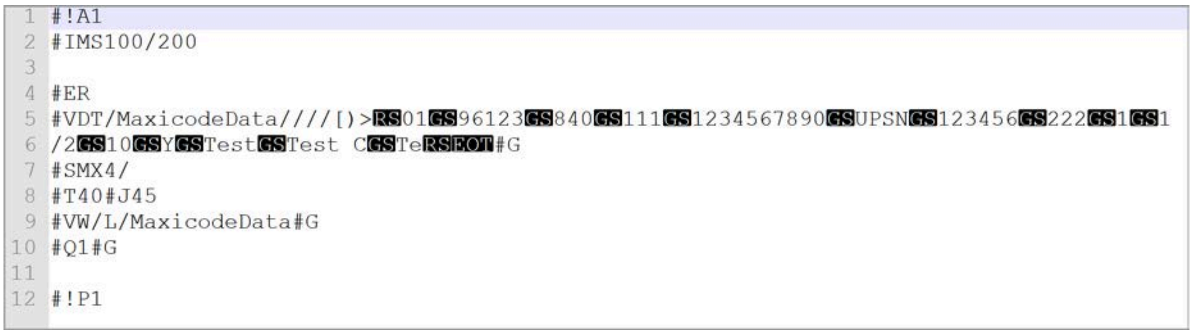
4 Depending on the selection of alphanumerical characters (0x00 – 0xff) results a different amount of user data, because more or less switches between subsets are necessary.

Fig. 15: Printout of the example.

Example with characters < 20h

‖ Important: Set parameter Character filter to “All characters”! ‖

The string “MSG” can also contain non-representable characters, i.e. characters with a hexadecimal code <20h. Such characters are, for example, the control characters <RS> (0x20) and <GS> (0x1E). Special editors make this invisible code visible by using placeholders:



```
#!A1
#IMS100/200

#ER
#VDT/MaxicodeData////[)>RS01GS96123GS840GS111GS1234567890GSUPSNGS123456GS222GS1GS1
/2GS10GSYGSTestGSTest CGSTeRSEOT#G
#SMX4/
#T40#J45
#VW/L/MaxicodeData#G
#Q1#G

#!P1
```

Such strings make it difficult to read or change the source code. The source code below shows an equivalent alternative, but one that is easier to read and change:

```
#!A1
#IMS100/200

#ER
#VDE/RS//Chr(30)#G
#VDE/GS//Chr(29)#G
#VDE/EOT/Chr(4)#G
#VDE/Header//"[" +Chr(41)+ ">" + RS + "01" + GS#G     !! use Chr(41) in▶
stead of ) to prevent a known bug.

#VDE/MaxicodeData//Header + "96123" + GS + "840" + GS + "111" + GS +
 "1234567890" + GS + "UPSN" + GS + "123456" + GS + "222" + GS + "1" + GS
 + "1/2" + GS + "10" + GS + "Y" + GS + "Test" + GS + "Test C" + GS + "Te"
 + RS + EOT#G
#SMX4/
#SMX4/
#T40#J10
#VW/L/MaxicodeData#G
#Q1#G

#!P1
```

Fig. 16: Printout of the example.

Related reference

#MXC - Bar code Maxicode on page 50
The #MXC command prints the 2-dimensional "MaxiCode" bar code.

#SPF - PDF417 definition

The #SPF command defines a “PDF 417” bar code. Printing of the bar code requires a subsequent #VW-command

Syntax

```
#SPFnt/s/l/z/w/h#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
n		Compression type
	0	EXC-Mode (Extended alphanumeric Compaction Mode)
	1	Binary ASCII Plus Mode
t	leer	Bi-directional reading (default setting)
	T	Unidirectional reading
s	int	Security Level [0...8]
l	0	Bar code width is set automatically
	int	Bar code width in code words (number of columns) [1...30]: The width should be so selected that a height of 90 lines is not exceeded with regards to the area of the bar code defined in the user data. A maximum of 928 code words are permitted for one bar code. Of these the following are preassigned:: • 1 word for the length input of the code • x words for the check total (Security Level) = 2 (1 + Security Level)
z	0	Number of lines is set automatically
	int	Number of lines [3...90]

w	int	Bar code width [1...16]
h	num	Height of a PDF bar code row [1...100] mm

#SQR - QR Matrix Code definition

The #SQR command defines a “QR matrix code” bar code. Printing of the bar code requires a subsequent #VW-command.

Syntax

```
#SQRm/ei/s/an/d/p#G
```

‖ The command must stand *between* #ER and #Q! ‖

Usage pattern:

1. Define data in variable:

```
#VDT/QR_Data////qr code#G
```

2. Set context parameters:

```
#SQR2/LU/6///#G
#FD/0/L#G#T5.0#J10.0
```

3. Write to media context “label”:

```
#VW/L/QR_Data#G
```

Parameter	Value	Description
m	int	
	1	QR matrix code model 1 (original standard)
	2	QR matrix code model 2 (enhanced standard1; default setting) [5]
e		Error correction levels:
	L	7% High density
	M	15% Standard (default)
	Q	25% High reliability
	H	30% Ultra high reliability
i		Character sets:
	A	Automatic (input data type is selected automatically; default)
	U	User defined (input data type is defined by the user)
s	int	Modulgröße in Pixel (Minimum: 4; Voreinstellung: 4)
a	S	Einzelnes Symbol (Voreinstellung)

5 Defined in ISO/IEC 18004:2000

	A	Aktiviert “Structured Append” (engl. für “strukturiertes Anhängen”, d. h. ein QR-Code-Symbol kann in mehrere kleinere Symbole aufgeteilt werden; so kann z. B. der zur Verfügung stehende Raum besser ausgenutzt werden)
n	int	Symbol-Sequenznummer: [1...“Anzahl der Sequenzen“] (gilt, wenn “Structured Append“ aktiviert ist)

d	int	Symbolanzahl: [2...16] (gilt, wenn “Structured Append“ aktiviert ist)

p	int	Parität: [0...255] (gilt, wenn “Structured Append“ aktiviert ist)

Marker	Data type
N	*Numeric data* (digits 0 - 9, ex. **N**0123456789); each 3 numbers are stored as a 10 Bit unit
A	*Alphanumeric data* (digits 0 - 9; upper case letters A -Z; nine other characters, ex. **A**---123-ABC %$+-/:); each 2 characters are stored as a 11 Bit unit
B	*Binary data*. The number of bytes must be specified after the ‘B’ by a 4 digit number in decimal (ex. **B**0023#G)
K	*Kanji characters*. Kanji characters in QR Code can be compacted into 13 bits

Table 14: Zulässige Benutzerdefinierte Datentypen. Der Buchstabe für den Datentyp muss den Eingabedaten unmittelbar vorangestellt sein.

Example

```
#!A1
#IMN100.0/100.0//
#N10
#ERN//
#VDT/QR_Data_StructuredAppend_Part1////AAAAAAAA#G
#VDT/QR_Data_StructuredAppend_Part2////BBBBBBBB#G
#VDT/QR_Data_StructuredAppend_Part3////CCCCCCCC#G
#VDT/QR_Data_StructuredAppend_Part4////DDDDDDD#G
#VDT/QR_Data_Manual////N0123456789,A---123-ABC %$+-/:,B0004ÿÿÿ#G
#VDE/QR_Data_4PartsInOne/QR_Data_StructuredAppend_Part1+
QR_Data_StructuredAppend_Part2+QR_Data_StructuredAppend_Part3+
QR_Data_StructuredAppend_Part4#G
```

```
#FD/0/L#G
#SQR2/QU/4///#G
#T40.0#J80.0
#VW/L/QR_Data_Manual#G
```

```
#G --- Binary data "Test<cr><lf>"
#SQR2/QU/4///#G
#T20.0#J40.0
#VW/L/"B0006Test"+CHR(10)+CHR(13)#G
```

```
#G --- All in one
#SQR2/LA/4///#G
#T50.0#J25.0
#VW/L/QR_Data_4PartsInOne#G
```

```
#G --- Structured Append 1..4
#SQR2/LA/4/A1/4/255#G
#T20.0#J10.0
#VW/L/QR_Data_StructuredAppend_Part1#G
```

```
#SQR2/LA/4/A2/4/255#G
#T40.0#J10.0
#VW/L/QR_Data_StructuredAppend_Part2#G
```

```
#SQR2/LA/4/A3/4/255#G
#T60.0#J10.0
#VW/L/QR_Data_StructuredAppend_Part3#G
```

```
#SQR2/LA/4/A4/4/255#G
#T80.0#J10.0
#VW/L/QR_Data_StructuredAppend_Part4#G
#Q1#G
```

#SRF - RFID read/write definition

The #SRF command defines the target (write) or source field and address (read) of any following #VW/T or #VR/T command.

Syntax

```
#SRFt/bfs/n/r#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
t	0	BLOCK ‖ Parameters „s“ and „n“ required ‖
	1	UID/TID ‖ “n” optional – for limiting size of data returned for read operations ‖
	2	EPC
	3	KILL password
	4	ACCESS password
	5	MEM BANK
	6	AFI
	7	MEMBLOCK ‖ Parameters „s“ and „n“ required ‖
	8	DSFID

b	L	(Default) The bytes of the expression value used in the #VW/T resp. #VR/T command are written to the transponder with the least significant (LSB) byte first.
	M	Same as above, with the most significant byte (MSB) first.
f	P	Write protection
s	int	If Data type = • BLOCK: start block address • MEM BANK: address/name of memory bank; (default: 0) • MEMBLOCK: combination of memory bank identifier and address offset in the memory bank. Schema for address calculation: `s = <EPC MemoryBankID> * 1000 + <BlockNumber>` EPC MemoryBankID: – 0: reserved (default setting) – 1: EPC – 2: TID – 3: USER – Other values are ignored • Other data type: parameter is ignored

n	int	If... • Data type = BLOCK, TID, MEM BANK or MEMBLOCK: number of blocks to be read (default: 0). • Other data type: parameter is ignored For reading a TID field (t=1), the size parameter can be used to limit the size of the returned data. Giving no size information (n=0) will return the whole content of the TID memory bank, which may contain more information than expected (depending of RFID transponder/chip). Typical values for n are: • 2: Read 32Bit Vendor-ID/Chip-ID • 4: Read 32Bit Vendor-ID/Chip-ID + 32Bit Unique Serial Number

r	int	Number of read/write retries (default setting = setting in RFID PARAMETER > Nr CMD retries).

Reading from / writing to a RFID tag

Reading data from and/or writing data to a RFID tag always requires a variable and two explicit commands:

1. Define a variable: #VDT (or #VDE / #VDD)
2. Define target field or data type: #SRF
3. Read or write data: #VR or #VW

#SRS - GS1 bar code definition

The #SRS command defines a “GS1 DataBar” (formerly “RSS”) bar code. Printing of the bar code requires a subsequent #VW-command.

Syntax

```
#SRSzt/s#G
```

The command must stand *between* #ER and #Q!

Parameter	Value	Description
z	int	Bar code number (see link below under „Related Information“) GS1 DataBar Expanded Stacked is applied automatically, if the following conditions are met: • Selected barcode type: GS1 DataBar Expanded (z = 6) • Width (t) is set to less than 22 symbol characters per row • Primary data exceeds the set width
t	Sn	Width in segments; value range: n = [4…22]
s	int	Number of printer dots for the bar code module width.

#SS - Speedo font definition

The #SS command defines the font for a scalable text field (speedo font). Printing of the bar code requires a subsequent #VW-command

Syntax

```
#SSz/ghrkp/ex/s#G
```

The command must stand *between* #ER and #Q!

Parameter	Value	Description
z	int	Font number (see link under „Related information“) If an invalid font number is specified, font no. 100 is used automatically .
g	O	Default type
	B	Bold
h	V	Default type

	I	Italics
r	K	Application of scalable fonts without Unicode (default setting)
	U	Application of Unicode fonts
	G	Application of arabic Unicode fonts (reading direction right to left)
k	Snum	Fixed distance in mm between all characters in TEXT. The distance is measured from character beginning to character beginning. ‖ The character S has to be succeeded without blank by the value “num”. ‖
p	N	(Default) Standard character-algorithm is applied.
	Q	2 D character algorithm is used. May improve the exactness of small characters. (e.g. printing of small chinese characters). This algorithm is about 30-40% slower than the standard algorithm.
e	int	Font size in dot in y-direction, depending on the font and its installed size, respectively (see example below). Value range: [5…6000] Dot
x	Xint	Font size in dot in x-direction, depending on the font and its installed size, respectively (see example below). Value range: [5…6000] Dot. ‖ The character X has to be succeeded without blank by the value “int”. ‖
s	int	Micro spacing measured in printhead dots. The value is added to the standard character distance and allows fine adjustment of the text width. Value range [0…16].

Related reference

Printer-internal fonts and line styles on page 148

#SV - Changing the content of a text variable

Syntax

```
#SV/name/TEXT#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
name		Variable name (must be defined with #VDT before!)
TEXT		New value of the variable (max. length: 4096 characters)

T

#T - Horizontal print position

The #T command determines the horizontal print position with absolute value in mm

Syntax

```
#Tx
```

The command must stand *between* #ER and #Q!

Parameter	Value	Description
x	num	Horizontal print position in mm in relation to the zero position. The horizontal zero position is always the position of the first dot of the printhead. Looked at in feed direction, this dot is placed at the right end of the printhead. Exception: The zero position was shifted with the #R command. If the printhead is positioned at the inner limit (right end if viewed in feed direction), the first dot has a distance of 1 mm to the material zero line. In other words: Looked at in feed direction there is a 1 mm wide stripe at the right end of the label, which stays unprinted.

Text outside of the defined label area is not printed.

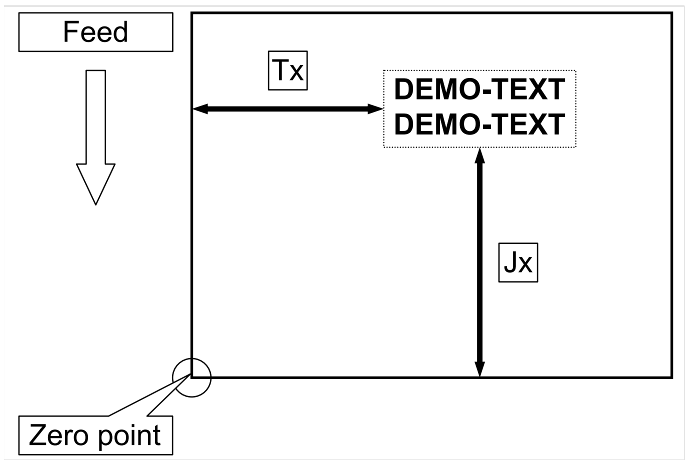


Fig. 17: Vertical (Jx) and horizontal (Tx) print position on the label.

Example

Text begins 5 mm from the left:

```
#T5
```

Text begins 20 mm from the left:

```
#T20
```

V

#VDD - Define date and time variable

The #VDD command defines a variable which can contain date and/or time.

Syntax

```
#VDD/name/uv/o/TIMETEXT#G
```

> The device must be equipped with a realtime-clock.
>
> The command must stand *between* #ER and #Q!

Parameter	Value	Description
name		Name of the variable. Admissible are all characters except of some special characters, see chapter .

u	leer W	(Default setting) No update. The time is read into the variable once. This is done during the compilation of the layout.
	U	The time is read repeatedly during the running print job. This happens immediately after printing the relevant output field (#VW…), but counts only for the same field on the next label.. The setting „U“ is not suitable for dispensers. Reason: If the next label is printed considerably later (e.g. in single-start mode with footswitch), the following happens: The output field on label 2 contains the time, at which the same field on label 1 was printed. Example: Label 1 is printed at 10.50 h, label 2 at 11.03 h. In this case, the imprint of label 2 shows the time 10.50!
	Y	(Recommended for dispensers) The time is read repeatedly during the printing of a label (approx. all 5 seconds).
v	A	Different times may be printed on a label (if option U or Y is applied).

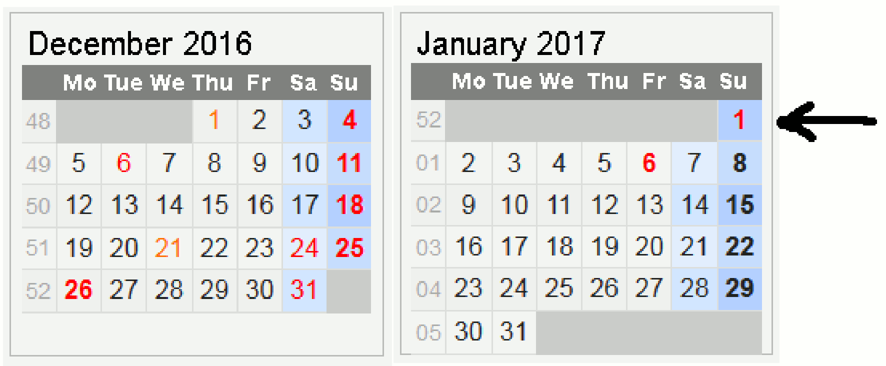
o	blank 0	The current date is printed.
	int	*Days offset*: The current date is calculated forward by the entered number of days [1...65000].
	Mint	*Months offset*: The current date is calculated forward by the entered number of months [1...65000]. Adding the letter M to the integer value changes the offset from day to month.
	Pint	*Minutes offset*: The current time is calculated forward by the entered number of minutes [1...65000].
	Hint	*Hours offset*: The current time is calculated forward by the entered number of hours [1...65000].

TIMETEXT		String containing Time and Text (ASCII-characters) in any order. Times are marked with the control character (^). Insert required Text as ASCII characters (without control characters) in the TIMETEXT-string.

Time formats:

^z	1/100 seconds
^s	Seconds
^m	Minutes
^h	Hours
^D	Day of the month
^d	Day of the year
^W	3-figure day (e.g. 001 for the 1st day of the year)
^w	Day of week (1-figure number of the day of week, e. g. 1 for Monday)
^M	Month
^Y	Year, two digit
^R	Year, four digit
^G	3-figure month, German
^E	3-figure month, English
^S	Name of a month from command #DM
^C	Week of the year always with two digits
^c	Week of the year without leading 0
^K	Corresponding year to the week (of the year) with 4 digits
^k	Corresponding year to the week (of the year) with 2 digits

Note for ^K and ^k: The year is calculated from the week of the year. Due to the fact that the year often starts in the middle of the week, the first days may count to the last week of the year of the preceding year (see fig. below).


December 2016

	Mo	Tue	We	Thu	Fr	Sa	Su
48				1	2	3	4
49	5	6	7	8	9	10	11
50	12	13	14	15	16	17	18
51	19	20	21	22	23	24	25
52	26	27	28	29	30	31	

January 2017

	Mo	Tue	We	Thu	Fr	Sa	Su
52							1
01	2	3	4	5	6	7	8
02	9	10	11	12	13	14	15
03	16	17	18	19	20	21	22
04	23	24	25	26	27	28	29
05	30	31					

Fig. 18: Example: The year for the 1.1.2017 would here be 2016, because the date lies in week 52.

Related reference

Variables on page 150
Functions on page 152
Expressions on page 150

#VDE - Define expression variable

The #VDE command defines a variable, to which an expression can be assigned

Syntax

```
#VDE/name/o/expression#G
```

Parameter	Value	Description
name		Name of the variable. ‖ Admissible are all characters except of some special characters, see chapter . ‖
o		Not applied option for future firmware versions
expression		An expression can contain constant values, variables and functions

Example

```
#VDE/BarcodeData//"(00)"+NVE+MOD10(NVE)#G
```

with:

" (00)"	Text
+	Operator
NVE	Variable
MOD10(NVE)	Function MOD10 with variable NVE as argument

Related reference

Variables on page 150
Functions on page 152
Expressions on page 150

#VDO - Define variable for OLV data access

The #VDO command defines a variable with the purpose of accessing the data provided by an online verifier (OLV).

Syntax

```
#VDO/name/TEXT#G
```

Parameter	Value	Description
name		Name of the variable. ‖ Admissible are all characters except of some special characters, see chapter . ‖

TEXT		OLV Data element

Description OLV Data element

For each bar code that was scanned by the OLV, an entry is made in a data array. The single entries can be addressed via an index. Indexing starts with „1“. The entries are numbered in the order their top edges appear on the label in feed direction (see fig. below). Each entry contains the data elements listed in the following table:

Data element	Value	Description
Barcode[idx].Data		Bar code data
Barcode[idx].bLimitsOK	0	The bar code limits are not o. k. because of one of the following reasons: • Bar code was not read • The bar code limits were exceeded
	1	The bar code limits are not o. k.
Barcode[idx].bChecked	0	The bar code could not be read by the OLV
	1	The bar code was read
OLVError	0	All bar codes were printed and read without a fault
	1	No OLV data available
	2	OLV limits exceeded
	4	Wrong bar code type

Table 15: Data elements and their possible values.

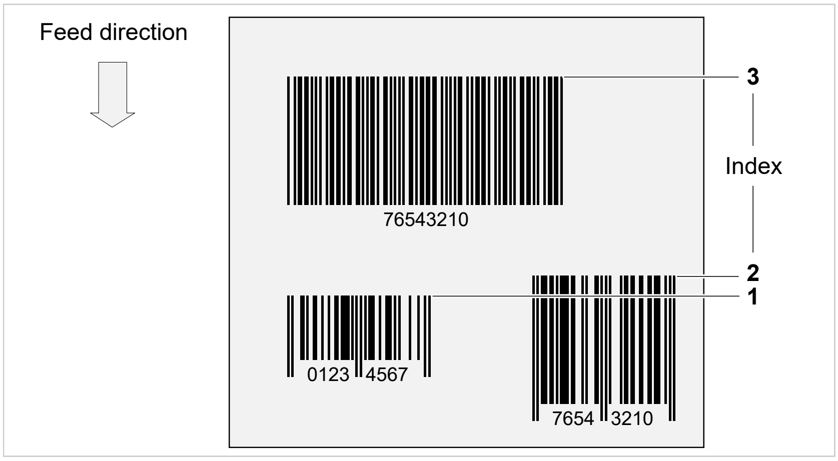


Fig. 19: The indexing of the bar codes is done in the order of the bar code top edges on the label in feed direction.

A faulty call of the variable causes the message:

```
Queuestatus 2011
OLV Variable
```

Possible error causes are e. g..:

- „Barcode“ wrong spelled, e. g. „Baarcode[3].Data“ or „Barcode[3].Date“
- Wrong index number referenced, e. g. „Barcode[36].Data“, if only 26 bar codes were read

Example

```
#IMN109.0/30
#ERN0///0
#SI1/#G
#VW/I/"<SOL>"+chr(10)+chr(13)#G
#G --------------------
#G Variables definition
#G --------------------
#VDO/OLVD1/Barcode[1].Data#G
#VDO/OLVOK1/Barcode[1].bLimitsOK#G
#VDO/OLVD2/Barcode[2].Data#G
#VDO/OLVOK2/Barcode[2].bLimitsOK#G
#VDO/OLVD3/Barcode[3].Data#G
#VDO/OLVOK3/Barcode[3].bLimitsOK#G
#VDO/OLVErr/OLVError#G
#G ------------------------------
#G Writing to interface definition
#G ------------------------------
#SI13/#G Trigger point at end of OLV scan area
#VW/I/OLVD1 + IfEqualThenElse(OLVOK1,"0"," NOK"," OK")+chr(10)+chr(13)#G
#VW/I/OLVD2 + IfEqualThenElse(OLVOK2,"0"," NOK"," OK")+chr(10)+chr(13)#G
#VW/I/OLVD3 + IfEqualThenElse(OLVOK3,"0"," NOK"," OK")+chr(10)+chr(13)#G
#VW/I/"Canceled : "  + IfEqualThenElse(OLVErr,"0","Nein","Ja")
 +chr(10)+chr(13)#G
#T5.4#J7.4#YB13/0O/10/3///1100640210#G
#T40.4#J7.4#YB13/0O/10/3///1100640211#G
#T75.5#J7.4#YB13/0O/10/3///1100640212#G
#SI8/#G
#VW/I/"<EOL>"+chr(10)+chr(13)#G
#Q1#G
```

Generated output at Easy Plug interface:

- Without label cancellation:

```
<SOL>
1100640210 OK
1100640211 OK
1100640212 OK
Abgebrochen: No
<EOL>
```

- With label cancellation:

```
<SOL>
1100640210 NOK
1100640211 OK
1100640212 NOK
Abgebrochen : Ja
1100640210 OK
1100640211 OK
1100640212 OK
Abgebrochen : No
<EOL>
```

#VDP - Access to print job related data

The #VDP command enables accessing print job related data like „print quantity“ or „label number“. This can be used to print an info label (e. g. “x of y”, with x == current label number, y == print quantity).

Syntax

```
#VDP/name/format/JobItemID#G
```

Parameter	Value	Description	
name		Name of the variable. ‖ Admissible are all characters except of some special characters, see chapter . ‖	
format		Format of the system variable. The format can consist of the following placeholders:	
	blank	No entry. The default format „%i“ (pure number) is used.	
	%[flag][width]i	**flag = 0**	Left-pads the number with zeroes (0) instead of spaces, when padding is specified (see „width“)
		width = number	Minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded with blank spaces. The value is not truncated even if the result is larger than the given number of characters.
JobItemID	I1	Number of the current label of the current print job	
	I2	Print quantity of the current print job	
	I3	Total number of the current label (if the current print job is reused by another #Q but without #ER)	

	I4	Total print quantity (if the current print job is reused by another #Q but without #ER)

Example

```
#!A1
#IMN104/35
#ER
#VDP/CurrentLabel/%3i/I1#G
#VDP/CurrentQuantity//I2#G
#VDP/TotalLabel/%03i/I3#G
#VDP/TotalQuantity/%06i/I4#G
#FD/0/L#G
#SF109#G
#T5#J15
#VW/L/"Label "+CurrentLabel+" of " + CurrentQuantity#G
#T5#J5
#VW/L/"Total "+TotalLabel+" of " + TotalQuantity#G
#Q3/
#Q2/
```

Printout:

```
Label 1 of 3
Total 001 of 000003
Label 2 of 3
Total 002 of 000003
Label 3 of 3
Total 003 of 000003
Label 1 of 2
Total 004 of 000005
Label 2 of 2
Total 005 of 000005
```

Related reference

Variables on page 150
Functions on page 152
Expressions on page 150

#VDS - Define system variable

The #VDS command defines a system variable, to which can be assigned name and value of each parameter in the printer menu

Syntax

```
#VDS/name/format/RemoteID#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
name		Name of the variable. ‖ Admissible are all characters except of some special characters, see chapter . ‖

format		Format of the system variable. The format can consist of the following placeholders:
	%N	Parameter name.
	%S	Parameter value + unit as string (equals the string on the printer display).
	%i	Parameter value as integer value (precondition for „%i“: the parameter must be of the integer type)

RemoteID	int	ID-no. of the parameter A listing of all available ID-numbers can be generated by calling SPECIAL FUNCTION > Store Parameters or Tools > Diagnostic > Store Parameters. See also command #!PG.
	Mint	ID-no. of the machine status List of ID numbers see #!XMn command.

Example

Example Parameter:

```
#G ------------------------------------------------------------------
#G Given setting: SYSTEM PARAMETERS > Temp. reduction = „20%‟
#G ------------------------------------------------------------------
#VDS/my_variable/%N/2026#G
#G  -------> Content of my_variable: „Temp. reduction‟
#VDS/my_variable/%S/2026#G
#G  -------> Content of my_variable: „ 20% ‟
#VDS/my_variable/%i/2026#G
#G  ----> Content of my_variable: „ 20‟
```

Example Machine status:

```
#!A1
#IMN100/100
#ER
#G ------------------------------------
#G Write the error code to the interface, if an error occured during
#G printing.
#G ------------------------------------
#VDS/ErrorCode//M1011#G
#SI10/#G
#VW/I/chr(10) + chr(13) + " Error code: " + ErrorCode #G
```

Related reference

Variables on page 150
Functions on page 152
Expressions on page 150

#VDT - Define text variable

The #VDT command defines a text variable with counter.

Syntax

```
#VDT/name/wz/vop/a/TEXT#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
name		Variable name. ‖ Admissible are all characters except of some special characters, see chapter . ‖
w	W	Counter field TEXT is incremented/decremented *without* a carry over, i.e. only the first unit position of the figure is increased/decreased
	C	Counter field TEXT is incremented/decremented *with* a carry over.
z	Z	Leading zeros in counter fields are printed.
	S	Suppresses leading zeros in counter fields. The remaining digits are printed at the same position they were printed at with leading zeros.
v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]
a	int	No. of labels with constant No. [1...255].
TEXT		Any alphanumerical text, whereby only numerical values are taken into account during offset. All numerical values in the text string are taken into account for the offset (e.g. 10A3B56 = 10356). Maximum number of characters: 255.

Related reference

Variables on page 150
Functions on page 152
Expressions on page 150

#VR - Read RFID Data

Syntax

```
#VR/Context/Expression#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
Context	T	Data are read from the RFID-transponder.
Expression		See link below.

Related reference

Expressions on page 150

#VTS - Define standalone variable

The #VTS command defines a variable, which gets a value in standalone mode via the printer operation panel or an external keyboard.

Syntax

```
#VTS/name/wzy/l/vop/a/TEXT#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
name		Name of the variable. ‖ Admissible are all characters except of some special characters, see chapter . ‖
w	W	Counter field TEXT is incremented/decremented without a carry over, i.e. only the first unit position of the figure is increased/decreased.
	C	Counter field TEXT is incremented/decremented with a carry over.
z	Z	Leading zeros in counter fields are printed.
	S	Suppresses leading zeros in counter fields. The remaining digits are printed at the same position they were printed at with leading zeros.
y	U	Update of the standalone input field. After each run of the print job, the latest counter value appears as default on the printer display.
	N	*No update* of the standalone input field. After each run of the print job, the default counter value appears as default on the printer display.
l	int	Maximum length of the input field on the operation panel in standalone mode.
v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign

p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]

a	int	No. of labels with constant No. [1...255].

TEXT		Any alphanumerical text, whereby only numerical values are taken into account during offset. All numerical values in the text string are taken into account for the offset (e.g. 10A3B56 = 10356). Maximum number of characters: 255.

Related reference

Variables on page 150
Functions on page 152
Expressions on page 150

#VW - Drawing/writing to target

Syntax

```
#VW/Context/Expression#G
```

Parameter	Value	Description
Context	T	Target = RFID tag, what means that the value of the expression is sent to a RFID transponder.
	I	Target = Interface, what means that the expression value is sent to the interface, which is defined in INTERFACE PARA > >OPTIONS > #VW/I Interface or Printer Language > EasyPlug Setting > #VW/I Interface.
	L	Target = Label, what means that the expression value is printed on a label.

Expression		See link below

Related reference

Expressions on page 150

NOVEXX SOLUTIONS

Y

#YB - Bar code definition

The #YB command is used for (optional) parameter settings and entering text.

Syntax

```
#YBz/dkbcejgrwlm/h/s/vop/a/TEXT#G
```

The command must stand *between* #ER and #Q!

The command must be closed with #G!

Parameter	Value	Description
z	int	Bar code number (for details click link under „Related Information“)

d	0	Bar code in normal write direction
	1	Bar code rotated by 90 degrees
	2	Bar code rotated by 180 degrees
	3	Bar code rotated by 270 degrees
k	M	Bar code with plain-copy line
	O	Bar code without plain-copy line
b	A	Position change of the plain-copy line. If the plain-copy line is normally printed below the bar code, it is shifted by this option to the top of the bar code, and the other way round.
c	C	A check digit is calculated and printed. Works only with bar codes with optional check digit.
e	V	Bar code is verified (see example below). Only with connected and activated OLV. Only applicable for bar codes which are rotated by 0° or 180°.
j	Z	*Centered*: The field is built up on the left and right in the centre of the print position.
	R	*Flush right*: The field is built up flush right to the print position. The field is built up to the left.
g	D	Text field consists of one variable data field
r	Pnum	Ratio of the bar code [2.0...3.0] The letter P (proportion) must stand immediately in front of the ratio (e.g. P2.5). A ratio without the letter P is invalid. Observe the sequence: The ratio must be entered after the parameter d (write direction)!
w	W	Counter field TEXT is incremented/decremented without a carry over, i.e. only the first unit position of the figure is increased/decreased.
l	H	Plain copy line justified (default setting)
	I	Plain copy line left-justified
	K	Plain copy line centered

	L	Plain copy line right-justified
m	B	EAN/UCC mode with brackets around the data designator. The data have to be sent in brackets! The brackets appear in the plain copy line but not as bar code.
	X	EAN/UCC mode without brackets around the data designator. Data have to be sent without brackets!

h	int	The *bar code height* calculates as follows: Bar code height = (h + 1) mm * PRINT PARAMETERS > Bar code multip. or Bar code height = (h + 1) mm * Print > Format > Bar code multip. The default setting for Bar code multip. is 1 This results in the following values for the bar code height: • 1 mm bar code height for h = 0 • 2 mm bar code height for h = 1 • 3 mm bar code height for h = 2

s	int	Bar code width factor [1...30] Dot

v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]

a	int	No. of labels with constant No. [1...255].

TEXT		Any alphanumerical text. The stipulations for the respective bar code must be taken into consideration. Max. number of characters: 255. The text field may alse be a *variable data field*. Precondition: the D-flag must be set. The text field may contain an *input field*.

NOVEXX SOLUTIONS

Examples

Command	Description
`#YB2/0M/2/2///1234567890123#G`	UPCA with plain-copy line, normal print direction, 3 mm high, width 2, without offset.
`#YB0/0/////12345678` `#YB11/0/////54321#G`	EAN 8 and ADD on 5 are positioned in rows one after the other.
`#YB1/0D/10/3///$01,12#G`	Bar code number 1 with variable data field, numbers 01 and 12, alphanumerical characters.
`#YB15/0B/2/3///(10)Charge1#G`	EAN/UCC mode with data designator in brackets (here: 10).
`#YB15/0X/2/3///10Charge1#G`	AN/UCC mode without brackets around the date designator (here: 10).
`#YB27/0M/10/3/o/a/xxxxxxxyyyyyyyy#G`	Code 128 Pharmacy xxxxxxx = 7-digit CNK number (Pharmacy Product Ident Number) yyyyyyy = 8-digit sequential number for the first code.

Verifying bar code:

Command sequence	Description

`#!A1`
`#IMS100/100`
`#ER`
`#T5#J5`
`#YT109/0///OLV Test`
`#OLVD/ /70///`	

`#T5#J50 #YB0/0V/4/4///7777777`	Bar code is verified (Decodability >= 70)

`#T5#J30#YB0/0/8/8///1234567`
`#OLVD/ /60//30/`
	Bar code is not verified

`#T5#J70 #YB0/0V/8/8///8888888`	Bar code is verified (decodability >= 70 and defects <= 30)
`#Q1/`	

Related reference

Input Fields on page 10
Variable Data Fields on page 9
Printer-internal bar codes on page 144
Relationship between bar code ratio and width factor on page 166

#YC - Real time as text

The #YC command defines a time in text format. The printer must be equipped with a real-time clock. Normal text can also be entered in conjunction with the time.

Syntax

```
#YCz/dbqjuk/o/TIMETEXT
```

The command must stand *between* #ER and #Q!

Any number of #YC commands are possible in the format. However, per print job only up to 3 commands (#YC or #YS) can be assigned the option U (update during the print job).

Parameter	Value	Description
z	int	Character set number (for details see link below under „Related Information“)

d	0	Text in normal write direction
	1	Text rotated by 90 degrees
	2	Text rotated by 180 degrees
	3	Text rotated by 270 degrees
b	A	Text printed inversely A dark background must first be applied to the print area (line or rectangle).
q	E	Existing Bitimage is inverted (see example 1 below).
j	M	*Middle*: The write command (#T/#J) refers to the centre of the text field. The field is built up on both sides of the print position.
	R	*Flush right*: The write command (#T/#J) refers to the right edge of the text field. The field is built up to the left.
u	blank W	(Default setting) No update. The time is read into the variable once. This is done during the compilation of the layout.
	U	The time is read repeatedly during the running print job. This happens immediately after printing the relevant output field (#VW…), but counts only for the same field on the *next* label. The setting is not suitable for dispensers. Reason: If the next label is printed considerably later (e.g. in single-start mode with footswitch), the following happens: The output field on label 2 contains the time, at which the same field on label 1 was printed. Example: Label 1 is printed at 10.50 h, label 2 at 11.03 h. In this case, the imprint of label 2 shows the time 10.50!
	Y	(Recommended for dispensers) The time is read repeatedly during the printing of a label (approx. all 5 seconds).
k	Snum	Fixed distance in mm between all characters in TEXT. The character S has to be succeeded without blank by the value.

o	blank 0	The current date is printed.
	int	*Days offset*: The current date is calculated forward by the entered number of days [1...65000].
	Mint	*Months offset*: The current date is calculated forward by the entered number of months [1...65000]. Adding the letter M to the integer value changes the offset from day to month.
	Pint	*Minutes offset*: The current time is calculated forward by the entered number of minutes [1...65000].

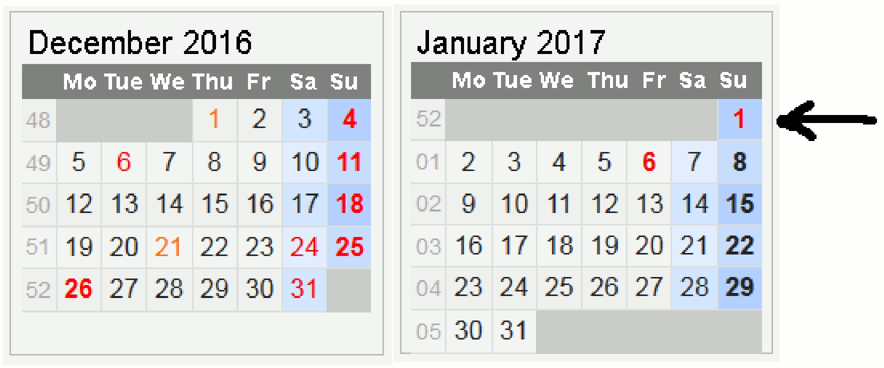
	Hint	*Hours offset*: The current time is calculated forward by the entered number of hours [1...65000].

TIMETEXT		String containing Time and Text (ASCII-characters) in any order. Times are marked with the control character (^). Insert required Text as ASCII characters (without control characters) in the TIMETEXT-string.

Time formats:

^z	1/100 seconds
^s	Seconds
^m	Minutes
^h	Hours
^D	Day of the month
^d	Day of the year
^W	3-figure day (e.g. 001 for the 1st day of the year)
^w	Day of week (1-figure number of the day of week, e. g. 1 for Monday)
^M	Month
^Y	Year, two digit
^R	Year, four digit
^G	3-figure month, German
^E	3-figure month, English
^S	Name of a month from command #DM
^C	Week of the year always with two digits
^c	Week of the year without leading 0
^K	Corresponding year to the week (of the year) with 4 digits
^k	Corresponding year to the week (of the year) with 2 digits

Note for ^K and ^k: The year is calculated from the week of the year. Due to the fact that the year often starts in the middle of the week, the first days may count to the last week of the year of the preceding year (see fig. below).



December 2016

	Mo	Tue	We	Thu	Fr	Sa	Su
48				1	2	3	4
49	5	6	7	8	9	10	11
50	12	13	14	15	16	17	18
51	19	20	21	22	23	24	25
52	26	27	28	29	30	31	

January 2017

	Mo	Tue	We	Thu	Fr	Sa	Su
52							1
01	2	3	4	5	6	7	8
02	9	10	11	12	13	14	15
03	16	17	18	19	20	21	22
04	23	24	25	26	27	28	29
05	30	31					

Fig. 20: Example: The year for the 1.1.2017 would here be 2016, because the date lies in week 52.

Examples

1. The part of the text which overlaps the line is printed in white, the rest in black:

```
#!A1
#ER
#T5#J5#YL0/0/2.5/90#G
#T5#J5#YC109/0E//Example#G
#Q1/
```

2. Printout which is updated during the current print job: Time 08:22.18,07 Date 23-Nov-1998:

```
#YC105/0U// Time ^h:^m.^s,^z Date ^D-^G-^R#G
```

3. If the current date is 01/31/2001, the following command generates the printing "02/28/2001":

```
#YC109/0/M1/^D.^M.^R#G
```

4. If the current date is 09/13/2001, the following command generates #G the printing "Day of the year : 256":

```
#YC109/0/Day of the year: ^d#G
```

Related reference

Printer-internal fonts and line styles on page 148

#YE - Circle or ellipse definition

The #YE command defines a circle or ellipse.

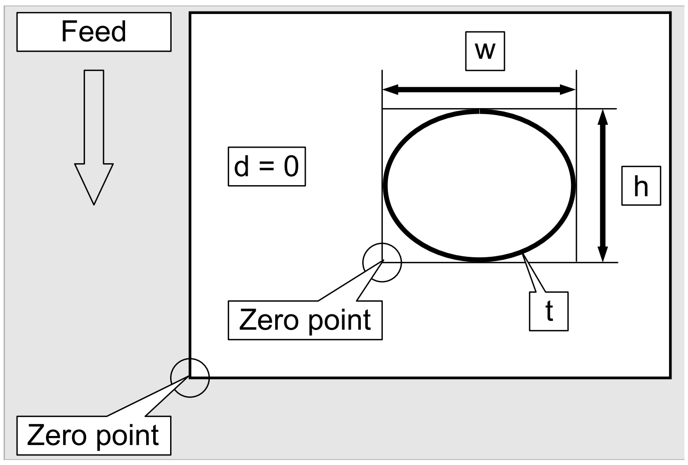
Syntax

```
#YE/d/t/w/h
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
d	0	Circle/ellipse in normal write direction
	1	Circle/ellipse rotated by 90 degrees
	2	Circle/ellipse rotated by 180 degrees
	3	Circle/ellipse rotated by 270 degrees
t	num	Line thickness in mm (min. 0.05 mm)
w	num	Width of the circle/ellipse in mm (min. 2.0 mm)
h	num	Height of the circle/ellipse in mm (min. 2.0 mm)

‖ If w and h have the same value, a circle is printed. ‖



Example

Ellipse in normal writing direction, line thickness 2 mm, 25 mm wide and 15 mm high.

```
#YE/0/2/25/15
```

#YG - Print graphics

The #YG command prints a graphics file.
Supported graphics formats: BMP, PCX, JPG, TIF, GIF

Coloured and gray scaled graphics are converted automatically to black and white, using an error diffusion algorithm. This may be very consuming regarding both, the processor capacity as well as the memory capacity. Therefore, it is strongly recommended to use black and white graphics.

Memory area for loading and converting of graphics files can be allocated using the parameter SYSTEM PARAMETER > Free store size or System > Memory > Free store size . For information on the current memory partitioning refer to info-printout PRINT INFO > Memory Status or Info > Status Printouts > Memory Status.

Syntax

```
#YG/djg/vo/a/Datei#G
```

Parameter	Value	Description
d	0	Rotating direction 0°
	1	Rotating direction 90°
	2	Rotating direction 180°
	3	Rotating direction 270°
j	M	*Centered*: The writing command (#T/#J) refers to the center of the graphics field. The field is printed on both sides of the print position.
	R	*Right justified*: The writing command (#T/#J) refers to the right margin of the graphics field. The field will be printed to the left side.
g	D	Graphics field consists of a variable field. Definition equals #YT command.
v	+	Increment – the offset is added to the text field
	-	Decrement – the offset is subtracted from the text field
o	int	Offset, which is added to (incremented) or subtracted from (decremented) the file name – depending on the polarity
a	int	No. of labels with constant number [1...255]
Datei		File name (including drive letter and path in capital letters) of the graphics file which is ought to be printed. If no path is indicated, the file will be searched in `C:\graphics` (on the external memory medium)

Example

Graphics turned 180°. The file must be stored in the `C:\graphics` directory (on the memory card).

```
#YG/2///EXAMPLE.BMP
```

Related reference

File operations on page 139
Description of the correct path specification for file operations.

NOVEXX SOLUTIONS

#YI - Write logo in EPT format directly into the image buffer

The #YI command writes the following data directly into the image buffer in a predefined position.

Syntax

```
#YI/s/s/s.../s#G
```

The command must stand *between* #ER and #Q!

The command must be closed with #G!

As the logo information is only stored in the image buffer, it will be lost if shifted beyond any layout border.

With firmware x.33 or higher, this command is also supported for multi web layouts.

Parameter	Value	Description
s	hex	Coding of a dot line in the logo matrix, hexadecimal with four dots respectively from left to right. • Assignment for hexadecimal coding: • – 1 = Dot is printed – 0 = Dot is not printed • Non-set dots at the end of the line can be left out.. • Coding of each dot line in the logo matrix using an s-parameter. Sequence: from the bottom (Line 1) to the top. Only capital letters and numbers may be used for hex. coding.

Example

Write the Logo with the following construction directly into the image buffer:

Line	Dot line	Hex
4	`1111 1111 1111`	`/FFF`
3	`1111 0000 1111 11`	`/F0FB`
2	`1110 0000`	`/E0`
1	`1100 0000 0011 0101`	`/C035`

```
#YI/C035/E0/F0FB/FFF#G
```

Related reference

Logos on page 140
A logo is an image made of black and white points that is used as part of a label layout.

#YIB - Write logo with binary data directly into image buffer

The #YIB command writes the binary data following the command directly into the image buffer in a predefined position.The data is represented directly by the 256 ASCII characters, i.e. 8 dots correspond to a byte which is to be transmitted

Syntax

```
#YIBc/d/bbb...b
```

The command must stand *between* #ER and #Q!

With firmware x.33 or higher, this command is also supported for multi web layouts.

As the logo information is only stored in the image buffer, it will be lost if shifted beyond any layout border.

Parameter	Value	Description
c	int	Number of dot lines in the logo matrix. Maximum number of lines: 65535
d	int	Number of bytes per line. Maximum: 65535 All the lines must be of the same length
b	bin	Parameter for one byte. Each byte encodes eight dots. Assignment for binary coding: • 1 = Dot is printed • 0 = Dot is not printed

Example

This command is used to write 3 lines each with 4 bytes (corresponding to 4 * 8 = 32 dots per line) directly into the image buffer

```
Line 3: 01111100 10011100 10111100 11011100

Line 2: 01111011 10011011 10111011 11011011

Line 1: 01111010 10011010 10111010 11011010
```

```
#YIB3/4/zÜ¦+{ø+_|£+_
```

Related reference

Logos on page 140
A logo is an image made of black and white points that is used as part of a label layout.

#YIR - Write logo in RLE format directly into image buffer

The #YIR command writes the following data directly into the image buffer in a predefined position. The data is coded in RLE format (Run Length Format)

Syntax

```
#YIRc/sss...s
```

> The command must stand *between* #ER and #Q!
>
> As the logo information is only stored in the image buffer, it will be lost if shifted beyond any layout border.

Parameter	Value	Description
c	int	Number of dot lines in the logo matrix. Maximum number of lines: 65535.
s	hex	Coding of a dot line in the logo matrix. The parameter s consists of a sequence of hexadecimal digits (hex numbers). Permitted value range for each of the hex numbers: 00 to FD. The values FE and FF are reserved for control characters as described below.

- *Start code*

  The start code must stand at the beginning of the data for a dot line. A dot line which is to be printed several times in succession only needs to be entered once with the start code FF. The number after FF shows how often the line is to be repeated. Example: the start code FF04 produces four identical lines.

  – FE (hex): Start code for lines which are to be printed once.

  – FF (hex): Start code for lines which are to be printed repeatedly.
- *End code*
  The last dot line in a logo matrix must be closed with the end code FE.
- *RLE format*

  The decisive factor when converting binary code into RLE format is the changing from ones to zeros and vice-versa. In order to convert binary code into RLE format, it is only necessary to count the sequential zeros and ones between the start and end code and to note them as hex numbers. Single figure hex numbers are prefixed with a zero.

Binary code:	`0001111001111111`
RLE-Code:	`FE03040207FE` (Start code FE, 03 zeros, 04 ones, 02 zeros, 07 ones, end code FE if no other dot lines follow)

Table 16: Example: RLE format

> To enter the Code in RLE-Format as shown in the example, you need to a) use an editor providing a hex-mode option and b) enable the hex-mode. Using an editor without hex-mode would require to enter the corresponding ASCII-character to every hex number, for example ♥ instead of 03.

- *Dot line*

  The actual data of a dot line is contained in pairs of hex numbers between the start and end code. The first hex number of the pair gives the number of consecutive zeros, the second hex number gives the number of consecutive ones.

Example: `05FC` (05 zeros, 252 ones)

If the number of zeros or ones in the binary code exceeds 253, a new pair of hex numbers must be started (the correspondences of the numbers 254 and 255 are the start codes FE and FF).

Binary code:	`262 ones, 3 zeros, 1 one`
RLE code:	`FE00FD00090301` (start code FE, 00 zeros, FD ones, 00 zeros, 09 ones, 03 zeros, 01 ones)

Table 17: Example: Dot line

The 262 ones are divided into two value pairs because there are more than 253: 00FD and 0009. As the binary code does not have any zeros at the beginning, the first byte in the first and second value pair is a zero.

- *Line length*

  RLE lines can be of any length.

- *Blank lines*
  A single blank line has the form FE0000.

  Several blank lines are represented as follows: start code FF, followed by the number of required blank lines.

  Example: `FF04` (04 blank lines in sequence); `FF01 = FE0000` (one blank line)

  > No zeros are required after FFxx (with xx blank lines). The start code for the next line can follow directly afterwards.

Example

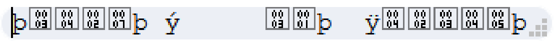
With this command 7 lines are written directly into the image buffer:

- Line 7: 00111000011111
- Line 6: 00111000011111
- Line 5: 00111000011111
- Line 4: 00111000011111
- Line 3: blank line
- Line 2: 111...(262 times)...1110001
- Line 1: 0001111001111111

Command line:

```
#YIR7/FE03040207FE00FD00090301FE0000FF0402030405FE
```

> BEWARE! - This example line cannot be copied from the pdf to an Easy Plug format. Before you can use this command line, you have to convert the data behind the "/", which is 2-Byte ASCII-Hex code, to 1-Byte ASCII characters. The result looks like shown below:

#YIR7/ 

Related reference

Logos on page 140
A logo is an image made of black and white points that is used as part of a label layout.

#YK - Logo definition

The #YK command is used to print a logo. The logo must be loaded in the memory first.
The logo is a data block and is positioned in the label according to a predefined reference point (#T and #J command).

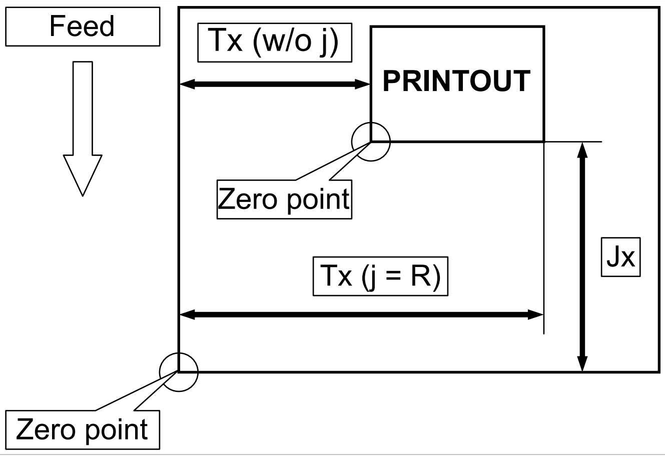
Syntax

```
#YKn/djm
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
n	int	Logo number [0...255]

d	0	Logo in normal write direction
	1	Logo rotated by 90 degrees
	2	Logo rotated by 180 degrees
	3	Logo rotated by 270 degrees
j	M	*Middle*: The command #T refers to the centre of the field. The logo is hereby created on both sides.
	R	*Right justified*: The command #T refers to the right edge of the field. The logo is built up to the left.
m	blank	The logo is expected on the internal RAM disc (default setting).
	A	The logo is expected on the internal RAM disc (default setting).
	C	The logo is expected on an external memory medium.



Example

Logo number 10 is printed, if available.

```
#YK10/0
```

Related reference

Logos on page 140
A logo is an image made of black and white points that is used as part of a label layout.

#YL - Line definition

The #YL command is used to define lines of varying lengths and thicknesses.

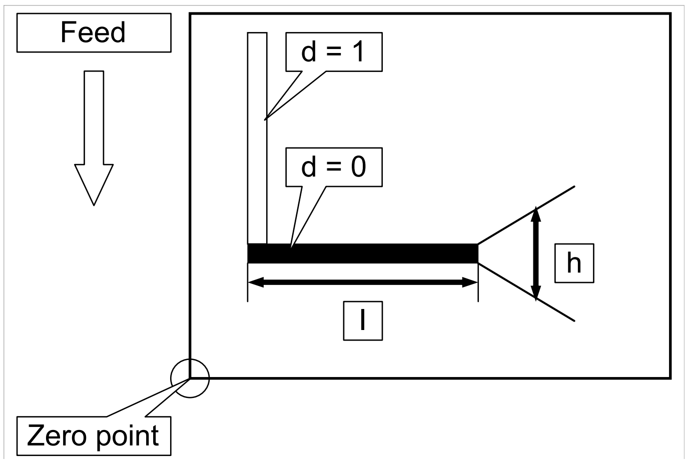
Syntax

```
#YLa/drz/h/l
```

The command must stand *between* #ER and #Q!

If lines are required as a background, they must be defined before the foreground.

Parameter	Value	Description
a	int	Line type number (for details click link under „Related Information“)
d	0	Line in normal write direction
	1	Line rotated by 90 degrees
	2	Line rotated by 180 degrees
	3	Line rotated by 270 degrees
r	R	The line is recreated for each label (e.g. as the background of a number field). Don´t use parameter r together with inverse fields! – otherwise the field contents will be overprinted by the line.
z	P	Normal black printout
	A	“White printout”, that is the printing is left blank. Requires a dark background
	E	Printout with inverted bitimage (black is left blank; white is printed black)
h	num	Line thickness in mm
l	num	Line length in mm



Examples

Line type 1, 0 degree rotation, 3 mm thick, 20 mm long:

```
#YL1/0/3/20
```

Line type 4, 270 degree rotation, 1 mm thick, 50 mm long

```
#YL4/3/1/50
```

Related reference

Printer-internal fonts and line styles on page 148

#YN - Text field

The #YN command prints text with scalable fonts.
The command #YN defines the size and font of a text, as well as the direction of rotation and consecutive numbering.

The following must be taken into consideration when using consecutive numbers:

- A sufficient number of positions must be defined corresponding to the biggest number to be used. Leading zeros are printed.
- Only numerical characters are incremented or decremented.
- All numerical characters within a text string are incremented or decremented.

Syntax

```
#YNz/dbqmnjwcghrkp/e/vop/a/TEXT#G
```

The command must stand *between* #ER and #Q!

Parameter	Value	Description
z	int	Font number (for details click link under „Related Information“) The fonts 100 and 101 are internal scalable fonts and are therefore always available. Alternatively, fonts can be loaded from an external memory medium (for details read the „External Flash Memory Handbook“).
d	0	Normal write direction
	1	Text rotated by 90 degrees
	2	Text rotated by 180 degrees
	3	Text rotated by 270 degrees
b	A	Text printed inversely A dark background must first be applied to the print area (line or rectangle).
q	E	Existing Bitimage is inverted (see example 1)
m	X	Text field consists of one variable data field
n	Y	Suppression of leading zeros in counting fields. The remaining digits are printed at the same position as with leading zeros (see example 2).
j	M	*Centered*: The write command (#T/#J) refers to the centre of the text field. The field is built up on both sides of the print position.

	R	*Right justified*: The write command (#T/#J) refers to the right edge of the text field. The field is built up to the left.
f	F	Field designation for changing data (see description of changing data)
w	W	Counter field TEXT is incremented/decremented without a carry over, i.e. only the first unit position of the figure is increased/decreased.
c	Dint	Rotation of TEXT in steps of one degree [0...359]°; Default setting: 0° The character D has to be succeeded without blank by the value. Parameter c takes priority over d.
g	O	Default type
	B	Bold
h	V	Default type
	I	Italics
r	blank	Application of scalable fonts without Unicode (default setting)
	K	
	U	Application of Unicode fonts
	G	Application of arabic Unicode fonts (characters will be printed right to left) Each Unicode-character has to be referred to in the TEXT-field as follows: \u0041 (example with 0041 being the index of the character "A"). There are some specific features to bear in mind, when using Unicode fonts!
k	Snum	Fixed distance in mm between all characters in TEXT. The character S has to be succeeded without blank by the value.
p	N	Standard character-algorithm is applied (default setting).
	Q	2 D character algorithm is used. May improve the exactness of small characters. (e.g. printing of small chinese characters). This algorithm is about 30-40% slower than the standard algorithm.

e	int	Font size [15...6000] Dot, depending on the font and its installed size, respectively. Optionally can be defined different font sizes for x- and y-direction (see example 3).

v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]

a	int	No. of labels with constant No. [1...255].

TEXT		Any alphanumerical text, whereby only numerical values are taken into account during offset. All numerical values of the text string are taken into account for the offset (e.g. 10A3B56 = 10356). Max. number of characters: 255. The text field may also be a variable data field. Precondition: the X-flag is set. The text field may contain an input field.

Examples

Example 1: Inverted Bitimage

```
#!A1
#ER
#T5#J5#YL0/0/2.5/90#G
#G -----------------------------------------------------------------
#G The part of the text which overlaps the line is printed in white,
#G the rest in black.
#G -----------------------------------------------------------------
#T5#J5#YN100/0E////Example#G
#Q1/
```

Example 2: Suppressing leading zeros

```
#G -----------------------------------------------------------------
#G Printout: Test  1, Test  2, …, Test 10, etc.
#G -----------------------------------------------------------------
#YN100/0Y/1/1//Test000#G
```

Example 3: Scaling speedo fonts

```
#G -----------------------------------------------------------------
#G The text „Speedo Test“ has font size 120 in y-direction and
#G font size 99 in x-direction.
#G -----------------------------------------------------------------
#YN100/0/120X99///Speedo Test#G
```

Example 4: Scaling speedo fonts

```
#G -----------------------------------------------------------------
#G Font 100 in size 10 points.  The 4-figure number is incremented on
#G every 5th label by the figure "1" (5 x 0001, 5 x 0002).
#G -----------------------------------------------------------------
#YN100/0/10/1/5/0001#G
```

Example 5: Printing Unicode characters

```
#G -----------------------------------------------------------------
#G Printout:
#G „Example printout of the character „A“ using a Unicode font: A”
#G -----------------------------------------------------------------
#!A1
#G ---------------------------------------------------------------
#G Endles material, 100 mm wide, 30 mm long
#G ---------------------------------------------------------------
#!IMN100/30 #ERN #T5#J10
#G ---------------------------------------------------------------
#G Using font 902; printout in normal writing direction; Unicode;
#G Font  size 60 dots („\u0041“ is the Unicode for „A“)
#G ---------------------------------------------------------------
#YN902/0U/60///Example printout of the character „A“ using a Unicode
 font:: \u0041#G
#Q1#G
```

Example 6:

```
#G ------------------------------------------------------------------
#G This  Easy Plug program generates the printing pictured in the
#G figure below. The printing includes four 2dimensional bar codes and
#G a 35-degrees-rotation.
#G ------------------------------------------------------------------
#!A1
#G ----------------------------------------------------------------
#G Punched material, 100 mm wide, 80 mm long
#G ----------------------------------------------------------------
#IMS100/80
#ER
#J60#T15#YN100/0D15/60///Testfile "64-2DIM" 15 Degrees#G
#J48#T5#YN100/0/90///2D Barcodes as standard#G
#J10#T55#MXC4/0/2/3///1234567890abcdefghijklmnopqrstuivwxyz#G
#J10#T30#PDF0/T0/1/0/0/2/2/1234567890ABCDEFG#G
#J10#T5#IDM/0/10///1234567890abc#G
#J7#T5
#YT105/0///IDM (Data Matrix Code)   PDF-417   MXC (Maxi Code)#G
#J30#T5
#CBF/0/2/2/8/0///CODABLOCK F 3456789012345678901004Odigit#G
#J26#T8#YT105/0///CODABLOCK F#G
#Q1/
```

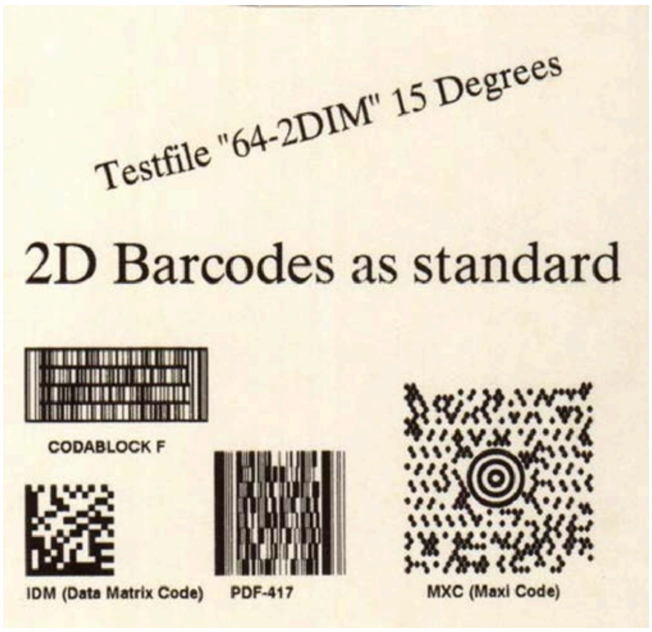


Fig. 21: Printout of example 6.

Example 7:

```
#G ------------------------------------------------------------------
#G Prints the chinese character (see fig. below) with the Unicode
#G u50D1.
#G ------------------------------------------------------------------
#!A1
#IMN100/100/
#ERN
#T10#J10
#YN902/0U/400///\u50D1#G
#G -------------------------------------------
#G „\u50D1“ is the Unicode of the printed character
#G -------------------------------------------
```

```
#Q1#G
```

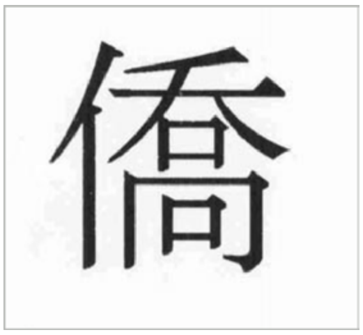


Fig. 22: Printout of example 7.

Related concepts

Unicode on page 163

Related reference

Input Fields on page 10
Variable Data Fields on page 9
Printer-internal fonts and line styles on page 148
Arabic glyphs on page 167

#YR - Rectangle definition

The #YR command defines rectangles of varying sizes and line thicknesses.

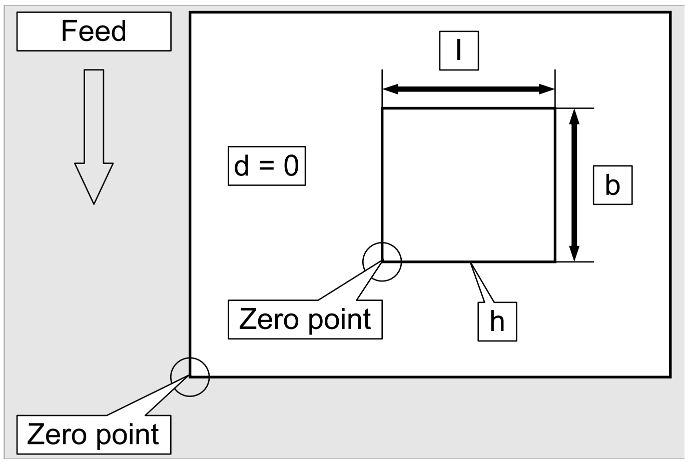
Syntax

```
#YRa/dr/h/l/b
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
a	int	Line type number (for details click the link below under „Related Information“)
d	0	Rectangle in normal write direction
	1	Rectangle rotated by 90 degrees ‖ The rectangle rotates around the reference point. ‖
	2	Rectangle rotated by 180 degrees
	3	Rectangle rotated by 270 degrees
r	R	The rectangle is recreated for each label (e.g. as the background of a number field). ‖ If rectangles are required as a background, they must be defined before the foreground. ‖
h	num	Line thickness in mm

l	num	Width of the rectangle in mm
b	num	Height of the rectangle in mm



Example

Rectangle line type 0, 0 degree rotation, line thickness 2 mm, 15 mm wide, 25 mm high.

```
#YR0/0/2/15/25
```

Related reference

Printer-internal fonts and line styles on page 148

#YS - Real time as bar code

The #YS command defines a time in text format the form of a bar code.

Syntax

```
#YSz/dkbcujlrm/h/s/o/ TIMETEXT#G
```

The printer must be equipped with a real-time clock.

The command must stand *between* #ER and #Q!

Parameter	Value	Description
z	int	Bar code number (for details click on link below under „Related Information“)
d	0	Bar code in normal write direction
	1	Bar code rotated by 90 degrees
	2	Bar code rotated by 180 degrees
	3	Bar code rotated by 270 degrees
k	M	Bar code with plain-copy line
	O	Bar code without plain-copy line

b	A	Position change of the plain-copy line. If the plain-copy line is normally printed below the bar code, it is shifted by this option to the top of the bar code, and the other way round.
c	C	A check digit according to Module 10 is calculated and printed.
u	blank	(Default setting) No update. The time is read into the variable once. This is done during the compilation of the layout.
	W	
	U	The time is read repeatedly during the running print job. This happens immediately after printing the relevant output field (#VW…), but counts only for the same field on the next label. The setting is not suitable for dispensers. Reason: If the next label is printed considerably later (e.g. in single-start mode with footswitch), the following happens: The output field on label 2 contains the time, at which the same field on label 1 was printed. Example: Label 1 is printed at 10.50 h, label 2 at 11.03 h. In this case, the imprint of label 2 shows the time 10.50!
	Y	(Recommended for dispensers) The time is read repeatedly during the printing of a label (approx. all 5 seconds).
j	Z	Field position centered, related to the print position.
	R	Field position right-justified, related to the print position. The field is built up starting at the print position to the left side.
	blank	(Default setting) Field position left-justified, related to the print position. The field is built up starting at the print position to the right side.
l	G	Plain copy line default position
	H	Plain copy line justified
	I	Plain copy line left-justified
	K	Plain copy line centered
	L	Plain copy line right-justified
r	Pnum	Ratio of the bar code [2.0...3.0]. The default setting is 2.5. The letter P (proportion) must stand immediately in front of the ratio (e.g. P2.5). A ratio without the letter P is invalid. Observe the sequence: The ratio must be entered after the parameter d (write direction)!
m	B	EAN/UCC mode with brackets around the data designator. The data have to be sent in brackets! The brackets appear in the plain copy line but not as bar code.
	X	EAN/UCC mode without brackets around the data designator. Data have to be sent without brackets!

h	int	Bar code height: 0 = 5 mm, 1 = 10 mm, ... from 10 = actual height in mm (e. g. 35 = 35 mm high). This value minus 1 is multiplied by the value in parameter BCHI set at the printer.

s	int	Bar code width [1...16]

o	blank	The current date is printed.
	0	
	int	*Days offset*: The current date is calculated forward by the entered number of days [1...65000].

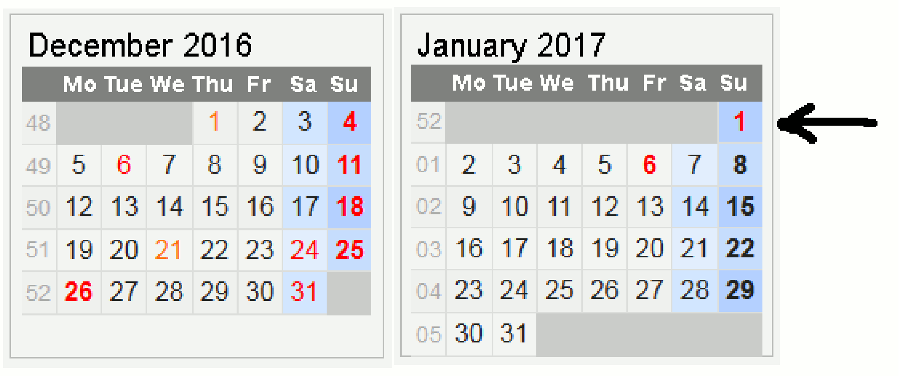
	Mint	*Months offset*: The current date is calculated forward by the entered number of months [1...65000]. ‖ Adding the letter M to the integer value changes the offset from day to month. ‖
	Pint	*Minutes offset*: The current time is calculated forward by the entered number of minutes [1...65000].
	Hint	*Hours offset*: The current time is calculated forward by the entered number of hours [1...65000].

TIMETEXT		String containing Time and Text (ASCII-characters) in any order. Times are marked with the control character (^). Insert required Text as ASCII characters (without control characters) in the TIMETEXT-string.

Time formats:

^z	1/100 seconds
^s	Seconds
^m	Minutes
^h	Hours
^D	Day of the month
^d	Day of the year
^W	3-figure day (e.g. 001 for the 1st day of the year)
^w	Day of week (1-figure number of the day of week, e. g. 1 for Monday)
^M	Month
^Y	Year, two digit
^R	Year, four digit
^G	3-figure month, German
^E	3-figure month, English
^S	Name of a month from command #DM
^C	Week of the year always with two digits
^c	Week of the year without leading 0
^K	Corresponding year to the week (of the year) with 4 digits
^k	Corresponding year to the week (of the year) with 2 digits

‖ Note for ^K and ^k: The year is calculated from the week of the year. Due to the fact that the year often starts in the middle of the week, the first days may count to the last week of the year of the preceding year (see fig. below). ‖



December 2016

	Mo	Tue	We	Thu	Fr	Sa	Su
48				1	2	3	4
49	5	6	7	8	9	10	11
50	12	13	14	15	16	17	18
51	19	20	21	22	23	24	25
52	26	27	28	29	30	31	

January 2017

	Mo	Tue	We	Thu	Fr	Sa	Su
52							1
01	2	3	4	5	6	7	8
02	9	10	11	12	13	14	15
03	16	17	18	19	20	21	22
04	23	24	25	26	27	28	29
05	30	31					

Fig. 23: Example: The year for the 1.1.2017 would here be 2016, because the date lies in week 52.

Example

Printout which is updated during the current print job: Code 128 with plain-copy line "23-Nov-1998", normal print direction, 10 mm high, width 2.

```
#YS13/0MU/2/2//^D-^G-^R
```

Related reference

Printer-internal bar codes on page 144

#YT - Text field definition

The #YT command is used for (optional) parameter settings and entering text.

Syntax

```
#YTz/dqgnwjk/vop/a/TEXT#G
```

‖ The command must stand *between* #ER and #Q! ‖

Parameter	Value	Description
z	int	Font number (for details click link below under „Related Information“). ‖ In case of an invalid font number, the text is printed with YT100. ‖

d	0	Normal write direction
	1	Text rotated by 90 degrees
	2	Text rotated by 180 degrees
	3	Text rotated by 270 degrees
q	P	Normal black printout
	A	“White printout”, that is the printing is left blank. Requires a dark background
	E	Printout with inverted bitimage (black is left blank; white is printed black)
g	D	Text field consists of one variable data field

n	Y	Suppresses leading zeros in counter fields. The remaining digits are printed at the same position they were printed at with leading zeros. See example 2.
w	W	Counter field TEXT is incremented/decremented without a carry over, i.e. only the first unit position of the figure is increased/decreased.
j	M	*Centered*: The write command (#T / #J) refers to the centre of the text field. The field is built up on both sides of the print position.
	R	*Right justified*: The write command (#T / #J) refers to the right edge of the text field. The field is built up to the left.
k	Snum	Fixed distance in mm between all characters in TEXT. The character S has to be succeeded without blank by the value.

v	+	Increment – offset is added to TEXT
	-	Decrement – offset is subtracted of TEXT
o	int	Offset, which is added to or subtracted of TEXT, depending on the leading sign
p		Base designator that defines the number base for the offset. A missing base designator automatically switches the number base to „decimal“.
	B	Binary [01]
	O	Octal [01234567]
	D	Decimal [0123456789]
	H	Hexadecimal [0123456789ABCDEF]

a	int	No. of labels with constant No. [1...255].

TEXT		Any alphanumerical text, whereby only numerical values are taken into account during offset. All numerical values in the text string are taken into account for the offset (e.g. 10A3B56 = 10356). Maximum number of characters: 255. The text field may also be a variable data field. Precondition: the D-flag must be set. The text field may contain an input field.

Examples

- *Example 1*: Inverted Bitimage:

```
#!A1
#ER
#T5#J5#YL0/0/2.5/90#G
#T5#J5#YT109/0E///Example#G
#Q1/
```

Printout: The part of the text which overlaps the line is printed in white, the rest in black

- *Example 2*: Suppression of leading zeroes:

```
#YT106/0Y/1/1/Test000#G
```

Printout: Test 1, Test 2, …, Test 10, etc.

- *Example 3*: Font 100, without parameter:

```
#YT100/0///DEMO#G
```

- *Example 4*: Font 101, italic text and rotated by 180 degrees.

```
#YT101/2K///DEMO#G
```

- *Example 5*: Font 100, text (10) is increased by 1 every 5 labels (DEMO-10, #G DEMO-11, DEMO-12 etc.).

```
#YT100/0/1/5/DEMO-10#G
```

- Example 6:

```
#YT109/0///T#G#YT103/0///ext#G
```

Printout: Text

- *Example 7*: Font 104 with variable data field Number 00 with 15 characters.

```
#YT104/0D///$00,15#G
```

- *Example 8*: Counter with decimal number base. Offset: +1110

Command	Result
`#YT107/0/+1110/1/12-O.17^T#G`	12-O.17^T 23-O.27^T 34-O.37^T ...

- *Example 9*: Counter with hexadecimal number base. Offset: -1H

Command	Result
`#YT107/0/-1H/1/0Kbf0#G`	0Kbf0 FKbfF FKbfE ...

- *Example 10*: Counter with binary number base. Offset: +1B

Command	Result
`#YT107/0/+1B/1/0000000#G`	0000000 0000001 0000010 ...

- *Example 11*: Counter with octal number base. Offset: +1O

Command	Result
`#YT107/0/+1O/1/0006#G`	0006 0007 0010 ...

Related reference

Input Fields on page 10
Variable Data Fields on page 9
Printer-internal fonts and line styles on page 148

#YV - Variable data field

The #YV command is used for filling a field with variable data. Field types for variable data are text fields and bar code fields. The variable data field must be made known to the printer before the command #YV is used (see #YT, #YB).

> To draw a maximum benefit of the printers performance, print jobs should send the variable data for the #YV field in the same order in which the fields are printed. Thus, the fields can be updated while printing. Otherwise, the printer has to wait with updating, until the respective #YV field is printed, what can slow down the throughput!
>
> Variable data fieds may not be combined with counting fields!

Syntax

```
#YVzn/TEXT#G
```

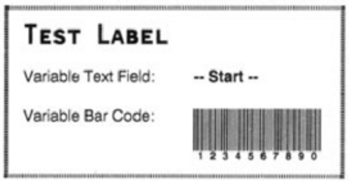
Parameter	Value	Description
z	int	Data field number [0...999]
n	blank	Trailing blanks in TEXT are deleted.
	B	Trailing blanks in TEXT are *not* deleted.

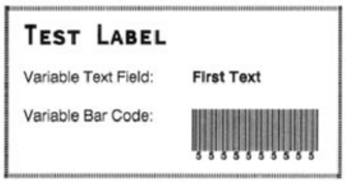
TEXT		Variable data (1 to 255 characters)

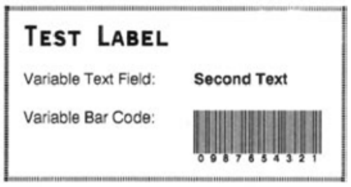
Examples

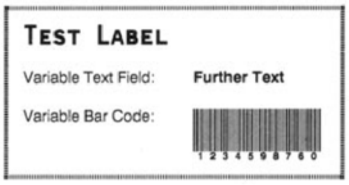
Command sequence	Printout

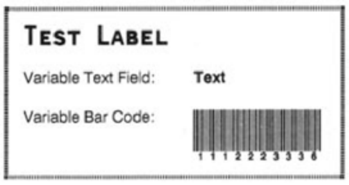
`#!A1`
`#IMS95/50`
`#ERN`
`#T5#J5`
`#YR3//1/90/45`
`#T10#J40`
`#YT109////Test Label#G`
`#T10#J30`
`#YT104////Variable Text Field:#G`
`#T10#J20`
`#YT104////Variable Bar Code:#G`
`#T55#J30`
`#YT107/D0///$00,15`
`#T55#J10`
`#YB6/D0/10/3///$01,10`	

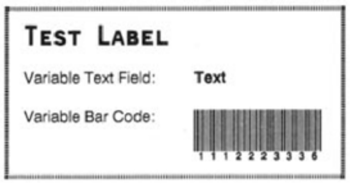
`#YV00/-- Start --#G`
`#YV01/1234567890#G`
`#Q1#G`
	 Test Label
	Variable Text Field:
	-- Start --
	Variable Bar Code:
	1 2 3 4 5 6 7 8 9 0

`#YV00/First Text#G`
`#YV01/5555555555#G`
`#Q1#G`
	 Test Label
	Variable Text Field:
	First Text
	Variable Bar Code:
	5 5 5 5 5 5 5 5 5 5

`#YV00/Second Text#G`
`#YV01/0987654321#G`
`#Q1#G`
	 Test Label
	Variable Text Field:
	Second Text
	Variable Bar Code:
	0 9 8 7 6 5 4 3 2 1

`#YV00/Further Text#G`
`#YV01/1234598760#G`
`#Q1#G`
	 Test Label
	Variable Text Field:
	Further Text
	Variable Bar Code:
	1 2 3 4 5 9 8 7 6 0

`#YV00/Text#G`
`#YV01/1112223336#G`
`#Q2#G`
	 Test Label
	Variable Text Field:
	Text
	Variable Bar Code:
	1 1 1 2 2 2 3 3 3 6

Command sequence	**Printout**
	 TEST LABEL Variable Text Field: Text Variable Bar Code: 1 1 1 2 2 2 3 3 3 6

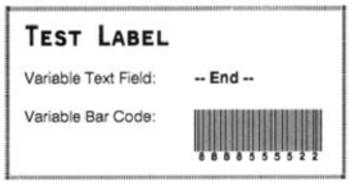
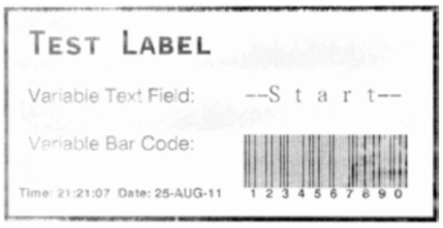
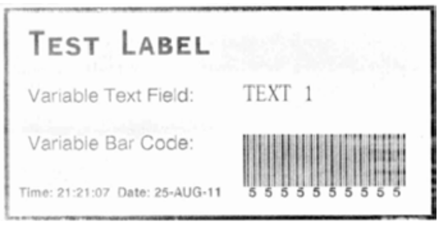
`#YV00/-- End --#G`
`#YV01/8888555522#G`
`#Q1#G`
	 TEST LABEL
	Variable Text Field: -- End --
	Variable Bar Code: 8 8 8 8 5 5 5 5 2 2

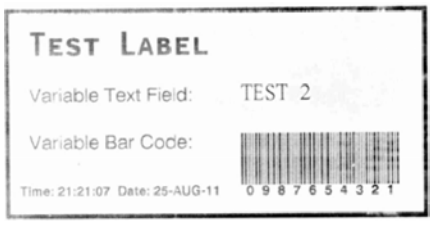
Table 18: Example with #YT

Command sequence	**Printout**

`#!A1`
`#IMN95/50`
`#ER`
`#T5 #J5`
`#YR0//1/90/45`
`#T8 #J10`
`#YC106/0//Time: ^h:^m:^s  Date: ^D-^G-^R#G`
`#T10#J40`
`#YT109////Test Label#G #T10#J30`
`#YT104////Variable Text Field:#G`
`#T10#J20 #YT104////Variable Bar Code:#G`
`#T55#J30 #YN902/0X/60///$00,15`
`#T55#J10 #YB6/D0/10/3///$01,10`	

`#YV00/--S t a r t--#G`
`#YV01/1234567890#G`
`#Q1#G`
	 TEST LABEL
	Variable Text Field: --S t a r t--
	Variable Bar Code: 1 2 3 4 5 6 7 8 9 0
	Time: 21:21:07 Date: 25-AUG-11

`#YV00/TEXT 1#G`
`#YV01/5555555555#G`
`#Q1#G`
	 TEST LABEL
	Variable Text Field: TEXT 1
	Variable Bar Code: 5 5 5 5 5 5 5 5 5 5
	Time: 21:21:07 Date: 25-AUG-11

`#YV00/TEST 2#G`
`#YV01/0987654321#G`
`#Q1#G`
	 TEST LABEL
	Variable Text Field: TEST 2
	Variable Bar Code: 0 9 8 7 6 5 4 3 2 1
	Time: 21:21:07 Date: 25-AUG-11

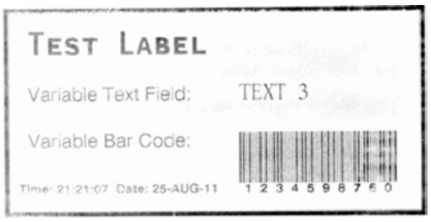
Command sequence	Printout
`#YV00/TEXT 3#G` `#YV01/1234598760#G` `#Q1#G`	 TEST LABEL Variable Text Field: TEXT 3 Variable Bar Code: 1 2 3 4 5 9 8 7 6 0 Time: 21:21:07 Date: 25-AUG-11

Table 19: Example with #YN

Related reference

Variable Data Fields on page 9

Z

#Z - Mirroring

The #Z command creates mirror images of texts, bar codes, logos, lines and rectangles.
All fields which stand before the command are mirrored.

The mirror axis is the centre axis of the label in relation to the label width given in the command #IM.

> The label width indicated in the #IM command is rounded up to a multiple of 2.66 mm (32/12 mm).
> The mirroring axle is the center of the label related to the rounded up label width.

Syntax

```
#Z
```

> The command must stand *between* #ER and #Q!
>
> Misprints occur with update fields such as number field, clock, line repeat and Easyline!
>
> If the example print job is printed repeatedly in standalone mode, only the first print job will contain mirrored text. The following print jobs will show the text unmirrored.

Example

```
#!A1
#IMN64/100
#ERN
#G --------------------------------------
#G Mirrored text
#G --------------------------------------
#T5#J5#YT106////Spiegel
#Z
#G --------------------------------------
#G Normal text
#G --------------------------------------
#T20#J50#YT107////Normal print
#Q10/
```

Appendix

OVERVIEW OF COMMAND GROUPS

General commands

Command	Designation
#BOF	Spooler buffer off
#BON	Spooler buffer on
#CIM	Cut
#EMU	300 dpi emulation for 600 dpi print head
#FC	Material feed with cut
#FF	Material feed
#G	Command end
#HP	Set print head contact pressure
#HV	Set print head temperature
#ME	Material ejection
#PC	Set parameter
#PR	Print speed
#RTC	Setting the realtime clock
#RX	Gap detection selection

Download commands

Command	Designation
#DC	Delete all download logos
#DF	Download a file
#DK	Download a logo
#DM	Download the name of a month
#DO	Delete a download logo

Logo commands

Command	Designation
#DC	Delete all download logos
#DK	Downloading a logo
#DO	Deleting one download logo
#YI	Write logo in EPT format directly into the image buffer

Command	Designation
#YIB	Write logo with binary data directly into the image buffer
#YIR	Write logo with RLE format directly into the image buffer
#YK	Logo definition

Print job commands

Command	Designation
#BR	Interrupt print job
#ER	Start label format
#FO	Read in Easy Plug file
#IM	Material information
#N	Nationality of character set
#PA	Offset print start
#PO	Gap offset
#Q	Print quantity
#S	Dispensing mode/dispensing position

Format commands

Command	Designation
#CBF	Bar code Codabar F
#CG	Code 49
#CFN	Adjust intercharacter gap
#IDM	Data Matrix Code
#J	Vertical print position
#M	Magnification factor
#MXC	Bar code MaxiCode
#PDF	Bar code PDF 417
#R	X/Y Offset data blocks
#RSS	GS1 DataBar & CC bar codes
#SQR	QR Matrix Code
#T	Horizontal print position
#YB	Define bar code
#YC	Real time as Text
#YG	Print graphics
#YI	Write logo in EPT format directly into the image buffer
#YIB	Write logo with binary data directly into the image buffer

Command	Designation
#YIR	Write logo in RLE format directly into the image buffer
#YK	Define logo
#YL	Define line
#YN	Text field (font scale option)
#YR	Define rectangle
#YS	Real time as bar code
#YT	Define text field
#YV	Variable data field
#Z	Mirroring

Format commands define fonts, bar codes, logos, lines and rectangles with regards to their type, size, direction of rotation and their position in the label based on fields. It is also possible to offset all fields together.

Format commands are managed in the memory and are only carried out after the format (#Qn/) has been correctly terminated. Easy Plug is only able to prepare one format for printing at a time, but it can following store subsequent formats in the queue.

When the entered number of labels has been processed, this format is deleted (together with the change and cutter information) and processing of the next format begins.

A format remains activated once it has been started. Deselecting the device only interrupts the processing of a label series which has been started.

Label zero-position	The bottom left-hand corner of the label has the horizontal (X) position 0 (zero) and the vertical (Y) position 0 (zero).
Field position	The commands #Tx and #Jx define the field position. This refers to the bottom left-hand corner of a field and the left-hand (X) bottom (Y) corner of the label (middle of the gap or the length position on reel material). If the fields are rotated, the field positions remain constant and the print position rotates around the field position.

Variable commands

Command	Designation
#SB	Bar code definition
#SCF	Codablock F definition
#SDM	Data Matrix definition
#SF	Fixfont definition
#SFN	Code49 definition
#SG	Graphic definition
#SMX	Maxicode definition
#SPF	PDF417 definition
#SRS	GS1 DataBar definition

Command	Designation
#SS	Speedo font definition
#SV	Changing the content of a text variable
#VDT	Define text variable
#VDD	Define date and time variable
#VDE	Define expression variable
#VDO	Define variable for OLV data access
#VDP	Define variable for print job data access
#VDS	Define system variable
#VTS	Define standalone variable
#VR	Read data from (RFID) tag
#VW	Drawing/writing on label

To print a text, a bar code or a graphic, the above listed definition commands must be followed by the #VW command, which initiates the printing.

RFID commands

Only applicable with RFID-equipped devices.

Command	Designation
#RT	Read and print RFID data
#RFC	Special RFID commands
#RFH	Request data – send to host
#RFL	Lock/unlock memory areas
#RFR	Read data
#RFW	Write typed data to RF tag
#RM	Measurement
#SI	Write data to interface
#SRF	Definition read/write target
#VR	Read data from RFID tag

Immediate commands

Command	Designation
#!A1	Activate interface
#!CA	Easy Plug, delete everything
#!CF	Delete format
#!D	Triggering a Single-Start
#!HP	Set print head contact pressure

Command	Designation
#!H	Set print head temperature
#!P1	Deactivate interface
#!PC6004	Set start offset
#!PG	Read out parameter setting
#!SP	Stop printer
#!SR	Start printer
#!XB	Status acknowledgement with bar code
#!XC	Status acknowledgement pharmacy code
#!Xn	Status acknowledgement
#!Xn	Bar code acknowledgement
#!XMn	Diagnose Dump / Read machine status
#!XS	Printer status acknowledgement

An immediate command is marked with an exclamation mark "!".

Immediate commands are not temporarily stored in the queue, but are carried out immediately.

All other commands are put on the waiting list until the immediate commands have been carried out, and are then processed in the order in which they have been received.

Immediate commands can not be applied for print jobs, which are processed in standalone mode.

Immediate commands can be implemented at any point – even within a format.

For example, the following process is possible:

1. Send print job A.
2. Send a second print job which is stored in the queue (provided that #BON has been set) while the first print job A is being printed.
3. Interrupt the first series with the immediate command #!CF.

Because the printing process has been halted and the current print series deleted with #!CF (situation such as series ready printed), processing of data collected in the queue continues and the second format processed.

Related reference

Print job structure on page 7

FILE OPERATIONS

Description of the correct path specification for file operations.

Drive names

The following drives letters may be used in path names:

Drive	File access on
A	Printer-internal RAM ‖ Switching the printer off will delete all data saved on the internal RAM! ‖
C	CompactFlash card (standard slot)
D	CompactFlash card (optional 2nd slot)
E	SD card
F	Thumb drive ‖ Drive letter is assigned to the first recognized USB host port. ‖

‖ Standalone operation is only possible with drive C. ‖

Mapping a drive letter: INTERFACE PARA > >DRIVEASSIGNMENT > DRIVE x (x representing the drive letter)

File names

Files which are referred to by an Easy-Plug command must be named according to the following conventions. Those are the same rules which were required by the MS-DOS operating system:

- Maximum file length: 8 characters
- Allowed are alphanumerical characters (20_{hex} up to $7E_{hex}$ on the ASCII table); which are:
  - The letters A-Z without country specific special characters
  - The numbers 0-9
- Not allowed are the following characters: . / : < > + = ; , ? [\ ] " * | and the blank
- The file name may be supplemented by 3 characters. This extension is separated from the file name by a decimal point

Related reference

#DF - Downloading a file on page 28
The command #DF downloads a file from a PC to the printer.

#CF - Clear file on page 25
The command #CF deletes a file store on the RAM disc or on an external memory medium.

#YG - Print graphics on page 113

LOGOS

A logo is an image made of black and white points that is used as part of a label layout.

What is a logo?

In the Easy-Plug context, a logo is an image made of black and white points that is used as part of a label layout. Logos are stored binary coded in the printer memory.

Logo application exists for historical reasons and was in most cases replaced by the use of graphic files. The use of logos continues being appropriate for older printer types, which can not process graphic files.

Structure of a logo

Easy Plug manages a memory area, in which logos defined by the operator can be saved under a number (0 to 255). A logo (text, image) consists of dots arranged in lines and columns. Each dot is one bit. Only dots which are set to 1 are printed, and not the others.

In Easy Plug, this dot matrix is in hexadecimal code as shown below:

- A line in the logo matrix is given in hexadecimal code with 4 dots from left to right.
- Non-set dots at the end of the line can be left out.
- Every line in the logo matrix is coded by one or more parameters per line, from bottom (Line 1) to top.
- Blank lines are defined by 2 slashes (//).
- The maximum size of a logo is only limited by the label layout and the available memory space.

Example

Priority	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Line 8: /C183	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	1
Line 7: /C3C3	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
Line 6: /C7E3	1	1	0	0	0	1	1	1	1	1	1	0	0	0	1	1
Line 5: /CFF3	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1
Line 4: /DFFB	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1
Line 3: /FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Line 2: //	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Line 1: /FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 24: Example of a logo matrix

The same logo matrix abbreviated:

```
Line 8: 1100 0001 1000 0011 (/C183)
Line 7: 1100 0011 1100 0011 (/C3C3)
Line 6: 1100 0111 1110 0011 (/C7E3)
Line 5: 1100 1111 1111 0011 (/CFF3)
Line 4: 1101 1111 1111 1011 (/DFFB)
Line 3: 1111 1111 1111 1111 (/FFFF)
Line 2: 0000 0000 0000 0000 (//)
Line 1: 1111 1111 1111 1111 (/FFFF)
```

Transferring the logo to the printer:

```
#G---------------------------------------------------------------
#G The logo is transmitted and stored as number 2 in the download
#G buffer of the printer.
-----------------------------------------------------------------
#DK2/FFFF//FFFF/DFFB/CFF3/C7E3/C3C3/C183#G
```

Related reference

#DK - Downloading a logo on page 29
The command #DK is used to download a logo (to send a logo from the PC to the printer) under a reference number (0 to max. 255) which must be entered.

#YI - Write logo in EPT format directly into the image buffer on page 114
The #YI command writes the following data directly into the image buffer in a predefined position.

#YIB - Write logo with binary data directly into image buffer on page 115

#YIR - Write logo in RLE format directly into image buffer on page 116

The #YIR command writes the following data directly into the image buffer in a predefined position. The data is coded in RLE format (Run Length Format)

#YK - Logo definition on page 118

UTILITY PROGRAM „MAKE_DF.EXE“

The utility program „Make_DF.exe“ generates a complete #DF-command and writes it to a text file.

Syntax

The #DF command loads data from the host computer to the printer. As a prerequisite, data to be transmitted must be binary.

Execute Make_DF without any parameters to see a list of the required commandline-parameters.

```
MAKE_DF <SourceFilename> <DestinationFilename> <PrinterFilename> <Over►
writeOption[N,O]>
```

Parameter	Description	Example
`<SourceFilename>`	Drive-, path- and filename of the file to be transmitted from PC to printer.	`C:\_temp\testfont.spd`
`<DestinationFilename>`	Drive-, path- and filename of the text file generated by make_df. This file contains – after successful program execution – the #DF command, including all necessary parameters and data.	`C:\_temp\df333.txt`
`<PrinterFilename>`	Drive-, path- and filename of the transmitted file on the printer after successful program execution (see parameter f of the #DFcommand).	`C:\fonts\font333.spd`
`<OverwriteOption[N,O]>`	Option overwrite (O) or not (N) overwrite an already existing file with the specified name (see parameter s of the #DF-command).	`N`

Table 20: Commandline-parameters for MAKE_DF.

Example

Using the example parameters, the program call would look like the following:

```
make_df c:\_temp\testfont.spd c:\_temp\df333.txt c:\fonts\font333.spd N
```

Displayed text lines after the execution:

```
File c:\_temp\DF333.txt created:
Printerfilename: C:\fonts\font333.spd
Filesize:   71680 Bytes
```

The file df333.txt now contains the complete #DF-command including all parameters as well as the binary data of the file to be transmitted. Preceding to #DF is the command #!A1, which tells the printer to understand the following as Easy Plug commands.

Cutout of the generated file:

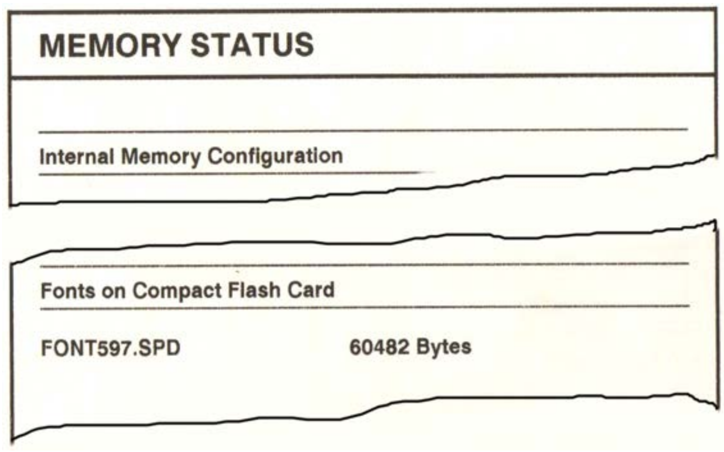
```
#!A1#DF/N/C:\fonts\Font333.spd/78193/D1.0   _1q  _n_z_¤ + _AmericanGara▶
mond                                                         27 Oct 92 BICS
 for Speedo  23 May 88                                           BX00Copy▶
right 1989-1992 by Bitstream Inc. All rights reserved.
   _4_5   __®___hAmericanGaramondBT-Roman        AmericanGaramondRoman
   _è ø_è_ô ø_ïÿYÿ _h_…ÿ• +  ì_ö_# __   é˜ š_T_ _3ÿ˜Q š_]_®_f  _®_f_ __ö
  __ö Ì_\f  _\f_c __°ß ÍÖ .Ñ 6Ñ (Ç _ï_ _Ç _Ç Ä }Ä øÅ kÃ /Á gþ Çÿ gÿ ìü
 Xý Mý óý †ú _ú Úû xû õû éø Õù Èö Ž÷ «ô ×õ Jò _ó Dó íð  ð þð ½ñ {ñ ‹ï í
 qê ,ë nè \æ cç ½ä œâ ‘ã à Âž êŸ Æ˜ °› ÿ˜ R™ _– _“ p• t’ _„ ã˜ òŽ _Œ a˜
 Š Æ‹ Üˆ }‰ q‰_N‰ _‰ Ý† ˜„ x„ Ñ… ¬, ×ƒ ˆ˜ á¿ F¼ A½ ~° ’» É¹ ‘· _· ‘µ O²
 _³ _° _± _® –¯ ¿– lª ä« ß¨ Õ© /© ù¦ 3¦ «¤ Ì¢ É  ê¡ ‰^ ¨\ Š] _]¯ÔZ zZ
 Ì[ ð[ ‘X ÑY ÆW _T ˆU nU 8S MQ ¿O _L ˆJ ÓH 6I ìF ¢D ;E |E ûB wB …C „@ »A
 (A îA ÄA ˜~ c~ =~ _~ ¹~ –˜ _˜ _˜ Å˜ ¸| _| ÿ| Ó| ˜} `} ò} è} Þ} ´} šz `z
 Dz *z _z Òz z œ{ `{ ]{ 4{ /{ _{ á{ Ø{ ³{ ª{ …x |y _v °t _r Øp ^q 9n ×o
 #l †j Th [i ¢f ef Vf ýf Óf Œg Vg _g ³d hd úe ve \e 3e (e _e ôe ëe Àe ·e
 ¬e ƒb xb ob Db ;b _b _b üb Ób Èb ¿b “b ‹c `c Wc Lc #c _c _c äc Úc ³c ©c
 ž` c` D` &` _` ø` Þ` ‡a |a Ua Ka  a _a _a åa
```

Copy the generated file into a file containing a print job or send the complete generated file (df333.txt) to the printer to execute the therin contained #DF command immediately. To do so, copy the file to the selected printer interface:

```
copy df333.txt lpt1: /b
```

‖ Add /b for copying binary data! ‖

The file font597.spd can now be found in folder `\fonts` on the CompactFlash card. To check this, print the info printout „Memory Status“, which shows a list of any font or logo files contained on the CompactFlash card or RAM disk:



MEMORY STATUS

Internal Memory Configuration

Fonts on Compact Flash Card

FONT597.SPD 60482 Bytes

Fig. 25: Info-printout „Memory Status“, listing the transmitted file font597.spd

Related reference

#DF - Downloading a file on page 28
The command #DF downloads a file from a PC to the printer.

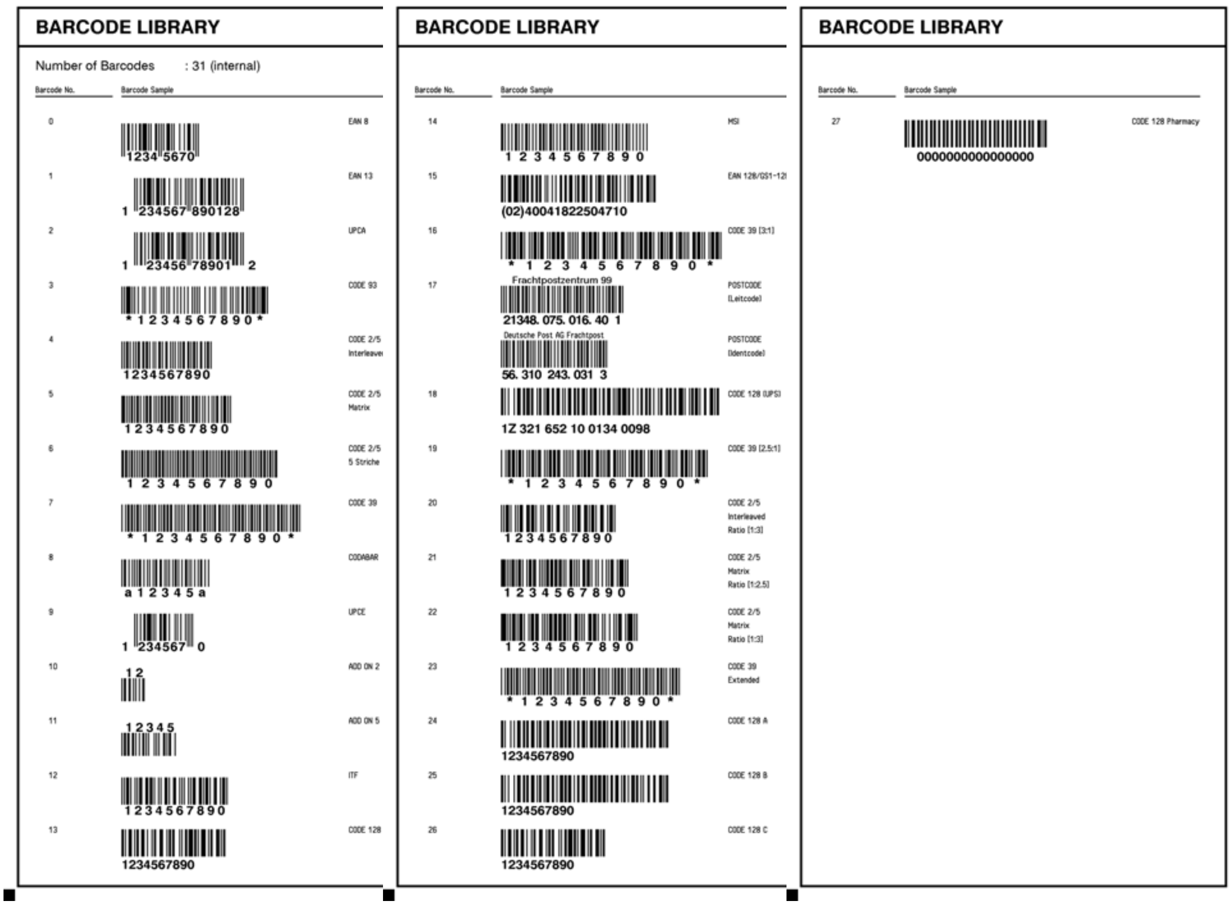
PRINTER-INTERNAL BAR CODES

One-dimensional bar codes

One-dimensional bar codes are printed with the Easy Plug command #YB.

No.	Bar code	No.	Bar code
0	EAN 8	14	MSI
1	EAN 13	15	EAN 128
2	UPCA	16	CODE 39 (3:1)
3	CODE 93	17	POSTCODE (guide and identity code)
4	CODE 2/5 Interleaved	18	CODE 128 (UPS)
5	CODE 2/5 Matrix	19	CODE 39 (2,5:1)
6	CODE 2/5 5-line	20	CODE 2/5 Interleaved Ratio 1:3
7	CODE 39	21	CODE 2/5 Matrix Ratio 1:2,5
8	CODABAR	22	CODE 2/5 Matrix Ratio 1:3
9	UPCE	23	CODE 39 Extended
10	ADD ON 2	24	CODE 128 A
11	ADD ON 5	25	CODE 128 B
12	ITF	26	CODE 128 C
13	CODE 128	27	CODE 128 Pharmacy

Table 21: Overview of one-dimensional bar codes.



BARCODE LIBRARY

Number of Barcodes : 31 (internal)

Barcode No.	Barcode Sample	
0	1234 5670	EAN 8
1	1 234567 890128	EAN 13
2	1 23456 78901 2	UPCA
3	* 1 2 3 4 5 6 7 8 9 0 *	CODE 93
4	1234567890	CODE 2/5 Interleave
5	1 2 3 4 5 6 7 8 9 0	CODE 2/5 Matrix
6	1 2 3 4 5 6 7 8 9 0	CODE 2/5 5 Striche
7	* 1 2 3 4 5 6 7 8 9 0 *	CODE 39
8	a 1 2 3 4 5 a	CODABAR
9	1 234567 0	UPCE
10	1 2	ADD ON 2
11	1 2 3 4 5	ADD ON 5
12	1 2 3 4 5 6 7 8 9 0	ITF
13	1234567890	CODE 128

BARCODE LIBRARY

Barcode No.	Barcode Sample	
14	1 2 3 4 5 6 7 8 9 0	MSI
15	(02)40041822504710	EAN 128/GS1-12
16	* 1 2 3 4 5 6 7 8 9 0 *	CODE 39 [3:1]
17	Frachtpostzentrum 99 21348. 075. 016. 40 1	POSTCODE [Leitcode]
	Deutsche Post AG Frachtpost 56. 310 243. 031 3	POSTCODE [Identcode]
18	1Z 321 652 10 0134 0098	CODE 128 (UPS)
19	* 1 2 3 4 5 6 7 8 9 0 *	CODE 39 [2.5:1]
20	1 2 3 4 5 6 7 8 9 0	CODE 2/5 Interleaved Ratio [1:3]
21	1 2 3 4 5 6 7 8 9 0	CODE 2/5 Matrix Ratio [1:2.5]
22	1 2 3 4 5 6 7 8 9 0	CODE 2/5 Matrix Ratio [1:3]
23	* 1 2 3 4 5 6 7 8 9 0 *	CODE 39 Extended
24	1234567890	CODE 128 A
25	1234567890	CODE 128 B
26	1234567890	CODE 128 C

BARCODE LIBRARY

Barcode No.	Barcode Sample	
27	0000000000000000	CODE 128 Pharmacy

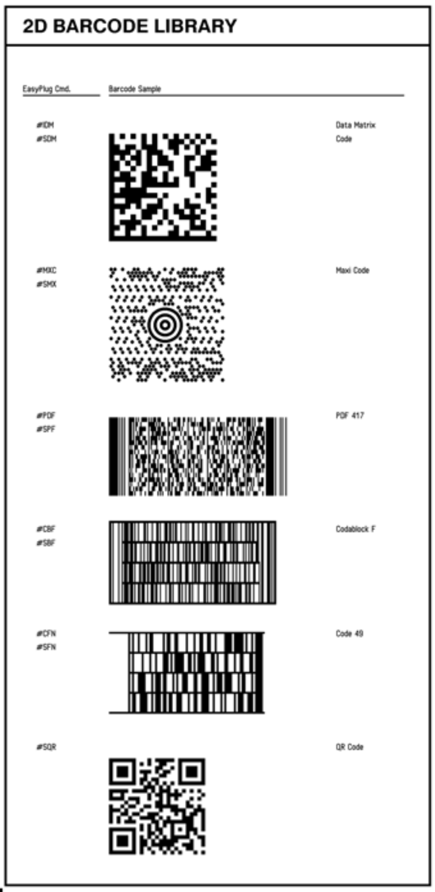
Fig. 26: Printer-internal one-dimensional bar codes (Info-Printout PRINT INFO > Font Status or Info > Status Printouts > Font Status, section „Barcode Library“).

Two-dimensional bar codes

Two-dimensional bar codes are printed with special Easy Plug commands (see table).

Easy Plug command	Bar code
#IDM	Data Matrix Code
#MXC	Maxi Code
#PDF	PDF 417
#CBF	Codabar F
#CFN	Code 49
#SQR	QR Matrix Code

Table 22: Easy Plug commands for two-dimensional bar codes.



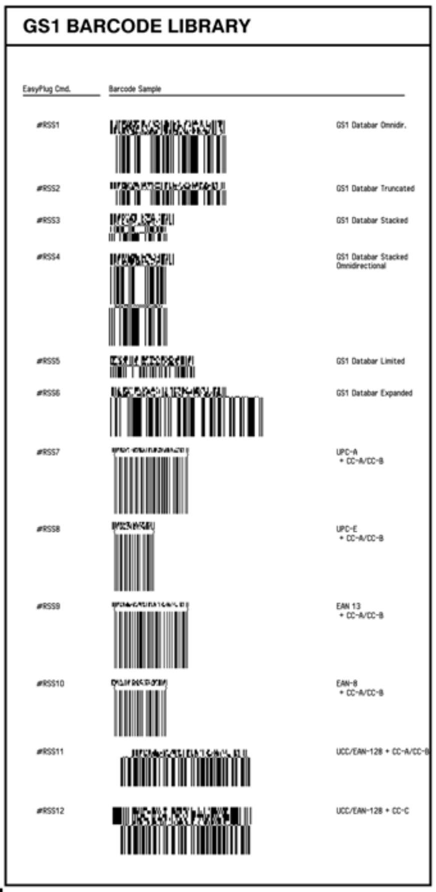
2D BARCODE LIBRARY

EasyPlug Cmd.	Barcode Sample	
#IDM #SDM		Data Matrix Code
#MXC #SMX		Maxi Code
#PDF #SPF		PDF 417
#CBF #SBF		Codablock F
#CFN #SFN		Code 49
#SQR		QR Code

Fig. 27: Printer-internal available 2-dim. bar codes (Info-printout PRINT INFO > Font Status or Info > Status Printouts > Font Status, section „Barcode Library“).

GS1 bar codes

GS1 DataBar (former RSS) and *Composite Component* (CC) bar codes are printed with the Easy Plug command #RSS. The bar code is chosen by the number in the first column of the table, which is added to the command as parameter.



GS1 BARCODE LIBRARY

EasyPlug Cmd.	Barcode Sample	
#RSS1		GS1 Databar Omnidir.
#RSS2		GS1 Databar Truncated
#RSS3		GS1 Databar Stacked
#RSS4		GS1 Databar Stacked Omnidirectional
#RSS5		GS1 Databar Limited
#RSS6		GS1 Databar Expanded
#RSS7		UPC-A + CC-A/CC-B
#RSS8		UPC-E + CC-A/CC-B
#RSS9		EAN 13 + CC-A/CC-B
#RSS10		EAN-8 + CC-A/CC-B
#RSS11		UCC/EAN-128 + CC-A/CC-B
#RSS12		UCC/EAN-128 + CC-C

Fig. 28: Printer-internal available GS1 codes (Info-printout PRINT INFO > Font Status or Info > Status Printouts > Font Status, section „Barcode Library“).

Related reference

#CBF - Bar code Codablock F on page 22
The command #CBF prints bar codes ot the type *Codablock F*.

#CFN - Code 49 on page 23
The command #CFN defines bar code of the type Code 49 (ANSI/AIM-BC6-2000 „Uniform Symbology Specification Code 49“)

#IDM - Data Matrix Code on page 41
The #IDM command prints a 2-dim. „Data Matrix“ bar code.

#MXC - Bar code Maxicode on page 50
The #MXC command prints the 2-dimensional "MaxiCode" bar code.

#PDF - Bar code PDF 417 on page 58
The #PDF command prints a bar code of the type PDF 417.

#RSS - GS1 DataBar & CC on page 67
The #RSS command prints a bar code field with a GS1 DataBar or a composite symbology bar code.

#YS - Real time as bar code on page 125
The #YS command defines a time in text format the form of a bar code.

#SB - Bar code definition on page 73
The #SB command defines of a bar code. Printing of the bar code requires a subsequent #VW-command.

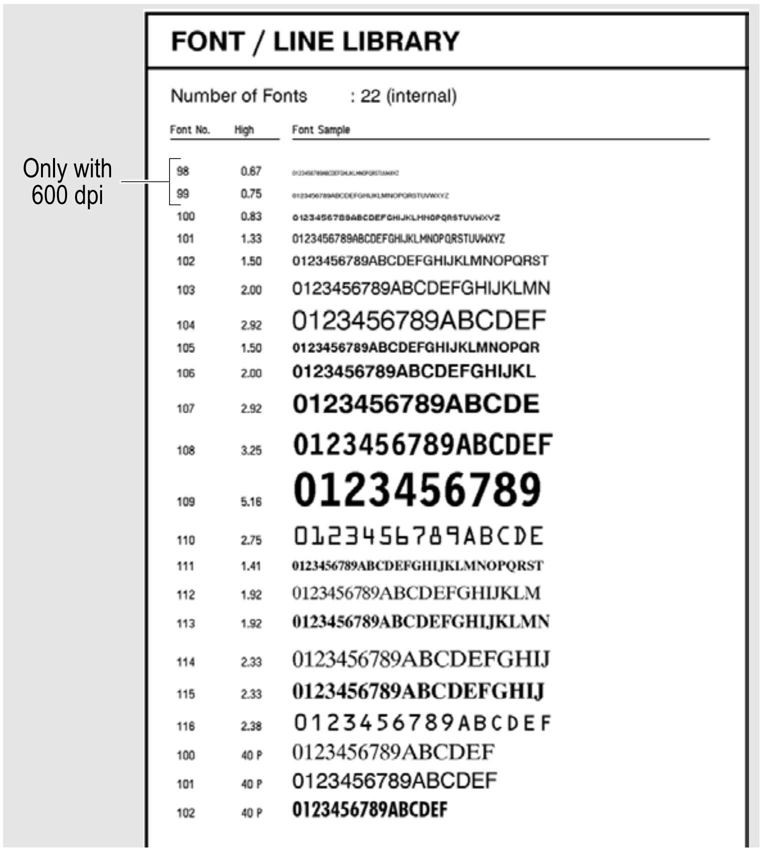
#YB - Bar code definition on page 106
The #YB command is used for (optional) parameter settings and entering text.

PRINTER-INTERNAL FONTS AND LINE STYLES

Printer-internal fonts

The fonts are selected by adding the number in the column „Font No.“ to the appropriate Easy Plug command.

Font No.	Description
98-99	Fonts to be used with 600 dpi printheads (XLP 504 600 dpi)
100-116	Fonts with fixed size (Fixfont); Printed with command #YT
100-102 (at the column end)	Scalable fonts (Speedo font); Printed with command #YN


Only with 600 dpi

FONT / LINE LIBRARY

Number of Fonts : 22 (internal)

Font No.	High	Font Sample
98	0.67	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
99	0.75	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
100	0.83	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
101	1.33	0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
102	1.50	0123456789ABCDEFGHIJKLMNOPQRST
103	2.00	0123456789ABCDEFGHIJKLMN
104	2.92	0123456789ABCDEF
105	1.50	0123456789ABCDEFGHIJKLMNOPQR
106	2.00	0123456789ABCDEFGHIJKL
107	2.92	0123456789ABCDE
108	3.25	0123456789ABCDEF
109	5.16	0123456789
110	2.75	0123456789ABCDE
111	1.41	0123456789ABCDEFGHIJKLMNOPQRST
112	1.92	0123456789ABCDEFGHIJKLM
113	1.92	0123456789ABCDEFGHIJKLMN
114	2.33	0123456789ABCDEFGHIJ
115	2.33	0123456789ABCDEFGHIJ
116	2.38	0123456789ABCDEF
100	40 P	0123456789ABCDEF
101	40 P	0123456789ABCDEF
102	40 P	0123456789ABCDEF

Fig. 29: Printer-internal fonts (Info-printout INFO AUSDRUCKEN > Font Status or Info > Status Ausdrucke > Font Status, section „Font /Line Library“).

Printer-internal line styles

The line styles are selected by adding the number in the column „Line Style“ to the appropriate Easy Plug command as parameter.

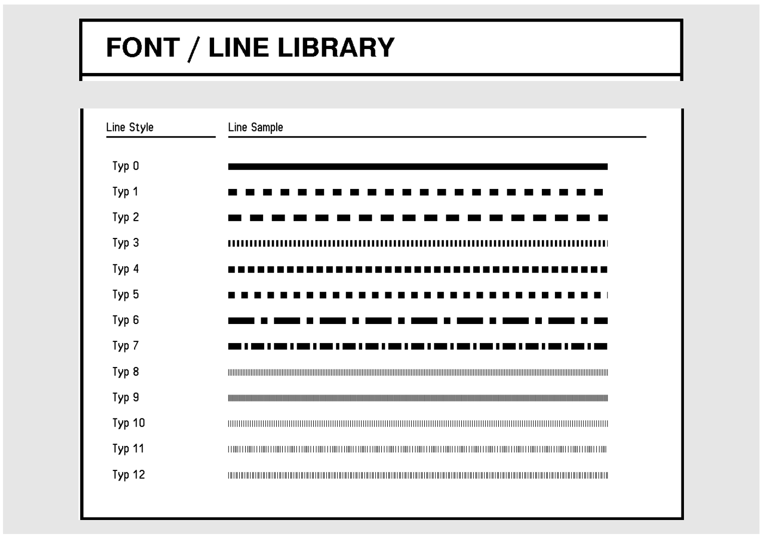


Fig. 30: Printer-internal line styles (Info-printout INFO AUSDRUCKEN > Font Status or Info > Status Ausdrucke > Font Status, section „Font /Line Library“).

Additionally to the styles on the printout, the following line styles are available:

- 13: Checked pattern with 3 dot edge length
- 14: Checked pattern with 1 mm edge length
- 15: Checked pattern with 5 mm edge length

The line width has to be defined as a multiple of the edge length of the checked pattern!

Related reference

#YC - Real time as text on page 108
The #YC command defines a time in text format. The printer must be equipped with a real-time clock. Normal text can also be entered in conjunction with the time.

#YL - Line definition on page 119
The #YL command is used to define lines of varying lengths and thicknesses.

#YN - Text field on page 120

#YR - Rectangle definition on page 124
The #YR command defines rectangles of varying sizes and line thicknesses.

#YT - Text field definition on page 128
The #YT command is used for (optional) parameter settings and entering text.

#RT - Read and print RFID data on page 70
The #RT command reads data of a RFID transponder and prints it on its label.

#SF - Fixfont definition on page 77
The #SF command defines the font for a not scalable text field (fixfont). Printing of the text field requires a subsequent #VW-command.

#SS - Speedo font definition on page 92
The #SS command defines the font for a scalable text field (speedo font). Printing of the bar code requires a subsequent #VW-command

EXPRESSIONS

An expression is a combination of the following elements:

- Constant values
- Variables
- Operator „+“
- Brackets
- Functions

Example:

Command sequence	**Description**
`#VDT/Date////02/02/2005#G`	Define text variable “Date”
`#VDT/Number////2223#G`	Define text variable „Number“
`Date+“:“DayOfYear(SubStr(Date,0,1)+“2“, SubStr(Date,3,2),Substr(Date,6,4))+“End“`	Expression 1
`Date+Number`	Expression 2

Related reference

Variables on page 150
Functions on page 152
Arithmetic functions on page 158

VARIABLES

Easy-Plug is able to calculate and use variable data and store it in *variables*.
Data content of a variable is called *value* of a variable. Values of variables can be composed and manipulated by *functions* and *operators*.

Variable types

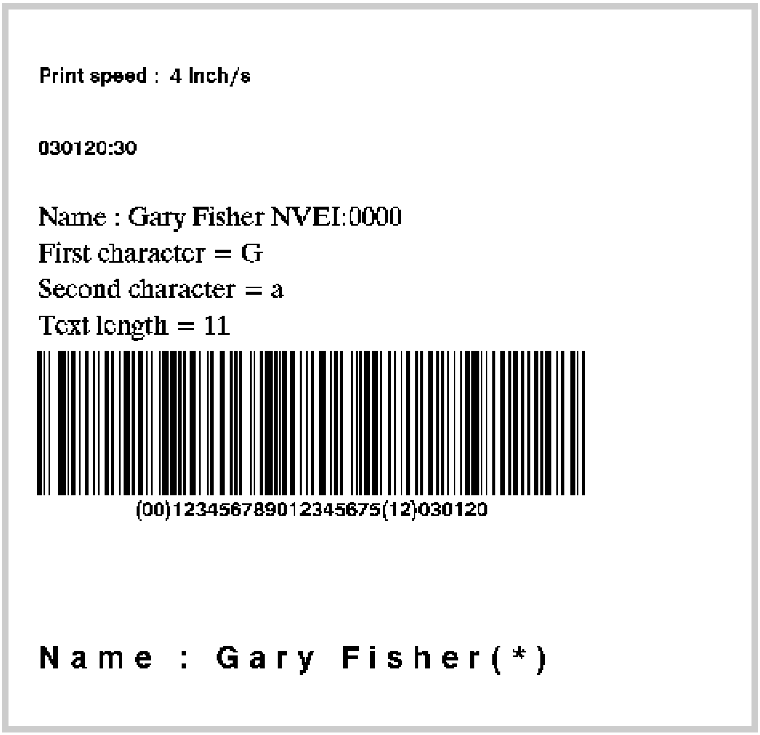
Variable type	**Content**	**Command**
Text variable	Alphanumerical characters, optional with counter	#VDT
Date and time variable	Alphanumerical characters, date and time	#VDD
Expression variable	Expression	#VDE
System variable	Parameter setting as in the printer menu	#VDS
Standalone variable	Alphanumerical characters; the values are requested at the operator panel	#VTS
Print job data variable	Print quantity or label number	#VDP

Table 23: Overview of variable types.

Example

Print job with different variable types:

```
#!A1
#IMS105/101
#ER
#G ------------------------------------------------------------------
#G Variable definitions
#G ------------------------------------------------------------------
#VDT/Name////Gary Fisher#G
#VDT/NVE//+1/1/12345678901234567#G
#VDD/Expire/U/M24/^D^M^Y:^s#G
#VDT/GRAFIKIDX//+1/1/00#G
#G --------------------------------------------
#G Standalone input field
#G --------------------------------------------
#VTS/NVEI/WZN/4/+5/1/0000#G
#G --------------------------------------------
#G System variables
#G --------------------------------------------
#VDS/PrintSpeed/%S/1003#G
#VDS/PrintSpeedName/%N/1003#G
#G --------------------------------------------
#G Expression variables
#G --------------------------------------------
#VDE/BarcodeData//"(00)" + NVE + Mod10(NVE) + "(12)" +SubStr(Ex▶
pire,0,6)#G
#G ------------------------------------------------------------------
#G Printing the elements
#G Rotation 0°, left side alignment
#G --------------------------------------------
#FD/0/L#G
#G --------------------------------------------
#G Fixfont 106 normal spacing
#G --------------------------------------------
#SF106//0#G
#T5#J90 #VW/L/PrintSpeedName + " : " + PrintSpeed#G
#T5#J80 #VW/L/Expire#G
#G --------------------------------------------
#G Speedo Font 100 (bold, 44Dot x and y size)
#G --------------------------------------------
#SS100/B/44/#G
#T5#J70 #VW/L/"Name : " + Name + " NVEI:" + NVEI#G
#T5#J65 #VW/L/"First character = " + SubStr(Name,0,1)#G
#T5#J60 #VW/L/"Second character = " + SubStr(Name,1,1)#G
#T5#J55 #VW/L/"Text length = " + Length(Name)#G
#G --------------------------------------------
#G EAN128 Code / width 3 / height 20 / readline centered
#G --------------------------------------------
#SB15/MK/20/3#G
#T5#J30 #VW/L/BarcodeData#G
#G --------------------------------------------
#G Fixfont 107 with 16 Dotc microspacing
#G --------------------------------------------
#SF107//16#G #T5#J9 #VW/L/"Name : " + Name + "(*)"#G
#G --------------------------------------------
#G Graphics
#G --------------------------------------------
#SG#G
#T82#J75#VW/L/"Graph" + GRAFIKIDX + ".bmp" #G
#Q3/
```


Print speed : 4 Inch/s

030120:30

Name : Gary Fisher NVEI:0000
First character = G
Second character = a
Text length = 11

(00)123456789012345675(12)030120

Name : Gary Fisher(*)

Fig. 31: Printout of the example.

Related reference

Functions on page 152
Arithmetic functions on page 158
Expressions on page 150

FUNCTIONS

Overview

Function	Operation	Description
Mod10	Calculates the checksum of the expression value according to modulo 10	see Mod10 (Expression) on page 154
SubStr	Cuts a substring out of the expression value	see SubStr (Expression, a, b) on page 154
Length	Calculates the length (= number of characters) of the expression value	see Length (Expression) on page 155
MergeRight	Merges 2 expressions and right justifies the result	see MergeRight (Expression1, Expression2) on page 155
MergeLeft	Merges 2 expressions and left justifies the result	see MergeLeft (Expression1, Expression2) on page 155

Function	Operation	Description
DayOfYear	Calculates the day of the year	see DayOfYear (Day, Month, Year) on page 155
Chr	Converts a ASCII code decimal number into the corresponding alphanumeric ASCII character	see Chr(a) on page 156
DecToBin	Converts the decimal part of an expression into a binary string of bytes	see DecToBin (Expression) on page 156
BinToDec	Converts an expression into a decimal number	see BinToDec (Expression) on page 156
HexToBin	Converts an expression given as a hexadecimal number represented in ASCII characters into binary representation	see HexToBin (Expression) on page 156
BinToHex	Converts an expression into a hexadecimal string	see BinToHex (Expression) on page 156
DualToBin	Converts a dual expression into binary representation	see DualToBin (Expression) on page 157
BinToDual	Converts an expression into dual representation	see BinToDual (Expression) on page 157
PadRight	Pads the expression from right with given „Character“, until the string length equals the number „a“	see PadRight (Expression, Character, a) on page 157
PadLeft	Pads the expression from left with given „Character“, until the string length equals the number „a“	see PadLeft (Expression, Character, a) on page 157
IfEqualThenElse	If-then-condition. The function compares two expressions. If those expressions are equal, “ThenExpression” is returned, otherwise “ElseExprssion” is returned	see IfEqualThenElse (CompareExpression1, CompareExpression2, ThenExpression, ElseExpression) on page 157

Table 24: Overview of functions.

Mod10 (Expression)

Calculates the checksum of the expression value according to modulo 10. Weight = 3/1.

Parameter	Description
Expression	For details click link „Expressions“ in chapter „Related Information“ at the end of this section

Calculation example:

Data	**1**	**4**	**6**	**3**	**7**	**6**	**2**	**1**
Weight (3/1)	1	3	1	3	1	3	1	3
Sum	1	12	6	9	7	18	2	3
Total sum	58							
Modulo	10							
Rest	8							
Checksum	10 - 8 = **2**							

SubStr (Expression, a, b)

Cuts a substring out of the expression value.

Parameter	Value	Description
Expression		For details click link „Expressions“ in chapter „Related Information“ at the end of this section
a	int	Position of the first character to be cut out. The counting of the characters starts with "0"
b	int	Number of subsequent characters to be cut out.

Example:

Command sequence	Description
`#VDT/Name////Gary Fisher#G`	Definition of the text variable „Name“
`#VW/L/"First character= " + SubStr(Name,0,1)#G`	Output: „First character = G“

Length (Expression)

Calculates the length (= number of characters) of the expression value.

Example:

Command sequence	Description
`#VDT/Name////Gary Fisher#G`	Definition of the text variable „Name“
`#VW/L/"Text length = " + Length(Name)#G`	Output: “Text length = 11”

MergeRight (Expression1, Expression2)

Copies Expression2 into Expression1 and justifies Expression2 to the right.

Example:

Command sequence	Description
`MergeRight(„00000000“,“123“)`	Result: 00000123

MergeLeft (Expression1, Expression2)

Copies Expression2 into Expression1 and justifies Expression2 to the left.

Example:

Command sequence	Description
`MergeLeft(„00000000“,“123“)`	Result: 12300000

DayOfYear (Day, Month, Year)

Calculates the day of the year.

Parameter	Description
Day	2-character string for the day
Month	2-character string for the month
Year	4-character string for the year

Command sequence	Description
`DayOfYear(„01“,“08“,“2005“)`	Result: 213

Chr(a)

Converts a ASCII code decimal number into the corresponding alphanumeric ASCII character.

Parameter	Value	Description
a	int	ASCII code number to be converted [0..255]

Example:

```
Chr(65)
```

Result: A

DecToBin (Expression)

Converts the decimal part of an expression into a binary string of bytes. The decimal number in the expression must not exceed 2^{64}.

Example:

```
DecToBin(„100")
```

Result: d

BinToDec (Expression)

Converts an expression into a decimal number.

Example:

```
BinToDec(„d")
```

Result: 100

HexToBin (Expression)

Converts an expression given as a hexadecimal number represented in ASCII characters into binary representation.

Example:

```
HexToBin(„3161")
```

Result: 1a

BinToHex (Expression)

Converts an expression into a hexadecimal string.

Example:

```
BinToHex(„1a")
```

Result: 3161

DualToBin (Expression)

Converts a dual expression into binary representation.

Example:

```
DualToBin(„0011000101100001")
```

Result: 1a

BinToDual (Expression)

Converts an expression into dual representation.

Example:

```
BinToDual(„1a")
```

Result: 0011000101100001

PadRight (Expression, Character, a)

Pads the expression from right with given „Character“, until the string length equals the number „a“.

Parameter	Value	Description
Expression		For details click link „Expressions“ in chapter „Related Information“ at the end of this section
Character		ASCII-character
a	int	String length

Example:

```
PadRight(„111","2",5)
```

Result: „11122“

PadLeft (Expression, Character, a)

Pads the expression from left with given „Character“, until the string length equals the number „a“.

Parameter	Value	Description
Expression		For details click link „Expressions“ in chapter „Related Information“ at the end of this section
Character		ASCII-character
a	int	String length

Example:

```
PadLeft(„10101","0",8)
```

Result: „00010101“

IfEqualThenElse (CompareExpression1, CompareExpression2, ThenExpression, ElseExpression)

If-then-condition. The function compares two expressions. If those expressions are equal, “ThenExpression” is returned, otherwise “ElseExprssion” is returned.

Parameter	Value	Description
CompareExpression1		First expression to compare. For details about expressions click link „Expressions“ in chapter „Related Information“ at the end of this section.
CompareExpression2		Second expression to compare. For details about expressions click link „Expressions“ in chapter „Related Information“ at the end of this section.
ThenExpression		Returned expression, if the statement „CompareExpression1 = CompareExpression2“ is *true*
ElseExpression		Returned expression, if the statement „CompareExpression1 = CompareExpression2“ is *false*

Example:

```
IfEqualThenElse(TESTVAR, “0“, “No“, “Yes“)
```

Result: If the variable „TESTVAR“ is set to 0, then „No“ is returned, else „Yes“ is returned.

Example:

```
#G ------------------------------------------------------------------
#G The example gets and prints the day of week as a text string.
#G Printout: “„Day of week: Monday“ (or „Tuesday“ or…)
#G If a day of week number exceeds the range [1..7], the following
#G is printed: „Day of week: Error“.
#G ------------------------------------------------------------------
#!A1
#IMN100/100
#ER
#VDD/DayOfWeek///^w#G
#VDE/DayOfWeekAsString//IfEqualThenElse(DayOfWeek,"1","Monday",
IfEqualThenElse(DayOfWeek,"2","Tuesday",
IfEqualThenElse(DayOfWeek,"3","Wednesday",
IfEqualThenElse(DayOfWeek,"4","Thursday",
IfEqualThenElse(DayOfWeek,"5","Friday",
IfEqualThenElse(DayOfWeek,"6","Saturday",
IfEqualThenElse(DayOfWeek,"7","Sunday"  ,"Error")))))))#G
#SF108
#T0#J10#VW/L/"Day of week: " + DayOfWeekAsString#G
#Q1/
```

Related reference

Variables on page 150
Expressions on page 150

ARITHMETIC FUNCTIONS

General conditions

The following functions for numeric arithmetics are supported:

Function	Operation	Description
Add	Addition	See chapter Add(Expression 1, Expression 2, Output format) on page 160

Function	Operation	Description
Sub	Subtraction	See chapter Sub(Expression 1, Expression 2, Output format) on page 160
Mul	Multiplication	See chapter Mul(Expression 1, Expression 2, Output format) on page 161
Div	Division	See chapter Div(Expression 1, Expression 2, Output format) on page 161
IfThenElse	Comparison of two numbers	See chapter IfThenElse(Expression 1, Operator, Expression 2, Expression 3, Expression 4) on page 161

Table 25: Overview of arithmetic functions.

The parameters of these functions can be any expression which results in a valid floating point number string.

A comma („,“) or a point („.“) can be used as separation of integer part of number and decimal place. If a comma is used at least within one parameter, the result of the calculation is printed with a comma. Otherwise the result is printed with a decimal point.

Examples for valid number strings:

- 2015
- -0.255
- 0,30
- -338.645E-1

Supported number range: ±2.22e-308 to ±1.80e+308

Significant mantissa places: 15-16

Result format

The result format follows this scheme:

```
%[flags][width][.precision][length]specifier
```

Whereat the specifier character at the end is the most significant component, since it defines the type and the interpretation of its corresponding argument:

Specifier	Output	Example
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65

The format specifier can also contain sub-specifiers: *flags*, *width*, *precision* and modifiers (in that order), which are optional and follow these specifications:

Flags	Description
-	Left-justify within the given field width; Right justification is the default (see width sub-specifier).
+	Forces to precede the result with a plus or minus sign (+ or -) even for positive numbers. By default, only negative numbers are preceded with a „-“ sign.
(space)	If no sign is going to be written, a blank space is inserted before the value.
0	Left-pads the number with zeroes (0) instead of spaces when padding is specified (see „width“ sub-specifier).

Width	Description
(number)	Minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded with blank spaces. The value is not truncated even if the result is larger.
*	The width is not specified in the format string, but as an additional integer value argument preceding the argument that has to be formatted.
.number	For e, E, f and F specifiers: this is the number of digits to be printed *after the decimal point* (by default, this is 6). For g and G specifiers: This is the maximum number of significant digits to be printed. If the period is specified without an explicit value for precision, 0 is assumed.

Examples:

Result format string	Example output
%F	392.65
%E	3.9265E+2
%.2f	3.93 (rounded to 2 decimal places)
%08.4f	003.9265 (rounded to 2 decimal places)(8 digits – 1 decimal point – 4 decimal places; padded left side with two zeros)

Add(Expression 1, Expression 2, Output format)

Addition; Return value = Expression 1 + Expression 2

Parameter	Description
Expression 1	First summand
Expression 2	Second summand
Output format	See chapter „Result format“ above

Example:

```
#VDT/Value1////33,64#G
#VW/L/Add(Value1, "3,33", "%.2f")#G
```

Result: "36,97"

Sub(Expression 1, Expression 2, Output format)

Subtraction; Return value = Expression 1 - Expression 2

Parameter	Description
Expression 1	Minuend
Expression 2	Subtrahend
Output format	See chapter „Result format“ above

Example:

```
#VW/L/Sub("20", "1.20", "%.2f")#G
```

Result: "18.80"

Mul(Expression 1, Expression 2, Output format)

Multiplication; Return value = Expression 1 * Expression 2

Parameter	Description
Expression 1	First factor
Expression 2	Second factor
Output format	See chapter „Result format“ above

Example:

```
#VDT/Value1////33,64#G
#VDT/Value2/C/10/1/10#G
#VW/L/Mul(Value1, Value2, "%.2f")#G
```

Result: "336,40"

Div(Expression 1, Expression 2, Output format)

Division; Return value = Expression 1 / Expression 2

Parameter	Description
Expression 1	Dividend
Expression 2	Divisor If the if the devisor is "0.0", the Div function returns "inf" as the result is undefined.
Output format	See chapter „Result format“ above

Example:

```
#VDT/Value2/C/10/1/3.33#G
#VW/L/Div("200.50", Value2, "%.2f")#G
```

Result: "60,21"

IfThenElse(Expression 1, Operator, Expression 2, Expression 3, Expression 4)

Parameter	Description
Expression 1	First number to be compared
Operator	Operator for comparison; Allowed are „>“, „>=“, „<“, „<=“, „==“, „!=“

Parameter	Description
Expression 2	Second number to be compared
Expression 3	Result, which is returned, if the compare condition is *true*
Expression 4	Result, which is returned, if the compare condition is *false*

Example:

```
#VDT/Number/C///3.33#G
#VDE/IfResult//IfThenElse(Number, "<", "0,0", "negative number", "posi►
tive number")#G
```

Result: "positive number"

Application example

```
#!A1
#IMN100/120
#ER
#VDT/Value0////-338,645E-1#G
#VDT/Value1////33,64#G
#VDT/Value2/C/10/1/010#G
#SF106/
#FD0
#T2#J110
#VW/L/Value1 + " * " + Value2 + " = " + Mul(Value1, Value2, "%E")
#T2#J105
#VW/L/"100.33" + " * " + Value2 + " = " + Mul("100"+".33", Value2,
 "%014.11f")
#T2#J100
#VW/L/Value1 + " * " + Value2 + " = " + Mul(Value1, Value2, "%.2f")
#T2#J95
#VW/L/Value1 + " * " + Value2 + " = " + Mul(Value1, Value2, "%.0f")
#T2#J85
#VW/L/Value1 + " / " + Value2 + " = " + Div(Value1, Value2, "%E")
#T2#J80
#VW/L/Value1 + " / " + Value2 + " = " + Div(Value1, Value2, "%014.11f")
#T2#J75
#VW/L/Value1 + " / " + Value2 + " = " + Div(Value1, Value2, "%.2f")
#T2#J70
#VW/L/Value1 + " / " + Value2 + " = " + Div(Value1, Value2, "%.0f")
#T2#J65
#VW/L/Value1 + " / " + "0.0" + " = " + Div(Value1, "0.0", "%.0f")
#T2#J60
#VW/L/Value1 + " + " + Value2 + " = " + Add(Value1, Value2, "%E")
#T2#J55
#VW/L/Value1 + " + " + Value2 + " = " + Add(Value1, Value2, "%014.11f")
#T2#J50
#VW/L/Value1 + " + " + Value2 + " = " + Add(Value1, Value2, "%.2f")
#T2#J45
#VW/L/Value1 + " + " + Value2 + " = " + Add(Value1, Value2, "%.0f")
#T2#J35
#VW/L/Value1 + " - " + Value2 + " = " + Sub(Value1, Value2, "%E")
#T2#J30
#VW/L/Value1 + " - " + Value2 + " = " + Sub(Value1, Value2, "%014.11f")
#T2#J25
#VW/L/Value1 + " - " + Value2 + " = " + Sub(Value1, Value2, "%.2f")
#T2#J20
#VW/L/Value1 + " - " + Value2 + " = " + Sub(Value1, Value2, "%.0f")
#VDE/Result//Div(Add(Value0, Value2, "%E"), Value2,"%.4f")#G
#T2#J10
```

```
#VW/L/"( " + Value0 + " + " + Value2 + " ) / " + Value2 + " = " + Re▶
sult#G
#VDE/IfResult//IfThenElse(Result, "<", "0,0", Sub(",.0",Result,"%.4f"),
 Result)#G
#T2#J5 #VW/L/"ABS( " + Result + " ) = " + IfResult#G
#T2#J1#YL0//1/90
#Q5#G
```

Related reference

Variables on page 150
Expressions on page 150

UNICODE

Unicode is a world-wide, platform independent encoding standard which assigns a unique number to every character. This encoding for example enables printing of Chinese, Japanese, Korean or Arabic characters via Easy Plug.

Requirements

- *Font*: TrueType font on an external memory medium.

> Some fonts contain only a subset of all unicode characters. Make sure that the font contains the relevant characters.
>
> To speed up the access to external fonts [6], the fonts should be loaded into the RAM of the printer. This happens shortly after switching on the printer, if the font number lies between 900 and 999 (for details refer to the „Handbook external flash memory“).

- *RAM*: The printer RAM must provide sufficient memory space for the fonts (for details refer to the „Handbook external flash memory“).

Limitations

- Unicode characters are not supported with *Speedo fonts*
- Unicode characters cannot be used with *counter fields* and *variable fields*.
- For further limitations regarding the application of *arabic glyphs* click the link at the end of the chapter under „Related Information“

Print commands

Unicode characters can be printed using one of the following commands, if the "U" parameter („G“ for arabic characters) is set:

- #YN
- #SS + #VW

Text string input

Input of the text string can be done in one of the following two ways:

- Typing in the Unicode index (\uxxxx) of each single character

6 Fonts that are not installed in the printer (internal fonts), e. g. fonts that are stored on an external memory medium.

- UTF-8 coding of the text: The text is typed or copied into an UTF-8 capable text editor (e. g. standard Windows text editor)

To print a Unicode character, the character index (hex) must be known.

> Unicode indices can be found on the internet under www.unicode.org or you convert the text with the Unicode Converter.

Command example:

```
#YN902/0U/60///Char A = \u0041#G
```

If one of the two parameters "U" or "G" in the #YN command are set and the sequence "\u" is found in the string with a valid hex number thereafter, the Unicode character with this hex index is printed. If the character is not present in the font, a square will be printed. If the sequence is wrong or no valid hex character is found after this sequence, all the characters will be printed as normal text.

Unicode characters can be printed together with ASCII characters. All #YN command options except printing with variable fields and counters can be used with Unicode.

UTF-8

The text string is typed into an UTF-8 capable text editor and saved as UTF-8 coded text file. The so created Easy-Plug print job can be sent directly to the printer data interface, if the printer was set to UTF-8 coding before. This can be done in the following 2 ways:

- Easy Plug: command #N13
- Parameter menu: SYSTEM PARAMETER > Character sets = „UTF-8“ or Printer Language > Easy-Plug Setting > Character sets = „UTF-8“.

> See examples 4 and 7.

Examples

Example 1:

```
#G ---------------------------------------------------
#G Printout: Char A = A
#G ---------------------------------------------------
#!A1
#IMN100/30
#ERN
#T5#J10
#YN902/U/60///Char A = \u0041#G
#Q1#G
```

Example 2:

```
#G -----------------------------------------------------
#G The sequence \u must be followed by a valid four-digit hex character,
#G otherwise the whole sequence will be printed.
#G Printout: Char A = \u004m
#G -----------------------------------------------------
#!A1
#IMN100/30
#ERN
#T5#J10
#YN902/0U/60///Char A = \u004m#G
#Q1#G
```

Example 3:

```
#G ---------------------------------------------------
```

```
#G Printout: Char A = \u041
#G --------------------------------------------------
#!A1
#IMN100/30
#ERN
#T5#J10
#YN902/0U/60///Char A = \u041#G
#Q1#G
```

Example 4: Cyrillic text UTF-8

```
#G --------------------------------------------------
#G Printout: Указания по тезнике безопасности при
#G эксплуатации машины для печатания этикеток
#G --------------------------------------------------
#!A1
#IMN200/100
#N13
#ER
#SS900/OV/32#G
#T01.0#J10.0
#VW/L/"Указания по тезнике безопасности при эксплуатации машины для
 печатания этикеток"
#Q1#G
```

Example 5: Arabic text #YN (printout see below)

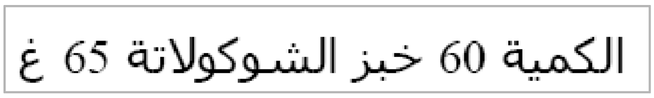
```
#!A1
#IMSR100/200
#ERN/1//0
#T090.50#J90.33
#YN901/RG/50///\u0627\u0644\u0643\u0645\u064A\u0629 60 \u062E
 \u0628\u0632 \u0627\u0644\u0634\u0648\u0643\u0648\u0644\u0627 \u062A
\u0629 65 \u063A#G
#Q1#G
```

Example 6: Arabic text #SS + #VW (printout see below)

```
#!A1
#IMSR100/200
#ERN/1//0
#T090.50#J90.33
#SS901/RG/42X42/0
#VW/L/"\u0627\u0644\u0643\u0645\u064A\u0629 60 \u062E\u0628
 \u0632\u0627\u0644\u0634\u0648\u0643\u0648\u0644\u0627\u062A \u0629 65
 \u063A"#G
#Q1#G
```

Example 7: Arabic text UTF-8 (printout see below)

```
#!A1
#IMSR100/200
#N13
#ERN/1//0
#T090.50#J90.33
#SS901/RG/42X42/0
#G --------------------------------------------------
#G UTF-8-Daten: D8 A7 D9 84 D9 83 D9 85 D9 8A D8 A9 20 36 30 20
#G D8 AE D8 A8 D8 B2 20 D8 A7 D9 84 D8 B4 D9 88 D9 83 D9 88 D9
#G 84 D8 A7 D8 AA D8 A9 20 36 35 20 D8 BA
#G --------------------------------------------------
#VW/L/"الكمية 60 خبز الشوكولاتة 65 غ"#G
#Q1#G
```



الكمية 60 خبز الشوكولاتة 65 غ

Fig. 32: Printout of the examples 5 to 7.

Related reference

#YN - Text field on page 120
Arabic glyphs on page 167

RELATIONSHIP BETWEEN BAR CODE RATIO AND WIDTH FACTOR

Due to the fact, that only whole dots can be printed, the given ratio value will be rounded to the next possible value. This is significantly influenced by the given barcode width factor (number of dots for small element).

Bigger barcode widths allow a better matching of the requested barcode ratio.

Examples

Bar code width factor 1 :

- The narrow element has a width of 1 dot
- The wide element can be either 2 or 3 dots wide
- Requested ratios from 1:2 to 1:2.5 barcode will be printed with ratio 1:2.0
- Requested ratios from 1:25 to 1:3 barcode will be printed with ratio 1:3.0

Bar code width factor 2 :

- The narrow element has a width of 2 dots
- The wide element can be 4, 5 or 6 dots wide
- Requested ratios from 1:2.0 to 1:2.25 barcode will be printed with ratio 1:2.0
- Requested ratios from 1:2.5 to 1:2.75 barcode will be printed with ratio 1:2.5
- Requested ratios from 1:2.75 to 1:3.0 barcode will be printed with ratio 1:3.0

Bar code width factor 3 :

- The narrow element has a width of 2 dots
- The wide element can be 6, 7, 8, or 9 dots wide
- ...

Related reference

#SB - Bar code definition on page 73
The #SB command defines of a bar code. Printing of the bar code requires a subsequent #VW-command.

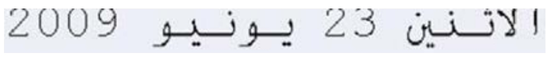
#YB - Bar code definition on page 106
The #YB command is used for (optional) parameter settings and entering text.

ARABIC GLYPHS

Limitations to the use of arabic glyphs with Easy Plug

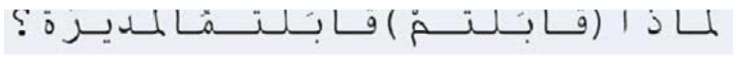
- Arabic glyphs in the range from 0x0600 – 0x06FF will be substituted to their respective contextual (Isolated, Initial, Medial, Final) forms. Urdu, Sindhi, Persian Glyphs in the above range will be printed in the isolated forms only.
- Using a very small font size could cause problems in glyph definition (glyph smudging).
- Only non-vocalised Arabic is supported. Not all ligatures and diacritical combinations for vocalised Arabic can be supported.
- If non-vocalised Arabic is used, only one transparent glyph (diacritical mark, 0x064B – 0x065E) is allowed between two non transparent chars. If two diacritical marks are used, they could overlap.
- Ligature Shaddah (0x0651) with Dammatan, Kasratan, Fatha, Damma, and Kasra only supported in their isolated forms, which means they can only be used at the end of Arabic Words. Medial, and initial forms need to be checked in Unicode.
- If substituted (Isolated, initial, medial, final, combined ligature) glyph is not supported, isolated form of glyph is printed.
- No error message is generated when glyph is not present in the used font.
- Unicode Format characters (e. g. RLM 0x200E, LRM 0x200F, LRE 0x202A, RLE 0x202B) are not filtered from data stream and should not be used for directional encoding of data being sent to the printer.
- Depending on the placement of non Arabic text within a sentence, a space may be required at the end to ensure correct printing.

```
#YN900/0G/100///\u0627\u0644\u0627\u062b\u0646\u064a\u0646 23 \u064a
\u0648\u0646\u064a\u0648 2009 #G
```


الاثنين 23 يونيو 2009

- Using Brackets. Since brackets are handled as non Arabic glyphs, bracket direction for Arabic is not printed correctly. For correct bracket representation in Arabic use the opposite bracket.

```
\u0644\u0645\u0627\u0630\u0627)\u0642\u0627\u0628\u064E\u0644 \u062A
\u0640\u064F\u0645\u0652(\u0642\u0627\u0628\u064E\u0644 \u062A
\u0640\u064F\u0645\u064F\u0627\u0644\u0645\u062F\u064A \u0631\u064E
\u0629\u064E\u061F#G
```


لماذا (قابَلتمْ) قابَلتمُالمديرة؟

Examples with arabic text

Example 1: Arabic and latin text combined

```
#!A1
#IM200/50
#ER
#G ---------------------------------------------------
#G Variable definition
#G ---------------------------------------------------
#VDT/User///NOVEXX Solutions 64-08
#G ---------------------------------------------------
#G Text: "User Name:"
```

```
#G ---------------------------------------------------
#SS900/OVG/44/#G
#T020.0 #J020.0
#VW/L/"\u0627\u0644\u0645\u0646\u062A\u062C
\u0627\u0644\u0648\u0635\u0641"+ User+" :"
#Q1/#!P1
```

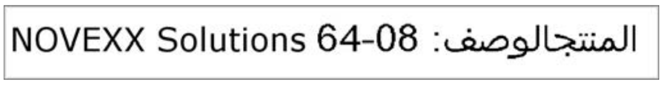

NOVEXX Solutions 64-08 :المنتجالوصف

Fig. 33: Printout of example 1.

Example 2: Variable data

```
#!A1
#IM200/120
#DM ^\u064A\u0646\u0627\u064A\u0631
^\u0641\u0628\u0631\u0627\u064A\u0631
^\u0627\u0644\u0645\u0633\u064A\u0631\u0629
^\u0623\u0628\u0631\u064A\u0644
^\u0631\u0628\u0645\u0627
^\u064A\u0648\u0646\u064A\u0648
^\u064A\u0648\u0644\u064A\u0648
^\u0623\u063A\u0633\u0637\u0633
^\u0633\u0628\u062A\u0645\u0628\u0631
^\u0623\u0643\u062A\u0648\u0628\u0631
^\u0646\u0648\u0641\u0645\u0628\u0631
^\u062F\u064A\u0633\u0645\u0628\u0631
#ER
#VDD/DateTime/Y//^R : ^S ^D
#SS900/OVG/44/#G
#FD/0/R#G
#T50.0 #J20.0
#VW/L/"\u0633\u062C\u0644\u062A\u0627\u0631\u064A\u062E "+DateTime#G
#Q1/#!P1
```

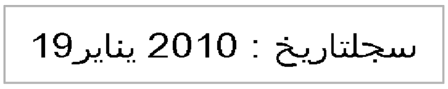

سجلتاريخ : 2010 يناير19

Fig. 34: Printout of example 2.

Examples with arabic text and numbers

Example 1:

```
#T100.0 #J50.0
#YN900/0GR/100x60///\u0627\u0644\u0633\u0639\u0631 100\u066B00 : \u062F
\u0648\u0644\u0627\u0631#G
```

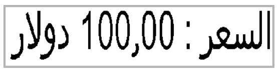

السعر : 100,00 دولار

Fig. 35: Printout of example 1 (engl.: „Price: 100,00 Dollars“).

Example 2:

```
#T100.0 #J50.0
#YN900/0GR/100x60///\u0627\u0644\u0648\u0632\u0646 20,2 :
 \u0643\u0644\u063A#G
```

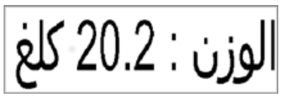

الوزن : 20.2 كلغ

Fig. 36: Printout of example 2 (engl.: „Weight: 20.2 kg“)

Example 3:

```
#T100.0 #J50.0
#YN900/0GR/100x60///\u0627\u0644\u0641\u0648\u0644\u0637\u064A\u0629
 14.4V :#G
```

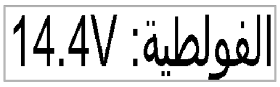

الفولطية: 14.4V

Fig. 37: Printout of example 3 (engl.: „Voltage: 14.4 V“)

Example 4:

```
#T100.0 #J50.0
#YN900/0GR/100x60///\u0645\u0626\u0648\u064A %100.00 :#G
```

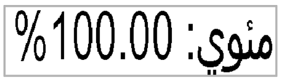

مئوي: %100.00

Fig. 38: Printout of example 4 (engl.: „Percentage: 100.00%“)

Examples with arabic numbers

Example 1:

```
#YN900/0GR/100x60///\u0661\u0662\u0663\u066B\u0660\u0664\u0665\u0666#G
```

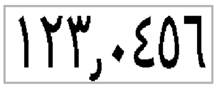

١٢٣٫٠٤٥٦

Fig. 39: Printout of example 1 (engl.: „123.0456“)

Example 2:

```
#YN900/0GR/100x60///
\u0660\u0661\u0662\u0663\u0664\u0665\u0666\u0667\u0668\u0669#G
```

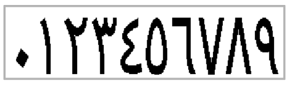

٠١٢٣٤٥٦٧٨٩

Fig. 40: Printout of example 2 (engl.: „123456789“)

Example 3:

```
#YN900/0GR/100x60///
\u06F0\u06F1\u06F2\u06F3\u06F4\u06F5\u06F6\u06F7\u06F8\u06F9#G
```

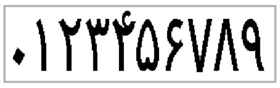

۰۱۲۳۴۵۶۷۸۹

Fig. 41: Printout of example 4 (engl.: „0123456789“)

Related concepts

Unicode on page 163

Related reference

#YN - Text field on page 120

NOVEXX
SOLUTIONS

Telephone: +49-8165-925-0 | www.novexx.com

Novexx Solutions GmbH

Ohmstraße 3
85386 Eching
Germany