Software Manual

Card Printer

Plus

State: April 2004

3090.02.099.90.06

Contents

1 ESC-sequences

This part of the manual contains printer commands to directly control the printer. It is subdivided into four chapters (control sequences, object sequences, objects and preferred control sequences).

1.1 Creation of data records

Here, the structure of a data record as well as the application of the different sequences is shown.

1.1.1 Symbols and conventions

Following some specific symbols and conventions will be used:

Bold printed characters are keywords and have to be written exactly in this way.

Italic characters within a sequence are fill-ins which are replaced by the user by defined values.

Underlined values are standard values.

Non-printable control characters are enclosed in angle brackets, e.g. <CR>.

Input of control characters can vary depending on the programme. Here are some examples:

<SX>,<STX>	þ	STX-character, 02 decimal, 02 hexadecimal, CTRL/B
<ET>,<EOT>	þ	EOT-character, 04 decimal, 04 hexadecimal, CTRL/D
<EQ>,<ENQ>	þ	ENQ-character, 05 decimal, 05 hexadecimal, CTRL/E
<AK>,<ACK>	þ	ACK-character, 06 decimal, 06 hexadecimal, CTRL/F
<CR>,<CR>	þ	CR-character, 13 decimal, 0D hexadecimal, CTRL/M
<EC>,<ESC>	þ	ESC-character, 27 decimal, 1B hexadecimal, CTRL/[

Input choices are displayed in brackets [ ].

Following conventions are valid for numerical values in ASCII-input format: Each digit position gets a fill-in in form of a letter. The letter defines the numerical format. Each number is represented by its ASCII-value.

dddd means: 4-digit decimal number (0000 - 9999) in ASCII-format

hhh means: 3-digit hexadecimal number (000 - fff) in ASCII-format

If the input is not bound to a fixed format, this will be shown by two dots (..):

d.. means for example: 1, 12, 123, 012, ...

Following conventions are valid for numerical values in binary input format: Each 8-bit-binary number is represented by its hexadecimal value. Each hexadecimal digit gets a fill-in (1 hexadecimal digit = 4 bits).

HH means: 2-digit hexadecimal number (00 - ff) in hexadecimal format

af means: 175 decimal, af hexadecimal, 10101111 binary

Following conventions are valid for character values: Each character position gets a fill-in in form of a letter. The letter defines the character format. Each character is represented by its ASCII-value.

c means: 1 character (A - Z, a - z)

a means: 1 character (A - Z, a - z, 0 - 9)

0|1 proposes input possibility of 0 or 1.

Lower$_{positioned}$ texts are for information purposes only.

d_{type} means: 1-digit type number, decimal.

1.1.2 Utilization of control sequences

Printer control depends on the printer configuration. The configuration can either be changed by menu or control sequences.

A control sequence has following structure:

It begins with <ESC> and a subsequent small letter as keyword.

All control sequences are terminated with <CR>.

The sum of all consecutive control sequences forms the control block.

Incorrect control sequences results in error messages on the screen with the reference 'control sequence'.

1.1.3 Utilization of object blocks

Object blocks define the actual printout after the printer configuration has been made.

The PLUS is equipped with its own job control language. This language is incompatible to other line-oriented matrix printers, but it offers considerable possibilities to create a layout. Lines of text as well as barcodes and logos can be printed at will on a card in various forms and directions.

First, to print a card, the whole card layout has to be defined as object blocks. Only after the whole layout block has been transferred to the printer a card can be printed.

Object blocks have following structure:

An object block is composed of object sequences, which start with **<ESC>** and a subsequent capital letter as keyword.

Object sequences within an object block can either be ended with or without **<CR>**.

All object blocks together form a layout block. A layout block has to be enclosed in **<STX>**....**<EOT>** in order to distinguish it from control blocks.

For most of the error messages resulting of faulty object sequences, the error message 'object sequence' is displayed on the screen.

Variable object sequences are formally used like control sequences.

All object parameters already have a default value, e. g. character spacing = 1 dot.

1.1.4 Structure of data records

A data record has following structure:

Data record	= {control block, layout block, control block }	
Control block	= { Control sequence 1 [Control sequence n] } n=[2...]	
Control sequence	= { <**ESC**>x.... <**CR**>}	x=[a|b|...|z]
Layout block	={<**STX**>Object block 1[Object block *n*] <**EOT**>}	*n* = [2...
Object block	= { [Object sequence *n*] Object }	*n* = [1...
Object sequence	= { <**ESC**> *X* ... [<CR>] }	*X* = [A|C|...|Z]
Object	= { <**ESC**> **B**|**T**|**L** ... [<CR>] }	

Data record
Defines the entire card layout including the printer configuration and print process.

Control block
Defines the entire printer configuration and the print process.

Control sequence
Sets a control parameter for the printer configuration respectively print process.

Object block
Defines the whole card layout.

Object sequence, object
The card layout includes different types of data fields. An object sequence transfers one parameter for the specification of this field. An object is formally an object sequence and specifies the field type (text, barcode, logo/line/frame). This sequence always has to be placed at the end of each object block. The object sequences before are related to this object.

First the data records are buffer stored and then edited successively. A data record already edited does not use any longer input memory. The following symbols in the standard display indicate the current memory state:

-> þ Data can be read in
| þ Input block complete
þ No data record edited
n þ Data record edited
þ Number of cards still to be printed by current data record

1.1.5 Example for a data record

ESC-sequence	Note
<ESC>k0000<CR>	**Start of data record** Control sequences: (1st control block) Printer parameter Printout without print button
<STX>	**Start of layout block**
<ESC>I35<CR> <ESC>G150<CR> <ESC>R0<CR> <ESC>BEAN13;H60;B3;P1>4012 34567890<CR>	Object block 1 (= barcode set)
<ESC>I35<CR> <ESC>G20<CR> <ESC>R270<CR> <ESC>D1<CR> <ESC>C2<CR> <ESC>F3<CR> <ESC>TARIAL18F;Drehung 270°<CR>	Object block 2 (=text data) Text = ' Rotation 270° '
<EOT>	**End of layout block**
<ESC>#1<CR>	Control sequences: (2nd control block) Print job **End of data record**

After these data have been transfered to the printer, a card should be printed.

If an error message appears on the screen after the transfer of the data record, the corresponding reason can be found in the appendix where the error codes are mentioned.

A possible reason for the error might be, that the connection PC - Printer has not been set up correctly or that the transfer parameters have not been adjusted correctly.

1.2 Control sequences

Structure:

Control block	= { Control sequence 1 [control sequence n] } n=[2...]
Control sequence	= { <**ESC**>x.... <**CR**>} x=[a|b|...|z]

Listing of the possible Control Sequences:
Number of cards
Image height
Image width
Printing speed
Printer specification
Variable object: Logo
Country code
Transponder data
Variable object: Text/Barcode
Heating time thermal print head

The following pages give a detailed description of the individual control sequences.

1.2.1 Number of cards

Menu choice:	CARD DATA / copies / 0000
Control set:	<**ESC**>#*d..*<**CR**>
Example:	<ESC>#10<CR>

The control sequence initiates the printout of the print job with the given number of cards.

Parameter: *d* = Number of cards

1.2.2 Image height

Menu choice:	CARD DATA / select card / 1024
Control set*:*	<**ESC**>**b***d..*<**CR**>
Example:	<ESC>b840<CR>

The *image height* defines the height of the print area.

Parameter: *d* = Image height * 12/mm (number of motor steps),
Minimum height = 10 mm (d = 120), Maximum height = 85.3 mm (d = 1024)

1.2.3 Image width

Menu choice:	CARD DATA / select card / 672
Control set:	<**ESC**>**c***d..*<**CR**>
Example:	<ESC>c450<CR>

The *image width* defines the width of the print area.

Parameter: *d* = Image width * 12/mm (number of dots),
Minimum width = 5.3 mm (d = 64), Maximum width = 56 mm (d = 672)

The value for the *image width* can always be smaller than the actual card format. The resulting print area is then shifted in a centered way.

1.2.4 Printing speed

Menu choice:	PRINT PARAMETERS / printing speed / 75
Control set:	<**ESC**>**j***d..*<**CR**>
Example:	<ESC>j100<CR>

The *Printing speed* defines the transportation speed of the card within the print area.

Parameter: *d* = Printing speed (75 and 100 mm/sec) in steps of 25 mm/sec

1.2.5 Printer specification

Menu choice:	PRINTER SPECS. /printer - mode /transfer...
Control set:	<**ESC**>**k***d..*[**;0**|**;1**]<**CR**>
Example:	<ESC>k101;1<CR>

The Control Sequence defines the setting of various printer parameters.

Parameter:

d =	*1*	printout only via print button
d =	*20*	use transfer printing
d =	*40*	Card feeder available
d =	*100*	automatic card feeding

By adding up the individual values several settings can be made at the same time. The optional switch (1|0) allows the setting/deletion of single parameters.

Examples:

ESC-sequence	**Description**
<ESC>k20<CR>	Reset all parameters, but • transfer device will be logged-on
<ESC>k21;1<CR>	All parameters remain unchanged, but • activate transfer device • inquire PRINT button prior to each printout

1.2.6 Variable object: logo

Control set:	<**ESC**> **l** $a_{obj.}$ **;** $d.._{Width}$ **;** $d.._{Height}$ **;** $HH_{Logo\ data}$ <**CR** >
	$a_{obj.}$ *:* Object name
	$d.._{Width}$ *:* Logo width : Number of dots in X-direction
	$d.._{Height}$ *:* Logo height : Number of dots in Y-direction
	$HH_{Logo\ data}$ Logo data in binary form
	Each bit of a byte represents 1 dot
	Bit = 0: do not print dot, = 1: print dot
	Data bit: 7 6 5 4 3 2 1 0
	Dot: 1 2 3 4 5 6 7 8
Example:	<ESC> l a ; 8 ; 5 ; 08 08 C8 28 10 <CR>
	<ESC> # 1 <CR>

The *Logo* should have been pre-defined as a variable object (see variable objects, chapter 1.3.10, page 16).

1.2.7 Country Code

Menu Choice:	CHARACTER/LCD / country / USA
Control Set:	<**ESC**>**n***d*<**CR**>
Example:	<ESC>n2<CR>

The control sequence defines the setting of the various country codes.

Parameter:

d =	0	USA		d =	5	Sweden
d =	1	England		d =	6	Italy
d =	2	Germany		d =	7	Spain
d =	3	Denmark		d =	8	Norway
d =	4	France		d =	9	Netherlands

1.2.8 Card feeding / Card output

Control set:	<**ESC**> **t** *d* <**CR** >	
	d_1 :	Card feeding
	d_2 :	Card output
e. g.:	<ESC> t *1* <CR>	Positioning of the card in print position by the card feeder
	<ESC> t *2* <CR>	Card output

1.2.9 Transponder data

Control set:	<**ESC**> **u** $d..._{Offset}$; $d..._{Length}$; a_{Action} ; [HH_{Data}] <**CR**>	
	d_{Offset} = Offset in blocks:	0 ... number of blocks
	d_{Length} = Length in byte:	0 ... transponder size *
	a_{Action} = r / w	r → read / w → write
	HH_{Data} = Transponder data:	data in binary form
Example:	<ESC>u1;5;r <CR>	
	the Offset is one block and then the printer reads 5 bytes	
	* max. read = 216 bytes/sequence; max. write = 32 bytes/sequence	

The printer reads the content of the transponder or writes data to the transponder and passes the corresponding data to the host. (see chapter 1.3.9, page 15)

The data output is according to ISO 15693 and is send via serial- / parallel interface.

Transponder Tag-it HF / I-CODE1:

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
E. g.					H	a	l	l	o								
Offset	0				1				2				3				4

Transponder MY-D:

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
E. g.									H	a	l	l	o				
Offset	0								1								2

ESC-sequence	**Description**
<ESC>u0;0;r;<CR>	read transponder ID return: 9 bytes 1st byte = transponder type 2nd – 9th byte = transponder ID
<ESC>u0;0;w;<CR>	delete all transponder data return: number of del. bytes
<ESC>u2;12;r;<CR>	the Offset is two blocks and the printer reads 12 bytes return: 12 bytes
<ESC>u2;12;w;1234567Hallo <CR>	the Offset is two blocks and the printer writes 12 bytes return: number of real written bytes

transponder model	**code**	**number of blocks**	**block size**	**capacity**
Philips I-CODE1™	0x00	11	4 bytes	44 bytes
Texas Instruments Tag-it HF™	0x01	6	4 bytes	22 bytes
Infineon my-d	0x03	125	8 bytes	1000 bytes

1.2.10 Variable object: Text / Barcode

Control set:	<**ESC**> **v** $a_{obj.}$ **;** *Data* <**CR** >	
	$a_{obj.}$:	Object name
	Data :	Text data or Barcode data
Example:	<ESC> v *a* ; *Text* <CR>	Text-object
	<ESC> # 1 <CR>	
	<ESC> v *a* ; *Barcode data* <CR>	Barcode-object
	<ESC> # 1 <CR>	

The t*ext respectively barcode* has to be pre-defined as variable object.
(see variable objects, chapter 1.3.10, page 16)

1.2.11 Thermal print head heating time

Menu choice:	PRINT PARAMETERS / heating time tph / +- 0%
Control set:	<**ESC**>**w**[+|-]*d*<**CR**>
Example:	<ESC>w-5<CR>

The control sequence allows to change the blackening degree by increasing respectively decreasing the thermal print head heating time between -30% and +30%.

Parameter: *d* = Heating time change (-30% to +30%) in 5% steps

1.3 Object sequences

Structure:

Data record	= { Control block, Layout block, Control block }
Layout block	= { <**STX**> Object block 1 [Object block *n*] <**EOT**> } *n* = [2...]
Object block	= { [Object sequence *n*] Object } *n* = [1...]
Object sequence	= { <**ESC**> *X* ... [<**CR**>] } *X* = [A|C|...|Z]
Object	= { <**ESC**> **B**|**T**|**L** ... [<CR>] }

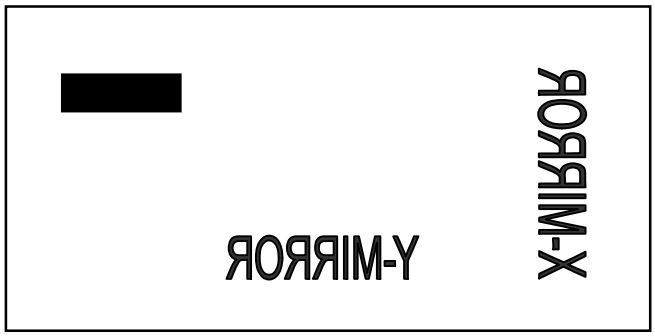
Object sequences are used to define a data layout (layout block).

By several object sequences an object can be defined in form and position. All object sequences which describe an object constitute an object block. The last object sequence of an object block defines the object type (text, barcode, logo/frame/line). The number of objects is not limited.

All object blocks together form a layout block for a card. The following pages describe the single object sequences in detail.

1.3.1 Object attributes

Object sequence:	<**ESC**> **A** *dddd* [<CR>]	
	dddd	= 0001 : Invert object
		= 0002 : Mirror object on X-axis
		= 0004 : Mirror object on Y-axis
		= 0010 : Switch off transparency (no OR switching operation)
Example:	<ESC>A0001	Default: A0000



It is also possible to define several attributes at the same time. This needs only a sum up of the single attributes.

Example:

ESC-sequence	**Description**
<ESC> A 0003	**inverted** and **mirrored** at the X-axis

1.3.2 Y-Enlargement

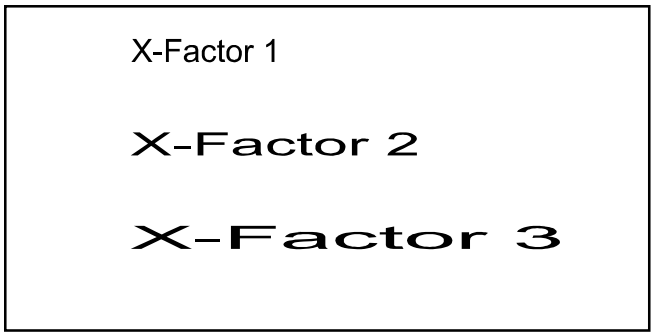
Object sequence:	<**ESC**> **C** *d..* [<CR>]	
	d	= 1 ... 255 : Y-Enlargement ratio
Example:	<ESC>C2	Default: C1

Example:

```
<STX>
<ESC>G50<ESC>I35<ESC>C1<ESC>TARIAL14f;Y-Factor 1
<ESC>G50<ESC>I70<ESC>C2<ESC>TARIAL14f;Y-Factor 2
<ESC>G50<ESC>I135<ESC>C3<ESC>TARIAL14f;Y-Factor 3
<EOT>
```

1.3.3 X-Enlargement

Object sequence:	<**ESC**> **D** *d..* [<CR>]	
	d	= 1 ... 255 : X-Enlargement ratio
Example:	<ESC>D2	Default: D1



Example:

```
<STX>
<ESC>G50<ESC>I50<ESC>D1<ESC>TARIAL14f;X-Factor 1
<ESC>G50<ESC>I90<ESC>D2<ESC>TARIAL14f;X-Factor 2
<ESC>G50<ESC>I140<ESC>D3<ESC>TARIAL14f;X-Factor 3
<EOT>
```

1.3.4 Distance between characters

Object sequence:	<**ESC**> **F** *d..* [<CR>]	
	d	= 1 ... 255 : Number of blank dots
Example:	<ESC>F3	Default: F1

Definition of the distance between 2 characters. The blank dots defined here are added to the space between characters pre-set in the drawing font.

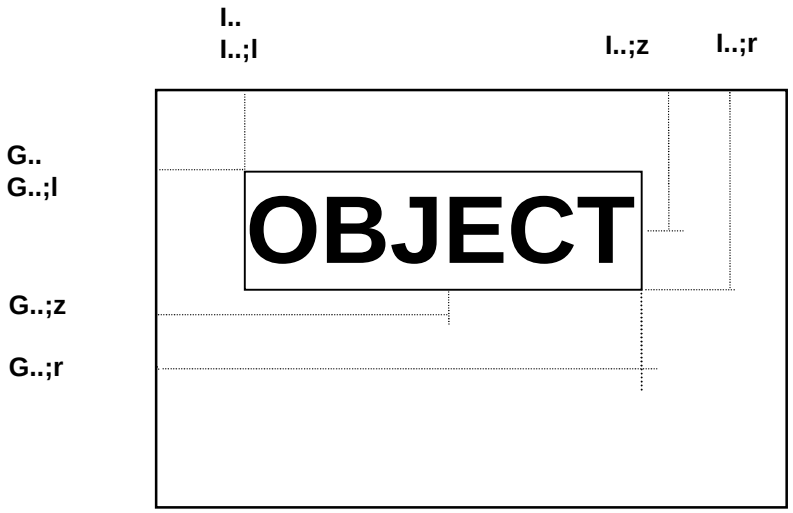
Character Spacing 1
Character Spacing 2
Character Spacing 3
Character Spacing 4
Character Spacing 5

Example:

```
<STX>
<ESC>G50<ESC>I50<ESC>F1<ESC>TARIAL14f;Character Spacing 1
<ESC>G50<ESC>I100<ESC>F2<ESC>TARIAL14f;Character Spacing 2
<ESC>G50<ESC>I150<ESC>F3<ESC>TARIAL14f;Character Spacing 3
<ESC>G50<ESC>I200<ESC>F4<ESC>TARIAL14f;Character Spacing 4
<ESC>G50<ESC>I250<ESC>F5<ESC>TARIAL14f;Character Spacing 5
<EOT>
```

1.3.5 Positioning

The positioning of objects is realized by defining the X- and Y-coordinates. Without any further specifications this defined coordinate dot is related to the upper left corner of the corresponding text-, barcode- or logo-object. By defining additional parameters, the middle as well as right respectively lower edge can be defined as reference point. Thus, it is possible to centre all objects of a card layout not only in X- and Y-direction but also to align them to both edges. When positioning, the minimum distance of 3 mm to the lower edge is to observe.



1.3.5.1 X-Coordinate

Object sequence:	<**ESC**> **G** *d..* [**;** *x - Alignment*] [<CR>]		(X - Coordinate)
	d	= 1 ... 672	Number of dots from left edge
	x - Alignment	= l	left
		r	right
		z	centred
Example:	<ESC>G10;z		

1.3.5.2 Y-Coordinate

Object sequence:	<**ESC**> **I** *d..* [**;** *y - Alignment*] [<CR>] (Y - Coordinate)			
	d	=	1 ... 1024	Number of dots from upper edge
	y - Alignment	=	l	left
			r	right
			z	centred
Example:	<ESC>I10			

1.3.6 Internal logo

Object sequence:	<**ESC**> **M** *Image name* **;** <CR >		
	Image name	=	Name of internally stored logo
Example:	<ESC>MFDLogo;		

Using this sequence, logos which are implemented into the printer can be integrated into a data record.

Those logos stored into the printer can be displayed and printed via the menu *PRINTER INFOS / Logos*.

1.3.7 Stepping

Object sequence:	**<ESC> Q** *d..*$_W$;d..$_Z$ [;*d..*$_F$ [;*d..*$_B$ [;*d..*$_A$]]] [<CR>]		
d..$_W$	= -9 ... +9	:	Stepping value
d..$_Z$	= 1 ... 255	:	Stepping cycle
			1-254: Number of cards without stepping
			255 : Stepping after each print job
d..$_F$	= 0 | 1	:	Filter for leading zeros (text-objects)
			0 : print leading zeros
			1 : suppress leading zeros
d..$_B$	= 1 ...	:	Start of stepping field
			Position of 1st digit of stepping field
d..$_A$	= 0 ...	:	Field Size: Number of digits
			= 0 : to the end of text-, barcode string
			> 0 : Number of relevant digits
Example:	<ESC>Q1;1		

The stepping function can be used for **text**- and **barcode**-objects.

No stepping will be made if the field definition is selected so that the defined field is not completely positioned inside the corresponding text or barcode string.

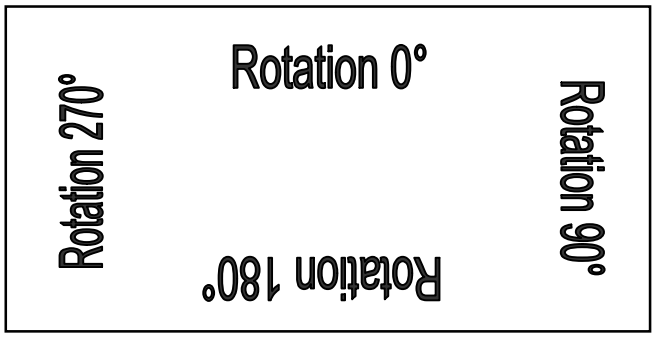
Example:

ESC-sequence	Description
<ESC>Q1;1	After each card the whole text string is stepped by +1. Leading zeros are printed.
<ESC>Q-1;2;1	Every 2 cards the whole text string is stepped by -1. Leading zeros are replaced by blanks. If the value of the corresponding field = 0, a '0' is printed at the end.
<ESC>Q1;1;0;4	After each card the relevant digit field is stepped by +1. Leading zeros are printed. The relevant digit field starts with the 4th place of the corresponding text- or barcode-string and ends at the string end.
<ESC>Q1;1;0;4;3	After each card the relevant digit field is stepped by +1. Leading zeros are printed. The relevant digit field starts with the 4th place of the corresponding text- or barcode string and includes 3 digits in total.
<ESC>Q1;255	After each completed print job the whole text string is stepped by +1. Leading zeros are printed. A print job is started with the control sequence: <ESC># d..

1.3.8 Rotation

Object sequence:	<**ESC**> **R** 0 | 90 | 180 | 270 [<CR>]
Example:	<ESC>R0

Each object can be rotated by 90°, 180° or 270°.



Example:

```
<STX>
<ESC>G396;z<ESC>I20<ESC>R0<ESC>TARIAL20f;Rotation 0°
<ESC>G700<ESC>I240;z<ESC>R90<ESC>TARIAL20f;Rotation 90°
<ESC>G396;z<ESC>I400<ESC>R180<ESC>TARIAL20f;Rotation 180°
<ESC>G20<ESC>I240;z<ESC>R270<ESC>TARIAL20f;Rotation 270°
<EOT>
```

1.3.9 Write transponder

Object sequence:	<**ESC**> **U** *d..*$_{Offset}$ *; d..*$_{Length}$ *; HH*$_{Data}$ <**CR**>	
	d $_{Offset}$ = Offset in blocks:	0 ... number of blocks
	d $_{Length}$ = Length in byte:	0 ... 32 bytes
	HH $_{Data}$ = Transponder data:	data in binary form
Example:	<ESC>U1;5;Hallo<CR>	
	The Offset is one block and then 5 bytes (Hallo) are written	

Before printing, the data are written on the transponder and a write control is executed simultaneously. In case of a successful write process a short signal tone can be heard. In case of a write error a long signal tone can be heard and the error is indicated on the screen. In case of an error the card is not printed.

The Transponder is partitioned in blocks of 4 bytes / 8 bytes each. If one block is not completely written the remainder of the block is filled with FF. (see chapter 1.2.9, page 9)

1.3.10 Variable objects

Object sequence:	<**ESC**> **V** a_{obj} [<CR>]
	a_{obj} = Object name : 0-9 , A-Z , a-z (1 digit)
Example:	<ESC>V1

Each object (barcode-, text- and logo-object) can be defined as a variable object. It allows to allocate to this object a new content outside of the layout block without having to transfer the entire data record anew.

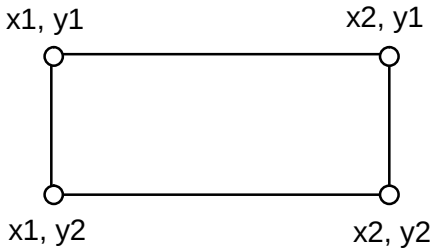
The original object defined in the layout block pre-defines the maximum field size. Thus, this object can later only be filled with the same maximum number of data. The transfer of new data is effected by the control sequence:

<ESC>v$a_{obj.}$	for text- / barcode-objects	(see chapter 1.2.10, page 10)
<ESC>l$a_{obj.}$	for logo-objects	(see chapter 1.2.6, page 8)

1.3.11 Lines and boxes

Object sequence:	<**ESC**> **X** *d..*$_{X1}$ **;** *d..*$_{Y1}$ **;** *d..*$_{X2}$ **;** *d..*$_{Y2}$ **;** *d..*$_{Width}$ [;*d..*$_{Fill}$] [<CR >]
	d..$_{X1}$, *d..*$_{Y1}$ *:* Coordinate 1 : 1 ... Label width
	d..$_{X2}$, *d..*$_{Y2}$ *:* Coordinate 2 : 1 ... Label height
	d..$_{Width}$ *:* Line width: Number of dots: 1 ... Label width
	d..$_{Fill}$ *:* Fill boxes

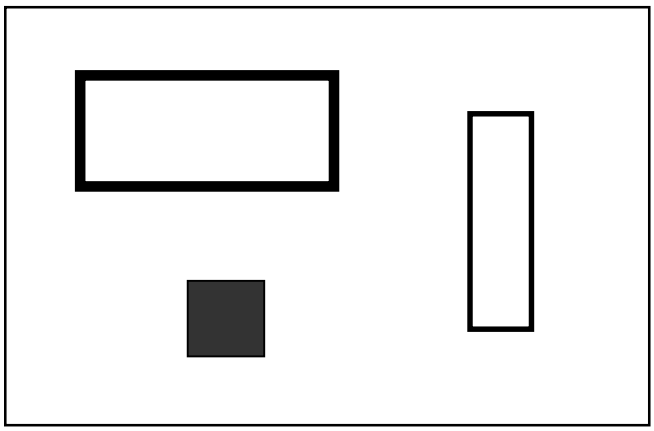
By this object sequence lines as well as boxes can be created. In case that [x1,y1] and [x2,y2] define a diagonal, a box is printed, otherwise a horizontal or vertical line.



It has to be observed that the 2nd pair of coordinates shall have a higher value than the first one. Lines are widened inwards.

Example:

```
<STX>
<ESC>X20;20;250;150;6
<ESC>X300;40;350;330;3
<ESC>X120;220;200;300;1;1
<EOT>
```



1.3.12 Image background

Object sequence:	<**ESC**> **Y** $HH_{Image\ data}$ <**CR** >
	$HH_{Image\ data}$ Image data in binary form
	Each bit of a byte represents 1 dot
	Bit = 0: do not print dot, = 1: print dot
	Data bit: 7 6 5 4 3 2 1 0
	Dot: 1 2 3 4 5 6 7 8

Normally the image background respectively the image memory is blank (white). By this sequence, the image background can be defined. As the image sequences are directly transferred into the image memory and object attributes cannot be applied, it is not a true object sequence in this case. Instead, the entire image area can be defined without being restricted by the 64 K limit and without using the main memory. As the data record is immediately processed, only 1 image line has to be buffer stored in the input memory.

In the image area, objects are normally processed by switching operation. Image sequences are overwriting the image memory. Thus, image sequences shall be transferred to the printer prior to objects.

Only 1 image line is transferred per ESC-sequence. With every sequence the next image line is addressed and written automatically. The maximum number of image sequences is identical to the image height. Too many image sequences will be ignored and a WARNING message is made

The number of image data per sequence is calculated by the image width / 8. Digits after the decimal point will be rounded up. The image width is identical to the card width, unless no other image area has been explicitly defined. The incorrect number of image data results in unpredictable errors, and thus has to be avoided.

Object sequence::	<**ESC**> **Z***d..* <**CR** >
	d = number of empty image lines

To reduce the multitude of data, image lines in which nothing will be printed can be skipped by <**ESC**>**Z***d*. The parameter *d* indicates the number of empty image lines.

Example:
Card width = 800 dots: 100 image data are to be sent per image line.
Card width = 480 dots: A maximum of 480 image sequences can be sent to the printer.

1.4 Objects

An object will be defined by a special object sequence. This special object sequence always has to be the last object sequence within an object block. All specifications in the object sequences before are related to this object.

The PLUS distinguishes the following objects:

<ESC> T ... TEXT-object

<ESC> B ... BARCODE-object

<ESC> L ... LOGO-object

Following the single objects are described in detail.

1.4.1 Text-object

Object sequence: <**ESC**> ***T****font type* ***;*** *text data* [<CR>]

font type	Text examples:
COURI06f	**Courier 06 bold**
COURI08f	**Courier 08 bold**
COURI10f	**Courier 10 bold**
COURI12f	**Courier 12 bold**
COURI14f	**Courier 14 bold**
ARIAL08f	**Arial 08 bold**
ARIAL09f	**Arial 09 bold**
ARIAL10f	**Arial 10 bold**
ARIAL12f	**Arial 12 bold**
ARIAL14f	**Arial 14 bold**
ARIAL16f	**Arial 16 bold**
ARIAL18f	**Arial 18 bold**

Text-objects are defined by specifying the drawing font and the corresponding drawing string. In case of incorrect or missing font specification, Courier 08 bold is used as default value.

Via menu CHARACTER/LCD / SCHEDULE, it is possible to select between ANSI-, ASCII- or multilingual character set chart.

Example:

<ESC>**T**ARIAL18F**;t**ext string

1.4.2 Logo-object

Object sequence: <**ESC**> **L** *d..*$_{\text{Width}}$ **;** *d..*$_{\text{Height}}$ **; l ;** *HH*$_{\text{Logo data}}$ <**CR** >

d..$_{\text{Width}}$ *:*	Logo-width : Number of dots in X-direction
d..$_{\text{Height}}$ *:*	Logo-height : Number of dots in Y-direction
l	Logo type (small L)
HH$_{\text{Logo data}}$	Logo data in binary form Each bit of a byte represents 1 dot Bit = 0: do not print dot, = 1: print dot Data bit: 7 6 5 4 3 2 1 0 dot: 1 2 3 4 5 6 7 8
<CR>	has to be placed directly behind the logo data !!

A logo is defined as a freely programmable image, which bitmap is transferred to the printer as data. As for other objects, the form and position of a logo-object can also be defined in detail using the corresponding object sequences.

Example:

Logo	Bit: 7654 3210	Hex value	decimal
....n...	0000 1000	08	8
....n...	0000 1000	08	8
nn..n...	1100 1000	C8	200
..n.n...	0010 1000	28	40
...n....	0001 0000	10	16

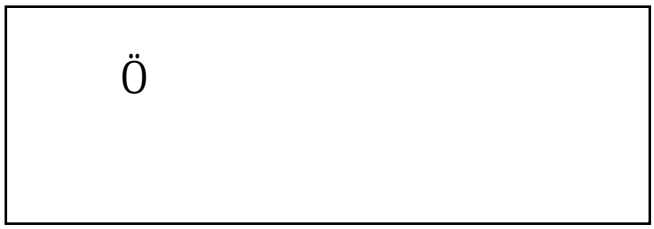
<STX>

<ESC>G50<ESC>I35<ESC>R0<ESC>C4<ESC>D4

<ESC>L8;5;|;08_H 08_H $C8_H$ 28_H 10_H<C_R>

<EOT>

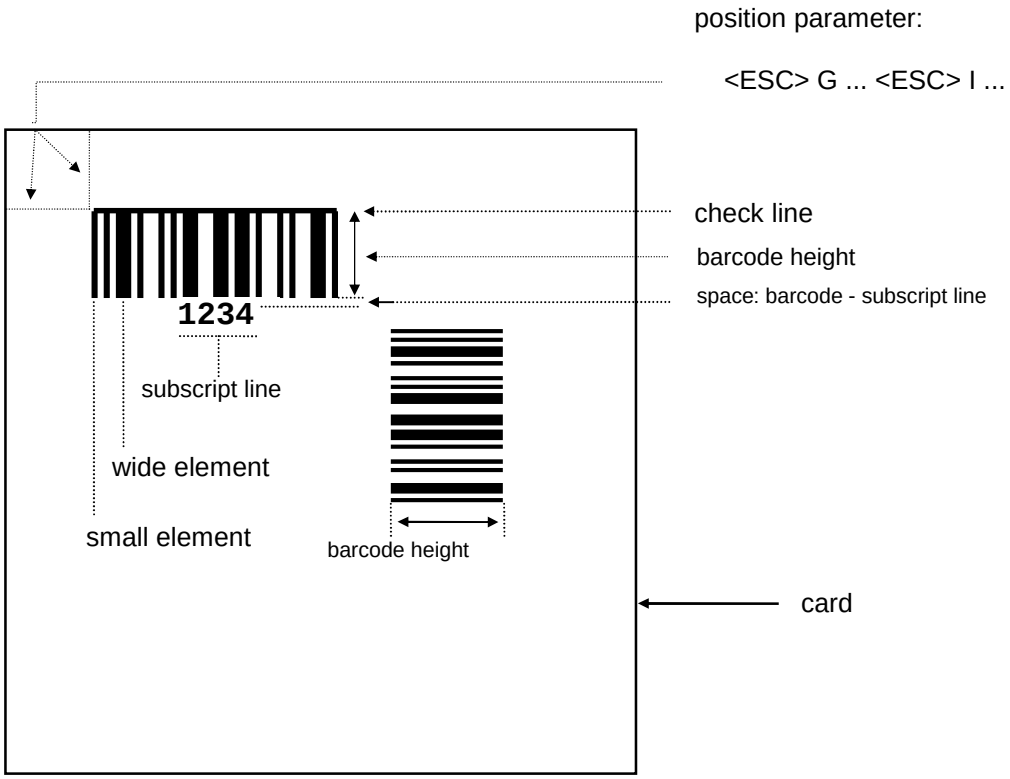
Logo-object sequence in BASIC notation:
CHR$(27)+"L8;5;|;"+CHR$(8)+CHR$(8)+CHR$(200)+CHR$(40)+CHR$(32)+<CR>



1.4.3 Barcode-object

1.4.3.1 Introduction

Barcode-objects can be defined in their form and function by a multitude of parameters. The sketch below illustrates these parameters.



By the following object sequence a barcode can be integrated:

<**ESC**> **B**	S_{Type} **;** [P*arameter* **;** ...] **>** B*arcode data* < **CR** >
	S_{Type} = Name of barcode type :

S_{Type}	Corresponding barcode:
C_25_I	Code 2 of 5 Interleaved
C_39	Code 39
C_128	Code 128
EAN8	EAN-8
EAN13	EAN-13
EAN128	EAN-128

The first parameter must always be the desired barcode type. Then, further optional parameters may follow. Every parameter input starts with a key letter and ends with a semicolon ' **;** '. The beginning of the barcode data is indicated with the sign ' **>** '.

Optional parameters:

Further barcode specific parameters can be specified. In case of no specification, a default value is used. The sequence of the parameter input (barcode, subscript) is arbitrary. The parameter specifications should be decimal numbers or text strings. Following optional parameters can be used:

Parameters for barcode:

H *d*..	Barcode **Height** in number of dots (1/12 mm) Default: H120
B *d*..	**Width** for a small element in number of dots Default: B3
R 2 | 3 | 5	**Ratio** = proportion of wide element to small element 2 = 2 : 1 3 = 3 : 1 5 = 5 : 2 Default: R3
K 0 | 1	**Check line**: 0 = WITHOUT, 1 = WITH check line Default: K0
Z 0 | 1 | 2	**Check digit**: 0 = WITHOUT, 1 | 2 = WITH check digit **Z1**: check digit does not appear in subscript line **Z2**: check digit will be printed in subscript line, too For some barcodes an additional check digit can optionally be generated. For barcodes in which the check digit is a regular part, it is not possible. Default: Z0
S 0 | a | b | c	**Start code,** only code 128 and EAN 128 Default: S0

Parameters for subscript line:

A subscript line can automatically be printed for each barcode. It is printed centred underneath the barcode. The subscript line can be defined in detail via the following parameters:

A *dddd*	**Subscript line: Object attributes** Default: A0000
T *font type*	**Subscript line: Drawing font** All fonts implemented in the printer can be used. Default: COURI08f
C *d*..	**Subscript line: Y-Factor** Default: C1
D *d*..	**Subscript line: X-Factor** Default: D1
F *d*..	**Subscript line: Character space** (Number of dots) Default: F1
P *d*..	**Subscript line: Space to barcode** (Number of dots) Reference point is the last dot line of the barcode symbol. P0 would print the subscript line flush below the barcode symbol. A negative value places the subscript line inside of the barcode symbol, whereas positive values move the subscript line away from the barcode.
P %	**Do not print subscript line**. Default: P1

Following object sequences can also be used to form and position the barcode symbol:

<ESC>A ...	Object attributes
<ESC>G ...	X-Position
<ESC>I ...	Y-Position
<ESC>Q ...	Numerical stepping
<ESC>R ...	Rotation
<ESC>V ...	Variable object

Example: **Card with Code 39.**

```
<STX>
<ESC>G50<ESC>I40<ESC>R0
<ESC>BC_39;H70;K1;B3;R2;Z1;TARIAL20f;F2;P1;>CODE39
<EOT>
```

1.4.3.2 Barcode: Code 2 of 5 Interleaved

Object sequence: <**ESC**> **BC_2o5_I;** [P*arameter* **;** ...] **>** B*arcode data* [CR]

Parameter :	**Barcode:**
H 1 ... 1000	Barcode **Height** in number of dots Default: H120
B 1 ... 99	**Width** of a small element in number of dots Default: B3
R 2 | 3 | 5	**Ratio** = proportion of wide element to small element 2 = 2 : 1 , 3 = 3 : 1 , 5 = 5 : 2 Default: R3
K 0 | 1	**Check line**: 0 = WITHOUT , 1 = WITH check line Default: K0
Z 0 | 1 | 2	Generation respectively printout of the check character in the subscript line 0 = do not generate check character 1 = check character in barcode, but not in subscript line 2 = check character in barcode as well as in subscript line Default: Z0
	Subscript line:
A *dddd*	**Subscript line: Object attributes** (see Object sequence: Object attributes) Default: A0000
T *fonttype*	**Subscript line: Character font** Default: COURI08f
C 1 ... 255	**Subscript line: Y-Factor** Default: C1
D 1 ... 255	**Subscript line: X-Factor** Default: D1
F 1 ... 255	**Subscript line: Character space** (Number of dots) Default: F1
P -99 .. +99	**Subscript line: Space to barcode** Reference point is the last dot line of the barcode symbol. A negative value places the subscript line inside of the barcode symbol, whereas positive values move the subscript line away from the barcode.
P %	**Do not print subscript line** Default: P1

Barcode data:

Valid:	0 – 9 (only numerical numbers) Code 2 of 5 Interleaved prescribes an even number of digits. Should the number of digits inputted be odd, a leading zero is automatically added
Number:	any number of valid characters, according to print area

Symbol structure, Symbol width:

Rest zone, start character, effective character, [check character], stop character, rest zone

Rest zone:	blank, width at least 10 times the element width
Start-, Stop character:	automatically generated by the programme
Check character:	can be generated automatically by the programme, but is not a regular component of Code 2 of 5 Interleaved. The check character is calculated from a Modulo-10 - check sum with a weight of 3.

Margin character, effective character, check character includes according to their ratio the following number of element widths:

Ratio	Margin character together	effective character, check character
R 2:1	8	7
R 3:1	9	9
R 5:2	17	16

Example: **Code 2 of 5 Interleaved**

```
<STX>
<ESC>G50<ESC>I40<ESC>R0
<ESC>BC_2o5_I;H70;K0;B3;R3;Z1;TARIAL20;F2;P1;>12345678
<EOT>
```

1.4.3.3 Barcode: Code 39

Object sequence: <**ESC**> **BC_39;** [P*arameter* **;** ...] **>** B*arcode data* [CR]

Parameter :	**Barcode:**
H 1 ... 1000	Barcode **Height** in number of dots Default: H120
B 1 ... 99	**Width** of a small element in number of dots Default: B3
R 2 | 3 | 5	**Ratio** = proportion of wide element to small element 2 = 2 : 1 , 3 = 3 : 1 , 5 = 5 : 2 Default: R3
K 0 | 1	**Check line**: 0 = WITHOUT , 1 = WITH check line Default: K0
Z 0 | 1 | 2	Generation respectively printout of the check character in the subscript line 0 = do not generate check character 1 = check character in barcode, but not in subscript line 2 = check character in barcode as well as in subscript line Default alternative: Z0
	Subscript line:
A *dddd*	**Subscript line: Object attributes** (see Object sequence: Object attributes) Default: A0000
T *fonttype*	**Subscript line: Character font** Default: COURI08f
C 1 ... 255	**Subscript line: Y-Factor** Default: C1
D 1 ... 255	**Subscript line: X-Factor** Default: D1
F 1 ... 255	**Subscript line: Character space** (Number of dots) Default: F1
P -99 .. +99	**Subscript line: Space to barcode** Reference point is the last dot line of the barcode symbol. A negative value places the subscript line inside of the barcode symbol, whereas positive values move the subscript line away from the barcode.
P %	**Do not print subscript line** Default: P1

Barcode data:

Valid: 0 - 9 , A - Z , space - . $ / + %

Number: any number of valid characters, according to print area

Symbol structure, Symbol width:

Rest zone, margin character, effective character, [check character], margin character, rest zone

Rest zone: blank, width at least 10 times the element width

Margin character: automatically generated by the programme

Check character: can be generated automatically by the programme, but is not a regular component of the Code 39. The check digit is calculated from a Modulo-43 -check sum.

Margin character, effective character, check character includes according to their ratio the following number of element widths:

R 2:1 = 13 elements, R 3:1 = 16 elements, R 5:2 = 29 elements

Example: Code 39

```
<STX>
<ESC>G50<ESC>I40<ESC>R0
<ESC>BC_39;H70;K0;B3;R3;Z1;TARIAL20;F2;P1;>CODE39
<EOT>
```

CODE39

1.4.3.4 Barcode: Code 128

Object sequence: <**ESC**> **BC_128 ;** [P*arameter* **;** ...] **>** B*arcode data* [CR]

Parameter :	**Barcode:**
H 1 ... 1000	Barcode **Height** in number of dots Default: H120
B 1 ... 99	**Width** of a small element in number of dots Default: B3
K 0 | 1	**Check line**: 0 = WITHOUT , 1 = WITH check line Default: K0
Z 1 | 2	Printout of start code and check character in subscript line 1 = do not print, 2 = print Default: Z1
S 0 | a | b | c	**Start code**: (The start code can also be defined by the first barcode data) a = Start code for character set A, otherwise no further conversion b = Start code for character set B, otherwise no further conversion c = Start code for character set C The programme converts the digit pairs. In case of an odd number of digits, a leading '0' is added. 0 = automatic compression: By switching in each case to the best appropriate character set a barcode symbol with the shortest possible width is created. Default: S0
	Subscript line:
A *dddd*	**Subscript line: Object attributes** (see Object sequence: Object attributes) Default: A0000
T *fonttype*	**Subscript line: Character font** Default: COURI08f
C 1 ... 255	**Subscript line: Y-Factor** Default: C1

D 1 ... 255	**Subscript line: X-Factor** Default: D1
F 1 ... 255	**Subscript line: Character space** (Number of dots) Default: F1
P -99 .. +99	**Subscript line: Space to barcode** Reference point is the last dot line of the barcode symbol. A negative value places the subscript line inside of the barcode symbol, whereas positive values move the subscript line away from the barcode.
P %	**Do not print subscript line** Default: P1

Barcode data:

Code 128 distinguishes between 3 different character sets. With the first barcode character or the parameter S*x* the corresponding character set can be chosen. If the first character is not a valid start code, the value of the parameter S*x* is took over and the corresponding start code is used.

Valid start code characters:
Start code A : ç (135 decimal)
Start code B : ê (136 decimal)
Start code C : ë (137 decimal)

Subsequent characters:

Character set C :	0 - 9, an even number of characters is prescribed, otherwise a leading '0' is added
Character set A,B:	all ASCII-characters in the range 32 - 127 : 0 - 9 , A - Z , a - z space character ! " # $ % & ' ( ) * + , - . / : ; < = >? @ [\ ] ^ _ { | } ~ DEL

Control characters in

Character set A	Character set B	ASCII-code to be sent
FNC3	FNC3	Ç (128 decimal)
FNC2	FNC2	ü (129 decimal)
SHIFT	SHIFT	é (130 decimal)
Code C	Code C	â (131decimal)
Code B	FNC4	ä (132 decimal)
FNC4	Code A	à (133 decimal)
FNC1	FNC1	å (134 decimal)

Automatic: as character set B, but without : SHIFT , Code A, Code C

Number: any number of valid characters, according to print area.

Symbol structure, Symbol width:

Rest zone, start code, effective character, check character, stop character, rest zone

Rest zone: blank, width at least 10 times the element width

Start code: as mentioned above

Stop code: automatically generated by the programme

Check character: automatically generated by the programme

Start code, effective character, check character consist each of 11 element widths. The stop character has 13 times the width of the small element.

Example: **Code 128**

```
<STX>
<ESC>G50<ESC>I40<ESC>R0
<ESC>BC_128;H70;K0;B3;TARIAL20;F2;P1;>Code128
<EOT>
```

Code128

1.4.3.5 Barcode: EAN-8

Object sequence: <**ESC**> **BEAN8;** [P*arameter* **;** ...] **>** B*arcode data* [CR]

Parameter :	**Barcode:**
H 1 ... 1000	Barcode **Height** in number of dots Default: H120
B 1 ... 4	**Width** of a small element in number of dots Default: B3 (In addition the whole barcode symbol can be multiplied via object sequences <ESC> C … and <ESC> D …in height and width)
K 0 | 1	**Check line**: 0 = WITHOUT , 1 = WITH check line Default: K0
	Subscript line:
P 1	Print subscript line
P %	**Do not** print subscript line Default: P1

Barcode data :

Valid:	0 - 9 (only numerical numbers)
Number:	The barcode consists of 8 digits including check digit. If 8 digits are inputted, the 8th digit is checked as check digit. If 7 digits are inputted, the check digit is calculated and added.

Symbol structure, Symbol width:

Rest zone, margin character, 4 effective digits, separation character, 4 effective digits, margin character, rest zone

Rest zone:	blank, width at least 10 times the element width
M*argin character*:	automatically generated by the programme

An EAN-8 - barcode symbol consists of 67 element widths in total.

Example: **EAN-8**

```
<STX>
<ESC>G50<ESC>I40<ESC>R0
<ESC>BEAN8;H70;K0;B3;>4012345
<EOT>
```

1.4.3.6 Barcode: EAN-13

Object sequence: <**ESC**> **BEAN13;** [P*arameter* **;** ...] **>** B*arcode data* [CR]

Parameter :	**Barcode:**
H 1 ... 1000	Barcode **height** in number of dots Default: H120
B 1 ... 4	**Width** of a small element in number of dots Default: B3 (In addition the whole barcode symbol can be multiplied via object sequence: <ESC> C ... and <ESC> D ... in height and width)
K 0 | 1	**Check line**: 0 = WITHOUT , 1 = WITH CHECK line Default: K0
	Subscript line:
P 1 **P %**	Print subscript line **Do not** print subscript line Default: P1

Barcode data :

Valid:	0 - 9 (only numerical numbers) A blank before the 1st digit has the effect that the 1st digit of the subscript line is printed left of the barcode. Without a blank all 13 digits are printed underneath the barcode.
Number:	The barcode consists of 13 digits including check digit. If 13 digits are inputted, the 13th digit is checked as check digit. If 12 digits are inputted, the check digit is calculated and added.

Symbol structure, Symbol width:

Rest zone, margin character, 7 effective digits, separation character, 6 effective digits, margin character, rest zone

Rest zone:	blank, width at least 10 times the element width
M*argin character*:	automatically generated by the programme

An EAN-13 - barcode symbol consists of 95 element widths respectively 106 element widths in total, in case the first digit is printed left of the barcode.

Example: EAN-13

```
<STX>
<ESC>G50<ESC>I40<ESC>R0
<ESC>BEAN13;H70;K0;B3;> 401234567890
<EOT>
```

1.4.3.7 Barcode: EAN-128

Object sequence: <**ESC**> **BEAN128 ;** [P*arameter* **;** ...] **>** B*arcode data* [CR]

Parameter :	**Barcode:**
H 1 ... 1000	Barcode **Height** in number of dots Default: H120
B 1 ... 99	**Width** of a small element in number of dots Default: B3
K 0 | 1	**Check line**: 0 = WITHOUT , 1 = WITH CHECK line Default: K0
Z 1 | 2	Print of the start code and check character in subscript line 1 = do not print , 2 = print Default: Z1
S 0 | a | b | c	Start code: (The start code can also be defined by the first barcode data) a = Start code for character set A, otherwise no further conversion b = Start code for character set B, otherwise no further conversion c = Start code for character set C The programme converts the digit pairs. In case of an odd number of digits, a leading '0' is added. 0 = automatic compression: By switching in each case to the best appropriate character set a barcode symbol with the shortest possible width is created. Instead of S0 this parameter can also be dropped. Default alternative: S0
	Subscript line:
A *dddd*	**Subscript line: Object attributes** (see Object sequence: Object attributes) Default: A0000
T *font type*	**Subscript line: Drawing font** Default: COURI08f
C 1 ... 255	**Subscript line: Y-Factor** Default: C1
D 1 ... 255	**Subscript line: X-Factor** Default: D1
F 1 ... 255	**Subscript line: Character space** (Number of dots) Default: F1
P -99 .. +99	**Subscript line: Space to barcode** Reference point is the last dot line of the barcode symbol. A negative value places the subscript line inside of the barcode symbol, whereas positive values move the subscript line away from the barcode.
P %	**Do no** print subscript line Default alternative: P1

Barcode data :

EAN-128 is a variant of the Code 128. In EAN-128 the code character FNC1 is always behind the start code. This code character is automatically generated by the printer programme. Within the barcode string, FNC1 can also be used as a closing character for variable fields. In this case it should be also inputted as data.

Valid start code characters:
Start code A : ç (135 decimal)
Start code B : ê (136 decimal)
Start code C : ë (137 decimal)

According to the character set (Sx) the following characters are valid:

Character set C (Sc):	0 - 9
Character set A/B (Sa/Sb):	all ASCII-characters in the range 32 - 127 0 - 9 , A - Z , a – z space character ! " # $ % & ' ( ) * + , - . / : ; < = >? @ [\ ] ^ _ { | } ~ DEL

Control Characters in

Character Set A	Character Set B	ASCII-Code to be sent
FNC3	FNC3	Ç (128 decimal)
FNC2	FNC2	ü (129 decimal)
SHIFT	SHIFT	é (130 decimal)
Code C	Code C	â (131 decimal)
Code B	FNC4	ä (132 decimal)
FNC4	Code A	à (133 decimal)
FNC1	FNC1	å (134 decimal)

automatic compression (S0) :	ASCII-characters 32-127 + FNC1
Number:	max. 48 effective characters, but the number of code characters (start-, stop-, control-, check- and coded effective characters) shall not exceed 35.

Symbol structure, Symbol width:

Rest zone, start code, FNC1, effective character, check character, stop character, rest zone

Rest zone:	blank, width at least 10 times the element width
Start code:	as mentioned above
FNC1:	this code character will be inserted automatically
Stop code:	automatically generated by the programme
Check character:	automatically generated by the programme (Modulo 103)

Start code, effective character, check character consist each of 11 element widths. The stop character has 13 times the width of the small element.

Example: EAN 128

```
<STX>
<ESC>G50<ESC>I40<ESC>R0
<ESC>BEAN128;H70;K0;B3;TARIAL20;F2;P1;>106593å211678
<EOT>
```

1.4.3.8 Barcode: PDF-417

Object sequence: <**ESC**> **BPDF417 ;** [*Parameter* **;** ...] **>** **B***arcode data* [CR]

Parameter:	**Meaning:**
L	**Specification of ERROR-level**. A PDF-417 includes an error detection code, i. e. a kind of check-digit. Supplementary an error correction code can be added in order to restore illegible code words. The higher the ERROR-level, the higher the restore coefficient. Nevertheless, the coefficient of the ERROR-level should be chosen moderately as a high coefficient does not only enlarge the barcode symbol but also reduces the number of effective data. Recommendation: 10% of the number of code words for the effective data should be used for the error detection. The following ERROR-levels are possible: L0 : only error detection, no error correction L1 : error detection + 2 Code words for correction L2 : " + 6 " L3 : " + 14 " L4 : " + 30 " L5 : " + 62 " L6 : " + 126 " L7 : " + 254 " L8 : " + 510 " (A PDF-symbol includes maximally 925 code words) The specification of the ERROR-Levels can also be made in %: L%d : d specifies the percentage of the correction code words to the effective data words (Example: L%10). If no L specification is made, L%10 will be used. Should a fix matrix for the barcode symbol be defined, the correction coefficient will be increased so far as free code words remain.
C	**Number of code columns** Defines the width of the PDF-symbol. A PDF-symbol consists of margin characters, line indicators and the code words for the effective data (= code columns). Each code word consists of 17 modules. The minimum module width should be 1/6 mm. Margin characters and line indicators together have a width of 69 modules. Possible data for PDF-code columns: **C1 ... C30**. This parameter is optional. In case no parameter is defined, the R-parameter must be defined! The programme determines the necessary width depending on the number of effective data as well as the chosen correction coefficient.
R	**Number of code lines** Defines the height of the PDF-symbol. A PDF-symbol consists of several stacked barcode lines: 3 ... 90. Possible data for PDF-code columns: **R3 ... R90**. This parameter is optional. In case no parameter is defined, the C-parameter must be defined! The programme determines the necessary height depending on the number of effective data as well as the chosen correction coefficient.
R, C	**Definition of a symbol matrix** If the C- and R-parameter are defined, the symbol size is always equally sized according to the definition. If less efficient data are defined than could be encoded in the symbol, it will be filled up with fill-in characters. Though, too many effective data would provoke an abnormal termination and error message on the screen.

T **Truncated PDF**
Layout of the PDF-symbol in a shortened form. In this case, the right line indicator as well as the right margin character are presented by a module and the number of modules for line indicators and margin characters is only 35 modules. This short form should not be used in cases where a high error correction is necessary.
T0 = normal mode (= pre-set value)
T1 = Truncated PDF

W **Width of a module**
Defines the module width in number of dots. A module should have at least a width of 1/6 mm (2 dots with a 12-dot-thermal board).
Example.: W2 = 2 dots / module
Pre-set value is: W2.

H **Height of a module (barcode line)**
Defines the height of a barcode line in number of dots. The height of a barcode line should be at least 3-4 times the module width.
Example.: H6 = 6 dots / barcode line
Preset value is: H6.

D **PDF - Effective data**
According to the type of data, a varying number of data can be encoded in a PDF-symbol. In this connection the data are transferred into so called code words. A PDF-symbol can maximally include 925 code words. The number of code words used for error correction must be deduced from these 925 code words. In a code word following data can be encoded:
2 alphanumerical data
2,93 numerical data
1,2 binary data (value range: 0 - 255)
With an ERROR-level = 0, following data could maximally be encoded in a PDF-symbol:
1850 alphanumerical data
2710 numerical data
1108 binary data
The printer software provides the correct encoding of each data type. Optimization is effected in that way that the number of code words is kept as small as possible.

Input of effective data:

All data inside the value range 0 - 255 are valid.
To avoid conflicts with other printer control characters and to recognize the end of the effective data, all data which ASCII-value is less than 32, must be inputted in a special form:
\ddd (ddd = 3-digit decimal value)
\\ (for characters: "\")
Example.: \013 (= "CR")

PDF417 as variable data field

The code PDF 417 can be used as other barcodes as a variable barcode field.
See example mentioned below.

Error messages:

In case of conflicts regarding the generation of the PDF-symbol due to faulty parameters, the following error message is created:
ERROR #074
PDF417 - data

Possible error reasons:

- incorrect ERROR-level
- incorrect specification of number of columns, number of barcode lines
- number of effective data too high
- no D-parameter respectively no effective data
- neither C-parameter, nor R-parameter is defined
- total of data code words + number of correction code words bigger than 925

After this error message no further printout is possible.

Example:

<S_X>
<E_C>G020<E_C>I0020<E_C>V1
<E_C>BPDF417;L%10;C3;T0;W2;H8;
D This is a PDF417-Barcode.\013\010
F+D Feinwerk- und Drucktechnik\013\010<C_R>
<E_T>

In this case the PDF 417-barcode is additionally defined as a variable barcode field (#1). The barcode width is defined to 3 code columns (= 3*17 + 69 = 120 modules). Each module has a width of 2 dots --> barcode width = 120 * 2 =240 dots.

The program calculates the single code words and adds 10% of the total number of code words for correction purposes. Thus, the number of barcode lines is determined. The height of each barcode line is determined to 8 dots.

Variable barcode field:

To print an additional card with new PDF-data, the following variable control set can for example be transferred to the printer:
<E_C>v1; New PDF417-data.\013\010 <C_R>
<E_C>#1<C_R>

Only the PDF-data have been changed. The card layout is kept.

By input of the variable control set:

<E_C>v1;<C_R>

no PDF-symbol would be printed.

1.5 Preferred control sequences

1.5.1 Status message

Status request is possible via V24. The status report can be requested at any time. The printer sends its current status, unless a printout is effected at this time. In this case, the status will be sent before the printout of the next card is made. Each output line will be ended by <CR><LF> .

Status request, by <ESC>!<ENQ> (<ENQ>=05h respectively CTRL/E):

TFK+ aS02/V0.16	- programme version
=00	- printer status (ss)
#0000	- number of cards
*65536	- remaining input memory
/054	- from here error listing in case of error
/024	

Printer status ss:

08	- at least one more card to print
10	- card can be taken out
20	- data record is stored
80	- stepping motor in motion

Short status, by <ESC>!<ACK> (<ACK>=06h respectively CTRL/F):

=00/000 - Printer-status + error code (highest priority)

Printer status ss:

02	- after reset
08	- at least one more card to print
10	- card can be taken out
20	- data record is stored
80	- stepping motor in motion

1.5.2 RFID status

If the optional RFID unit is integrated, it is possible to ask for the version of the transponder electronic. E. g.: „Moby-D V4.5"

If the RFID unit is not integrated, the message " Not Present " is displayed on the screen.

RFID status, by <ESC>!<BEL> (<BEL>=07h respectively CTRL/G):

1.5.3 Software reset

The control sequence <ESC>!! results in a printer-reset. In this case, the printer will be re-started. All data still in the input memory will be deleted.

2 Appendix

2.1 Appendix A: Error messages

2.1.1 Error level 1 - Warning

Message	Reason	Measure
WARNING #002 Control Sequence	Incorrect value after <ESC>b: invalid character Value < 80	Correct sequence
WARNING #003 Control Sequence	Incorrect value after <ESC>c: invalid character Value < 64 Value > number of dots thermal print head	Correct sequence
WARNING #004 Control Sequence	Incorrect value after <ESC>d: invalid character	Correct sequence
WARNING #005 Control Sequence	Incorrect value after <ESC>e: invalid character	Correct sequence
WARNING #010 Control Sequence	Incorrect value after <ESC>j: invalid character	Correct sequence
WARNING #011 Control Sequence	Incorrect value after <ESC>k: switch not equal to 0 or 1 Transfer printing not allowed, as not logged-on in set-up	**à** correct sequence **à** Service
WARNING #012 Control Sequence	Incorrect value after <ESC>l: invalid character	Correct sequence
WARNING #014 Control Sequence	Incorrect value after <ESC>n: Value > 9	Correct sequence
WARNING #022 Control Sequence	Incorrect value after <ESC>v: invalid character	Correct sequence
WARNING #023 Control Sequence	Incorrect value after <ESC>w: invalid character	Correct sequence
WARNING #024 Control Sequence	Incorrect value after <ESC>x: invalid character	Correct sequence
WARNING #025 Control Sequence	Incorrect value after <ESC>y: incorrect syntax invalid character X/Y-position = 0 image width/image height <16 image width > card width	Correct sequence
WARNING #026 Control Sequence	Incorrect value after <ESC>z: invalid character	Correct sequence
WARNING #027 Control Sequence	Invalid control sequence: Incorrect small letter after <ESC>	Correct sequence
WARNING #028 obj. indicator ?	Object identification has not been found	Carry out identification of object via <ESC>V Allocate new data to object with similar identification via <ESC>v
WARNING #029 Var. logo length	Number of var. logo data not identical to original logo	Var. logo will be ignored. Set var. logo to same format as original logo.
WARNING #031 Object sequence	Incorrect value after <ESC>A: invalid character	Correct sequence
WARNING #032 Object sequence	Incorrect value after <ESC>B: invalid character	Correct sequence

Message	Reason	Measure
WARNING #034 Object sequence	Incorrect value after <ESC>D: X-Factor = 0 X-Factor > 255	Correct sequence. Default alternative: single character width
WARNING #036 Object sequence	Incorrect value after <ESC>F: Distance between characters>255	Correct sequence. Default alternative: space between characters = 0
WARNING #037 Object sequence	Incorrect value after <ESC>G: X-position = 0 X-position outside of image area	Correct sequence. X-position should not be bigger than image width. Default alternative: X-position = 1
WARNING #039 Object sequence	Incorrect value after <ESC>I: Y-position = 0 Y-position outside of image area	Correct sequence. Y-position should not be bigger than image height. Default alternative: Y-position = 1
WARNING #043 Object sequence	Incorrect value after <ESC>M: invalid character	Correct sequence
WARNING #047 Object sequence	Incorrect specification after <ESC>Q: Stepping value or stepping cycle missing Stepping value > +/- 9 Stepping cycle = 0 or > 255 Switch for zero suppression not equal to 0 or 1 Start of stepping field < 1 Size of stepping field < 0	Correct sequence
WARNING #048 Object sequence	Incorrect value after <ESC>R: Rotation not equal to 0/90/180/270 degrees	Correct sequence. Default alternative: Rotation 0 degrees
WARNING #050 Object sequence	Incorrect value after <ESC>T: invalid character	Correct sequence
WARNING #052 Object sequence	Incorrect specification after <ESC>V: Identification consists of more than 1 character	Correct sequence. Only 1 character allowed for identification. Default alternative: object specified without identification.
WARNING #054 Object sequence	Incorrect value after <ESC>X: invalid character	Correct sequence
WARNING #055 Image line length	Image line overflow. Too many <ESC>Y-sequences has been sent to the printer.	Check number of image-sequences. Number should not be bigger than number of image lines within image area.
WARNING #056 Image line <CR>	Too many or too little image data has been transferred within <ESC>Y.	Number of image data must correspond to image width / 8.
WARNING #057 Object sequence	Invalid object sequence: Incorrect capital letter after <ESC>	Correct sequence
WARNING #058 Object > 64 Kbytes	Object requires more than 64 Kbytes memory	Reduce object in size or divide into 2 objects.
WARNING #060 Font-Type	Incorrect <ESC>T-sequence: Chosen font type is not available in the printer.	Choose existing font type. Default alternative: COURI08F
WARNING #061 Barcode-Type	Incorrect <ESC>B-sequence: Chosen barcode type is not available in the printer.	Choose other barcode type or delete sequence. Default alternative: none, sequence will be ignored.
WARNING #062 Code 2of5 - Data	Incorrect <ESC>B-sequence: Unacceptable barcode data	Correct sequence

Message	Reason	Measure
WARNING #064 Code 128 - Data	Incorrect <ESC>B-sequence: Unacceptable barcode data	Correct sequence
WARNING #065 EAN 8 – Data	Incorrect <ESC>B-sequence: Unacceptable barcode data	Correct sequence
WARNING #066 EAN 13 - Data	Incorrect <ESC>B-sequence: Unacceptable barcode data	Correct sequence
WARNING #067 UPC-A - Data	Incorrect <ESC>B-sequence: Unacceptable barcode data	Correct sequence
WARNING #068 PDF417 - Data	Incorrect <ESC>B-sequence: Unacceptable barcode data	Correct sequence
WARNING #069 Barcode - Data	Incorrect <ESC>B-sequence: Unacceptable barcode data	Correct sequence
WARNING #070 Character?	Character cannot be allocated outside an ESC-sequence.	Character will be ignored. Check further error messages and correct them first.
WARNING #080 Object -> Image	Due to its measurements, object does not fit into image area	Object will be ignored. Place object anew in its X-/Y-position.
WARNING #081 Rotation memory	Not enough memory available to rotate object.	Object will be copied in its initial form into the image. Remedy: Reduce image memory or input memory. Otherwise extend memory (**à** Service)
WARNING #082 Working memory	Not enough memory available to edit objects.	Object will be ignored. Remedy: Reduce image memory or input memory. Reduce number of objects or do not use var. objects. Otherwise extend memory (**à** Service).
WARNING #083 Var. image memory	Not enough memory available for var. image memory.	All var. object will be ignored. Remedy: Reduce image memory or input memory. Reduce number of objects or do not use var. objects. Otherwise extend memory (**à** Service)
WARNING #084 Image memory	Not enough memory available for image area.	Image area will be reduced in its height according to memory capacity. Thus, possibly not all objects will be edited. Remedy: Optimize manually dimension and position of image area, reduce input memory or do not use var. objects. Otherwise extend memory (**à** Service)

2.1.2 Error level 2 - Error

Message	**Reason**	**Measure**
ERROR #142 Object sequence	Incorrect value after <ESC>L: invalid character	correct sequence
ERROR #159 Amount of objects	Max. 32 variable objects are allowed.	Reduce number of variable objects to max. 32. No further process of print job possible. Initialize printer anew.
ERROR #185 Input memory	Input memory over flown.	In case no interface handling is possible, enlarge input memory. Initialize printer anew.
ERROR #191 Logo...<CR>	End criterion of logo <ESC>L-sequence not recognized.	At the end of the logo sequence must be a <CR>. Check logo data respectively logo format. Initialize printer anew.
ERROR #192 RFID...<CR>	End criterion of Transponder Data <ESC>U-sequence not recognized.	At the end of the Transponder sequence must be a <CR>. Check transponder data respectively transponder format. Initialize printer anew.
ERROR #201 Ribbon	Transfer ribbon end or ribbon transportation error.	Insert new transfer ribbon or check ribbon transportation. Afterwards press FF-key. Note: bigger loops between the printer module and the ribbon unwinding device can also results in this error message.
ERROR #205 Transponder	no Transponder available	Insert transponder anew or check card transportation. Afterwards press **clear** key.
ERROR #206 Cards	Card feeder empty	Insert new cards in the card feeder. Afterwards press **clear** key.
ERROR #207 Cards	Trouble on card transport	Open upper part of printer and check card transportation. Afterwards press **clear** key.
ERROR #213 Card jam	Card jam recognized	Press **clear** key to remove card.

2.1.3 Error level 3 - Hardware

Message	**Reason**	**Measure**
HARDWARE #241 Print module open	Upper part of the printer not shut correctly	Check upper part of the printer for correct locking
HARDWARE #242 Platen?	thermal head lifting not operative	Initialize printer anew / Service
HARDWARE #243 serial IC	memory IC not present or defect	Service
HARDWARE #245 Transponder	Transponder-Device not ready	Service
HARDWARE #247 Invalid - CPLD	CPLD defect	Service
HARDWARE #248 aut Cards	Card feeder defect	Service

2.2 Appendix B: Commands

2.2.1 Control sequences

Control block	= {Control sequence 1 [Control sequence n]} n=[2...]
Control sequence	= {<**ESC**>x.... <**CR**>} x=[a|b|...|z]

Control sequence	**Function**	**Page**
<**ESC**> **b** *d..* [;*d..*] <**CR**>	Image height	7
<**ESC**> **c** *d..* <**CR**>	Image width	7
<**ESC**> **j** *d..* <**CR**>	Printing speed	6
<**ESC**> **k** *d..* [;0|;1] <**CR**>	Printer data (reset|set) 1: printout via print button 20: use transfer printing 40: Card feeder available 100: automatic card feeding	7
<**ESC**> **l** a_{Obj} ;*d..*$_{Width}$;*d..*$_{Height}$;*Data* <**CR**>	Variable object (logo)	8
<**ESC**> **n** *d..* <**CR**>	Country code	8
<**ESC**> **t** *d..* <**CR**>	Card feeding / Card output	9
<**ESC**> **u** *d..*$_{Offset}$;*d..*$_{Length}$;*a* $_{Action}$; [HH_{Data}] <**CR**>	Transponder data	9
<**ESC**> **v** a_{Obj} ;*Data* <**CR**> **Fehler! Textmarke nicht definiert.**	Variable object (Text, Barcode)	
<**ESC**> **w** [+/-]*d..* <**CR**>	Thermal print head heating time	10
<**ESC**> # *d..* <**CR**> # *d..* +	Print job and number of cards 7 Print without Start-/Stop-ramp	

Note:		**Examples:**
a	= 1 ASCII-character	<ESC>Va
d..	= Decimal number	<ESC>G120
s	= String (max. 8-digits)	<ESC>Tarial18f;Text line

2.2.2 Object sequences

Layout block	= { <**STX**> Object block 1 [Object block *n*] <**EOT**> } *n* = [2...]
Object block	= { [Object sequence *n*] Object } *n* = [1...]
Object sequence	= { <**ESC**> *X* ... [<CR>] } *X* = [A|C|...|Z]
Object	= { <**ESC**> **B**|**T**|**L** ... [<CR>] }

Object sequence	**Function**	**Page**
<**ESC**> **A** *d..*	Object attributes 1: invert 2: mirror on X-Axis 4: mirror on Y-Axis 10: switch off transparency	11
<**ESC**> **B** s_{Type}; *Parameter* ;**>***Data*	Barcode s_{Type} = Barcode-Type *Parameter* see ESC-sequences **>** = Barcode data	20
<**ESC**> **C** *d..*	Y-Factor	11
<**ESC**> **D** *d..*	X-Factor	12
<**ESC**> **F** *d..*	Distance between characters	12
<**ESC**> **G** *d..* [;**l**|;**r**|;**z**]	X-Position/Alignment/Reflection ;l: X-Alignment: left-justified ;r: X-Alignment: right- justified ;c: X-Alignment: centred	13
<**ESC**> **I** *d..* [**;l**|;**r**|;**z**]	Y-Position/Alignment/Reflection ;l: Y-Alignment: left-justified ;r: Y-Alignment: right- justified ;c: Y-Alignment: centred	14
<**ESC**> **L** $d..{}_{Width}$;$d..{}_{Height}$;**l ;***Data* <**CR**>	Logo	18
<**ESC**> **M** *Image Name;* <**CR**>	Internal Logo	14
<**ESC**> **Q** *Parameter*	Numerical stepping	14
<**ESC**> **R** [**0**|**90**|**180**|**270**]	Rotation	15
<**ESC**> **T** s_{Type} ;*Data*	Text	18
<**ESC**> **U** $d..{}_{Offset}$ *;* $d..{}_{Length}$ *;* HH_{Data} <**CR**>	Write transponder	15
<**ESC**> **V** *a*	a_{Obj} Object name	16
<**ESC**> **X** $d..{}_{x1}$;$d..{}_{y1}$;$d..{}_{x2}$;$d..{}_{y2}$;$d..{}_{Width}$ [;$d..{}_{Fill}$]	Line / Frame	16
<**ESC**> **Y** *Data* <**CR**>	Data for 1 Image line	17
<**ESC**> **Z** *d*	Number of empty image lines	17

2.2.3 Preferred Sequences

<**ESC**> **!** <**05**h>	Status (full status)	33
<**ESC**> **!** <**06**h>	Status (shortened status)	33
<**ESC**> **!** <**07**h>	RFID status	33
<**ESC**> **! !**	Software reset	33

F+D Feinwerk- und Drucktechnik GmbH
Kirchenstr. 38
69239 Neckarsteinach, Germany

Phone: +049 (0) 06229 / 700-0
Fax: +049 (0) 06229 / 700-67
Mail: hotline@FuDdruck.de
Web: www.FuDdruck.de