Programming Manual

A+ series

Mach 4

PX Print Module

EOS series

XD-series

XC- series

Hermes+

J-Script and abc for cab printers

Edition 7a

JScript - the programming language for cab printers.

The usage of all described functions in this manual requires firmware version 3.17 or higher.
This is a generic manual which describes the commands for different printer models,which means that it may contain descriptions or explanations of features which are not available on every printer model.

cab Programming Manual

valid for following printer types:

A+ -Series ™
XD -Series ™
XC -Series ™
Mach 4 ™
PX -Print Module ™
Hermes+ -Series ™
and all printing systems based on the cab „X2“ board
as well as the
EOS - Series™

cab Produkttechnik GmbH & Co KG
Wilhelm Schickard Str. 14
76131 Karlsruhe / Germany

Tel: +49 - 721-6626-0
Fax:+49 - 721-6626-239
Email: support@cab.de
http://www.cab.de

Table of contents

Chapter 1: Introduction

Introduction

IMPORTANT : *We highly recommend to read the introduction first !!*

• The described commands and sequences are tested and approved with original cab printers.
cab Produkttechnik can not guarantee that all functions are available on OEM products.

• All sample labels are created with a 300 dpi printer,

• All measurements are in millimeters for the usage in international markets. Label positions have to be recalculated if the printer is set to „country = USA“, if no measurement command is transmitted.

• Some described functions are only available if your printer contains the actual firmware. We recommend to download and install the **actual firmware** release from our website at:

http://www.cabgmbh.com

• We tried our best to write an easy understandable programmer´s manual which should contain every possible function of cab printers.
Multiple different methods have been used to make sure that every shown example works properly and a few proof reads have been done to avoid any error in this manual.
Nevertheless - we would appreciate your comments, where more explanation is required and where we have to do things better. Every comment is welcome and will influence our future work.
And if you find any error,- then please let us know. Thank you for your help !

Nomenclature, Syntax of the commands

• All commands are accepted when the line end identifier is transmitted, with the exception of ESC commands, they are processed as soon as the required character is received.

• Carriage returns are not displayed in the headlines and not in the example files of this manual, to keep a better overview. Carriage Returns (ASCII 13, HEX 0D) are only shown in the syntax description in italic letters (*CR*).
You may use either *CR* (carriage return), *LF* (line feed) or *CR/LF* (carriage return/ line feed)
(See also the ASCII table in the APPENDIX of this manual)

• It is not required to use special characters to create a label format. Data can be keyed in with a simple text editor.

• For a better overview it is allowed to add spaces or tabs within a command line. Numeric parameters accept additional zeros.

• Separators for the parameters are either semicolons or commas.

Usage of this manual

- The commands are sorted in different sections. In each section we further sorted the commands in alphabetical order. We used following structure:

  1. ESC commands
  2. Commands which start with lower case letters
  3. Commands which start with uppercase letters
  4. Special content fields sorted by:
     a: Time functions
     b: Date functions
     c: Mathematical functions
     d: Special Functions
     e:RFID Functions
  5. Description of the cab DataBase connector
  6. Description of the abc - Basic compiler
  7. Appendix A shows a few charts and tables
  8. Appendix B contains some tips and tricks shown on special samples
  9. Last but not least we added a Unicode character list of the internal TrueType fonts.

- Special Notes and infos are shown in italic characters where the "finger" ☞ points to them.

- The examples are mostly reduced to the minimum requirements to print a label, to keep it as simple as possible.

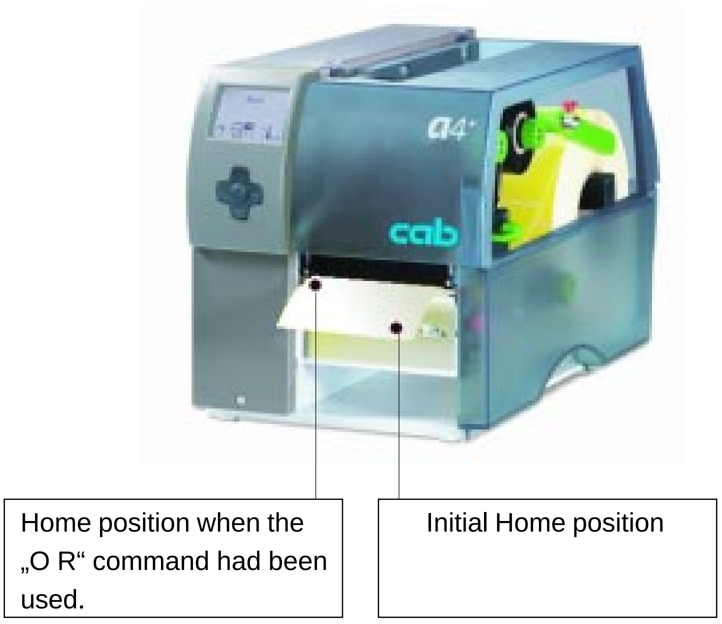
- Not all commands are available or all printer types. This depends on if the described function needs additional equipment such as the RFID functions which are not available in every machine. Please referto the further documentation of your printer.

- In all cases when it was possible we printed an example label, which helps to explain the function of each command. You may copy or type the sample in your editor to see how it works.

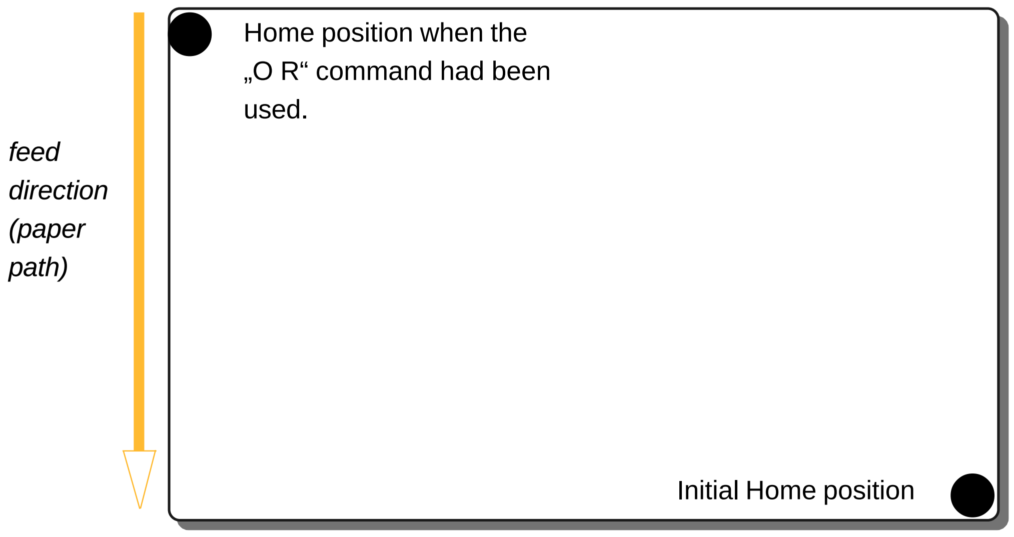
- All examples have been tested and the printouts have been scanned. The original files have been copied into the sample text to make sure to keep the amount of mistakes on a minimum. Nevertheless - please inform us whenever you find anything wrong. We will correct that in the next release of this manual.

Print Positions:

The Home position or „Zero point“ of a label is shown on the picture below.The „Headline“appears first, as it is usual on all laser printers etc. Most users prefer to get the printed label „foot first“ out of the printer. This can easily be done when the „O R“ command is added to the shown examples.
We did not add this command in the samples to keep a better overview. You may add this whenever it is required. „O R“ rotates the orientation of the label by 180 degrees. So all shown examples which do not contain the „O R“ command have been rotated for a better view in this manual.



Home position when the „O R“ command had been used.


feed direction (paper path)
Home position when the „O R“ command had been used.
Initial Home position

Overview

The programming language of the cab Printers is based almost completely on ASCII characters. Together with the selectability of different codepages it is possible to connect to nearly each computer system.

The printers accept additionally all types of line end identifiers (CR, LF, CR/LF), so that the descriptions of labels can be created with the most simple text editors, such as „Notepad“ or „Wordpad“ - saved as plain text files.

Instruction types

cab printers are using basically three types of instructions

- ESC instructions,
- Instructions with lowercase letters and
- Instructions with uppercase letters.

1. ESC instructions

are responsible for status queries, control functions, memory management etc. and are usually executed immediately, i.e. even if a printing job runs. They are not absolutely required to print labels, but they offer additional features and possibilities

Example:

ESC ?	-	Request for free memory.
ESC c	-	Cancel Job
ESC p0	-	Ends printer pause state
ESC s	-	Printer status request

2. Immediate Commands

Instructions with lowercase letters are used for adjustments and settings which must not have something to do with the actual printjob.

These are for example requests of fonts or graphics which have previously downloaded to the printer.

Example:

a	-	Activate the ASCII dump mode
c	-	Immediate cut
f	-	Formfeed
t	-	Performs a test print

3. Label Format Commands

Instructions with uppercase letters are used to describe the label itself.
This has a fix structure, beginning with the start command, the description of the label size and description of each object in the label.
At the end of the label the printer expects the amount of labels.

Example:

J	-	Job start
S	-	Set label size
H	-	Heat, speed, and printing method
O	-	Set print options
T	-	Text field definition
B	-	Barcode field definition
G	-	Graphic field definition
I	-	Image field definition
A	-	Amount of labels

cab printers use additionally to that 3 command types following special commands for special text formatting, calculations, comparisons etc.:

Special content fields
cab database connector commands
abc - a-series basic compiler commands

4. Special Content Fields

are used within Label Format commands.
They consist of instructions in squared brackets, [ ], which offers various data insertion and data manipulation functions.

Example:

[DATE]	- Print date
[/ :op1,op2]	- Divide
[>: op1,op2]	- Greater than

A huge amount of more complex and powerful commands are explained later in this manual in the „Special Content fields“ section.

cab database connector command and „abc“ - commands will not be explained here. Please refer to the special sections in this manual.

On the next pages you will find a short training class which shall help you to become familiar with the cab printer programming language „JSCRIPT“. We recommend that you try this course first, before you start with your own projects.

Simple programming lesson

Target:

Learn how easy it is to teach your printer to do what you want.
Understand the language structure of JScript by testing the following sample.
Get the feeling what might go wrong if the syntax is not correct.
Modify this sample with other items of this manual.

Create your first label:

1. Connect your printer to the PC, select „Country United Kingdom“ on the printer´s control panel. The handling is explained in the operator´s manual (the language changes to „English“ and the measurements to „millimeters“ - as the label is designed in millimeters)
2. Start your preferred plain texteditor (we will use Notepad for this example)
3. Key in following data and don´t forget to press the ENTER key on your keyboard after the „A 1“ in the last line is keyed in.

Example:

```
m m
J
H 100
S l1;0,0,68,70,100
O R
T 10,10,0,5,pt20;sample
B 10,20,0,EAN-13,SC2;401234512345
G 8,4,0;R:30,9,0.3,0.3
A 1
```

Explanation of this example

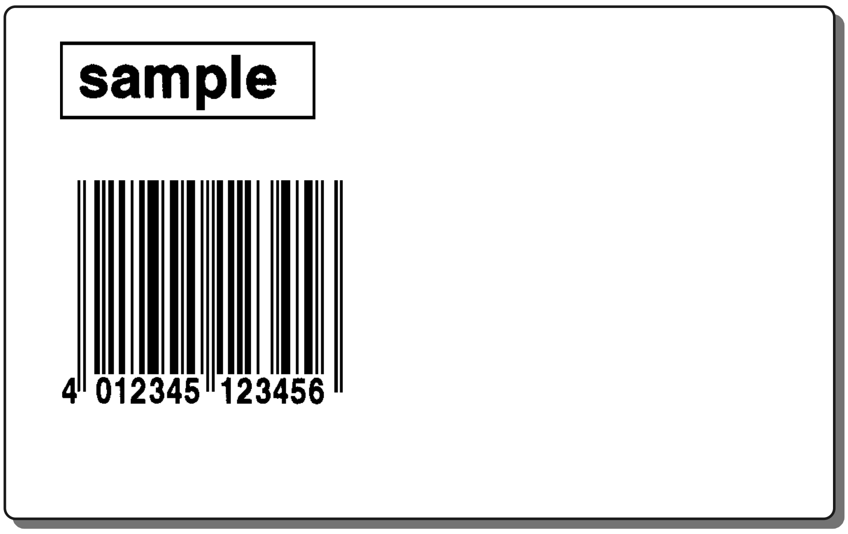
(Details are described in the respective sections of this manual)

```
J                                   Job start
H 100                               Heat (Speed) setting (100mm/sec)
S l1;0,0,68,70,100                  Size of the Label (68 x100mm, gap 2mm)
O R                                 Orientation Rotated by 180°
T 10,10,0,5,pt20;sample             Text line- font:Swiss bold, 20 pt
B 10,20,0,EAN-13,SC2;401234512345   Barcode EAN 13, size SC 2
G 8,3.5,0;R:30,9,0.3,0.3            Graphic, Rectangle  30x9mm, 0.3mm
A 1                                 Amount of labels (in this sample 1)
```

4. Save that file now with the name „sample1.txt“ in your root directory of Harddrive C:

5. Switch to the DOS - mode or to the command prompt (depending on your operating system version)

6. At the command prompt key in: C:\> copy/b sample1.txt LPT1: (LPT1: - if the printer is connected to the parallel port of the PC.) -Requires the optional parallel adapter.
The result should be that the printer prints the label which is shown below.
The better possibility is to transmit data via network connection with an FTP client.
We recommend to download "NOTEPAD++" which is an open source text editor with a built in FTP client. Copy the file directly from the editor to the "execute" folder which is shown by the FTP client. When setting up your FTP connection you will be asked for the server name which is the IP address of your printer. Furthermore you have to key in the Login which is always "root" and the password which is the previously set PIN of your printer´s menu.



... and if it does not work as expected ? - Then following points might be the reason:

1. The printer receives no data:

a: The wrong interface or wrong transmission speed is selected on your printer.
- Check the interface settings in the setup menu of the printer

b: Your interface is blocked by another application.

c: The cable might be defect- check the connecting cable

2. Printer receives data but shows „ribbon out“

a: No ribbon in the printer

b: Ribbon is not fixed on the ribbon unwinder

3. Printer receives data but shows „Protocol error“ in its display

a: Transmitted data is wrong - this might be a missing comma or a accidentially set semicolon instead of a comma or any other wrong data. Spaces after a command may cause a protocol error, too! Check your label data carefully.

Chapter 2: Command Overview

Command Overview

The following pages are showing lists of all available JScript printer commands
Details are explained later in this manual.

ESC Commands

ESCESC	Replaces ESC in binary data
ESC!ESC!	Hard reset
ESC*	Activate all RS 485 printers
ESC.	Start and Stop value for binary data
ESC:	Start description of binary data*
ESC<	Back feed of the material behind the photocell
ESC?	Request for free memory.
ESCa	Request for **a**bc-status
ESCc	**c**ancel printjob
ESCend-of-data	End description of binary data
ESCf	form**f**eed (Equal to pressing „form feed“ on the navigator pad)
ESCi	Send value from the INF-memory
ESCj	Request for the latest printed job
ESCl	Request of synchronisation Info
ESCp0	End printer´s **p**ause mode
ESCp1	Set printer into **p**ause mode
ESCs	Printer **s**tatus query
ESCt	**t**otal cancel of all jobs
ESCz	Extended status request

*) available for Hermes A only !

Immediate Commands

All Immediate commands are processed when a line end identifier is sent (CR, LF or CR/LF)

<abc>	start of „**abc**“ (a-Series basic compiler)
</abc>	end of „**abc**“ (a-Series basic compiler)
; comment	Comment line
a	set printer in **a**SCII dump mode
c	Direct **c**ut
d t;name....	**d**ownload graphic or font data
e t;name....	**e**rase data
f	**f**orm feed
j	**j**ob-ID
l name	Set **l**anguage (country)
m unit	Set **m**easuring unit
p status	**p**ause printer
q b;name	**q**uery **b**itmap font
q d;name	**q**uery **d**Base file on memory card
q e;name	**q**uery format file on memory card
q f	**q**uery **f**ree memory
q i;name	**q**uery **i**mage availability
q l;name	**q**uery **l**abel file on memory card
q m	**q**uery **m**emory type
q p	**q**uery **p**eripheral types
q r	**q**uery **r**ibbon diameter
q s;name	**q**uery **s**caleable font availability
q t	**q**uery **t**ime and date
r	**r**eset to default values
s n	**s**et date/time

Immediate Commands

All Immediate Commands are processed when a line end identifier is sent (CR, LF or CR/LF)

t[x**]**	Run printer self-**t**est
v	Request firmware **v**ersion
x d;uo	Set peripheral (**x**) bits **d**irectly
x e;uo	Set peripheral (**x**) **e**rror value

Label Format Commands

Label format commands are processed when a line end identifier is sent (CR, LF or CR/LF)

A [NO] n	Amount of labels (end job/print)
B [:name;] x, y, r, type,size,text	Barcode field definition
C cnt[,disp1[,disp2]]	Set Cutter parameters
C e	Set Cutter to end-of-job
D x,y	Global Object Offset (Distance to margins)
E DBF;name	Defines a DBF (database) file
E LOG;name	Defines a LOG file
E RFID;...	Define Files (Extension RFID)
E TMP;name	Defines TMP (temporary) serial file
E SQL;[IP of cabDatabaseconnector]:portnr	Sets IP adress for SQL database access
F number;name	Font number
G [:name;] x, y, r; type:options, . . .	Graphic field definition
H speed[,h][,t][,r][,b]	Heat, speed, and printing method
I [:name;]x,y,r[,mx,my];imgname	Image field definition
J [comment]	Job start
M c	Memory card: content request
M d type;name	Memory card: delete file from card
M f;name	Memory card: format card
M l type;[path]name	Memory card: load file from card
M r	Memory card: repeat last label
M s type;name	Memory card: store data on card
M u type;[path]name	uploads data to the host
O [M,][R,][N,][p][T,][U,]	Set print Options
P [disp]	Set Peel-off mode
R name;value	Replace field contents
S [type:]yo,xo,length,dy,wide. . .	Set label Size

Label Format Commands

Label format commands are processed when a line end identifier is sent (CR, LF or CR/LF)

T [:name;] x,y,r, font,size . . ;data	Text field definition
X y[;uo]	Synchronous setting of peripheral (eXternal) signal

Special Content Fields

Time Functions

[H12]	Print Hour in 12-hour form (1-12)
[H24]	Print Hour in 24-hour form (0-23)
[H012]	Print H0ur in 12-hour form (01-12) - always 2 digits
[H024]	Print H0ur in 24-hour form (01-24) - always 2 digits
[ISOTIME]	Prints the Time in ISO standard format
[MIN]	Print MINutes (00-59)
[SEC]	Print SEConds (00-59)
[TIME]	Print actual TIME in the format of the preset country
[XM]	am / pm indicator

Date Functions

[DATE{:+DD{,+MM{,+YY}}}]	Print actual DATE in the format of the preset country
[DAY{:+DD{,+MM{,+YY}}}]	Print numeric DAY of the month (1-31)
[DAY02{:+DD{,+MM{,+YY}}}]	Print numeric 2-digit DAY of the month (01-31)
[DOFY{:+DD{,+MM{,+YY}}}]	Print numeric Day OF Year(1-366)
[ISODATE{:+DD{,+MM{,+YY}}}]	Print ISO date
[ISOORDINAL{:+DD{,+MM{,+YY}}}]	Print ISO ordinal
[ODATE:+DD{,+MM{,+YY}}]	Print DATE with Offset *
[wday{:+DD{,+MM{,+YY}}}]	Print complete weekday name (0 = sunday) *
[WDAY{:+DD{,+MM{,+YY}}}]	Print numeric WeekDAY(0-6)*
[wday2{:+DD{,+MM{,+YY}}}]	Print weekday name, 2 - digits shortened *
[wday3{:+DD{,+MM{,+YY}}}]	Print weekday name, 3 - digits shortened*
[ISOWDAY{:+DD{,+MM{,+YY}}}]	Print numeric WeekDAY (1-7)
[WEEK{:+DD{,+MM{,+YY}}}]	Print numeric WEEK (1-53)
[WEEK02{:+DD{,+MM{,+YY}}}]	Print numeric WEEK with 2 - digits (01-53)
[OWEEK:+WW]	Print WEEK with Offset (1-53)

* (in the format of the preset country)

Special Content Fields

Date Functions (continued)

[mon{:+DD{,+MM{,+YY}}}]	Print 3-character **mon**th name (i.e. jan)*
[month{:+DD{,+MM{,+YY}}}]	Print complete **month** name (i.e. january)*
[MONTH{:+DD{,+MM{,+YY}}}]	Print **2**-digit **MONTH** (1-12)
[MONTH02{:+DD{,+MM{,+YY}}}]	Print **02**-digit **MONTH** (01-12) (leading zeros, always 2 digits)
[YY{:+DD{,+MM{,+YY}}}]	Print **2**-digit **Y**ear (00-99)
[YYYY{:+DD{,+MM{,+YY}}}]	Print **4**-digit **Y**ear (1970-2069)

* (in the format of the preset country)

Special Content Fields

Jalali Date Functions (Arab date)

[JYEAR{:+DD{,+MM{,+YY}}}]	Print Jalali-YEAR, 4 digits
[JDAY{:+DD{,+MM{,+YY}}}]	Print Jalali-DAY
[JDAY02{:+DD{,+MM{,+YY}}}]	Print Jalali-DAY, 02 digits
[JMONTH{:+DD{,+MM{,+YY}}}]	Print Jalali-MONTH
[JMONTH02{:+DD{,+MM{,+YY}}}]	Print Jalali-MONTH, 02 digits
[jmonth{:+DD{,+MM{,+YY}}}]	Print Jalali-month, complete name
[JDOFY{:+DD{,+MM{,+YY}}}]	Print Jalali-Day OF Year
[JWDAY{:+DD{,+MM{,+YY}}}]	Print Jalali-Week DAY (1=saturday)

Suriyakati Date Functions (official date in Thailand)

[SYEAR{:+DD{,+MM{,+YY}}}]	Print Suriyakati-YEAR, 4 digits

Special Content Fields

Mathematical functions
Field Calculations and Comparisons

[+:op1,op2. . ,]	Addition
[-:op1,op2]	Subtraction
[*:op1,op2. . ,]	Multiplication
[/:op1,op2]	Division
[%: op1,op2]	Modulo
[|:op1,op2]	Logical Or (Result 1, if minimum one operator is not equal to 0)
[&:op1,op2]	Logical And (Result 0, if min. one operator is 0)
[<: op1,op2]	Comparison - Less than (1=TRUE, 0=FALSE)
[=: op1,op2]	Comparison - Equal (1=TRUE, 0=FALSE)
[>: op1,op2]	Comparison - Greater than (1=TRUE, 0=FALSE)
[MOD10:x]	Calculates and prints the Modulo 10 Check digit
[MOD36:x]	Calculates and prints the Modulo 36 Check digit
[MOD43:x]	Calculates and prints the Modulo 43 Check digit
[P:name,mn{o}]	Print result in Price format
[R:x]	Rounding method
[==:text1,text2]	String comparision (1=TRUE, 0=FALSE)

Special Content Fields

Special functions (miscellaneous)

[?:x,y,z,{D},{Lx},{Mx},{R},{J}]	Prompt line on the printer´s display
[ABC:x]	Insert ABC value
[BIN:x{,y...}]	Insert Binary data
[C:fill{,base}]	Leading zero replacement
[D:m,n]	Set number of Digits to print
[DBF:keyfield,keyvalue,entryfield]	DataBase Field
[HEX:x]	Hexadecimal conversion
[I{!}{:cond}]	Invisible fields
[JOBID]	print JOB ID
[J:ml]	Justification
[LEN:x]	Returns the Length of a variable
[LOWER:x]	Converts the input data in lower case characters
[LTRIM:x]	Trim data Left
[name]	Access a field with a name
[name,m{,n}]	Insert substring from another field
[RTMP{:x}]	Read from a TMP (serial) file
[RTRIM:x]	Trim data Right
[S:name]	Numeric Script style
[SER:start{incr,{freq}}]	Insert SERial numbering
[SPLIT:field,index]	Splits table values
[U:x]	Insert Unicode character
[UPPER:x]	Converts the input data in upper case characters
[WINF]	Writes value into the „INF“ buffer
[WLOG]	Write to LOG file
[WTMP]	Write to TMP (temporary) serial file

Special Content Fields

RFID Functions

[LTAG...]	Lock RFID TAG area
[RTAG...]	Read RFID TAG
[RTAGBIN...]	Read RFID TAG binary
[TAGID]	Read TAG ID
[WTAG...]	Write RFID TAG

Special Content Fields

Database Connector commands

[SQL:Select field from table where Searchvalue]	SQL - Query function
[SQLLOG:xx]	SQL - Logging function

Special Content Fields

Special Barcode functions (not supported by all barcodes)

[ECE: 123456]	Adds information for extended channel to barcodes
[APPEND:m,n,id1,id2]	
[APPEND:x,id]	Adds information for linked barcodes
[ANSI_DI]	Adds information for ANSI - data identifier
[ANSI_AI]	Adds information for ANSI - application identifier

I M P O R T A N T !!

All measurements of the examples in this manual are in millimeters !

The examples will not work properly when „country“ is set to USA in the printer´s setup menu.
Select „Country = United Kingdom“ in the setup menu of the printer, or add „m m CR“
for metric measurement setting in the first line of your label sample.
We highly recommend to add the mesurement command at the beginning of all of your labels, to avoid trouble with a different setup the printer, unless we did not show this
command in our examples in this manual to keep the examples as small as possible.

Chapter 3: ESC Commands

ESC commands

are responsible for status queries, control functions, memory management etc. and are usually executed immediately, i.e. even if a printing job runs. They are not absolutely required to print labels, but they offer additional features and possibilities.

ESC commands cannot be handled by the most text editors. All other commands can be transmitted to the printer by using simple text editors.

ESC commands are used for activating printers via RS-485, while the printers are „listening“ to the bus, for resetting printers, requesting for free memory or for getting a direct status request.
Details about each command are described on the following pages.

ESCESC Replaces ESC in binary data

ESC ESC is used to replace single ESC (ASCII 27 or Hex 1D) in binary data to avoid unexpected reactions of the printers if graphics or fonts are downloaded.

Graphics or fonts may contain data which can be identical to a ESC printer command. Replacing these ESC characters into double ESCs will tell the printer that this is part of a graphics or part of a font.

Data formats must be checked before they are transmitted to the printer.

File transfer through a FTP connection requires no data conversion if the file is downloaded to the memory card.

Syntax:	`ESCESC`

ESC!ESC! Hard Reset

forces the printer to perform a hard reset. This has the same effect as turning the printer off and on again.

Syntax:	`ESC!ESC!`

The system starts up with the preset default values and shows in the display that data can be received. The display message depends on the preset language selection.

☞ ***The printer is not able to receive data when the Hard Reset is accomplished. Please wait until the printer is restarted again and shows "Ready" in the display to receive data. Otherwise incoming data is discarded.***

ESC* Activate all RS-485 printers

Activates all printers in a RS-485 network at the same time.

Syntax:	`ESC*`

Sends the following data to all attached printers at the same time. This function is only available for printers which are (optional) equipped with the RS485 interface.

Please note that this optional interface hardware is not available for all label printing systems.

ESC. Start and stop value for binary data

Start and Stop value for binary data.

Syntax:	`ESC.`

To transmit binary data -such as graphics or fonts etc. - it is highly recommended to use this method of data transmission. All ESC characters in a binary file have to be replaced by a double ESC (ESCESC) to avoid unexpected reactions by the printer.

A binary constellation -for example- which contains ESC c would be interpreted as „CANCEL JOB“, as soon as it is received by the printer. Therefore all ESC characters should be exchanged.

☞ *Data transmission through ftp requires no conversion.*

ESC: Start description of binary data

Start description of binary data

Syntax:	`ESC:`

cab printers offer a limited possibility to download data without converting them previously. (see also ESC.)
In this case ESC: is required as start sequence, followed by the binary data and finished with ESCend-of-data.

Note: *The binary data cannot contain any ESC character (ASCII 27 or HEX 1B) ! This would be automatically misinterpreted by the system.*
ESC: cannot be used in networks

The better and cleaner way to download binary data is the usage of ESC. We highly recommend to use the sequence.

ESC< Back feed of the material behind the photocell

Back feed material behind the photocell

Syntax:	`ESC<`

The ESC< enables the printer to pull the label backward behind the internal photocell which detects the gap of the material.
This function is only available on printing systems which are equipped with additional mechanics to control the material. (Hermes - applicator series). Otherwise labels would slip out of the feed roller.

ESC? Request for free memory

query for free printer memory input buffer - printer returns a response of 0...9 through its interface.

Syntax:

```
ESC?
```

value	percentage of used memory
0	= 0-9%
1	= 10-19%
2	= 20-29%
3	= 30-39%
4	= 40-49%
5	= 50-59%
6	= 60-69%
7	= 70-79%
8	= 80-89%
9	= 90-99%

☞ *Bidirectional communications must be enabled on the requesting computer.*

ESCa abc-status

Request for abc-status. (Response: XNNNNN)

(abc = a-series basic compiler)

Syntax:

```
ESCa
```

X	=	Condition abc,
I	=	idle,
C	=	compiling,
R	=	running,
E	=	error,
S	=	syntax error during compilation

NNNNN = actual line numbers (empty lines will not be counted!)

A descripton about abc and the available abc commands is shown later in this manual.

ESCc - Cancel Printjob

The current printjob will be **c**ancelled when this command is received by the printer. Equivalent to pressing the „CANCEL“ button on the printer.

Syntax:

```
ESCc
```

Additional labels will processed if they are in the buffer. Please see also „**ESC t**“ command.

☞ ***Wait for minimum one second before transmitting additional data, otherwise the printer will not recognize the following commands.***

ESCend-of-data   End description of binary data

End description of binary data.

Syntax:	`ESCend-of-data`

Finishes the download of binary data. ESC: must be used first, followed by the binary data and closed by ESCend-of-data. Used for font and graphics download.

☞ *Note: **ESCend-of-data** cannot be used in a RS-485 network!*

ESCf formfeed

formfeed - This command is equal to pressing „form feed“ on the navigator pad. Causes the printer to search the start position of the next label.

Syntax:	`ESCf`

Sending a „ESC f“ is a simple method to see immediately if an attached printer receives data and if the connection is setup properly.

ESCi Send value from the INF-memory

ESCi responds the last value of the INF memory.This can be used to get the value of the last printed label. The value uses the actual selected codepage and is finished with a carriage Return.

For more details please view the **[WINF]** command, which writes to the INF memory - described in the section of „Special commands“

Syntax:	`ESCi`

ESCj Request for the latest printed job

ESCj is used together with the command " j " -described later in this manual. Using this command responds the name of the latest printed job. Can be used to get information about, if the print job was finished successfully.
The responded value uses the actual selected codepage and ends with a carriage return.

Syntax:

```
ESCj
```

Example:

```
m m
J
S l1;0,0,68,70,100
T 25,25,0,3,13;Beer
A1

ESCj
```

would generate a generic name if the " j " commmand has not been used and could look like this:

FTP-20091031-14:38:15

Example:

```
m m
J
S l1;0,0,68,70,100
T 25,25,0,3,13;Beer
j another way to control the printer
A1

ESCj
```

would respond:
another way to control the printer

ESCl Request of synchronisation info

ESCl (small letter L) sends information if labels are synchronized and if they are in print position. Delivers also the information about the measured label distance.

Syntax:	`ESCl`

Answer: **XNNNN**

X	=	Paper synchronized (**Y/N**)
NNNN	=	Label distance in millimeters If the distance is unknown, the response will be „0000“

ESCp0 End printer´s pause mode

ends the printer´s **p**ause mode. PAUSE on the printer´s front panel extinguishes and the printjob in the buffer proceeds.

Syntax:	`ESCp0`

Note: This command cancels also existing errors when they are shown in the display of your printer.
- Same function like pressing the PSE button on the navigatior pad.

ESCp1 Set printer into pause mode

causes the printer immediately to set the **p**ause mode. This command has the same function such as pressing the „PAUSE“ button on the printer

Syntax:	`ESCp1`

ESCs Printer status query

ESCs Printer status query,which responds through the interface

Syntax:

```
ESCs
```

Answer: **XYNNNNNNZ**

where:	
X	= Online (Y=Yes, N=No)
Y	= Type of error:
NNNNNN	= amount of labels to print
Z	= Interpreter active (Y=Yes = print job is in process, N=No= printer in Standby mode)

Error types:

- -- No error
a ---- Applicator error- ----- Applicator did not reach the upper position (1)
b ---- Applicator error- ------ Applicator did not reach the lower position (1)
c ---- Applicator error– ----------------------------- Vacuum plate is empty (1)
d ---- Applicator error- ------------------------------------ Label not deposit (1)
e ---- Applicator error- -- Host stop/error (1)
f ----- Applicator error- ------------------------- Reflective sensor blocked (1)
g ---- Applicator error -------------------------------------- Tamp pad 90° error
h ---- Applicator error -- Tamp pad 0° error
i ----- Applicator error ------------------------------- Table not in front position
j ----- Applicator error ------------------------------- Table not in rear position
k ---- Applicator error -- Head liftet
l ----- Applicator error --- Head down
m -- Scanresult negative(4)
n -- global Network error (3)
(this can be: no link, no timeserver, no SQL client,
no SMTP server, no DHCP server or IP adress conflict)
o --- Compressed air-error
r -- RFID -error
s ----------------------------------- System fault (immediately after power on)
u --- USB error

... continued on the next page...

ESCs   Printer status query

Error types: (continued)

A	Applicator error (only older firmware releases)
B	Protocol error/ invalid barcode data
C	Memory card error
D	Printhead open
E	Synchronization error (No label found)
F	Out of Ribbon
G	PPP reload required
H	Heating voltage problem
M	Cutter jammed (2)
N	Label material too thick (cutter) (2)
O	Out of memory
P	Out of paper
R-	Ribbon dectected in Thermal direct mode
S	Ribonsaver malfunction
V	Input buffer overflow
W	Print head overheated
X	External I/O error
Y	Print head error
Z	Printhead damaged

ESCs   Printer status query

☞ *Note: Immediately when a job has started the printer will send a Y and sets this value back to N when the last label of this job is printed.*

(1) This status request can only be processed on printing systems which are equipped with an attached applicator !

(2) Error messages for optional devices such as „cutter jammed“ depend on the availability of the optional device and may vary between different printer types. i.e. No cutter errors on Hermes A4 applicators (These options are not available for these models.)

(3) Network error: Only on printers with an optional or built in network interface. (No print server)

(4) Scanresult negative requires an optional barcode scanner. The availability of the optional barcode scanner depends on the printing system.

(5) Status requests should not be sent in very short cycles ! Minimum time between a status request should be not less than 0.5 seconds.

ESCt total cancel

ESC t = total cancel - terminates the actual printjob and clears the complete input buffer.
Resets also errors in the display. Same effect like pressing „Cancel“ button on the control panel multiple times.

Syntax:	`ESCt`

Please see also **ESCc** which cancels only the actual printjob.

Wait for minimum <u>three seconds</u> before transmitting additional data, otherwise the printer will not recognize the following commands.

ESCz Extended status request

ESC z = exteded status request which is also accessible using the **PEEK „xstatus“** in abc

Syntax:

```
ESCz
```

Answer: **ABCDEFGHIJK** *CR*

A	=	Y =	Printer is paused
B	=	Y =	Printer has a job
C	=	Y =	Printer not ready for print data
D	=	Y =	Paper is moving
E	=	Y =	Ribbon pre-warning (hardware dependend)
F	=	Y =	Paper prewarning (hardware dependend)
G	=	Y =	Label in demand position
H	=	Y =	Label on vacuum plate (hardware dependend)
I	=	Y =	Applicator not ready (hardware dependend)
J	=	Y =	External pause signal active (hardware dependend)
K	=	Y =	External print signal acive (hardware dependend)

All characters are normally N. In addition to ESCs this string is finalized with a carriage return, which allows additional status information in the future.

Immediate commands

Instructions with (almost) **lowercase letters** are used for adjustments and settings which must not have something to do with the actual printjob. They are active as long as the printer is powered up or when these values get overwritten.

<ABC> - Start of the abc Basic Compiler

This command starts the internal Basic compiler. The Basic compiler offers the functions of the basic programming language „YABASIC“. The usage of abc requires good programming knowledge.

abc can be used to create functionalities which are not covered by JScript. The usage of the basic compiler could be to convert incoming data into a format which can be processed by the printer (JScript) or for additional calculations and further influence on the printer.
So an additional programming language has been added to your printer.

Syntax:

```
<ABC>CR
```

Possible usage is to convert text strings - sent by a scale into JScript, or to convert incoming data which was prepared for competitive printers into an understandable format for cab printers.

See also the command: **</ABC>** End of the abc Basic Compiler.

abc is not an emulator !! More information can be found in the „abc a-series basic compiler“ chapter - later in this manual.
abc is not required for the programming of „standard labels“, but it offers nearly unlimited functions.
abc is still a beta release.
More infos about the possibilities of abc can be found later in this manual.

Detailed information about Yabasic can be found at http://www.yabasic.de

</ABC> - End of the abc Basic Compiler

Sets the end mark for the abc compiler (internal BASIC language)

Syntax:	`</ABC>`*`CR`*

See also: **<ABC>** - Start of the abc Basic Compiler.

<ENCRYPTED LABEL...> - Start of an ENCRYPTED label

This command marks the start of an encrypted label file, followed by the board number.
Important: This command requires additional action from cab. It cannot be used without the help through cab.

Syntax:	`<ENCRYPTED LABEL; nnnnnnnnnnnn>`*CR*

nnnnnnnnnnnn = unique main board number

Each "X2" mainboard has a unique serial number which can be used beneath a lot of other features to encrypt label contents to protect your programming work.

Label encryption needs to be done by cab or by registered cab resellers only !

A label which looks like this here:

```
J
S l1;0,0,68,71,104
T 10,10,0,3,5;Test label, encrypted
A 1
```

may look like the 2 lines below after it is encryped.

```
<ENCRYPTED LABEL: 111063523313>
r??@,?h??)(?H=J??2?*?r0?e???1??H??7?‘Q>
```

This file can then be loaded for example from on a memory card. It will only execute on this specific printer with the serial number "111063523313"

Please contact the cab representative if you need more details.

The description of this command has been added for your understanding, just in case if you are confronted with this command in the ASCII dump mode.

<ENCRYPTED JOB> - Start of an ENCRYPTED job

This command starts a previously encrypted print job.

Syntax:

```
<ENCRYPTED JOB>CR
```

Encrypted printjobs need some special support from cab.

The description of this command has been added for your understanding, just in case if you are confronted with this command in the ASCII dump mode.

</ENCRYPTED JOB> - End of an ENCRYPTED job

This command finishes an encrypted print job.

Syntax:

```
</ENCRYPTED JOB>CR
```

Encrypted printjobs need some special support from cab.

The description of this command has been added for your understanding, just in case if you are confronted with this command in the ASCII dump mode.

; - Comment line

The semicolon „ **;** „ is used to identify a comment line. Comments may be placed anywhere in your program code, in a separate line.

Comment lines are ignored by the printer.
Comment lines are very helpful to keep a better overview on the programming data.

Syntax:

```
; comment line CR
```

Example:

```
; My first label - Jobstart
; m m sets the printer to measurement"Millimeters"
m m
; "J" starts my print job
J
; set size of the label
S l1;0,0,68,70,100
; create a text line
T 10,40,0,3,16;Hello
; print one label with the command "A" (amount)
A 1
```

Please note that comment lines need additional time to be transmitted to the printer. Avoid to use comments in time critical situations.

a - ASCII Dump Mode

The a command starts the ASCII dump mode. The ASCII dump mode shows all received data and is a very important instrument to detect wrong data in the program code.
The printer´s LCD panel shows „ASCII dump mode“ in the selected language.
All received data is printed „transparent“ and the printer doesn´t interpret it.

The ASCII Dump Mode is also selectable through the navigator pad.
Note: After ASCII Dump Mode is selected you must confirm this selection with the ENTER button of the navigator pad.

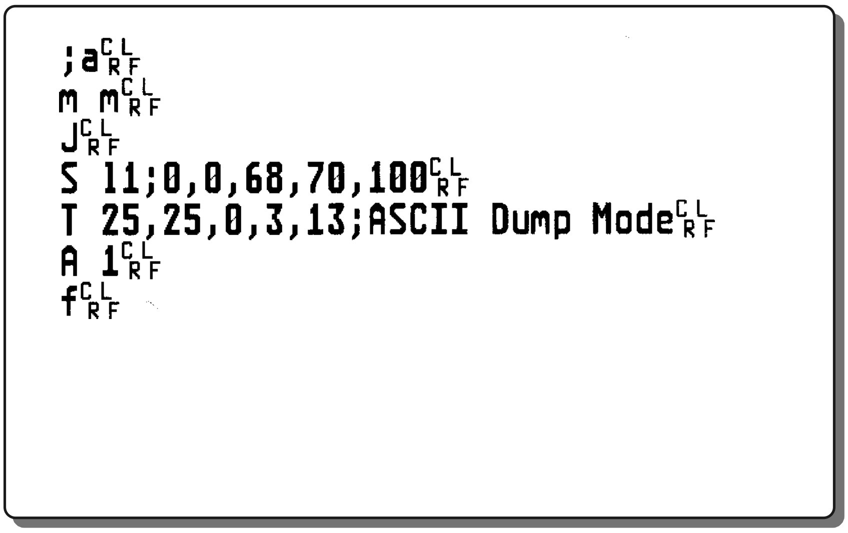
Syntax:

```
a CR
```

The following data creates a label with one line of text. Please view the picture below which shows the same label in ASCII Dump mode.

```
a
m m
J
S l1;0,0,68,70,100
T 25,25,0,3,13;ASCII Dump Mode
A1
f
```

If „protocol errors“ are shown on the label means, that there is a mistake in the program code!



```
;a␍␊
m m␍␊
J␍␊
S l1;0,0,68,70,100␍␊
T 25,25,0,3,13;ASCII Dump Mode␍␊
A 1␍␊
f␍␊
```

a - ASCII Dump Mode

The following example shows that something is wrong in the text line. We used a font (font number 20 which is maked in bold charcters in the sample below and which is not available in the printer. This is recognized by the printer which points us to the line which needs to be corrected.

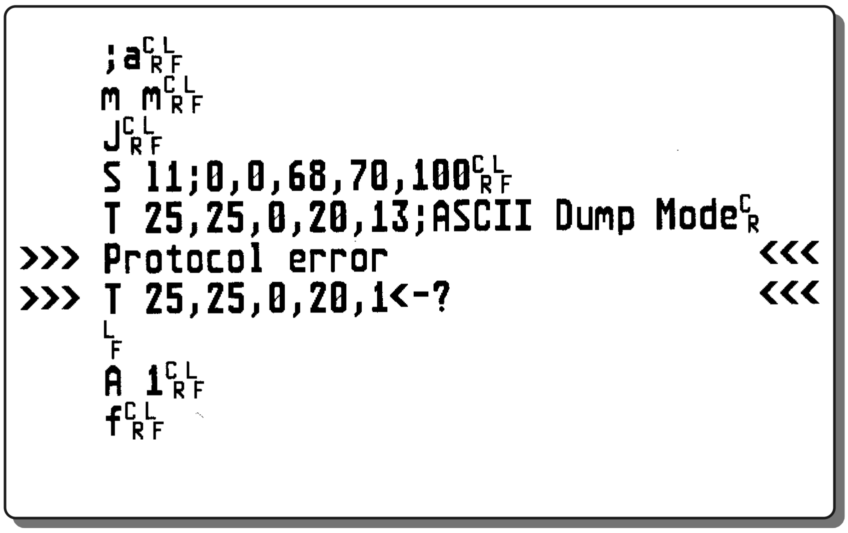
There is no list of "possible Protocol errors" as nearly everything which can not be interpreted by the printer can be shown in the printer´s display or in the printout of the ASCII dump mode.
Pressing the blinking "pause button" skips the most Protocol errors and finishes the label (unless there is some content which is totally wrong or if no label size is defined)

Pressing the printer´s "cancel button" cancels the print job.

Example:

```
;a
m m
J
S l1;0,0,68,70,100
T 25,25,0,20,13;ASCII Dump Mode
A 1
f
```

If „protocol errors“ are shown on the label means, that there is a mistake in the program code!



```
    ;aCRLF
    m mCRLF
    JCRLF
    S l1;0,0,68,70,100CRLF
    T 25,25,0,20,13;ASCII Dump ModeCR
>>> Protocol error                         <<<
>>> T 25,25,0,20,1<-?                      <<<
    LF
    A 1CRLF
    fCRLF
```

c - Direct cut

The c ommand forces the printer to cut immediately when it is received.
If required, the printer will do a formfeed before the cut is processed.

This command is not available for Hermes and for the PX module.

Syntax:	`c` *CR*

The printer shows „Protocol error   c<--“ on the display if no cutter is attached.

d - download data (pictures, fonts etc...)

The d command is used to download data files to the printer. It is used to download graphics, fonts, databases and serial files (temporary files).Two methods are available to download such data to the printer:

1st Method:
The procedure which we highly recommend, unless this requires that the data has to be prepared for downloading.

Syntax:
```
d type;name[SAVE] [B:± value]CR ESC.binary data ESC.
```

2nd Method:
will transmit the data as it is, but it may occasionally misinterpret embedded ESC characters in the data as a printer command. (i.e. ESC t would be misinterpreted as memory reset).

Syntax:
```
d type;name[SAVE] [B:± value]CR ESC:binary data ESCend-of-data
```

d	= download data
type	= the type of data that will follow, using standard file name extensions

Graphic formats:

BMP	-	Windows bitmap format	Monochrome, 256 Colors, 24 Bit Truecolor, plane only, uncompressed
GIF	-	Graphic Interchange Format	(GIF 87a and GIF 89a)
IMG	-	GEM Image format	Monochrome
MAC	-	MacPaint format	
PCX	-	Paintbrush format	Monochrome, 16 and 256colors
PNG	-	Portable Network Graphics	
TIF	-	TIFF Format© Aldus Corp.	Monochrome, Greyscale and color. (4Bit and 8Bit per pixel, RGB 8 Bit per pixel)-Compression: Only packbits and uncompressed.
ASC	**-**	Graphic in ASCII format	

Vector font format:

TTF	-	TrueType font format

Database format:

DBF	-	dBASE IV Database formats (Field type must be text)

others:

TMP	-	Serial numbering (temporary) file in ASCII format

d - download data (pictures, fonts etc...)

We recommend to use monochrome graphics only! The resolution should not be higher than the printer´s printhead resolution.

name	=	Filename to be downloaded with a maximum length of 8-digits. This filename will be recalled on later programming.
[SAVE]	=	This optional parameter is used for downloading to the printer´s memory card. (The memory card commands (M ... explain more possibilities, - please see there for more details) The [SAVE] option copies the file from the printers memory to the memory card.
B: ± value	=	Sets the brightness of dithering on graphics. Valid values are ± 20.

Example:

```
B:+5 makes the picture 5 steps darker.
```

ESC.<*graphics data*> **ESC**

= 1st Method for downloading data. Data format is binary, where the ESC characters (ASCII 27 or HEX 1B) have to be replaced first through a double ESC (ESCESC) to avoid unexpected reactions of the printer.
ESC commands, (requests etc.) can be used during the download of this data.
cab offers the tool: DOWNLOAD.EXE (downloadable at http://www.cabgmbh.com) to convert existing files.

Syntax:

```
d BMP;CABLOGO CR ESC. binary data ESC.
```

Downloads the graphics: cablogo.BMP to the printer

ESC: <*graphicsdata*> **ESCend-of-data**

= 2nd Method for downloading data. Data format is binary, starting with ESC: and followed by ESCend-of-data (ASCII 27 or HEX 1B) followed by ASCII text string < end-of-data >.
With this method it is allowed that the data stream contains ESC sequences in the data stream until the ESCendofdata is received.

d - download data (pictures, fonts etc...)

Example:

```
d TTF;ARIAL<CR> ESC: data ESCend-of-data
```

We highly recommend to use the 1st Method for data download !!

Example:

```
d DBF;CDPlayer [SAVE] CR ESC.binarydata ESC.
```

Downloads the database file CDPlayer.DBF to the printer.

Database files have to be downloaded with the **[SAVE]** option, as they are only used together with the memory card. This function is useful for „small“ databases. Big databases need a long search time for single records. In this case we recommend the usage of the optional cab Database connector.
See more at the DataBaseConnector command area.
(cab Databaseconnector is not available for the M-series printers)

cab sells a helpful tool (the cab card manager) which can be used to download files through the serial interface to the memory card. This simplifies data conversion and download.

An alternative tool for downloading and editing directly on the memory card is the cab network manager which connects through the ethernet interface to the printer and which offers more direct access to the printer.

Data can also be saved on a card drive for Compact Flash cards. Please note, that the CF-cards have to be formatted (erased) in the printers memory card slot. This automatically generates also the required folders on the card.

d - download data (pictures, fonts etc...)

DOWNLOAD ASCII graphics

ASCII-Graphic format

The stucture is similar to the IMG format, but uses only ASCII characters, to enable a easy usage for host devices or ERP systems.

Following rules are used:

- all data are hex bytes, i.e. 0-9 and a-f or A-F
- The printer waits as long for data until the defined picture size is received.
- Spaces and carriage returns can be added on different locations. It is required that a carriage return is sent at the end of the picture data.
- The image data can be compressed with a simple algorithm which is black/white optimized.
- The image data are transmitted from top to bottom, each time from left to right. A value byte 80 stands left of 01.
- The first line describes the width and the height of a picture. Width and height are 16 bit values each in the Big-Endian format.
- Also if the width is not devidable by 8it is required that the missing pixel must be transmitted

Each line will be transmitted with following values:

- 1. Optional repetition factor, caused by 00 00 FF xx, whereby xx describes the amount of copies of the actual line.
- Picture data - whereby different descriptions are optional possible.
  a: Zerobytes are displayed through the amount of bytes.Valid input: 00 to FF.
  b: Blackbytes (FF) can also be described through the amount of bytes, beginning from 81 (81 means 1 time FF, - valid values are 81 to FF .
  c: A directly encoded number of bytes starts with 80 - followed by the amount of data, i.e. 80 03 123456. The amout of transmitted bytes can be between 01 and 7F.
  d: A repeated pattern of arbitrary bytes can be initiated with a sequence 00 nn xx, which means that xx bytes will be inserted nn times.
  Example: 00 04 AA generates AAAAAAAA.

d - download data (pictures, fonts etc...)

The following example shows how a graphic file may look as ASCII data. We download this file with the name "cab.asc" in the graphics folder of the optional memory card of the printer (or in the internal Flash File System - IFFS) to recall it with the label data shown on the next page.
The example below is not length optimized. The explanation in italic letters does not belong to the graphics data.

Example:

```
0053 0020 CR                               - describes  a picture
                                             with 83 pixels width
                                             and 32 pixels height.
0000FF09                                   - repeats the actual line
                                             9 times
06                                         - 6 zero bytes
800207F0                                   - one bitstring, consists
                                             of 2 bytes with 07 and F0
03 CR                                      - three zero bytes
800B007FFF003FFFE7F7FF0000 CR              - picture data directly
                                             sent as bit string
800101 82 800103 82 8005E7F7FFF000 CR      - picture data, mixed,
                                             compressed and direct.
800107 82 800107 82 8005E7F7FFF800 CR
80010F 82 80011F 82 8005E7F7FFFE00 CR
80011F 82 80013F 82 8002E7F7 82 01 CR
80013F 82 80013F 82 8002E7F7 82 01 CR
80013F 82 80017F 82 8002E7F782800180 CR
800B7F80007F800FE7F0007F80 CR
80017F 02 8008FE000FE7F0001FC0 CR
80017E 02 8008FE000FE7F0001FC0 CR
0000FF04                                - repeats the line 4times
80017E 02 8008FC000FE7F0000FC0 CR
80017F 02 8008FE000FE7F0001FC0 CR
800B7F8000FF000FE7F0003FC0 CR
800B3FF000FFC00FE7F0007F80 CR
80013F 82 80047FFFEFE7 83 800180 CR
80011F 82 80043FFFEFE7 83 01 CR
80010F 82 80041FFFEFE7 82 8002FE00 CR
800107 82 80040FFFEFE7 82 8002FC00 CR
800101 82 800407FFEFE7 82 8002F800 CR
8007003FFF00FFEFE7 82 8002E000 CR
```

d - download data (pictures, fonts etc...)

The sample below recalls the graphic file from memory card and prints the image on the defined position. In this case we used the data shown on the previous page of this manual.

Example:

```
M l IMG;cab
m m
J
S l1;0,0,68,73,100
I:TEST;3,30,0,2,2;cab
A1
```

e - erase data

The e command is used to erase data from the printer´s memory (RAM), such as fonts and graphics. Data on the memory card will not be affected by this sequence. Separate commands are available for erasing files from the memory card. (see also the „M“ commandlater in this manual)

Syntax:

```
e type;name CR
```

e - erase data command	
type	= The file types being removed, with following valid file extensions: Images: **BMP, GIF, IMG, MAC, PCX, PNG, TIF** Fonts: **FNT, TTF**. (**FNT** can be used for all font types and **IMG** can be used for all picture types)
name	= The name attached to the font or graphic when it was sent to the printer. A wildcard (*) may be used to delete all files of the same type. "name" is not case sensitive.

Example:

```
e FNT;*
```

Erases all true type fonts which are currently in the printer’s memory.

Example:

```
e IMG;logo
```

Erases the picture with the name "logo" in the printer´s memory

The printer keeps the received graphic fles in its internal memory until it will be switched off or until these files will be erased or overwritten.

f - formfeed

This command feeds the media forward until the top-of-form of the next label reaches the printhead. It does the same as pressing the feed button on the printer´s control panel.
This process is controlled by the label photocell if die cut label material is used. The printer feeds the material in continuous form mode in the length which had been selected for the last printed label.
The label photocell is disabled for gap detection and controls only if paper is out.
In continuous form mode the printer counts the steps of the stepper motor to reach the expected print length.

Syntax:

```
f CR
```

Example:

```
f CR
f CR
```

feeds 2 empty labels.

j - job-ID

Sets the job ID for the actual print job / part of the print job. This command is used together with "ESCj". The printer generates a generic name if the "j" command is used without additional information. This string has following structure: source interface / label name-date-time.
The "j" command needs to be positioned after the job start command ("J"), otherwise the job ID would be overwritten.

Syntax:

```
j Job-ID CR
```

```
m m
J
S l1;0,0,68,70,100
T 25,25,0,3,13;Beer
A1

ESCj
```

would generate a generic name if the " j " commmand has not been used and could look like this:

FTP-20091031-14:38:15

("ESC j" is used to show the result. The infomation is sent to the interface)

```
m m
J
S l1;0,0,68,70,100
T 25,25,0,3,13;Beer
j another way to control the printer
A1

ESCj
```

would respond:
another way to control the printer

l - Change Language (country)

Date format, currency, measurement etc. are changed with this command to the country specific values.
Time and date will be printed as it is usual in the specified country. (See also „Special Content Fields)
The display on the printers LCD will not be changed. (This can be done using the printer´s setup through the control panel)

Syntax:

```
l name CR
```

l - Change language/country command.

name = DOS short keyboard code for the country. Valid values are:

BE	Belgium / french	**RU**	Russia
BF	Belgium / flamic	**SA**	South Africa
BG	Bulgaria	**SE**	Sweden
CZ	Czech Republic	**SF**	Switzerland / french
DK	Denmark	**SG**	Switzerland / german
FR	France	**SP**	Spain
GK	Greece	**SR**	Serbia
GR	Germany	**SU**	Suomi (Finland)
HR	Kroatia	**TH**	Thailand
HU	Hungary	**TR**	Turkey
IR	Iran	**UK**	United Kingdom (Great Britain)
IT	Italy	**US**	USA*
LA	Latinoamerica	**ZH**	China
LT	Lituvia		
MK	Macedonia		
MX	Mexico		
NL	Netherlands		
NO	Norway		
PL	Poland		
PT	Portugal		

**selects measurements in inches !*

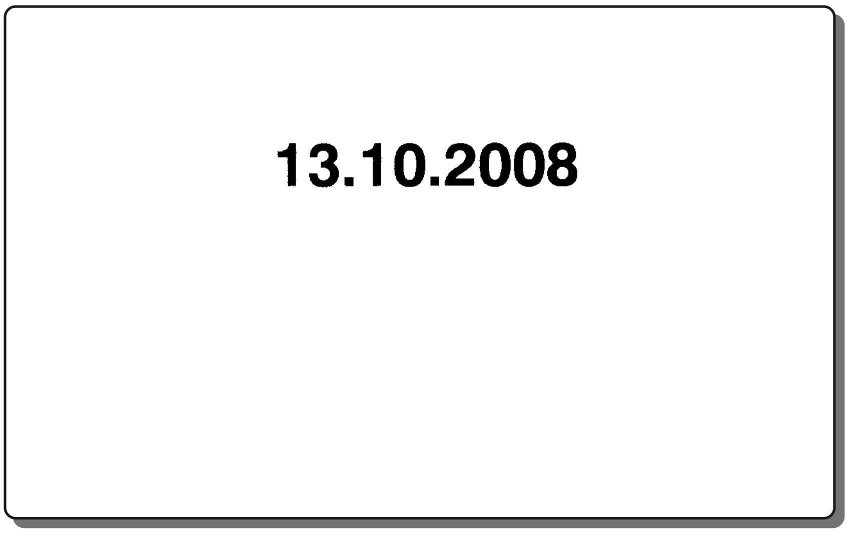
The "r" command resets the language to the default value in the printer´s setup

l - Change Language (country)

The following example prints the date, while the " l "command changes the language into "german", which causes that the date prints in german style: Day**.**Month**.**Year (separated with dots)

Example:

```
l GR
J
S l1;0,0,68,71,100
T 25,25,0,5,8;[DATE]
A1
```



m - set measuring unit

This command sets the measuring unit for the following label data.
Once it is sent, all following settings in a label are measured in the selected unit.

The printer´s default value depends on the selected display language. For all selectable countries the measurement is millimeters, with the exception when country USA was set through the control panel. We recommend to use this command always, especially for international companies where different programmers create labels as the measuring unit is only changed for the individual label being printed.

The measuring unit cannot change within one label. All internal calculations are processed in millimeters, as these values are better to overview and they follow a worldwide standard.

Syntax:

```
m t CR
```

m - Set measuring unit command.	
t	= The measuring system desired, „**m**“ for metric (millimeters) or „**i**“ for historical (inches, tenths and hundredths of an inch).

m - set measuring unit

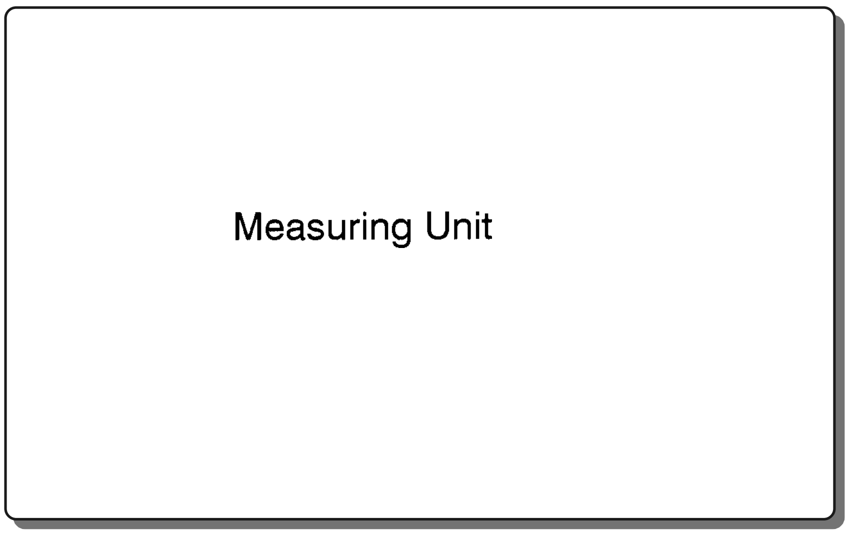
The next example shows the same label programmed with different measurement settings. The result is the same. The first example is programmed in inches, the second example is programmed with metric measurement settings.

Example:

```
m i
J
T 0.79,1.18,0,3,0.2;Measuring Unit
A1
```

Example:

```
m m
J
T 20,30,0,3,5;Measuring Unit
A1
```



p - pause Printer

The printer is set in the pause mode or removes it from pause - depending on the parameter.

Syntax:
```
p n CR
```

p - pause printer			
n	=	**0** =	Pause off
		1 =	Pause on

Example:
```
p 1
```

Sets the printer into pause mode. If a print job runs, it will stop after the label is printed.
Pause lights on the front panel.

q - query Printer

The query printer command is used to get multiple information back from the printer and is e.g.. used to find out if a font or a picture exists, so that has not to be downloaded a second time. The q command responds through the printer´s interface. All bidirectional interfaces can be used.

Syntax:	`q X;name` ***CR***

q - query different infos from the printer,where **X** =

b;name ***CR***	=	Query for a **bitmap font**. Answer: **Y/N.** Requests the printer if a specified bitmap font is available.
d;name ***CR***	=	Query for a **database**. Answer: **Y/N** Requests the printer if the dBase database (DBF) file called „name“ is available on the memory card.
e;name ***CR***	**=**	Query for **media**. Answer: **Y/N** Requests the printer if the media (FMT) file called „name“ is available.
f ***CR***	=	Query for **free memory**. Answer: **xxxxxxxbytes free** Reports the free (available) memory, which may be used for downloaded data. Requests the printer if the image (IMG) file called „name“ is available either in memory or on memorycard.
i;name ***CR***	=	Query for **image** Answer: **Y/N** if available in memory, or **C** if the pictogram is available on memory card.
l;name ***CR***	**=**	Query for **label** Answer: **label name** Requests the printer if a specified label is available.
m ***CR***	=	Query for the **memory card type** Answer: **Format "type, xxx kByte.*CR*"**, **-** The response will be "**No card *CR***“ if no memory card is attached to the printer

continued on the next page

q - query Printer

The query printer command is used to get multiple information back from the printer and is e.g.. used

q - query , **X** =		
p *CR*	=	Query for **peripheral equipment** Reports the type of peripheral devices that are connected. Possible responses are: **NONE *CR*,** **CUTTER *CR*,** **REWINDER *CR*,** **DEMAND SENSOR *CR*,** **BLOW ON *CR*,** **TRIGGER *CR*** (Applicator) Possible answers depend on the printer type and it´s available options !! Used to verify if a label can be processed on the selected printer. Very helpful if multiple printers with different peripheral equipments are connected.
r *CR*	=	Query for **ribbon diameter**. Answer: **diameter of the ribbon roll in mm.** If the ribbon roll has not been measured, the answer will be -1 Can be used to get an early warning when the ribbon is close to be finished.
s;name **CR**	**=**	Query for **scaleable fonts** Answer: **Y/N or C** if the font had been found on the memory card. This command is used to check if a specified font is available to find out if it has to be downloaded (again).
t *CR*	=	Query for **time and date** Answer: **yymmddhhmmss** CR **yy** = Year - 2 digits **mm** = Month. - 2 digits **dd** = day - 2 digits **hh** = hour - 2 digits **mm** = minutes - 2 digits **ss** = seconds - 2 digits

.

r - reset to default values

This command resets JScript to the printer´s default values.

- resets the language
- resets slashed zero setting
- resets the selected measurement system
- erases the fontcache
-sets the date setting back to the selected country in the setup

Syntax:

```
r  CR
```

s - set Date/Time

Used to set date and time to be recalled on a label. The printer has an internal real time clock which keeps date and time. If it is required this command can be used to synchronize the attached device and the printer.

Syntax:

```
s n[ss] CR
```

s = Set date / time command.

n	=	ASCII - string in following format to adjust date and time in the printer of following format: **YYMMDDhhmmss**

YY	=	Year	- 2 digits (values between 70 and 99 are interpreted as 1970-1999. Anything else is treated as year2000)
MM	=	Month.	- 2 digits
DD	=	day	- 2 digits
hh	=	hour	- 2 digits
mm	=	minutes	- 2 digits
[ss]	=	seconds	- 2 digits (setting of ss is optional)

Example:

```
s 081105091500
```

Sets printer date and time to:
November 24, 2008 9:15 a.m.

t - Run Printer Self-test

cab printers have multiple built in self -tests. A self test can be processed through the printer´s control panel (see operator´s manual) or by software.
The printout of the status information may look different on different printer types. Information about optional equipment, such as interfaces, cutter etc. will only be shown if they are attached.

Syntax:	`t{n} CR`

t - run printer selftest	
n	= 0 - prints status information
	= 1 - prints the font list
	= 2 - prints the device list
	= 3 - prints the label profile
	= 4 - reserved
	= 5 - prints the test grid
	= 6 - wireless network status (requires installed WLAN card) *
	= 7 - RFID measurement (requires installed cab RFID reader)**

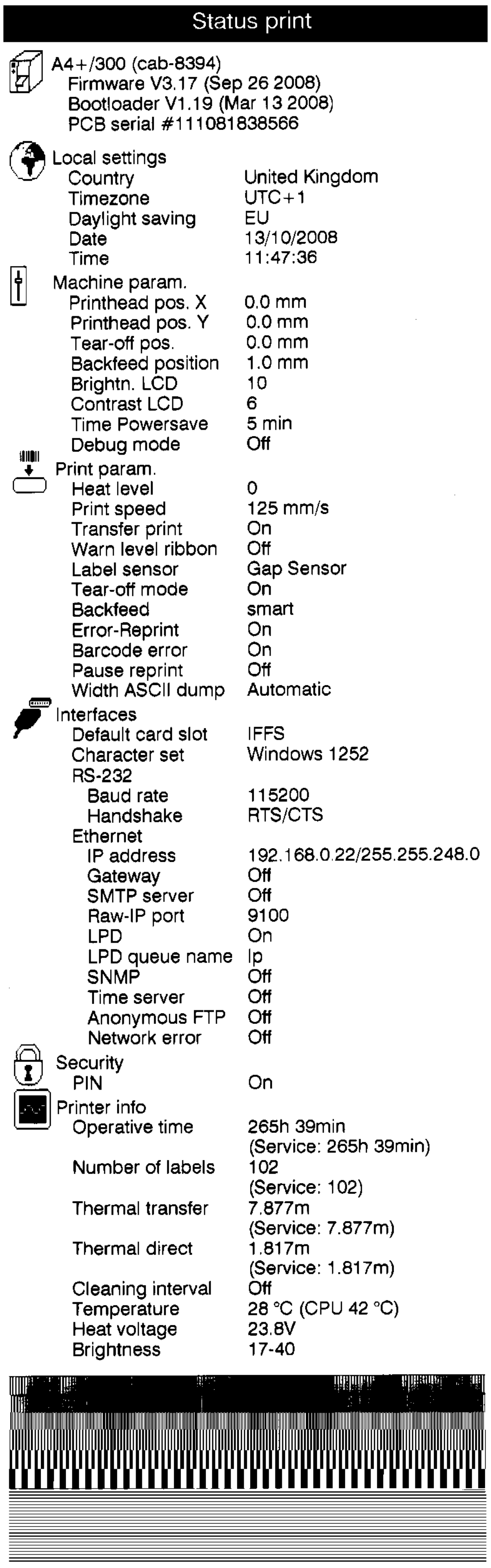
The status test is displayed in the selected language of the printer.
* Only available for printers which support the WLAN card (X2 board)
** Only available on Mach 4.

t - Run Printer Self-test

Example: `t0`

prints the **status information**
(here A4+/300)

The status printout is different when printed by different printer types. A detailed description of the listed values can be found in the operator´s manual.
Transmitting „t“ without any additional number causes the printer also to do a status printout.



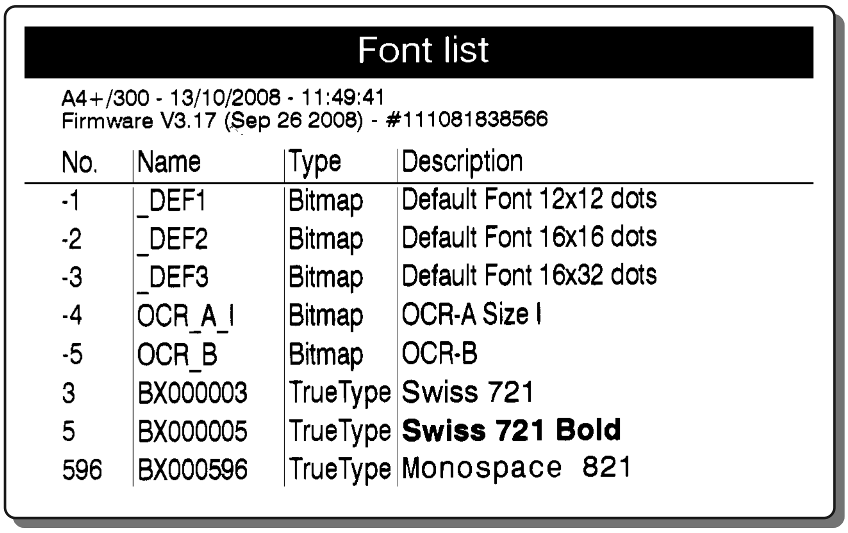
```
Status print

A4+/300 (cab-8394)
  Firmware V3.17 (Sep 26 2008)
  Bootloader V1.19 (Mar 13 2008)
  PCB serial #111081838566
Local settings
  Country                 United Kingdom
  Timezone                UTC+1
  Daylight saving         EU
  Date                    13/10/2008
  Time                    11:47:36
Machine param.
  Printhead pos. X        0.0 mm
  Printhead pos. Y        0.0 mm
  Tear-off pos.           0.0 mm
  Backfeed position       1.0 mm
  Brightn. LCD            10
  Contrast LCD            6
  Time Powersave          5 min
  Debug mode              Off
Print param.
  Heat level              0
  Print speed             125 mm/s
  Transfer print          On
  Warn level ribbon       Off
  Label sensor            Gap Sensor
  Tear-off mode           On
  Backfeed                smart
  Error-Reprint           On
  Barcode error           On
  Pause reprint           Off
  Width ASCII dump        Automatic
Interfaces
  Default card slot       IFFS
  Character set           Windows 1252
  RS-232
    Baud rate             115200
    Handshake             RTS/CTS
  Ethernet
    IP address            192.168.0.22/255.255.248.0
    Gateway               Off
    SMTP server           Off
    Raw-IP port           9100
    LPD                   On
    LPD queue name        lp
    SNMP                  Off
    Time server           Off
    Anonymous FTP         Off
    Network error         Off
Security
  PIN                     On
Printer info
  Operative time          265h 39min
                          (Service: 265h 39min)
  Number of labels        102
                          (Service: 102)
  Thermal transfer        7.877m
                          (Service: 7.877m)
  Thermal direct          1.817m
                          (Service: 1.817m)
  Cleaning interval       Off
  Temperature             28 °C (CPU 42 °C)
  Heat voltage            23.8V
  Brightness              17-40
```

t - Run Printer Self-test

The label below shows a list of the printer´s internal fonts. If additionally downloaded, True type fonts will also be shown on the printout in their actual shape. (see the font list below)

Example: `t1`

prints a label with a list of all existing fonts. (**fontlist**)
A detailed description about the internal fonts is shown later in the manual where the usage of textfields is describedand in Appendix C.



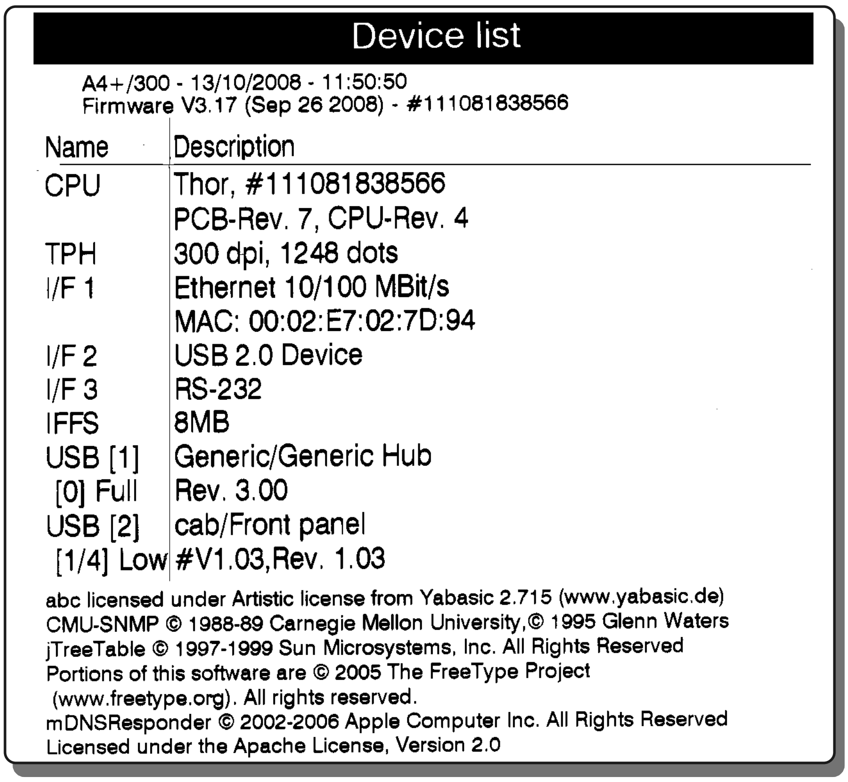
Font list

A4+/300 - 13/10/2008 - 11:49:41
Firmware V3.17 (Sep 26 2008) - #111081838566

No.	Name	Type	Description
-1	_DEF1	Bitmap	Default Font 12x12 dots
-2	_DEF2	Bitmap	Default Font 16x16 dots
-3	_DEF3	Bitmap	Default Font 16x32 dots
-4	OCR_A_I	Bitmap	OCR-A Size I
-5	OCR_B	Bitmap	OCR-B
3	BX000003	TrueType	Swiss 721
5	BX000005	TrueType	**Swiss 721 Bold**
596	BX000596	TrueType	Monospace 821

Example: `t 2`

prints the list with all attached devices.



Device list

A4+/300 - 13/10/2008 - 11:50:50
Firmware V3.17 (Sep 26 2008) - #111081838566

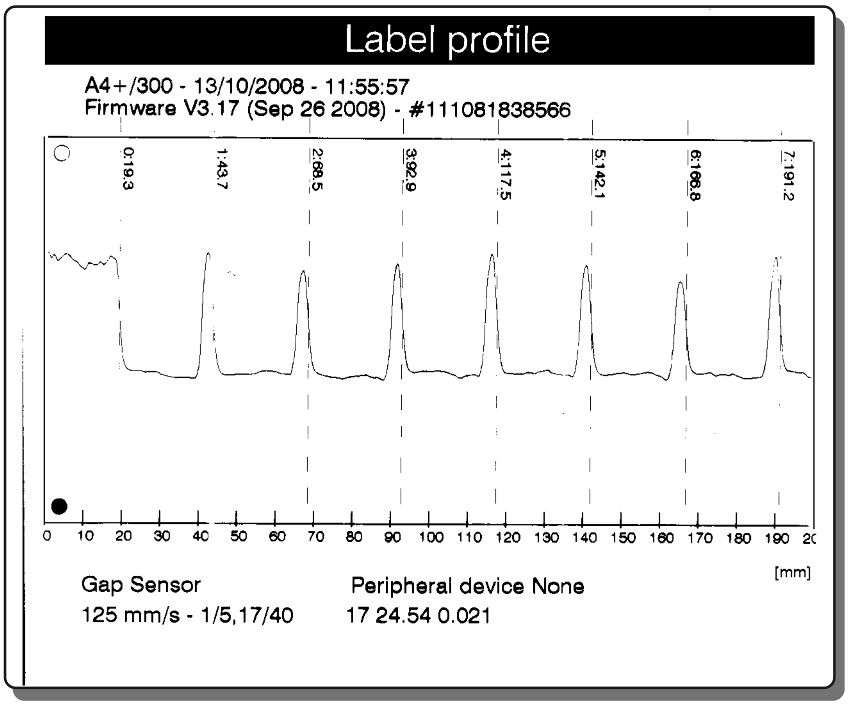
Name	Description
CPU	Thor, #111081838566
	PCB-Rev. 7, CPU-Rev. 4
TPH	300 dpi, 1248 dots
I/F 1	Ethernet 10/100 MBit/s
	MAC: 00:02:E7:02:7D:94
I/F 2	USB 2.0 Device
I/F 3	RS-232
IFFS	8MB
USB [1]	Generic/Generic Hub
[0] Full	Rev. 3.00
USB [2]	cab/Front panel
[1/4] Low	#V1.03,Rev. 1.03

abc licensed under Artistic license from Yabasic 2.715 (www.yabasic.de)
CMU-SNMP © 1988-89 Carnegie Mellon University,© 1995 Glenn Waters
jTreeTable © 1997-1999 Sun Microsystems, Inc. All Rights Reserved
Portions of this software are © 2005 The FreeType Project
(www.freetype.org). All rights reserved.
mDNSResponder © 2002-2006 Apple Computer Inc. All Rights Reserved
Licensed under the Apache License, Version 2.0

t - Run Printer Self-test

Example: `t3`

produces following result after the printer feeded a few empty labels for the measurement process. **(Label profile)**



Label profile

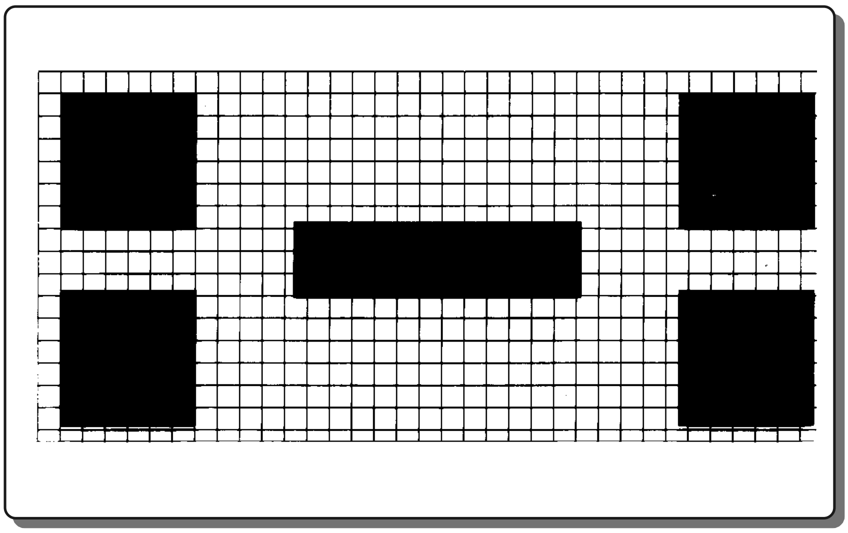
A4+/300 - 13/10/2008 - 11:55:57
Firmware V3.17 (Sep 26 2008) - #111081838566

Gap Sensor
125 mm/s - 1/5,17/40

Peripheral device None
17 24.54 0.021

Example: `t5`

prints a text grid which can be used for the printhead adjustment or to control the print quality of the printer. **(Testgrid)**

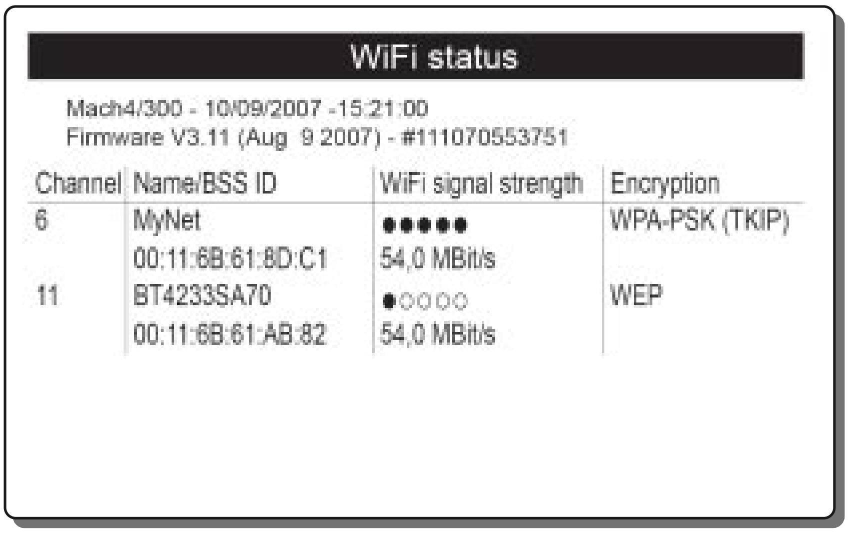


t - Run Printer Self-test

Example:

```
t6
```

shows information about the optional wireless network card. **(WiFi status)**
(A wireless network card needs to be installed to run this test)



WiFi status

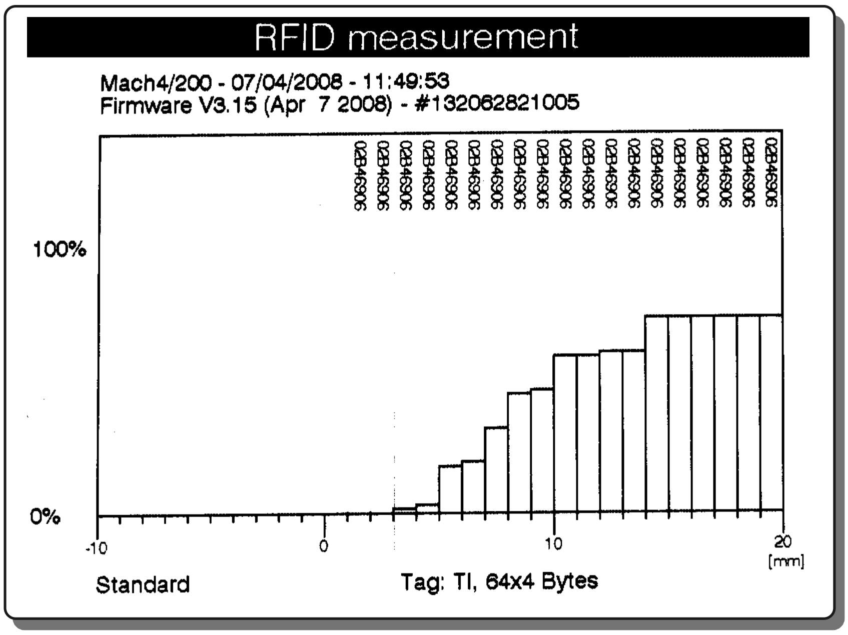
Mach4/300 - 10/09/2007 -15:21:00
Firmware V3.11 (Aug 9 2007) - #111070553751

Channel	Name/BSS ID	WiFi signal strength	Encryption
6	MyNet	●●●●●	WPA-PSK (TKIP)
	00:11:6B:61:8D:C1	54,0 MBit/s	
11	BT4233SA70	●○○○○	WEP
	00:11:6B:61:AB:82	54,0 MBit/s	

Example:

```
t7
```

prints the RFID measurement info. (Mach 4 only) **(RFID measurement)**
(The printer must be equipped with the optional RFID unit)



RFID measurement

Mach4/200 - 07/04/2008 - 11:49:53
Firmware V3.15 (Apr 7 2008) - #132062821005

v - Firmware version

The v command requests the firmware version, release date and printer model. The printer responds through the interface.

Syntax:

```
v  CR
```

Example:

```
v  CR
```

An A4+/300 printer will respond on this request with following string:

3.17 Sep 26 2008 (A4+/300)

3.17	**Sep 26 2008**	**(A4+/300)**
Firmware version	Release date	Printer model

x - Synchronous Peripheral Signal Settings

The signal bits of the peripheral connector for external connections can be set with this command.
Usage: Together with an optional adapter with electrical protected interface.
The availability of these adapters depends on the used printing system.

IMPORTANT: Never connect any non cab item directly to the printers auxiliary interface !
In all cases you will need an optional adapter with the required interface !!!
Connections directly on the auxiliary interface may damage the printer electronics !
The auxiliary interface does not deliver the following signals directly.

This command controls the status of the output pins. The x command was added to take control over peripheral device, which is usually other than the offered cab equipment. The four signal bits can be set as follows:

Control bit 0, set on when a label starts printing
Control bit 1, toggled when a new print job starts
Control bit 2, set on for error
Control bit 3, set on when label is in the peel-off position

Each of these bits can be set or reset for individual needs. The bit signals can be used to control external - non cab - devices.

To reset all of these bits, use ESC!ESC! (see ESC commands)

Syntax:	**x** m;m *CR*

x - Snchronous Peripheral Signal Setting Command		
m	=	Mask (hex nibble).

The usage of this command depends on the printer type. The description of the pin assignment can be found in the available documentation for the optional adapters

z - print slashed / unslashed zero

The default setting for the zero character is unslashed. With this command the printer can be forced to change the style of the zero character. It can be printed as 0 (unslashed) or Ø (slashed).

This command can only be used with internal bitmap fonts. It is not available for internal vectorfonts (Swiss, Swiss bold and Monotype) or for truetype fonts: The selected method is valid for the complete label.

Syntax:

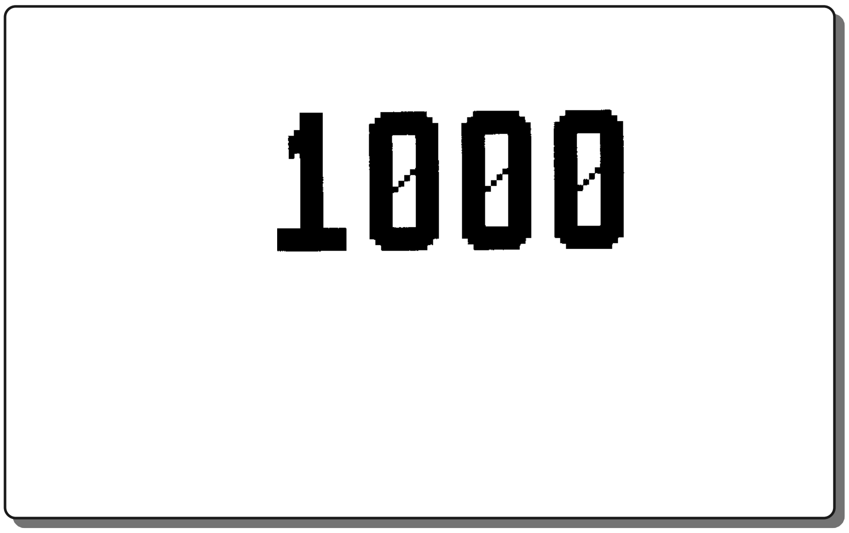
```
z t CR
```

z - Select slashed zero		
t	=	0 - (zero - prints slashed zeros (Ø))
	=	O -(upper case letter O - prints unslashed zeros (0))

Example:

```
z0
J
S l1;0,0,68,71,100
T 25,25,0,-3,x9,y9;1000
A1
```

Prints the number 1000 with slashed zeroes.



Chapter 5: Label Format Commands

Label Format Commands

Instructions with uppercase letters are used to describe the label itself.
This has a fix structure, beginning with the start command, the description of the labelsize and description of each object in the label. At the end of the label the printer expects the command for amount of labels to print.
The printer starts printing when the amount command is received, unless it is suppressed by special options.

A - Amount of Labels

The A command is used to define the end of the label definition and to set the amount of labels to be printed. The printer repeats internally the defined label where the amount is defined by this command. The label will stay in the printer´s internal buffer, after it has been sent to the printer.
sending the A command multiple times afterwards will print the amount of labels which is specified by the A command.

Syntax:

```
A [n] CR
```

A - amount of labels		
n = number of labels to print (Multiple options are available:)		
[NOPRINT]	=	receives and processes the label, but suppresses a printout. (Used for saving a label on memorycard). It is also possible to key in **[NO]** instead of **[NOPRINT]**
[?]	=	printer prompts on its display for the quantity or is also used to be replaced from any attached system
[REPEAT]	=	Repeats the label at the end (makes only sense together with the [?]option. It is also possible to use **[R]** instead of **[REPEAT]**
[$DBF]	=	Prints each record of a database. Number of records = number of labels.
A -without any value prints until the print job is cancelled		

Example:

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

prints 550 labels with the text line: „LABEL PRINTER“

Example:

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

prints "infinite" amount of labels

A - Amount of Labels

Example:

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

Transmits the label for further usage into the label buffer. The Printout is suppressed with the **[NOPRINT]** option.

*It is also possible to shorten the **[NOPRINT]** option into **[NO] -** which has the same function.*

Example:

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

Requests the user (on the printer´s display) for data entry ([?:Input?]) and prompts for the amount of labels to print.
The data entry will be done through the printers control panel or through an attached keyboard.

Example:

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

Prints all records of the database CDPLAYER.DBF, where the serial numbering function is used to create the index file, starting at 100.

Special function: Transmitting „A“ without parameter causes the printer to print a infinite number of labels.
Don´t forget the „carriage return“ after the last command in the label !

B - Barcode Definition

The B command defines a barcode field in the label format. The most common barcode types are supported by the cab printers.

The parameters for each barcode are different, depending on the selected barcode type.
Barcodes can be printed in one of four different directions (0°,90°,180° and 270°). Height and width of the barcode elements are adjustable. Human readable text lines can be easily added.

Syntax:

```
B[:name;]x,y,r,type[+options],size;text CR
```

B - Barcode field		
[:name;]	=	Optional fieldname
x	=	X - Coordinate
y	=	Y - Coordinate
r	=	Rotation
type	=	Barcode type
[+options]	=	Optional parameters
size	=	Barcode height and width, ratio
text	=	Barcode data

This is the global structure of a barcode field, a detailed description follows on the next pages

B - Barcode Definition

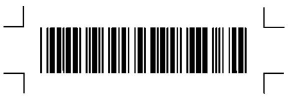
B - Descriptor of a Barcode field, this is identified by the printer that the following data is used to create a barcode.

[:name;]	=	describes the **field name** and is optional. The maximum length of this name is 10 characters, no special characters allowed. A field name can be used for further operations, such as calculations ,as linked field, for field replacements or for the enhanced usage when downloaded to a memorycard etc. The field name must be unique in each label.
x	=	The **x - coordinate** is the horizontal start position of a barcode (in millimeters or inches), the distance between the left margin of a label and the upper left corner of the barcode.
y	=	The **y - coordinate** is the vertical start position of a barcode, the distance between the top margin of a label and the upper left corner of the barcode. The maximum coordinate depends on the printer type. Please refer to the operator´s manual.
r	=	**Rotation** - Rotates a barcode in 4 directions. Valid values are 0, 90, 180 and 270. Measurement in degrees.
type	**=**	**Barcode type** - This defines the barcode symbology. Barcode types with upper case names produce barcodes with human readable characters, while lower case names for the barcodes suppress the human readable line. The size of the human readable characters are depending on the selected barcode type. More details are shown in the examples on the following pages. cab printers are able to extract necessary portions of a barcode ame, which means that e.g. EAN-13, EAN 13 and EAN13 will print identical results.

continued on the next page

B - Barcode Definition - options overview

[+options] Depending on the barcode type, several options are available. Which option is valid for which barcode is described for each barcode type on the next pages. Following options are available:

+MODxx	=	offers the possibility to add a modulo check digit to a barcode **MOD10** adds a modulo 10 check digit **MOD11** adds a modulo 11 check digit **MOD16** adds a modulo 16 check digit **MOD36** adds a modulo 36 check digit **MOD43** adds a modulo 43 check digit The available check digits depend on the barcode type
 **+WSarea**	=	white space area **-** prints white zone markers for design purposes. The white space size defines the quiet zone which is required for a good scanability of the printed code. **„area“** defines the size of the markers which are shown with this command. (can be also "0")
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+XHRI	**=**	(Extended Human Readable Interpretation) adds start - and stop characters (*) for Code 39. Adds start and stop boxes for Code 93. Reduces the size of UPC-A and UPC-E (see details in the examples)
+NOCHECK	=	suppresses the check digit calculation for variable weight barcodes (EAN-13 and UPC-A with specific start numbers : 20...29)
+ELx	=	Error Level . sets the redundancy of some 2D barcodes. Valid values for x depends on the barcode type - please see the details later in the manual
+RECT	=	Barcode type DataMatrix can be printed as a rectangle or as a square. The default value is square. The **+RECT** option forces the printer to print this barcode as a rectangle.

continued on the next page

B - Barcode Definition - options overview

+VERIFYn	=	Used to verify the barcode data. +VERIFYn needs a barcode testing equipment which is available as an option. If required please ask cab Produkttechnik for that additional barcode reader and describe the application. cab offers a solution for 1 D and 2D codes whereby the scanner is attached through a specific interface directly in front of the printer. **+VERIFYn** does a string comparision with the data received by the printer plus the calculated checksum. „**n**“ is the starting value in millimeters or inches, whatever is set up in your label. *Restrictions:* *1. + VERIFYn can be used only once in a label and starts the scan when the barcode arrives in the read window of the scanner.* *2. +VERIFYn does not work when a barcode is sent as graphics to the printer. For graphical barcodes use the „GOODBAD“ function, described later in the chapter.* *3. Functionality and technical possibilities depend strongly on the barcode reader type. Please refer to the barcode reader manual for detailed information. Please contact cab for further information.*

Example:

```
J
S l1;0,0,68,70,100
O R
B 10,16,0,CODE39+VERIFY0,20,.5,4;987656789
A 1
```

GOODBADn	=	and content check - attached USB sanner verifies the data. In this example, the scanner starts at 0 mm from top of the barcode with scanning and compares the read data with the transmitted data string.

B - Barcode Definition - options overview

+GOODBADn	=	Same function as +VERIFYn without checking the content. Only good read or bad read will be controlled. Checks the answer on NoReadString „?“ „**n**“ is the starting value in millimeters or inches, whatever is set up in your label.

Example:

```
m m
J
S l1;0,0,68,70,100
O R
B 5,12,0,CODE39+GOODBAD0,5,.5,4;1234567890
A 1
```

In this example, the scanner starts at 5 mm from top of the barcode with scanning and verifies only if the barcode is readable or not (GOOD or BAD)
NO content check will be done in this case.

,GOODBADn	=	Controls the readability of barcodes which have been transmitted as graphics (i.e. by some labelling programs). Controls only good read or bad read. „**n**“ is the starting value in millimeters or inches, whatever is set up in your label.

Example:

```
m m
J
S l1;0,0,68,70,100
O R
I 10,10,0,1,1,GOODBAD0;PICT1
A 1
```

In this example, the scanner reads the previously downloaded graphical barcode and does a good read or bad read check.

+VERIFYn, +GOODBADn and ,GOODBADn are available for all barcodes, this will not be mentioned explicit in the decription of each single barcode on the following pages.

+CCn	=	defines the height of a composite line in module width. Default value is 2 and the maximum value is 99.

continued on the next page

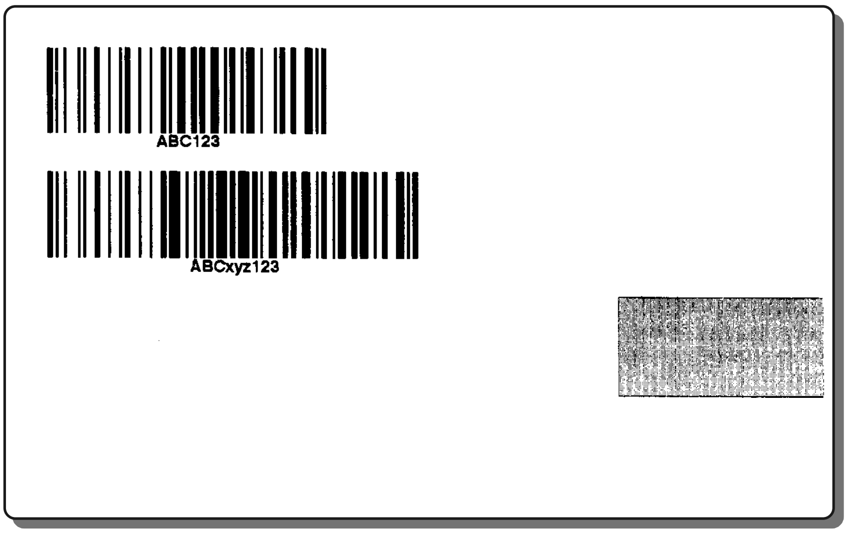
B - Barcode Definition - overview

size	=	**S**tandard **C**odesize. Defines the height and width of the bars in a barcode. Height and narrow element is defined for ratio oriented barcodes. For EAN, JAN or UPC it is also possible to define the standard code size which is expressed through „SCx“. The height calculation includes the human readable characters if enabled. Unified barcode sizes of EAN and UPCbarcodes. Sets the size of the barcode to a defined standard code size. x is a numeric value (0-9) and the possible barcode size depends on the printer´s resolution. Used <u>instead</u> of height and ne (narrow element)
height	=	Defines the barcode height in the pre selected measurement - millimeters or inches. A-series printers will print a grey rastered field if the barcode does not fit including the white space area on the label.
ne	**=**	**narrow element** Defines the width of the smallest element of the barcode. The input is in millimeters or inches. The narrow element (ne) size depends on the printer´s resolution. One dot is the smallest possible element - therefor it depends on the printhead resolution-how big or how small the thinnest line can be printed. (it is not possible to print a „half“ dot)
ratio	**=**	The ratio between narrow and wide bars. (i.e. 3:1 means that the widebar is three times the width of the small bar)
text	=	contains the barcode data to be encoded in a barcode. Depending on the selected barcode type. Different rules are used for different barcodes. Some barcodes allow only characters, some others have a fixed length etc. More information can be found at the samples of each barcode.

B - Barcode Definition

cab printers will print a rastered area if a barcode would not fit on the label. The printers intelligence checks this for you to avoid later reading problems. This includes also the required white space for the barcode readability. Check the barcode witdh, height and x / y positions to make sure that the barcode is placed correct.

The following picture shows what happens when a barcode is misplaced.
A-series printers will print a raster instead of a barcode as demonstrated on the following label in the lower right corner.



misplaced barcode

Most printers also allow the selection in the printer setup to switch to „barcode error on“ to verify if the incoming data is correct for the selected barcode. In case of an error the printer will show an error message in its display.

Barcode overview list

Size options on ratio barcodes are different to the size options of non ratio barcodes.
Capital letter for the barcode name produce barcodes with human readable text line, as far as this is defined in the barcode specs. Capital or lower case letters have no influence on barcodes which are not specified to have a human readable textline.

Shortcode: For a limited time shortcodes have been used alternatively which are no longer supported. Therefor we highly recommend that these short codes will no longer be used !! We added these short codes to the overview table, in the case if you need to debug some old program code.

Barcode name	old Shortcode	Ratio	1D /2D code*
2 of 5 Interleaved	D	yes	1D
Add-On 2	M	no	1D
Add-On 5	N	no	1D
Aztec Code	—-	no	2D
Codabar	I	yes	1D
Codablock F	—-	no	stacked
Code 39	A	yes	1D
Code 93	O	no	1D
Code 128	E	no	1D
Data Matrix	W	no	2D
DBP (German Post code)	—-	yes	1D
EAN 8	G	no	1D
EAN 13	F	no	1D
EAN 128	Q	no	1D
FIM	S	no	1D
German Parcel	—-	yes	1D
JAN 8	—-	no	1D
JAN 13	—-	no	1D
HIBC	H	yes	1D
MaxiCode	U	no	2D
Micro PDF	—-	no	2D
MSI	K	yes	1D
PDF-417	Z	no	2D
Plessey	X	yes	1D
Postnet	P	no	1D
QR -Code	—-	no	2D

**1D = One dimensional barcode, 2D = Two dimensional barcode*

Barcode name	old Shortcode	Ratio	1D /2D code*
RSS-14	-		1D
RSS-14 composite CC-A	-		composite
RSS-14 truncated	-		1D
RSS-14 truncated composite	-		composite
RSS-14 truncated composite	-		composite
RSS-14 stacked	-		stacked
RSS-14 stacked composite	-		composite
RSS-14 stacked composite	-		composite
RSS-14 stacked omnidirectional	-		
RSS-14 stacked omnidirectional composite	-		composite
RSS-14 stacked omnidirectional composite	-		composite
RSS limited	-		
RSS limited composite	-		composite
RSS limited composite	-		composite
RSS expanded	-		
RSS expanded composite	-		composite
RSS expanded composite	-		composite
RSS expanded stacked	-		
RSS expanded stacked half line	-		
RSS expanded stacked composite (CC-A)	-		composite
RSS expanded stacked composite (CC-B)	-		composite
UCC 128	Q	no	1D
UPC-E0	C	no	1D
UPC-A	B	no	1D
UPC-E	Y	no	1D

**1D = One dimensional barcode, 2D = Two dimensional barcode*
A composite barcode contains 1D and 2D code elements
We highly recommend to read carefully the specifications of the required barcode which is available from the responible organisation, whenever a barcode needs to be printed !
The usage of a barcode reader / verifier is also recommended, when barcodes are used, to verify the contents and the readability of the printout.

Available check digits:

MOD 10	(numerical data only).
MOD 10	(for MSI is calculated different (Weighting 2/1 instead of 3/1).
MOD 10	GP (2 of 5, Weighting 3/1 + 1, - German Parcel only).
MOD 11	(numerical data only).
MOD 16	(Codabar only).
MOD 36	(CODE 39 only)
MOD 43	(only Code 39 and Code 128).

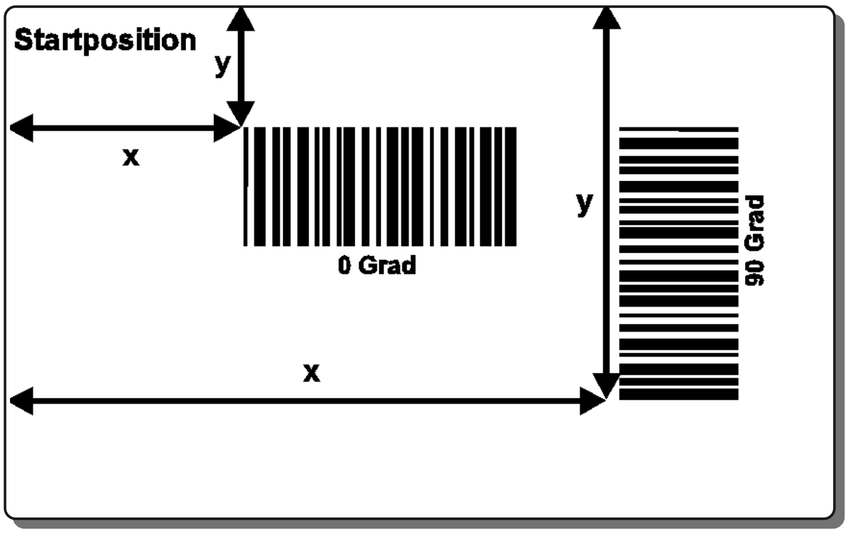
Code 128 and EAN/UCC-128 use automatically modulo 103 check digit.
EAN-13, EAN-8, UPC-A, UPC-E and UPC-E0 use automatically modulo 10 check digit.

POSTNET uses automatically modulo 10 (without weighting).
DBP is the 12- or 14-digit barcode of the Deutsche Post AG. It uses automatically modulo 10 check digit with weighting 4/9. It is allowed to add dots and spaces as much as it might be required.

Each barcode has own specs which are defined by the responsible organization who developed the specific barcode type.
We recommend to read and follow the barcode specifications of the responsible organisations.
It is also recommended to test the printed barcodes for scanability !

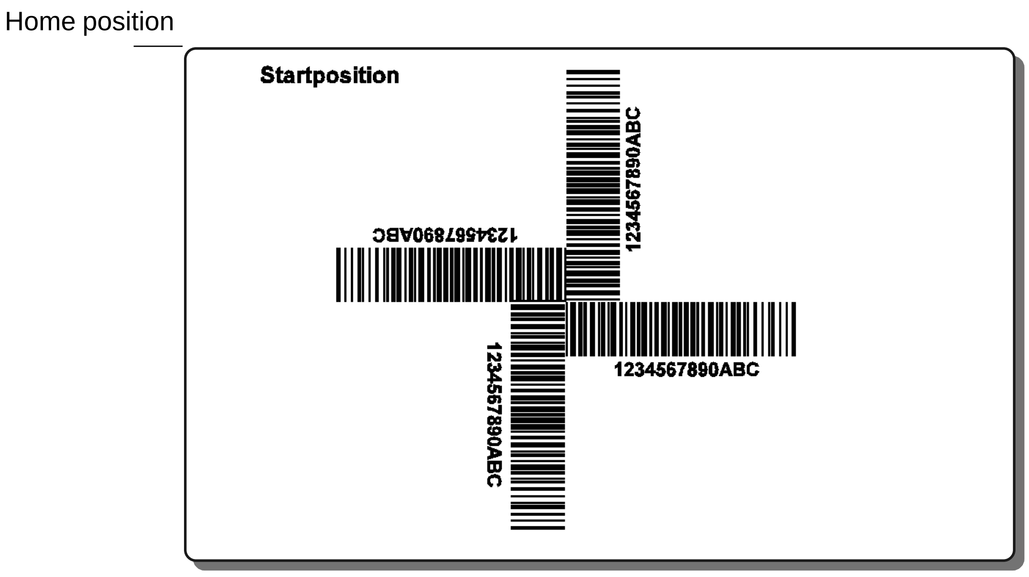
Startpositions of Barcodes

The picture below shows the start position of barcodes. Please see also the option command „O“, which offers a couple of possibilities to manipulate the complete label.



Barcodes - printing direction

In the following picture it is shown how it looks when a barcode is rotated. The X and Y starting points are identical. Only the rotation parameter has been changed. Barcodes can be rotated in an angle of 90 degrees. So rotation 0,90,180 and 270 degrees has been used for the label below.



B - Barcode 2 of 5 Interleaved

Barcode type:	**2 of 5 Interleaved**
Length:	variable, always even.
Valid characters:	numeric,
	digits: 0-9,
check digits:	optional
ratio oriented:	yes
	Encodes numbers in pairs

The 2 of 5 interleaved (interleaved 2/5) is a numerical barcode which encodes the numbers pairwise. Automatically a leading zero is added, if the number is odd. Interleaved 2of 5 can be printed very small as it contains only numeric values.

Syntax:

```
B[:name;]x,y,r,2OF5INTERLEAVED[+options],height,ne,ratio;textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**2OF5INTERLEAVED**)

[+options] Following options are available:

+MODxx	=	calculation of modulo check digit. (MOD10)
+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

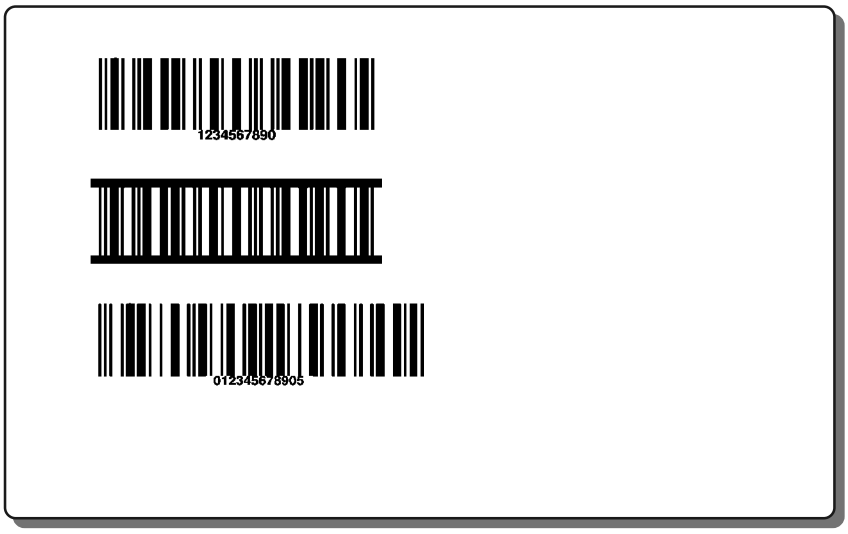
Detailed descriptions at the beginning of the barcode chapter.

B - Barcode 2 of 5 Interleaved

Example:

```
m m
J
S l1;0,0,68,71,100
B 5,5,0,2 OF 5 INTERLEAVED,10,0.3,3;1234567890
B 5,20,0,2of5interleaved+BARS,10,0.3,3;1234567890
B:Bar3;5,35,0,2OF5 INTERLEAVED+MOD10,10,0.3,3;1234567890
A 1
```

Print three barcodes with some modifications (with an without human readable characters, upper and lower bar and with a modulo 10 checksum.



B - Barcode Add-On2

Barcode type:	Add-on2 (EAN/UPC Addendum 2)
Length:	fixed 2-digits
Valid characters:	numeric only
check digits:	no
ratio oriented:	yes

Add-On2 is an addendum code which is used together with EAN or UPC barcodes. Mainly used for magazines to diplay the magazine publication release (normally a 2 digit number of the week or month)
The size must fit to the printed size of the EAN or UPC code. We recommend to use SC sizes with this barcode.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**ADDON2**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

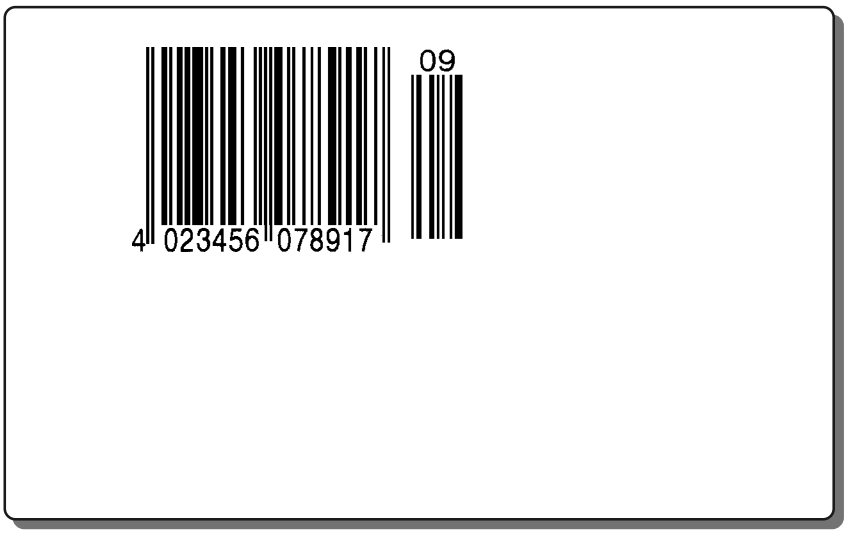
size	=	Standard Codesize **SCx** (instead of height and ne)
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Add-On2

Example:

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



B - Barcode Add-On5

Barcode type:	Add-on5 (EAN/UPC Addendum 5)
Length:	fixed - 5 digits
Valid characters:	numeric only
check digits:	no
ratio oriented:	yes

Add-On5 is an addendum code which is used together with EAN or UPC barcodes. Mainly used for books (ISBN number) and magazines to diplay the magazine publication release or the price.
The size must fit to the printed size of the EAN or UPC code. We recommend to use SC sizes with this barcode.

Syntax:

```
B[:name;]x,y,r,ADDON5[+options],height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**ADDON5**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

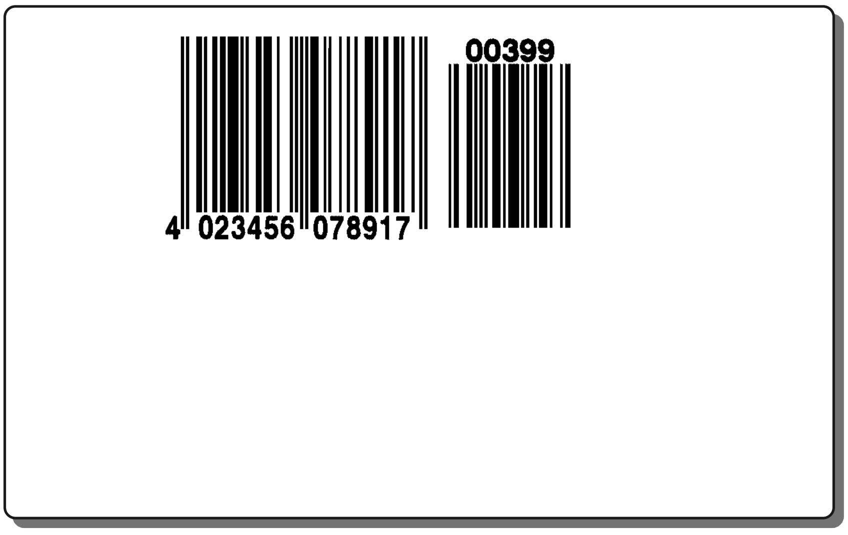
size	=	Standard Codesize **SCx** (instead of height and ne)
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Add-On5

Example:

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



B - Barcode Aztec - Code

Barcode type:	Aztec - Code
Length:	2D - Code with variable Length
Valid characters:	alphanumeric

Aztec Code is a 2 - dimensional matrix symbol developed by Welch Allyn. It was designed using the combination of the best characteristics of the first generation 2D codes.

Syntax:

```
B[:name;]x,y,r,AZTEC,[+options],dotsize;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**AZTEC**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional 2D barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.
+ELx	=	Error Level (5 - 95)

dotsize	=	dot size in millimeters or inches
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Aztec - Code

Example:

```
m m
J
S l1;0,0,68,71,100
B 5, 5,0,Aztec+EL55,1;CAB Produkttechnik GmbH & Co KG
B 45,5,0,Aztec+EL90,0.6;CAB Produkttechnik GmbH & Co KG
A 1
```

The same barcode contents with variations on error level and dot size.

B - Barcode Codabar

Barcode type:	Codabar
Length:	variable
Valid characters:	numeric, special characters: - $: /. + and special start stop codes (A,B,C,D)
check digits:	yes (Mod 16)
ratio oriented:	yes

Each character of this barcode is built with 7 elements (bars and spaces), where the spaces do not contain information. Codabar ist mostly used in medical environments for photo laboratories and libraries. The exact specifications are described in the Norm: EN 798. The start and stop characters are additionaly A,B,C or D.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**CODABAR**)

[+options] Following options are available:

+MODxx	=	calculation of modulo check digit (**MOD 16**)
+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

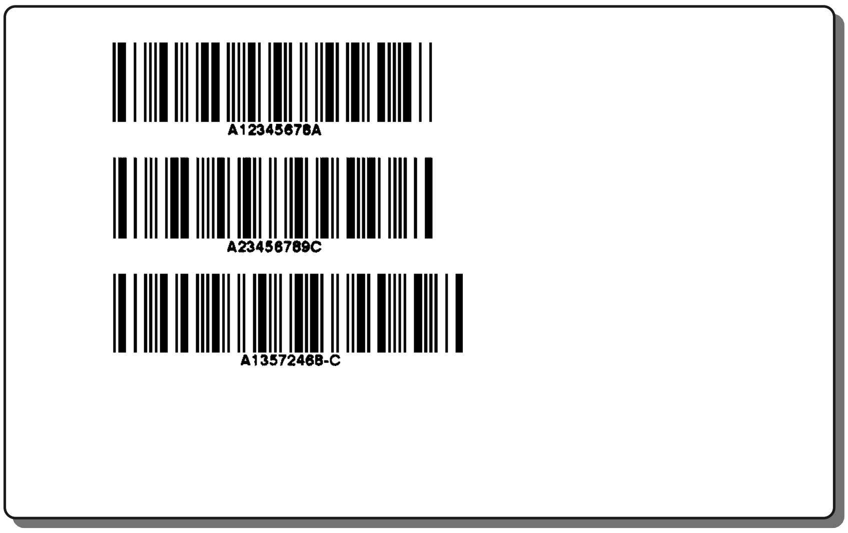
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Codabar

Example:

```
m m
J
S l1;0,0,68,71,100
B 5, 5,0,CODABAR,12,0.3,3;A12345678A
B 5,20,0,CODABAR,12,0.3,3;A23456789C
B 5,35,0,CODABAR+MOD16,12,0.3,3;A13572468C
A 1
```



B - Barcode Codablock F

Barcode type:	Codablock F
Length:	variable
Valid characters:	alpha numeric, max. 2725 Characters stacked barcode
check digits:	yes (Mod 43)
ratio oriented:	no

Codablock F: Based on the structure of Code 128, can consist of 2 - 44 lines in a length of 4-62 characters. Requires big space for printing.
Codablock was developed at a time where more information needed to be encoded in a barcode, before 2D codes existed. Today Codablock F is a seldom used barcode, as 2D codes offer better compression and smaller sizes.

Syntax:

```
B[:name;]x,y,r,CODABLOCKF[+options],height,ne,ratio;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**CODABLOCKF**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

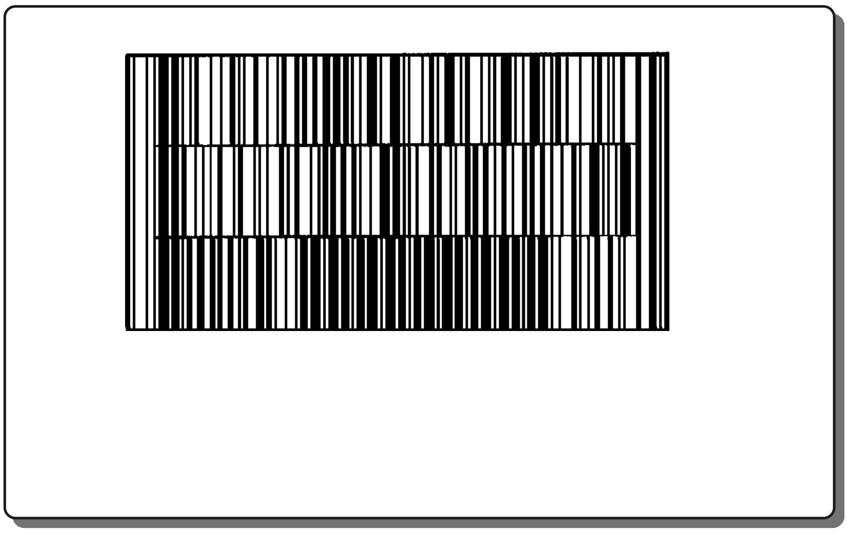
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Codablock F

Example:

```
m m
J
S l1;0,0,68,71,100
B 5, 5,0,CODABLOCKF,12,0.3,3;CAB Produkttechnik GmbH & Co KG
A 1
```



B - Barcode Code 39

Barcode type:	Code 39 (Code 3 of 9)
Length:	variable
Valid characters:	alphanumeric, uppercase A-Z, digits: 0-9, special characters: $ / + % .- and space
check digits::	no
ratio oriented:	yes

Code39 is designed to encode 26 upper case letters, 10 digits and 7 special characters.Start/ Stop characters are added automatically. Invalid characters are automatically transformed into spaces.
Start/stop characters will be printed as „ * „ when the option +XHRI (Extended Human Readable Interpretation) is used. Most common ration for this barcode is 3:1 .The printers convert automatically lower case letters into upper case letters, if lower case letters are keyed in.

Syntax:

```
B[:name;]x,y,r,CODE39[+options],height,width,ratio;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**CODE39**)

[+options] Following options are available:

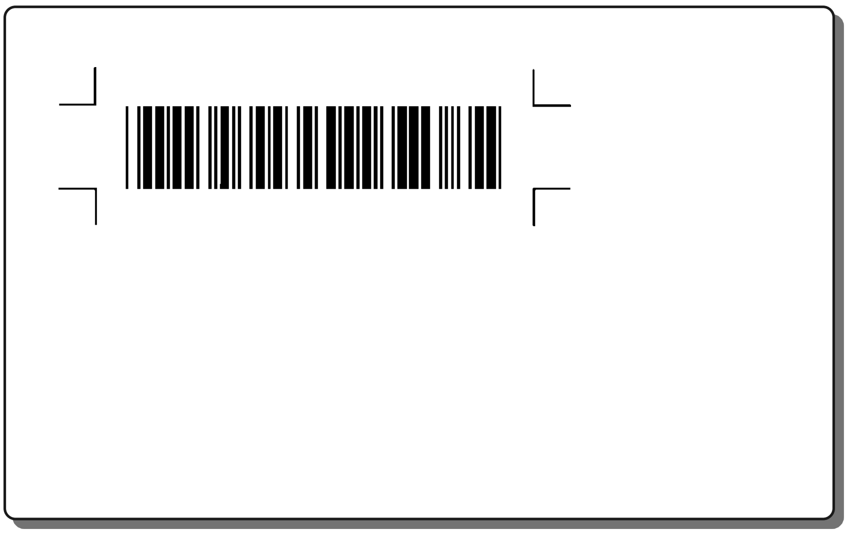
+MODxx	=	calculation of modulo check digit (Here **MOD 43**)
+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.
+XHRI	**=**	(Extended Human Readable Interpretation

height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Code 39

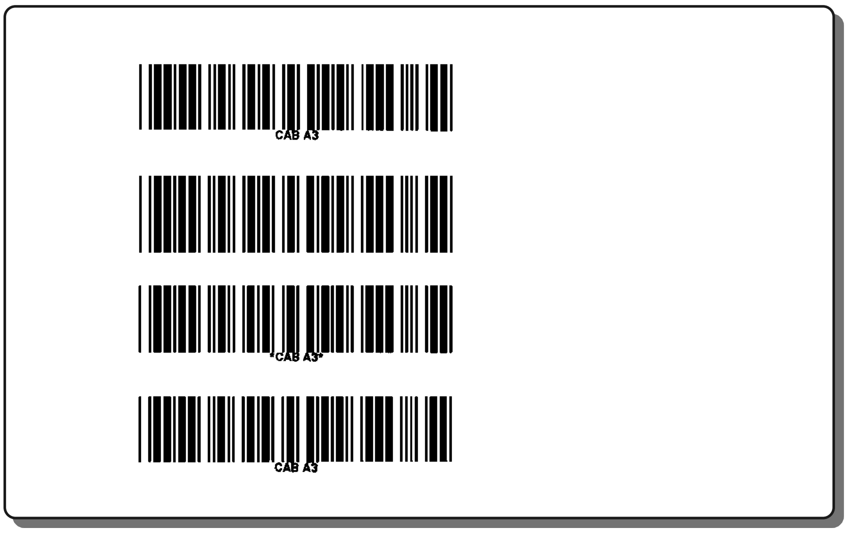
This picture shows the functionality of the WSarea



Example:

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

This example shows how the barcode varies with different options



B - Barcode Code 39 FULL ASCII

Barcode type: Code 39 (Code 3 of 9)

Length: variable
Valid characters: alphanumeric, Full ASCII

check digits: no
ratio oriented: yes

Code 39 Extended (Full ASCII) – this encoding variant allows the full ASCII table, 128 characters to be encoded.
Start/ Stop characters are added automatically. Invalid characters are automatically transformed into spaces.
Start/stop characters will be printed as „ * „ when the option +XHRI (Extended Human Readable Interpretation) is used. Most common ratio for this barcode is 3:1

.

Syntax:

```
B[:name;]x,y,r,CODE39FULL[+options],height,width,ratio;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**CODE39FULL**)

[+options] Following options are available:

+MODxx	=	calculation of modulo check digit (Here **MOD 43**)
+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.
+XHRI	**=**	(Extended Human Readable Interpretation

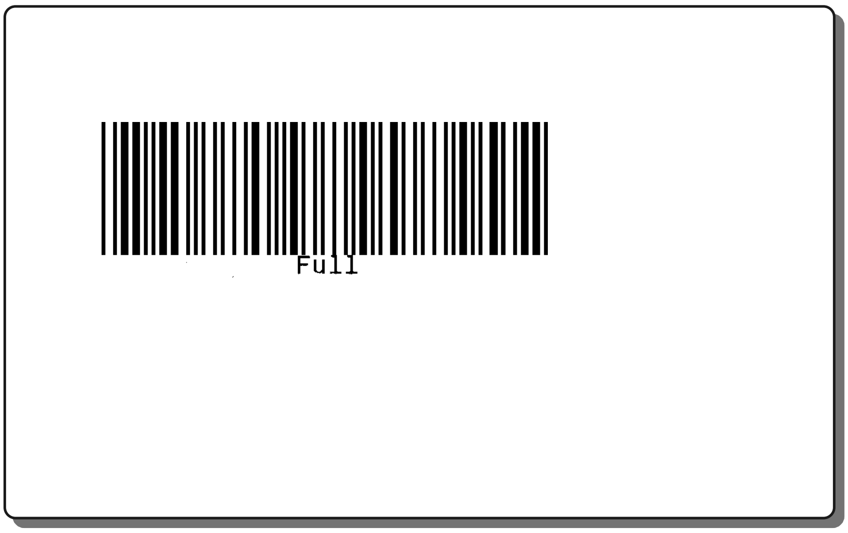
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode   Code 39 FULL ASCII

Example:

```
m m
J
S l1;0,0,68,71,100
B 10,30,0,CODE39FULL,20,0.5;Full
A 1
```



B - Barcode   Code 93

Barcode type:	Code 93
Length:	variable
Valid characters:	alphanumeric, encodes all 128 ASCII characters including control characters
check digits:	yes
ratio oriented:	no

Code 93 is a alphanumeric barcode which can contain all 128 ASCII characters including the control characters. The checksum is automatically calculated by the cab printers.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**CODE93**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.
+XHRI	**=**	(Extended Human Readable Interpretation

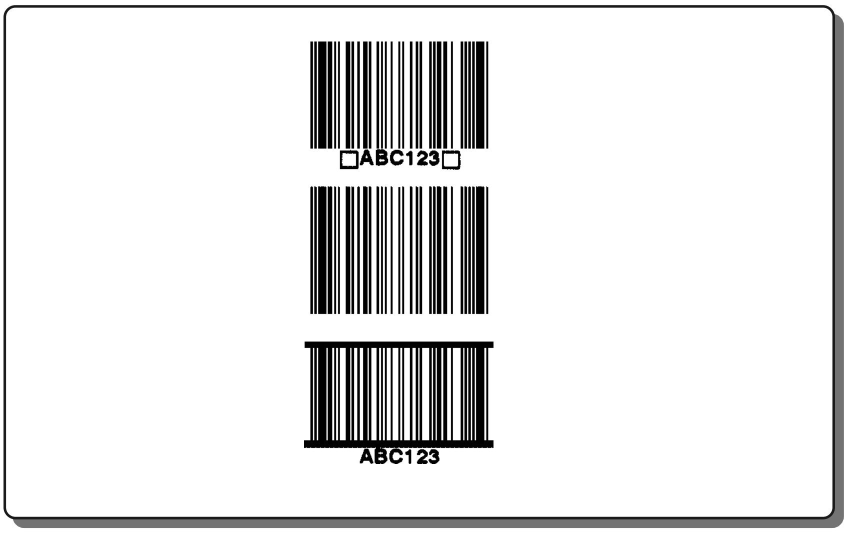
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Code 93

Example:

```
m m
J
S l1;0,0,68,71,100
B 25, 5,0,CODE93+XHRI,16,0.28,3;ABC123
B 25,24,0,code93,16,0.28,3;ABC123
B 25,44,0,CODE93+BARS,16,0.28,3;ABC123
A 1
```



B - Barcode Code 128

Barcode type:	Code 128
Length:	variable
Valid characters:	all 128 ASCII characters
check digits:	yes (MOD 103)
ratio oriented:	no

Code 128 has a modulo 103 check digit which is the standard check digit of this barcode. An additional check digit can be added with the +MOD option if required. Code 128 consists of 3 code subsets. cab printers select automatically the best subset of this barcode as described in the code 128 specification.The best subset is the subset with the highest data compression as described in the original specs of code128.

Syntax:

```
B[:name;]x,y,r,CODE128[+options],height,ne;[U:subcode]text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**CODE128**)

[+options] Following options are available:

+MODxx	=	calculation of modulo check digit (**MOD43** and **MOD10**)
+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data
[U:subcode]	=	Enables the selection of a specific subcode, Valid input: [U:CODEA], [U:CODEB] or [U:CODEC]

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Code 128

Subcode A
contains uppercase alphanumeric characters, special characters and control characters. The printer can be forced to use subcode A with the option: [U:CODEC] in the barcode text string.

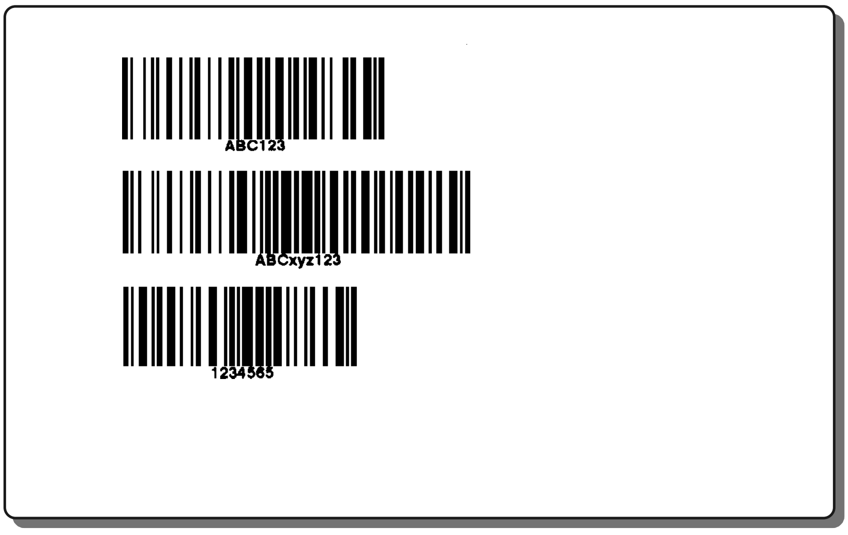
Subcode B
contains all standard characters, upper case, lower case, special characters and control characters. Subset B is the default value when data is transmitted.
The printer can be forced to use subcode B with the option:
[U:CODEB] in the barcode text string.

Subcode C
is used to encode exeptional numeric values with a good compression rate.
Encodes pairs of numbers.
The printer can be forced to use subcode C with the option: [U:CODEC] in the barcode text string.

FNC1 can be added in the barcode data as " [FNC1] "

Example:

```
m m
J
S l1;0,0,68,71,100
B 5, 5,0,CODE128,12,0.3;ABC123
B 5,20,0,CODE 128,12,0.3;ABCxyz123
B 5,35,0,CODE128+MOD10,12,0.3;[U:CODEC]123456
A 1
```



B - Barcode Data Matrix

Barcode type:	Datamatrix
Length:	2D - Barcode - up to 2335 ASCII characters
Valid characters:	alpha numeric
	all 128 ASCII characters

The Data Matrix symbol is a 2 Dimensional symbology used to encode large amounts of text and data securely and inexpensively. Up to about 2335 ASCII characters can be encoded in a Data Matrix symbol. We recommend to limit this to maximum 800 characters, as the most 2D barcode readers have problems to decode symbols which use a higher amount of data.
The cells of a Data Matrix code are made up of square modules that encode letters, numbers, text and actual bytes of data, and encode just about anything including extended characters, unicode characters and photos.

Syntax:

```
B[:name;]x,y,r,DATAMATRIX[+options],dotsize;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**DATAMATRIX**)

[+options] Following options are available:

+WSarea	=	white space area
+RECT	=	forces the printer to print this barcode as rectangle
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

dotsize	=	dot size in millimeters or inches
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Data Matrix

The encoding and decoding process of Data Matrix is very complex and several methods have been used for error correction in the past. ECC200 is the newest and most standard version of data matrix error correction. It supports advanced encoding and error checking with Reed Solomon error correction algorithms. These algorithms allow the recognition of barcodes that are up to 60% damaged.

Example:

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

B - Barcode DBP - German Post Identcode

Barcode type:	DBP - German Post Identcode Code (**DBP -** Ident- und Leitcode der Deutschen Bundespost)
Length:	11 or 13 digits
Valid characters:	numeric,
check digits:	yes
ratio oriented:	yes

Developed by the Deutsche Post AG for automated sorting of mails. Base code is a 2of 5 interleaved barcode with the fixed length of 11or 13 digits and an additional check digit.
cab printers convert invalid characters automatically into zeroes, while the human readable shows a hash sign.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**DBP**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

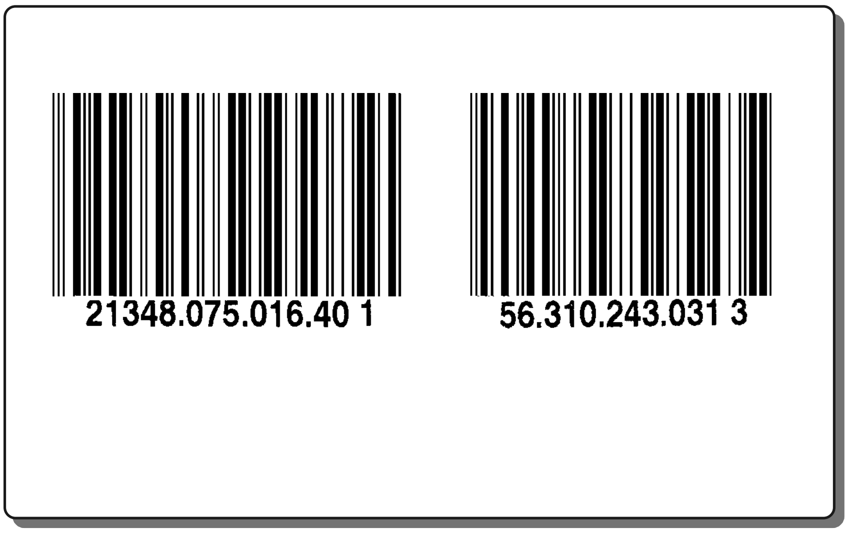
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode   DBP - German Post Identcode

Example:

```
m m
J
S l1;0,0,68,71,105
B  5,10,0,DBP,10,0.3;2134807501640
B 60,10,0,DBP,10,0.3;56.310.243.031
A 1
```



B - Barcode EAN-8 / JAN-8

Barcode type:	EAN-8 / JAN-8 (European / Japanese Article Numbering)
Length:	fixed - 8 digits
Valid characters:	numeric, digits: 0-9,
check digits:	yes
ratio oriented:	no

The EAN 13 code is used in retail environment in Europe with a fixed length of 8 digits. The 8th digit contains the calculated checksum. cab printers expect 7 digits, while the 8th digit is calculated by the printer.
JAN 8 is the japanese version of EAN 8.

Syntax:

```
B[:name;]x,y,r,EAN8[+Options],height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	=	Barcode type (**EAN8**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+XHRI	=	(Extended Human Readable Interpretation
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

size	=	Standard Codesize **SCx** (instead of height and ne)
height	=	Barcode height
ne	=	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode EAN-8 / JAN-8

Example:

```
m m
J
S l1;0,0,68,71,100
B 10, 5,0,EAN8,SC1;4023456
B 10,26,0,EAN8,16,0.35;4023456
B 10,44,0,JAN8,16,0.35;4900056
A 1
```

4023 4564
4023 4564
4900 0566

B - Barcode EAN-13 / JAN-13

Barcode type:	EAN-13 / JAN-13 (European / Japanese Article Numbering)
Length:	fixed - 13 digits
Valid characters:	numeric, digits: 0-9,
check digits:	yes
ratio oriented:	no

The EAN 13 code is used in retail environment in Europe with a fixed length of 13 digits. The 13th digit contains the calculated checksum. cab printers expect 12 digits, while the 13th digit is calculated by the printer.
JAN 13 is the japanese version of EAN 13.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**EAN13**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+XHRI	**=**	(Extended Human Readable Interpretation
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.
+NOCHECK	=	Check digit suppression when start no is 20...29

size	=	Standard Codesize **SCx** (instead of height and ne)
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode EAN-13 / JAN-13

Example:

```
m m
J
S l1;0,0,68,71,100
B 10,5,0,EAN13,SC1;402345607891
B 10,30,0,EAN13,16,0.35;270072610950
B 10,48,0,JAN13,16,0.35;490005607891
A 1
```

This example prints an EAN code with standard code size 1 (SC1), an EAN code where the size is defined and a JAN code with defined size.

4 023456 078917
4 023456 078917
4 900056 078915

B - Barcode EAN 128 / UCC 128 / GS1-128

Barcode type:	EAN 128 / UCC128
Length:	variable
Valid characters:	ASCII characters
check digits:	yes (Mod 103)
ratio oriented:	yes

EAN = European Article Numbering
UCC = Uniform Code Council
EAN 128 / UCC 128 is based on Code 128 and contains shipping information.
Additional info on the next page.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**EAN128**) or (**UCC128**) or (**GS1-128**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode EAN 128 / UCC 128

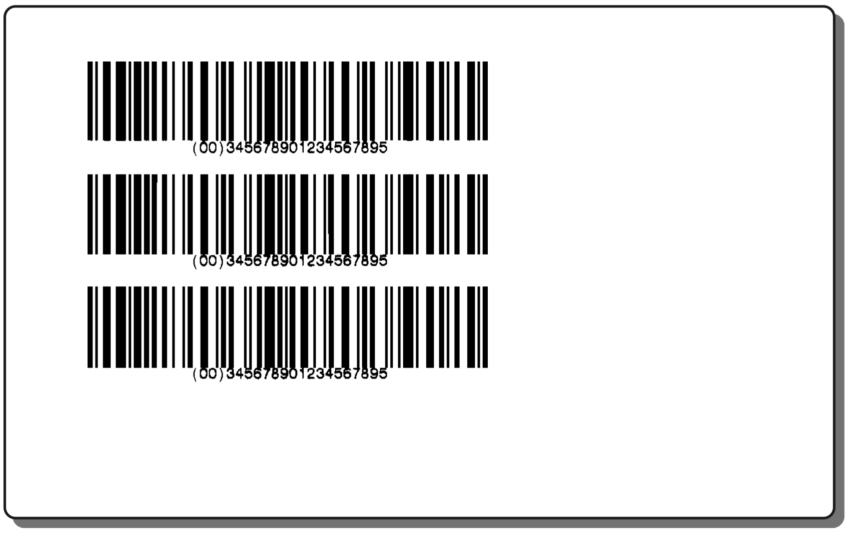
EAN 128 has very specialized contents which are described in the barcode specs of the responsible organisation. This huge amount of rules have to be used to create this barcode.

EAN 128/UCC 128 contains application identifiers which are clearly described in the specs. This barcode needs additionally a start code and some so called Application identifiers (AI).

The application identifiers are described in the barcode specifications. Allowed data contents which follows after the application identifiers depend on the application identifier its self.

Example:

```
m m
J
S l1;0,0,68,71,100
B 5, 5,0,EAN128,12,0.3;(00)345678901234567890
B 5,20,0,UCC128,12,0.3;(00)345678901234567890
B 5,35,0,GS1-128,12,0.3;(00)345678901234567890
A 1
```



B - Barcode EAN-18 / NVE / SSCC-18

Barcode type:	EAN-18 / NVE / SSCC-18 based on (EAN 128 / UCC128)
Length:	18 digits
Valid characters:	ASCII characters
check digits:	yes (Mod 10)
ratio oriented:	yes

EAN = European Article Numbering
NVE = Nummer der Versandeinheit (German name for this code)
SSCC = Serial Shipping Container Code
More details about this barcode on the next page.

Syntax:

```
B[:name;]x,y,r,EAN18[+options],height,ne; text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**EAN128**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

Do not use this barcode unless you have read the specification - available at the EAN organisation in your country !!

B - Barcode EAN-18 / NVE / SSCC-18 *

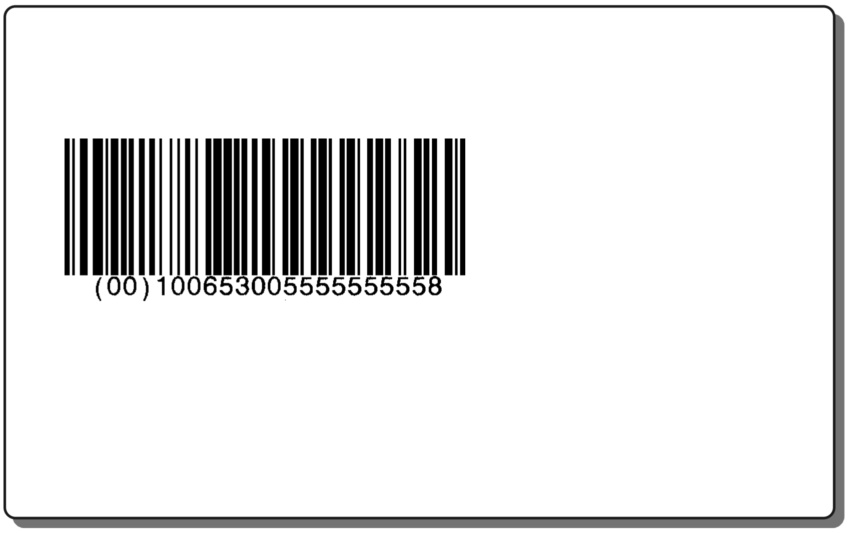
The EAN-18 / NVE / SSCC-18 is used throughout the supply chain as an identifier for product tracing and internal control. It consists always of 18 digits.

There is no special command available, as this code is based on EAN 128.
We added this description,as we got multiple requests for that barcode type.

Please see also EAN 128/UCC 128.
Structure:
- The first 2 numbers are the Application Identifier of the EAN-128: (00).
- The first digit of the data field is the extension digit. Currently a „3“ is standard.
- The next 7 digits is the company prefix.
- The following 9 digits are the serial reference number.
- The last digit is the check digit.

Example:

```
m m
J
S l1;0,0,68,71,100
B 5,20,0,EAN128,20,0.3;(00)1006530055555555558
A 1
```



B - Barcode EAN Data Matrix / GS1-Data Matrix

Barcode type:	EAN Datamatrix
Length:	2D code - more than 200 characters
Valid characters:	alphanumeric

EAN Datamatrix is a 2 dimensional symbology, where the GS1- organisation plans to improve the visibility and efficiency of supply chains across multiple sectors

GS1 developed this as a series of standards, to improve supply chain management.
Further information isavailable on the website of the GS1 organisation.
A list of all existing GS1 organisations in the respective countries can be found at Wikipedia. Search at Wikipedia for: " List of GS1 member organisations ".

Syntax:

```
B 5,20,0,EANDATAMATRIX[+options],dotsize;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**EANDATAMATRIX**) or (**GS1-DATAMATRIX**)

[+options] Following options are available:

+WSarea	=	white space area
+RECT	=	forces the printer to print this barcode as rectangle
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

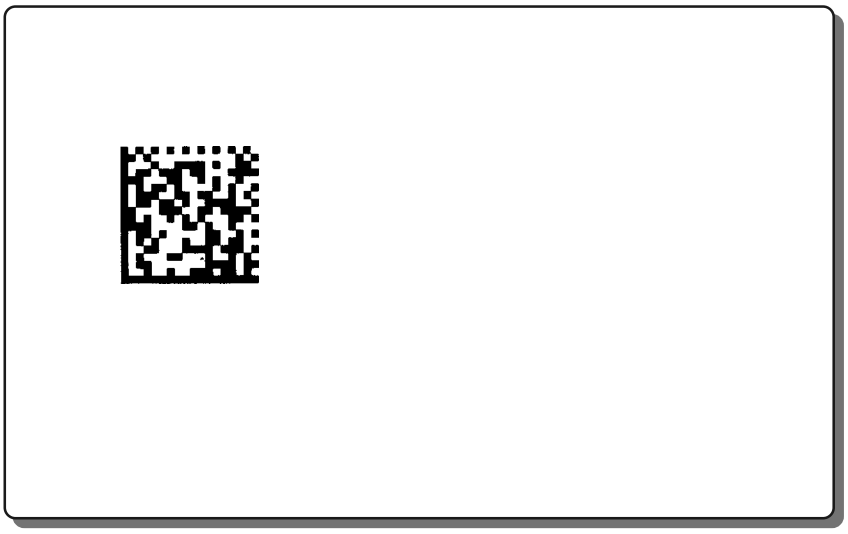
dotsize	=	dot size in millimeters or inches
text	=	Barcode data [FNC1] can be added to the barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode EAN Data Matrix

Example:

```
m m
J
S l1;0,0,68,71,100
B 5,20,0,EANDATAMATRIX,1;(01)34012345123457(10)12345(17)101231
A 1
```



B - Barcode FIM

Barcode type:	FIM (Facing Identification Mark)
Length:	fixed
Valid characters:	A,B,C or D
check digits:	yes (Mod 16)
ratio oriented:	yes

FIM Code is a barcode which is used by some postal organisations and contains only 4 patterns: A, B, C or D. FIM (Facing Identification Mark) is designed for automatic mail sorters.

Syntax:

```
B [:name;]x,y,r,FIM[+options],height;text   CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**FIM**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

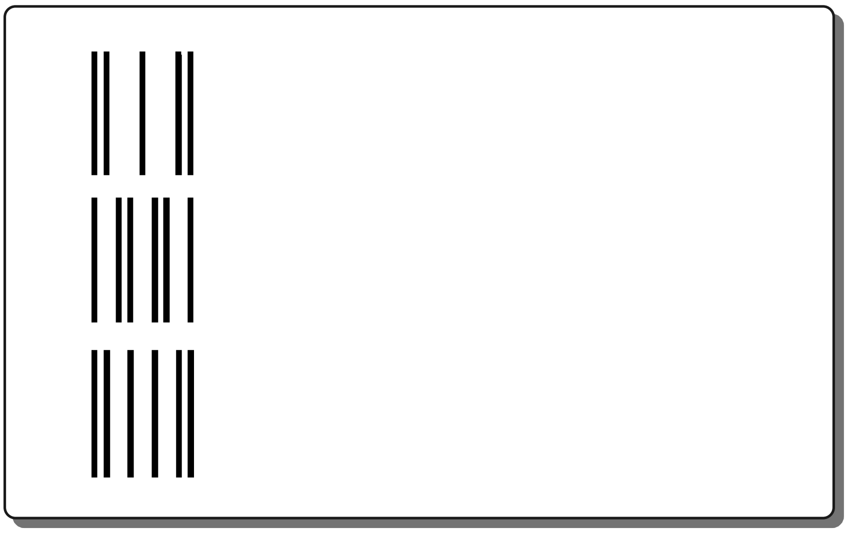
height	=	Barcode height
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode FIM

Example:

```
m m
J
S l1;0,0,68,71,100
B 5, 5,0,FIM,16,0.3,3;A
B 5,24,0,FIM,16,0.3,3;B
B 5,44,0,FIM,16,0.3,3;C
A 1
```



B - Barcode HIBC (Health Industry Barcode)

Barcode type:	HIBC
Length:	variable
Valid characters:	alphanumeric, uppercase A-Z, digits: 0-9, special characters: $ / + % .- and space
check digits:	yes (Mod 43)
ratio oriented:	yes

HIBC (Health Industry Barcode) is a modified Code 39 with amodulo 43 check digit and added start and stop characters. Leading „+“characters need to be added manually to the data string.

Syntax:

```
B[:name;]x,y,r,HIBC[+options],height,ne,ratio;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**HIBC**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

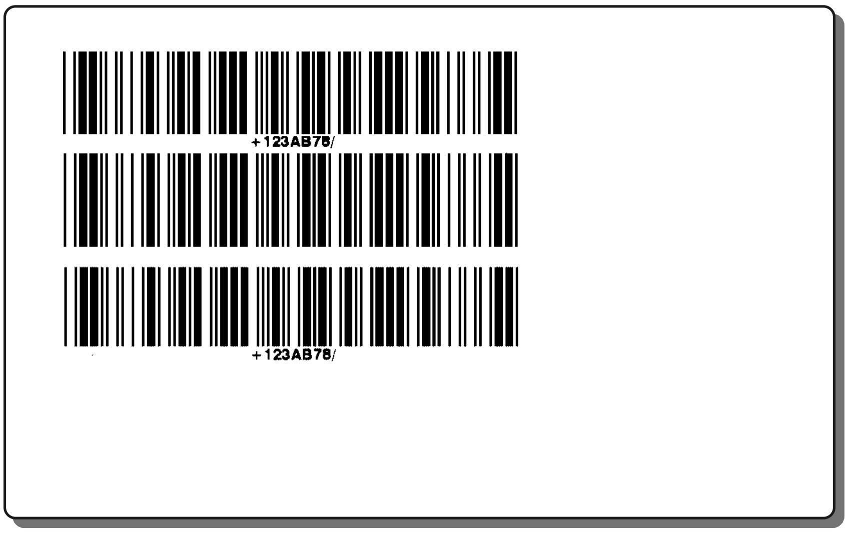
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode HIBC (Health Industry Barcode)

Example:

```
m m
J
S l1;0,0,68,71,100
B 5, 5,0,HIBC,12,0.3,3;+123AB78
B 5,18,0,hibc,12,0.3,3;+123AB78
B 5,33,0,HIBC,12,0.3,3;+123AB78
A 1
```



B - Barcode ITF-14 * / SCC-14 *

Barcode type:	ITF-14 (This code is based on the „2 of 5 Interleaved“ barcode)
	SCC-14 (Shipping container code - same barcode type)
Length:	14 digits
Valid characters:	numeric, digits: 0-9,
check digits:	Modulo 10
ratio oriented:	yes - encodes numbers in pairs

The ITF-14 is not an independently barcode.The name ITF-14 is a composition of the interleaved 2 of 5 barcode.Therefor it is no separate command available.

Here is how it works:

ITF-14 is based on the 2 of 5 interleaved (interleaved 2/5) barcode and has some restrictions. The length of this code is 14 digits fixed length. It is a numerical barcode which encodes the numbers pairwise. The first digit is a number which describes the „logistic variant“ (Packaging indicator) , followed by the contents of an EAN-13 barcode (12 digits) . The last digit is the Mod 10 check digit.

Syntax:

```
B[:name;]x,y,r,2OF5INTERLEAVED[+options],height,ne,ratio;textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**2OF5INTERLEAVED**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

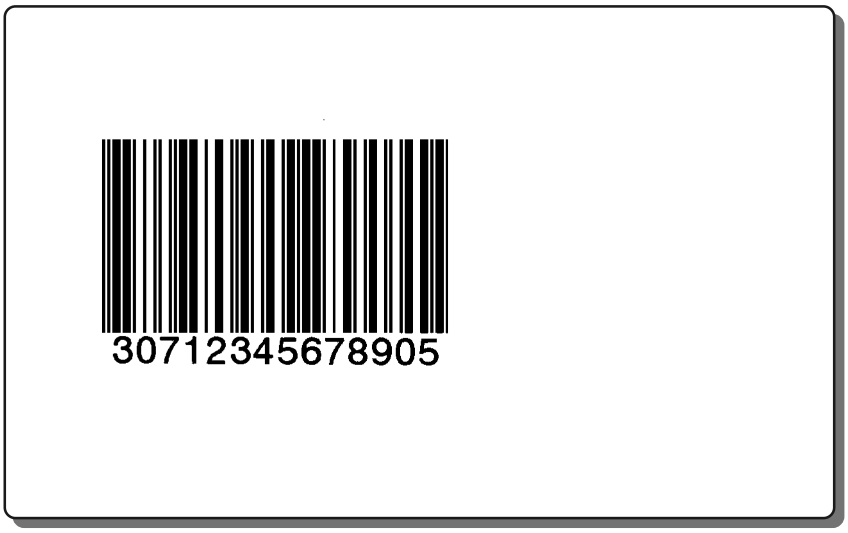
Detailed descriptions are at the beginning of the barcode chapter.

** This barcode type is based on the interleaved 2 of 5 barcode. We highly recommend to read the original specification of this barcode.*

B - Barcode ITF-14 * / SCC-14 *

Example:

```
m m
J
S l1;0,0,68,71,100
B 5,20,0,2OF5 INTERLEAVED+MOD10,30,.3,3;3071234567890
A1
```



B - Barcode Maxicode

Barcode type:	MaxiCode
Length:	2D
Valid characters:	alphanumeric

Uses different Modes
Used for transportation industry

Maxicode is a fixed-size matrix barcode which prints hexagonal dots arround a circled finder pattern with omnidirectional readability. This barcode is mostly used used by UPS for package tracking.

Syntax:

```
B[:name;]x,y,r,MAXICODE[+options];[ZIPCODE],[COUNTRY],[SERVICE],
. . . . . . . [TEXT] CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**MAXICODE**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.
+MODE	=	2,3,4,6 (see also next page)

text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Maxicode

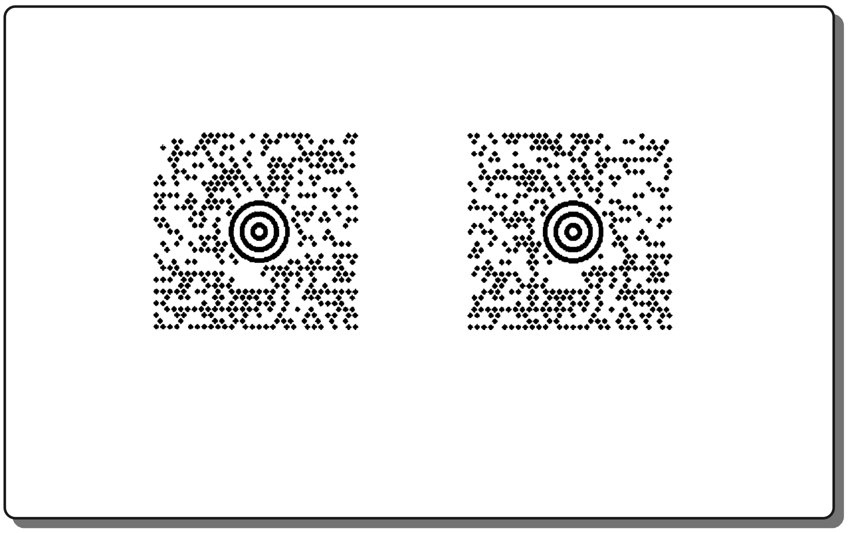
Following modes are available:

Mode 2 -	developed for the transport industry, Mode 2 encodes zip codes as numeric data. Usage in USA.
Mode 3 -	developed for the transport industry, Mode 3 encodes zip codes as alphanumeric data. Usage international
Mode 4 -	encodes text messages and has a fixed length of 93 characters
Mode 6 -	encodes also text messages of 93 characters. This mode is used for programming the barcode reader.

Example:

```
; UPS Maxicode certification labels
m m
J sample message 1
S l1;0,0,68,70,100
O R
B20,25,0,maxicode+mode2;[U:ANSI_TM]96841706672,840,024,1Z12345677
[U:GS]UPSN[U:GS]12345E[U:GS]100[U:GS][U:GS]1/2[U:GS]12[U:GS]N[U:GS]
123 MAIN ST B3F4[U:GS]SALT LAKE CITY[U:GS]UT[U:RS]
;sample message 2
B60,25,0,maxicode+mode2;[U:ANSI_TM]9684170,840,024,1Z12345677
[U:GS]UPSN[U:GS]12345E[U:GS]100[U:GS][U:GS]1/2[U:GS]12[U:GS]N[U:GS]
123 MAIN ST B3 F4[U:GS]SALT LAKE CITY[U:GS]UT[U:RS]
A 1
```

Please note that there is only a carriage return at the end of the barcode contents and not in the barcode expression. The barcode must be in one single line
Based on the length of the encoded information it was not possible to display this in another way.

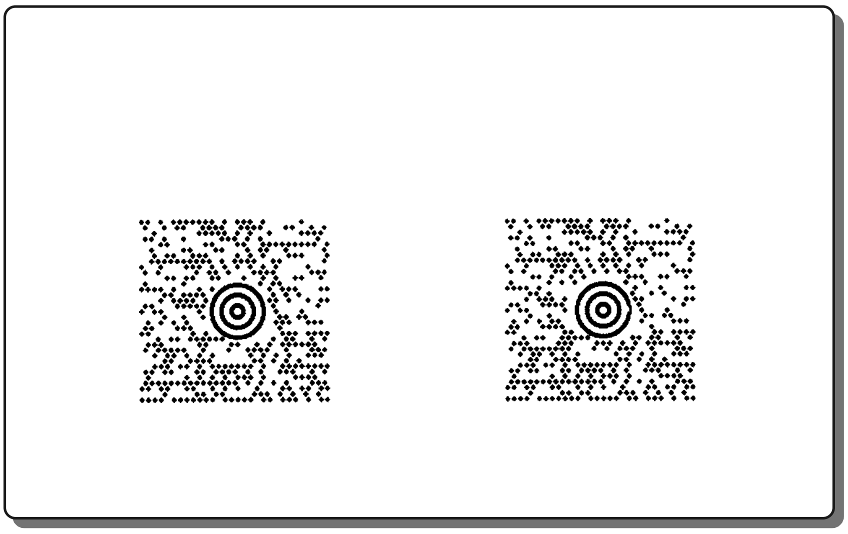


B - Barcode Maxicode

Example:

```
m m
J sample message 3
OR
H 20
S l1;0,0,68,70,100
B 15,14,0,maxicode+mode3;[U:ANSI_TM]96123ABC,222,024,1Z123
45677[U:GS]UPSN[U:GS]12345E[U:GS]100[U:GS][U:GS]1/
2[U:GS]12[U:GS]N[U:GS]123 MAIN ST B3 F4[U:GS]SALT LAKE
CITY[U:GS]UT[U:RS]
;sample message 4
B 65,14,0,maxicode+mode3;[U:ANSI_TM]9612AB,222,024,1Z12345
677[U:GS]UPSN[U:GS]12345E[U:GS]100[U:GS][U:GS]1/
2[U:GS]12[U:GS]N[U:GS]123 MAIN ST B3 F4[U:GS]SALT LAKE
CITY[U:GS]UT[U:RS]
A 1
```

Please note that there is only a carriage return at the end of the barcode contents and not in the barcode expression. The barcode must be in one single line
Based on the length of the encoded information it was not possible to display this in another way.



B - Barcode Maxicode

Example:

```
m m
J sample message 5
OR
H 20
S l1;0,0,68,70,100
B 20,14,0,maxicode+mode3;[U:ANSI_TM]96123ABCD,222,024
,Z12345677[U:GS]UPSN[U:GS]12345E[U:GS]100[U:GS][U:GS]1/
2[U:GS]12[U:GS]N[U:GS]123 MAIN ST B3F4[U:GS]SALT LAKE
CITY[U:GS]UT[U:RS]
;sample message 6
B 50,14,0,maxicode+mode2;[U:ANSI_TM]9612345678,840,024,1Z1234
5677[U:GS]UPSN[U:GS]12345E[U:GS]100[U:GS][U:GS]1/
2[U:GS]12[U:GS]N[U:GS]123 MAIN ST B3 F4[U:GS]SALT LAKE
CITY[U:GS]UT[U:RS]
A 1
```

Please note that there is only a carriage return at the end of the barcode contents and not in the barcode expression. The barcode must be in one single line
Based on the length of the encoded information it was not possible to display this in another way.

B - Barcode Micro PDF 417

Barcode type:	Micro PDF 417
Length:	2D - Code
Valid characters:	ASCII characters (more than 1000 bytes)

Micro PDF 417 is a multi-row symbology based on PDF 417 and designed for applications requiring a greater area efficiency but lower data capacity than PDF417.Micro PDF 417 has a fixed level of error correction.

Syntax:

```
B[:name;]x,y,r,Micro[+options],height,ne,ratio;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**Micro**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.
+COLSx	=	number of columns

height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

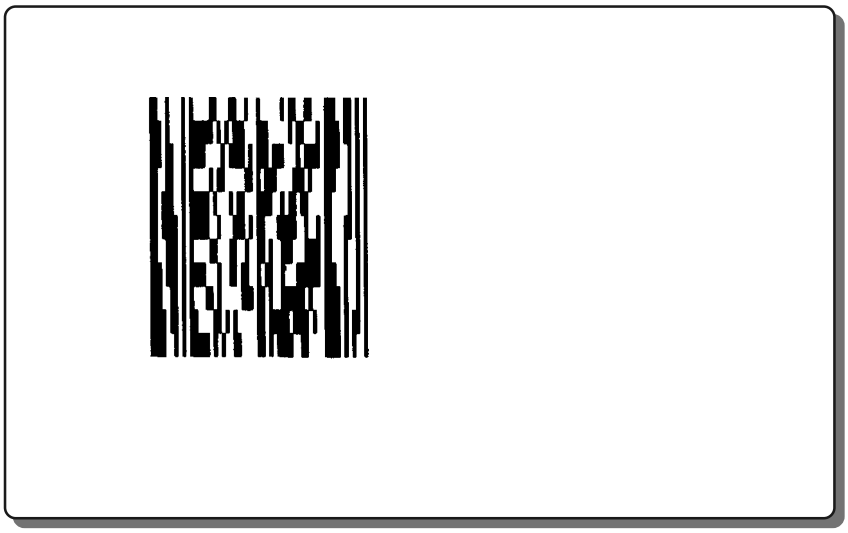
B - Barcode Micro PDF 417

MicroPDF417 provides for three encoding modes: Text Byte and Numeric compaction. Text is for general text Numeric for encoding data consisting only of digits and Byte to allow for the first 127 ASCII characters but with a reduced level of efficiency. Four symbol widths are permitted each specifying the number of data columns (1 – 4). Within each symbol width a variable number of rows provide for a maximum data capacity of:

Text compaction mode 0: 250 characters (2 data characters per codeword)
Byte compaction mode 1: 150 characters (1.2 data characters per codeword)
Numeric compaction mode 2: 366 characters (2.93 data characters per codeword)
The Level parameter for MicroPDF barcodes set the number of data columns within the barcode which may be 1 – 4.m m

Example:

```
J
S 0,0,68,71,100
B 10,10,0,Micro+COLS2,3,0.5;cab Produkttechnik
A 1
```



B - Barcode MSI (MSI Plessey)

Barcode type:	MSI (MSI Plessey)
Length:	variable
Valid characters:	numeric,
check digits:	yes (Mod 10)
ratio oriented:	yes

The MSI Plessey code is a numeric barcode with variable length and a modulo 10 check digit which is automatically added by the printer. Additional modulo check digits can be added to this code.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**MSI**)

[+options] Following options are available:

+MODxx	=	calculation of modulo check digit (**MOD10** and **MOD11**)
+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

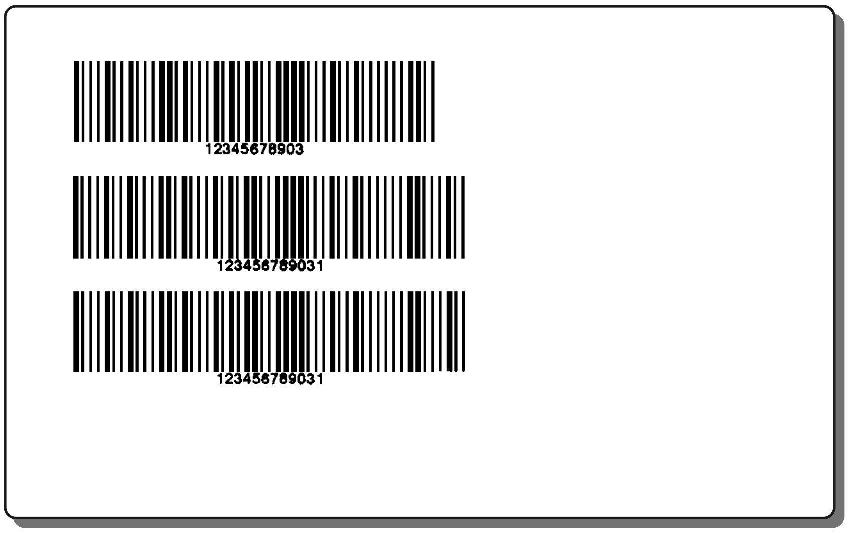
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode MSI (MSI Plessey)

Example:

```
m m
J
S l1;0,0,68,71,100
B 5, 5,0,MSI,12,0.3,2;1234567890
B 5,20,0,MSI+MOD10,12,0.3,2;1234567890
B 5,35,0,MSI+MOD11,12,0.3,2;1234567890
A 1
```



B - Barcode PDF 417

Barcode type:	PDF-417
Length:	2D - Barcode
Valid characters:	alphanumeric

PDF417 is a high-capacity two dimensional bar code. A PDF417 symbol can hold approximately 2000 characters of information.
The key characteristic of PDF417 is its large information capacity. This also explains its name. „PDF“ stands for Portable Data File. PDF417 is designed with enough capacity to contain an entire data file of information.

PDF417 is used today in a wide variety of applications, including logistics & transportation, retailing, healthcare, government, identification, and manufacturing
PDF417 uses error levels to ensure a good reading quality

Syntax:

```
B[:name;]x,y,r,PDF417[+options],height,ne,ratio;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**PDF417**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+ELx	=	Error Level (1-8)
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

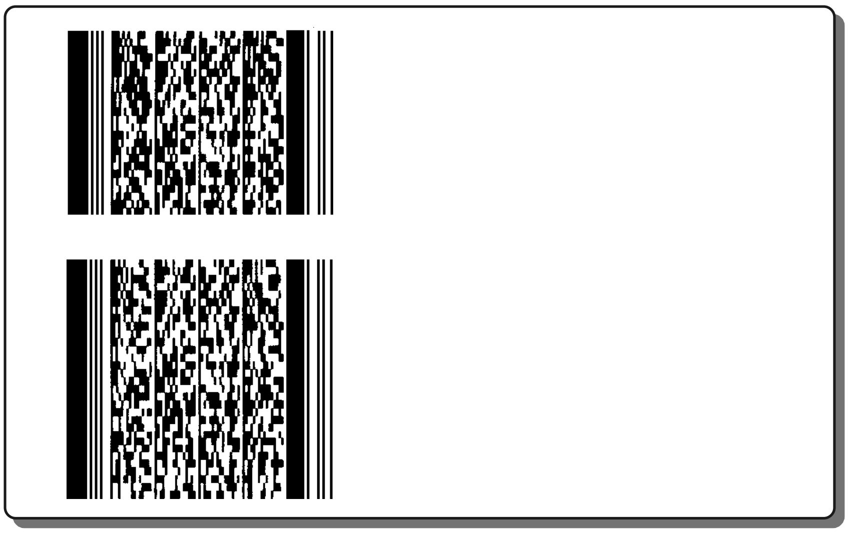
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode PDF 417

Example:

```
m m
J
S l1;0,0,68,71,100
B 2, 5,0,PDF417+EL0,0.1,0.38,1;cab Produkttechnik
GmbH[U:13][U:10]Wilhelm Schickard Strasse[U:13][U:10]D-76131
Karlsruhe
B 2,35,0,PDF417+EL3,0.1,0.38,1;cab Produkttechnik
GmbH[U:13][U:10]Wilhelm Schickard Strasse [U:13][U:10]D-76131
Karlsruhe
A 1
```



B - Barcode Plessey

Barcode type:	Plessey
Length:	variable
Valid characters:	A-F and 0-9
check digits:	no
ratio oriented:	yes

Plessey Barcode is a seldom used barcode which encoding possibilities are limited, as only numbers and 6 characters are encoded

Syntax:

```
B[:name;]x,y,r,PLESSEY[+options],height,ne,ratio;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**PLESSEY**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

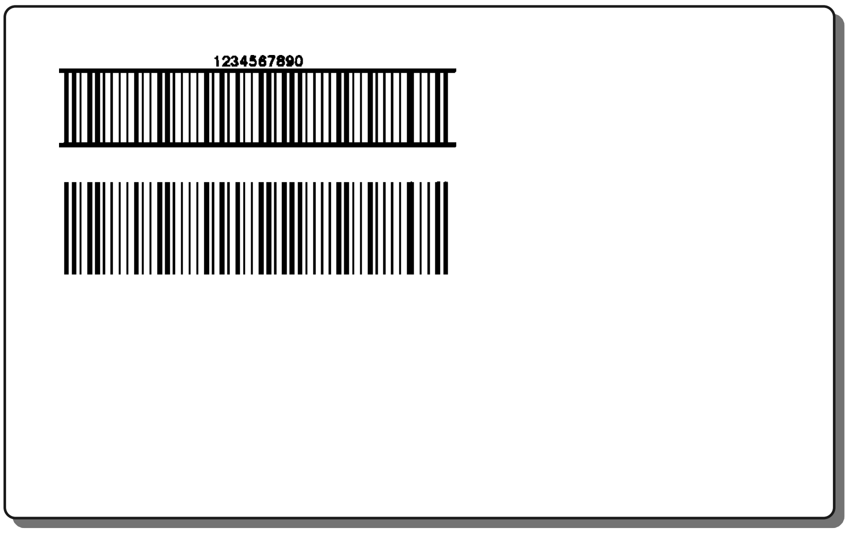
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Plessey

Example:

```
m m
J
S l1;0,0,68,71,100
B 5,20,0,PLESSEY+BARS,12,0.3,2;1234567890
B 5,35,0,plessey,12,0.3,2;1234567890
A 1
```



B - Barcode Postnet

Barcode type:	Postnet
Length:	variable - normally 9 characters
Valid characters:	numeric,
check digits:	no
ratio oriented:	no

Postnet is a barcode which is exclusively used in USA by the US Post Service.
It contains data to route letters to the correct location.

Syntax:

```
B[:name;]x,y,r,POSTNET[+options];text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**POSTNET**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

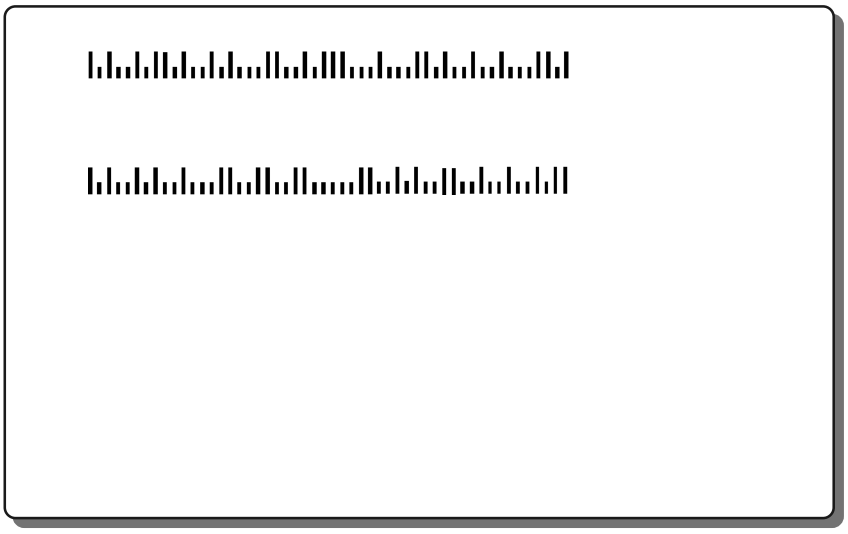
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode Postnet

Example:

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



B - Barcode PZN-Barcode *

Barcode type: PZN-Code (Special version of Code 39 (Code 3 of 9))

Length: 7 Digits
Valid characters: numerical
digits: 0-9,
check digits: no
ratio oriented: yes

PZN (Pharma-Zentral-Nummer) is a code for medicine identification in Germany. In Germany it's issued by the" Informationsstelle für Arzneispezialitäten GmbH", Frankfurt , Germany. The PZN is based on Code39 and has a fixed length of 7 digits. The last digit is a check digit. It uses the Code39-start sign „*" in combination with „-" as the start sign. The stop sign is the standard code39 stop sign „*". These start and stop signs and the characters „PZN „ do not need to be entered in order to produce a PZN because they are a fixed part of the PZN. The characters „PZN" are not coded in the barcode.

Syntax:

```
B[:name;]x,y,r,CODE39[+options],height,width,ratio;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**CODE39**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

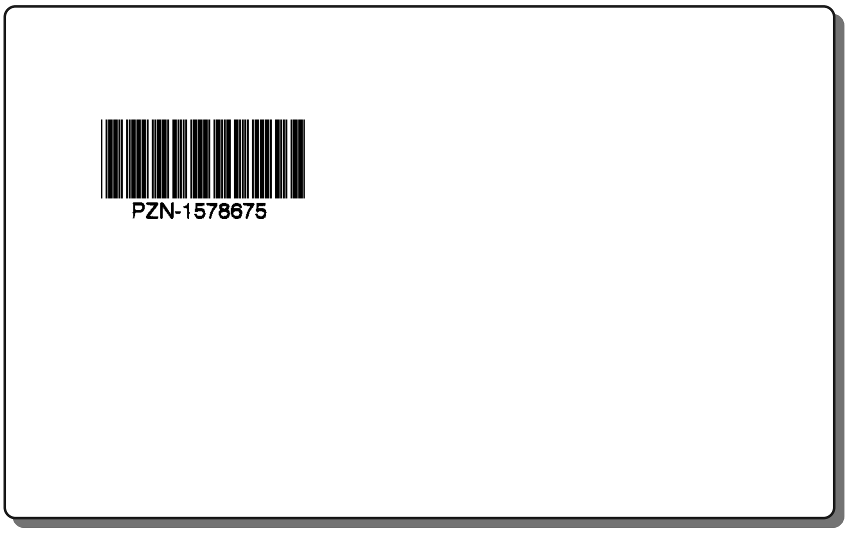
☞ ** PZN-Code is a special version of Code 39*
** It is highly reommended to contact the responsable organisation to get the complete description for this barcode. The responsable organisation may charge licenses for the usage of this code*

B - Barcode   PZN-Barcode *

Example:

```
m m
J
H 100,8
S l1;0,0,68,71,100
B 5,17,0,code39,10,0.2,3;-1578675
T 9,30,0,3,3;PZN-1578675
A 1
```

This example was printed without human readable characters. The human readable characters have been added in a separate text line to setup the text in a specific size.



B - Barcode QR-Code

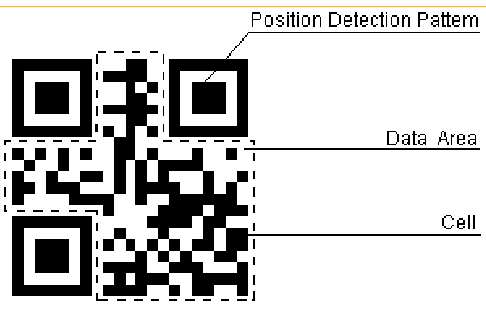
Barcode type: QR-Code

Length: 2DCode
Valid characters: alpha numeric

Omni-directional ultra-fast reading
error correction capability

QR (Quick Response) Code, is a matrix symbology consisting of an array of nominally square cells, allows omni-directional, high-speed reading of large amounts of data. Widely implemented in Japan, used in the automotive industry.

Three Position Detection Patterns in the symbol make omni-directional ultra fast reading possible.



Syntax:

```
B[:name;]x,y,r,QRCODE[+options],size;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**QRCODE**)

[+options] Following options are available:

+WSarea	=	white space area
+ELx	=	Error Level - valid values: 1-4,L,M,Q,H Default =1
+MODELx	=	valid input 1 and 2, Default value is 1
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

size	=	Barcode size
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode QR-Code

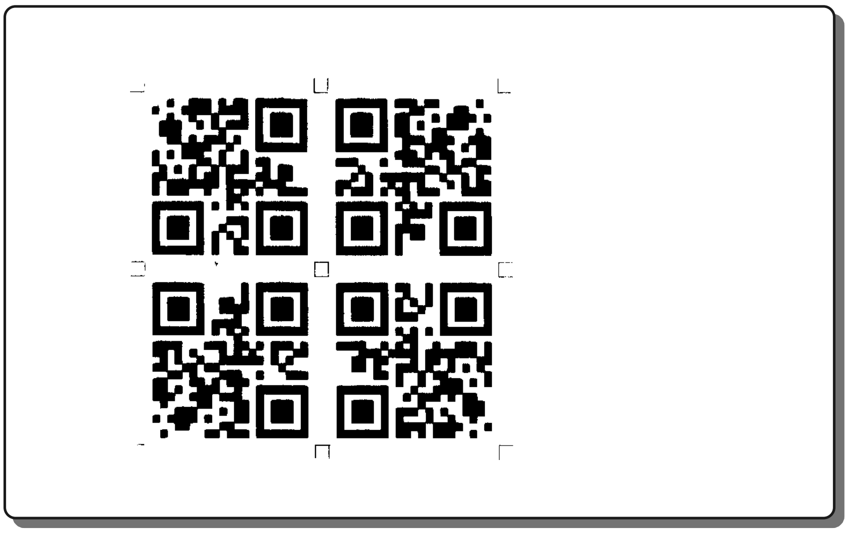
Dirty or damaged symbols can be read.
QR Code has error correction capability. Data can be restored even if a part of the symbol has become dirty or been damaged.

The QR Code is capable of handling numeric, alphanumeric, byte data as well as Japanese kanji and kana characters. Some thousend characters can be encoded using this symbol. Therefore, less space is required. The maximum characters depend on the character type (numeric, alphanumeric, kanji ..)

Please refer to the original specification of this barcode before using it.

Example:

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



B - Barcode RSS-14

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	14 digits
Valid characters:	numeric, digits: 0-9,
check digits:	yes
ratio oriented:	no

This compact linear symbol encodes a full 14-digit Global Trade Item Number and, optionally, a code indicating a link with a two-dimensional symbol carrying supplementary information.

It has the ability to encode up to 20 trillion values. There are actually 15 characters that make up the barcode, but only 14 characters are encoded.

Syntax:

```
B[:name;]x,y,r,RSS14[+options],height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

B - Barcode RSS-14

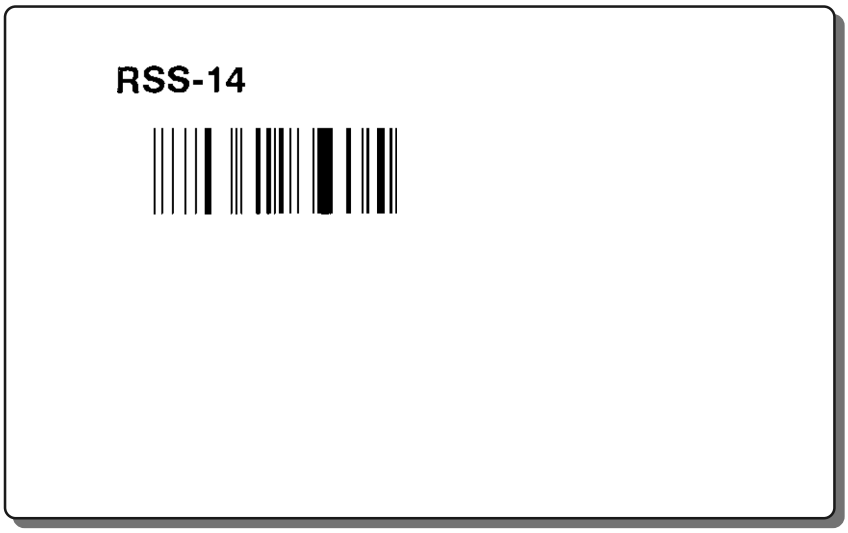
The first character is a linkage flag which determines if there is a Composite 2D barcode (see later on the next pages) associated with the bar code. This is the first character encoded and it should not be included in the DataToEncode property.

The control encodes either a „1“ (true) or „0“ (false) value as the first character in the barcode based on the property of the barcode control.

The next 14 characters in RSS14 are the 13 data characters plus an implied check digit. The check digit is not actually encoded in the barcode (as per the RSS standards), but should be included as part of the DataToEncode property.
If less than 14 characters are entered in the DataToEncode property, zeroes are padded to the front after the linkage flag. Non-numeric characters are stripped from the DataToEncode property.
There is an implied AI for standard for RSS-14 of 01 that should not be part of the DataToEncode.

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14
B 10,15,0,RSS14,10,.3;0441234567890
A 1
```



B - Barcode RSS-14 composite (CC-A)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	1D Code + 2D Code (Composite code)
Valid characters:	

RSS-14 composite (CC-A) uses a 1D component and a 2D component. For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

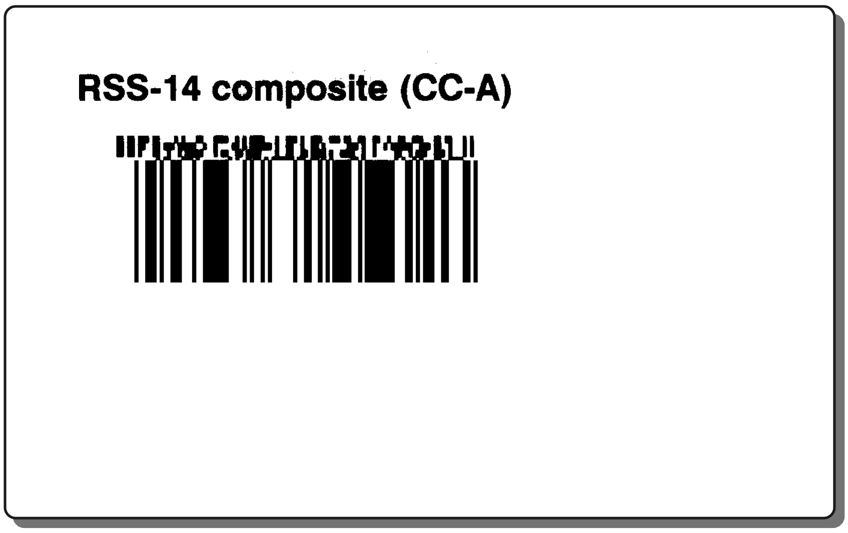
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode   RSS-14 composite (CC-A)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 composite (CC-A)
B 10,15,0,RSS14,16.5,.5;0361234567890[U:2D](11)990102
A 1
```



B - Barcode RSS-14 composite (CC-B)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	1DCode
Valid characters:	alpha numeric

RSS-14 composite (CC-B) uses a 1D component and a 2D component. For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

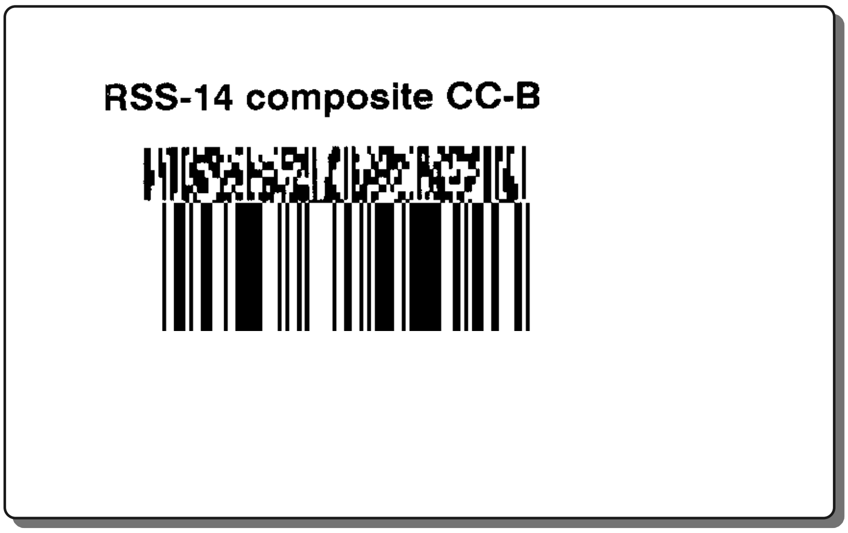
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS-14 composite (CC-B)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 composite CC-B
B 10,15,0,RSS14,16.5,.5;0361234567890[U:2D](21)abcdefghijklmnopqrst
A 1
```



B - Barcode RSS-14 truncated

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	14 digits
Valid characters:	numeric, digits: 0-9,
check digits:	yes
ratio oriented:	no Fixed height - 13 times the size of the module width

RSS-14 Truncated has the exact same data characteristics as the Standard RSS-14 barcode, except the bar height is set to the RSS standard of 13 times of the X dimension. It is possible to scan this symbology omni-directional.

Syntax:

```
B[:name;]x,y,r,RSS14+TRUNCATED,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	=	Barcode type (**RSS14+TRUNCATED**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

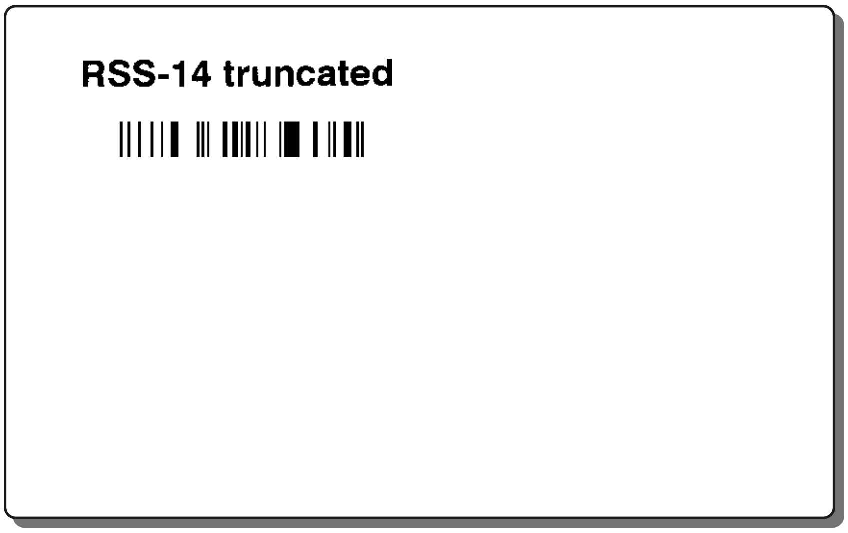
height	=	Barcode height
ne	=	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS-14 truncated

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 truncated
B 10,15,0,RSS14+TRUNCATED,4,.3;0441234567890
A 1
```



B - Barcode RSS-14 truncated composite (CC-A)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	1D Code + 2D Code (composite code) (The 2D component is based on Mirco PDF 417)
check digits:	yes
ratio oriented:	no Fixed height of the 1D code- 13 times the size of the module width.

RSS-14 Truncated has the exact same data characteristics as the Standard RSS-14 barcode, except the bar height is set to the RSS standard of 13 times of the X dimension. Additionally it is printed with a 2D compnent for additional information.

Syntax:

```
B[:name;]x,y,r,RSS14+TRUNCATED,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14+TRUNCATED**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

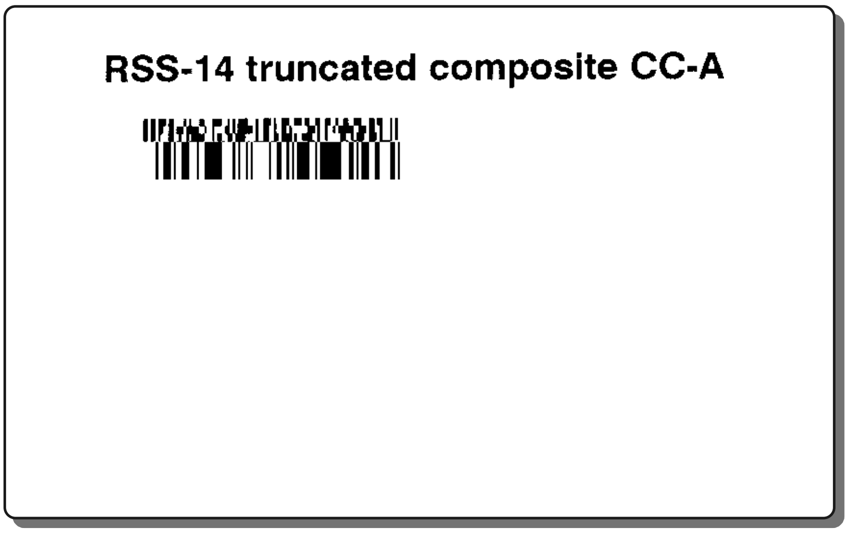
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS-14 truncated composite (CC-A)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 truncated composite CC-A
B10,15,0,RSS14+TRUNCATED+CC3,4,.3;0361234567890[U:2D](11)990102
A1
```



RSS-14 truncated composite CC-A

B - Barcode RSS-14 truncated composite (CC-B)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	1D Code + 2D Code (composite code) (The 2D component is based on Mirco PDF 417)
check digits:	yes
ratio oriented:	no Fixed height of the 1D code- 13 times the size of the module width.

RSS-14 Truncated has the exact same data characteristics as the Standard RSS-14 barcode, except the bar height is set to the RSS standard of 13 times of the X dimension. Additionally it is printed with a 2D compnent for additional information.

Syntax:

```
B[:name;]x,y,r,RSS14+TRUNCATED,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14+TRUNCATED**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

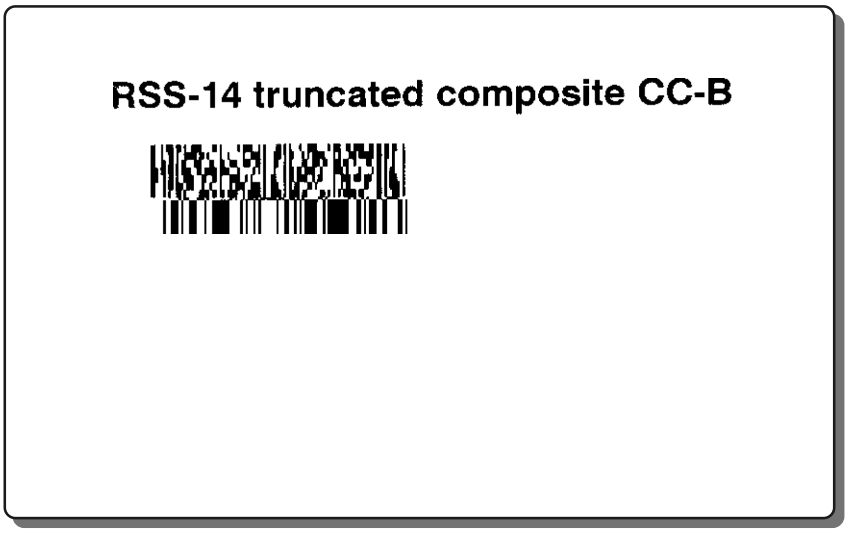
[U:2D] starts the description of the 2D component

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS-14 truncated composite (CC-B)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 truncated composite CC-B
B
10,15,0,RSS14+TRUNCATED+CC3,4,.3;0361234567890[U:2D](21)abcdefghijklmnopqrst
A 1
```



B - Barcode RSS-14 stacked

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	fixed - 14 digits
Valid characters:	numeric, digits: 0-9,
check digits:	yes
ratio oriented:	no Fixed height - 13 times the size of the module width

This version of the RSS symbology also encodes a 14-digit Global Trade Item Number. It is presented in two stacked segments. This feature enables making optimal use of space available. RSS-14 Stacked has two versions, a truncated version used for small item marking applications and a taller one which is designed to be read by omnidirectional scanners.

Syntax:

```
B[:name;]x,y,r,RSS14+STACKED,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14+STACKED**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

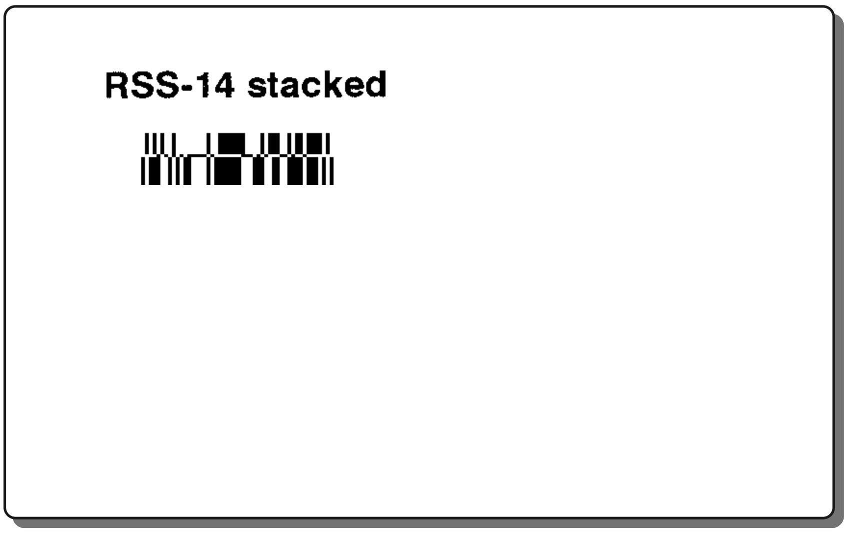
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS-14 stacked

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 stacked
B 10,15,0,RSS14+STACKED,12,0.5;0001234567890
A 1
```



B - Barcode RSS-14 stacked composite (CC-A)

Barcode type: RSS-Code (RSS= Reduced Space Symbology)

Length: Composite Code

Valid characters:

The RSS Stacked composite Barcode utilises an RSS Expanded stacked bar code symbol a linear component. For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14+STACKED,height,ne;text [U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14+STACKED**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

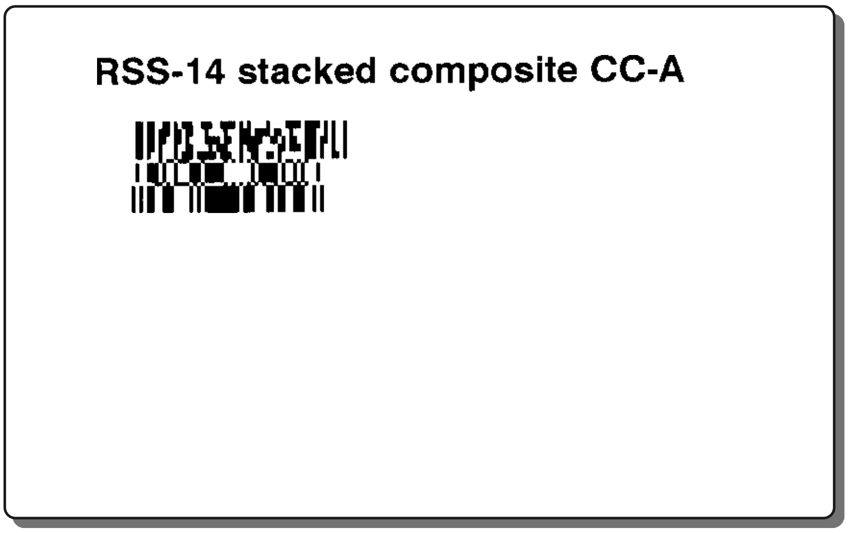
[U:2D] starts the description of the 2D component

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode   RSS-14 stacked composite (CC-A)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 stacked composite CC-A
B 10,15,0,RSS14+STACKED,12,0.5;0341234567890[U:2D](17)010200
A 1
```



RSS-14 stacked composite CC-A

B - Barcode RSS-14 stacked composite (CC-B)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite Code
Valid characters:	alpha numeric

For a detailed description of the RSS-14 stacked composite code please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14+STACKED,height,ne;text [U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14+STACKED**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

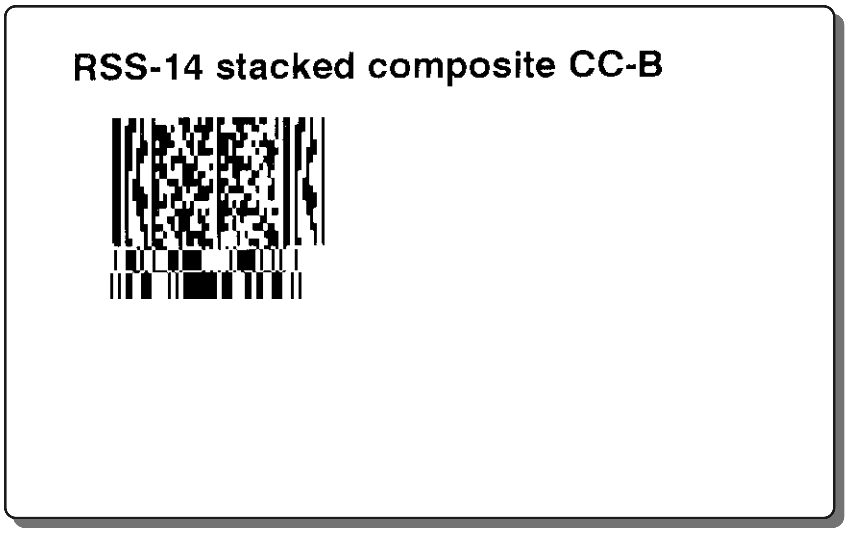
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS-14 stacked composite (CC-B)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 stacked composite CC-B
B
10,15,0,RSS14+STACKED,12,.5;0341234567890[U:2D](21)abcdefghijklmnopqrst
A 1
```



B - Barcode RSS-14 stacked omnidirectional

Barcode type: RSS-Code (RSS= Reduced Space Symbology)

Length: Composite code
Valid characters:

Omni-directional reading

RSS-14 is a composite barcode which has a omnidirectional readability. For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14+STACKEDOMNI,height,ne;textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14+STACKEDOMNI**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

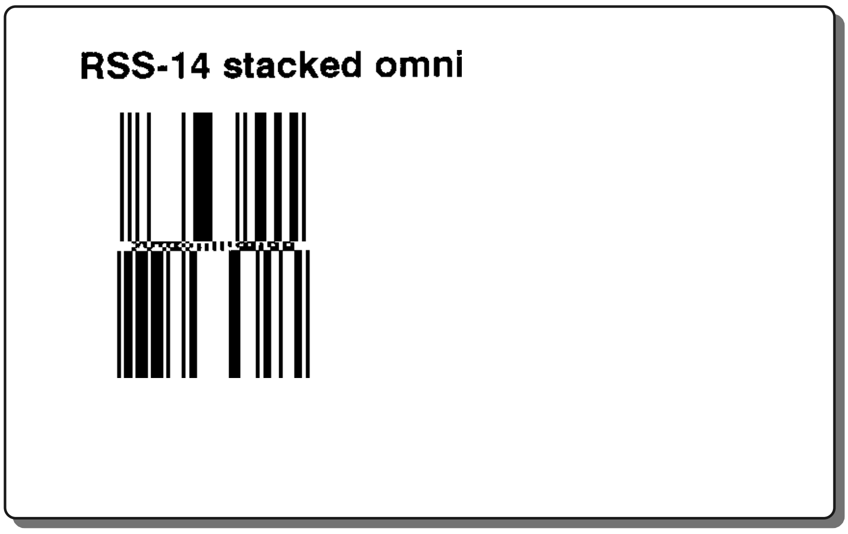
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS-14 stacked omnidirectional

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 stacked omni
B 10,15,0,RSS14+STACKEDOMNI,16.5,.5;0003456789012
A 1
```



B - Barcode RSS-14 stacked omnidirectional composite (CC-A)

Barcode type: RSS-Code (RSS= Reduced Space Symbology)

Length: Composite Code
Valid characters: alpha numeric

Omnidirectional readability

For a detailed description of the RSS-14 stacked omnidirctional composite code please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14+STACKEDOMNI,height,ne;text [U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	=	Barcode type (**RSS14+STACKEDOMNI**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

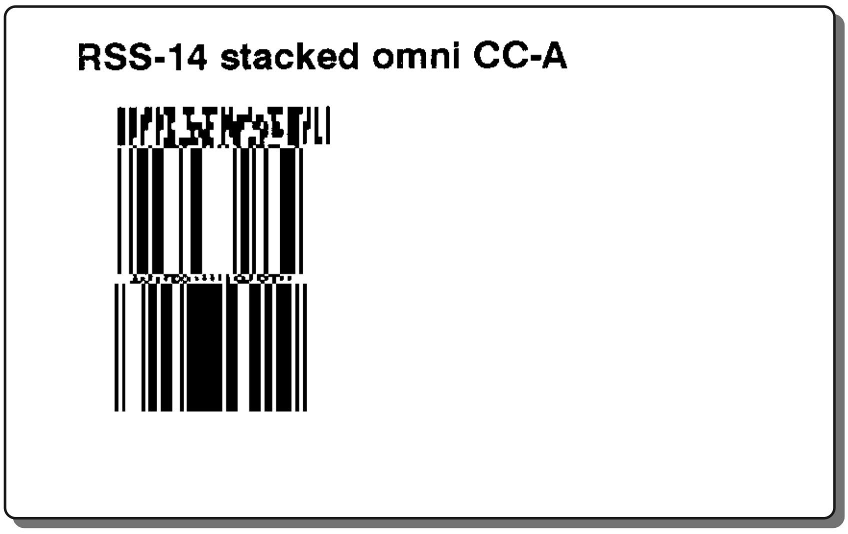
height	=	Barcode height
ne	=	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS-14 stacked omnidirectional composite (CC-A)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 stacked omni CC-A
B 10,15,0,RSS14+STACKEDOMNI,16.5,.5;0003456789012[U:2D](17)010200
A 1
```



B - Barcode RSS-14 stacked omnidirectional composite (CC-B)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite Code
Valid characters:	alpha numeric

Omni-directional ultra-fast reading
error correction capability

The RSS-14 stacked omnidirectional composite barcode has a omnidirectional readability. For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14+STACKEDOMNI,height,ne;text [U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14+STACKEDOMNI**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

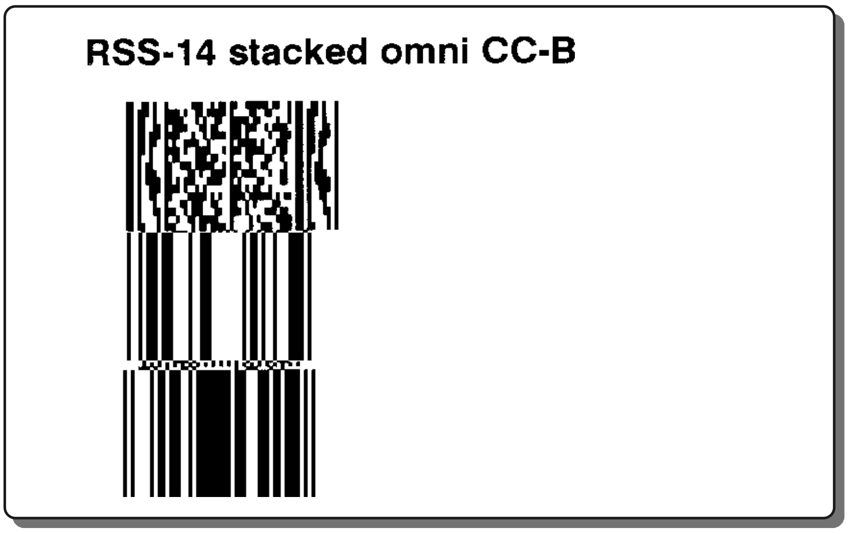
Detailed descriptions are at the beginning of the barcode chapter.

We highly recommend to read the original specifications of this barcode, before it is used !

B - Barcode RSS-14 stacked omnidirectional composite (CC-B)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS-14 stacked omni CC-B
B
10,15,0,RSS14+STACKEDOMNI,16.5,.5;0003456789012[U:2D](21)abcdefghijklmnopqrst
A 1
```



B - Barcode RSS limited

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	1DCode -14 digits max.
Valid characters:	alpha numeric

Note: No Omni-directional readability , no application identifier

For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14LIMITED,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14LIMITED**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS limited

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS limited
B 10,15,0,RSSLIMITED,5,.5;1501234567890
A 1
```

RSS limited

B - Barcode RSS limited composite (CC-A)

Barcode type: RSS-Code (RSS= Reduced Space Symbology)

Length: Composite code
Valid characters: numeric

For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSSLIMITED,height,ne;text [U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14LIMITED**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

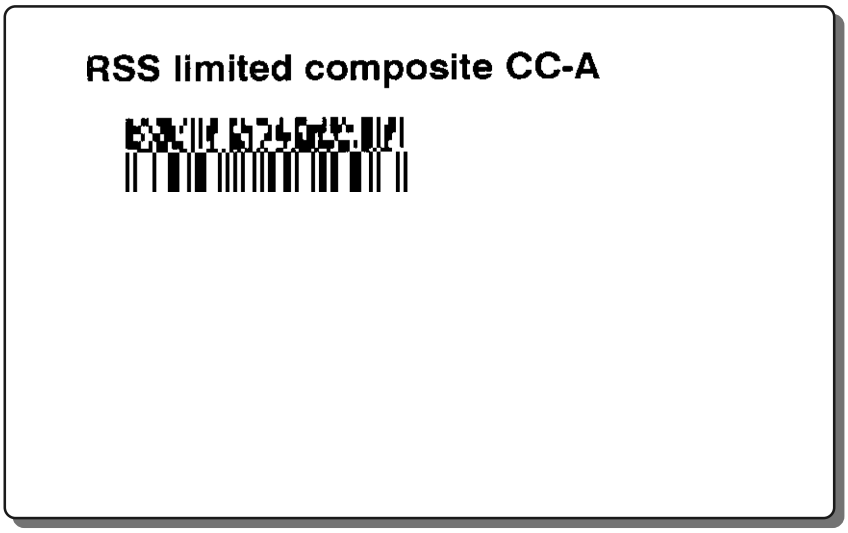
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS limited composite (CC-A)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS limited composite CC-A
B 10,15,0,RSSLIMITED,5,.5;0351234567890[U:2D](11)990102
A 1
```



B - Barcode RSS limited composite (CC-B)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite
Valid characters:	alpha numeric

For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSS14LIMITED,height,ne;text [U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSS14LIMITED**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

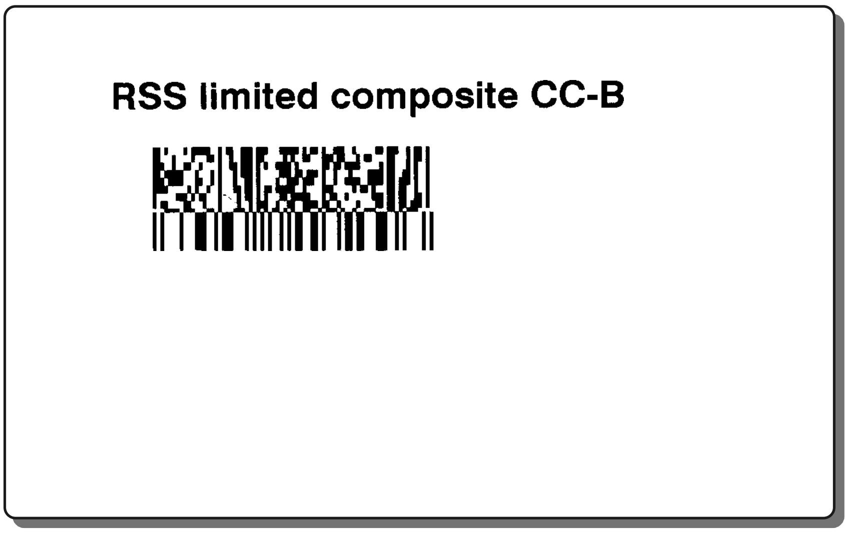
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS limited composite (CC-B)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS limited composite CC-B
B
10,15,0,RSSLIMITED,5,.5;0351234567890[U:2D](21)abcdefghijklmnopqrst
A 1
```


RSS limited composite CC-B

B - Barcode RSS expanded

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	1DCode
Valid characters:	alpha numeric

For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSSEXPANDED,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSSEXPANDED**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

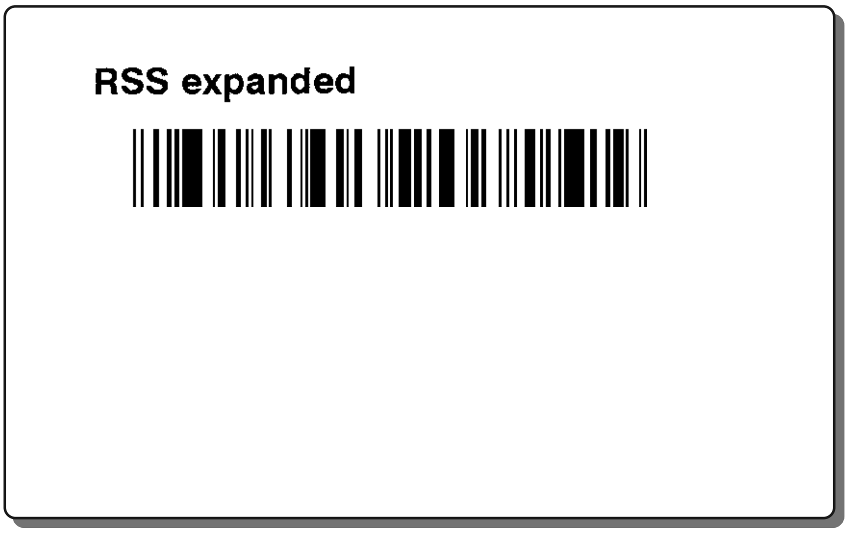
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS expanded

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS expanded
B10,15,0,RSSEXPANDED,10,.3;(01)98898765432106(3202)012345(15)991231
A 1
```



B - Barcode RSS expanded composite (CC-A)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite Code
Valid characters:	alpha numeric

For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSSEXPANDED,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSSEXPANDED**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

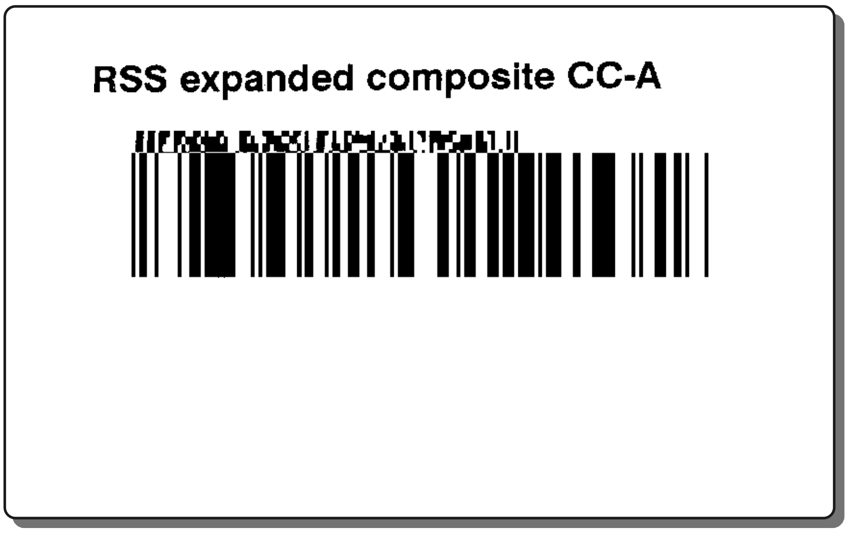
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS expanded composite (CC-A)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS expanded composite CC-A
B
10,15,0,RSSEXPANDED,16.5,.5;(01)93712345678904(3103)001234[U:2D](91)1A2B3C4D5E
A 1
```



B - Barcode RSS expanded composite CC-B

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite Code
Valid characters:	alpha numeric

For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSSEXPANDED,height,ne;text [U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSSEXPANDED**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

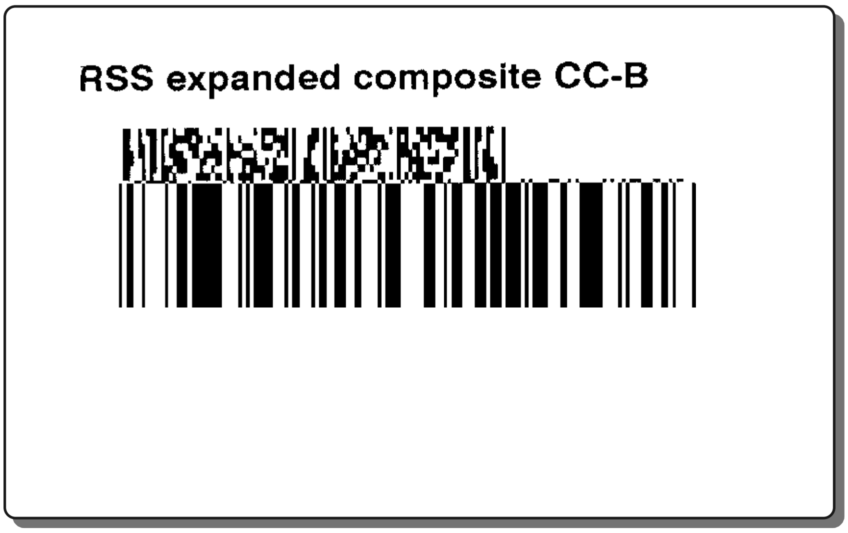
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS expanded composite CC-B

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS expanded composite CC-B
B
10,15,0,RSSEXPANDED,16.5,.5;(01)93712345678904(3103)001234[U:2D](21)abcdefghijklmnopqrst
A 1
```



B - Barcode RSS expanded stacked

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite Code
Valid characters:	numeric

For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSSEXPANDED+STACKED4,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSSEXPANDED+STACKED**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

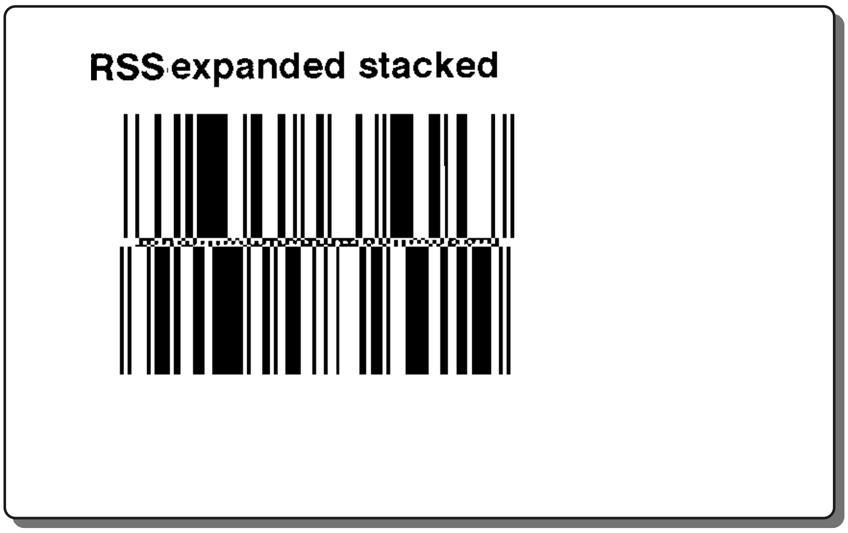
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS expanded stacked

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS expanded stacked
B10,15,0,RSSEXPANDED+STACKED4,16.5,.5;(01)98898765432106(3202)012345(15)991231
A 1
```



B - Barcode RSS expanded stacked half line

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite Code
Valid characters:	numeric

RSS expanded stacked half line is another code combination which used 1D and 2D components.
For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSSEXPANDED+STACKED4,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSSEXPANDED**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

We highly recommend to read the original specifications of this barcode, before it is used !

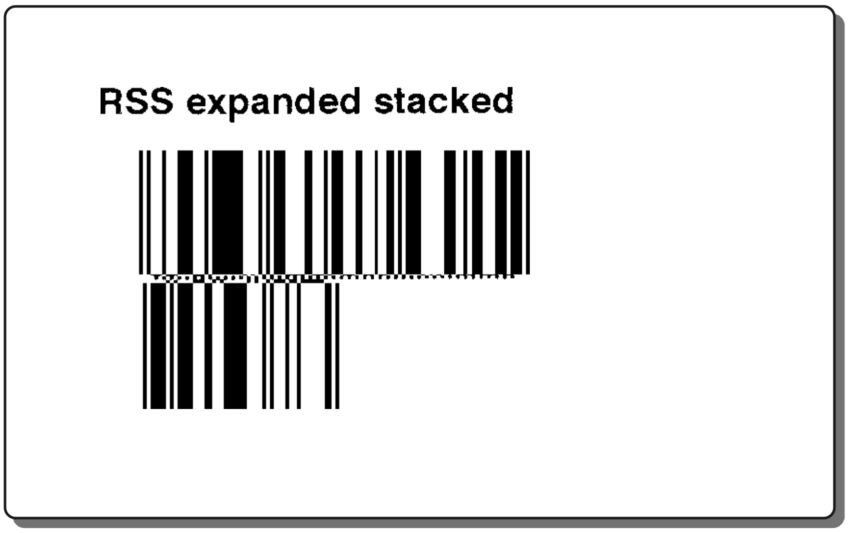
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode   RSS expanded stacked half line

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS expanded stacked
B 10,15,0,RSSEXPANDED+STACKED4,16.5,.5;(01)95012345678903(3103)000123
A 1
```



B - Barcode RSS expanded stacked composite (CC-A)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite Code
Valid characters:	alphanumeric

The RSS expanded stacked composite code is a mixture of 1D and 2D barcodes which can contain numeric and alphanumeric components. For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSSEXPANDED+STACKED4,height,ne;text[U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSSEXPANDED**)

[+options] <u>Following options are available:</u>

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

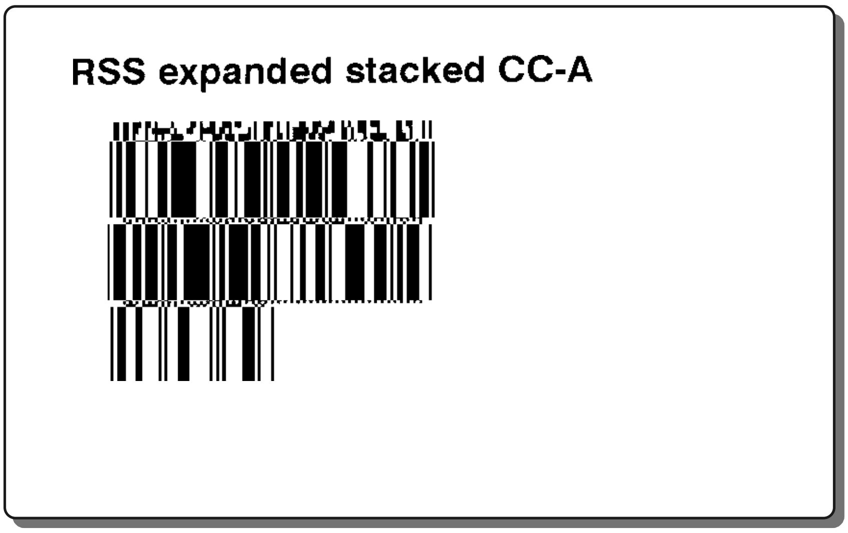
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode RSS expanded stacked composite (CC-A)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS expanded stacked CC-A
B10,15,0,RSSEXPANDED+STACKED4,10,.4;(01)00012345678905(10)ABCDEF[U:2D](21)12345678
A 1
```



B - Barcode RSS expanded stacked composite (CC-B)

Barcode type:	RSS-Code (RSS= Reduced Space Symbology)
Length:	Composite Code
Valid characters:	alpha numeric

The RSS expanded stacked composite code is a mixture of 1D and 2D barcodes which can contain numeric and alphanumeric components. For a detailed description please refer to the original description of this code - available at your local UCC / EAN organisation.

Syntax:

```
B[:name;]x,y,r,RSSEXPANDED+STACKED4,height,ne;text[U:2D] textCR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**RSSEXPANDED+STACKED4**)

[+options] Following options are available:

+WSarea	=	white space area
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

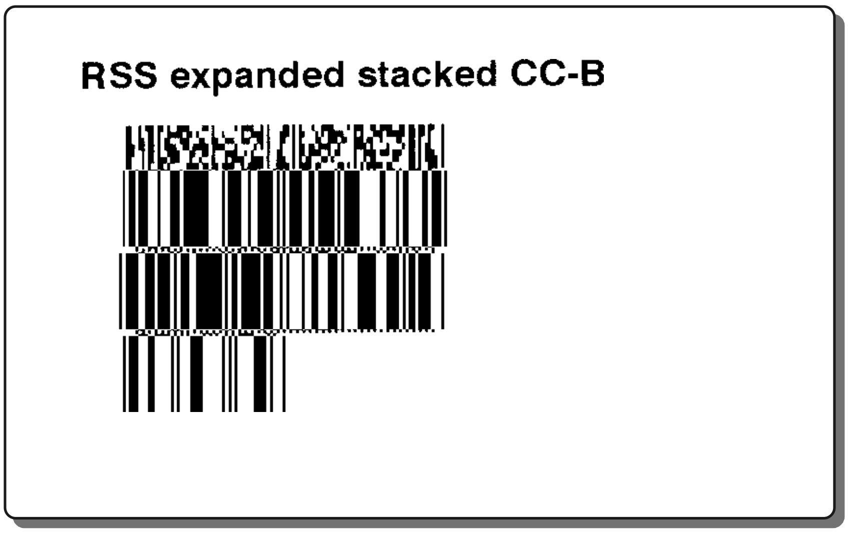
☞ *We highly recommend to read the original specifications of this barcode, before it is used !*

B - Barcode RSS expanded stacked composite (CC-B)

Example:

```
m m
J
S l1;0,0,68,71,104
T 5,10,0,5,5;RSS expanded stacked CC-B
B 10,15,0,RSSEXPANDED+STACKED4,10,.4;(01)00012345678905(10)
ABCDEF[U:2D](21)abcdefghijklmnopqrst
A 1
```

Please note: There is no carriage return in the barcode line. The barcode data must be in one line.



B - Barcode UPC-A

Barcode type:	UPC-A
Length:	fixed - 12 digits
Valid characters:	numeric only
	digits: 0-9,
check digits:	yes (Mod 10)
ratio oriented:	no

UPC-A is a retail barcode with a fixed length of 12 digits. The 12th digit is a modulo 10 check digit. cab printers require only 11 digits. The 12th digit is calculated by the printer.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**UPCA**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+XHRI	**=**	(Extended Human Readable Interpretation
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.
+NOCHECK	=	Check digit suppression when start no is 20...29

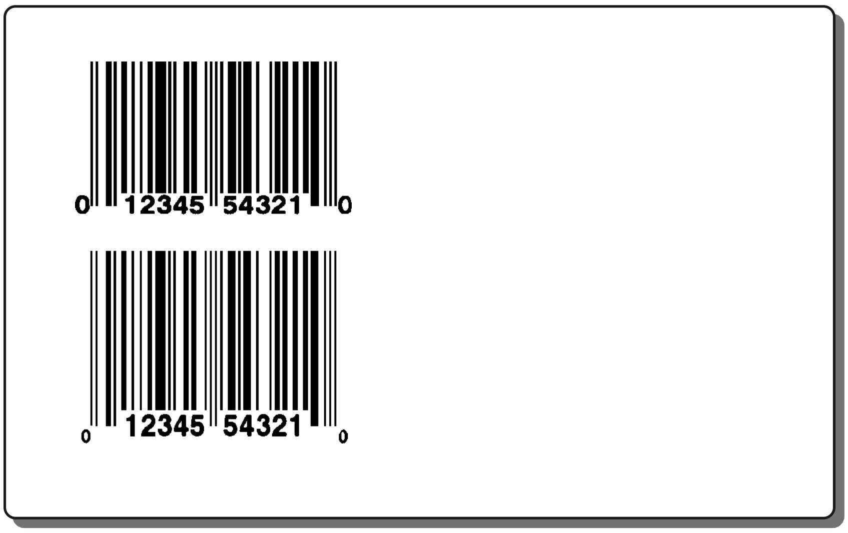
size	=	Standard Codesize **SCx** (instead of height and ne)
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode UPC-A

Example:

```
m m
J
O R
S l1;0,0,68,71,100
B 10,5,0,UPC-A,20,0.35;01234554321
B 10,30,0,UPCA+XHRI,SC1;01234554321
A 1
```



B - Barcode UPC-E

Barcode type:	UPC-E
Length:	fixed - 8 digits
Valid characters:	numeric, digits: 0-9,
check digits:	yes (Mod 10)
ratio oriented:	no

UPC-E is a retail barcode with a fixed length of 8 digits.The 8th digit is a modulo 10 check digit. cab printers require only 7 digits.The 8th digit is calculated by the printer.

Syntax:

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

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**UPCE**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+XHRI	**=**	(Extended Human Readable Interpretation
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

size	=	Standard Codesize **SCx** (instead of height and ne)
height	=	Barcode height
ne	**=**	Narrow element
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode UPC-E

Example:

```
m m
J
S l1;0,0,68,71,100
B 10, 5,0,UPC-E,20,0.35;0123456
B 10,30,0,UPCE+XHRI,SC1;0123456
A 1
```

0 123456 5
0 123456 5

B - Barcode UPC-E0

Barcode type:	UPC-E0
Length:	fixed - 8 characters *
Valid characters:	numeric
check digits:	yes (Mod 16)
ratio oriented:	yes

UPC-E0 is a numerical barcode with 8 characters. The 8th character is the check digit. The check digit is calculated automatically by the printer.
Invalid characters are converted into zeroes.
* A zero suppression converts the barcode into a more compact version. This offers the possibility to key in up to 12 characters which are compressed into 6 characters by the printer. Inthis case the first character must be zero !!
Detailed information is available by the UCC, Inc (Uniform Code Council, Inc.)

Syntax:

```
B[:Name;]x,y,r,UPCE0,height,ne;text CR
```

B - Barcode field definition

[:name;]	=	field name
x	=	x - coordinate
y	=	y - coordinate
r	=	Rotation 0, 90, 180 and 270 degrees
type	**=**	Barcode type (**UPCE0**)

[+options] Following options are available:

+WSarea	=	white space area
+BARS	=	Prints boundary lines above and below the barcode.
+UPBAR	=	Prints a boundary line above the barcode
+DOWNBAR	=	Prints a boundary line below the barcode
+VERIFYn	=	Verify the barcode data. (optional barcode reader required)
+GOODBADn	=	Same function as +VERIFYn without checking the content.

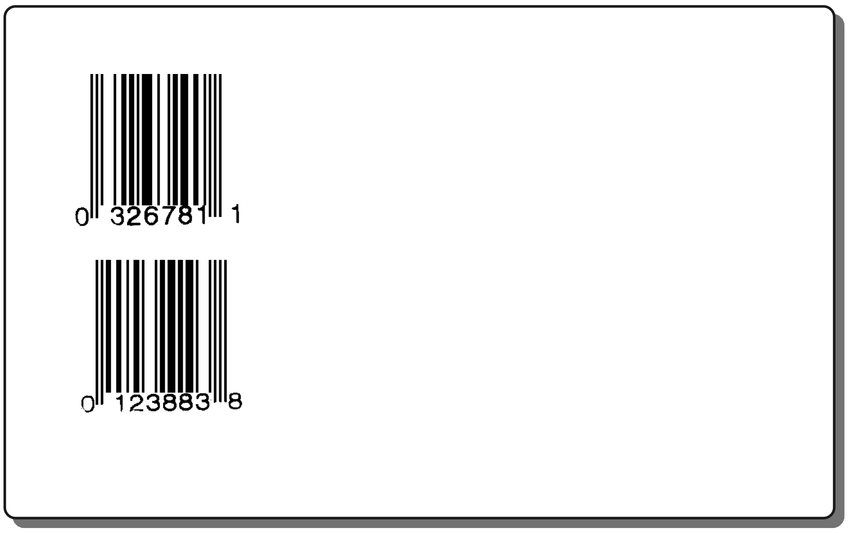
height	=	Barcode height
ne	**=**	Narrow element
ratio	**=**	Ratio between narrow and wide bars.
text	=	Barcode data

Detailed descriptions are at the beginning of the barcode chapter.

B - Barcode UPC-E0

Example:

```
m m
J
S l1;0,0,68,71,100
B 10, 5,0,UPCE0,20,0.35;03210000678
B 10,30,0,UPCE0,SC1;01230000088
A 1
```



C - Cutter Parameters

The C command is used to set the parameters for the optional cutter or perforation cutter. The cutting command uses the label counter to cut after a specified amount of printed labels or can be set to cut at the job end. Additonally it is possible to perform a second cut (double-cut) in one label.
Furthermore an optional perforation cutter is available, which can perforate and which is also able to do a "regular" cut.

Syntax:

```
C x[,disp1[,disp2]] CR
```

C - cutting command				
x =	cutting method - valid parameters are:			
	amount	=	amount of labels after which a cut is processed. Possible values 1-9999	
	e	=	cutting at the job end. Cuts once at the job end which is defined by the "A" (amount) command.	
	s	=	cut at print start (before the first label. This command is only executed once in the job and can be combined with " C amount ". disp1 is an optional offset in the chosen unit.	
	p	=	perforate - requires the optional perforation cutter !	
	sp	=	perforate at the start of the printjob (requires the optional perforation cutter !, and can be combined with " C amount ". disp1 is an optional offset in the chosen unit.	
disp1 =	(displacement 1) - offset to the end of the defined label			
disp2 =	(displacement 2) - offset to the first cutting position. (always positive values !)This double cut option offers the possibility to cut off portions of a label. [disp2] is not available when the „cut before first label (s) parameter is used. disp2 is only available for regular cuts and **not for perforations !**			

Please see also the "O" command to adjust the cutting time (cutting depth) for the perforation cutter. All measurements in millimeters or in inches (see the „m“ command)

Optional cutter required

C - Cutter Parameters

Important ! This command must be placed after the label size is defined !! (S - command)
This command requires the optional cutter or perforation cutter.
It depends on your printer type if a cutter or perforation cutter is available.
Applicator models (Hermes,PX module...) do not support the cutter functionality.

The offset value must be always smaller than the label height.
The cutting commands allow some senseless combinations, especially when a perforation cutter is used,- there are no limitations. i.e. using the perforation command together with the cut command " C 1" would always cut after one label and no perforation could be recognized.

The offset value must be always smaller than the label height.

C - Cutter Parameters

Example:

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

Prints 10 labels and cuts always after the second label

„**Double cut**“ possibility: The following example cuts 5 labels and performs a second cut after 2 mm.

Example:

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

Using the Cutter command „C“ together with Replace commands „R“ offers additional possibilities. (See also „Replace Field Command“)
The next sample shows the usage of the cutter together with the "Replace" command.

Example:

```
m m
J
S l1;0,0,68,71,100
T:Var1;12,25,0,3,9;cut after 5 labels
C 5
A 100
R Var1;cut after 2 labels
C 2
A 60
```

cuts the first print job of 100 labels after each 5th and in the second job with a total amount of 60 labels every 2. label will be cut.

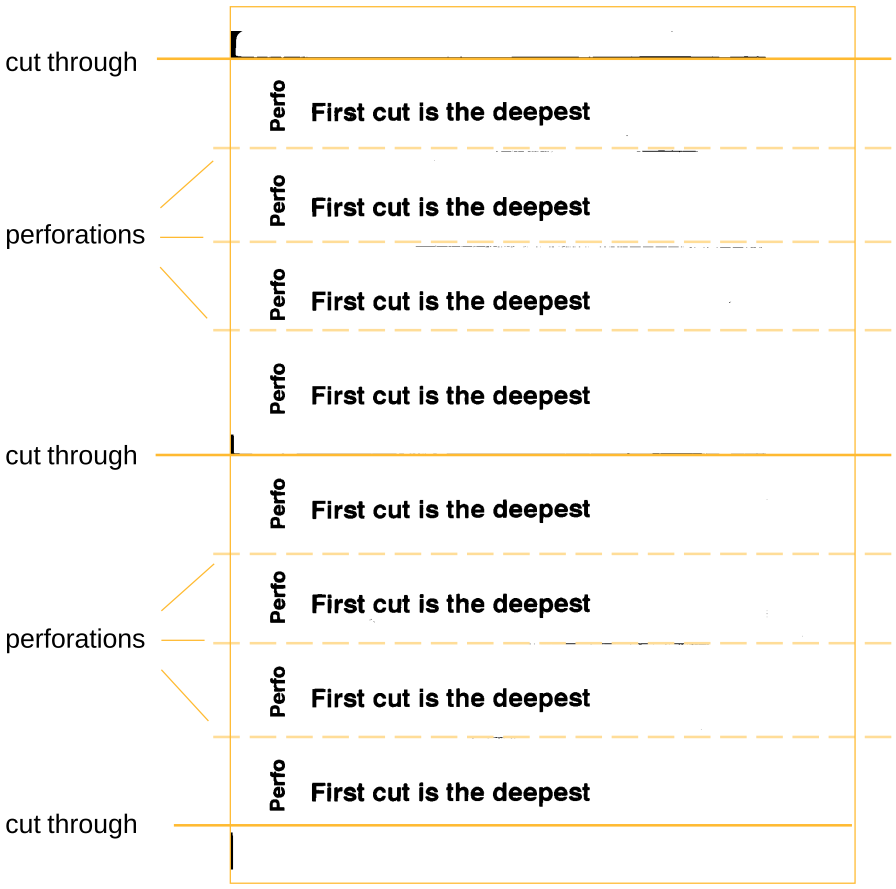
C - Cutter Parameters

The following sample requires the optional **Perforation Cutter**.

Example:

```
m m
J
O R
S e;0,0,18,18,100
T 10,14,90,5,4;Perfo
T 15,12,0,5,5;First cut is the deepest
C s
C 4
C p
A 12
```

This example cuts at the print start (C s), does a perforation cut after each label (C p) and cuts the material completely after each 4th label (C,4,0).
All together 12 labels will be produced. (A 12) - the picture blow shows just 8 of them...
The label was defined 18 mm high on continuous material.



D - Global Object Offset

The D command is used to move the complete label content to the specified location. All objects positions are influenced by this command. The starting point for the label contents is shifted by this values.
The usage of this command is normally if new label stock is used which is not identical to the label stock which was used up to now.This might be that the side margin of the liner is wider or smaller than before. The minimum and maximum values depend on the printer type (printhead width and label length). All measurements in millimeters or in inches (see the „m“ command)

Syntax:

```
D x,y CR
```

D - Displacement

x	= offset value in horizontal direction
y	= offset value in vertical direction All measurements in millimeters or in inches (see the „m“ command)

Example:

```
m m
J
D 30,20
S l1;0,0,68,71,100
T 12,25,0,3,7;Displacement
A3
```

Displacement

E DBF ... - Define Files (Extension DBF)

E DBF defines a DBase IV compatible database file which will be used in the label.

Syntax:

```
E DBF;name CR
```

E - Define Extension		
DBF	=	Define Database File(.dbf) (*) - tells the printer the database name for further operations. Used together with the **[DBF]** text option,later described in this manual.
name	=	File name

Example:

```
E DBF;article
```

Uses ARTICLE.DBF as external file on memory card. ARTICLE.DBF must be present on the printer´s memory card to get access.

☞ *(*) Filenames have to be in the 8.3 format (8 characters name and 3 characters extension no special characters allowed). Please note, that DBase does not support Unicode characters !*
(i.e. chines characters are not supported by Dbase)
Using the DBase functionality is ideal for smaller databases. For big databases and high data volume it is recommended to use the optional cab database connector as the access for the files might be to slow. (The funcionality of database connector is described later in this manual).

The dBase file supports only <u>TEXT</u> /<u>STRING</u> fields, which means you may need to change the field types from anything else i.e.numeric or memo etc... into TEXT format..

E LOG ... - Define Files (Extension LOG)

E LOG... defines the name of a external protocol file (LOG file).

Syntax:

```
E LOG;name CR
```

E - Define Extension		
LOG	=	define file name for the .LOG file
name	=	File name without the extension ".LOG" !

Example:

```
E LOG;PROTOCOL
```

Defines the log file PROTOCOL.LOG for use on printer´s optional memory card (or internal memory). Used together with the **[RLOG]** and **[WLOG]** text options.

☞ *Filenames have to be in the 8.3 format (8 characters name and 3 characters extension)*

The E LOG command cannot be used with the internal flash file system (IFFS).

E TMP ... - Define TMP File (Extension TMP)

E TMP... defines the name of an external temporary file (TMP file). TMP files can be used e.g. for serial numbering where the incremented or decremented value is saved in the printer. This value can be the starting value for the next label.

Syntax:

```
E TMP;name_type CR
```

E - Define Extension		
TMP	=	Define filetype .TMP
name	=	File name without the extension ".TMP" !

Example:

```
E TMP;SERNUM
```

Uses SERNUM.TMP as file for serial numbering from memorycard. Used together with the **[RLOG]** and **[WLOG]** text options.

☞ *Filenames have to be in the 8.3 format (8 characters name and 3 characters extension, no special characters allowed)*

The E TMP command cannot be used with the internal flash file system (IFFS).

E SQL - Define Files (Extension SQL)

E SQL tells the printer the IP - address of an external database server.

Syntax:

```
E SQL;IPaddress:portaddress CR
```

E - Define Extension		
SQL	=	Defines the address of a database server Used together with database connector features.
IPaddress	=	IP-address of the external database server
portaddress	=	port address of the external database server

Important notes: The usage of the SQL function requires that the printer is connected with its network interface. The usage on printers which are equipped with the X2 board (A+ series, Mach 4 etc. need a software legitimation to use the E SQL possibilities.)

The usage of this commands requires optional components.
The DBF, TMP and LOG functions require an optional Compact Flash memory card .

(*) Filenames have to be in the 8.3 format (8 characters name and 3 characters extension) Please note, that dbase does not support Unicode characters
Using the DBase functionality is ideal for smaller dataabases. For big databses and high data volume it is recommended to use the optional cab database connector. (Later described in this manual)

The E SQL command cannot be used with the internal flash file system (IFFS).

E RFID - Define Files (Extension RFID)

Define parameters for RFID tag. (Requires that the cab RFID unit is installed)

Syntax:

```
E RFID;T:tagtype[,R:Retries][,C:cp][,P:pos][E:power]  CR
```

E - Define Extension

tagtype	=	**Auto** (detects Tagtype automatically) - (get system info) Auto is default value. **ISO 15693** ISO 15693 tags, fixed block size 32 bits
retries	=	**0-10** Amount of retries to read or write a tag if internal errors occur. (default value is 0)
cp	=	codepage for data conversion: **Auto** = codepage from the setup **name**= name of the codepage (must be identical to the codepage names in the setup.
pos	=	**-10 ... +20** Reading position relatively to the printhead. (default value is 0)
power	=	field strength (default is the value from the setup) **S** = normal **H** = high

Example:

```
E RFID;T:ISO 15693,R:2,cp:Auto,pos:-3,E:H
```

☞ *This command is not available on printers with separate RFID interface. (A+ series)*

F - Font Number

The F command assigns an alternate number to a font name. The reason for this command is to simplify the font handling, keeping a better overview on the used fonts in a label and enables the programmer to exchange a font in a label very easy.
The resident fonts in the cab printers have fixed names, but they can be redefined with this command. Once the font number is defined it is valid for the complete label.

Syntax:

```
F number;name CR
```

Assigns the number to a name

F - Font command		
number	=	New font number.
name	=	Fontname which will be replaced by „number“.

On TrueType fonts, the number found in the typeface file is used as the default.

Example:

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

Uses TrueType™ names

Example:

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

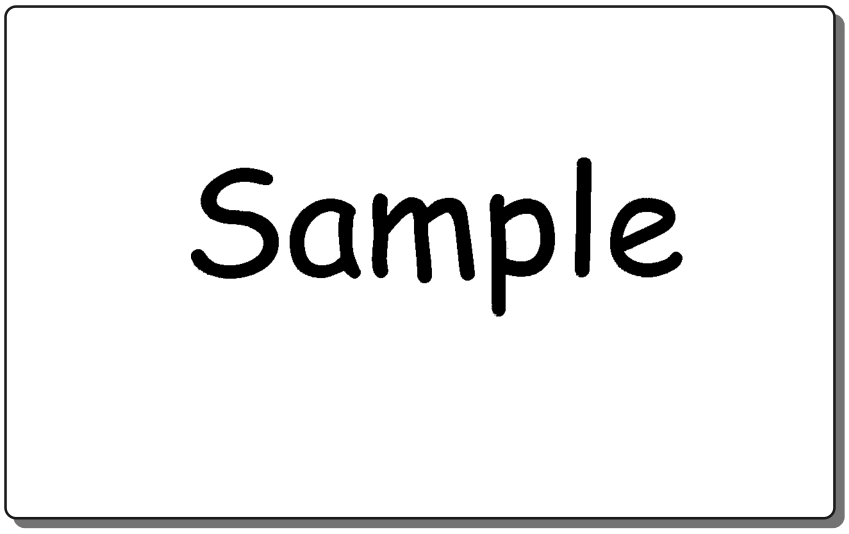
Assigns the alternate number 40 to the printer´s resident Swiss™ 721 Bold Italic font.

F - Font Number

Example:

```
M l fnt;Comix
m m
J
H 66
S l1;0,0,68,71,100
F 10;Comix
T 0,35,0,10,20;Sample[J:c100]
A 1
```

The example above assigns font number 10 to the previously downloaded font Comix.



G - Graphic Field Definition

cab printers are able to print graphic elements, such as lines, rectangles, circles and elipses. These graphic elements are defined by the G command.

Syntax:

```
G[:name;]x,y,r;ge:settings[,options] CR
```

G - Graphic field definition command.		
[:name;]	=	Optional field name, for further usage as a variable. . Maximum length 10 characters, no special characters allowed, fieldname must be unique. The field name can be used for further operations, such as Replace field name . (See the „R“ command for details) or just as a comment.
x	=	Horizontal coordinate of the start position in millimeters or inches from the left edge of the printable area to the start position of the graphic field.
y	=	Vertical coordinate of the start position in millimeters or inches from the top edge of the printable area to the start position of the graphic field.
		Starting points of the graphic elements are: Lines: Center of the starting point of the line Rectangles: upper left corner, outside of the rectangle Circles: Center Ellipses: Center
r	=	Rotation. Graphic elements can be rotated in steps of 1 degrees from 0 to 359 degrees.
ge	=	graphic element: Here we define the type of the graphic element which shall be printed. **C** = Circle (Ellipse is defined with the circle command) **L** = Line **R** = Rectangle

G - Graphic Field Definition

settings		=	specific graphic element settings, depending on the selected graphic element.
[,options]=	**,fill**	=	filling of the graphic object with a specified pattern or with dot density. (see graphic option „fill“)
	,shade	=	shading option (gradient filling - see graphic option „shade“)
	,outline	=	outline option - prints an outline around the filled graphic object with the thickness of 1 dot. (see graphic option „outline“)

Details about the settings for each graphic element are shown on the next pages.

G - Graphic Definition - Circle

Graphic Type: C - Circle, Ellipse

Syntax:

```
G[:name;]x,y,r;C:radius1[,radius2[,width]][,options] CR
```

G = Graphic field definition command.

[:name;]	= Optional field name. Maximum length 10 characters, no special characters allowed, field name must be unique. The field name can be used for further operations, such as Replace field name (See the „R“ command for details) or just as a comment.
x	= Horizontal coordinate of the start position in millimeters or inches from the left edge of the printable area to the center of the circle.
y	= Vertical coordinate of the start position in millimeters or inches from the left edge of the printable area to the center of the circle. *Starting point of Circles or Ellipses is in the center*
r	= Rotation.Circles and ellipses can be rotated in steps of 1degrees from 0 to 359 degrees. This makes for sure less sense for circles. Visible effects can be seen on ellipses...
C	= Circle
radius1	= horizontal radius
radius1	= vertical radius
width	= width of the circle line in millimeters or inches *Filled circles or ellipses can be printed if the width is not set*

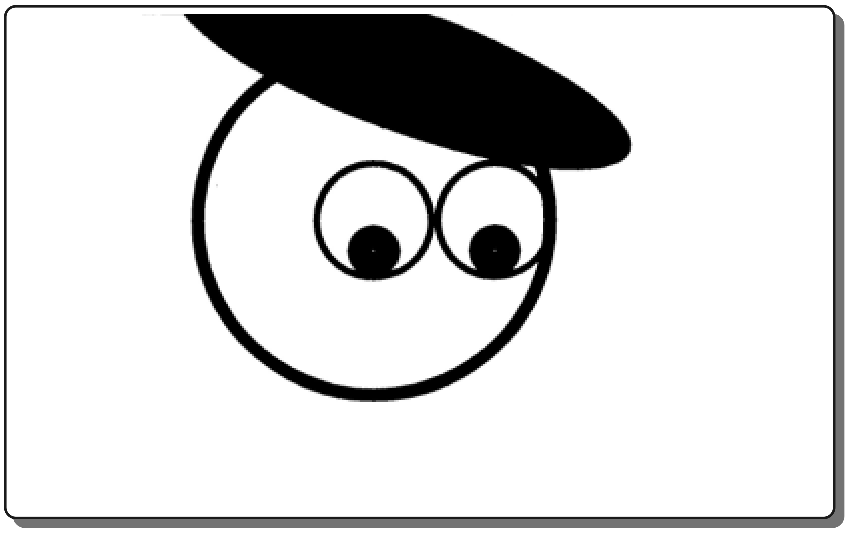
continued on the next page

G - Graphic Definition - Circle

[,options] =	
,fill	= filling of the graphic object with a specified pattern or with dot density. (see graphic option „fill“)
,shade	= shading option (gradient filling - see graphic option „shade“)
,outline	= outline option - prints an outline around the filled graphic object with the thickness of 1 dot. (see graphic option „outline“)

Example:

```
m m
J
S l1;0,0,68,71,100
G 45,10,340;C:40,10,44[S:100,50,80]
G 40,35,0;C:30,30,2
G 40,35,0;C:10,10,1
G 60,35,0;C:10,10,1
G 40,40,0;C:4,4,4
G 60,40,0;C:4,4,4
A 1
```



G - Graphic Definition - Line

Graphic Type: L - Line

Syntax: `G[:name;]x,y,r;L:length,width[,start[,end]][,options] CR`

G	=	Graphic field definition command.
[:name;]	=	Optional field name. Maximum length 10 characters, no special characters allowed, field name must be unique. The field name can be used for further operations, such as Replace field name (See the „R“ command for details) or just as a comment.
x	=	Horizontal coordinate of the start position in millimeters or inches from the left edge of the printable area to the start point of the line
y	=	Vertical coordinate of the start position in millimeters or inches from the left edge of the printable area to the start point of the line *Starting point of Lines is the center of the starting point of the line*
r	=	Rotation.Lines can be rotated in steps of 1degrees from 0 to 359 degrees.
L	=	Line
length	=	length of the line in millimeters or inches
width	=	width of the line in millimeters or inches
start	=	line start type. **s**= squared **r**=rounded **a**=arrowed
end	=	line end type **s**= squared **r**=rounded **a**=arrowed

Lines will print squared without the start / end parameters

Continued on the next page.

G - Graphic Definition - Line

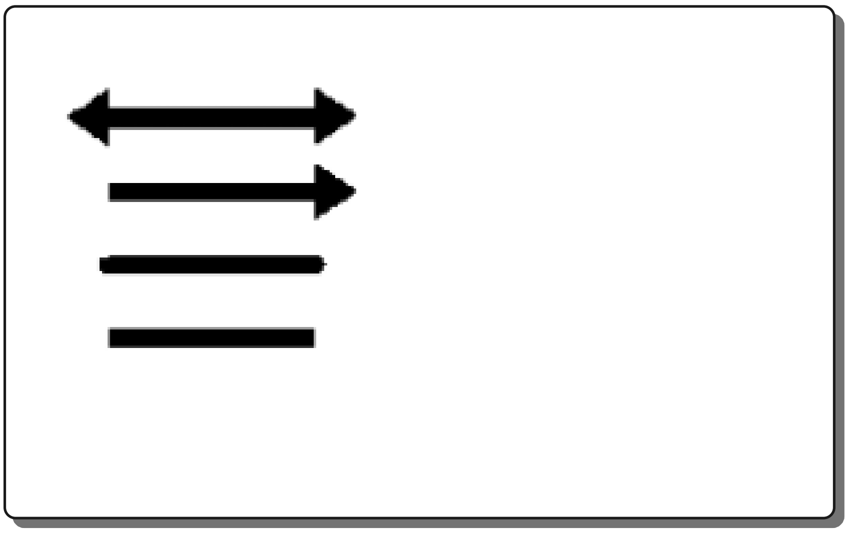
Graphic Type: L - Line

[,options] =	
,fill	= filling of the graphic object with a specified pattern or with dot density. (see graphic option „fill")
,shade	= shading option (gradient filling - see graphic option „shade")
,outline	= outline option - prints an outline around the filled graphic object with the thickness of 1 dot. (see graphic option „outline")

Example:

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

This example demonstrates how the different line start / end parameters are printing, depending which option is used.



G - Graphic Definition - Rectangle

Graphic Type: R - Rectangle

Syntax:

```
G[:name;]x,y,r;R:width,height[,ht [,vt]][,options] CR
```

G = Graphic field definition command.

[:name;]	=	Optional field name. Maximum length 10 characters, no special characters allowed, field name must be unique. The field name can be used for further operations, such as Replace field name (See the „R“ command for details) or just as a comment.
x	=	Horizontal coordinate of the start position in millimeters or inches from the left edge of the printable area to the start point of the rectangle.
y	=	Vertical coordinate of the start position in millimeters or inches from the left edge of the printable area to the start point of the rectangle. *Starting point of rectangles is the upper left corner, outside of the rectangle*
r	=	Rotation.Rectangles can be rotated in steps of 1degrees from 0 to 359 degrees.
R	=	Rectangle
width	=	width (horizontal) of the rectangle in millimeters or inches
height	=	height (vertical) of the rectangle in millimeters or inches
ht	=	horizontal line thickness in millimeters or inches
vt	=	vertical line thickness in millimeters or inches

Filled rectangles are printed, if "width" is not set.

continued on the next page

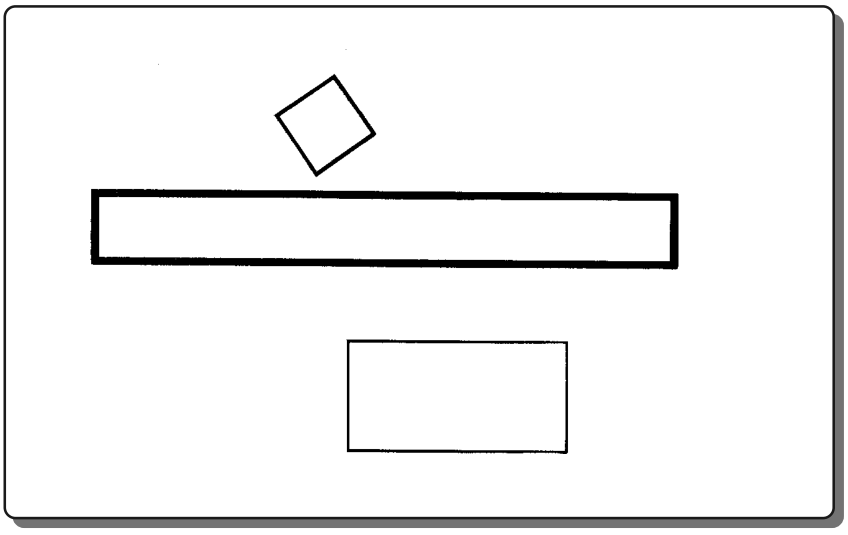
G - Graphic Definition - Rectangle

Graphic Type: R - Rectangle

[,options] =		
,fill	=	filling of the graphic object with a specified pattern or with dot density. (see graphic option „fill“)
,shade	=	shading option (gradient filling - see graphic option „shade“)
,outline	=	outline option - prints an outline around the filled graphic object with the thickness of 1 dot. (see graphic option „outline“)

Example:

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



G - Graphic Definition - Option: Fill

Graphic Option: Fill

Fills a graphic object with redifined patterns

Syntax:

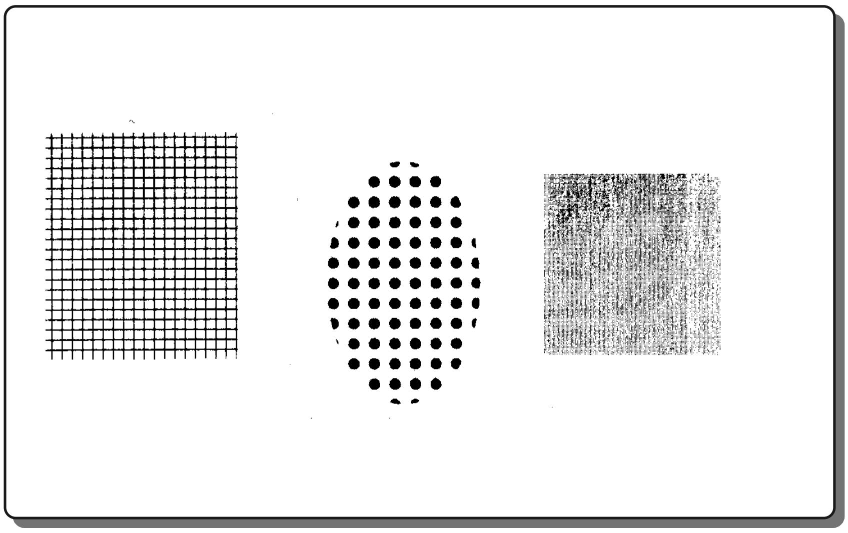
```
G[:name;]x,y,r;ge:settings[F:options] CR
```

F: = Fill parameter.

options	=	Fill pattern option, with following valid input: 0%, 6%, 12%, 25%, 38%, 50%, 100% (for dot density) predefined patterns: left, right, dots, grid, and diamond user1, user2, user3, user4 (downloaded images 32 by 32 dots)

Example:

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



G - Graphic Definition - Option Shade

Graphic Option: Shade

Produces a shading effect (gradient filling) of a graphic object.

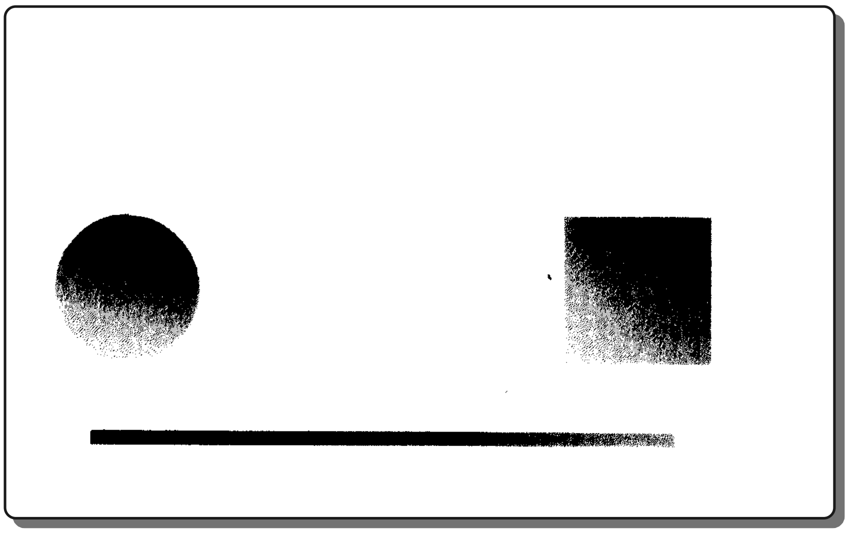
Syntax:

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

S = Shade option	
%1	= Darkness value at the beginning, as a percent of black.
%2	= Darkness value at the end, as a percent of black.
direction	= Shading angle

Example:

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



G - Graphic Definition - Option: Outline

Graphic Option: Outline

Prints an outline around the filled graphic object with the thickness of 1 dot.

Syntax:

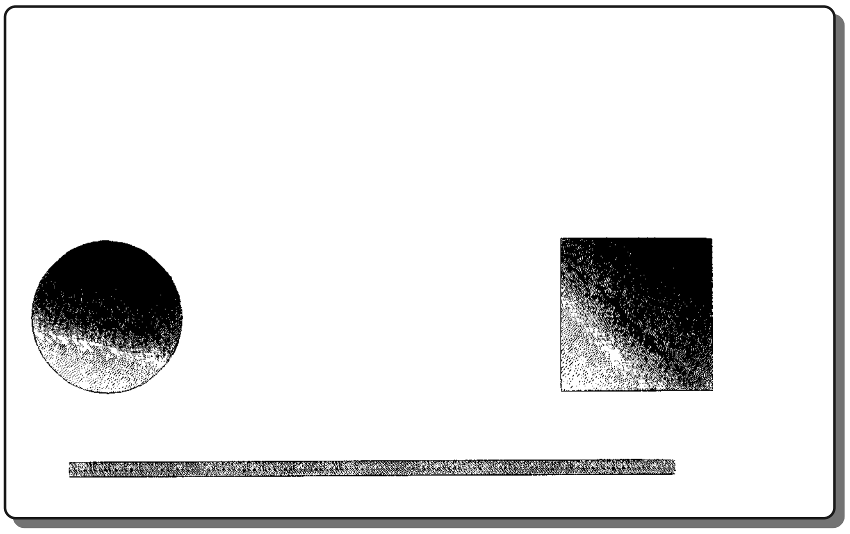
```
G[:name;]x,y,r;type:type options [shade options][O]CR
```

The outline option outlines filled objects. The outline option prints black objects, if outline **[O]** is used for objects which are not filled. (see sample on the next page)

[O] = Outline

Example:

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

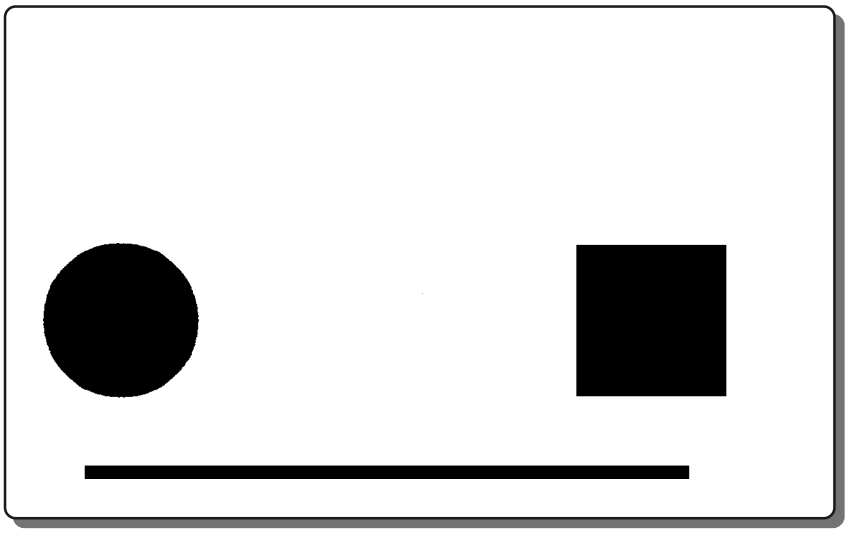


G - Graphic Definition - Option: Outline

Graphic Option: Outline

Example:

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



H - Heat, Speed, Method of Printing, Ribbon

This command sets printing heat, speed and the method of printing for the current label.
Print quality is influenced by the used material and by the print heat and print speed.

Syntax:

```
H speed[,h][,t][,r][,Bb] CR
```

H - Heat, speed, method of printing, ribbon		
speed	=	Print speed in millimeters or inches These values depend on the printer type, please see the operator´s manual for details. A „wrong“ value will automatically rounded by the printer to the next possible value.
h	=	Heat setting (-10 up to +10)
t	=	Type: T=Transfer, D= Direct thermal (Default: T)
r	=	Ribbon saver on/off R0=off, R1=on *
b	=	Back feed speed in millimeters or inches

Example:

```
H 150,0,D,R1
```

Sets print speed to 150mm/s , Heat setting zero, Direct thermal mode and switches the ribbon saver on. (The printer must be equipped with a ribbon saver to use this option)

☞ *The maximum print speed depends on the used printer model. The print speed is automatically set to the maximum if accidentially a higher printspeed is transmitted.*
** The functionality of the ribbon saver command depends on the used printer model and the availablity of a ribbon saver.*

I - Image Field Definition

The I command is used for image printing. (Image stands for pictures, pictograms, logos etc.).
It defines the position and the size of an image on the label.The image has to be downloaded first, before it can be placed on the label. (See „d“ - download command for more details)

Syntax:

```
I[:name;]x,y,r[,mx,my, GOODBADn][,a];name CR
```

I = Image field definition		
[:name;]	**=**	describes the field name and is optional. The maximum length of this name is 10 characters, no special characters allowed. A field name can be used for further operations, such as replacements etc. (See „R“ command for details).
X	=	The x - coordinate is the horizontal start position of an image (in millimeters or inches), the distance between the left margin of a label and the upper left corner of the image.
y	=	The y - coordinate is the vertical start position of an image, the distance between the top margin of a label and the upper left corner of the image. The maximum coordinate depends on the printer type. Please refer to the operator´s manual.
r	=	Rotation -rotates an image in 4 directions. Valid values are 0, 90, 180 and 270. Measurement in degrees.
mx	=	Horizontal magnification factor. Values 1-10. This parameter is optional. Enlarges the image horizontally multiplied by this factor.
my	=	Vertical magnification factor. Values 1-10. This parameter is optional. Enlarges the image horizontally multiplied by this factor.
GOODBADn	=	Used to check the image with the optional barcode verifier. The verifier checks for "Good read" or" Bad read).This is helpful for barcodes with complex contents such as EAN 128.
a	=	Autoload -allows to recall a picture from memorycard.The printer leaves the field empty if no picture has been found. It is required to set the values for mx and my, when Autoload is used ! Please see also the examples on the next pages.

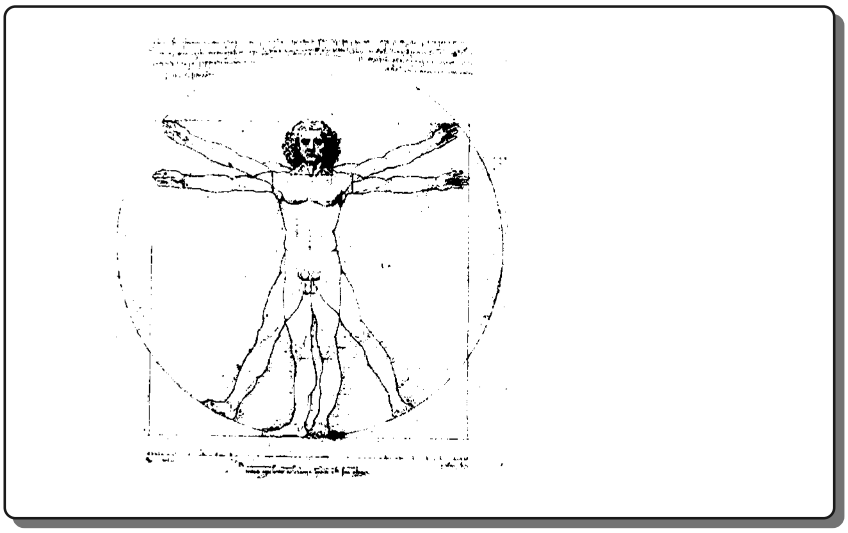
I - Image Field Definition

For best print quality it is recommended to use Images which have been scanned in the same resolution as the printer resolution.
Lower scan resolutions will cause bad print quality, higher resolutions may exceed the available space on the label. Furthermore it is recommended to use pure black and white pictures. Grayscaled pictures may show a loss of data if the grey areas are not dark enough.
By the way: JPEG is a typical compression algorythm or photographic pictures which makes no sense to support this format in label printers.

Example:

```
m m
J
S l1;0,0,68,71,100
I:IMAGE1;20,5,0;HUMAN
A1
```

Prints the picture „HUMAN“ which had previously downloaded to the printer.

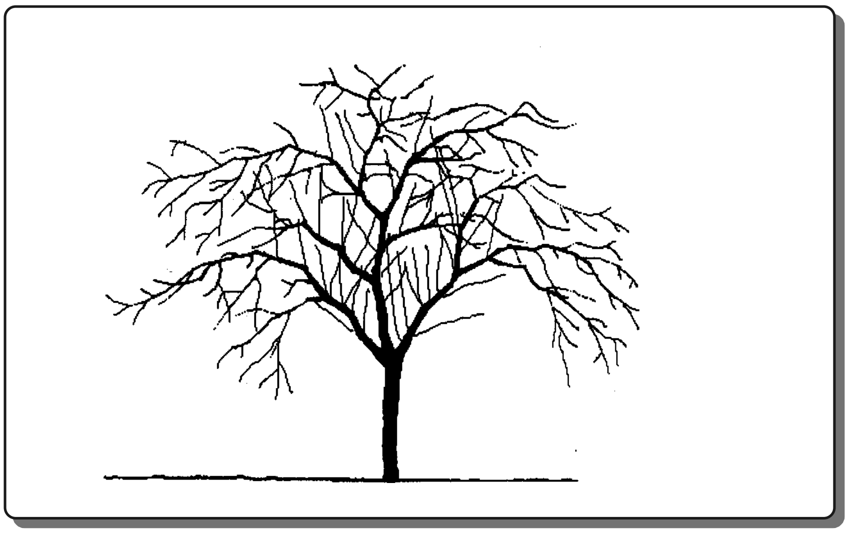


I - Image Field Definition

Example:

```
m m
J
S l1;0,0,68,71,100
I:IMAGE1;10,10,0,2,2,a;tree
A1
```

This example recalls the picture with the name " tree.bmp " from any memory card of the printer and prints it resized (enlarged) by the factor 2 in x- direction and factor 2 in y direction. Please keep in mind that enlarging pictures can have a negative influence on the printout quality.



J - Job Start

The J command „tells „ the printer, that the following data contains label specific data. It starts a new print job.

Syntax:

```
J [comment] CR
```

J - **J**ob start command.

comment	=	Optional text which may describe the label. This optional text will be displayed on the printers LC Display instead of the original filename when it is recalled from the optional memory card. Maximum length is 16 characters.

Example:

```
J Adress Label
```

Defines the job start and names the label „ Adress Label“.
Adress Label will be displayed in the printer´s LC Display when the label is recalled from the optional memory card. The printer „looks“ into each label on the memory card and controls if an alternative Label description is available. This description is shown instead of the original label name which is limited to 8 characters.

M - Memory Card Access

cab printers are prepared for multiple possibilities if the built in or the optional memory is used.
The M commands (Memorycard -commands) are used for a couple of operations, described on the next pages.

Following memory types are supported:

1. Internal Flash File system - called IFFS in the following text.
IFFS is not required for regular applications and has some restrictions. We recommend to use Compact flash memory cards for the most applications and for the highest flexibility.
The restrictions are mainly all commands which try to write intothe memory, as the "E..." commands, "[WLOG]" and "[WTMP]".

2. Compact flash cards - at the moment up to a theoretical maximum of 256 GB memory size.

3. PC Card - also called PCMCIA card memory, which might be SRAM or Flash memory
(Usage not recommmended, but helpful to transfer data from previous printers which used that memory type.

4. USB MSD devices (USB - Mass Storage Devices) such as the most „USB memory sticks“
(It is not possible to guarantee that all of the USB devices on the market will work properly. Validation of good or bad quality USB sticks must be done by yourself).
Furthermore external harddisks can be connected which may require in the most cases external power supplies. Maximum supported size is 2 TB. (Maximum file size is 4 GB)
Please note that only FAT16 and FAT 32 filesystems are supported. NTFS, EXT2 or EXT3 etc. are not supported. It is not recommended to format a huge harddisk in the printer, as the USB 1.1 port would require incredible time to finish that.

Why use additional memory ?

Memory cards are normally used, if a printer runs in „Stand Alone Mode“. Data from memory cards can be easily recalled or filled with variable data with an optional PC keyboard or barcode scanner, which can be attached through the USB port of the printer.
Furthermore the optional cab database connector (later described in this manual) can be used to recall fixed data from the memory card and connect additionally to the network to recall information from a SQL database.

M - Memory Card Access

Some applications use the memory card to recall labels for printing and send the variable field contents from an other application.
This is one of the simple methods which is often used to connect cab printers to SAP or to IBM mainframe computers.

Syntax:	`Mx...`*CR*

M... - Memory card access with following variations for x:

c [path]	=	Memory card **c**ontent request
d [path]	=	Memory card **d**elete files
f	=	**F**ormat memory card
l type;[path]name=		**L**oad file from memory card
n type;[path] old , new	=	Re**n**ame file on memory card. It is not allowed to set a path name for "new"
r	=	**R**eturn to the beginning of the file. allows simple loops
s type;[path]name	=	**S**ave file on card
u type;[path]name	=	**U**pload data from memory to the attached computer

Details and examples for each command are described on the next pages.

IMPORTANT ! *We highly recommend to use Compact Flash cards which are manufactured by* ***SANDISK*** *who is the original developer of Compact Flash cards.*
Other CF cards may cause problems , such as data loss, incompatibility or read and write errors.

M - Memory Card Access

Depending on the used memory type you may recognize different folders on the memory card. Best viewed by connecting the printer through its network interface, using FTP access.

Memory card access with FTP connection:

The of the most powerful possibility to run a cab printer is to connect it in a network.

As the printing systems are equipped with an ethernet interface it is an easy way to access them by using FTP.
To get full access to the printer requires that user name and password are transmitted by FTP.

The user (login) name is always preset to „***root***“ and the ***password is the 4 digit PIN of the printer***
(PIN settings can be done In the setup menu of the printer or through the web applet)

Following memory card folders may appear if the printer is accessed by FTP:

card -	Default memory card (This might be either the compact flash card or the pccard, IFFS or USB memory, whatever is selected as default in the setup of the printer
cf -	CompactFlash card (appears if a cf card is inserted, but any other memory is selected as default memory)
cfext -	External Compact Flash card - if an additional external operation panel is used and a CF card is plugged in. (Appears if a cf card is inserted, but any other memory is selected as default memory)
iffs -	„Internal Flash File System“ - offers the possibility to save data like on all other memory cards. Is always shown as IFFS unless it had been selected as default memory.
pccard -	PCMCIA memory card (PCcard) (appears if the cf card is selected as default card and the pccard is additionally plugged in)
usbmem -	USB memory (MSD - subclass 6,Protocol 0x50 - FAT 16 or FAT32 formatted, max. size of the first partition is 2 GB). USB memory sticks need to follow this specs, otherwise they are not useable in the printer. Only one USB Mass storage device is supported. The printer connects to the USB device which is fastest detected.

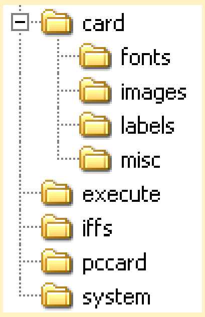
Memory which is not attached to the printer will not be shown.

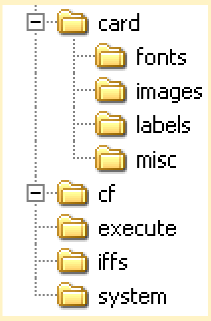
M - Memory Card Access

Additional folders which are displayed by using FTP connection:

execute -	is a folder which executes immediately the label which is transmitted by FTP to that folder. (a label will be processed as soon as it is copied into that folder)
system -	contains the firmware of the printer which also can be simply updated, just by copying the new firmware version by FTP to the printer.

Example:


card
fonts
images
labels
misc
execute
iffs
pccard
system


card
fonts
images
labels
misc
cf
execute
iffs
system

We highly recommend to use CF cards for future developments.
Pccards will not show the subdirectory structure where the files are sorted into the folders: fonts, images, labels and misc.
Please note, that the CF connection in the printer is much faster than the external CF card.
(Time critical applications may require this built in card slot)
Specialnote about the internal flash file system (IFFS): It is not possible to use TMP files (for serial numbering) or write to the IFFS memory during printing

M - Memory Card Access - content request

Syntax: `Mc [path] CR`

Mc... - Memory card: **c**ontent request. Requests the content of a directory path on the memory card (analog to the DOS command „DIR“)

path	=	optional parameter to select the pathname where the files are located.	
	=	**/card/**	-recalls the card content of the optional compact flash card. Leaving this option blank recalls automatically the content of the Default memory card.
	=	**/iffs/**	-recalls the content of the internal flash file system
	=	/**cfext**/	-recalls the content of the external Compact Flash card
	=	/**pccard**/	-recalls the content of the PCMCIA card
	=	/**usbmem**/	-recalls the content of the USB memory

Example: `Mc`

Response from the printer:

```
Directory of ‘A3/300      ‘:
ARIAL        TTF  79804    20.05.08 14:37
COMIX        TTF  66080    20.05.08 14:38
MINSTREL     TTF  65692    20.05.08 14:39
NORM101      LBL   1420    20.05.08 14:51
COMPANY      IMG   1012    20.05.08 14:41
BEDANO       TTF  83260    20.05.08 14:43
NORM44       LBL   1530    20.05.08 14:43
EXPLOSIV     IMG   2098    20.05.08 14:49
NORM42       LBL   2104    20.10.08 16:19
102          LBL   1420    20.05.08 14:52
CDPLAYER     DBF   2858    08.11.08 13:03
 15807062 bytes free
```

M - Memory Card Access - delete file from card

Syntax: `Md type;[path]name CR`

Md... - **M**emory card: **d**elete file from card. Deletes (erases) data on memory card

type=	LBL (label), FNT (font), IMG (image), FMT (label format) TMP (temporary file i.e. file which contains a serial number) „type“: FNT erases all TTF fonts, „type“: IMG erases all graphic types with the same name.
path	= optional parameter to select the pathname where the files are located. = **/card/** -deletes the card content of the optional compact flash card. Leaving this option blank deletes automatically the content of the Default memory card. = **/iffs/** -deletes the content of the internal flash file system = /**cfext**/ -deletes the content of the external Compact Flash card = /**pccard**/ -deletes the content of the PCMCIA card = /**usbmem**/ -deletes the content of the USB memory
name	= File name of the file on memory card

Example: `M d IMG;logo`

Deletes all graphic files on memory card with the name „logo“. e.g. this might be logo.bmp, logo.pcx etc.

IMPORTANT: Some labelling programs use also the extension .LBL or .FMT. These file types are totally different and do not contain J-Script commands !

☞

M - Memory Card Access - format card

Syntax:

```
M f;name CR
```

M f...		**- M**emory card: **f**ormat card. Formats the memory card (creates a file system) All rinters create automatically a folder structure to separate the data to the specified locations.
name	=	Name for the memory card

Example:

```
M f;MYDATA
```

formats the memory card and writes the volume name „MYDATA“ which is usually the name of the used printer.

Following folders will be generated on the memory card:

Fonts
Labels
Graphics
Misc

The Fonts folder is used to save all true type fonts. (Extension .TTF)
The Labelsfolder is used to save labels in JScript Format (Extension .LBL)
The Graphics folder contains all possible graphic formats. (Extensions: .IMG, .PCX, .BMP, .GIF, .MAC, .TIF, .PNG)
The Misc Folder is used to save DBase IV databases, serial numbers and temporarary files (Extensions: .DBF, .SER, .TMP)

The Misc folder can also contain one or more firmware files, which are displayed in the „SERVICE“ menu of the printer to update the firmware from memory card or XML files which can contain a backup of the printer´s settings.

M - Memory Card Access - load file from card

Syntax:

```
M l type;[path]name CR
```

M l... - Memory card: **l**oad file from card. Load data from memory card	
type=	LBL (label), FNT (font), IMG (image), FMT (label format)
path	= optional parameter to select the pathname where the files are located. = **/card/** Leaving this option blank accesses automatically the file of the Default memory card. - loads the file of the optional compact flash card. = **/iffs/** - loads a file from the internal flash file system = **/cfext/** - loads a file from the external Compact Flash card = **/pccard/** - loads a file from the PCMCIA card = **/usbmem/** - loads a file from the USB memory
name	= Name of the file

Example:

```
Ml LBL;TESTLBL
A2
```

Loads the label with the name TESTLBL from the default memory card and prints 2 labels

Example:

```
Ml LBL;/IFFS/TESTLBL
A4
```

Loads the label with the name TESTLBL from the internal flash file system and prints 4 labels

M - Memory Card Access - rename file on card

Syntax:	`M n type;[path]old,new CR`

M n... - Rename a file on the memory card.

type=		LBL (label), FNT (font), IMG (image), FMT (label format) TMP (temporary file i.e. file which contains a serial number) „type“: FNT erases all TTF fonts, „type“: IMG erases all graphic types with the same name.
path	=	optional parameter to select the pathname where the files are located.
	=	**/card/** -deletes the card content of the optional compact flash card. Leaving this option blank deletes automatically the content of the Default memory card.
	=	**/iffs/** -renames files in the internal flash file system
	=	/**cfext**/ -renames files on the external Compact Flash card
	=	/**pccard**/ -renames files on the PCMCIA card
	=	/**usbmem**/ -renames files in the USB memory
old	=	Existing file name of the file on memory card
new	=	New file name of the file on memory card

M - Memory Card Access - repeat last file content

Syntax:

```
M r CR
```

M r - **M**emory card: **r**epeat last file content. Jump to start of file. This command can be used to implement simple loops.

Example:

```
Ms LBL;LOOP
m m
J
S l1;0,0,68,70,100
T:Text1;20,10,0,3,7;[?:Art-No:]
A3
Mr
Ms LBL
```

Saves the label „LOOP“ on the printer´s memory card. This label will show the word „Art-No:“ in the display and waits for data input. After data is keyed in it will print 3 labels and repeats the question for the „Art-No“ in the display.

M - Memory Card Access - store data

Syntax:

```
M s type;[path]name CR
```

M s... - **M**emory card: **s**tore data on card. Stores data on memory card.

type=		LBL (label), FNT (font), IMG (image), FMT (label format)		
path	=	optional parameter to select the pathname where the files are located.		
	=	**/card/**		Leaving this option blank saves automatically the content on the Default memory card.
			-	saves the file on the optional compact flash card.
	=	**/iffs/**	-	saves the file in the internal flash file system
	=	/**cfext**/	-	saves the file on the external Compact Flash card
	=	/**pccard**/	-	saves the file on the PCMCIA card
	=	/**usbmem**/	-	saves the file in the USB memory
name	=	File name of the file which shall be saved on memory card		

Example:

```
Ms LBL;ADDRESS
S l1;0,0,36,38,89
T:Text1;20,10,0,3,pt25;Worldwide
A5
Ms LBL
```

☞ Saves the label „ADDRESS“ on the printer´s memory card. This label will automatically print 5 labels when it is recalled .

*A label will immediatly start printing when the printer is switched on, if the label has been saved with the reserved name „**DEFAULT.LBL**“ !*

Files are saved on the memory card in UNICODE format ! An editor which can handle Unicode files is required to edit these files. Wordpad can be used as editor for Unicode files. Notepad is not able to handle that file format.

***IMPORTANT NOTE:** The „Ms“ command causes the printer to save a label to the memory card, which is plugged into a printer.*

Do NOT use this command, if the data is saved by FTP directly to the memory card or if the data is saved directly on a memory card which is plugged in a PC.

*This would cause a infinite loop on the printer, as the printer tries to recall the label where the first command tells to save the label on card and so on - and the display would show „**Memory overflow**“.*

M - Memory Card Access - upload data

Syntax:

```
M u type;[path] name CR
```

M u... -**M**emory card: **u**pload data. Uploads file contents from memory card as binary data.

Example:

```
M u LBL;TESTLBL
```

Uploads a label named TESTLBL from the memory card. If Hyperterminal is used to receive the data it is possible to copy the file to the clipboard and paste it into a text editor such as Wordpad.

Note: When uploading other types of files, such as IMG, the data is sent as raw binary data.

O - Set Print Options

The O command is used to set a wide range of options which influences the complete label.
Important: *The "O" command must be located directly after the label size command "S....."*

Syntax `O [Ax=y][,B][,Cx][,D][,E][,F][,Hx][,M][,N][,P][,R][,S][,T][,U] CR`

O - Print Options command.		
		Applicator parameters The applicator parameters are only available for printers with (optional) applicator. The applicator parameters options are not available for Mach4. Hermes A does not support that command with the existing applicators.

Ax=y		Set parameter x to y (in ms, 0-1000ms).
		x=0: Start delay supporting air (0-1000ms)
		x=1: Stop delay supporting air (0-1000ms)
		x=2: Start delay print (0-1000ms)
		x=3: Lock time (0-1000ms)
		x=4: Blow time (0-1000ms)

B	=	Lower side is copy of the upper side. (Only available on double sided printers)
Cx	=	additional cutting time for the optional perforation cutter. Values for x = 0.0 - 10.0 (This value has influence on the cutting depth)
D	=	Cutting or dispensing labels always with back feed
E	=	Ignore paper end (not allowed if the printer runs in continuous form mode) - Settings are displayed in the section which describes the Size command (S....)
F	=	Discard the label positions, causes new synchronisation of the material
Hx	=	additional Offset between upper and lower printhead (Only available on double sided printers) x value is in millimeters
M	=	Mirrored label printing

continued on the next page

O - Set Print Options

N	=	**N**egative (inverted) printout of the complete label
R	**=**	**R**otate the label contents 180 degrees
P	=	**P**rintmode **-** backfeed option always / smart backfeed „always“ feeds the label back and starts printing at the label margin, while „smart“ suppresses the feedback. „**P**“ activates the smart option while „**D**“ activates the „always“ option. This option overwrites temporarily the settings in the printer´s setup. Using the „smart“ mode has the benefit that the printer processes thelabels faster as the time is saved for pulling the labels back. Nevertheless a negative effect may appear in the area where the label is stopped under the printhead. This may cause a small horizontal white line in the area. If this happens within an object, then you must select the „**D**“ option to avoid this effect.
S	=	**S**ingle label buffer. The following label will be processed when the actual one has finished printing.
T	=	Enables the „**T**ear off mode“ which feeds the label more forward after printing, so that it could be taken easier away.
U	=	**U**nique label **-** suppresses the Pause / Reprint possibility to avoid that a label will be printed twice.

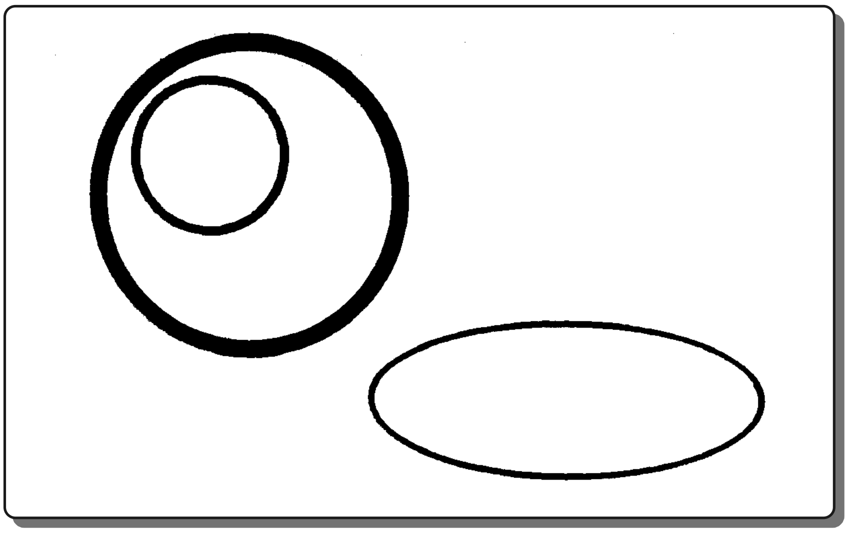
☞

Important: *The "O" command must be located directly after the label size command "S....."*

O - Set Print Options

Example:

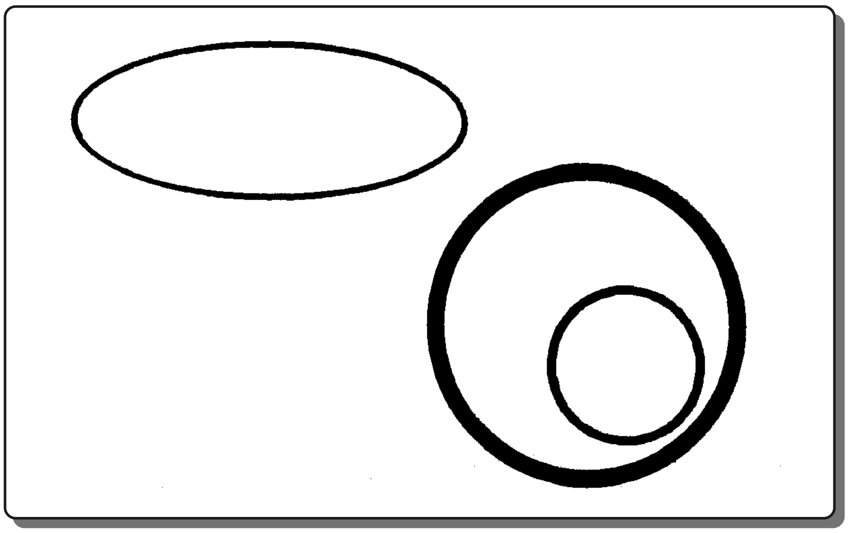
```
J
S l1;0,0,68,71,100
G 65,50,0;C:25,10,.7
G 25,25,0;C:20,20,2
G 20,20,35;C:10,10,1
A 1
```



Example:

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

The **O R** command rotates the complete printout of a label. The first example does not use the „O“ command.



P - Set Peel-Off Mode

This command needs an optional peel off sensor, which varies from printer type to printer type.
This command pauses the printer after each label. The next label prints, when the actual label is removed.
The P command is very important if an applicator is used.

Syntax:

```
P[disp] CR
```

P - Peel-Off Mode command.

disp	=	displacement in millimeters or inches (optional parameter) positive and negative values can be used, depending in which direction the displacement should work.

☞ *The „P“ command needs to be placed after the definition of the page size ! („S“- command)*

R - Replace Field Contents

The usage of the „R“ command is to replace data contents of previously downloaded label. Normally this is a label which is recalled from memory card into the printer´s internal memory. The R command offers an easy way to print multiple labels with a minimum data transmission.

The R command identifies the data by its field name and inserts a new value.

Syntax:

```
R name;data CR
```

R - Replace command.

name	=	The name of the text data field or barcode data field.
data	=	The new value of the field, which will replace the data of the former label.

Example:

```
m m
J
O R
S l1;0,0,68,71,100
T:REP; 12,25,0,3,6;Good Morning
A1


R REP;cab printers
A2
R REP; Hello together
A1
R REP; Last label
A1
```

This example transmits a label and replaces the single variable in this label with other data.

☞ *Additional information about using cut commands together with Replace fields can be found at „C - Cutter Parameters“.*

S - Set Label Size

This command defines the width and length of a label and has some additional options.

Syntax:

```
S[ptype;]xo,yo,ho,dy,wd[,dx,col][;name]CR
```

S - Set label size		
ptype;		= photocell type. Sets the type of label sensing. Optional parameter. It is recommended to set it in the label definition.
	e	= endless (continuous) label material without die cuts. Labels sensor is switched off and the height is measured by the amount of micro steps of the printer´s transport motor.
		***Important:** the following character is a lower case* **L** *followed either by 0,1 or 2 !!*
	l0	= senses the reflective marker on the upper side of the label material. (only if the printer is equipped with this sensor!!!) (l0 = small letter L + 0)
	l1	= sets the printer´s sensors for die cut labels with gap. (l1 = small letter L + 1)
	l2	= senses the reflective marker on the lower side of the label material. (l2 = small letter L + 2)
xo		= horizontal displacement, shifts the starting point (zero point) of all horizontal measurements to the left margin of the label.
yo		= vertical displacement, shifts the starting point (zero point) of all vertical measurements to the top margin of the label.
ho		= height of the label in transportation direction.
dy		= height of the label plus height of the gap. (Distance from the starting point of the first label to the starting point of the next label)
wd		= label width measured from the right margin to the left margin.

continued on the next page....

S - Set Label Size

Optional parameters when multiple labels are placed horizontally:		
dx	=	defines the distance from the margin of the first label to the second label in horizontal direction
col	=	number of labels horizontally (default value =1)
name	=	optional text which is shown in the printer´s display. Can be used i.e. to display the required label material which has to be inserted.

please refer also to the "option command" (" O ") to get more infos for special options such as mirroring, reverse printing or double sided printing etc.

Example:

```
S 11;0,0,50,52,100
....
```

This example defines a label size of 50 mm height, distance from one label to the next label (label height + gap) is 52 mm and the width of the label is 100 mm. Displacement horizontal and vertical is zero.

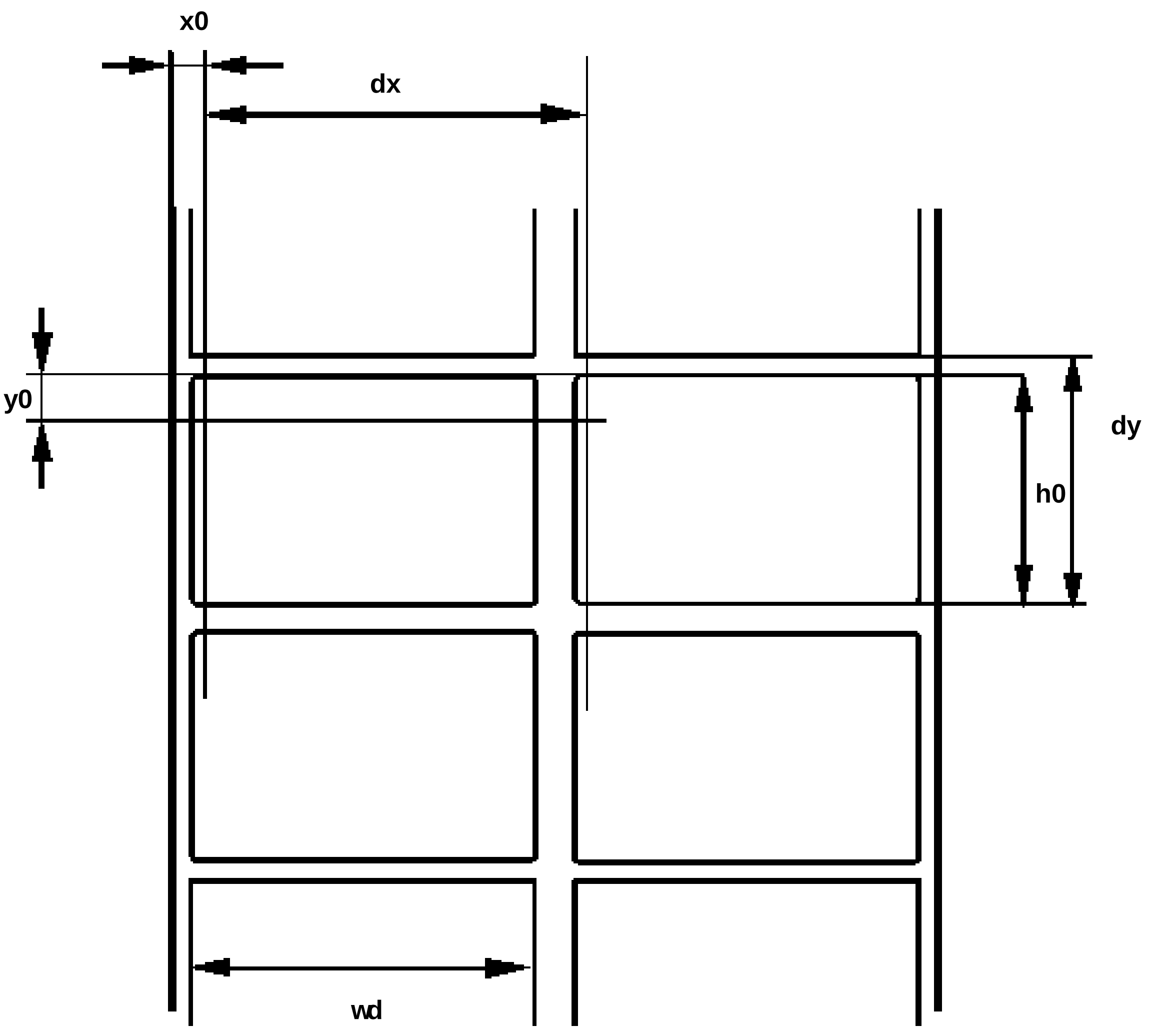
A couple of dependencies:
All numeric values are either in millimeters or in inches, depending on the selected country setting of the printer or depending on the „m „ command.
Maximum values depend on the width of the printhead and on the amount of memory which is responsible for the maximum height of the label. Both parameters depend on the used printer type. Please refer to the operator´s manual for more information.

☞ ***Special note for double sided printers (XD4+...):***
The printheads are treated like a 8 inch printhead, splitted in 2 sections. One good method is to create a label in the full width of an 8 inch wide printhead and position the required data on the left half for the lower printhead and the right half for the upper printhead.
Maximum width would be 2 x105.6 mm on the XD+ with 300 dpi printhead.

Setting the correct label size is one of the important points to get a precise position of your label contents.

S - Set Label Size

```
S[ptype;]xo,yo,ho,dy,wd[,dx,col][;name]CR
```



S - Set Label Size

The settings and the positioning of different fields on the XD (double sided) printer requires a clear understanding where all the content has to be placed. The next sample shall help to get a better understanding. Additionally some cutting commands have been added.

Example:

```
m m
J Top/Bottom different
H 100,10,T
O R
O F
S e;.0,.0,40.0,40.0,211.0
T:TEXT22;55,0,90.0,5,3.5;[J:c40] cab XD4 printer
T 52,0,90.0,5,3.0,q90;[J:c40]Double sided-Bottom
T:TEXT26;157.7,0,90.0,5,3.5,q90;[J:c40]Double sided-Top
T 161.5,0,90.0,5,3.5;[J:c40] cab XD4 printer
C s
C p
C e
A [?]
```

The print width is on both heads 105,6mm. That means, the middle of the first print head is at 52,8mm and the middle of the second print head is at 158,4mm.
If you want to place f.e. a text on a continous material in the middle at the upper side, you have to place it at 158,4.

T - Text Field Definition

The most used command to program a label is the „T“ command which is used for text field definitions.This command influences the size, shape, rotation etc. of any shown textlines on a label.

Syntax:

```
T[:name;]x,y,r,font,size[,effects];text CR
```

T =		Text field definition command.
:name;	=	A field name can be set for further operations such as replacing text contents in a predefined text field or for calculations or for the concatenation of multiple fields. The field name is an optional parameter. Maximum length 10 digits, ALPHA signs and digits only. Text field names are case sensitive.
x	=	horizontal start position - distance from the left starting point of the label in millimeters or inches.
y	=	vertical start position - distance from the top margin starting point of the label in millimeters or inches.
r	=	Text field rotation. Vector fonts and downloadable true type fonts can be rotated 360 degrees in steps of 1 degree. Bitmap fonts can be rotated in 4 directions (0, 90, 180 and 270 degrees)
font	=	specifies a font type, set by a number which might be an internal printer font (vector or bitmap) or a downloaded true type ™ font. Vector fonts are scalable fonts which appear in a smooth shape when magnified. Following font types are available: **BItmap fonts:**

font no.	Name	Type	Description
-1	_DEF1	Bitmap	Default-size 12x12 dots
-2	_DEF2	Bitmap	Default-size 16x16 dots
-3	_DEF3	Bitmap	Default-size 16x32 dots
-4	OCR_A_I	Bitmap	OCR-A Size I
-5	OCR_B	Bitmap	OCR-B

continued on the next page....

T - Text Field Definition

	Vektorfonts **font no. Name Type Description** 3 BX000003 VectorSwiss 721™ 5 BX000005 VectorSwiss 721 Bold ™ 596 BX000596 VectorMonospace 821 ™ **Optional internal cab fonts:** 1000 GEHEI21M VectorAR Heiti Medium *(Mandarin - (chinese) font)* 1010 GARUDA VectorGaruda (Thai font) ☞ *Garuda is available free of charge from the cab website, AR Heiti medium is not free of charge.*
size	= sets the the character size The size of scaleable (vector) fonts can be set in millimeters or inches, or by point size "pt x". The size of bitmap fonts is predefined and can be enlarged by the usage of magnification factors in horizontal and vertical direction. xn,yn where xn is the horizontal magnification (1-10 times) and yn stands for the vertical expansion (1-10 times)
effects	= Defining effects is optional. Special effects can be applied to the used fonts. Which effects are available depends on the used font. Following can be applied: **b** = bold **s** = slanted **i** = italic **n** = negative (reverse print) **u** = underlined **l** = light **z** = slanted left **k** = kerning **v** = print text in vertical alignment. **qn** = squeeze characters, default value is 100. Possible values: 10-1000 **hn** = width of upper case "H" , with n millimeters or in inches. **mn** = horizontal text spacing , with n millimeters or in inches.

continued on the next page....

T - Text Field Definition

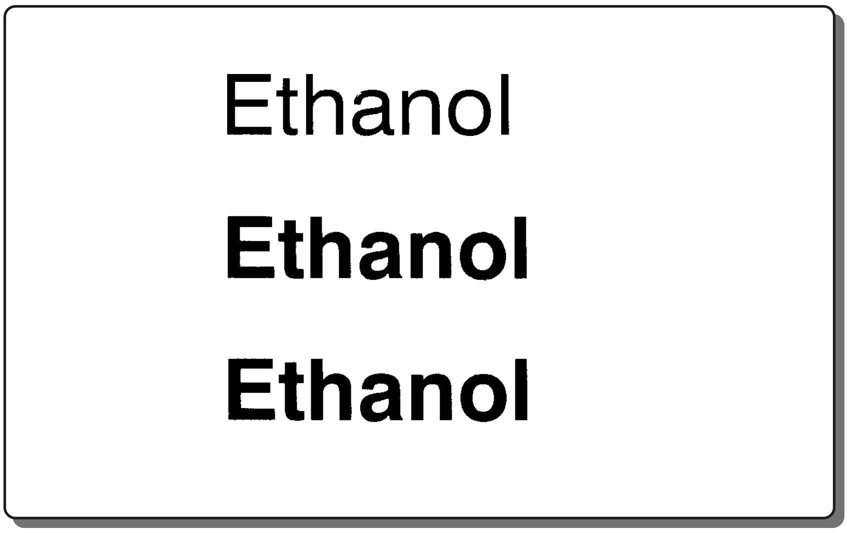
effects =	The following effects are only available together with internal vector font and additional True type fonts :
	frn = **r**ight frame for text objects **fln** = **l**eft frame for text objects **fun** = **u**pper frame for text objects **fdn** = lower (**d**own) frame for text objects
	The following effects are only available together with internal bitmap fonts: **o** = **o**utlined (not available for OCR font) **g** = **g**ray (not available for OCR font) **xn** = horizontal expansion factor (n = 1-10) **yn** = vertical expansion factor, (n = 1-10)
text =	data string in a selected codepage. Please have a look to the setup menu of your printer. The text area allows also the usage of special functions and options, described later later in this manual.

T - Text Field Definition

Example:

```
J
S l1;0,0,68,71,100
T 16,20,0,3,12;Ethanol
T 16,40,0,3,12,b;Ethanol
T 16,60,0,5,12;Ethanol
A2
```

In this example we want to explain, that the same effect can be shown when a text is bold from the original structure or when the option „b“ is used to print a bold font.

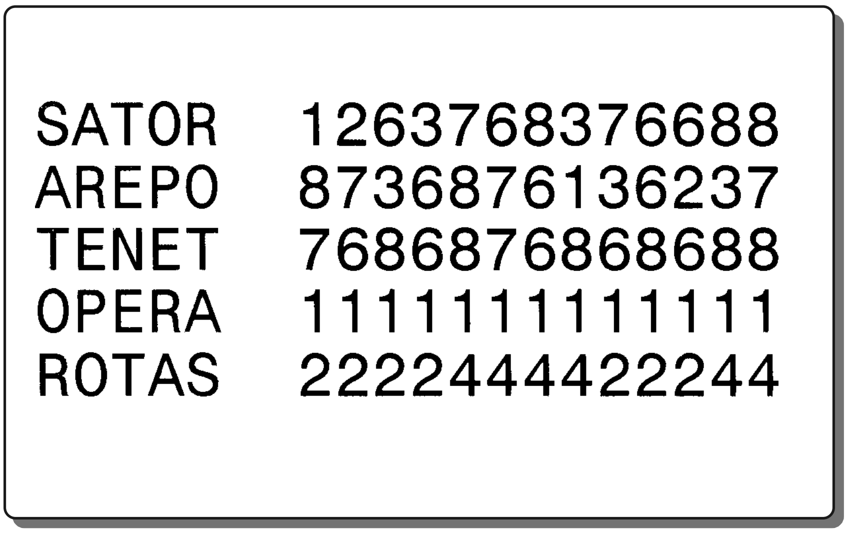


T - Text Field Definition

Example:

```
J
S l1;0,0,68,71,100
T 2,15,0,596,8;SATOR  1263768376688
T 2,23,0,596,8;AREPO  8736876136237
T 2,31,0,596,8;TENET  7686876868688
T 2,39,0,596,8;OPERA  1111111111111
T 2,47,0,596,8;ROTAS  2222444422244
A2
```

The internal Monotype font can be used to define tables. The characters of that font have always the same width. This font can be used for tables where all characters or numbers need to be placed in the same column.

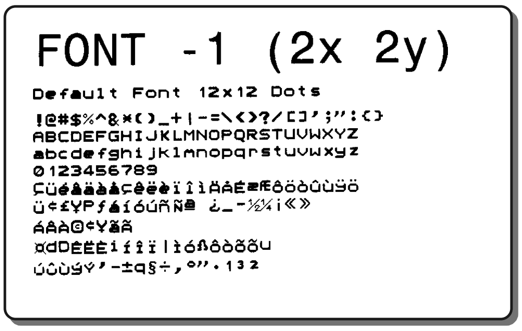


```
SATOR   1263768376688
AREPO   8736876136237
TENET   7686876868688
OPERA   1111111111111
ROTAS   2222444422244
```

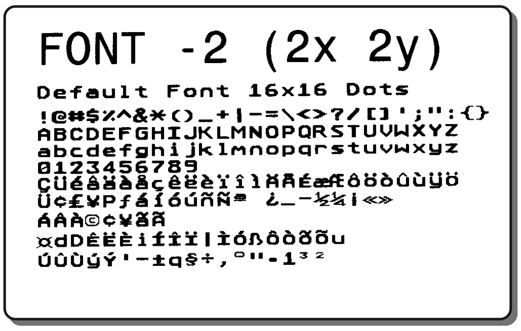
T - Text Field Definition

Internal bitmap fonts
On this page you can see a printout of the printer´s internal bit mapped fonts.

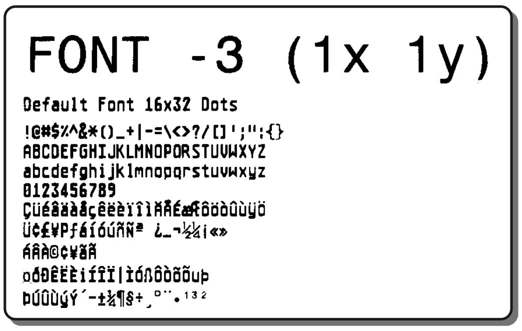
The size of the characters has been enlarged for a better readability



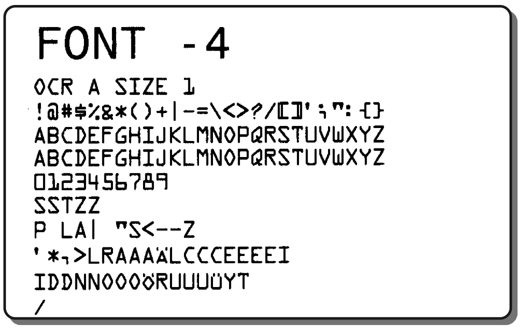
FONT -1 (2x 2y)
Default Font 12x12 Dots
!@#$%^&*()_+|-=\<>?/[]';":{}
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÇüéâäàåçêëèïîìÄÅÉæÆôöòûùÿÖ
Ü¢£¥PƒáíóúñѪ ¿_¬½¼¡«»
ÁÂÀ©¢¥ãÃ
¤ðÐÊËÈıÍÎÏ|ÌÓßÔÒõÕu
ÚÛÙýÝ´-±q§÷,°¨.¹³²



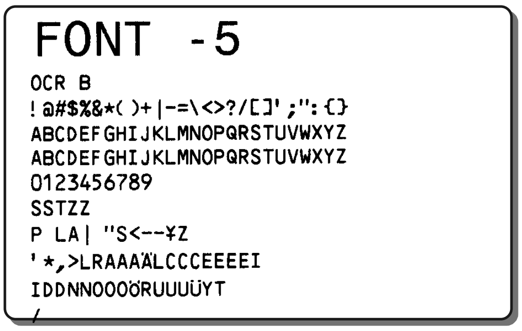
FONT -2 (2x 2y)
Default Font 16x16 Dots
!@#$%^&*()_+|-=\<>?/[]';":{}
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÇüéâäàåçêëèïîìÄÅÉæÆôöòûùÿÖ
Ü¢£¥PƒáíóúñѪ ¿_¬½¼¡«»
ÁÂÀ©¢¥ãÃ
¤ðÐÊËÈıÍÎÏ|ÌÓßÔÒõÕu
ÚÛÙýÝ'-±q§÷,°".¹³²



FONT -3 (1x 1y)
Default Font 16x32 Dots
!@#$%^&*()_+|-=\<>?/[]';":{}
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÇüéâäàåçêëèïîìÄÅÉæÆôöòûùÿÖ
Ü¢£¥PƒáíóúñѪ ¿_¬½¼¡«»
ÁÂÀ©¢¥ãÃ
¤ðÐÊËÈıÍÎÏ|ÌÓßÔÒõÕuþ
ÞÚÛÙýÝ´-±¾¶§÷¸°¨.¹³²



FONT -4
OCR A SIZE 1
!@#$%&*()+|-=\<>?/[]';": {}
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
SSTZZ
P LA| "S<--Z
'*,>LRAAAÄLCCCEEEEI
IDDNNOOOÖRUUUÜYT
/



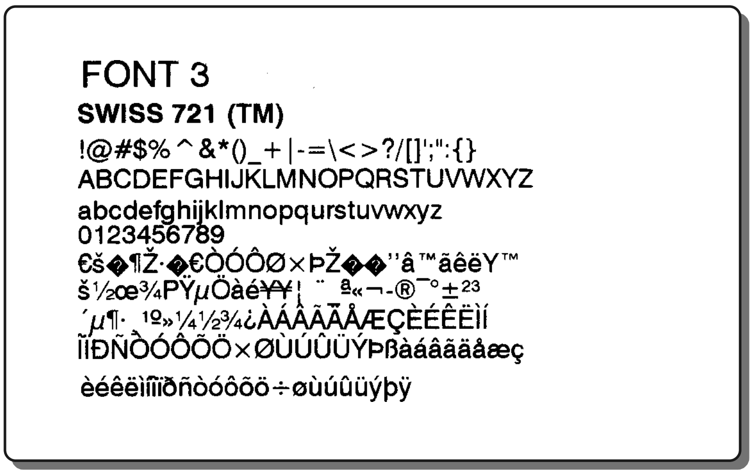
FONT -5
OCR B
!@#$%&*()+|-=\<>?/[]';":{}
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
SSTZZ
P LA| "S<--¥Z
'*,>LRAAAÄLCCCEEEEI
IDDNNOOOÖRUUUÜYT
/

T - Text Field Definition

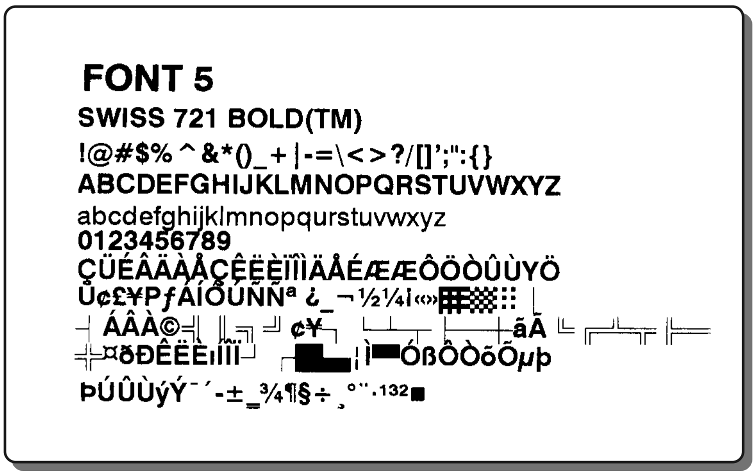
Internal Fonts

This examples show a printout of the scalable fonts of the cab printers. Special characters can be recalled using the [U:... option to recall and print Unicode characters.

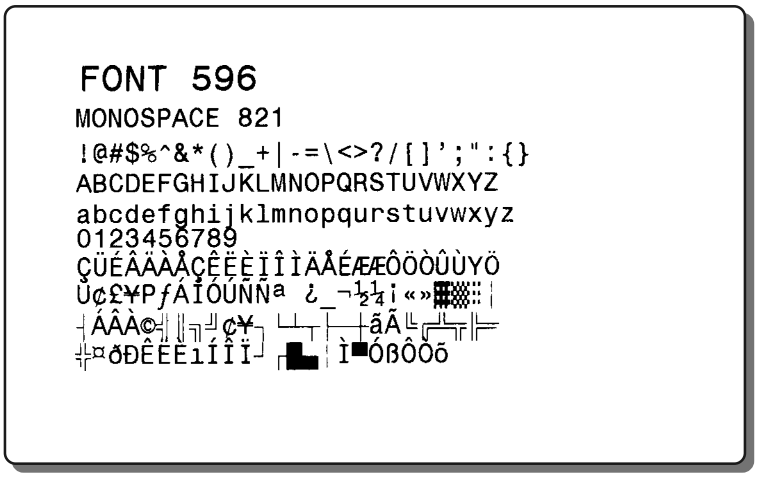
Please see the [U:... option for more details.



FONT 3
SWISS 721 (TM)
!@#$%^&*()_+|-=\<>?/[]';":{}
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqurstuvwxyz
0123456789
€š�¶Ž·�€ÒÓÔØ×ÞŽ��''â™ãêëY™
š½œ¾PŸµÖàé¥¥¦ ¨ ª«¬-®¯°±²³
´µ¶·¸¹º»¼½¾¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍ
ÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæç
èéêëìíîïðñòóôõö÷øùúûüýþÿ



FONT 5
SWISS 721 BOLD(TM)
!@#$%^&*()_+|-=\<>?/[]';":{}
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqurstuvwxyz
0123456789
ÇÜÉÂÄÀÅÇÊËÈÏÎÌÄÅÉÆÆÔÖÒÛÙYÖ
Ü¢£¥PƒÁÍÓÚÑÑª ¿_¬½¼¡«»▓▒░ ¦
┤ÁÂÀ©╣║╗╝¢¥┐ └┴┬├─┼ãÃ╚╔╩╦╠═
╬¤ðÐÊËÈıÍÎÏ┘┌█▄¦Ì▀ÓßÔÒõÕµþ
ÞÚÛÙýÝ¯´-±‗¾¶§÷¸°¨·¹³²■



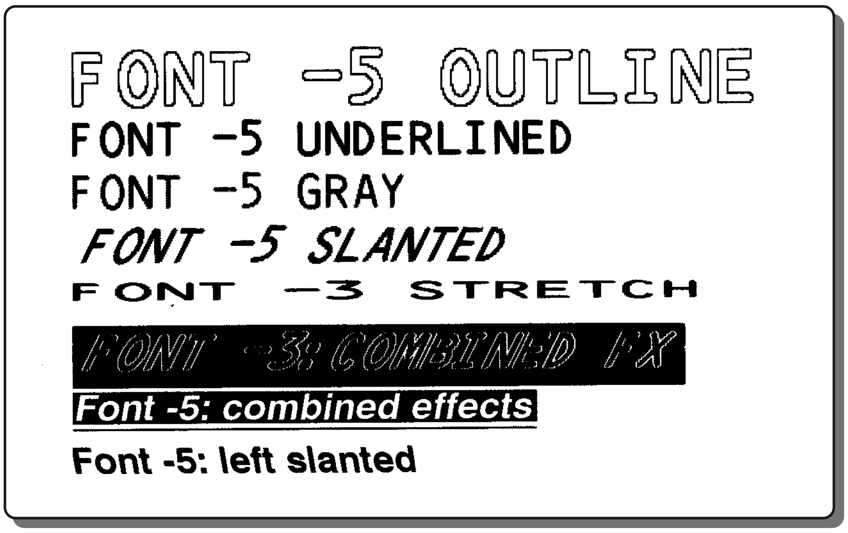
FONT 596
MONOSPACE 821
!@#$%^&*()_+|-=\<>?/[]';":{}
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqurstuvwxyz
0123456789
ÇÜÉÂÄÀÅÇÊËÈÏÎÌÄÅÉÆÆÔÖÒÛÙYÖ
Ü¢£¥PƒÁÍÓÚÑª ¿_¬½¼¡«»▓▒░¦
┤ÁÂÀ©╣║╗╝¢¥┐└┴┬├─┼ãÃ╚╔╩╦╠═
╬¤ðÐÊËÈıÍÎÏ┘┌█▄¦Ì▀ÓßÔÒõ

T - Text Field Definition

This example shows some special effects of the cab printers „Swiss“ font.

Example:

```
J
S 0,0,68,71,100
OR
D 0,5
T 10, 7,0,-5,x3,y3,o;Font -5 outline
T 10,14,0,-5,x2,y2,u;Font -5 underlined
T 10,21,0,-5,x2,y2,g;Font -5 gray
T 10,28,0,-5,x2,y2,s;Font -5 slanted
T 10,33,0,-5,x3,y1;Font -3 stretch
T 10,42,0,-5,x2,y2,s,u,o,n;Font -3:combined FX
T 10,49,0,5,5,s,u,n;Font -5: combined effects
T 10,56,0,5,5,z;Font -5: left slanted
A 1
```

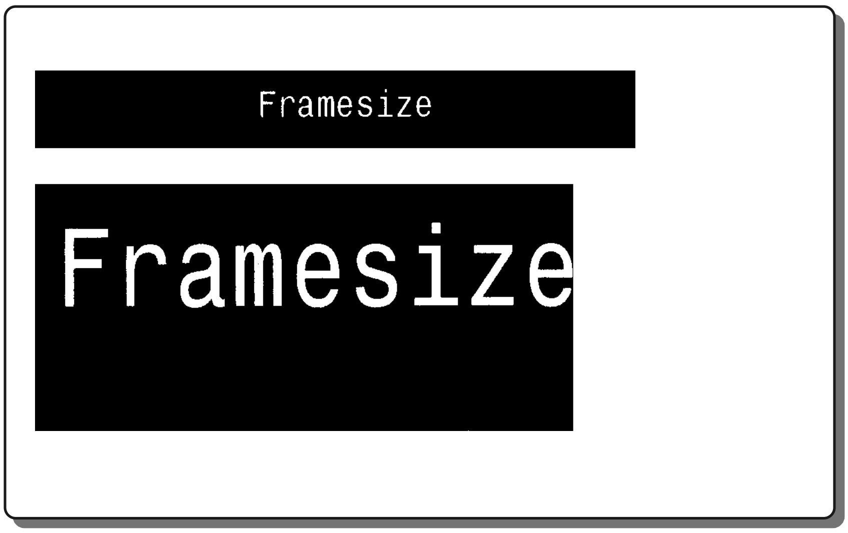


T - Text Field Definition

Sample for printing inverted text with different frame sizes. Please have a closer view how the Justification commmand (... [J:c80] ...) influences the printout.

Example:

```
J
O R
H100,-5
S l1;0,0,68,70,100
T:F1;10,40,0,596,15,n,q85,b,fu17,fd17,fl3,fr1;Framesize
T:F2;10,15,0,596,5,n,q85,b,fu6,fd4,fl3,fr3;[J:c80]Framesize
A1
```

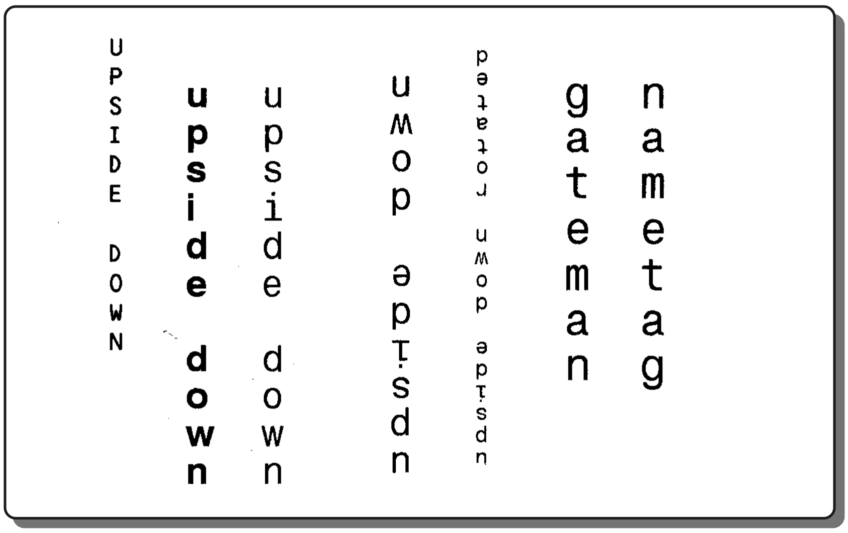


T - Text Field Definition

Writing upside down is as well possible as rotating text.

Example:

```
m m
J
S 0,0,68,71,100
T 10, 7,0,-5,x1,y1,v;upside down
T 20,14,0,5,5,v;upside down
T 30,14,0,596,5,v;upside down
T 50,59,180,596,5,v;upside down
T 60,59,180,596,3,v;upside down rotated
T 70,14,00,596,6,v;gateman
T 80,14,00,596,6,v;nametag
A 1
```



X - Synchronous Peripheral Signal Settings

The **X** command can be used to control external devices through the interface in the front of the printer.

Syntax:

```
X y[;ao] CR
```

X - Synchronous Peripheral Signal Setting Command		
y	=	Printing coordinate when a signal should be set. Distance from print start to start of the signal in millimeters or inches. (See the " m " command for the measurement settings.)
ao	=	hex nibbles to set or to reset the signal. The a -value is an AND-mask - while the o-value is an OR-mask. Both values are hex nibbles, written together as a hex byte. These values can be used to set or to reset the peripheral signal. If the ao operand is omitted entirely, the item is cleared from the internal list.

Function and settings depend on the used printer type and the peripheral connector. Please refer to the operator´s manual and to the documentation for the optional devices for each printer model.
Note: The list of positions (all signal settings) is cleared when starting a new job.

The „X“ command needs to be placed after the definition of the page size ! („S“- command)

Example:

```
X 14;E0
```

Clears bit 0 when the printhead reaches the defined position 14 mm from beginning of the label.

Special Content fields

Special content fields are defined in squared brackets [ ]. This brackets can be used in regular text field, as long as they do not include a special content field command.
Special content fields consist of reserved words, special phrases or special parameters.
cab printers will interpret this fields as a special command instead of printing these as text values.
Special content fields offer the most powerful functions in JScript.
In the following description optional parameters are shown in these brackets { }.
The following examples will help you to understand the functions of special content fields.
It is possible to link values, but it is not allowed to insert an option into another option:

Possible:

Example: ✓

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

Not possible !!!

Example: ✗

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

Values must be clearly defined to avoid that the JScript interpreter gets into „trouble“

Possible:

Example: ✓

```
J
S l1;0,0,68,71,100
T 12,30,0,3,7;[ISODATE]
T 13,55,0,3,7;[ISODATE:5,2,11]
A1
```

Not possible !!!

Example: ✗

```
J
S l1;0,0,68,71,100
T:VALUE1; 12,30,0,3,7;15[I]
T 12,55,0,3,7;[ISODATE:+VALUE1]   *
A1
```

* This expression would work properly when the plus sign is not used:

```
T 12,55,0,3,7;[ISODATE:VALUE1]
```

Time functions

Time functions are used to recall the time from the internal real time clock which is available in each printer. Additional time calculations allow to modify the time stamp with added or subtracted hours, minutes or seconds.

Please remember that it is possible to connect the printers with a time server to get the fully accuracy of time and date.

[H12]	Print Hour in 12-hour form (1-12)
[H24]	Print Hour in 24-hour form (0-23)
[H012]	Print H0ur in 12-hour form (01-12) -always 2 digits
[H024]	Print H0ur in 24-hour form (01-24) -always 2 digits
[ISOTIME]	Prints the Time in ISO standard format
[MIN]	Print MINutes (00-59)
[SEC]	Print SEConds (00-59)
[TIME]	Print actual TIME in the format of the preset country
[XM]	am / pm indicator

[H12...] Print Hour in 12-hour form (1-12)

This option is used to recall the time from the printer´s internal clock. The result will be the actual hour on the label in the 12 hour format. Usually this option is used together with the options [MM] and [SS] . The single digits (1 to 9) are printed without leading zeroes.

Syntax:

```
[H12{:+HH{,+MM{,+SS}}}]
```

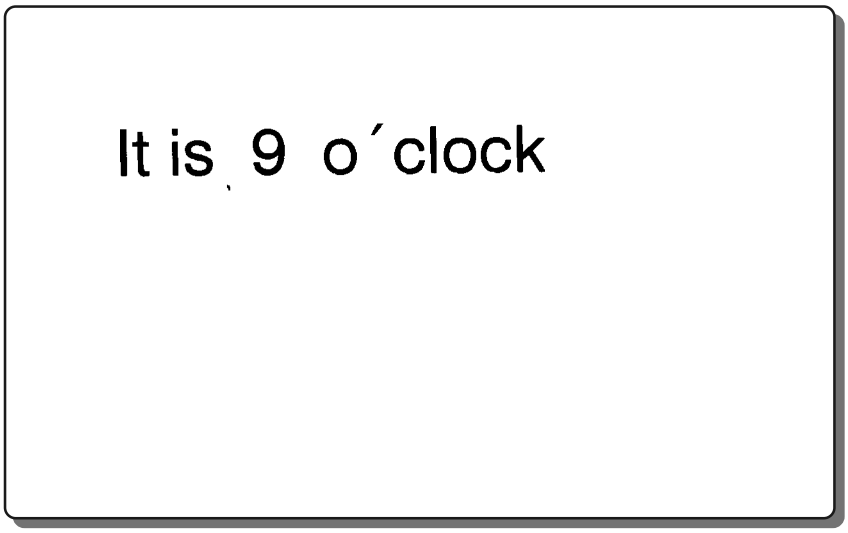
[H12...] - Print hour in 12-hour form (1-12)

+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,25,0,3,9;It is [H12]  o´clock
A1
```

Here we do not know if it is 9 o´clock in the morning or in the evening. This option should be used with the **[XM]** option (please see there for more details).



[H24...] Print Hour in 24-hour form (0-23)

This option is used to recall the time from the printer´s internal clock. The result will be the actual hour on the label in the 24 hour format. Usually this option is used together with the options [MM] and [SS] .The single digits (1..9) are printed without leading zeroes.

Syntax:

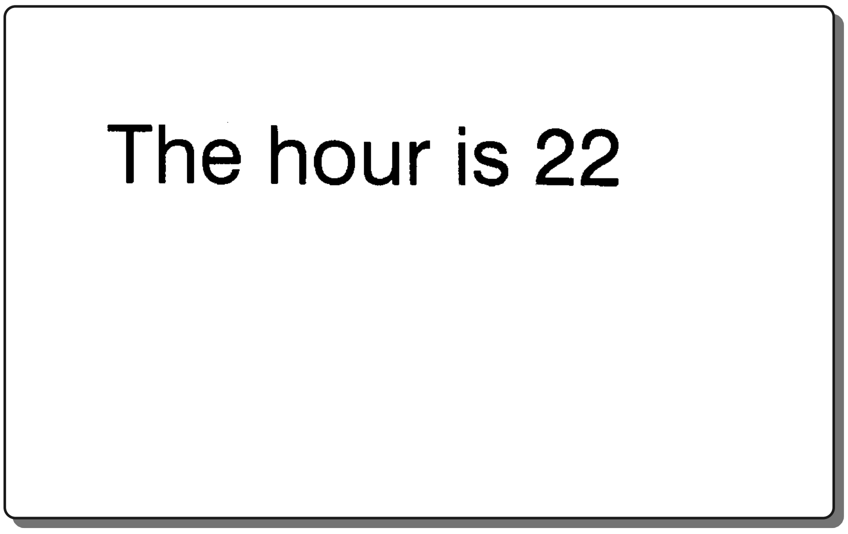
```
[H24{:+HH{,+MM{,+SS}}}]
```

[H24...] - Print hour in 24-hour form

+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,25,0,3,9;The hour is [H24]
A1
```



[H012...] Print H0ur in 12-hour form (01-12) -always 2 digits

This option is used to recall the time from the printer´s internal clock. The result will be the actual hour on the label in the 12 hour format. Usually this option is used together with the options [MM] and [SS] .The „single"digits (1 to 9) will always print with leading zeroes (01 to 09).

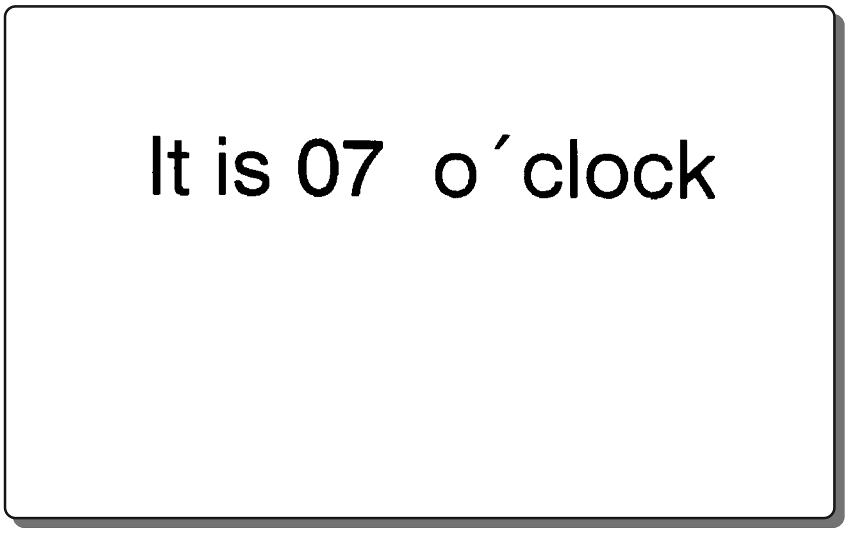
Syntax:

```
[H012{:+HH{,+MM{,+SS}}}]
```

[H012...] - Print Hour in 12-hour format (0-12) -always 2 digits		
+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

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



[H024...] Print H0ur in 24-hour form (01-24) -always 2 digits

This option is used to recall the time from the printer´s internal clock. The result will be the actual hour on the label in the 24 hour format. Usually this option is used together with the options [MM] and [SS]. The „single“digits (1 to 9) will always print with leading zeroes (01 to 09).

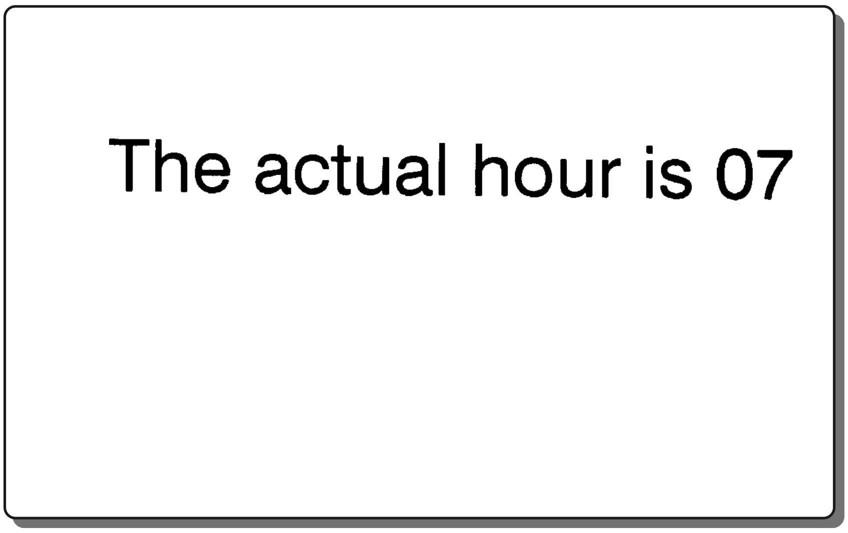
Syntax:

```
[H024{:+HH{,+MM{,+SS}}}]
```

[H024...] - Print hour in 24-hour form (01-24)always 2 digits		
+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

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



[ISOTIME...] Prints the Time in ISO standard format

[ISOTIME] prints the time in ISO format - as 6 digit value without separator sign.

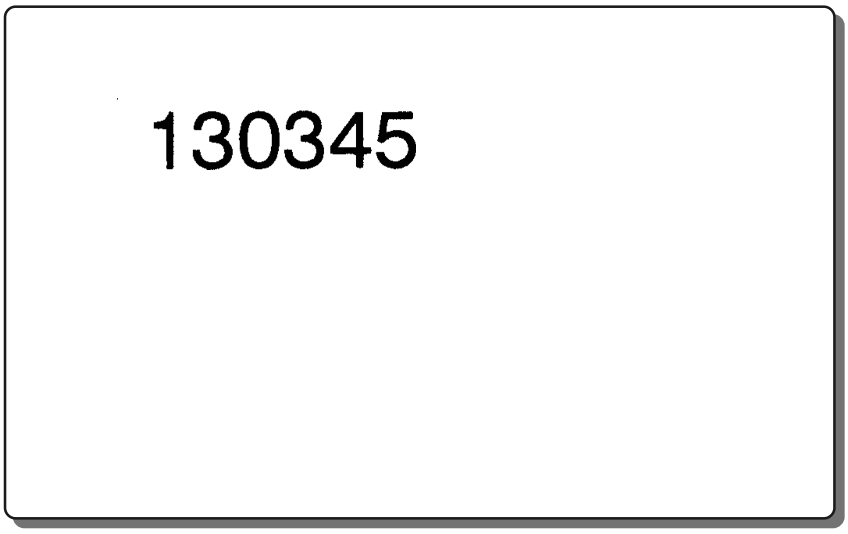
Syntax:

```
[ISOTIME{:+HH{,+MM{,+SS}}}]
```

[ISOTIME...] - Prints the time in ISO standard format		
+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,25,0,3,9;[ISOTIME]
A1
```



[MIN] Print MINutes (00-59)

This option is used to recall the actual minutes from the printer´s internal clock. Usually this option is used together with the options [HH] and [SS] .

Syntax:

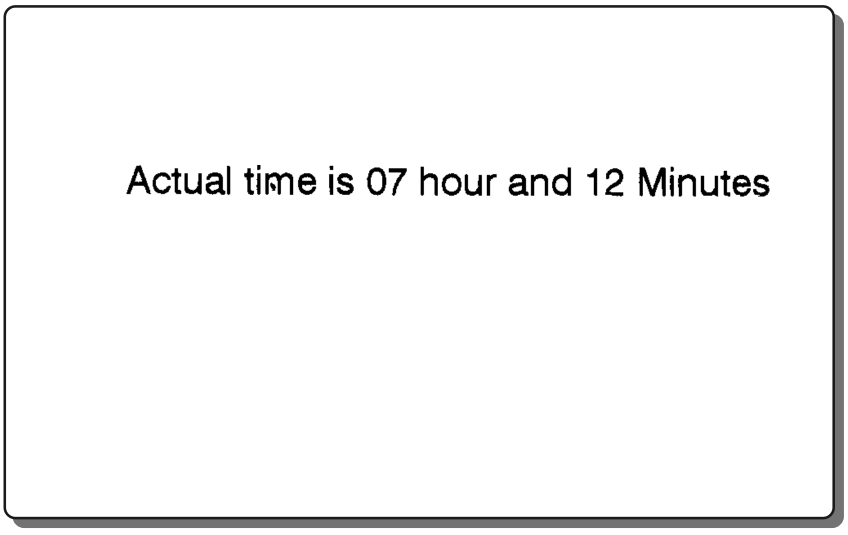
```
[MIN{:+HH{,+MM{,+SS}}}]
```

[MIN...] - print minutes

+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,25,0,3,4;Actual time is [H024] hour and [MIN] Minutes
A1
```



[SEC...] Print SEConds (00-59)

This option is used to recall the actual seconds from the printer´s internal clock. Usually this option is used together with the options [HH] and [MM].

Syntax:

```
[SEC{:+HH{,+MM{,+SS}}}]
```

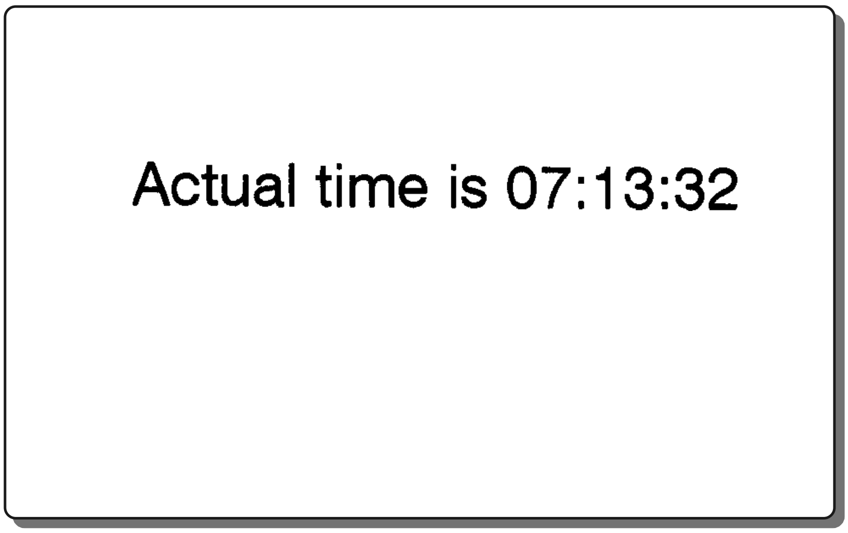
[SEC...] - Print seconds

+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

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

In this example the result is identical to the TIME option.
The difference is that the seconds can be printed separately.



[TIME ...] Print actual TIME

The time option prints the actual time in the format of the preset country.
Format: HH:MM:SS

Syntax:

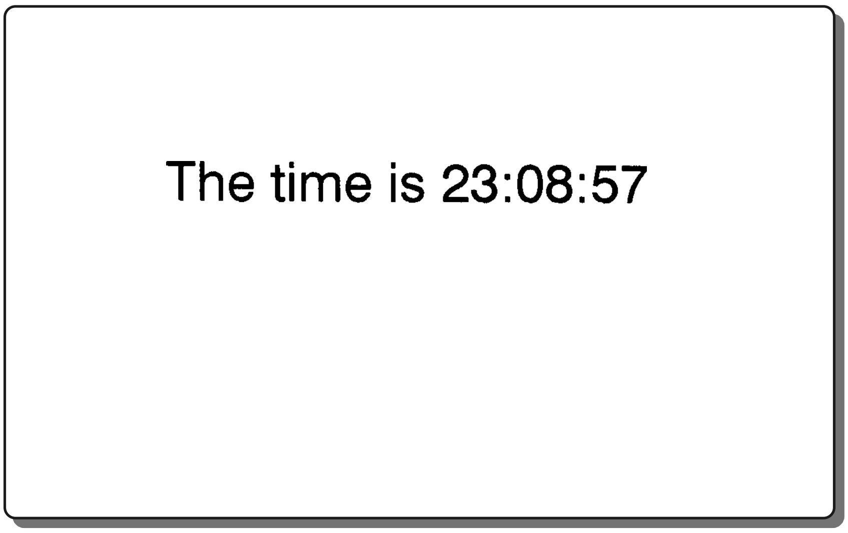
```
[TIME{:+HH{,+MM{,+SS}}}]
```

[TIME...] - print actual time		
+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

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

This example prints one label with the timestamp. The printer has been set to „country= United kingdom". The same result will be printed if the parameters would be sent in this way, separated by colons. [HH]:[MM]:[SS]



[XM...] am/pm indicator

This option was implemented for the usage in countries, where the time is displayed as „am“ (morning) and „pm“ (afternoon), when 12 hour time format is selected.

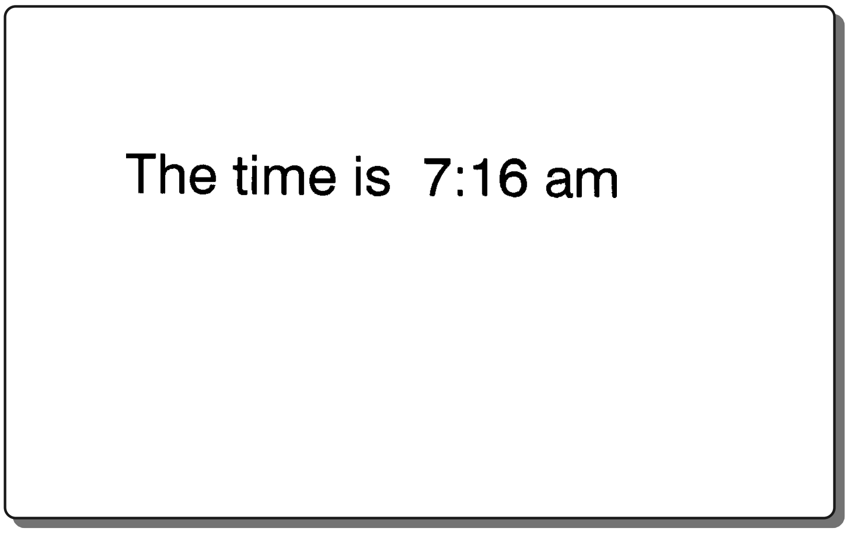
Syntax:

```
[XM{:+HH{,+MM{,+SS}}}]
```

[XM...] - am/pm indicator		
+HH	=	adds the amount of additional hours as numerical value
+MM	=	adds the amount of additional minutes as numerical value
+SS	=	adds the amount of additional seconds as numerical value

Example:

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



Date functions

Date functions are used to recall the date from the internal real time clock which is available in each printer. Additional date calculation options allow to modify the date stamp with added or subtracted days, months or years, i. e. to calculate "best before" dates.

Special note:The printers calculate months always as 30 days.
Please remember that it is possible to connect the printers with a time server to get the fully accuracy of time and date.

[DATE{:+DD{,+MM{,+YY}}}]	Print actual DATE in the format of the preset country
[DAY{:+DD{,+MM{,+YY}}}]	Print numeric DAY of the month (1-31)
[DAY02{:+DD{,+MM{,+YY}}}]	Print numeric 2-digit DAY of the month (01-31)
[DOFY{:+DD{,+MM{,+YY}}}]	Print numeric Day OF Year(1-366)
[ISODATE{:+DD{,+MM{,+YY}}}]	Print ISO date
[ISOORDINAL{:+DD{,+MM{,+YY}}}]	Print ISO ordinal
[ODATE:+DD{,+MM{,+YY}}]	Print DATE with Offset (in the format of the preset country)
[wday{:+DD{,+MM{,+YY}}}]	Print complete weekday name (0 = sunday)
[WDAY{:+DD{,+MM{,+YY}}}]	Print numeric WeekDAY(0-6)
[wday2{:+DD{,+MM{,+YY}}}]	Print weekday name, 2 - digits shortened (i.e. su)
[wday3{:+DD{,+MM{,+YY}}}]	Print weekday name, 3 - digits shortened (i.e. sun)
[ISOWDAY{:+DD{,+MM{,+YY}}}]	Print numeric WeekDAY(1-7)
[WEEK{:+DD{,+MM{,+YY}}}]	Print numeric WEEK (1-53)
[WEEK02{:+DD{,+MM{,+YY}}}]	Print numeric WEEK with 2 -digits (01-53)
[OWEEK:+WW]	Print WEEK with Offset(1-53)
[mon{:+DD{,+MM{,+YY}}}]	Print 3-character **mon**th name (i.e. jan)
[month{:+DD{,+MM{,+YY}}}]	Print complete **month** name (i.e. january)
[MONTH{:+DD{,+MM{,+YY}}}]	Print **2**-digit **MONTH** (1-12)
[MONTH02{:+DD{,+MM{,+YY}}}]	Print **02**-digit **MONTH** (01-12) (leading zeros, always 2 digits)
[YY{:+DD{,+MM{,+YY}}}]	Print **2**-digit **Y**ear (00-99)
[YYYY{:+DD{,+MM{,+YY}}}]	Print **4**-digit **Y**ear (1970-2069)

[DATE...] Print actual DATE

Recalls the date from the printer and prints it in the defined size and in the format of the selected country.

Syntax: `[DATE{:+DD{,+MM{,+YY}}}]`

[DATE...] - print actual date	
+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Alternative it is possible to use a variable to add additional days, months or years

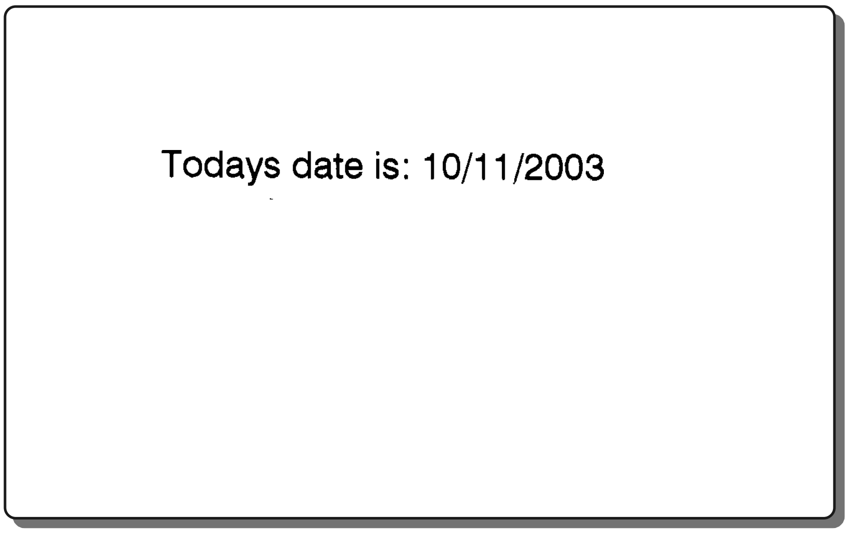
Syntax: `[DATE{:VARIABLE}]`

+VARIABLE = adds the value of a predefined variable as numerical value

IMPORTANT NOTE: In that case when variables are used, it is not allowed to use the „+“ sign !!

Example:

```
;This example simply recalls the date from the printer
 m m
J
S l1;0,0,68,71,100
T 12,25,0,3,5;Todays date is: [DATE]
A1
```

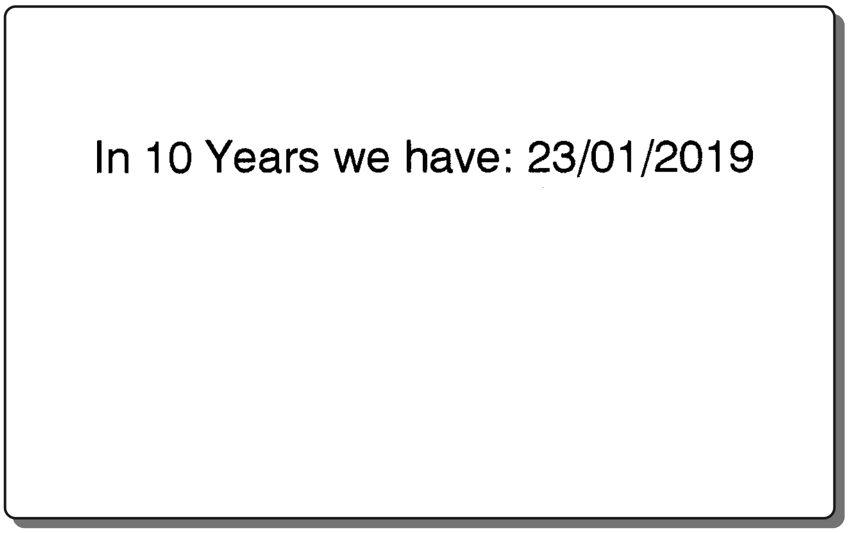


[DATE...] Print actual DATE

Example:

```
m m
J
S l1;0,0,68,71,100
T 3,25,0,3,6;In 10 Years we have: [DATE:03,02,10]
A1
```

This example adds 3 days, 2 months and 10 years


In 10 Years we have: 23/01/2019

[DAY...] Print numeric DAY of the month (1-31)

The numeric day of the actual month is recalled from the printer´s clock

Syntax:

```
[DAY{:+DD{,+MM{,+YY}}}]
```

[DAY...] - print numeric day of the month (1-31)

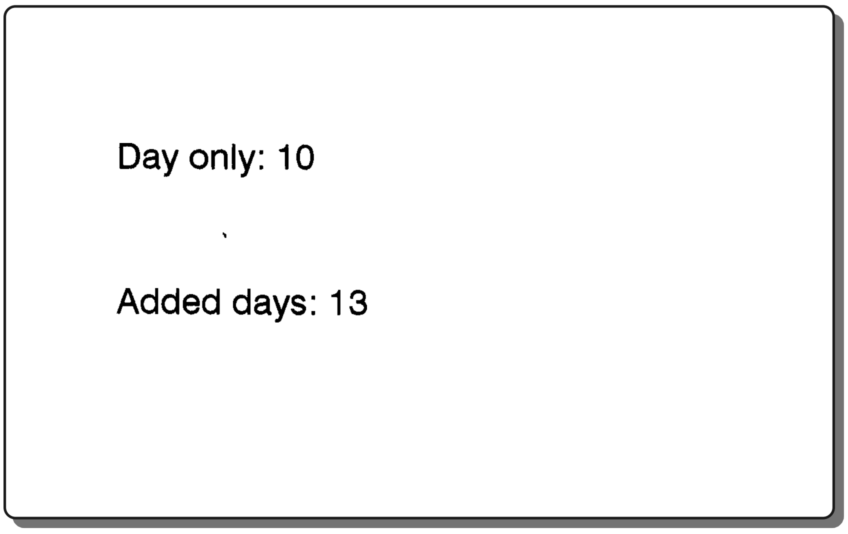
+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Syntax:

```
[DAY{:VARIABLE}]
```

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,25,0,3,5;Day only: [DAY]
T 12,45,0,3,5;Added days: [DAY:03,02,10]
A1
```



Day only: 10

Added days: 13

[DAY02...] Print numeric 2-digit DAY of the month (01-31)

Recalls the date from the printer and prints it in the defined size and in the format of the selected country. (see also the“l“ command).

Syntax:

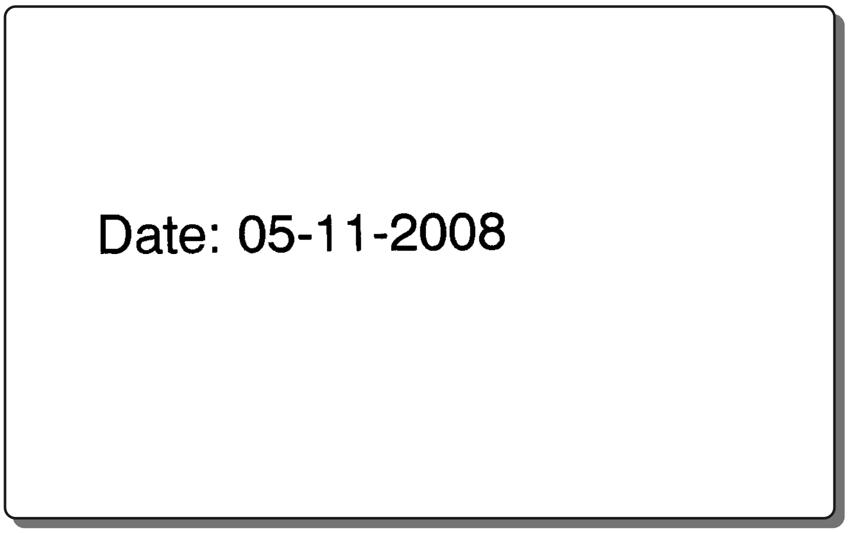
```
[DAY02{:+DD{,+MM{,+YY}}}]
```

[DAY02...] - print numeric 2-digit day of the month (01-31)	
+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Example:

```
m m
s 031105091500
J
S l1;0,0,68,71,100
T 12,30,0,3,7;Date: [DAY02]-[MONTH02]-[YYYY]
A1
```

Prints a label where the day is displayed with 2 digits



[DOFY...] Print numeric Day OF Year(001-366)

Prints the Day of Year. Possible values: 001-366.

Syntax:

```
[DOFY{:+DD{,+MM{,+YY}}}]
```

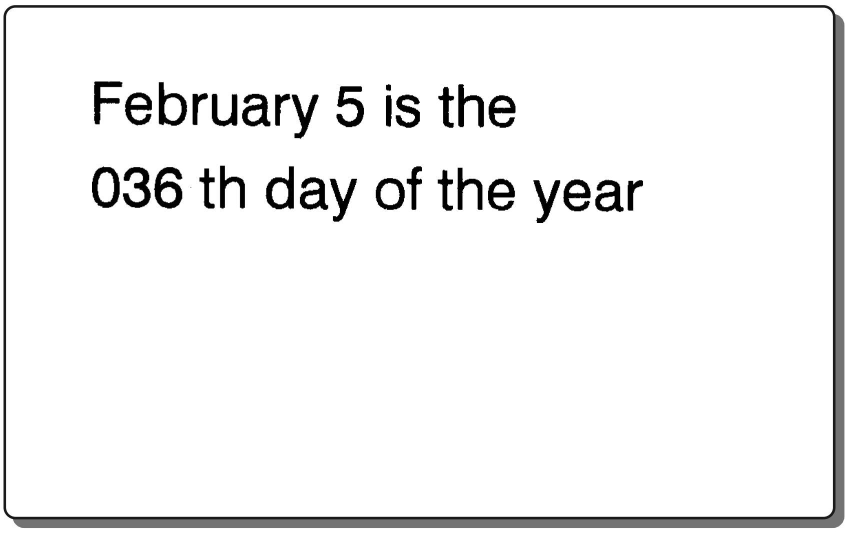
[DOFY...] - print numeric day of the year

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
s 090205091500
J
S l1;0,0,68,71,100
T 12,20,0,3,7;February 5 is the
T 12,30,0,3,7;[DOFY] th day of the year
A1
```

The preset date in this example is February 5 2009. The result appears in 3 digits.


February 5 is the
036 th day of the year

[ISODATE:...] Prints date following the ISO specs

Prints the date in ISO Format, following the rules of the ISO 8601-2000 standard.
Days, months and years can be added.
The ISO date specifies the representation of dates in the Gregorian calendar. Identification of a particular calender day by its calender year, its calendar month and its ordinal number within the calendar month.

Syntax:

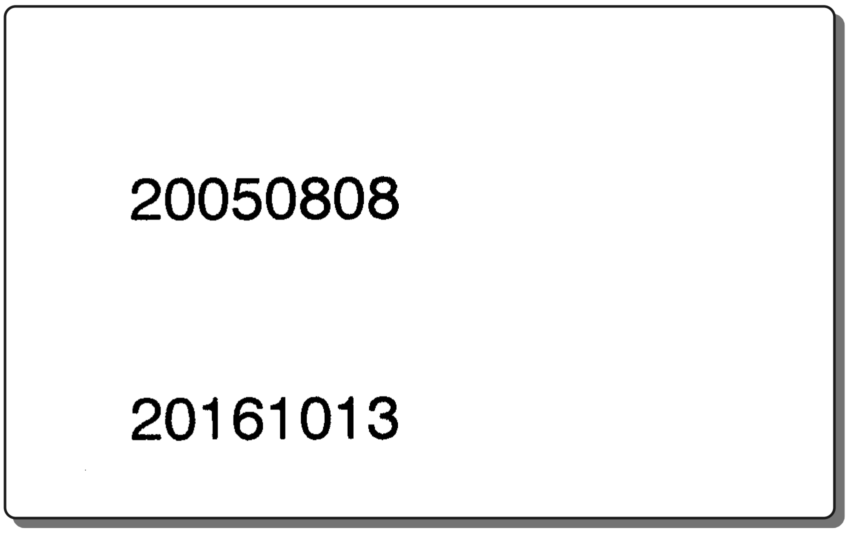
```
[ISODATE{:+DD{,+MM{,+YY}}}]
```

[ISODATE...] - prints date following the ISO specs	
+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,30,0,3,7;[ISODATE]
T 12,55,0,3,7;[ISODATE:5,2,11]
A1
```

For a detailed description, please refer to ISO standard 8601-2000.



[ISOORDINAL: ...] Prints date following the ISO specs

Prints the particular calendar day and its ordinal number within its calendar year. Result is printed in ISO 8601:2000 format (YYYYDDD) whereby YYYY stands for the 4 -digit year and DDD displays the day of the year.

Syntax:

```
[ISOORDINAL{:+DD{,+MM{,+YY}}}]
```

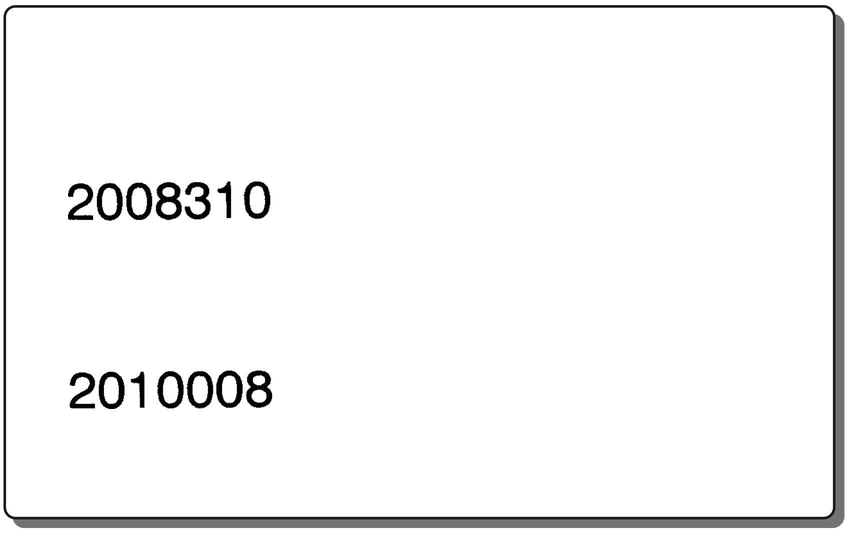
[ISOORDINAL...] - prints date following the ISO specs

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,30,0,3,7;[ISOORDINAL]
T 12,55,0,3,7;[ISOORDINAL:3,2,1]
A1
```

For detailed description, please refer to ISO standard 8601-2000.



[WDAY...] Print numeric WeekDAY(0-6)

This function prints the numeric week day - starting on sunday with 0 and ends at saturday with 6. Please see also the **[ISOWDAY]** command which numbers each weekday from 1-7, starting on monday.

Syntax:

```
[WDAY{:+DD{,+MM{,+YY}}}]
```

[WDAY...] - print numeric weekday (0-6)		
+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

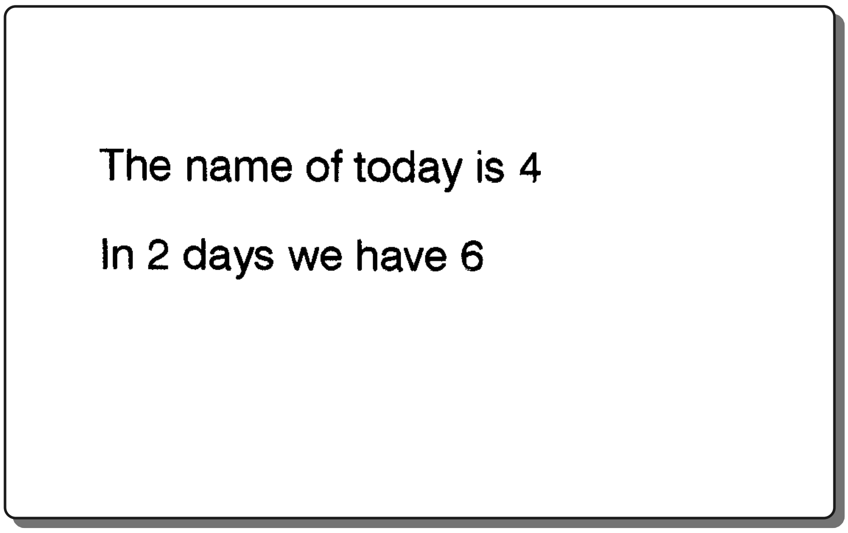
Example:

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

This is the same sample as on the previous page with the difference that we wrote „WDAY“ in capital letters.

0	= sunday	**4**	=	thursday
1	= monday	**5**	=	friday
2	= tuesday	**6**	=	saturday
3	= wednesday			

So we have Thursday today and in two days we have saturday



[wday...] Print complete weekday name

Print the complete weekday name. The name of the day depends on the selected language of the printer or on the previously sent „ **l** „ (language) command.

Syntax:

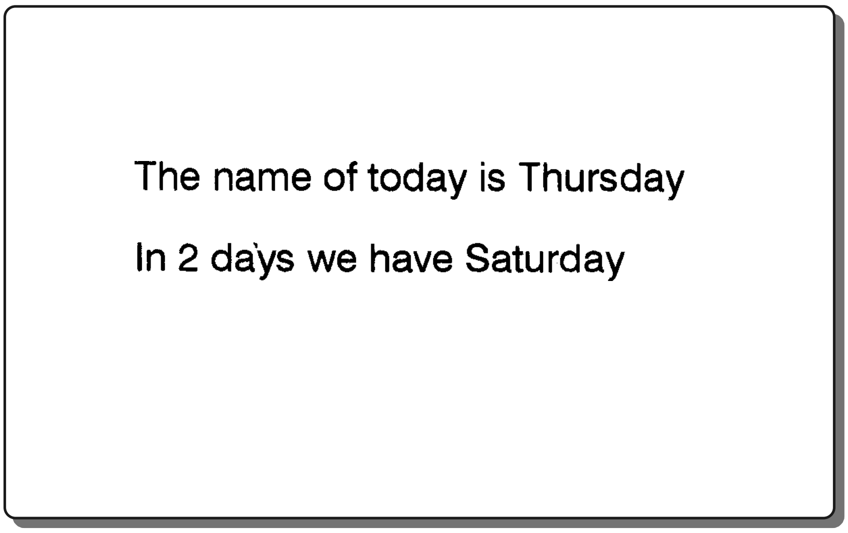
```
[wday{:+DD{,+MM{,+YY}}}]
```

[wday...] - print complete weekday name

+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Example:

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



[wday2...] Print weekday name, 2 - digits shortened

Print the first 2 characters of the weekday name. The name of the day depends on the selected language of the printer or on the previously sent „**l**“ (language) command.

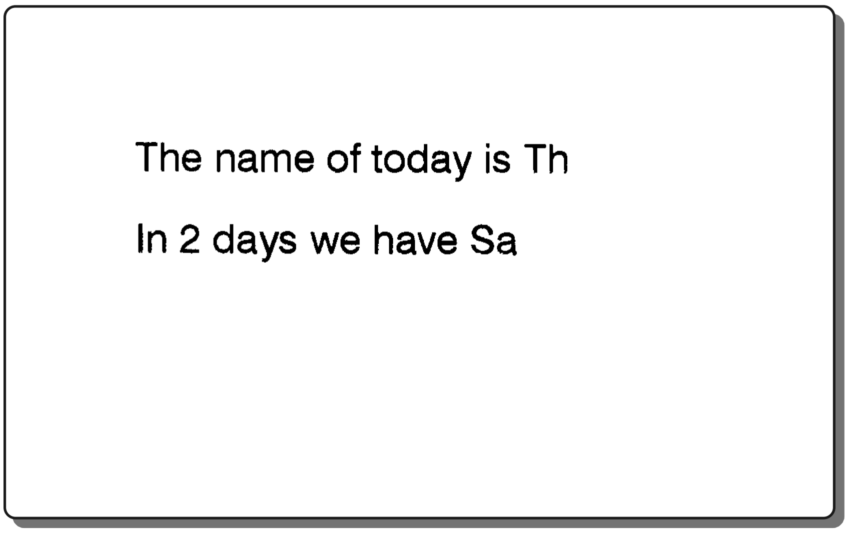
Syntax:

```
[wday2{:+DD{,+MM{,+YY}}}]
```

[wday2...] - print weekday name, 2-digits shortened		
+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

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


The name of today is Th

In 2 days we have Sa

[wday3...]   Print weekday name, 3 - digits shortened

Prints the first 3 characters of the weekday name. The name of the day depends on the preset language of the printer or on the previously sent „**l** = language“ command.

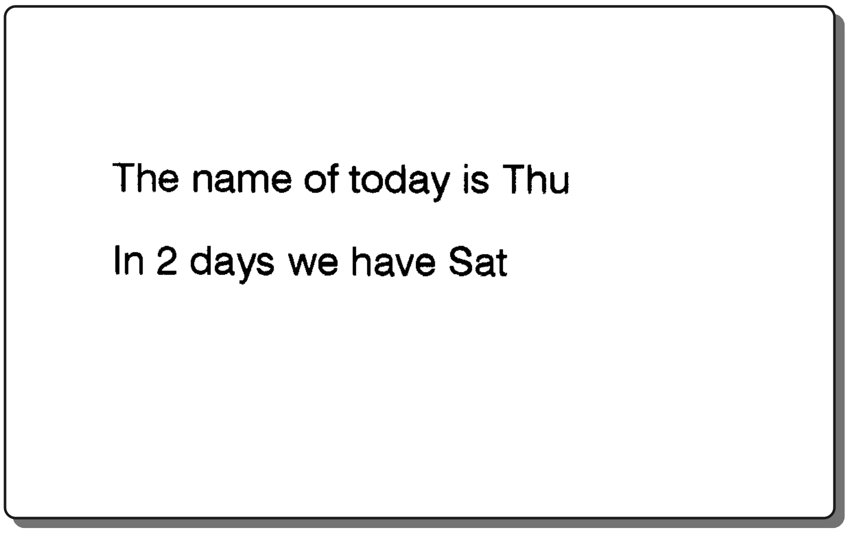
Syntax:

```
[wday3{:+DD{,+MM{,+YY}}}]
```

[wday3...] - print weekday name, 3-digits shortened		
+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,25,0,3,5;The name of today is [wday3]
T 12,35,0,3,5;In 2 days we have [wday3:02,00,00]
A1
```



[ISOWDAY: ...] Print date following the ISO specs

This function prints the numeric week day - starting on monday with 1 and it ends at sunday with 7. Please see also the **[WDAY]** command which numbers each weekday from 0-6, starting on sunday.

Syntax:

```
[ISOWDAY{:+DD{,+MM{,+YY}}}]
```

[ISOWDAY...] - print date following the ISO specifications

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

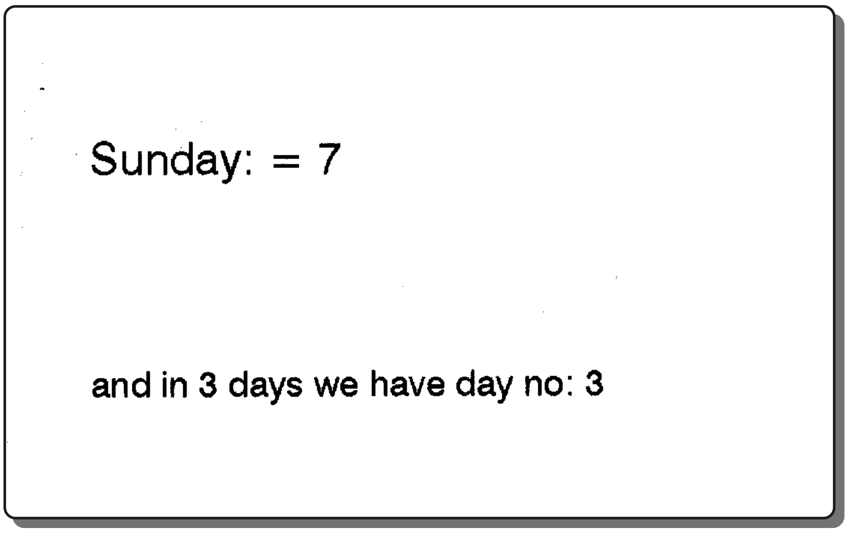
Example:

```
m m
l UK
s 060326184500
J
S l1;0,0,68,71,100
T 8,30,0,3,5;[wday]: = [ISOWDAY]
T 8,55,0,3,4;and in 3 days we have day no: [ISOWDAY:3,0,0]
A1
```

Following are the results:

1	=	monday	**4**	=	thursday	**7**	=	sunday
2	=	tuesday	**5**	=	friday			
3	=	wednesday	**6**	=	saturday			

For further information, please refer to ISO standard 8601-2000.



Sunday: = 7

and in 3 days we have day no: 3

[WEEK...] Print numeric WEEK (1-53)

Prints the week number (1 -53)The week will print without leading zeroes if a week has only one digit. The command **[WEEK02...]** needs to be used, if leading zeroes are required for the first weeks of the year.

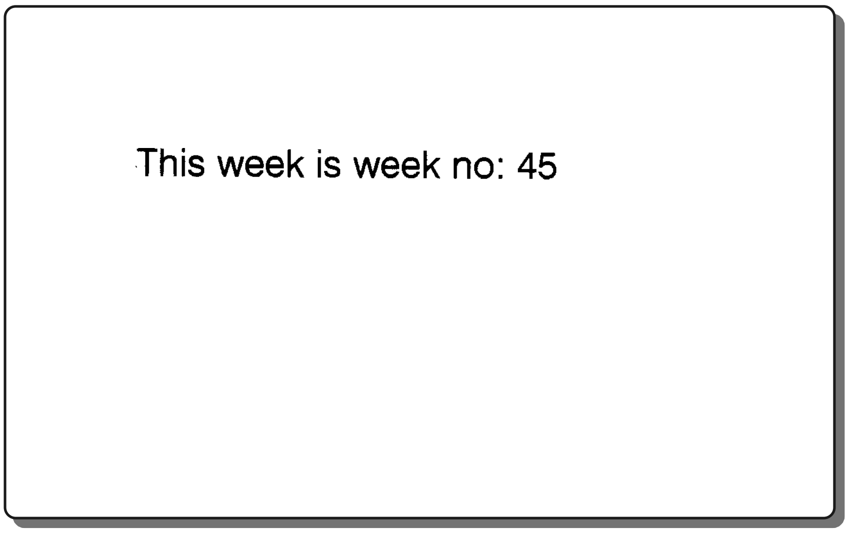
Syntax:

```
[WEEK{:+DD{,+MM{,+YY}}}]
```

[WEEK...] - print numeric week		
+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,25,0,3,5;This week is week no: [WEEK]
A1
```



[WEEK02...] Print numeric WEEK with 2 -digits (01-53)

Print the week number with 2 digits. The week will print with leading zeroes. The printer creates the number of the week (01-53)

Syntax:

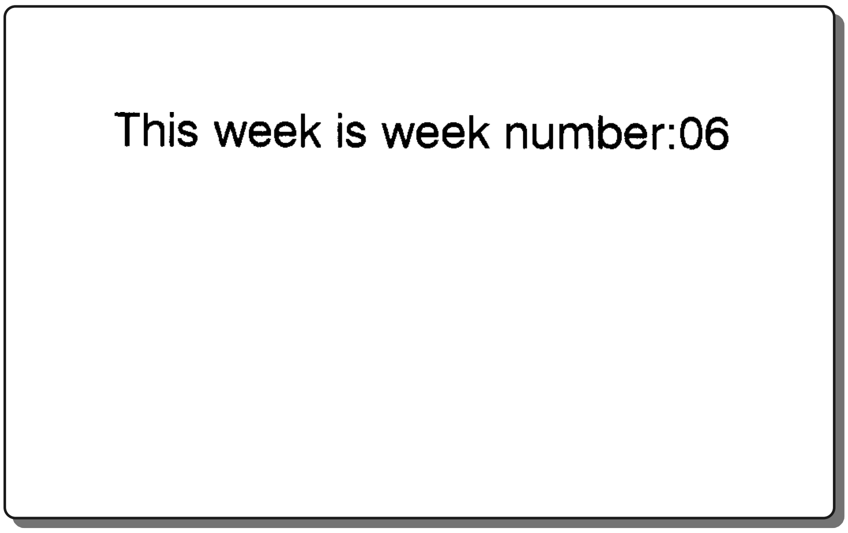
```
[WEEK02{:+DD{,+MM{,+YY}}}]
```

[WEEK02...] - print numeric week with 2 -digits (01-53)

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

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



[OWEEK...] Print WEEK with Offset(1-53)

Print week with offset (1-53)

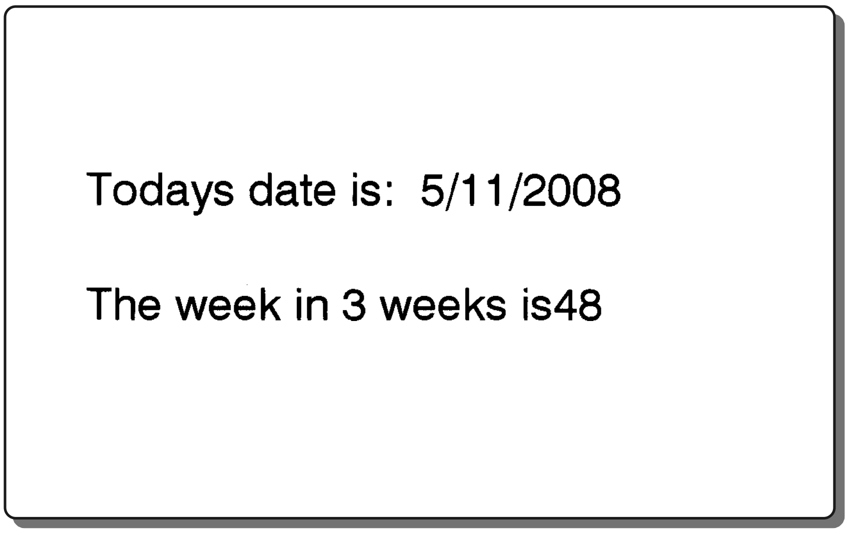
Syntax:

```
[OWEEK:+WW]
```

[OWEEK...] - print week with offset (1-53)	
+WW	= adds the amount of additional weeks as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 12,25,0,3,6;Todays date is: [DATE]
T 12,40,0,3,6;The week in 3 weeks is[OWEEK:3]
A1
```



Todays date is: 5/11/2008

The week in 3 weeks is48

[mon...] Print 3-character month name

Prints the first 3 characters of the month name. The name of the month depends on the selected language of the printer or on the previously sent „**l** = language“ command.

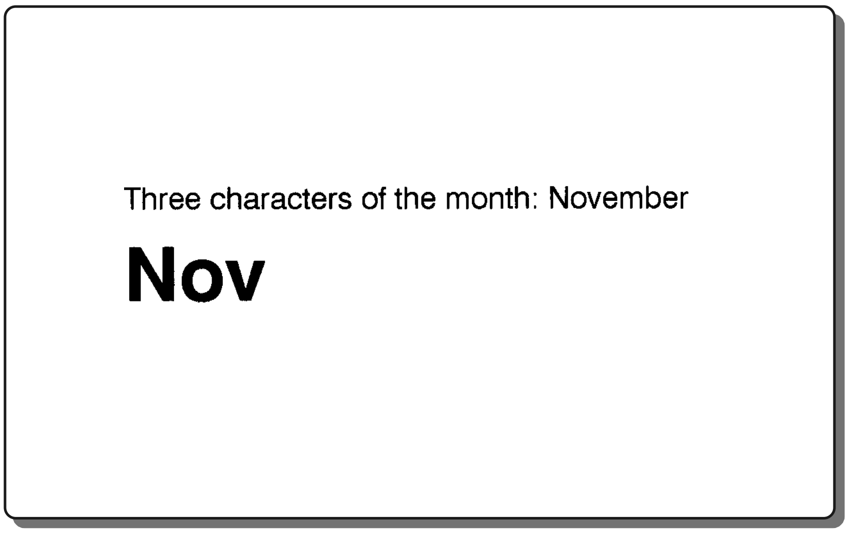
Syntax:

```
[mon{:+DD{,+MM{,+YY}}}]
```

[mon...] - print 3-character month name		
+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,28,0,3,4;Three characters of the month: [month]
T 10,40,0,5,10;[mon]
A1
```



[month...] Print complete month name

Prints the complete month name. The name of the month depends on the selected language of the printer or on the previously sent „**l** = language“ command.

Syntax: `[month{:+DD{,+MM{,+YY}}}]`

[month...] - print complete month name		
+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

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

[MONTH...] Print 2-digit MONTH (1-12)

Print digits of month. (1-12) (no leading zeroes). If leading zeroes are required, please see the command **[MONTH02...]**.

Syntax:

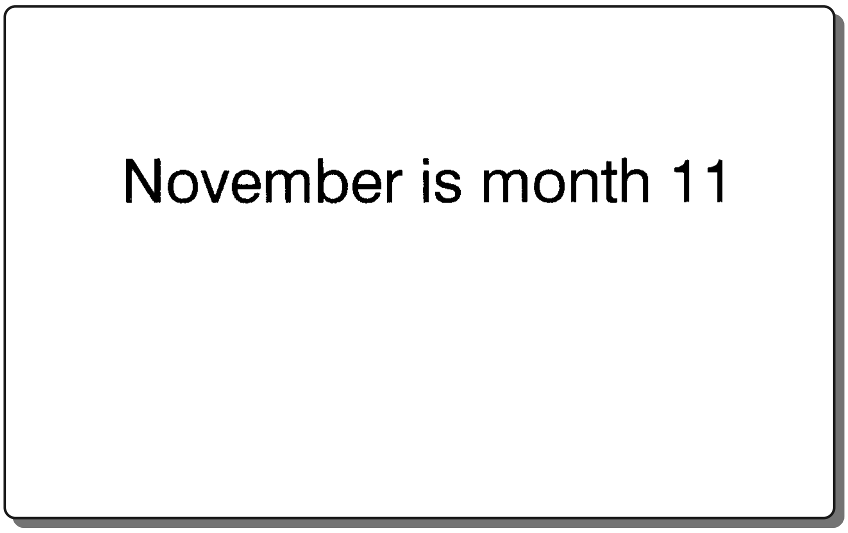
```
[MONTH{:+DD{,+MM{,+YY}}}]
```

[MONTH...] - print 2-digit month (1-12)

+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Example:

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



[MONTH02...] Print 02-digit MONTH (01-12)

Print 2 digits month. (01-12) (leading zeroes, always 2 digits). Please see the command **[MONTH...]** , if leading zeroes should be suppressd.

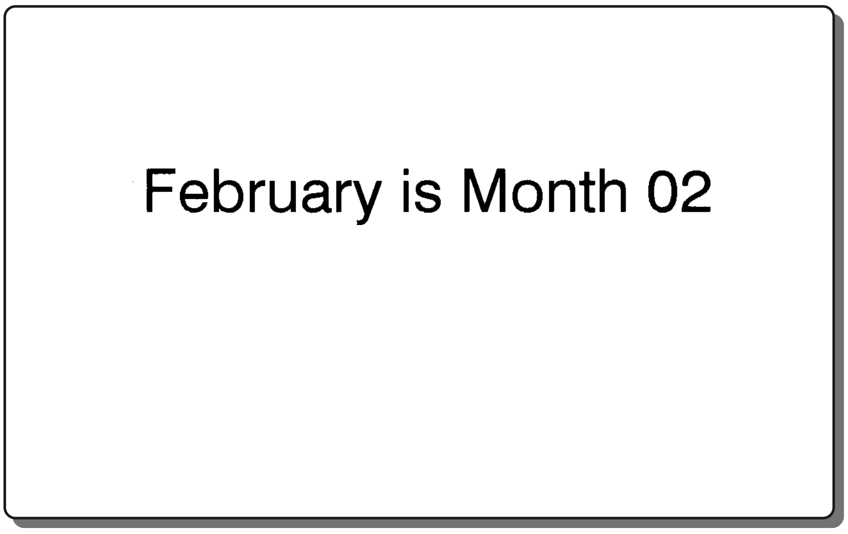
Syntax:

```
[MONTH02{:+DD{,+MM{,+YY}}}]
```

[MONTH02...] - print 02-digit month (01-12)	
+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Example:

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


February is Month 02

[MONTH02...]   Print 02-digit MONTH (01-12)

Print a ONE DIGIT MONTHCODE

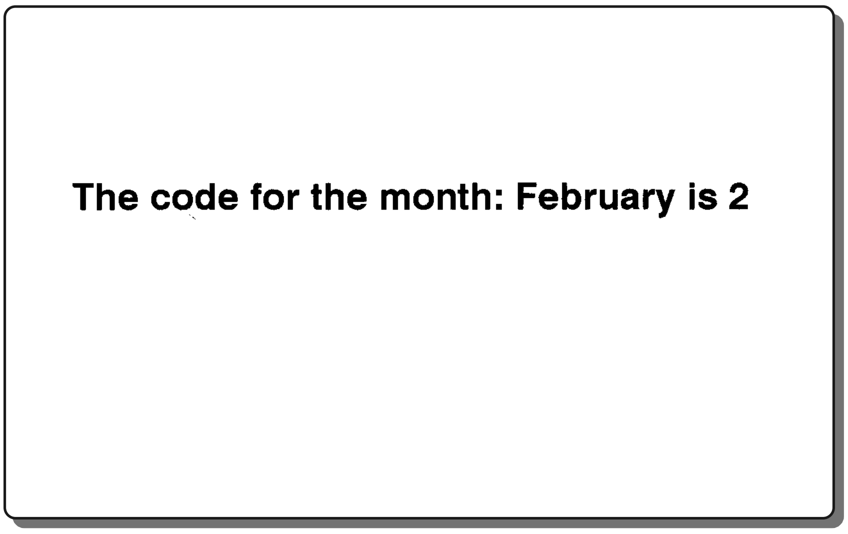
The following example creates a label with a one digit Month code 1...9 and O...D using the [MONTH02] command. This is sometimes requested for industrial applications.
The months are encoded as follows:

1...9	=> January ... September
O...D	=> October ... December

Example:

```
J
S l1;0,0,68,71,100
T:MON;5,10,0,3,4;[MONTH02][I]
T:CHAIN; 5,15,0,3,4;123456789OND[I]
T 0,30,0,5,5;The code for the month: [month] is [CHAIN,MON,1]
A 1
```

Please note, that the printed month name ([month])in this example depends on the language settings of the printer.



[YY...] Print 2-digit Year (00-99)

Print 2 digits year. (0-99) (leading zeroes, always 2 digits)

Syntax:

```
[YY{:+DD{,+MM{,+YY}}}]
```

[YY...] - print 2-digit year

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

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

February-08

[YYYY...] Print 4-digit Year (1970-2069)

Print 4 digits year. (1970-2069)

Syntax:

```
[YYYY{:+DD{,+MM{,+YY}}}]
```

[YYYY...] - print 4-digit year (1979-2069)

+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Example:

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

February-2008

Jalali Date functions

The Jalali Calender is used in Arab countries. The date calculation is similar to the other date commands, with the difference that the Jalali calendar is used for the date calculation which delivers other results. The handling of these functions is identical.

[JYEAR{:+DD{,+MM{,+YY}}}**]**	**Print** **J**alali-**YEAR**, 4 digits
[JDAY{:+DD{,+MM{,+YY}}}**]**	**Print** **J**alali-**DAY**
[JDAY02{:+DD{,+MM{,+YY}}}**]**	**Print** **J**alali-**DAY**, 02 digits
[JMONTH{:+DD{,+MM{,+YY}}}**]**	**Print** **J**alali-**Month**
[JMONTH02{:+DD{,+MM{,+YY}}}**]**	**Print** **J**alali-**Month**,02 digits
[jmonth{:+DD{,+MM{,+YY}}}**]**	**Print** **J**alali-**Month**, complete name
[JDOFY{:+DD{,+MM{,+YY}}}**]**	**Print** **J**alali-**D**ay **OF** **Y**ear
[JWDAY{:+DD{,+MM{,+YY}}}**]**	**Print** **J**alali-**DAY** of the **W**eek (1=saturday)

The printers need to be set up for an arabic characters (Farsi) language to get the expected result.

Suriyakati Date

[SYEAR{:+DD{,+MM{,+YY}}}**]**	**Print** **S**uriyakati-**YEAR**, 4 digits

[JYEAR...] Print 4-digit Jalali Year

Print 4 digits year, based on the Jalali calendar.
The output of this date can be influenced with the [S:...] command to print the numbers either in arabic or in latin style.

Syntax:

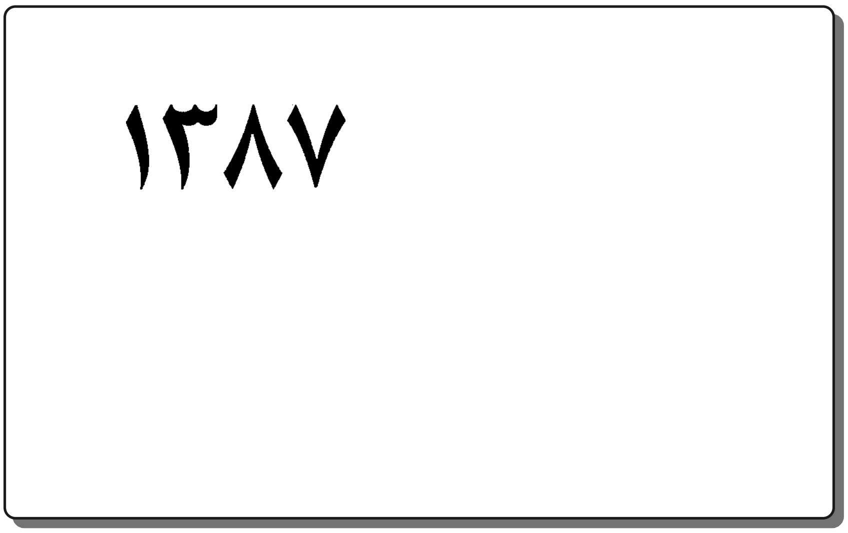
```
[JYEAR{:+DD{,+MM{,+YY}}}]
```

[JYEAR...] - print 4-digit Jalali year

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,3,20;[JYEAR][S:arabic]
A1
```



[JDAY...] Print Jalali-DAY

Prints the day in Jalali calender format.
The output of this date can be influenced with the [S:...] command to print the numbers either in arabic or in latin style.

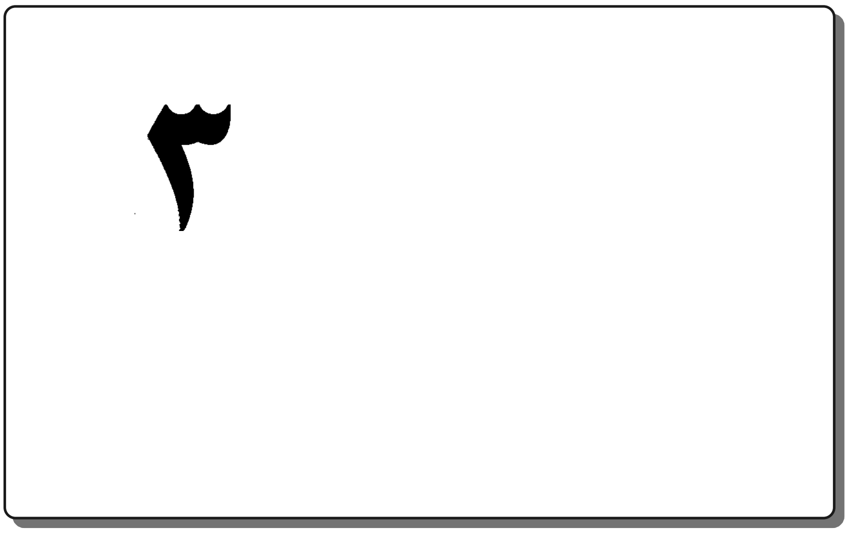
Syntax:

```
[JDAY{:+DD{,+MM{,+YY}}}]
```

[JDAY...] - print jalali-day	
+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,5,30;[JDAY][S:arabic]
A1
```



[JDAY02...] Print Jalali-DAY, 02 digits

Prints the first 2 characters of the day of the Jalali calendar.
The output of this date can be influenced with the [S:...] command to print the numbers either in arabic or in latin style.

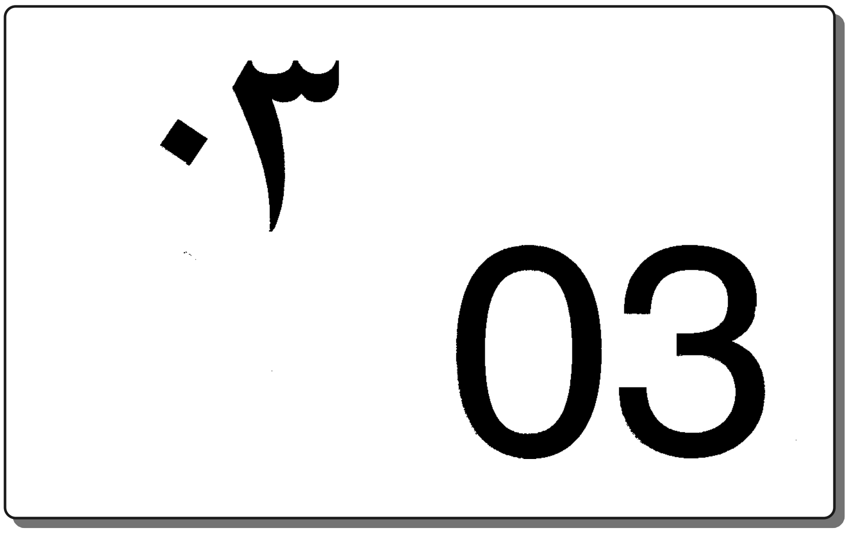
Syntax:

```
[JDAY02{:+DD{,+MM{,+YY}}}]
```

[JDAY02...] - print jalali-day, 02 digits	
+DD	= adds the amount of additional days as numerical value
+MM	= adds the amount of additional months as numerical value
+YY	= adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,3,40;[JDAY02][S:arabic]
T 50,60,0,3,40;[JDAY02]
A1
```



[JMONTH...] Print Jalali-Month

Prints the Jalali month.
The output of this date can be influenced with the [S:...] command to print the numbers either in arabic or in latin style.

Syntax:

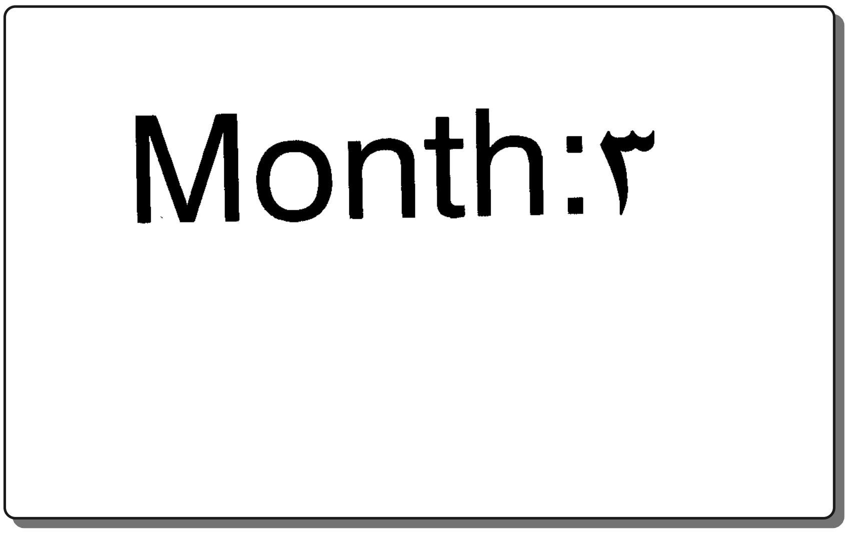
```
[JMONTH{:+DD{,+MM{,+YY}}}]
```

[JMONTH...] - print Jalali Month

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,3,20;Month:[JMONTH][S:arabic]
A1
```



[JMONTH02...] Print Jalali-Month - 2 digits

Print Jalali-Month,02 digits
The output of this date can be influenced with the **[S:...]** command to print the numbers either in arabic or in latin style.

Syntax:

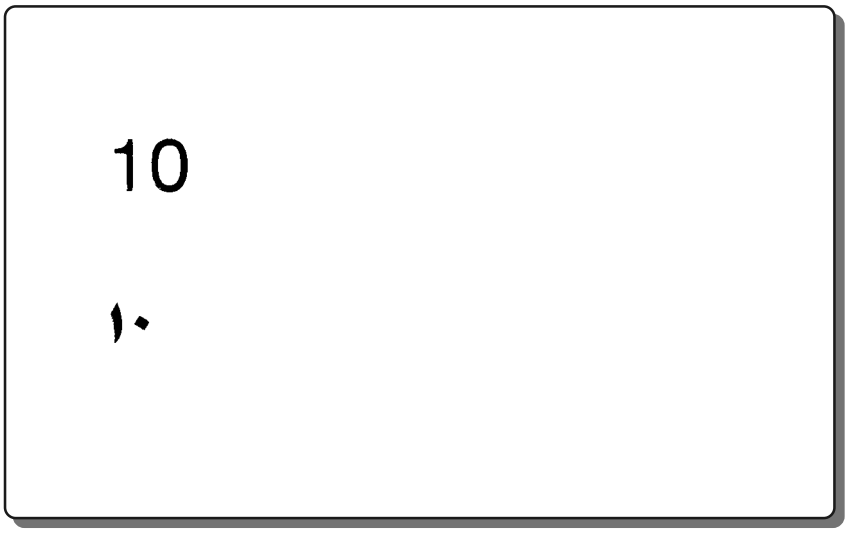
```
[JMONTH02{:+DD{,+MM{,+YY}}}]
```

[JMONTH02...] - print Jalali month 2 - digits

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,3,10;[JMONTH02]
T 10,50,0,5,10;[JMONTH02][S:arabic]
A1
```



[JDOFY...] Print Jalali-Day OF Year

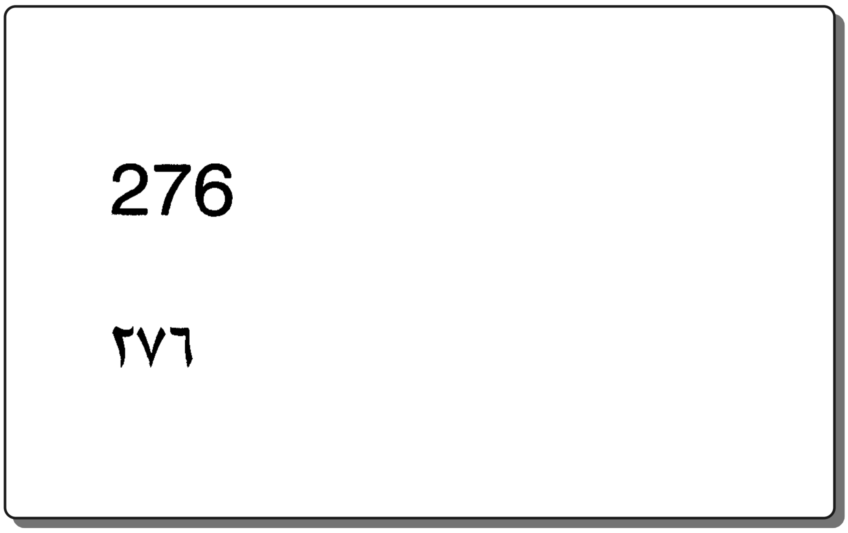
Prints the day of the year in the Jalali calendar format.
The output of this date can be influenced with the [S:...] command to print the numbers either in arabic or in latin style.

Syntax:	`[JDOFY{:+DD{,+MM{,+YY}}}]`

[JDOFY...] - Print Jalali-day of year		
+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,3,10;[JDOFY]
T 10,50,0,3,10;[JDOFY][S:arabic]
A1
```



[jmonth...] Print complete Jalali month name

Prints the complete month name. The name of the month depends on the selected language of the printer or on the previously sent „**l** = language“ command.

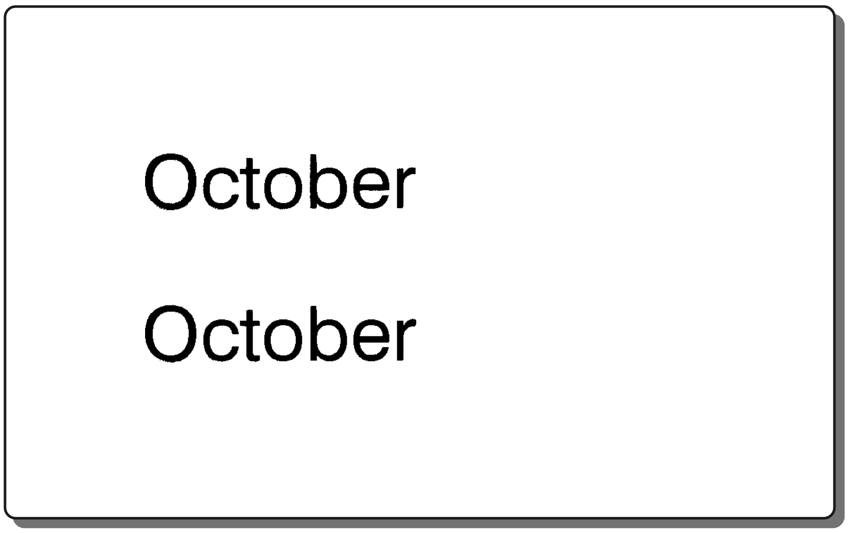
Syntax:

```
[jmonth{:+DD{,+MM{,+YY}}}]
```

[jmonth...] - print complete Jalali month name		
+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,3,10;[jmonth][S:arabic]
T 10,50,0,3,10;[jmonth]
A1
```



[JWDAY...] Print Jalali-Week-DAY

Prints the Week day of the Jalali calendar. The output of this date can be influenced with the [S:...] command to print the numbers either in arabic or in latin style.

Syntax:

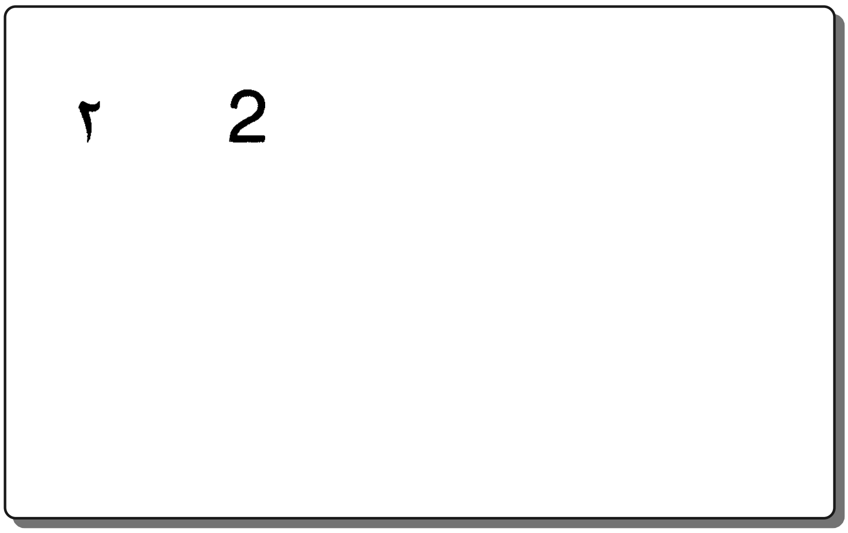
```
[JWDAY{:+DD{,+MM{,+YY}}}]
```

[JWDAY{:+DD{,+MM{,+YY}}}] - print Jalali week day

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,3,10;[JWDAY][S:arabic]
T 30,30,0,3,10;[JWDAY]
A1
```



[SYEAR...] Print 4-digit Suriyakati Year

Print 4 digits year,based on the Suriyakati calendar. The Suriyakati calendar (also called sun calendar or Buddha calendar) is the official calendar in Thailand.

Syntax:

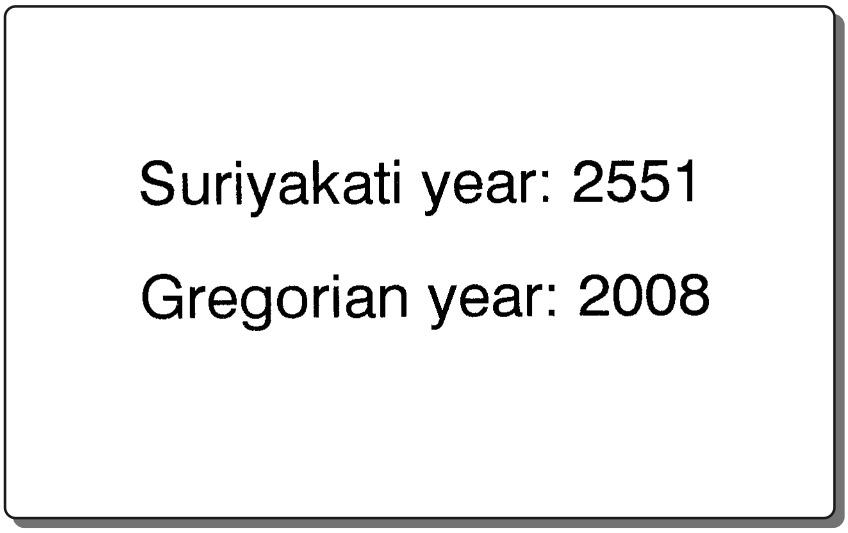
```
[SYEAR{:+DD{,+MM{,+YY}}}]
```

[SYEAR...] - print a 4-digit Suriyakati Year

+DD	=	adds the amount of additional days as numerical value
+MM	=	adds the amount of additional months as numerical value
+YY	=	adds the amount of additional years as numerical value

Example:

```
m m
J
S l1;0,0,68,71,100
T 10,30,0,3,8;Suriyakati year: [SYEAR]
T 10,45,0,3,8;Gregorian year: [YYYY]
A1
```



Suriyakati year: 2551

Gregorian year: 2008

Mathematical functions

The printer offer very powerful mathematical functions for calculation and comparison of different field values.

Mathematical functions
Field Calculations and Comparisons

[+:op1,op2. . ,]	Addition
[-:op1,op2]	Subtraction
[*:op1,op2. . ,]	Multiplication
[/:op1,op2]	Division
[%: op1,op2]	Modulo
[|:op1,op2]	Logical Or (Result 1, if minimum one operator is not equal to 0)
[&:op1,op2]	Logical And (Result 0, if min. one operator is 0)
[<: op1,op2]	Comparison - Less than (1=TRUE, 0=FALSE)
[=: op1,op2]	Comparison - Equal (1=TRUE, 0=FALSE)
[>: op1,op2]	Comparison - Greater than (1=TRUE, 0=FALSE)
[MOD10:x]	Calculates and prints the Modulo 10 Check digit
[MOD36:x]	Calculates and prints the Modulo 36 Check digit
[MOD43:x]	Calculates and prints the Modulo 43 Check digit
[P:name,mn{o}]	Print result in Price format
[R:x]	Rounding method
[==:text1,text2]	String comparision (1=TRUE, 0=FALSE)

[+:op1,op2, . . .] Addition

Addition options can be used to add several values of text - or barcode fields to print the result on the label.

Syntax: `[+:op1,op2,... ]`

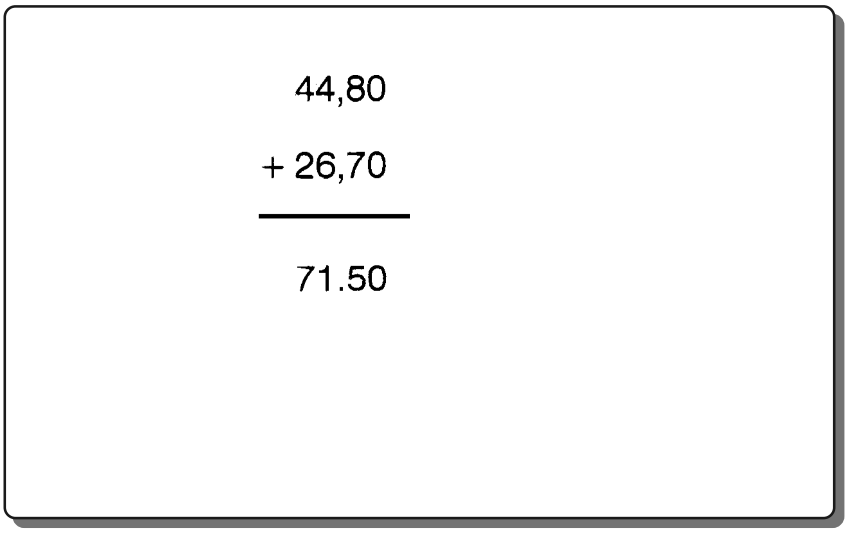
[+:...] - Addition	
op1,op2,...	= Operand 1, Operand 2,Operand 3 ...

2 digits behind the comma are preset as default value, multiple values are allowed. The values might be existing informations of other fields and numbers. Field operators might also be marked „invisible“ - see option **[I] (invisible)** to show only the result**.**

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;44,80
T:var2;20,20,0,3,5;+
T:var3;25,20,0,3,5;26,70
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[+:var1,var3]
A1
```

This simple example adds var1 (44,80) and var3 (26,70) which are defined as fixed values in the label. The addition sign and the line shall help to have a better overview. The result (res) uses the calculation options.



[-:op1,op2,...] Subtraction

Subtraction options can be used to subtract several values of text - or barcode fields to print the result on the label.

Syntax:

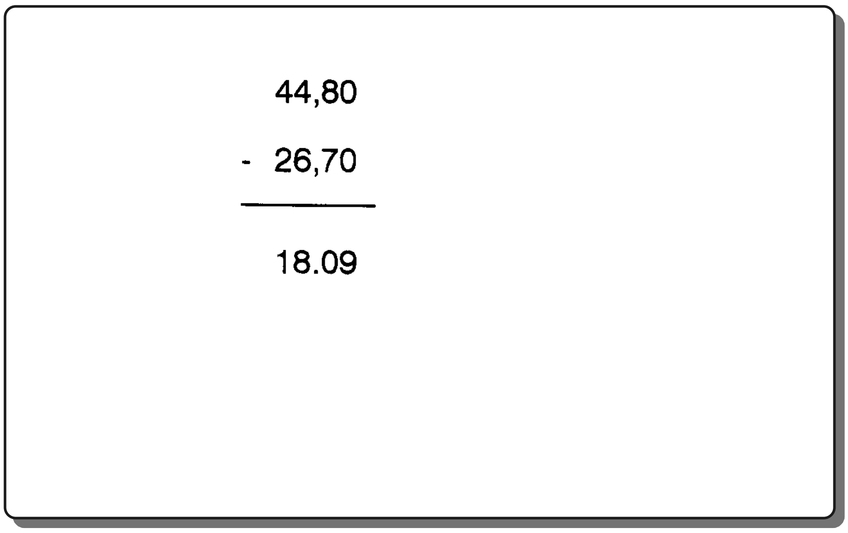
```
[-:op1,op2,...]
```

[-:...]	
op1,op2,...	= minuend (op1) minus subtrahend (op2) ....

2 digits behind the comma are preset as default value, multiple values are allowed. The values might be existing informations of other fields and numbers. Field operators might also be marked „invisible" - see option **[I]**) to show only the result.

Example:

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



[*:op1,op2, . .] Multiplication

Multiplication of several operands of text or barcode fields and prints the result in the defined field on the label.

Syntax: `[*:op1,op2,..]`

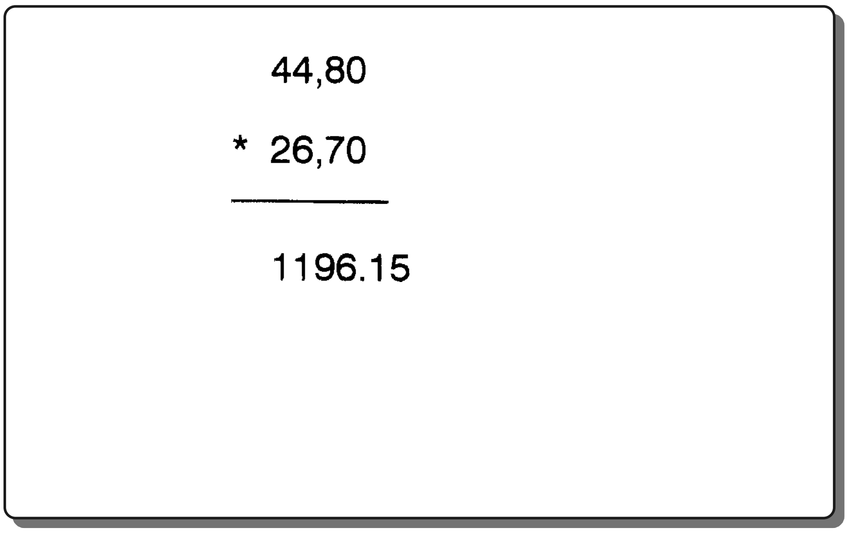
[*:...] - Multiplication	
op1,op2,..	= operand1 (op1) * operand 2 (op2)...

2 digits behind the comma are preset as default value, multiple values are allowed. The values might be existing informations of other fields and numbers. Field operators might also be marked „invisible“ - see option **[I]** to print only the result.

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;44,80
T:var2;20,20,0,3,5;*
T:var3;25,20,0,3,5;26,70
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[*:var1,var3]
A1
```

This example multiplies var1 (44,80) and var3 (26,70) which are defined as fixed values in the label. The field with the multiply sign and the line are only added to get a better overview. The text field (res) uses the calculation options.
This option is useful to calculate the total price of a weighted product, where the data of var1 might be the weight of the product and var3 might be a fixed value which is the price per unit.



[/ :op1,op2] Division

Divides operand1 (op1) by operand2 (op2) and prints the result in the defined field on the label.

Syntax:

```
[/:op1,op2,...]
```

[/ :...] - Division

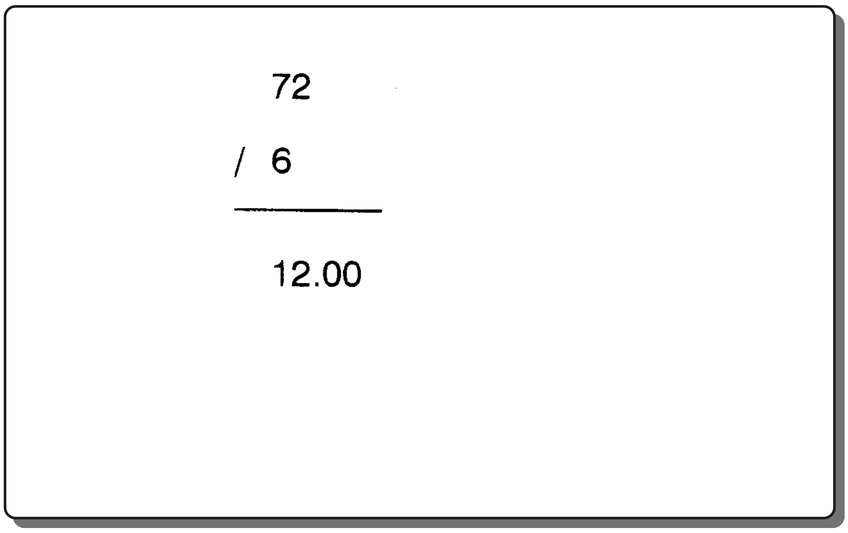
op1,op2...	=	Operand1 (op1) divided by operand2 (op2) ...

2 digits behind the comma are preset as default value. The values might be existing informations of other fields and numbers. Field operators might also be marked „invisible“ - see option **[I]** to print only the result.

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;72
T:var2;20,20,0,3,5;/
T:var3;25,20,0,3,5;6
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[/:var1,var3]
A1
```

This example divides var1 (72) by var3 (6) which are defined as fixed values in the label. The addition sign and the line shall help to have a better overview. The result (res) uses the calculation options. This option is for example useful to calculate the total price of a weighted product, where the data of var1 might be the weight of the product and var3 might be a fixed value which could be the price per unit.



[%: op1,op2] Modulo

The remainder of the two operands is the modulo.

Syntax:

```
[%: op1,op2]
```

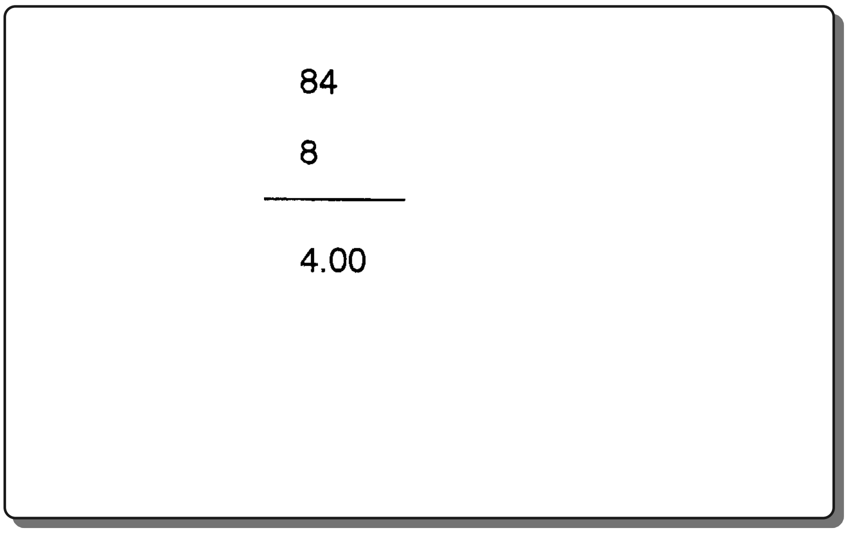
[%: ...] - Modulo	
op1,op2,...	= operand1 (op1), operand2(op2)

2 digits behind the comma are preset as default value. The values might be existing informations of other fields and numbers. Field operators might also be marked „invisible“ - see option **[I]** to print only the result.

Example:

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

The remainder of 84, divided by 8 is 4.



[%: op1,op2] Modulo

Example:

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

The sample above produces a counter from 1 to 15 and sets it back to 1, to restart the counter from the beginning.

[|:op1,op2] Logical Or

Logical **Or** (Result will be „1“, if minimum one operator is not equal to 0, Result will be „0“ on all other conditons.

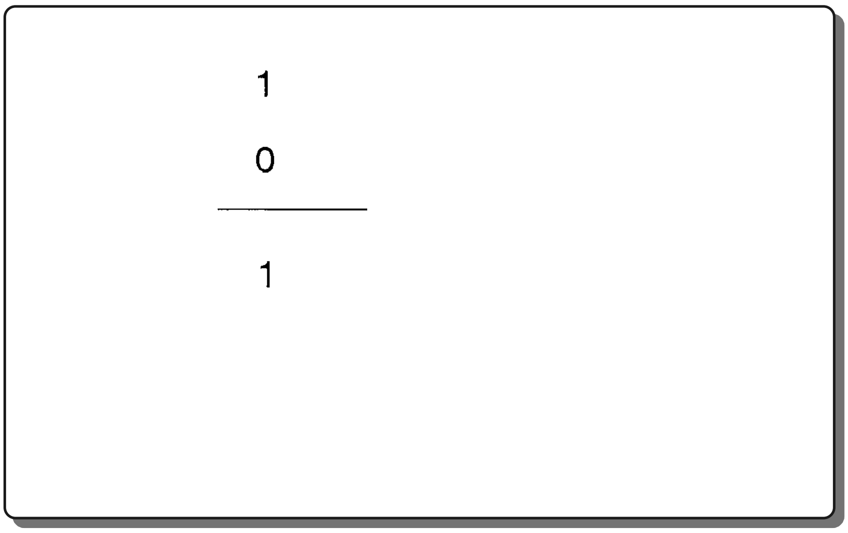
Syntax:	`[|:op1,op2]`

[|:...] - Logical OR		
op1,op2	=	operator1 (op1) is compared with operator 2 (op2)

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;1
T:var2;25,20,0,3,5;0
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[|:var1,var2]
A1
```

Result 1, because the first variable (var1) is not 0.

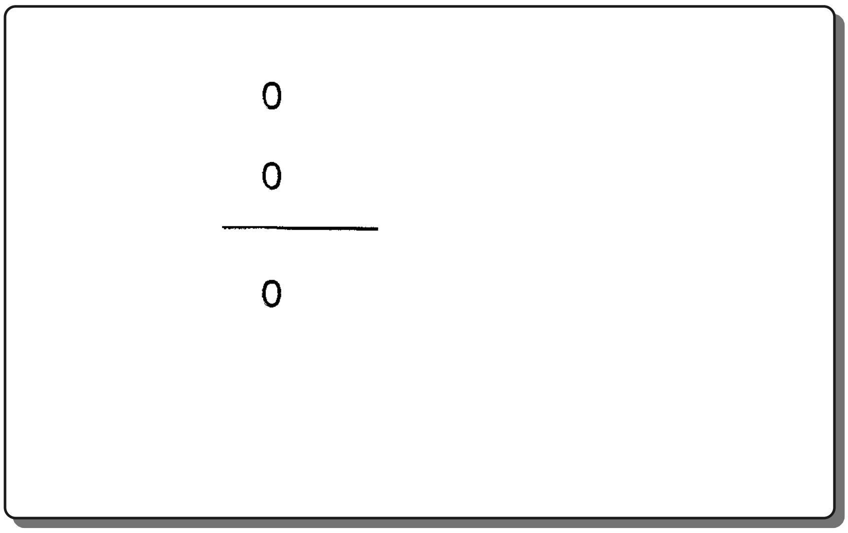


[|:op1,op2] Logical Or

Example:

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

Result 0, because both variables are 0.



[&:op1,op2] Logical AND

Compares 2 values and prints the result which is defined in that field. Result is „1“ if both values for the comparision are identical“ - otherwise the result is 0.

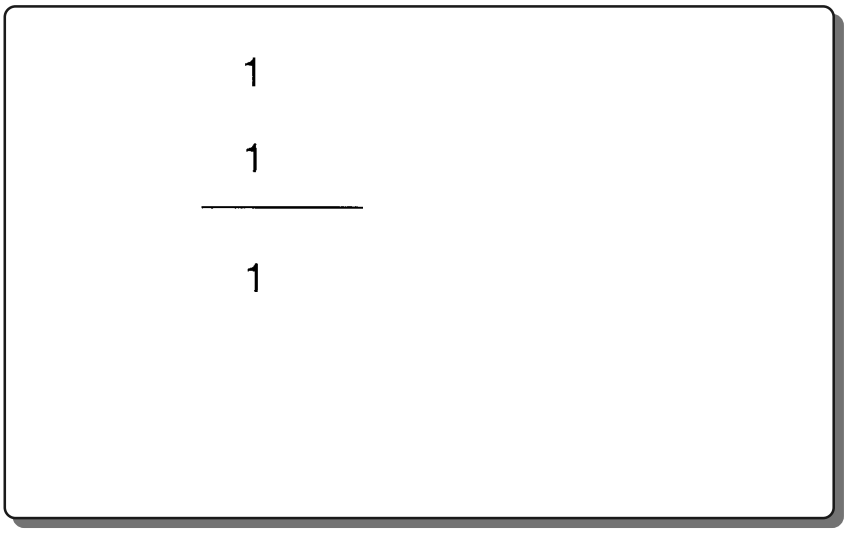
Syntax:

```
[&:op1,op2]
```

[&:...] - Logical AND	
op1,op2	= operator1 (op1) is compared with operator 2 (op2)

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;1
T:var2;25,20,0,3,5;1
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[&:var1,var2]
A1
```



[<: op1,op2] Comparision < Less than

Compares 2 values and has the result „1“ if the expression is true, otherwise 0

Syntax:

```
[<:op1,op2]
```

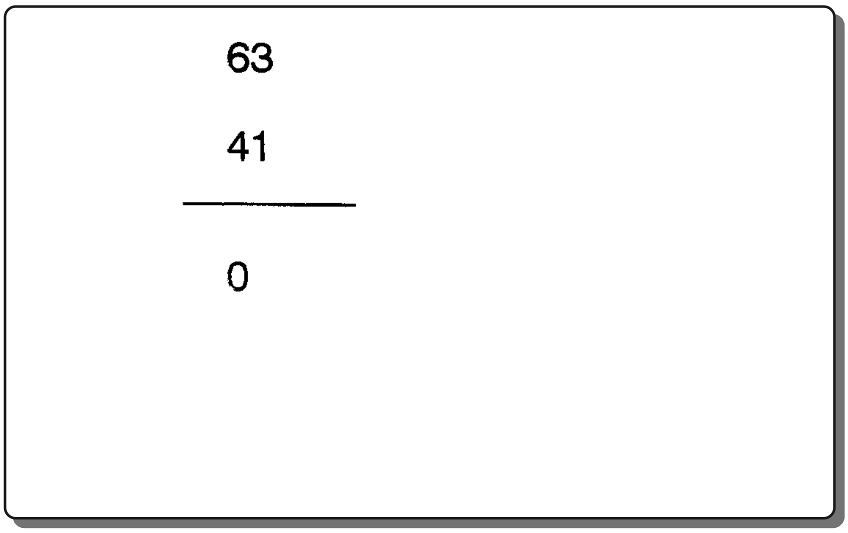
[<:...]		
op1,op2	=	operand 1 (op1) less than operand 2 (op2)

The result is true (1), when operand1 (op1) is less than operand2 (op2)

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;63
T:var2;25,20,0,3,5;41
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[<:var1,var2]
A1
```

In our example: Operand1 (var1 =63) is not less than operand2 (var2 =41) - the result is false (0)



[=: op1,op2] Comparision = Equal

Compares 2 values and has the result true (1), when the values are equal or false. (0) when these two values are not equal.

Syntax:

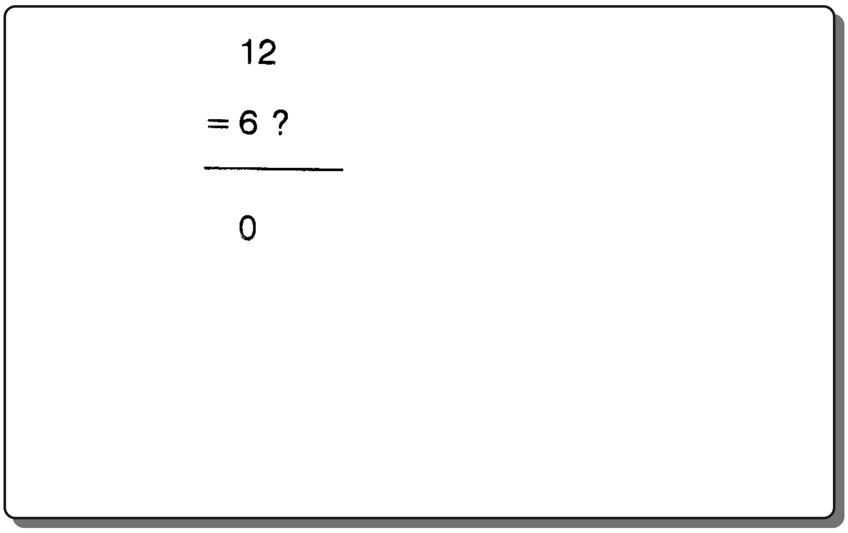
```
[=: op1,op2]
```

[=:...]	
op1,op2	= Operand1 (op1) compared with operand 2 (op2)

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;12
T:var2;20,20,0,3,5;=    ?
T:var3;25,20,0,3,5;6
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[=:var1,var3]
A1
```

Compares 12 and 6 and has the result „false“ (0)



[==: text1,text2] String Comparision == Equal

Compares 2 text strings and has the result true (1), when the text strings are equal or false. (0) when these two strings are not equal.

Syntax:

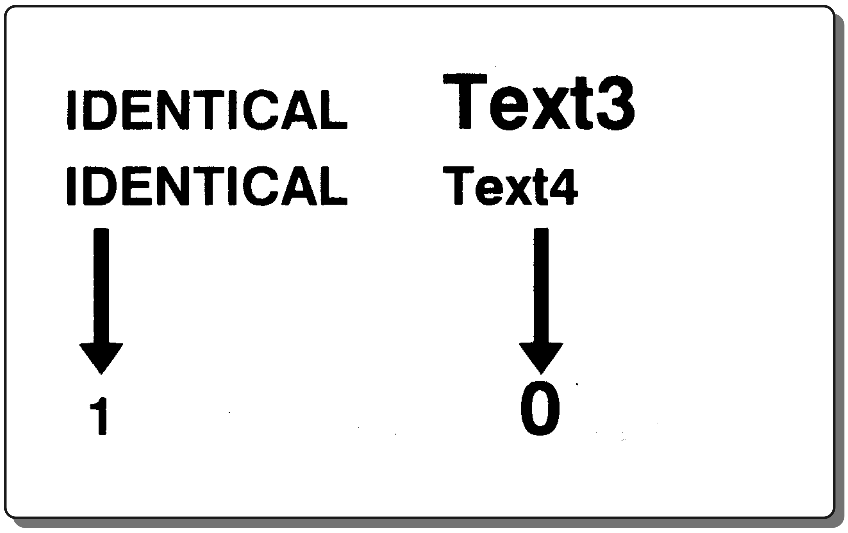
```
[==:text1,text2]
```

[==:...] - String comparision		
text1,text2	=	textstring1 (text1) compared with textstring2 (text2)

Example:

```
m m
J
O R
S l1;0,0,68,70,100
T:VAR1;5,20,0,5,pt20;IDENTICAL
T:VAR2;5,30,0,5,pt20;IDENTICAL
G 10,33,270;L:15,2,s,a
T:VAR3;8,60,0,5,pt20;[==:VAR1,VAR2]
T:VAR4;55,20,0,5,10;Text3
T:VAR5;55,30,0,5,pt20;Text4
G 68,33,270;L:15,2,s,a
T:VAR6;65,60,0,5,10;[==:VAR4,VAR5]
A 1
```

Compares identical text strings with the result true (1) and compares 2 other text strings and has the result „false“ (0)



[>: op1,op2] Comparision > Greater than

This option compares 2 values and has the result = true (1) or false (0)

Syntax:

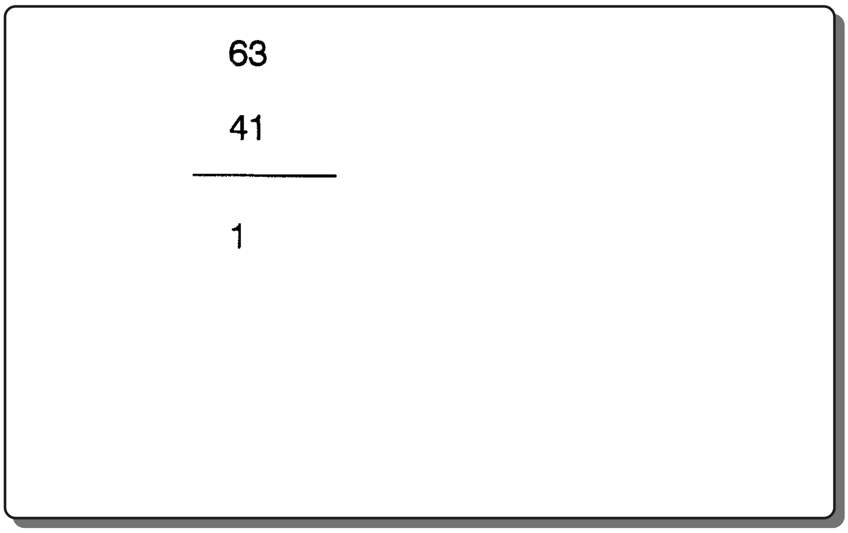
```
[>: op1,op2]
```

[>: ...] - comparision greater than	
op1,op2	= compares operater1 (op1) with operator2 (op2)

The result is true (1), when operand1 (op1) is greater than operand2 (op2)

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;25,10,0,3,5;63
T:var2;25,20,0,3,5;41
G 20,25,0;L:20,0.3
T:res;25,35.0,.0,3,5.0;[>:var1,var2]
A1
```



[MOD10:x] Calculate the Modulo 10 check digit

Calculates and prints the Modulo 10 Check digit for numerical barcodes

Syntax:	`[MOD10:x]`

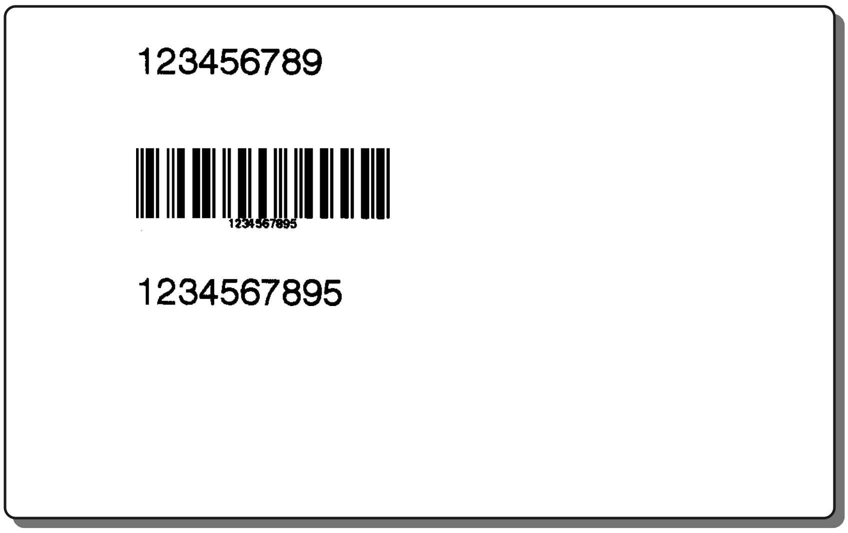
[MOD10:...] - calculate the MOD 10 digit	
x	= value which is used to calculate the check digit

This function can be used to visualize check digits of barcodes, which are sometimes invisible. Some barcodes use a check digit for the scanner only which is not displayed in the human readable line. Some applications require this check digit for internal usage. This can be done with the „Mod10“ function.

Example:

```
m m
J
S l1;0,0,68,71,100
T:input;10,10,0,3,5;123456789
B 10,20,0,2OF5+MOD10,10,.3;[input]
T 10,40,0,3,5;[input] [MOD10:input]
A 1
```

This example uses the input variable for a interleaved 2 of 5 barcode, which has to contain a modulo 10 digit. Usually only the input data is copied to a second field. As the printer cannot know, that the - normally invisible check digit shall be shown on the label. Therefor [MOD10:input] is used.



[MOD36:x] Calculate the Modulo 36 check digit

Calculates and prints the Modulo 36 Check digit.

Syntax:

```
[MOD36:x]
```

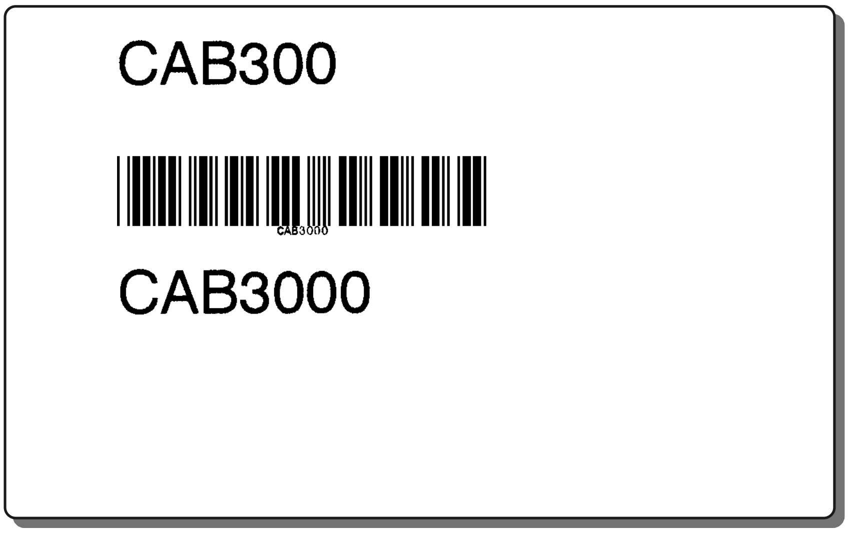
[MOD36:x] - calculate the MOD 36 checkdigit	
x	= value which is used to calculate the check digit

This function can be used to visualize check digits of barcodes, which are sometimes invisible. Some barcodes use a check digit for the scanner only which is not displayed in the human readable line. Some applications require this check digit for internal usage. This can be done with the „Mod36“ function. This function makes only sense together with Code39.

Example:

```
m m
J
S l1;0,0,68,71,100
T:input;10,20,0,3,8;CAB300
B 10,30,0,CODE39+MOD36,10,.3;[input]
T 10,50,0,3,8;[input] [MOD36:input]
A 1
```

This example uses the input variable for a Code 39 barcode. Usually only the input data is copied to a second field, as the printer can not know, that the - normally invisible check digit shall be shown on the label. Therefor [MOD36:input] is used.



[MOD43:x] Calculates the Modulo 43 Check digit

Calculates and prints the Modulo 43 Check digit.

Syntax:

```
[MOD43:x]
```

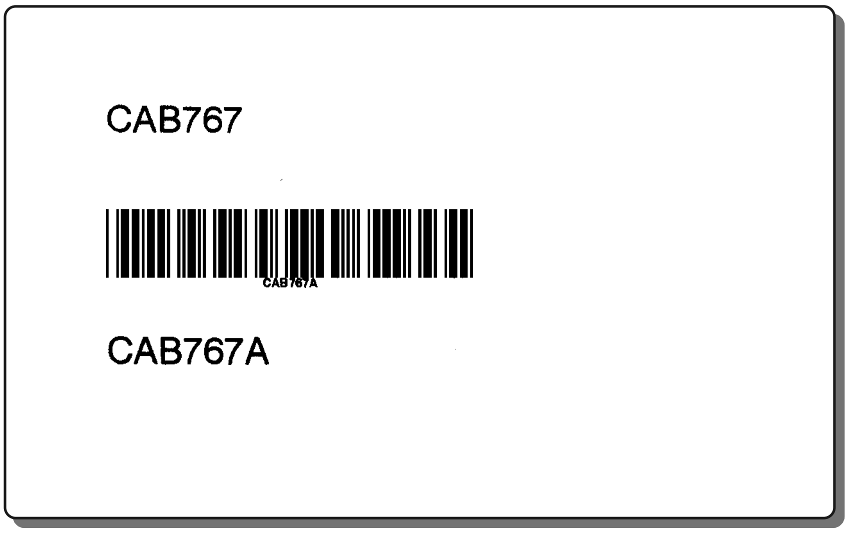
[MOD43:x]	
x	= value which is used to calculate the check digit

This function can be used to visualize check digits of barcodes, which are sometimes invisible. Some barcodes use a check digit for the scanner only which is not displayed in the human readable line. Some applications require this check digit for internal usage. This can be done with the „Mod43“ function. This function makes only sense together with CODE128 and Code39.

Example:

```
m m
J
S l1;0,0,68,71,100
T:input;10,20,0,3,8;CAB767
B 10,30,0,CODE39+MOD43,10,.3;[input]
T 10,50,0,3,8;[input] [MOD43:input]
A 1
```

This example uses the input variable for a Code 39 barcode. Usually only the input data is copied to a second field, as the printer can not know, that the - normally invisible check digit shall be shown on the label. Therefor [MOD43:input] is used.



[P: ...] Print result in Price format

Prints result in price format

Syntax:

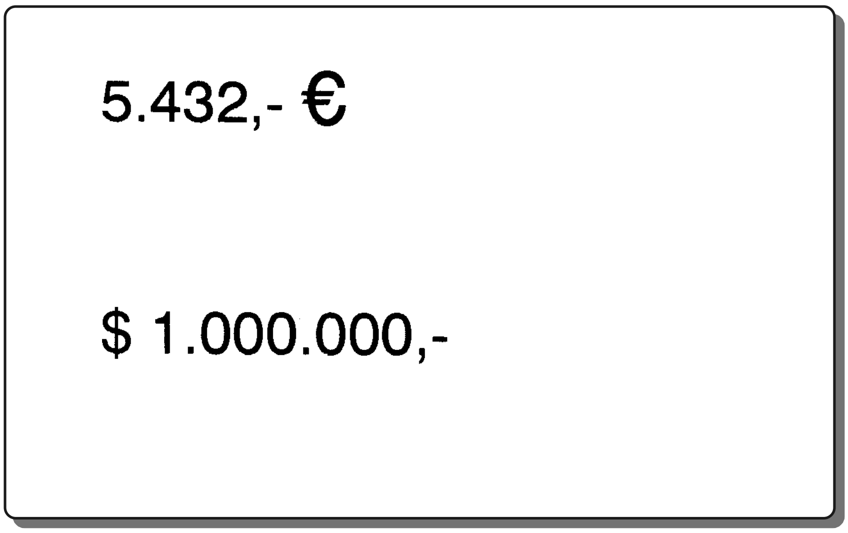
```
[P:name,td{o}]
```

[P:...] - price format option

name	=	field name
t	=	thousands separator
d	=	decimal point character
o	=	optional addendum characters

Example:

```
m m
J
S l1;0,0,68,71,100
T:Price1;10,20,0,3,8;[P:5432,.,-] [U:$20AC]
T:Price;10,50,0,3,8;$ [P:1000000,.,-]
A 1
```



[R:x] Rounding method

cab printers „know“ several rounding methods. To select a specified rounding method use the **[R:x]** option.

Syntax:

```
[R:x]
```

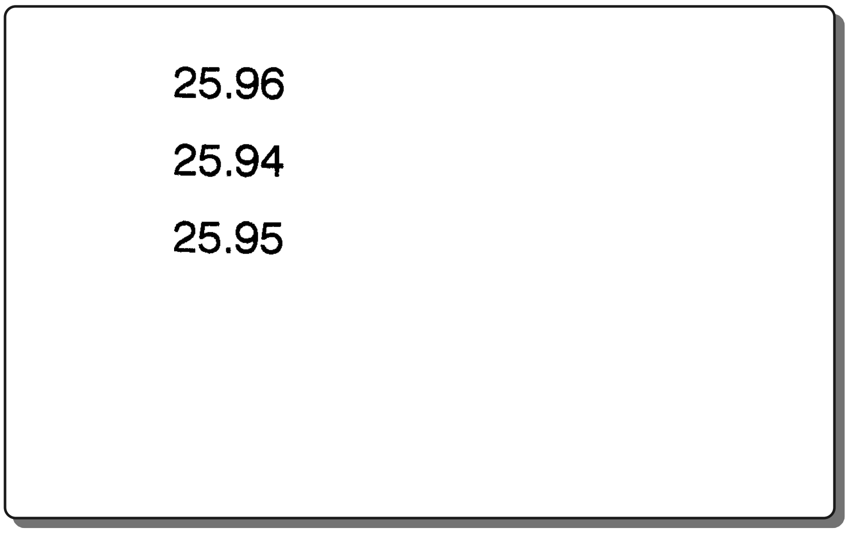
[R:x] - rounding method

x	= n =	no rounding (default)
	u =	rounding up
	d =	rounding down
	m =	round mathematically

The following example shows the functionality:

Example:

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



Special functions

The Special Functions are completing the JScript programming language. On the following pages we describe how to handle display prompts, we show how to write data into a LOG file and offer some examples how data can be formatted.

Special functions (miscellaneous)

[?:x,y,z,{D},{Lx},{Mx},{R},{J}]	Prompt line on the printer´s display
[ABC:x]	
[BIN:x{,y...}]	
[C:fill{,base}]	Leading zero replacement
[D:m,n]	Set number of Digits to print
[DBF:keyfield,keyvalue,entryfield]	DataBase Field
[HEX:x]	
[I{!}{:cond}]	Invisible fields
[JOBID]	
[J:ml]	Justification
[LEN:x]	
[LOWER:x]	Converts the input data in lower case characters
[LTRIM:x]	
[name]	Access a field with a name
[name,m{,n}]	Insert substring from another field
[RTMP{:x}]	Read from a TMP (serial) file
[RTRIM:x]	
[S:name]	Numeric Script style
[SER:start{incr,{freq}}]	Insert SERial numbering
[SPLIT:field,index]	Splits table values
[U:x]	Insert Unicode character
[UPPER:x]	Converts the input data in upper case characters
[WINF]	Writes value into the „INF“ buffer
[WLOG]	Write to LOG file
[WTMP]	Write to TMP (temporary) serial file

[?: ...] LCD prompt

cab printers offer the feature that a standard PC keyboard with USB connector can be connected the printers.All actual printers have this possibility as standard feature.

Labels, graphics, databases and fonts can be saved on the printer´s optional memory card, in the internal memory (IFFS), the external CF card or on an USB memory stick.
Recalling labels can easily be done through the attached keyboard,attached USB scanner or in the worst case through the printer´s control panel buttons - (which is useful only for easy applications)

The printers allow also for variable input, the prompt on the LC display is defined with this command.

[?: ...   ]   LCD prompt

Syntax: `[?:x,y,z{,D}{,Lx}{,Mx}{,R}{,J}]`

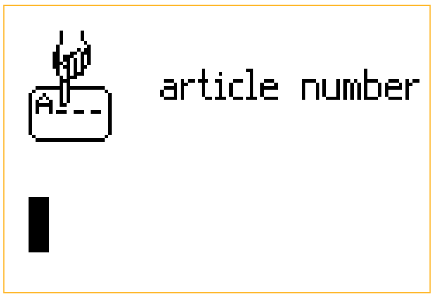
?	**=**	command for the LCD prompt
x	**=**	Text line which appears on the printers LCD (16 characters max.)
y	**=**	optional default value which is displayed on the LCD for the first input otherwise the previous input appears.
z	**=**	defines how often the input has to be entered
D	**=**	**Optional parameters:** deletes the previous input
Lx	=	length of the input line (x=1-200) - which means 1-200 characters
Mx	=	Masks the input with following parameters: **x** = 0 numeric, decimal separators and sign 1 numeric values 2 lower case letters 3 alphanumeric lower case characters 4 upper case letters 5 alphanumeric upper case characters 6 upper and lower case characters 7 alphanumeric upper and lower case characters 8 all characters 0 sign and decimal point No space character is allowed if the exclamation mark " ! “ is placed directly after the **M** option
R	=	Repeats the input prompt if a record could not be found in a database
J	=	repeats the prompt when the printer asks for the input of the amount of labels. (A[?,R]) defines a simple loop for the amount of labels.

[?: ...] LCD prompt

Example:

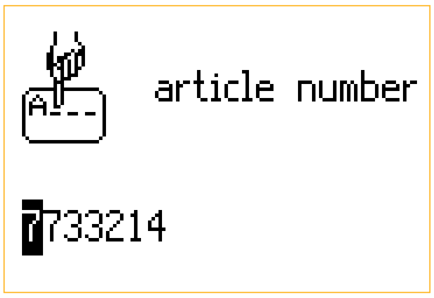
```
m m
O R
J
S l1;0,0,68,70,100
T 10,10,0,5,5;[?:article number]
A1
```

Requests in the display for **article number** and appears like shown in the picture below. Data can now be exchanged through an attached keyboard or scanner or through the navigator pad.


article number

Example:

```
m m
O R
J
S l1;0,0,68,70,100
T 10,10,0,5,5;[?:article number,7733214]
A1
```


article number

7733214

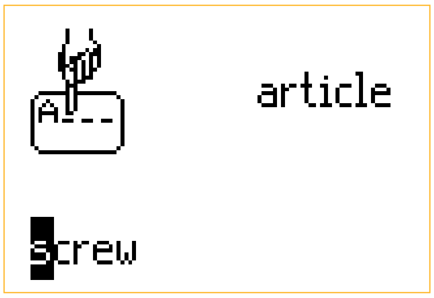
Requests in the display for **article number** and the preset value 7733214. .Data can now be exchanged through an attached keyboard or scanner or through the navigator pad.

[?: ...] LCD prompt

Example:

```
m m
J
O R
S l1;0,0,68,70,100
T 10,10,0,5,5;[?:article,screw,3]
A6
```

Presets in the word screw in the display.



Example:

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

Prompts with the headline **article no**: and the preset value **7733214** each three labels and erases the last input, which is only shown for the first time when the label is recalled.

Example:

```
[?:article,screw,,L8]
```

Prompts with the headline **article no**: and the preset value **7733214.** The maximum length of input data is limited to 8 digits.

Example:

```
[?:number,7733214,,M1111111]
```

Prompts for number with the preset value of **7733214** and masks the input for numeric values only.

Example:

```
[?:artno?,,1,M1114444]
```

Prompts for artno, has no preset value and expects 3 numeric an 4 upper case characters

[?: ...] LCD prompt

Example:

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

Prompts for article number without a preset value, limited to 7 digits and repeated prompt if database record was not found.

Example:

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

Prompts for article with preset value 2200333 and masks the input for 6 digits without space character.

Example for a simple loop:

Example:

```
J simple loop
S l1;0,0,68,71,100
T 10,15,0,3,10;[SER:1]
T 10,30,0,3,10;[?:INPUT?]              (This request prompts only once)
T 10,45,0,3,10;[?:Second INPUT?,,,J]   (This request repeats prompting)
A [?,R]
```

Repeats the prompt until the cancel button is pressed

[ABC:x] Insert ABC value

Inserts a value from ABC (a-series basic compiler). This enables the printer to use abc programs as function.

Syntax:	`[ABC:x]`

[ABC:...] - Insert ABC value	
x =	parameter which is transmitted by abc

[BIN:x{,y ...}]   Insert Binary data

Converts data into binary values. Converted data are 8 bit data. This can be used e.g. for for 2D barcodes which require sometimes special contents.

Syntax:

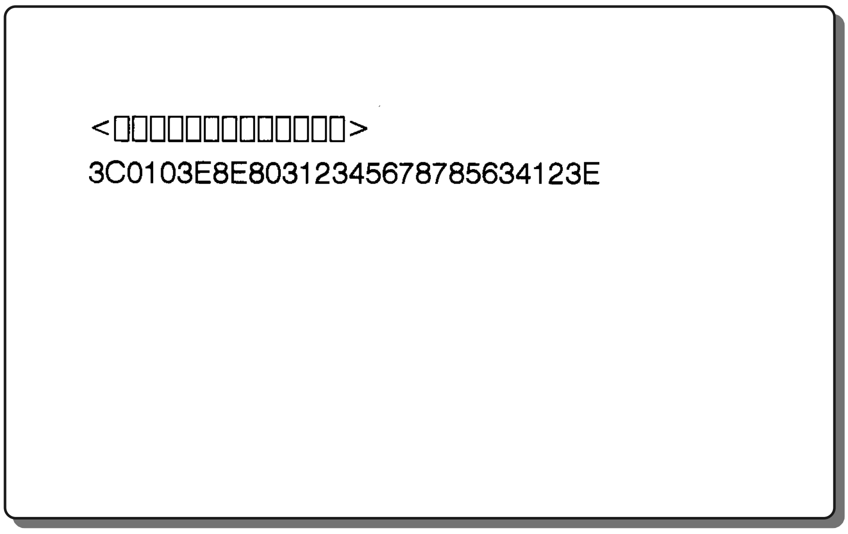
```
[BIN:x{,y...}]
```

[BIN:...] - Insert Binary data	
x =	input data, whereby multiple data can be converted,separated by a comma.

Example:

```
J
mm
S e;0,0,68,70,100
T:aa;10,10,0,3,4;<[BIN:1] [BIN16B:1000] [BIN16L:1000] [BIN32B:$12345678] [BIN32L:$12345678]>
T 10,16,0,3,4;[HEX:aa]
A 1
```

The data is visible in this sample after copying the binary value into a hex value.



```
<□□□□□□□□□□□□□>
3C0103E8E8031234567878563412 3E
```

[BIN16B:x{,y ...}] Insert Binary data, 16 bit - Big Endian

allows to insert binary data in Big Endian format. For further details about binary data Little Endian and Big Endian please refer to Wikipedia at http://en/wikipedia.org/wiki/Endian

Syntax:

```
[BIN16B:x{,y ...} ]
```

[BIN16B:...] - Insert binary data, 16 bit Big Endian	
x{,y ...} =	Binary data

[BIN16L:x{,y ...}]   Insert Binary data, 16 bit - Little Endian

allows to insert binary data in Little Endian format. For further details about binary data Little Endian and Big Endian please refer to Wikipedia at http://en/wikipedia.org/wiki/Endian

Syntax:	`[BIN16L:x{,y ...} ]`

[BIN16L:...] - Insert binary data, 16 bit Little Endian

x{,y ...}	=	Binary data

[BIN32B:x{,y ...}] Insert Binary data, 32 bit - Big Endian

allows to insert binary data in Big Endian format. For further details about binary data Little Endian and Big Endian please refer to Wikipedia at http://en/wikipedia.org/wiki/Endian

Syntax:

```
[BIN32B:x{,y ...}]
```

[BIN32B:...] - Insert binary data, 32 bit Big Endian	
x{,y ...}	= Binary data

[BIN32L:x{,y ...}] Insert Binary data, 32 bit - Little Endian

allows to insert binary data in Little Endian format. For further details about binary data Little Endian and Big Endian please refer to Wikipedia at http://en/wikipedia.org/wiki/Endian

Syntax:

```
[BIN32L:x{,y ...}]
```

[BIN32L:...] - Insert binary data, 32 bit Little Endian	
x{,y ...}	= Binary data

[BITFIELD:...] Bitwise encoded data field

Bitfield creates a bitwise encoded data field. It fills up 8 bits in the Big - Endian - Mode

Syntax:

```
[BITFIELD:bits1,bits2,...bitsn:val1,val2,...val3n]
```

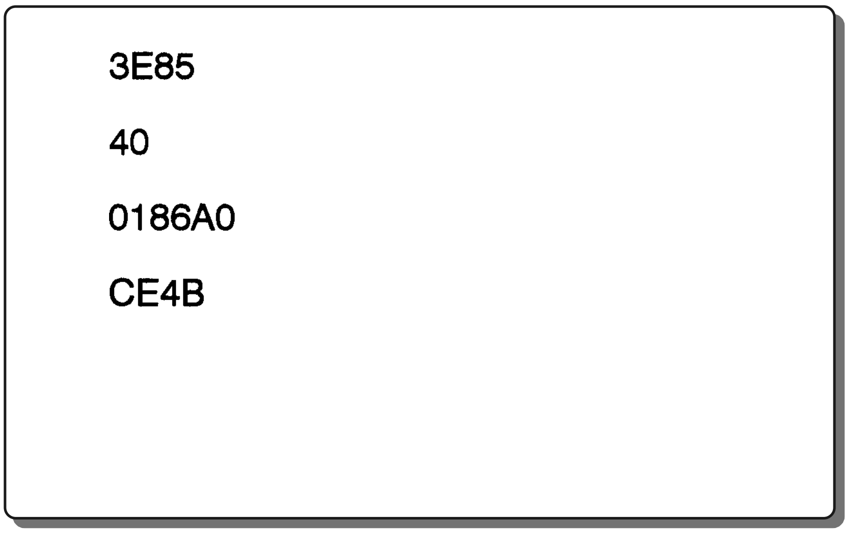
[BITFIELD:bits1,bits2,...bitsn:val1,val2,...val3n**]**	
bits	= 1-32
val	= Value

The amount of bit width (bits1,...) and the amount of values (val1,...) must be identical !

Example:

```
; Testlabel for BITFIELD
m m
J
S l1;0,0,68,71,104
T:t1;10,10,0,3,5;[BITFIELD:12,4:1000,5][I]
T 10,10,0,3,5;[HEX:t1]
T:t2;10,20,0,3,5;[BITFIELD:3:2][I]
T 10,20,0,3,5;[HEX:t2]
T:t3;10,30,0,3,5;[BITFIELD:24:100000][I]
T 10,30,0,3,5;[HEX:t3]
T:t4;10,40,0,3,5;[BITFIELD:5,7,3,1:25,100,5,1][I]
T 10,40,0,3,5;[HEX:t4]
A 1
```

The example above creates 4 bitfields, marked as invisible (non printable) . The second programming line converts the value into a HEX value for the printout.



```
3E85
40
0186A0
CE4B
```

[C: ...] Leading zero replacement

Leading zeroes can be replaced with this function. The default counting system for serialized fields (base) is 10 and can be replaced with values from 2...36. This command with some date or time functions to suppress leading zeroes for single digit month or time.

Syntax:

```
[C:fill{,base}]
```

C=	Leading zero replacement
fill	= fill characters
base	= optional parameter to set the counting system

Please see the example on the next page

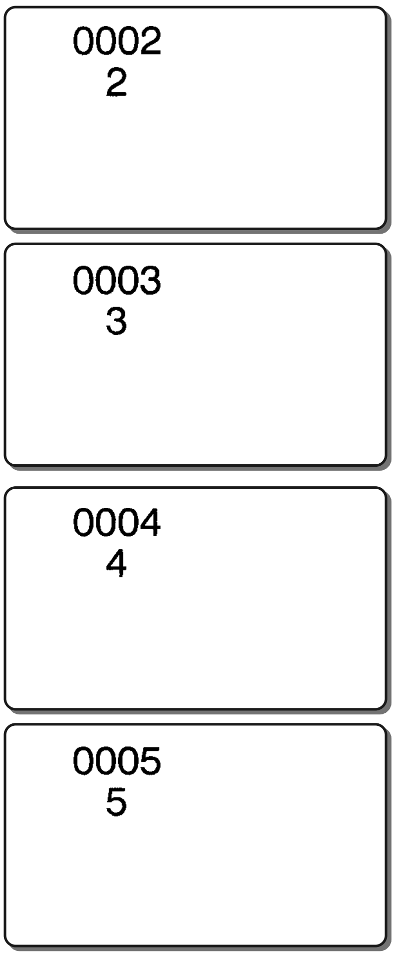
[C: ...] Leading zero replacement

Example:

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

Prints 4 labels with 2 counters- one counter with leading zero and the other counter without leading zeroes. The counter starts with the number 2.

Please see option " [Ser ...] " for more details about serial numbering.



[D:...] Set number of Digits

This option allows for special formatting on a calculated field.

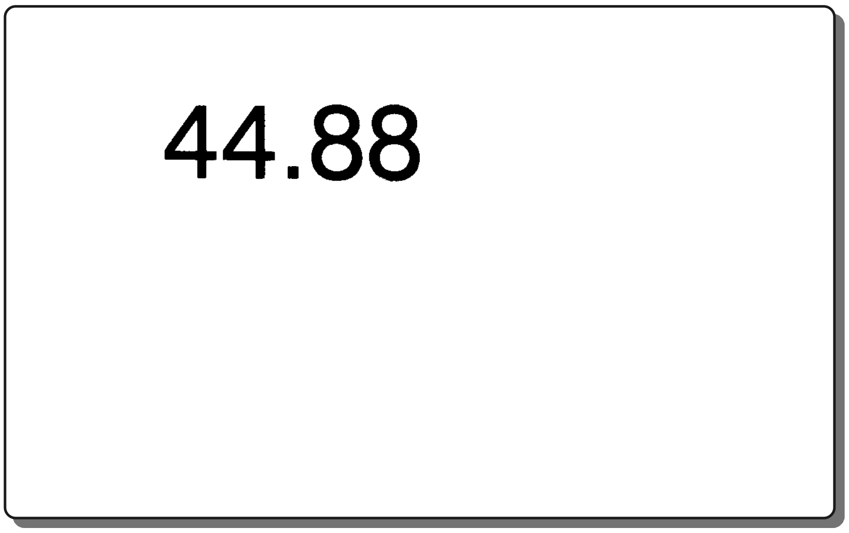
Syntax:

```
[D:m,n]
```

D=	Set number of Digits	
m	=	amount of digits
n	=	digits after the comma (2 is default value)

Example:

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



[DBF:...] Database file access

Syntax:

```
[DBF:key,keyvalue,entryfield]
```

Command to access data from a DBase IV ™ compatible database on the optional memory card.

[DBF:...] - Database file access		
key	**=**	Search value of the database
keyvalue	**=**	is defined by the alphanumeric value in the actual record
entryfield	=	transmits the value of the actual record

Example:

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

Searches in the database for the keyvalue NUMBER, in the field NUMBERTA and transmits the value of ARTICLE.

The“ E „command must be defined,before this command can be used.
Only one database can be used at the same time in a label.
This function makes only sense if small databases are used. More database possibilites are available with the cab database connector, later described in this manual.

[HEX:x ...] Hexadecimal conversion

Converts binary data into a hexadecimal string. If "normal“ data is included, only the least significant byte of the unicode is converted.

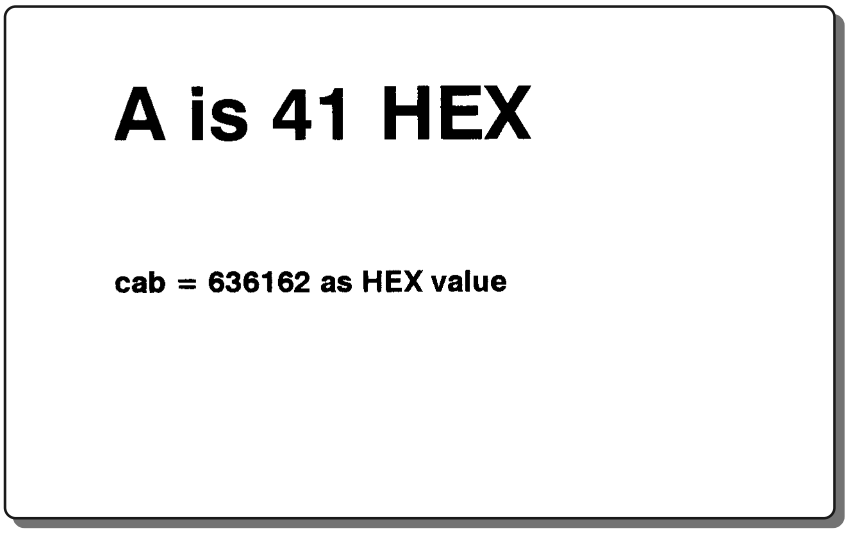
Syntax:

```
[HEX:x...]
```

[HEX:x...] - Hexadecimal conversion	
x	= data

Example:

```
m m
J
S l1;0,0,68,70,100
T:Original;0,0,0,5,5;A[I]
T:HEX;10,20,0,5,10;[Original] is [HEX:Original] HEX
T:Original1;0,0,0,5,5;cab[I]
T:HEX1;10,40,0,5,4;[Original1] = [HEX:Original1] as HEX value
A1
```



[I: ...] Invisible fields

This function defines a field as invisible (it will not appear on the printout). The invisible function is very helpful when some items shall not shown on the label, but they might be required for other operations, such as calculations or for substring operations etc.

Syntax:

```
[I{:Condition}]
```

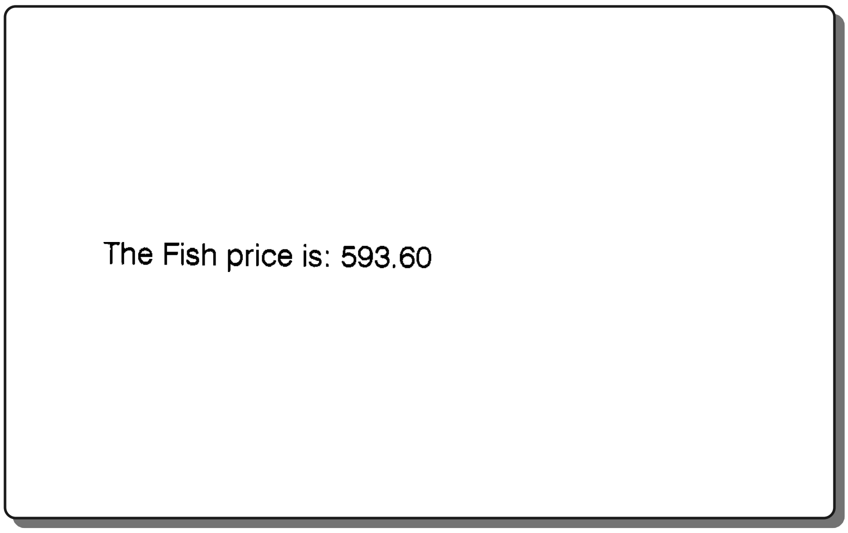
[I...] - Invisible Field (suppresses the printout of a field)

Condition	=	Field will print if Condition is not „0“
!Condition	=	inverted function of „Condition“

Example:

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

This example requests for input on the LC Display of the printer and multiplies this value with the priceunit which is defined as fixed value. Both fields are invisible. Only the result of the price calculation will print.
In our example the fish weight was 12 Kilogramms.
Invisible fields must be defined such as regular or visible fields and the syntax must be correct.
They may be located on the same position. That doesn´t matter as they do not appear on the label


The Fish price is: 593.60

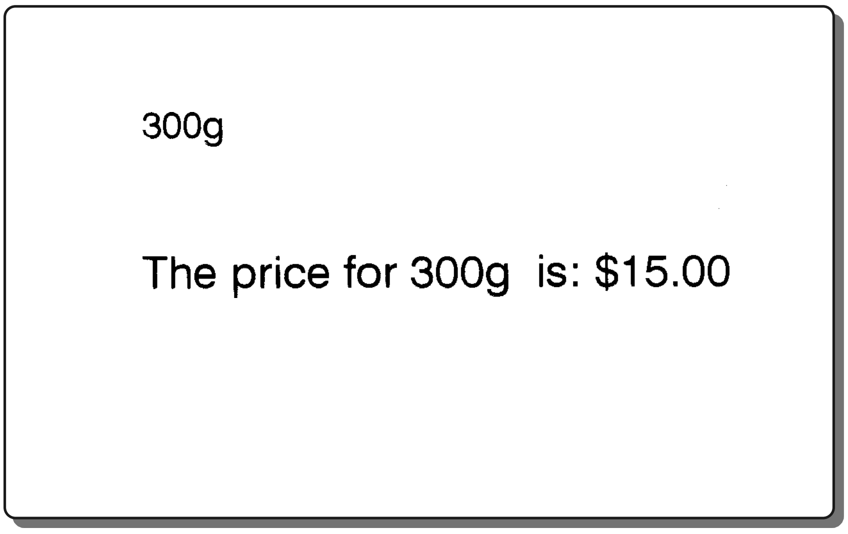
[I: ...] Invisible fields

Example:

```
J
S l1;0,0,68,71,100
T:VISIBLE;10,20,0,3,5;[?:Show Weight? (Y/N),,,,M4][I]
T:VISIBLE1;50,20,0,3,5;[==:VISIBLE,N][I]
T:WEIGHT;10,20,0,3,5;[?:Weight?:]g [I:VISIBLE1]
T:PRICEUNIT;10,20,0,3,5;[I] 0.05
T:RESULT;10,40,0,3,6;The price for [WEIGHT] is: $
[*:WEIGHT,PRICEUNIT]
A 1
```

This example requests for input on the LC Display of the printer and waits for the upper case character „N“ to suppress the printout of the keyed in value „WEIGHT“. (Anything else than „N“ will cause the WEIGHT field to print.) In the example below we did not key in „N“, so the value prints in the upper left corner. The result depends on your input value.

Invisible fields must be defined such as regular or visible fields and the syntax must be correct. They may be located on the same position. That doesn´t matter as they do not appear on the label.



[JOBID] print JOB ID

The JOBID command prints the Identification of the print job. For further information please see also "j Job-ID" and "ESC j".

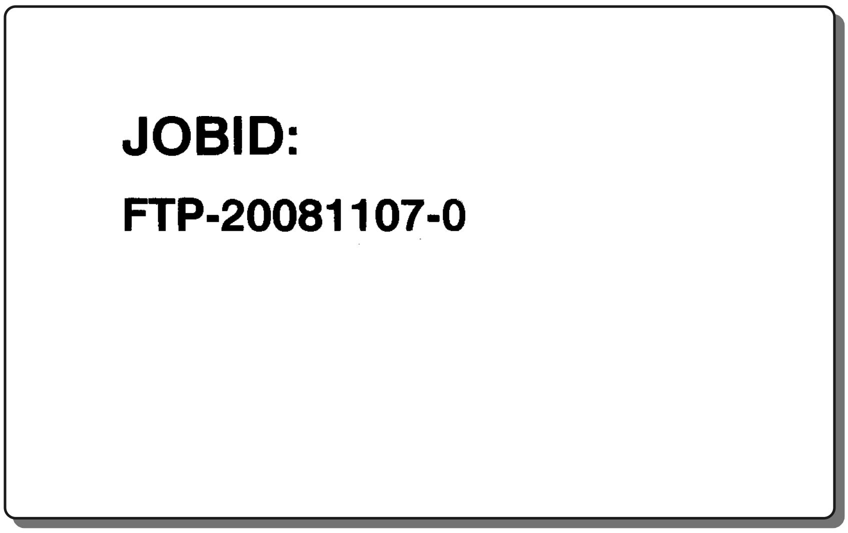
Syntax:

```
[JOBID]
```

[JOBID] - print Job ID

Example:

```
m m
J
O R
S l1;0,0,68,70,55
T 10,20,0,5,7;JOBID:
T 10,30,0,5,6;[JOBID]
A 1
```



[J: ...] Justification

The J command can be used to set the orientation of a text string or for a 1D barcode in a specified area.

Syntax:

```
[J:ml]
```

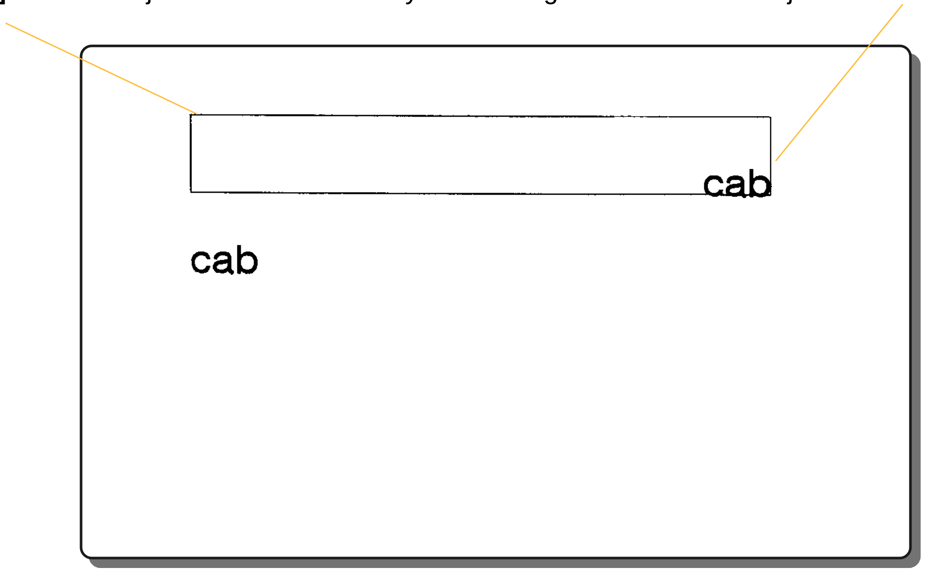
J - Justification	
m	= **l** - left = **c** -centered = **r** - right
l	= length of the specified area where the text string will be justified

Positions are measured in millimeters or in inches, whatever is set by the "m" command.

Example:

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

The Field „NOADJUST“ is transmitted without modification and the Field „ADJUST“ adjusts the textline to the right side of the defined area. (Shown with added rectangle.)
[J:r70] = area of justification -marked by the rectangle. In this ara we adjust the text on the right side.

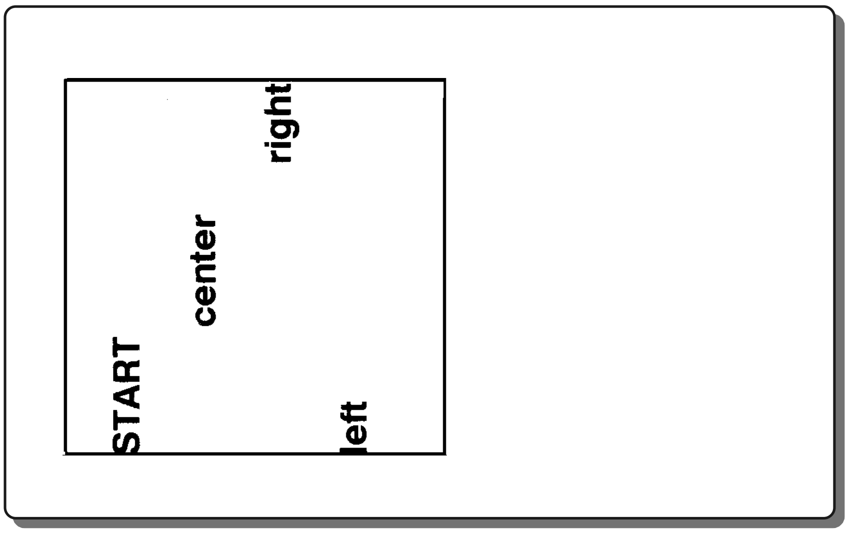


[J: ...] Justification

Another example where the text is rotated. It is helpful to experiment with this command to understand clearly how it works.

Example:

```
m m
J
S l1;0,0,68,71,104
G:AREA;0,10,0;R:50,50,.4,.4
T:NOADJUST;10,60,90,5,5;START
T:ADJUST;20,60,90,5,5;center[J:c50]
T:RightADJ;30,60,90,5,5;right[J:r50]
T:LeftADJ;40,60,90,5,5;left[J:l50]
A 1
```



[LEN:x] Text Length detection

This special command delivers the length of the specified text (x)

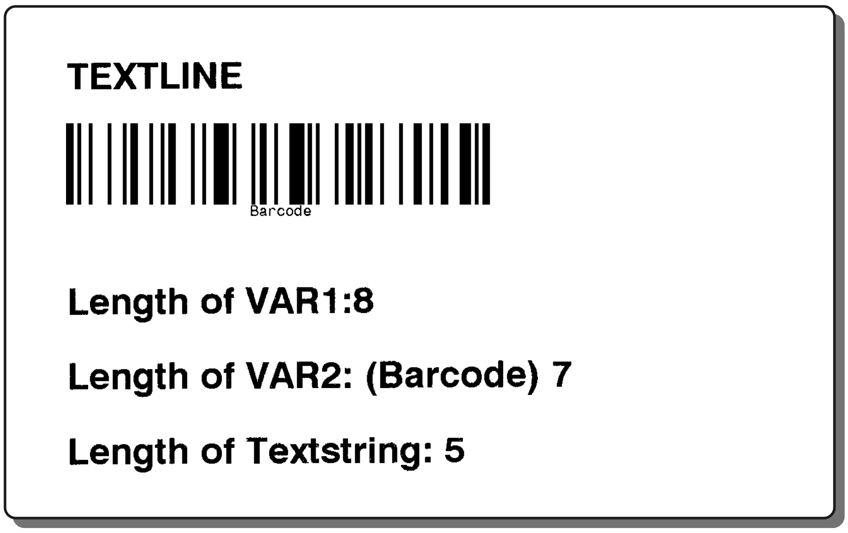
Syntax:

```
[LEN:x]
```

[LEN:...] - text length detection	
x =	Textstring or variable name

Example:

```
mm
J
O R
S l1;0,0,68,70,100
T:VAR1; 10,10,0,5,5;TEXTLINE
B:VAR2; 10,15,0,CODE128,12,.5;Barcode
T 10,40,0,596,5;Length of VAR1:[LEN:VAR1]
T 10,50,0,5,5;Length of VAR2: (Barcode) [LEN:VAR2]
T 10,60,0,5,5;Length of Textstring: [LEN:Hallo]
A1
```



[LOWER:...] Converts to lower case characters

The „LOWER“ function converts text contents into lower case characters

Syntax:

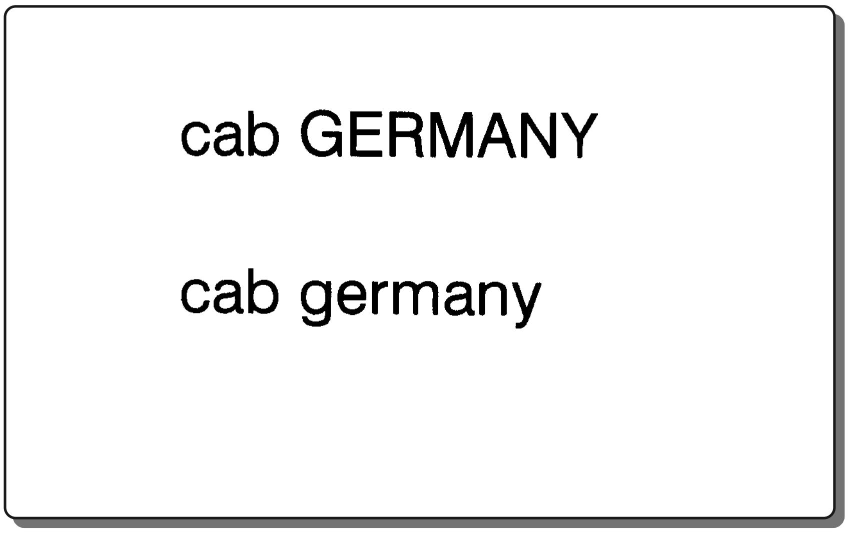
```
[LOWER:Name]
```

[LOWER:...]	
Name	= Variable name

Example:

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

Prints the field „Input“ as it is keyed in, and prints the same data in field „LOWERCASE“ as lowercase characters.



[LTRIM:...] Trim data Left

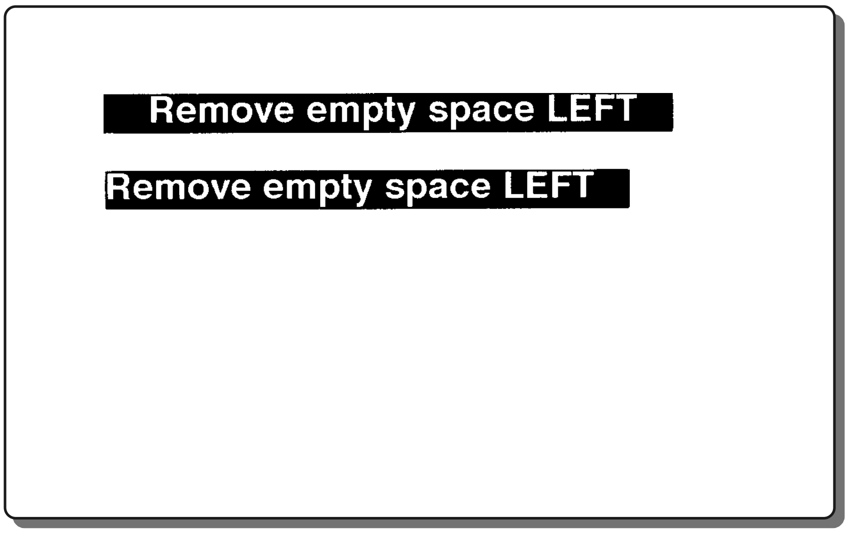
The LTrim command removes space characters and Tab characters at the beginning of a text line.

Syntax: `[LTRIM:x]`

[LTRIM:...] - Trim data	
x	**= data**

Example:

```
m m
J
S l1;0,0,68,70,100
T:CutMe;10,20,0,5,5,n;     Remove empty space
T:CutOff;10,30,0,5,5,n;[TRIM:CutMe]
A1
```



[name]   Access a field with a name

Uses previously defined field contents of text or barcode fields for further operations. This might be to concetenate the values of different fields, to use the values for mathematical operations etc. It is required that the predefined field names are unique.

The name option can use a predifined field content multiple times within a label.

Syntax:

```
[name]
```

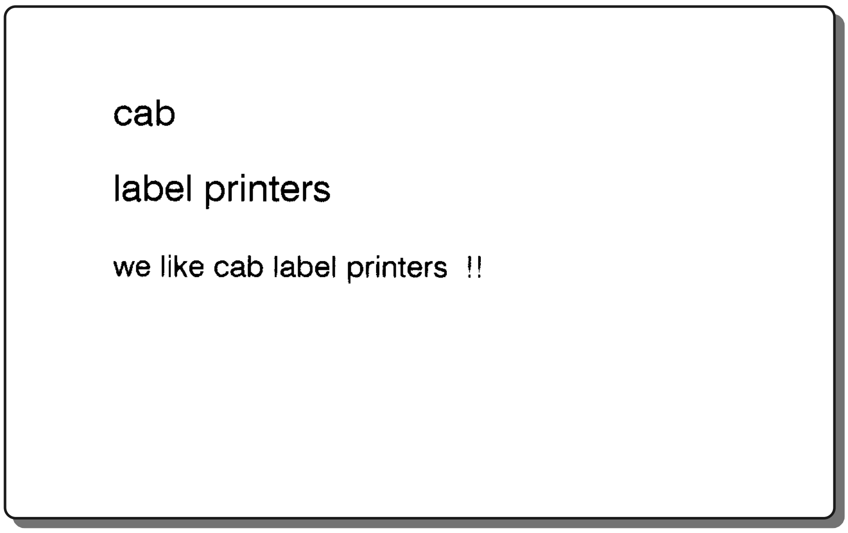
name = previously defined fieldname

Example:

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

FIELD1 and FIELD2 are linked with additional standard text in FIELD3

Note: Field names are case sensitive !!
A fieldname must be defined unique. Using the same name twice or more often is not allowed and causes a "Protocoll error" in the printer´s display..



[name,m{,n}] insert substring

Extracts data from an existing data string of an other previously defined field. Parts of field contents can be used for further operations in another field.

Syntax:

```
[name,m{,n}]
```

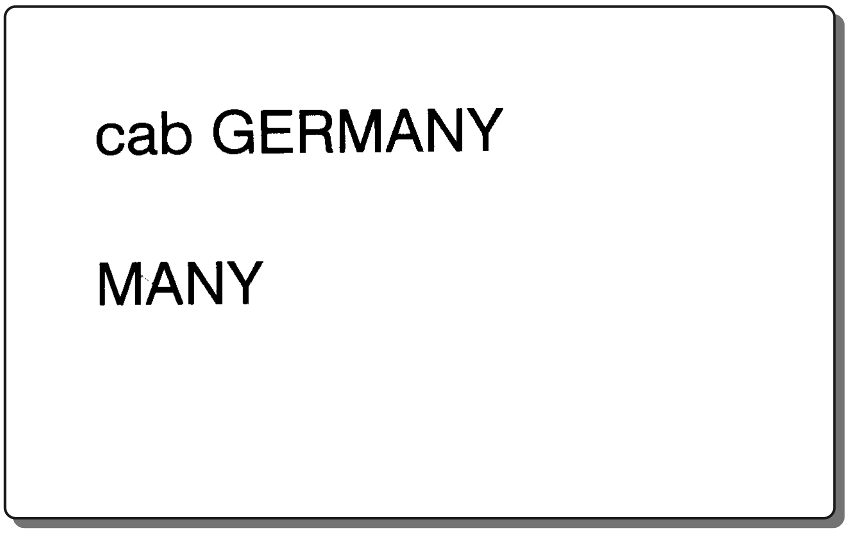
name	=	previously defined field name
m	=	position of the first character to be copied
n	=	amount of characters to copy

m and **n** could be also variables from prior calculations

Example:

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

This example uses the previously defined field with the field name „ORIGINAL“ and cuts from the content "cab GERMANY“ 4 characters, starting at character number 8.
The result is shown below.



[RTMP...] Read value from serial (TMP) file

Reads the value from a serial file of the optional memory card

Syntax:	`[RTMP{,x}]`

[RTMP:...] - Read value from serial file	
x	= defines how many times the value will repeated

See also the command [WTMP] Write value as serial temp file.

[RTRIM:...] Trim data Right

The RTrim command removes space characters or Tab characters at the end of a text line.

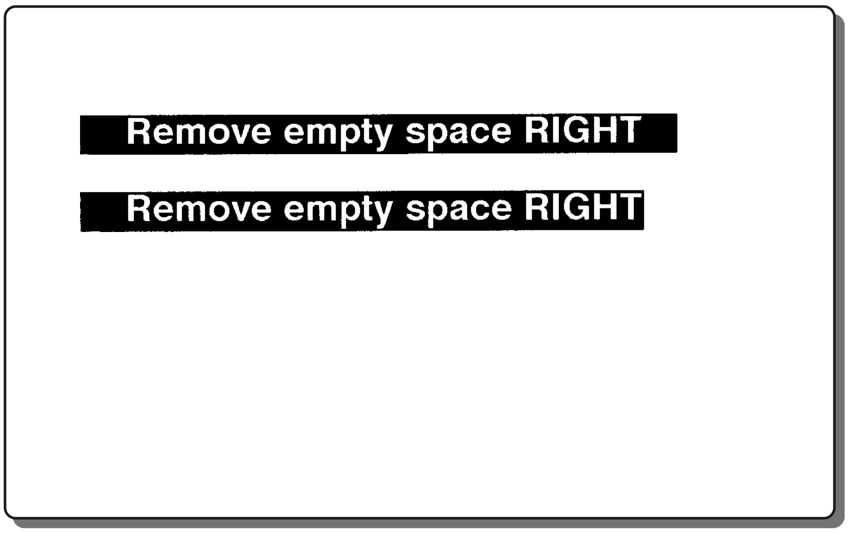
Syntax:

```
[RTRIM:x]
```

[RTRIM:x] - Trim data right	
x	= data

Example:

```
m m
J
S l1;0,0,68,70,100
T:CutMe;10,20,0,5,5,n;     Remove empty space RIGHT
T:CutOff;10,30,0,5,5,n;[RTRIM:CutMe]
A1
```



[RUSER...] Read value from (user) memory

Reads the value from the „user memory". Maximum length is 32 bytes.

Syntax: `[RUSER{,x}]]`

RUSER	=	Read USER file, e.g. serial number
x	=	defines how many time the value will repeated

See also the command "[WUSER]". - Write value to user memory.

[S:...] Script style for numeric values

Influences the script style for numeric values. LATIN or ARABIC are valid values. Selecting ARABIC is only possible with font type -3 or special arabic truetype fonts. This command has no influence on barcodes.

Syntax:

```
[S:name]
```

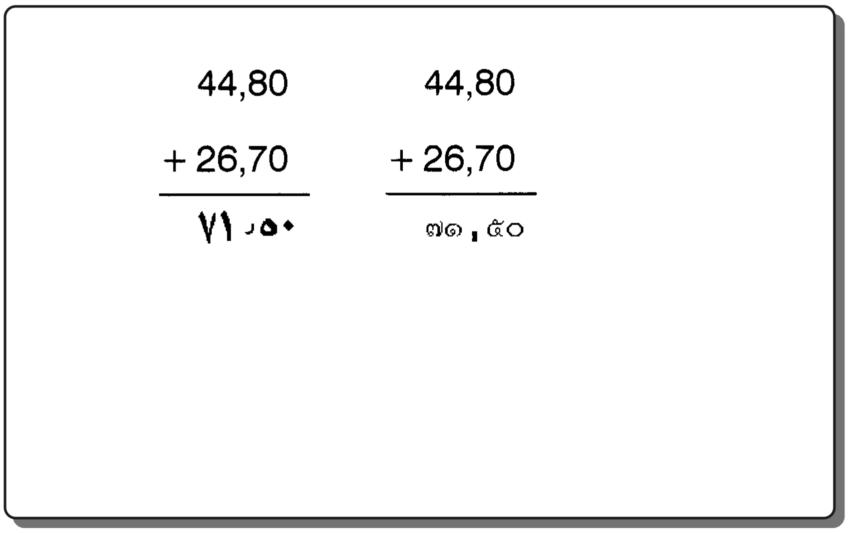
[S:...] - Script style for numeric values

name	=	Arabic
	=	Latin
	=	Thai

Example:

```
m m
J
S l1;0,0,68,71,100
T:var1;15,10,0,3,5;44,80
T:var2;10,20,0,3,5;+
T:var3;15,20,0,3,5;26,70
G 10,23,0;L:20,0.3
T:res;15,28,0,-3,x2,y2;[+:var1,var3][S:ARABIC]
T:var4;45,10,0,3,5;44,80
T:var5;40,20,0,3,5;+
T:var6;45,20,0,3,5;26,70
G 40,23,0;L:20,0.3
T:res1;45,28,0,-3,x2,y2;[+:var1,var3][S:THAI]
A1
```

Prints the result of this calculation in arabic script style.



[SER:...] - Serial numbering

Causes the printer to print serial numbers.

Syntax: `[SER:start{,incr,{freq}}]`

[SER:...]	**=**	Serial numbering
start	=	Initialisation value - sets the start number
incr	=	increment value - presets the number which is added to the start number
freq	=	frequency - defines the number of identical values on the labels before the serial number increments. "16" will cause a hexadecimal base 16 counting.

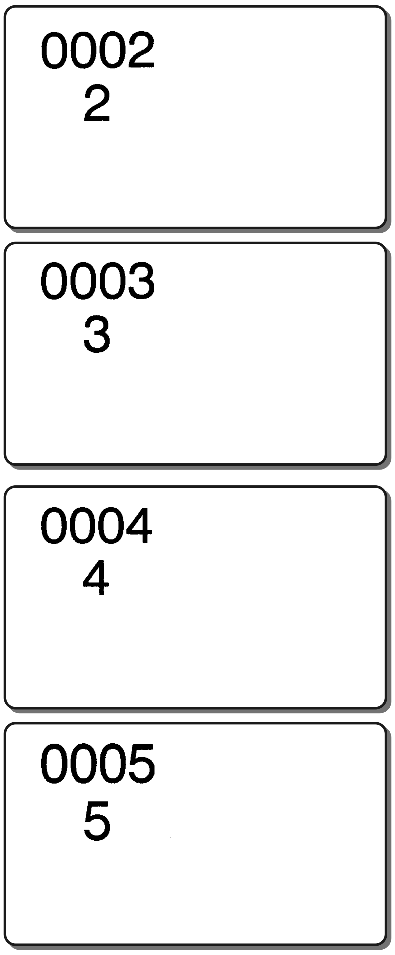
The printers will use automatically "1“ if incr and freq are not set. Please see also the samples on the next pages.

[SER:...] - Serial numbering

Example:

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

The same example as for the „C:Fill..“ command has been used (leading zero replacement)
Please see there to get more information about these functions.



[SER:...] - Serial numbering

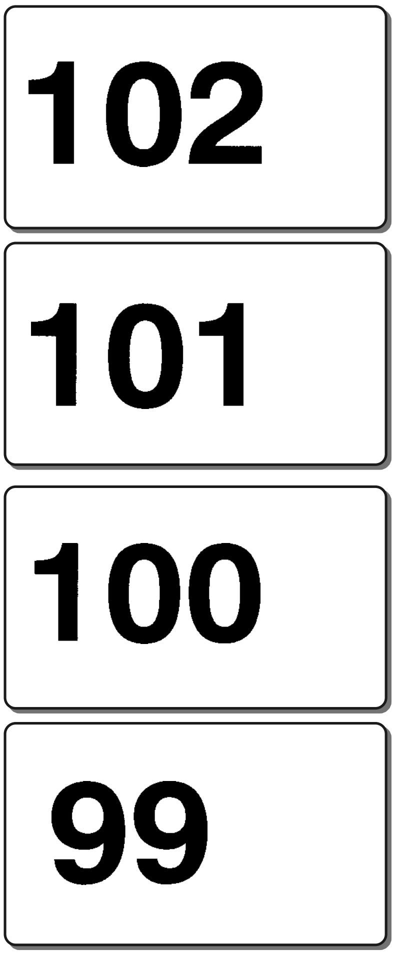
Example: Counter with variable start value

The following example shows a counter which uses a variable start value.
We define 2 invisible (non printable) fields which contain the start value and the counting part.
The mathematical sum of both fields will be printed as result of both fields.
The result is defined without digits behind the comma.

The start value is defined for the keyboard input and will be requested in the printer´s display.
In the example below the start value of 99 was keyed in.

Example:

```
m m
J
O R
S l1;0,0,68,71,100
T:start;0,0,0,5,5;[?:Counter-Start value?][I]
T:offset;0,0,0,5,5;[SER:000][I]
T 10,50,0,5,40;[+:start,offset][C:0][D:1,0]
A 4
```



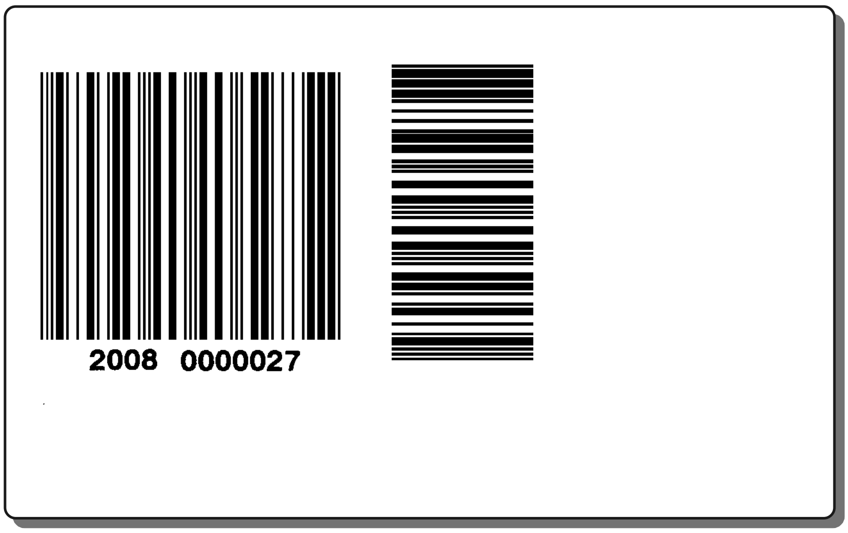
[SER:...] - Serial numbering

The following example shows a label which will be saved on the printers memory card and the variable start value is sent by the attached computer.
Please refer also to the "**M s**“ command which explains how to save labels on a memory card.

Example:

```
Ms LBL;NUMBER
m m
J
H 100,0
S l1;.0,.0,50.0,53.5,70.0
T:YEAR;60.3,4.8,180.0,5,4.0;[YYYY]
T:NR;0,0,0,3,2;0000000[I]
T:OS;0,0,0,3,2;[SER:0000000][I]
T:SER;48.3,4.7,180.0,5,4.0;[+:NR,OS][C:0][D:7,0]
B:BAR2;66.7,43.9,180.0,2of5interleaved+MOD10,35.0,.34,3.0;[YEAR][SER]
B:BAR3;19.9,6.0,270.0,2of5interleaved+MOD10,18.0,.34,3.0;[BAR2]
Ms LBL
A 1[NOPRINT]


Ml LBL;NUMBER
R OS;[SER:0000025]
A 3
```

The Ml command recalls the label,the R command replaces the variable "OS“ and the printer prints 3 labels.



[SER:...] - Serial numbering

Example: Counter with restart from the beginning

The following example shows how to program a counter which restarts after a specific amount of labels.
Here the counter starts at one, counts up until the value „3“ is reached and restarts again counting from „1“. Totally 10 labels will be printed.

Example:

```
m m
J
O R
S l1;0,0,68,71,100
T:COUNTER;0,0,0,5,5;[SER:0][I]
T:MAXLAB;0,0,0,5,5;[%:COUNTER,3][I]
T:RESULT; 30,30,0,5,12;[+:MAXLAB,1][D:2,0]
A 10
```

[SQL:xx] SQL database access

Enables the printer to access a SQL database. This command is used together with the cab database-connector.

It requires that a file has been select first with the command "**E SQL....**". See also the cab database connector section later in this manual.

Syntax: `[SQL:xx]`

[SQL:...] - SQL database access

xx	= any SQL query

e.g. **SELECT** DESCRIPTION **FROM** TABLE **WHERE** SEARCHVALUE='{Fieldname}'

This example below shows a typical request from the SQL database

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

The command [**SPLIT**] can be used if multiple fields are requested. These fields will be delivered, separated by group separators (GS).
[**SPLIT**] helps to separate this content. Please see also the [**SPLIT**] command.

[SQLLOG:...] SQL logging into database

Same function as the **[SQL:xx]** command. SQLLOG will be processed when the label is printed. This enables data logging into a database.

Syntax:

```
[SQLLOG:xx]
```

[SQLLOG:...] - SQL logging into database

xx =	any SQL query

For further information please see the command **[SQL:xx]** and have a view to the cab databaseConnector section later in this manual.

Please note: The maximum length is 128 characters.

[TRIM:...] Trim data

The Trim command can be used to remove space characters at the beginning and at the end of a text line.

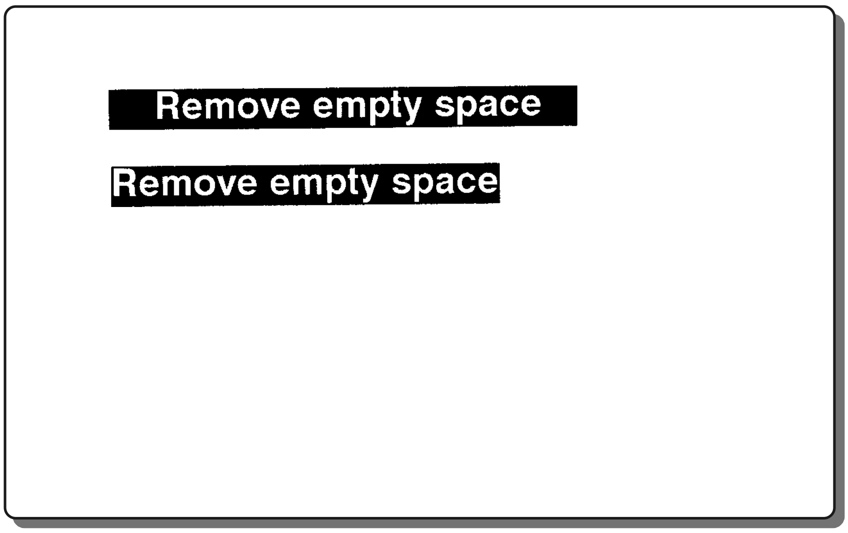
Syntax:

```
[TRIM:x]
```

[TRIM:...]	- trim data
x	= data

Example:

```
m m
J
S l1;0,0,68,70,100
T:CutMe;10,20,0,5,5,n;     Remove empty space
T:CutOff;10,30,0,5,5,n;[TRIM:CutMe]
A1
```



[U:x] Insert Unicode characters

This option inserts UNICODE characters in the data string of your text or barcode fields.

Syntax: `[U:x]`

U - Select unicode character		
x	=	Hexadecimal value, indicated by a dollar sign ($) or ASCII control code name, such as: NUL, SOH, STX, ETX, EOT, ENQ, ACK, BEL, BS, HT, LF, VT, FF, CR, SO, SI, DLE, DC1, DC2, DC3, DC4, NAK, SYN, ETB, CAN, EM, SU, ESC, FS, GS, RS and US or Control codes for Code 128 such as FNC1, CODEA, CODEB, CODEC.

Example:

Some examples:

[U:$20AC] creates the Euro currency symbol
[U:FNC1] creates a function code 1 character (Used for barcode typeCode 128)
[U:$D] or [U:13] creates a carriage return and [U:$A] or [U:10] creates a line feed

All described printers in this manual work internally with Unicode, no special option required.
The availability of unicode characters depends on the selected font.

To recall Thai characters or chinese characters requires that these fonts are installed. Thai can be downloaded free of charge from the cab website at http://www.cab.de . The Chinese font is optional and can be ordered.

[U:x] Insert Unicode characters

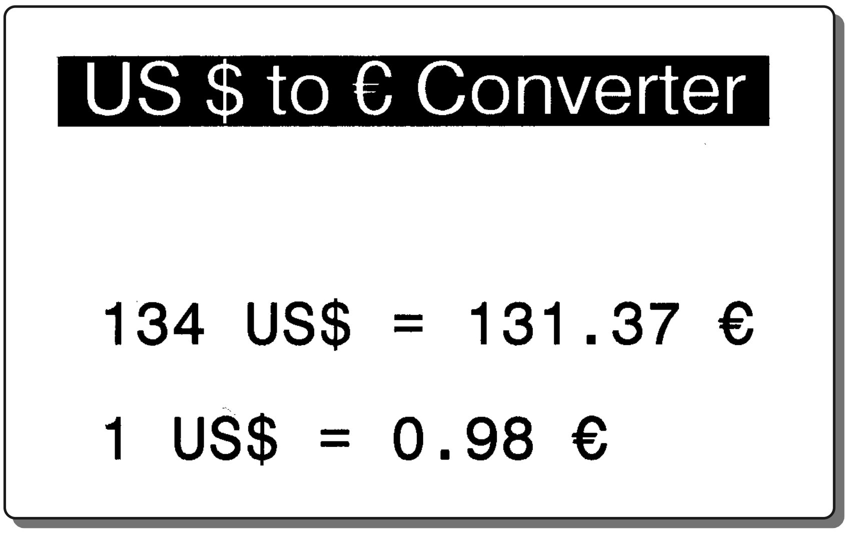
The following example shows a little application which converts US Dollars into Euro (just to show how to recall the Euro sign simply using the unicode feature of cab printers.)

Example:

```
m m
J
S l1;0,0,68,71,100
OR
T:Amount;20,30,0,3,20;[?:Amount in US$:][I]
T:factor;0,0,0,3,3;[?:1 Euro= ? USD][I]
T 5,15,0,3,10,n; US $ to [U:$20AC] Converter
;T 10,30,0,596,8;[Amount] US$ = [*:Amount,factor] US$
T:dollars; 10,60,0,596,8;1 US$ = [/:1,factor] [U:$20AC]
T 10,45,0,596,8;[Amount] US$ = [/:Amount,factor] [U:$20AC]
A1
```

This example starts with a request in the display (attached USB - keyboard recommended), asks for the amount of US Dollars and the converting factor. You may select your preferred exchange rate... (we used 1.02 as factor ...)

Appendix C shows all characters including the unicode values of the built in Truetype fonts.



US $ to € Converter

134 US$ = 131.37 €

1 US$ = 0.98 €

[UPPER:...] Convert to upper case characters

The „upper“ function converts text contents into upper case characters

Syntax:

```
[UPPER:Name]
```

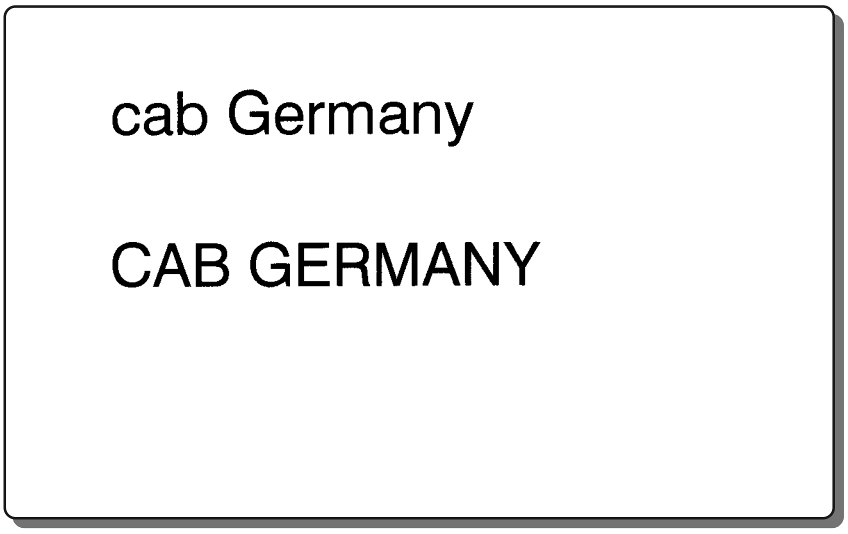
[UPPER:...] - convert to upper case characters

Name	= data - content of a previously defined field (field name)

Example:

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

Prints the field „INPUT“ as it is keyed in, and prints the same data in field „UPPERCASE“ as uppercase characters.



[WINF] Mark a line for writing into the info buffer

[WINF] marks a line to be written in the info buffer. This can be recalled with the "**ESC i**“ command. This value will be set if the label is completely processed.

Syntax:

```
[WINF]
```

[WINF] - Mark line for writing into the info buffer

Example:

```
m m
J
S l1;0,0,68,71,100
T 5,6,0,3,3;[SER:1000,4][WINF]
A500
```

This example prints a label with a counter - starting at 1000 and incrementing by 4. When the label is completely processed, the value of the counter will be written into the WINF buffer.

Completely processed means, that a label in demand mode will write the value into the WINF buffer if it is printed **and** removed from the demand photo cell.

The selected value for the WINF buffer can also be marked as invisible (non-printing) using the [I] command.

Requesting this value can be done with the „ESC i“ command. In our example we would receive the values 1000, 1004, 1008 , 1012 ...... etc.

☞ *This command is useful if it needs to be controlled that the last label has been totally processed before the next label will be sent.*
Please note: The maximum length is 128 characters.

[WLOG] Write LOG file

Writes data to a log file on the memory card. The log file can be is used to keep track of printed labels and can be used to create a report of these data.

Syntax:

```
[WLOG]
```

[WLOG] - Write LOG file

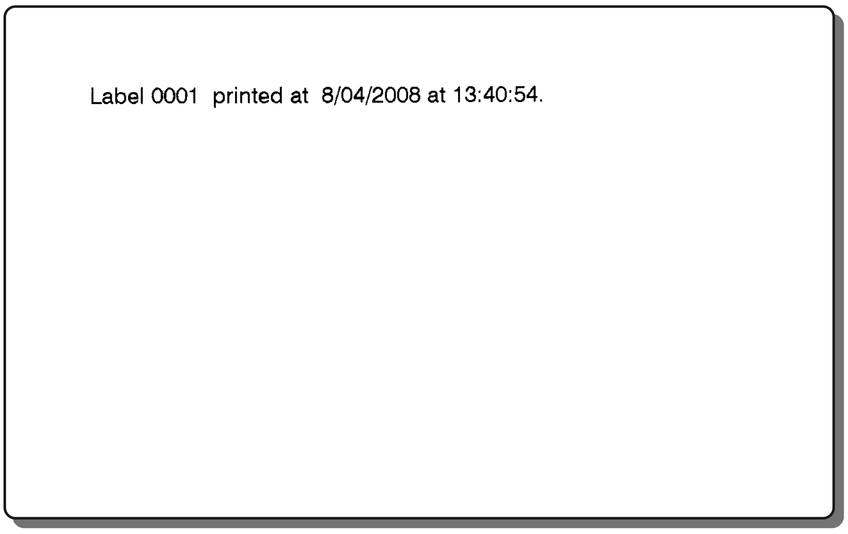
Example:

```
m m
J
S l1;0,0,68,71,100
E LOG;INFO
T:VAL;   5,6,0,3,3;[SER:0001][I]
T:PRINT;5,15,0,3,3;Label [VAL] printed at [DATE] at [TIME].[WLOG]
A3
```

This example keeps track of the labels, based on the counter value VAL which will be written to the LOG file "INFO“. Requires also the command: "**E LOG...**".

Contents of the file INFO.LOG:
Label 0001 printed at 8/04/2008 at 13:40:54.
Label 0002 printed at 8/04/2008 at 13:40:54.
Label 0003 printed at 8/04/2008 at 13:40:55.

Please note: The maximum length is 128 characters. Never switch your printer off while data is written to the memory card.
Loss of information or damage of the memory card would be the result. This command can not be used together with the internal flash file system (IFFS).


Label 0001 printed at 8/04/2008 at 13:40:54.

[WTMP] Write value to serial (TMP) file

Writes a value to a previously defined temporary file on the printer´s memory card.

Syntax:

```
[WTMP]
```

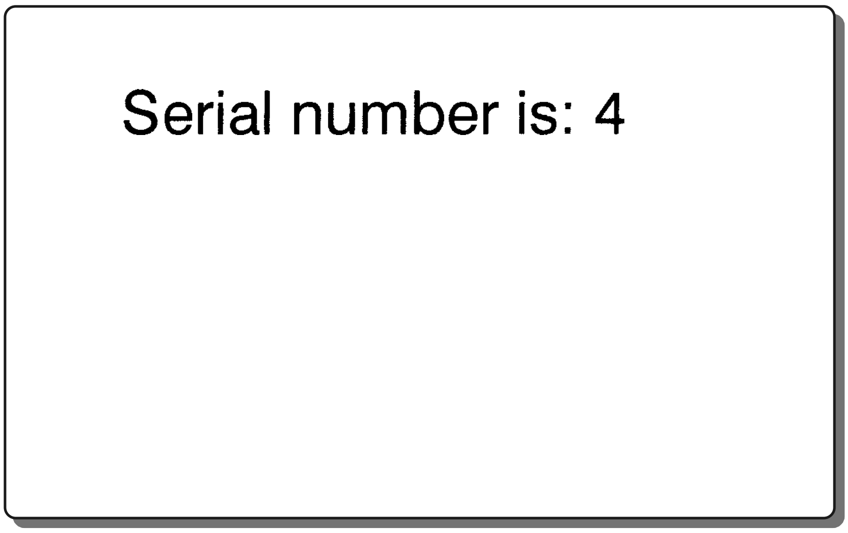
[WTMP] - Write value to serial file

Example:

```
m m
J
S l1;0,0,68,71,100
E TMP;EXAMPLE
T:XVAL;10,10,0,3,3;[RTMP,1][I]
T:SERNO;10,10,0,3,3;[+:XVAL,1][D:0,0][I][WTMP]
T:TESTFELD;10,20,0,3,8;Serial number is: [SERNO]
A4
```

The value of the file EXAMPLE will be saved in the value XVAL.
The value increases in our example in steps of 1 whereby the result is saved on the memory card in the file EXAMPLE.TMP.
EXAMPLE.TMP is located in the „MISC“ folder on the memory card. The value in the example.TMP file is "4" after printing these 4 labels. (The printout shows only the last printed label)

☞ *Please note: The maximum length is 128 characters. This command cannot be used with the internal Flash File system (IFFS). Never switch your printer off while data is written to the memory card. Loss of information or damage of the memory card would be the result. This command can not be used together with the internal flash file system (IFFS).*
See also the command [RTMP] Read value from serial (TMP) file.



[WUSER...] Write value to USER memory

Writes the value into the "user memory“. The function is similar to the **[WTMP]** command, with the exception that only one user file can be used at the same time, the total amount of characters is less, but it requires no additional memory card to read or write a value in the printer.

Syntax:

```
[WUSER]
```

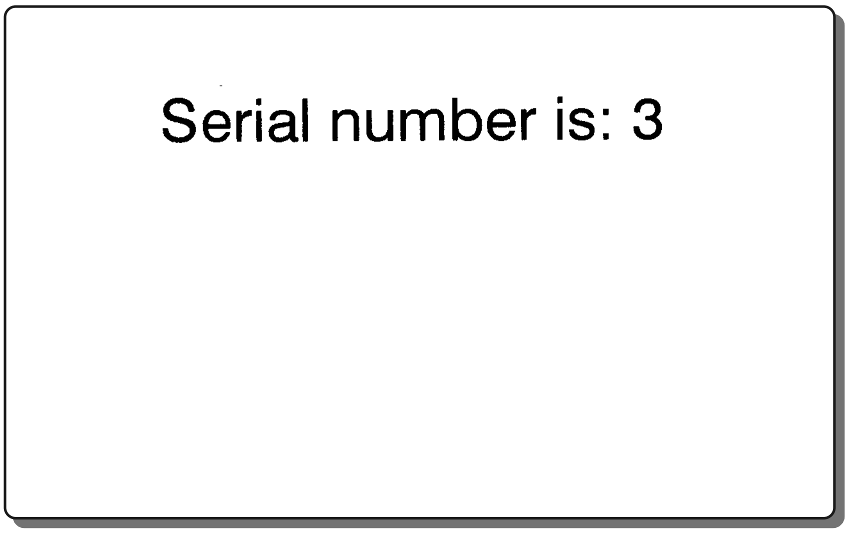
WUSER	- Write into user memory maximum length is 32 bytes

Example:

```
m m
J
S l1;0,0,68,71,100
T:XVAL;10,10,0,3,3;[RUSER,1][I]
T:SERNO;10,10,0,3,3;[+:XVAL,1][D:0,0][I][WUSER]
T:TESTFLD;10,20,0,3,8;Serial number is: [SERNO]
A3
```

This sample prints three labels where the counter counts from 1 to 3. The first label is shown below.

See also the command **[RUSER]** - Read value from user memory.



RFID Functions

The following pages describe special commands which require the additional cab RFID module. RFID modules which have been used with extra port for the RFID control on A- series or A+ series printers do not support these commands.

RFID Functions

[LTAG...]	Lock RFID TAG area
[RTAG...]	Read RFID TAG
[RTAGBIN...]	Read RFID TAG binary
[TAGID]	Read TAG ID
[WTAG...]	Write RFID TAG

[LTAG ...] Lock RFID TAG area

Used to lock some blocks in the RFID Tag.

Syntax:

```
[LTAG:start,len]
```

[LTAG:...] - Lock RFID Tag area	
start	**=** start address (Byte)
len	= length (Byte)

Lock a block of the TAG whereby "start" and "len" are bytes. First address in a TAG is " 0 ". Depending on the tag structure it is only allowed to lock complete blocks, e.g. if the block size is 4 and LTAG is 2, then the complete block will be locked.

Example:

```
mm
J
S l1;0,0,68,70,100
T 10,10,0,3,5;CABRFID[SER:1][WTAG:0][I]
T 10,10,0,3,5;[LTAG:0,8][I]
A1
```

The sample above writes new content to the RFID tag ([WTAG:0]) and locks the content in the next line to avoid that it can be changed.

This function requires that the printer is equipped with the optional cab RFID reader

[RTAG ...] Read RFID TAG

Reads the RFID Tag.

Syntax:

```
[RTAG:start,len]
```

[RTAG:....] - Read RFID Tag	
start	**=** start address (Byte)
len	= length (Byte)

Reads the TAG whereby "start" and "len" are bytes.
First adress in a TAG is " 0 ". Read data are converted in the codepage which had been previously defined with the "E command".

Example:

```
mm
J
S l1;0,0,68,70,100
T 10,10,0,3,5;[RTAG:0,8]
A1
```

Reads and prints the first 8 bytes of a RFID tag.

This function requires that the printer is equipped with the optional cab RFID reader

[RTAGBIN ...] Read RFID TAG binary

Reads the RFID Tag as binary data

Syntax:

```
[RTAGBIN:start,len]
```

[RTAGBIN:...] - ReadRFID Tag binary	
start	**=** start address (Byte)
len	= length (Byte)

Reads the TAG whereby "start" and "len" are bytes.
First adress in a TAG is " 0 ". Read data is handled as binary data without any conversion.

This function requires that the printer is equipped with the optional cab RFID reader

[TAGID] read TAG ID

Shows the value of the read ID of a RFID Tag as HEX value

Syntax:

```
[TAGID]
```

[TAGID]	- readTag ID	**Answer** =	Tag ID

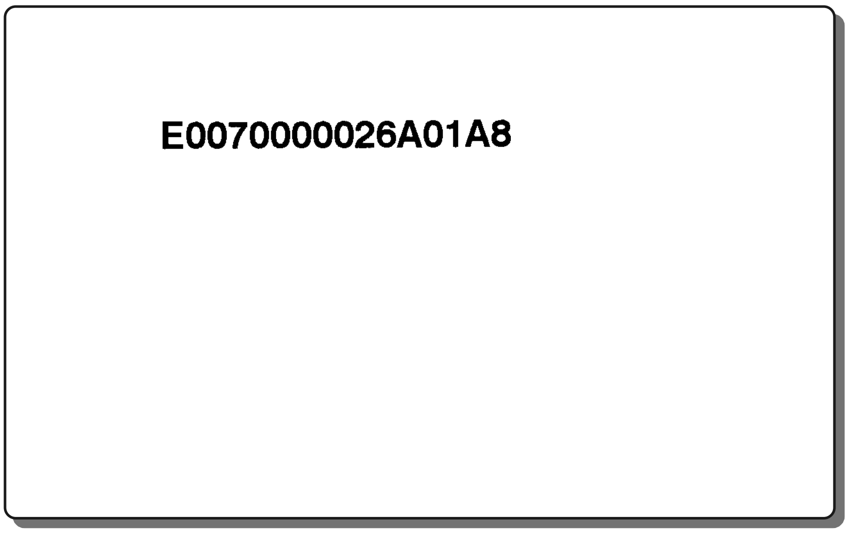
In case of an error the printer responds 00 00 00 00 00 00 00 00

Example:

```
m m
J
S l1;0,0,68,70,100
T 20,20,0,5,5;[TAGID]
A1
```

This example reads the Tag ID of a ISO 15693 tag and prints the ID

This function requires that the printer is equipped with the optional cab RFID reader.



[WTAG ...] Write RFID TAG

Writes the RFID Tag in bytes

Syntax:

```
[WTAG:start{,len}]
```

[WTAG:...] - write tag ID

start	**=**	start address (Byte)
len	=	length (Byte)

Writes the RFID TAG whereby "start" and "len" are bytes. If the content is too short it will be filled up with zero bytes. This command writes blockwise ! If len is missing the printer writes as much as data is available. Start must be devideable through the block size. First address in a TAG is " 0 ".

Writes data in the codepage which had been previously defined with the "E command".

Example:

```
m m
J
S l1;0,0,68,70,100
T 20,20,0,5,5;CABRFID[SER:1][WTAG:0][I]
A1
```

The example writes new content into a tag

This function requires that the printer is equipped with the optional cab RFID reader

cab DataBase Connector commands

Special license needs to be bought to use this functionality

cab Database Connector

This software allows in connection with a printer of the cab A-series, A+ series etc via TCP/IP, to print a label which contains data from a SQL compatible data base. The data is recalled from the printer through its attached keyboard.
A+ series, Mach 4 and the printers with built in network interface (X2 board) need to be activated with a license code. Further information about this procedure can be requested by the cab resellers.

With the methods up to now it was necessary to load databases in a fixed format on a memory card into the printer.
This has the disadvantage that the data has to be converted, they never had been actual and the access time became slower the more the database was growing.
Changings in the central data base required an update on the printers memorycard to have access to the actual data.
cabDatabaseConnector works different. It can recall data form and existing database somewhere in the network. Changes, which are made in this database, are immediately available, if a new label is printed.
The care expenditure for the memory card is no longer needed. The printers can be somewhere in the network. - Theoretically they might be anywhere in the world.

The following components are necessary:

- Printer with X2 mainboard
- A+ series, Mach 4 etc need the license code activation
- Compact Flash memory card recommended
- An input device (USB barcode scanner or USB keyboard)
- cab DataBase Connector software

The cab SQLClient - implemented in the printers - can have access the database server directly on-line through the cab Database Connector and Ethernet TCP/IP.
All data bases with ODBC or a Microsoft OLEDB interface can be accessed.
With cabData Base Connector Server several tables and fields can be queried at the same time.
Multiple predefined labels can be selected through the table of contents of the memory card.

cab DataBase Connector

How it works:

The cab SQLClient contacts the cabDataBasConnector via Ethernet TCP and sends a SQL Query. Cab Database Connector receives the SQL inquiry and sends it via ADO (ActiveX DATA Object) to the database server.

cab Database Connector receives a data record from the database server and sends it via TCP to the cab SQLClient. The cab SQLClient receives the requested data record as a character field.

Supported Databases:

MS ACCESS, Ms SQLServer, Oracle, Dbase and ODBC connections.

Important: Jet40Sp3_Comp.exe and mdac_typ.exe must be installed.
Usually these files are present, if Office 2000 or Windows 2000 is installed.
These files can also be downloaded from www.microsoft.com/data.

cab Database Connector and SQLClient

With the cab Database Connector and the built in SQL client , printers can retrieve data online via Ethernet TCP/IP directly from a database.
When the printer works as a stand alone print station, you do not need to store and maintain the database files on the compact flash cards anymore.

You can access all types of databases with an ODBC driver or a Microsoft ADO-Interface.

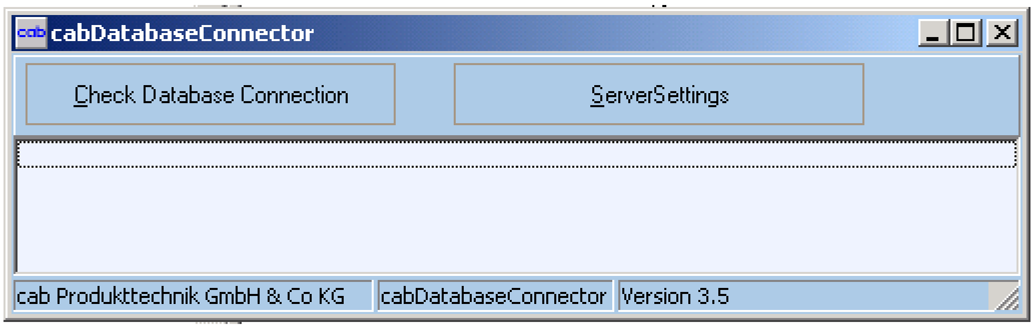
It is now possible to access more than one table and it is much faster than accessing data on the flash card.

cab DataBase Connector

Installation

Step 1
Simply copy the program cabDatabaseConnector.exe on any PC in your network and start it.

The program appears on screen as shown on the picture below.



Step 2
Click on [Server Settings] and type in the complete database connection string. Database connector has an implemented wizard, to help you to find the correct settings. This requires your knowledge about your database !

Sample connectionstrings

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

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

MSSQLServer: Provider=SQLOLEDB.1;Integrated Security=SSPI; Persist SecurityInfo=False;Initial

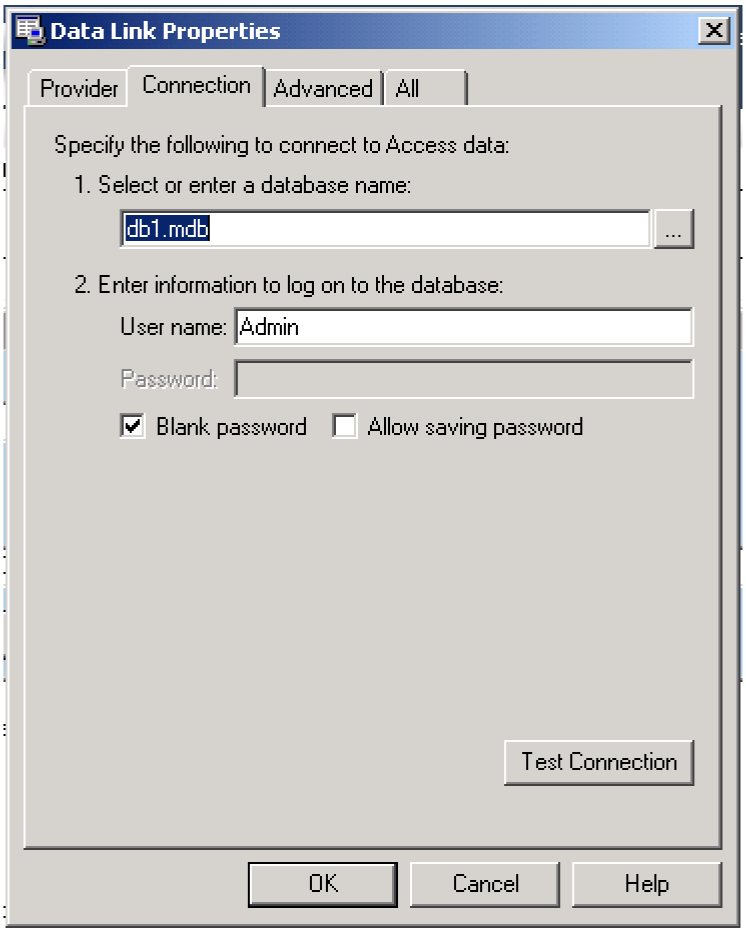
Catalog=cab; Data Source=hostname

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

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

cab DataBase Connector

The connection can be keyed in manually if it is known for the database connection or the built in wizard may be called up which appears in on screen as shown below.



Specify the following to connect to Access data:

1. Select or enter a database name: db1.mdb

2. Enter information to log on to the database:

User name: Admin

Password:

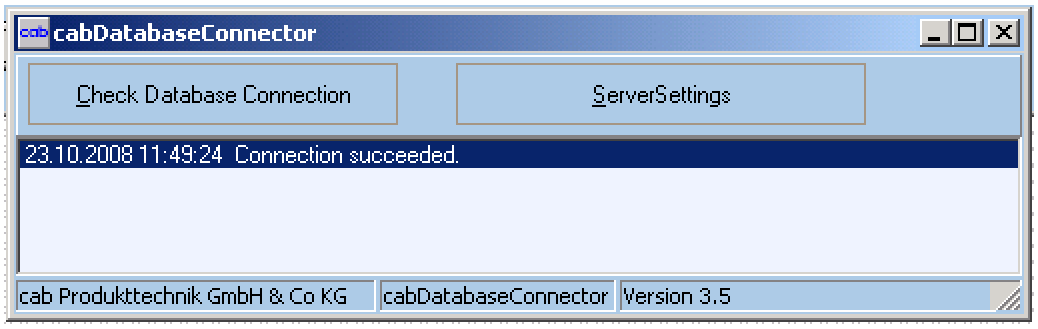
[✓] Blank password [ ] Allow saving password

Details about the wizard are described in the built in help file. You need good knowledge about your data base do a proper setup !

cab Database connector can be started multiple times in a network or multiple times on one PC.

cab DataBase Connector

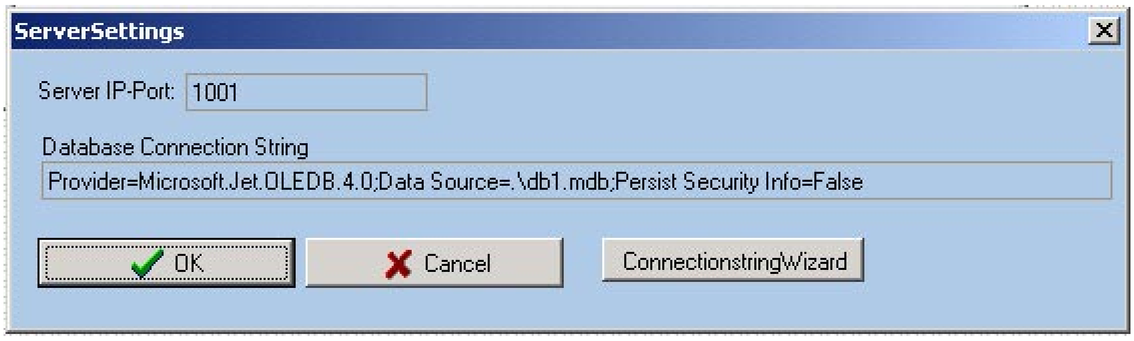
The picture below shows a test of the connection settings, where a Microsoft Access database is connected.



```
23.10.2008 11:49:24  Connection succeeded.
```

cab Produkttechnik GmbH & Co KG | cabDatabaseConnector | Version 3.5

Click on [**Test Database Connection**] to test the datasource.
If DatabaseConnector reports any errors in a popup, then install Jet40Sp3_Comp.exe and mdac_typ.exe. (This is usually only required together with windows 98)
You can download this files at http://www.microsoft.com/data.
If DatabaseConnector reports - Connection open failed- in the list box, then something is wrong with the connectionstring. Correct the connection string.
A sample which connects to a MS Access database is shown on the picture below.



Server IP-Port: 1001

Database Connection String
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=.\db1.mdb;Persist Security Info=False

Step3
Save the prepared label on the memory card of your printer. A sample label is shown on the next pages. Please note that this requires additional commands to get access to your database.

These additional commands are required:

The E-Command: (previously decribed in this manual)

Syntax:	`E SQL;<IP of  cabDatabase connector>:Portnumber`

Defines the IP adress of the computer where cab database Connector is installed. The portnumber can be set in the database connector program itself and must be identical to the port address which is set with the „ E „ command.

cab DataBase Connector

Example:
```
E SQL;192.168.0.80:1001
```

The command sets the connection to the computer with the IP address: 192.168.0.80 where the port number was set to „1001“ in cab database connector program

Required Query-Function:

Syntax:
```
[SQL:Select Field from Table where Searchvalue='{Fieldname}']
```

SQL command language is used to access data from an existing SQL Database

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

The SPLIT - Command:

Syntax:
```
[SPLIT:Field,Index]
```

Example:
```
T 10,5,0,3,5;[SPLIT:RESULT,1]
```

cab DataBase Connector

Following is required to process the example successfully

- Your A-series printer is equipped with a USB keyboard
- An optional memory card must be installed
- The printer must be connected to your network with the special network card !!
- cab database connector has been started and set up correctly.
- The database must be available- we used the table name TA, the database search field name is ARTICLE which is compared with the search value „{ARTNR} „ which is a field name of the label definition. The content of PRODNAME will be recalled from the database
- The following label example must be saved on the optional memory card.

The file below can be recalled from the printers memory card when F1 is pressed on the attached USB keyboard (this recalls the label) and has be followed by the label name

The content of the label is as follows:

Example:

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

Note: The line numbering is used for a better explanation, it does not belong to the program code.

Explanation:

Line *1.*	Selects metric measurement (m m)
Line *2.*	Job start (J)
Line *3.*	select the label size (S l1;.....) - in our case: 68 mm high and 100 mm wide
Line *4.*	print speed (H 200) - here 200 mm/s
Line *5.*	Tells the printer IP and port adress of the device where the database connector is installed. (in our case: IP - adress: 192.168.0.128 and the port adress: 1001)
Line *6.*	Defines a text field which defines the text which will be shown in the display (T:ARTNR.....) - here we ask for a articlenumber in the SQL database. The printer expects here an input which contains a value from the SQL database.
Line 7.	Defines the SQL request and defines also the position and the font of the data field.
Line *8.*	Sets the amount of labels which will be printed. (in our case 1 label)

abc - a-Series basic compiler

An internal basic compiler has been implemented for applications which require more than "only" print commands.
Originally designed for A-series printers (where the name comes from..) -meanwhile also implemented in all actual cab printing sytems and it will be used in future printers - but the name will not change...
The a-series basic compiler is free of charge.

☞ *We highly recommend to update the firmware first before abc is used. The following description is based on the actual firmware release. Please install the actual firmware before you use abc !!!!!*
The actual firmware release can be downloaded from http://www.cabgmbh.com.
The short status or status printout - selectable through the printer´s naviagor pad in the test menu- shows which firmware version is installed.
The usage of abc requires good programming knowledge of the programming language BASIC.

abc is a command subset from Yabasic. Except from the restrictions listed below it is 100% compatible to it, so you can use the original binaries to test your programs under
Windows or Linux (downloads and documentation from www.yabasic.de).

Requirements:

- Running abc needs at least 300 kByte of free memory to work smoothly. Parts of this memory are not being released after finishing the program, so restarting abc is faster.

Restrictions:

- No window and mouse functions
- No PRINT AT
- No COMPILE, no libraries
- No BEEP and BELL, hower the in this manual described printers have SOUND "name" command to play sounds.
- abc and JScript work with cooperative multitasking, i.e. a complex JScript command can delay abc commands and vice versa.
- The content of a file has priority over abc output to JScript.
  This way abc can e.g. send „M l lbl;sample“ to JScript. However this means that when a file is executed from card abc output is delayed until the file has been completely read and closed by Jscript!

Import differences to Yabasic PC versions:

- To switch off the ESC command interpretation of JScript you can use POKE „transparent“, 0 or 1.
  However all data which is already in the input buffer (64 kwords) has been filtered. So do not send data with ESC in it before the POKE command has been executed!
- abc works internally with Unicode, so multilingual data processing is no problem for abc programs.
  abc can also handle chr$(0) within a string which is interpreted as string end in yabasic.
- Programs can be stopped by total CANCEL (pressing CAN more than three seconds on front panel), this can be disabled by ON INTERRUPT command.
- No SYSTEM$() function.

Temporary restrictions/known bugs:

- Printing ESC sequences to JScript has no effect

Window-Handling:

abc uses a hidden window which can be (partially) mapped to the front panel LCD. The printer handles the window as a bitmap with 8 bit indexed colours. So each dot can have a value of 0 (black) to 255 (white). During mapping to the LCD, each colour is mapped according to its brightness which is predefined as grayscales, i.e. 128 to 255 gives white pixels, 0 to 127 black pixels.
The mapping can be changed with the POKE command to RGB colors which are useful if you want to write the graphic to the card.

- ‘OPEN WINDOW width, height’ opens the window. Only one is allowed. As this window is stored internally in standard memory, define it only the size you really need. (E.g. a window 100,100 takes 10kByte memory). For the front panel’s LCD a window of 120 by 32 is sufficient. (depending on the display of the printer type)
- There’s only one font (16 dots high), variable width with support of latin, greek, cyrillic, hebrew and arabic scripts. The origin is in the upper left corner of the first character’s bounding box. For right-to-left writing countries, the origin is in the upper right corner.

New functions compared to Yabasic:

- **POKE** „color#“,rgb, #=1 to 254, 0 stays always black, 255 stays always white,
  e.g. POKE „color#15“,dec(„ff0000“) sets color no. 15 to red.
- **WINDOW TRANSFER TO „name“** transfers the window content to a JScript image „name“ which can be used e.g. with the I command.
- **WINDOW TRANSFER FROM** „name“ loads the window with a JScript image. If the windows and image size are not identical the result is clipped.
- **WINDOW WRITE TO** „name“ saves the actual window as PNG on the memory card.
- **WINDOW READ FROM** „name“ load a PNG into the actual window. Path names are allowed here.
  The window has to be big enough to hold the image, else loading will fail! Supported formats are:
  - grayscale 1 to 8 bits per pixel
  - paletted images 8 bits per pixel

- **JGET$** and **JPUT** are used to exchange data between JScript and abc. The exchange is synchronized, so you can use abc as JScript function. Use always as a pair, else execution of JScript and / or abc can be blocked !
- abc has a command check for the existence of files or devices:
  EXISTS ("filename" or EXISTS("/dev/rawip")

Restrictions compared to Yabasic:

- No CIRCLE command.
- No BITBLT, GETBIT$ and so on.
- WINDOW ORIGIN is not supported, i.e. the origin 0,0 is always in the upper left corner.

The modifiers **CLEAR** and **FILL** have the following results (shown for the RECT command):

RECT:	frame in foreground color
CLEAR RECT:	frame in background color
FILL RECT:	filled area in foreground color
CLEAR FILL RECT:	filled area in background color

abc - PEEK Variables:

command	type: (**S**=string, **I**=integer, **F**=float)	description
„os“	S	Delivers „cab A-Series“ - only for compatibility with Yabasic
„version“	F	Version of Yabasic
„resolution“	F	Resolution of printer in dpi
„width“	F	Maximum print width in mm
„transparent“	I	Value: 0 or 1. 1 switches off ESC-command interpretation
„mlength“	F	measured length of last label distance (mm), if not known it is 0
„direction“	I	direction of paper move 1 if forward, -1 if backward and 0 if standing
„slength“	F	stored label distance (mm), if not known or invalid it is 0. This is effectively the distance of the last defined label before being switched off
„imageheight:name“	I	gives the height of an image „name“ in dots, 0 if not known
„imagewidth:name“	I	gives the width of an image „name“ in dots, 0 if not known
„freememory“	I	gives the free main memory (available for abc or Jscript)
„status“	S	state of the printer (same as ESC s answer string)
„xstatus“	S	Extended state of the printer (same as ESC z answer string, but without CR)
„xinput“	I	status of the peripheral connector input pin (XSTART)
„xoutput“	I	reads actual peripheral control bits
„line“	I	number of the actually printed label

abc - PEEK Variables:

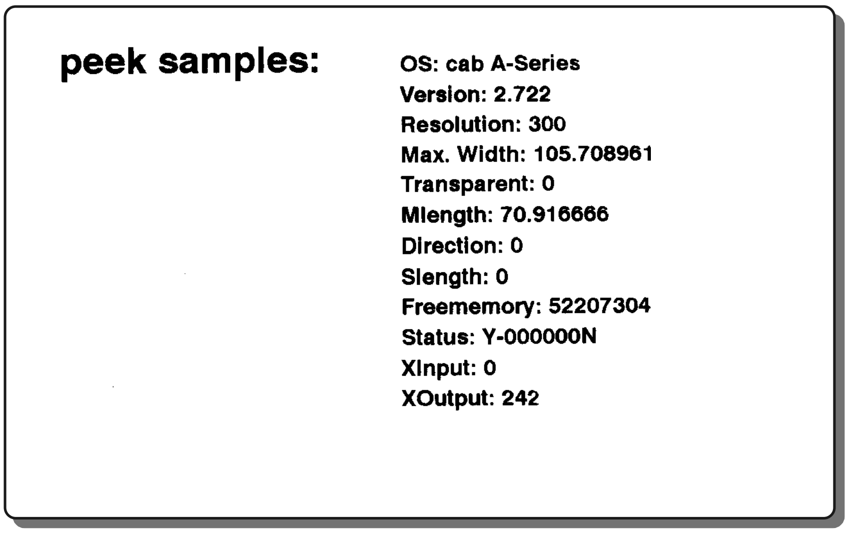
command	type: (**S**=string, **I**=integer, **F**=float)	description
„**jphase**“	I	Phase of JScript-Interpreter: **0** waiting for label definition **1** in process of label definition **2** during printing **3** standby, waiting for new job or new data for old one
„**source**“	S	Name of last data source: „RS232“, „RS422“, „RS485“, „IEEE1284“, „RAWIP“, „USB“, „FTP“, „LPD“, „unknown“
„**ticks**“	I	timer tick since startup of printer in 1/128th seconds
„**sec70**“	I	time in unix format - i.e. seconds since Jan 1, 1970.
„**peri**“	S	Returns name of peripheral (similar to JScript " **q p**" command
„**winf**“	S	Returns the contents of the WINF buffer (similar to the ESC i command)
„peelpos“	I	Returns a 1 if the label is in peel-off position.
„manufacturer“	S	Returns the manufacturer of the machine (e.g. „cab“).
„machine“	S	Returns the type and name of the printer (e.g. „A4+/300“).
„firmware“	S	Returns the firmware version of the machine („e.g. „V3.05 (Sep 13 2006)“)
„iobox“	I	Returns the input state of the I/O box on USB. Returns -1 if not available. Input data is binary ORed, values ranging from 1 for input 1 to 8 for input 4.
„serial“	S	Returns the serial number of the PCB.
„user“	S	Returns the content of the non-volatile user space (A+/Mach only).
„rfid_rssi“	I	Returns the signal quality of a detected RFID tag. Range is 0 to 100.

abc - PEEK Variables:

The following example uses a few of the Peek variables and prints the result on a label

Example:

```
<ABC>
     a$=peek$("os")
     b=peek("version")
     c=peek("resolution")
     d=peek("width")
     f=peek("mlength")
     g=peek("direction")
     h=peek("slength")
     i=peek("freememory")
     j$=peek$("status")
     k=peek("xinput")
     l=peek("xoutput")
   print "m m"
   print "J"
   print "O R"
   print "S l1;0,0,68,70,100"
   print "T 5,8,0,5,5;peek samples:"
   print "T 50,8,0,5,3;OS: ",a$
   print "T 50,12,0,5,3;Version: ",b
   print "T 50,16,0,5,3;Resolution: ",c
   print "T 50,20,0,5,3;Max. Width: ",d
   print "T 50,24,0,5,3;Transparent: ",e
   print "T 50,28,0,5,3;Mlength: ",f
   print "T 50,32,0,5,3;Direction: ",g
   print "T 50,36,0,5,3;Slength: ",h
   print "T 50,40,0,5,3;Freememory: ",i
   print "T 50,44,0,5,3;Status: ",j$
   print "T 50,48,0,5,3;XInput: ",k
   print "T 50,52,0,5,3;XOutput: ",l
   print "A 1"
</ABC>
```



```
peek samples:        OS: cab A-Series
                     Version: 2.722
                     Resolution: 300
                     Max. Width: 105.708961
                     Transparent: 0
                     Mlength: 70.916666
                     Direction: 0
                     Slength: 0
                     Freememory: 52207304
                     Status: Y-000000N
                     XInput: 0
                     XOutput: 242
```

abc - POKE Variables:

command	type: (**S**=string, **I**=integer, **F**=float)	description
„xoutput“	I	status of the peripheral connector control bits (output) Note: you have to set the peripheral mask to 0 (x m command) before!
„read_controls“	I	Value: 0 or 1. 1 allows control characters to pass thru INPUT or INKEY$. All characters are passed to abc, including the character terminating the input line (e.g. CR). (This CR can be removed e.g. with TRIM$.)
„bypass“	I	Value:0 or 1. 1 allows data from interfaces to go directly to JScript.
„httpswap“	S	Can be used to swap the normal root directory and the memory card on the webserver. E.g. POKE „httpswap“,“/secret“ moves the applet to /secret/index.htm and /card/index.htm to /index.htm.
„lcd“	I	Controls the source for the LCD. 0 is standard, JScript content. 1 is the abc window.
„lcdx“, „lcdy“	I	Offset for the LCD in the abc window.
„led“	I	Controls the state of the front panel LEDs (if „lcd“ is 1). Bit coded: 1 = Cancel 2 = Mode (A-Series), Error (M-Series) 4 = Feed 8 = Pause 16 = Arrows (A-Series only) A+/Mach4 and newer machines: 1=Menu 2=Cancel 4=Feed 8=Pause 16=Enter 32=Up arrow 64=Left arrow 128=Right arrow 256=Down arrow
„ledmask“	I	Masks the LEDs to be lit. Independent of „lcd“-value. Same bit coding as „led“. A 0 masks the respective LED.
„backlight“	I	Controls the backlight of the LCD if „lcd“ is 1. 1 is on, 0 is off, 2 is controlled by JScript (Default).

abc - POKE Variables:

command	type: (**S**=string, **I**=integer, **F**=float)	description
„fcolor“, „bcolor“	I	Sets the fore- and background colors for abc window operations.
„color#x“	I	Sets the RGB value for color #x. x is valid from 1 to 254. Color 0 (black) and 255 (white) cannot be modified.
„nice“	I	Sets the multitasking priority of abc vs. JScript. Ranges from 1 (JScript fast) to 20 (abc fast). Default is 10.
„key“	I	Puts a character into the key buffer. E.g. POKE „key“,dec(„F001“) simulates pressing the MODE key.
„winf“	S	Writes a value into the „winf“-Buffer.
„print_with_verify“	I	Controls the usage of a barcode scanner by the printengine of an enabled machine. Set to 1 for the printengine to wait for „scanresult“ after each label.
„scanresult“	I	Sets the result of the barcode verification scan: 1 Good, apply the label 2 Bad, display error (depending on user decision on front panel reprint will occur or not) 3 Bad, keep label on liner (reprint will occur) 4 Bad, put label in recycle position (if hardware available, reprint will occur) 5 Bad, put label on product (reprint will occur) 3+8 Bad, keep label on liner (no reprint) 4+8 Bad, put label in recycle position (if hardware available, no reprint) 5+8 Bad, put label on product (no reprint)
„widget“	S	Puts text into abc debug widget. Up to four characters printable (only digits and upper case letters). (Only available on A+/Mach4 machines.)
„iobox“	I	Sets the outpuf state of the I/O box on USB. Returns error if not available. Output data is binary ORed, values ranging from 1 for output 1 to 8 for output 4.
„wakeup“	I	Wakes the printer resp. prevents it from falling asleep.
„user“	S	Writes a value into the non-volatile user space (A+/Mach only). Max. 31 UTF-8 characters allowed.
„syserror“	S	Puts the first character of the string into the error message puffer. Allowed characters are the same as in the ESC s response.

abc - Streams:

Filename	Direction/Bit	Description
„/dev/rs232:baud,handshake“	**I/O,8**	Baud: 1200-230400, handshake: -,RTS/CTS,XON/XOFF
„/dev/ieee1284“	**I/O,8**	bidirectional parallel interface
„/dev/rs422:baud,handshake“	**I/O,8¹**	RS-422 interface, baud: 1200-230400, handshake: -,XON/XOFF
„/dev/rs485:baud,address“	**I/O,8 r**	RS-485 interface, baud: 1200-230400, address: A-Z
„/dev/usb“	**I/O,8°**	USB-Client
„/dev/rawip“	**I/O,8**	raw-IP interface
„/dev/lpr“	**I,8°**	lpr server
„/dev/panel“	**I,16**	input from front panel keys, key values are $F001 Mode $F002 Formfeed $F003 Cancel $F004 Pause $F090 Cancel longer than 3 seconds
„/dev/keyboard“	**I,16**	input from external keyboard *There are too many keycode to list them here - please use the program listed in the sample section of this document.*
„/dev/jscript“	**I,16**	JScript-Interpreter - needed for reading back answers
„/card/filename.ext“	**I/O* ,8/16**	file from memory card
„/iffs/name.ext“	**I,8/16**	file from internal memory
„mailto:address“	**O,8**	Writes an email to the specified address. An SMTP-Server address and a return address has to be set in the setup! The subject is the first line printed into the stream.

***** no random writing within a file, only append or overwriting, according to the filename extension the files are automatically sorted into the appropriate directories (i.e. /images, /labels, /fonts and /misc) on the card

° not yet implemented

¹ note: on A3 setting the baudrate on RS-422 sets the RS-232 baudrate too and vice versa!

abc - **Modes:**

„r“, „w“, „a“	read, write and append (file reading and writing automatically transforms Unicode to ASCII and vice versa according to selected codepage, reading a Unicode or ASCII file is automatically detected)
„rb“, „wb“, „ab“	read, write and append without transforming (file reading and writing uses only low-byte of e.g. string)
„wu“, „au“	write and append using Unicode

Notes:

- Some streams like „/dev/panel“ are always Unicode-streams. Using ‘b’ or ‘u’ modifiers can have strange effects!
- Writing to an interface (e.g. /dev/rs232) will fail if the printer cannot send the data. There’s a time out of 10 seconds.
- Opening an interface as file stops ESC interpretation on this device.
- abc has an additional command called FLUSH which enables you to clear the input buffer of /dev-streams in read mode (e.g. FLUSH #1 when 1 ist /dev/rawip). FLUSH #0 clears standard input.
- abc has an additional command to erase files: ERASE „name“.

Communication with Web Browsers:

cab printers have a web server which is usually used for administration, but can also be used to access data like images or HTML pages from the card. So it is only logical to seek a way to transmit data from the browser *to* the printer. This is normally done by CGI scripts using forms. We do it the same way :-) You can however not define CGI scripts your own, but we provide a way to get form data into your abc program:

HTML

You simply define a form in your HTML page which uses get_form.cgi as ACTION.

```
<form action=“/get_form.cgi“ method=“post“>
<input type=“hidden“ name=“nextpage“ value=“thanks.htm“>
<input type=“text“ name=“example“>
<input type=“submit“ value=“Send data“>
</form>
```

This form lets the user enter some data in a text field called „example“. After clicking the „Send data“ button, the form content is sent from the browser to the web server and parsed there. Then the extracted data is put into the input buffer which can be read by abc or directly by JScript. There are two special field names available:

- **nextpage** this defines the name of the html page which is loaded after sending the form. Default is index.htm.
- **jscript** Can be used to send a JScript command before the data. So you can e.g. send a „M l lbl“ command before the data of the form.

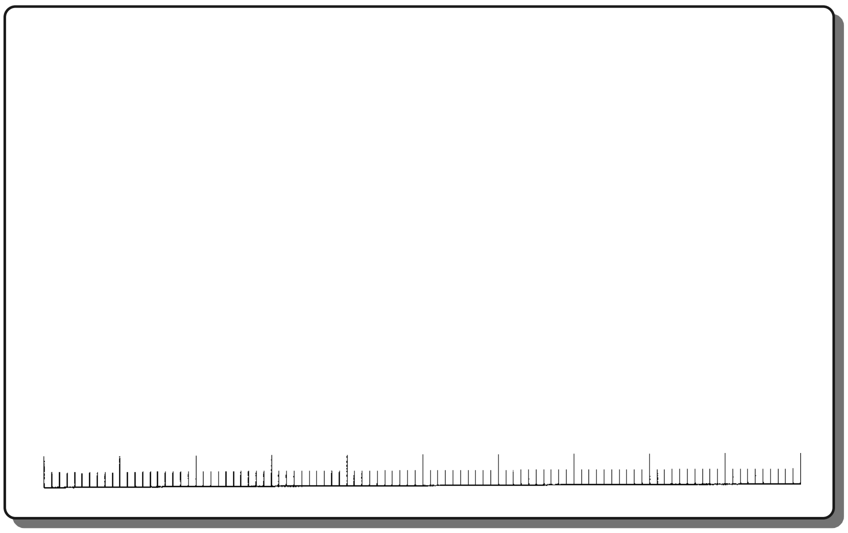
A more complex example showing most of the possibilities of the CGI interface is the „cinema ticket“ program. This is available on request. In this case you can contact „support@cabgmbh.com“

a-Series basic compiler

Small program to print a 100mm long ruler with 1mm markings:

Example:

```
; Test label for ruler
m m
J
S l1;0,0,68,71,104
G 0,10,0;L:100,.1
<ABC>
FOR X=0 TO 100
   IF MOD(X,10) = 0 THEN
     PRINT "G ",X,",10,270;L:4,.1"
   ELSE
     PRINT "G ",X,",10,270;L:2,.1"
END IF
NEXT X
END
</ABC>
A 1
```

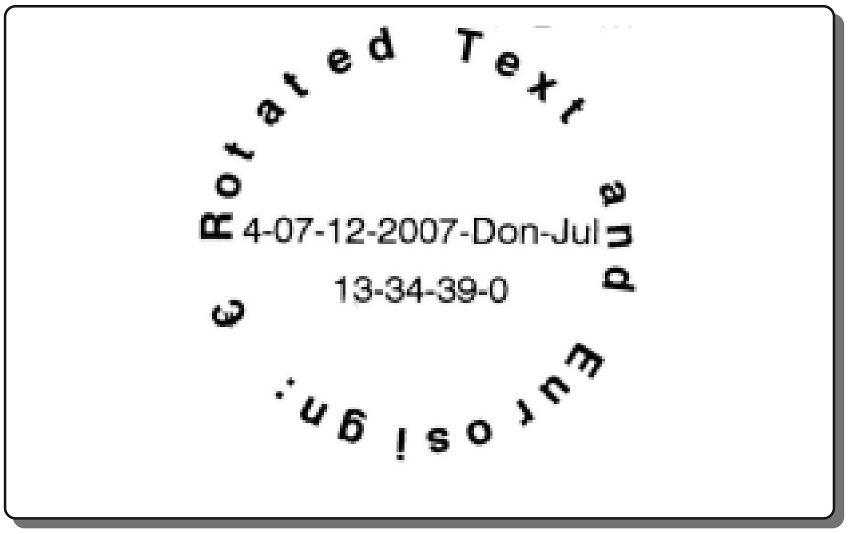


a-Series basic compiler

Small program to print a text in a circle:

Example:

```
; Test label for rotated text
J
S l1;0,0,68,71,104
<ABC>
A$="Rotated text with Euro sign: "+CHR$(DEC("20AC"))+" "
N=LEN(A$)
D=360/N
FOR I=1 TO N
   W=((I-1)*D)/180*PI
   X=50-25*COS(W)
   Y=30-25*SIN(W)
   R=90-(I-1)*D
   IF R<0 THEN
     R = R + 360
   ENDIF
   PRINT "T ",X,",",Y,",",R,",3,6,b;",MID$(A$,I,1)
NEXT I
PRINT "T 0,30,0,3,5;[J:c100]",date$
PRINT "T 0,38,0,3,5;[J:c100]",time$
END
</ABC>
A 1
```



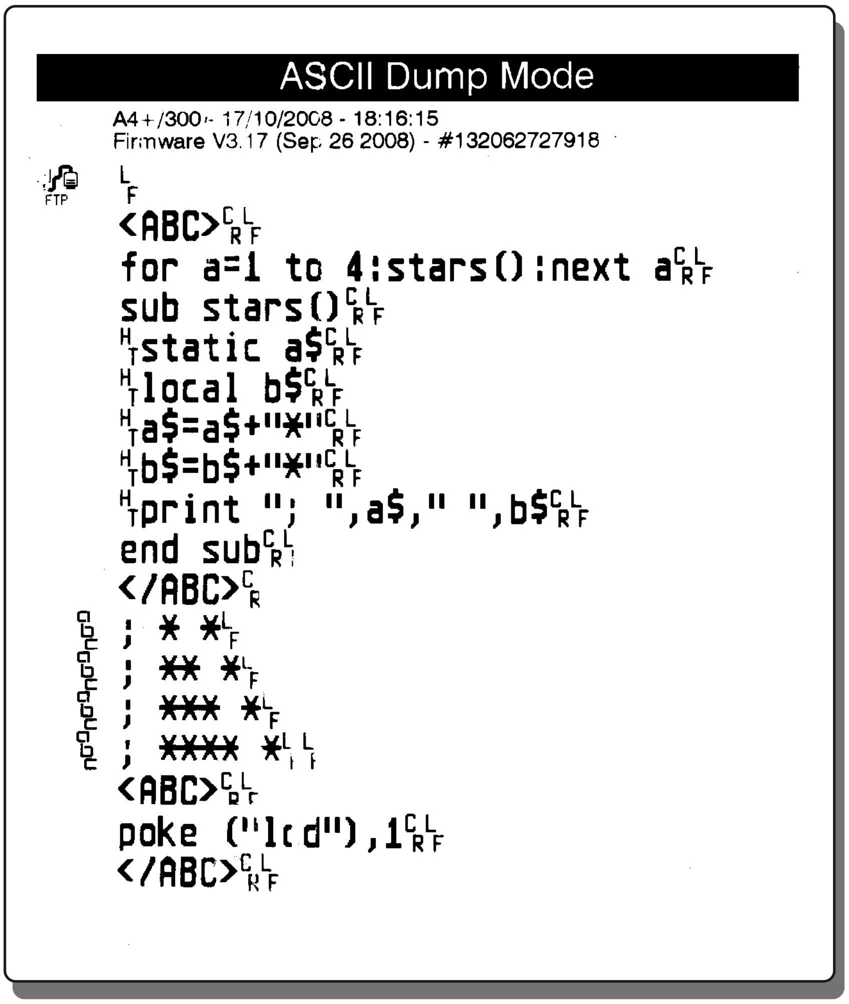
a-Series basic compiler

Small program to show usage of local and static variables.

Uses ASCII dump mode to show what happens:

Example:

```
a
<ABC>
for a=1 to 4:stars():next a
sub stars()
   static a$
   local b$
   a$=a$+"*"
   b$=b$+"*"
   print "; ",a$," ",b$
end sub
</ABC>
```



ASCII Dump Mode

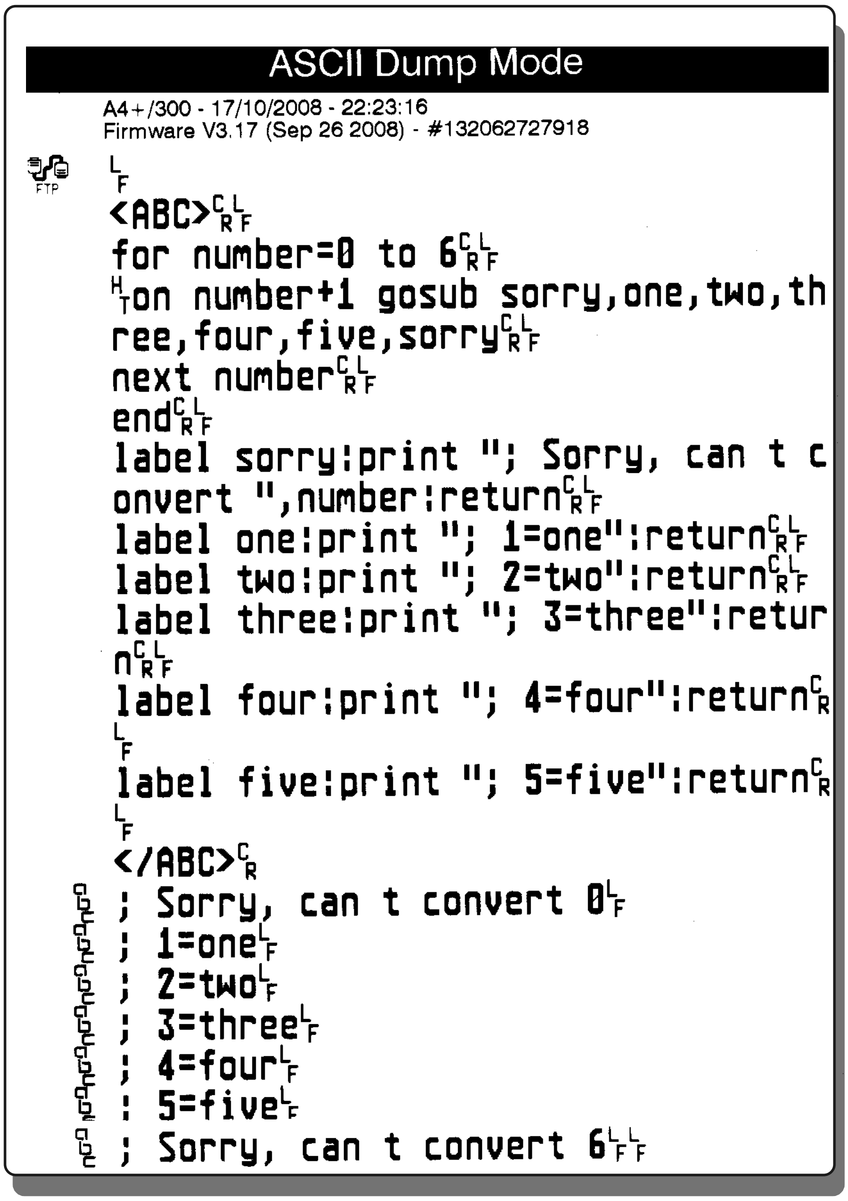
A4+/300 - 17/10/2008 - 18:16:15
Firmware V3.17 (Sep 26 2008) - #132062727918

```
LF
<ABC>CRLF
for a=1 to 4:stars():next aCRLF
sub stars()CRLF
HTstatic a$CRLF
HTlocal b$CRLF
HTa$=a$+"*"CRLF
HTb$=b$+"*"CRLF
HTprint "; ",a$," ",b$CRLF
end subCRLF
</ABC>CR
; * *LF
; ** *LF
; *** *LF
; **** *LFLF
<ABC>CRLF
poke ("lcd"),1CRLF
</ABC>CRLF
```

Small program to show ON GOSUB. Uses ASCII dump mode to show what happens:

Example:

```
a
<ABC>
for number=0 to 6
   on number+1 gosub sorry,one,two,three,four,five,sorry
next number
end
label sorry:print "; Sorry, can’t convert ",number:return
label one:print "; 1=one":return
label two:print "; 2=two":return
label three:print "; 3=three":return
label four:print "; 4=four":return
label five:print "; 5=five":return
</ABC>
```



ASCII Dump Mode

A4+/300 - 17/10/2008 - 22:23:16
Firmware V3.17 (Sep 26 2008) - #132062727918

```
␊
<ABC>␍␊
for number=0 to 6␍␊
␉on number+1 gosub sorry,one,two,th
ree,four,five,sorry␍␊
next number␍␊
end␍␊
label sorry:print "; Sorry, can t c
onvert ",number:return␍␊
label one:print "; 1=one":return␍␊
label two:print "; 2=two":return␍␊
label three:print "; 3=three":retur
n␍␊
label four:print "; 4=four":return␍
␊
label five:print "; 5=five":return␍
␊
</ABC>␍
; Sorry, can t convert 0␊
; 1=one␊
; 2=two␊
; 3=three␊
; 4=four␊
; 5=five␊
; Sorry, can t convert 6␊␊
```

a-Series basic compiler

Small program to show READ,DATA and RESTORE. Use ASCII dump mode to show what happens:

Example:

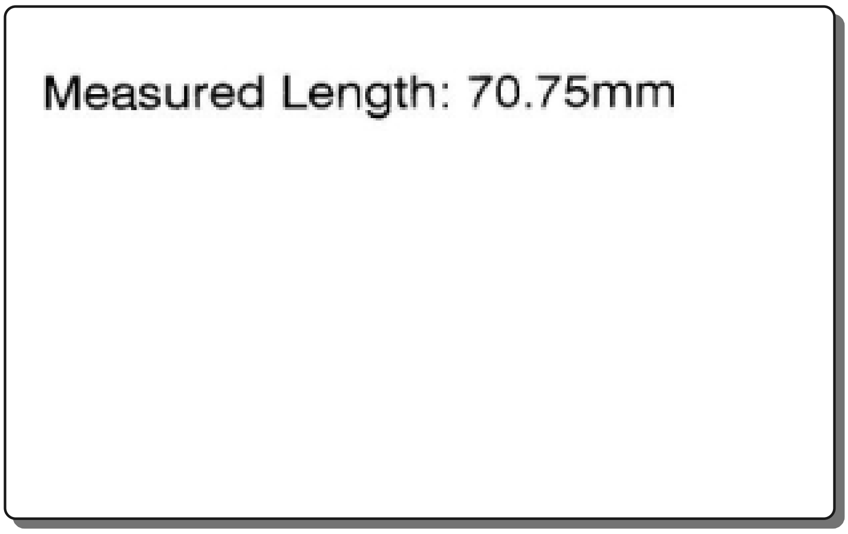
```
a
<ABC>
restore names

read maxnum
dim names$(maxnum)
for a=1 to maxnum:read names$(a):next a
for number=0 to 10
   if (number>=1 and number<=maxnum) then
     print „; „,number,“=“,names$(number)
else
     print „; Sorry, can’t convert „,number
   endif
next number
error „Program finished“
label names
data 9,“one“,“two“,“three“,“four“,“five“,“six“
data „seven“,“eight“,“nine“
</ABC>
```

a-Series basic compiler

Small program for measuring the label distance:

Example:

```
<ABC>
DO
   REM read measured distance
   dy=PEEK(„mlength")
   IF dy>0 BREAK
   PRINT „f"
   WAIT 0.25
   REM wait until standing again REPEAT
   REPEAT UNTIL (PEEK(„direction")=0)
LOOP
PRINT „J"
PRINT „S l1;0,0,",dy-2,",",dy,",100"
PRINT „T 0,10,0,3,5;Measured label distance: „,dy,"mm"
PRINT „A 1"
</ABC>
```



This program demonstrates the differences for file handling (a compactflash drive and a hex editor are useful to see the difference):

Example:

```
<ABC>
a$="Hello „+CHR$(DEC(„20AC"))
OPEN 1,"test.dat","w"
PRINT #1 a$
CLOSE 1
OPEN 1,"testu.dat","wu"
PRINT #1 a$
CLOSE 1
OPEN 1,"testb.dat","wb"
PRINT #1 a$
CLOSE 1
</ABC>
```

a-Series basic compiler

This program does also writing using files but on the RS-232:

Example:

```
<ABC>
a$="Hello „+CHR$(DEC(„20AC"))
OPEN 1,"/DEV/RS232:57600,RTS/CTS","w"
PRINT #1 a$,chr$(13);
FOR i=1 TO 10
PRINT #1 i,chr$(13);
NEXT i
CLOSE 1
</ABC>
```

This demonstrates the file path and name handling of abc (it is necessary to have test.dat on the card, e.g. from the last demo program):

Example:

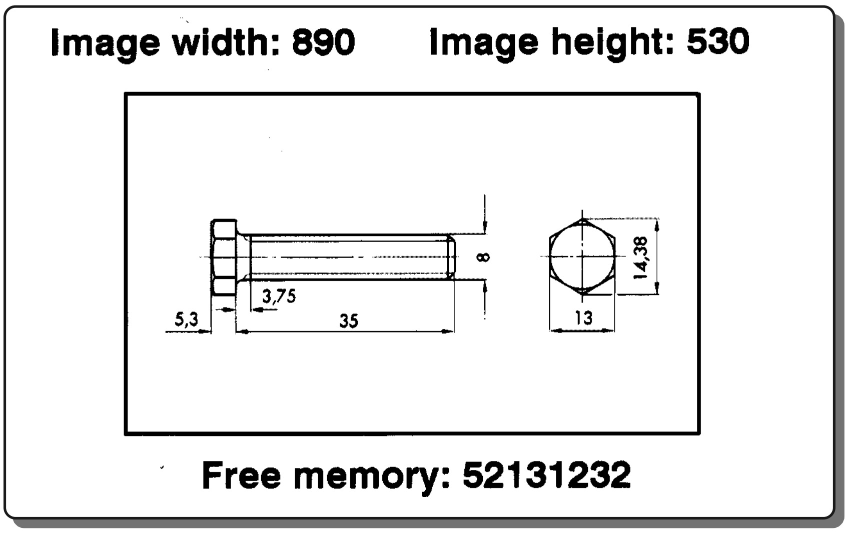
```
<ABC>
PRINT „a"
PRINT „; test.dat: „,exists(„test.dat")
PRINT „; test.dat: „,exists(„TEST.DAT")
PRINT „; test.dat: „,exists(„/card/misc/test.dat")
PRINT „; test.dat: „,exists(„/CARD/TEST.dat")
PRINT „; test2.dat: „,exists(„test2.dat")
</ABC>
```

If you want to know the dimensions of an image try this:

Example:

```
<ABC>
PRINT"M l img;screw"
w=0
h=0
DO
w=PEEK("imagewidth:SAMPLE")
h=PEEK("imageheight:SAMPLE")
IF w>0 AND h>0 BREAK
LOOP
PRINT "J"
PRINT "H 75,10"
PRINT "S l1;0,0,68,70,100"
PRINT "T 0,8,0,5,5;Image width: ",w
PRINT "T 50,8,0,5,5;Image height: ",h
PRINT "T 20,64,0,5,5;Free memory: ",PEEK("freememory")
PRINT "I 10,13,0;screw"
PRINT "A1"
</ABC>
```

This sample shows the image size of the previously downloaded image in pixels



```
Image width: 890      Image height: 530

Free memory: 52131232
```

a-Series basic compiler

Simple program to show the capture of interface data, parsing it, extracting the data and sending it forward to the JScript interpreter:

Here we convert data which drives another printer model into data which will be understood by a cab printer. The incoming data is shown on the next page. The program runs in a loop, always ready to receive new data.
The label is prepared first in JScript, then incoming data is analysed and finally we replace the field contents with the extracted data.

Example:

```
J
S l1;0,0,68,71,104
T:t1;20,10,0,3,8;
T:t2;20,20,0,3,8;
T:t3;40,40,0,3,8;
<ABC>
label start
line input a$
if left$(a$,15)="194300301480070" then
   print „R t2;",mid$(a$,16)
endif
if left$(a$,15)="194300300580172" then
   print „R t3;",mid$(a$,16)
endif
if left$(a$,15)="194300301970073" then
   print „R t1;",mid$(a$,16)
endif
if a$="Q0001" then
   print „A 1"
endif
goto start
</ABC>
```

☞ *Please see also further information on the next pages*

This is the original data that had been sent by a labelling software:

The data below produced the same printout on another label printer.

```
M3000
<STX>d
<STX>e
<STX>f260
<STX>O0220
<STX>V0
<STX>L
D11
PA
SA
H10
z
194300301480070Rot
19430030058017248
194300301970073Bernd
W
Q0001
E
<STX>L
D11
PA
SA
H10
z
194300301480070gelb
19430030058017248
194300301970073Bertha
W
Q0001
E
```

Program to read keyboard codes:

Example:

```
<ABC>
OPEN 1,"/dev/keyboard","r"
OPEN WINDOW 120,32
POKE „lcd",1
DO
 DO
  x=PEEK(#1)
  IF x<>-1 BREAK
 LOOP
 CLEAR WINDOW
 TEXT 0,0,"Last character:"
 TEXT 0,16,"$"+hex$(x)+" = „+chr$(x)
LOOP
CLOSE WINDOW
</ABC>
```

Program to show readback of JScript-Commands and the FLUSH command:

Example:

```
<ABC>
OPEN 1,"/dev/jscript","r"
OPEN 2,"/dev/rs232","w"
PRINT „qm"
LINE INPUT #1 a$
PRINT #2 a$
CLOSE 2
CLOSE 1
rem FLUSH #0
PRINT „f"
</ABC>
```

Here is text which would normally trigger protocol error.
It is deleted by FLUSH #0, so the PRINT „f“ can work without problems.

a-Series basic compiler

Program to show how to „press“ a key using a program:

Example:

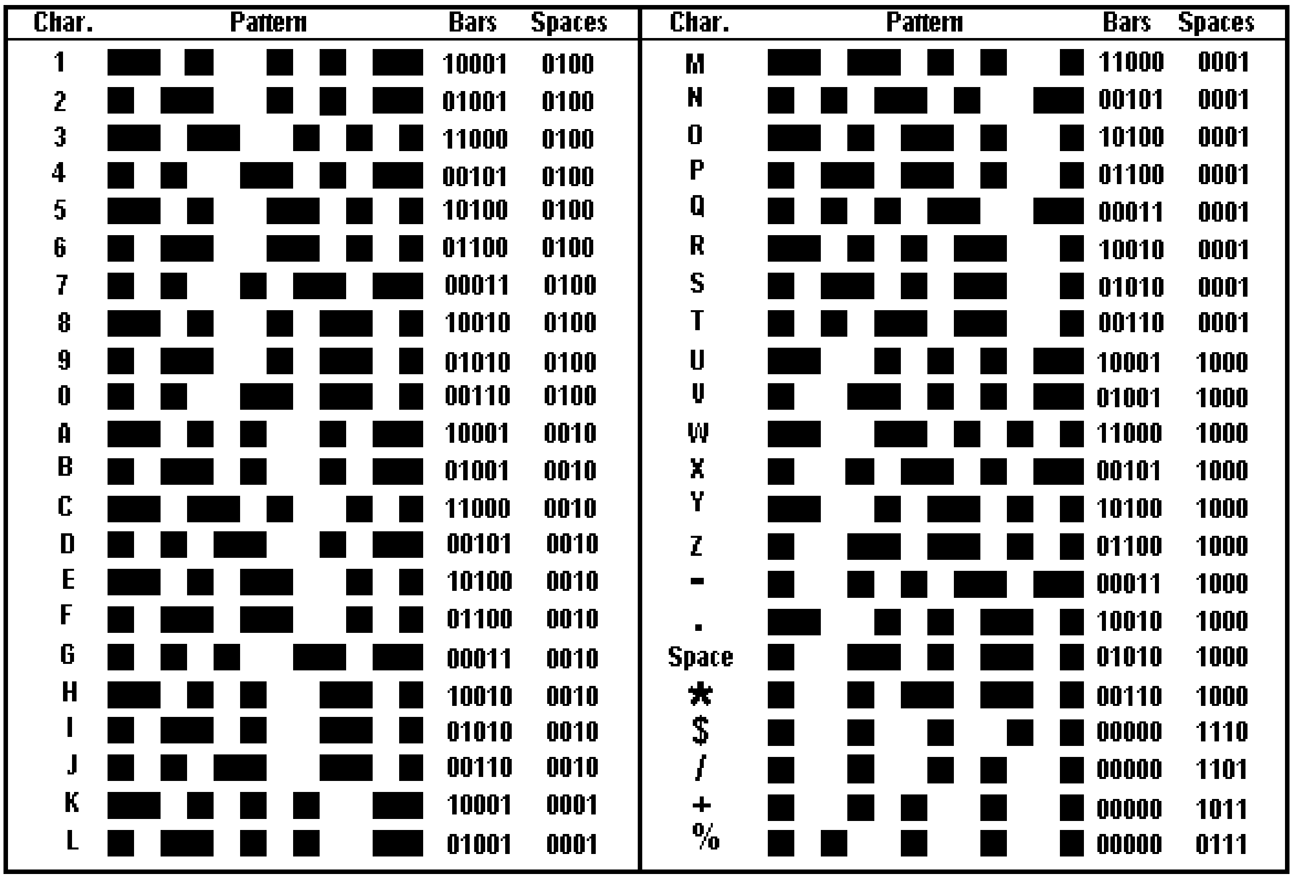
```
; Label does an endless loop which is terminated by pressing
„total Cancel“
<ABC>
x=0
DO
 IF x=0 THEN
  x=1
   POKE „key“,dec(„F090“)
 ENDIF
LOOP
</ABC>
```

ASCII Table

Control characters

Decimal	Hex	ASCII
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	HT
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

Code 39 pattern chart



Char.	Bars	Spaces	Char.	Bars	Spaces
1	10001	0100	M	11000	0001
2	01001	0100	N	00101	0001
3	11000	0100	O	10100	0001
4	00101	0100	P	01100	0001
5	10100	0100	Q	00011	0001
6	01100	0100	R	10010	0001
7	00011	0100	S	01010	0001
8	10010	0100	T	00110	0001
9	01010	0100	U	10001	1000
0	00110	0100	V	01001	1000
A	10001	0010	W	11000	1000
B	01001	0010	X	00101	1000
C	11000	0010	Y	10100	1000
D	00101	0010	Z	01100	1000
E	10100	0010	-	00011	1000
F	01100	0010	.	10010	1000
G	00011	0010	Space	01010	1000
H	10010	0010	*	00110	1000
I	01010	0010	$	00000	1110
J	00110	0010	/	00000	1101
K	10001	0001	+	00000	1011
L	01001	0001	%	00000	0111

Code 39 Full ASCII chart

ASCII	CODE 39	ASCII	CODE 39	ASCII	CODE 39	ASCII	CODE 39
NUL	%U	SP	SPACE	@	%V	`	%W
SOH	$A	!	/A	A	A	a	+A
STX	$B	"	/B	B	B	b	+B
ETX	$C	#	/C	C	C	c	+C
EOT	$D	$	/D	D	D	d	+D
ENQ	$E	%	/E	E	E	e	+E
ACK	$F	&	/F	F	F	f	+F
BEL	$G	'	/G	G	G	g	+G
BS	$H	(	/H	H	H	h	+H
HT	$I	)	/I	I	I	i	+I
LF	$J	*	/J	J	J	j	+J
VT	$K	+	/K	K	K	k	+K
FF	$L	,	/L	L	L	l	+L
CR	$M	-	-	M	M	m	+M
SO	$N	.	.	N	N	n	+N
SI	$O	/	/O	O	O	o	+O
DLE	$P	0	0	P	P	p	+P
DC1	$Q	1	1	Q	Q	q	+Q
DC2	$R	2	2	R	R	r	+R
DC3	$S	3	3	S	S	s	+S
DC4	$T	4	4	T	T	t	+T
NAK	$U	5	5	U	U	u	+U
SYN	$V	6	6	V	V	v	+V
ETB	$W	7	7	W	W	w	+W
CAN	$X	8	8	X	X	x	+X
EM	$Y	9	9	Y	Y	y	+Y
SUB	$Z	:	/Z	Z	Z	z	+Z
ESC	%A	;	%F	[	%K	{	%P
FS	%B	<	%G	/	%L	:	%Q
GS	%C	=	%H	]	%M	}	%R
RS	%D	>	%I	^	%N	~	%S
US	%E	?	%J	_	%O	DEL	%T,%X,%Y,%Z

Tips and Tricks

In this Appendix we will publish some frequently asked programming samples, which shall help to create some special labels.

Variable day offset
Hexadecimal counter (BASE 16,0-F)
Invisible field on condition
Memory card "reload"
Automatic start with Pause
Using Replace sequence and split the content
Leading zero suppression after calculation
Replacing graphics dynamically

Variable day offset

Example:

```
; variable day offset
m m
J
S l1;0,0,68,70,104
O R
T:INPUT;0,0,0,5,pt1;[?:Input Dayoffset:]
T 10,25,0,5,18;[DATE:INPUT,0,0]
A 1
```

Hexadecimal counter (Base 16, 0-F)

Example:

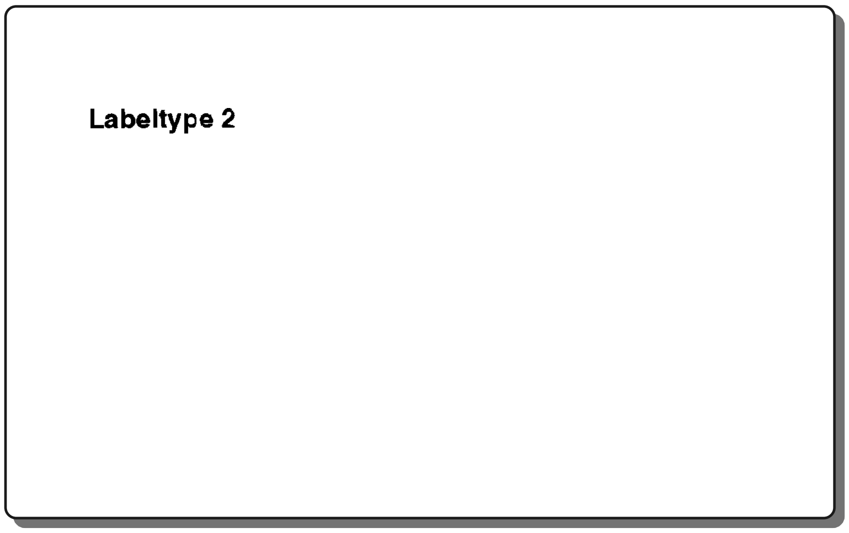
```
; Hexadecimal counter  (BASE 16, 0-F)
m m
J
S l1;0,0,68,70,100
O R
T 30,50,0,5,25;[SER:1,1,16]
A 20
```

Invisible field - depending on condition

Example:

```
; Invisible field - depending on condition
m m
J
S l1;0,0,68,70,104
O R
T:INPUT;0,0,0,5,pt1;[?:Which Type(1 or 2)?,,,L1,M!1]
T:TYPE1;0,0,0,5,pt1;[=:INPUT,1][I]
T:TYPE2;0,0,0,5,pt1;[=:INPUT,2][I]
T 10,10,0,5,pt10;Labeltype 1 [I:TYPE1]
T 10,20,0,5,pt10;Labeltype 2 [I:TYPE2]
A 1
```

A different result appears on the label, depending on the input gthe printer prints only one line with the word "Labeltype 1" or "Labeltype 2" or both lines.



Memory card „reload“

Example:

```
; Memory card "reload"
m m
J
S l1;0,0,68,70,104
O R
T 10,10,0,5,pt10;[?:Article No.:]
A 1
M r
```

This sample has to besaved on the printer´s memory card or IFFs etc.
It will show "Article No.:" on the display, prints one label and shows "Article No.:" again after the label is printed.
So we generated that this labelruns in a loop.

Automatic start with pause

Example:

```
; Automatic start with pause
p 1
m m
J
S l1;0,0,68,70,104
O R
T 10,10,0,5,pt10;Pause before Print
A 1
```

Using Replace sequence and split the content

Example:

```
; Using Replace sequence and split the content
; Stored on CF Card (SAMPLE.LBL)
m m
J
S l1;0,0,68,70,104
O R
T:CONTENT;0,0,0,5,pt1;
T 10,10,0,5,pt10;[SPLIT:CONTENT,1]
T 10,20,0,5,pt10;[SPLIT:CONTENT,2]
T 10,30,0,5,pt10;[SPLIT:CONTENT,3]
T 10,40,0,5,pt10;[SPLIT:CONTENT,4]


; Replacesequence
M l LBL;SAMPLE
R CONTENT;FIELD1-Content[U:GS]FIELD2-Content[U:GS]FIELD3-
Content[U:GS]FIELD4-Content
A 1
```

Leading zero suppression after calculation

Example:

```
; Leading zero suppression after calculation
m m
J
S l1;0,0,68,70,104
O R
T:COUNT;10,10,0,5,8;[SER:0001][C:]
T:COUNT2;10,20,0,5,8;[*:COUNT,1][D:0,0]
A 5
```

Replacing graphics dynamically

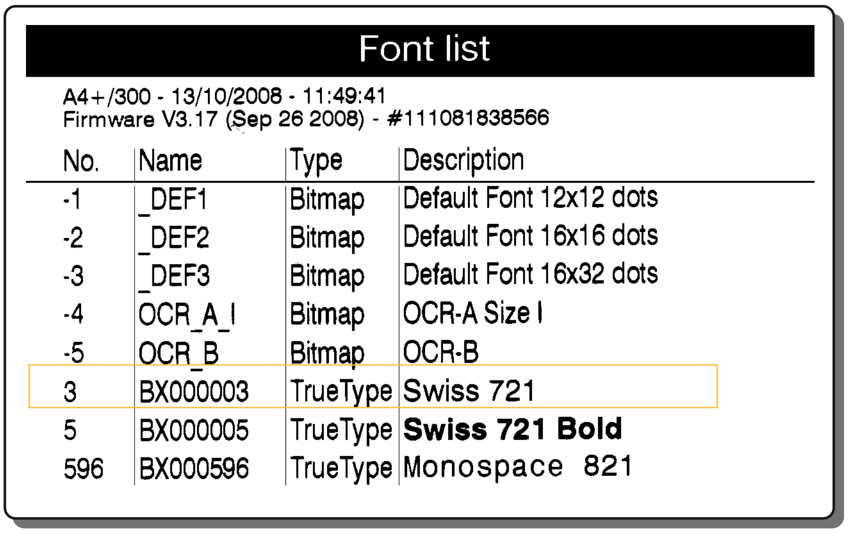
Example:

```
; Replacing graphics dynamically
; Label on memory card (SAMPLE.LBL)
; Images LOGO1.BMP, LOGO2.BMP,LOGO3.BMP also on mem.card
m m
J
O R
S l1;0,0,68,70,104
T 10,10,0,5,pt10;Dynamic Loading and placing of Graphics


; Replacesequence (from Host)
M l LBL;SAMPLE
M l BMP;LOGO1
I 10,20,0;LOGO1
A 1
M l BMP;LOGO2
I 10,20,0;LOGO2
A 1
M l BMP;LOGO3
I 10,20,0;LOGO3
A 1
```

Appendix C - Character lists

The following pages show the available characters of the truetype fonts in the printer.
Each character can be recalled by using the the unicode command [U....]

Please note:The built in Bitmap fonts do not support Unicode.



Font list

A4+/300 - 13/10/2008 - 11:49:41
Firmware V3.17 (Sep 26 2008) - #111081838566

No.	Name	Type	Description
-1	_DEF1	Bitmap	Default Font 12x12 dots
-2	_DEF2	Bitmap	Default Font 16x16 dots
-3	_DEF3	Bitmap	Default Font 16x32 dots
-4	OCR_A_I	Bitmap	OCR-A Size I
-5	OCR_B	Bitmap	OCR-B
3	BX000003	TrueType	Swiss 721
5	BX000005	TrueType	**Swiss 721 Bold**
596	BX000596	TrueType	Monospace 821

Character list Swiss 721

0020	! 0021	" 0022	# # 0023	$ 0024	% 0025	& 0026	' 0027
(0028	)) 0029	* 002A	+ + 002B	, , 002C	- - 002D	. . 002E	/ 002F
0 0 0030	1 1 0031	2 2 0032	3 3 0033	4 4 0034	5 5 0035	6 6 0036	7 7 0037
8 8 0038	9 9 0039	: 003A	; 003B	< < 003C	= 003D	> 003E	? 003F
@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047
H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057
X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
` 0060	a A 0061	b B 0062	c C 0063	d D 0064	e E 0065	f F 0066	g G 0067
h H 0068	i I 0069	j J 006A	k K 006B	l L 006C	m M 006D	n N 006E	o O 006F
p P 0070	q Q 0071	r R 0072	s S 0073	t T 0074	u U 0075	v V 0076	w W 0077

Character list Swiss 721

x X 0078	y Y 0079	z Z 007A	{ AltGr + 7 007B	| AltGr + < 007C	} AltGr + 0 007D	~ AltGr + + 007E	€ 0080
00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ Umschalt + 3 00A7
¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
° Umschalt + ZIRKUMFLEX 00B0	± 00B1	2 AltGr + 2 00B2	3 AltGr + 3 00B3	´ AKUT 00B4	µ AltGr + M 00B5	¶ 00B6	· 00B7
¸ 00B8	1 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä Umschalt + ä 00C4	Å 00C5	Æ 00C6	Ç 00C7
È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö Umschalt + ö 00D6	× 00D7
Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü Umschalt + ü 00DC	Ý 00DD	Þ 00DE	ß ß 00DF
à 00E0	á 00E1	â 00E2	ã 00E3	ä ä 00E4	å 00E5	æ 00E6	ç 00E7
è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF

Character list Swiss 721

ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö ö 00F6	÷ 00F7
ø 00F8	ù 00F9	ú 00FA	û 00FB	ü ü 00FC	ý 00FD	þ 00FE	ÿ 00FF
Ā 0100	ā 0101	Ă 0102	ă 0103	Ą 0104	ą 0105	Ć 0106	ć 0107
Ĉ 0108	ĉ 0109	Ċ 010A	ċ 010B	Č 010C	č 010D	Ď 010E	ď 010F
Đ 0110	đ 0111	Ē 0112	ē 0113	Ĕ 0114	ĕ 0115	Ė 0116	ė 0117
Ę 0118	ę 0119	Ě 011A	ě 011B	Ĝ 011C	ĝ 011D	Ğ 011E	ğ 011F
Ġ 0120	ġ 0121	Ģ 0122	ģ 0123	Ĥ 0124	ĥ 0125	Ħ 0126	ħ 0127
Ĩ 0128	ĩ 0129	Ī 012A	ī 012B	Ĭ 012C	ĭ 012D	Į 012E	į 012F
İ 0130	ı 0131	Ĳ 0132	ĳ 0133	Ĵ 0134	ĵ 0135	Ķ 0136	ķ 0137
ĸ 0138	Ĺ 0139	ĺ 013A	Ļ 013B	ļ 013C	Ľ 013D	ľ 013E	Ŀ 013F
ŀ 0140	Ł 0141	ł 0142	Ń 0143	ń 0144	Ņ 0145	ņ 0146	Ň 0147

Character list Swiss 721

ň 0148	ŉ 0149	Ŋ 014A	ŋ 014B	Ō 014C	ō 014D	Ŏ 014E	ŏ 014F
Ő 0150	ő 0151	Œ 0152	œ 0153	Ŕ 0154	ŕ 0155	Ŗ 0156	ŗ 0157
Ř 0158	ř 0159	Ś 015A	ś 015B	Ŝ 015C	ŝ 015D	Ş 015E	ş 015F
Š 0160	š 0161	Ţ 0162	ţ 0163	Ť 0164	ť 0165	Ŧ 0166	ŧ 0167
Ũ 0168	ũ 0169	Ū 016A	ū 016B	Ŭ 016C	ŭ 016D	Ů 016E	ů 016F
Ű 0170	ű 0171	Ų 0172	ų 0173	Ŵ 0174	ŵ 0175	Ŷ 0176	ŷ 0177
Ÿ 0178	Ź 0179	ź 017A	Ż 017B	ż 017C	Ž 017D	ž 017E	ſ 017F
ƒ 0192	Ǧ 01E6	ǧ 01E7	Ǻ 01FA	ǻ 01FB	Ǽ 01FC	ǽ 01FD	Ǿ 01FE
ǿ 01FF	ʼ 02BC	ʽ 02BD	ˆ 02C6	ˇ 02C7	ˉ 02C9	˘ 02D8	˙ 02D9
˚ 02DA	˛ 02DB	˜ 02DC	˝ 02DD	; 037E	΄ 0384	΅ 0385	Ά 0386
· 0387	Έ 0388	Ή 0389	Ί 038A	Ό 038C	Ύ 038E	Ώ 038F	ΐ 0390

Character list Swiss 721

Α 0391	Β 0392	Γ 0393	Δ 0394	Ε 0395	Ζ 0396	Η 0397	Θ 0398
Ι 0399	Κ 039A	Λ 039B	Μ 039C	Ν 039D	Ξ 039E	Ο 039F	Π 03A0
Ρ 03A1	Σ 03A3	Τ 03A4	Υ 03A5	Φ 03A6	Χ 03A7	Ψ 03A8	Ω 03A9
Ϊ 03AA	Ϋ 03AB	ά 03AC	έ 03AD	ή 03AE	ί 03AF	ΰ 03B0	α 03B1
β 03B2	γ 03B3	δ 03B4	ε 03B5	ζ 03B6	η 03B7	θ 03B8	ι 03B9
κ 03BA	λ 03BB	μ 03BC	ν 03BD	ξ 03BE	ο 03BF	π 03C0	ρ 03C1
ς 03C2	σ 03C3	τ 03C4	υ 03C5	φ 03C6	χ 03C7	ψ 03C8	ω 03C9
ϊ 03CA	ϋ 03CB	ό 03CC	ύ 03CD	ώ 03CE	Ё 0401	Ђ 0402	Ѓ 0403
Є 0404	Ѕ 0405	І 0406	Ї 0407	Ј 0408	Љ 0409	Њ 040A	Ћ 040B
Ќ 040C	Ў 040E	Џ 040F	А 0410	Б 0411	В 0412	Г 0413	Д 0414
Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C

Character list Swiss 721

Н 041D	О 041E	П 041F	Р 0420	С 0421	Т 0422	У 0423	Ф 0424
Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C
Э 042D	Ю 042E	Я 042F	а 0430	б 0431	в 0432	г 0433	д 0434
е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C
н 043D	о 043E	п 043F	р 0440	с 0441	т 0442	у 0443	ф 0444
х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C
э 044D	ю 044E	я 044F	ё 0451	ђ 0452	ѓ 0453	є 0454	ѕ 0455
і 0456	ї 0457	ј 0458	љ 0459	њ 045A	ћ 045B	ќ 045C	ў 045E
џ 045F	Ґ 0490	ґ 0491	ְ 05B0	ֱ 05B1	ֲ 05B2	ֳ 05B3	ִ 05B4
ֵ 05B5	ֶ 05B6	ַ 05B7	ָ 05B8	ֹ 05B9	ֻ 05BB	ּ 05BC	ֽ 05BD
־ 05BE	ֿ 05BF	׀ 05C0	ׁ 05C1	ׂ 05C2	׃ 05C3	ׄ 05C4	א 05D0

Character list Swiss 721

ב 05D1	ג 05D2	ד 05D3	ה 05D4	ו 05D5	ז 05D6	ח 05D7	ט 05D8
י 05D9	ך 05DA	כ 05DB	ל 05DC	ם 05DD	מ 05DE	ן 05DF	נ 05E0
ס 05E1	ע 05E2	ף 05E3	פ 05E4	ץ 05E5	צ 05E6	ק 05E7	ר 05E8
ש 05E9	ת 05EA	װ 05F0	ױ 05F1	ײ 05F2	׳ 05F3	״ 05F4	، 060C
؛ 061B	؟ 061F	ء 0621	آ 0622	أ 0623	ؤ 0624	إ 0625	ئ 0626
ا 0627	ب 0628	ة 0629	ت 062A	ث 062B	ج 062C	ح 062D	خ 062E
د 062F	ذ 0630	ر 0631	ز 0632	س 0633	ش 0634	ص 0635	ض 0636
ط 0637	ظ 0638	ع 0639	غ 063A	ـ 0640	ف 0641	ق 0642	ك 0643
ل 0644	م 0645	ن 0646	ه 0647	و 0648	ى 0649	ي 064A	ً 064B
ٌ 064C	ٍ 064D	َ 064E	ُ 064F	ِ 0650	ّ 0651	ْ 0652	٠ 0660
١ 0661	٢ 0662	٣ 0663	٤ 0664	٥ 0665	٦ 0666	٧ 0667	٨ 0668

Character list Swiss 721

٩ 0669	٪ 066A	٫ 066B	٭ 066D	ٷ 0677	ٸ 0678	ٹ 0679	ٺ 067A
ٻ 067B	ټ 067C	ٽ 067D	پ 067E	ٿ 067F	ڀ 0680	ځ 0681	ڂ 0682
ڃ 0683	ڄ 0684	څ 0685	چ 0686	ڇ 0687	ڈ 0688	ډ 0689	ڊ 068A
ڋ 068B	ڌ 068C	ڍ 068D	ڎ 068E	ڏ 068F	ڐ 0690	ڑ 0691	ڒ 0692
ړ 0693	ڔ 0694	ڕ 0695	ږ 0696	ڗ 0697	ژ 0698	ڙ 0699	ښ 069A
ڛ 069B	ڜ 069C	ڝ 069D	ڞ 069E	ڟ 069F	ڠ 06A0	ڡ 06A1	ڢ 06A2
ڣ 06A3	ڤ 06A4	ڥ 06A5	ڦ 06A6	ڧ 06A7	ڨ 06A8	ک 06A9	ڪ 06AA
ګ 06AB	ڬ 06AC	ڭ 06AD	ڮ 06AE	گ 06AF	ڰ 06B0	ڱ 06B1	ڲ 06B2
ڳ 06B3	ڴ 06B4	ڵ 06B5	ڶ 06B6	ڷ 06B7	ں 06BA	ڻ 06BB	ڼ 06BC
ڽ 06BD	ھ 06BE	ۀ 06C0	ہ 06C1	ۂ 06C2	ۃ 06C3	ۄ 06C4	ۅ 06C5
ۆ 06C6	ۇ 06C7	ۈ 06C8	ۉ 06C9	ۊ 06CA	ۋ 06CB	ی 06CC	ۍ 06CD

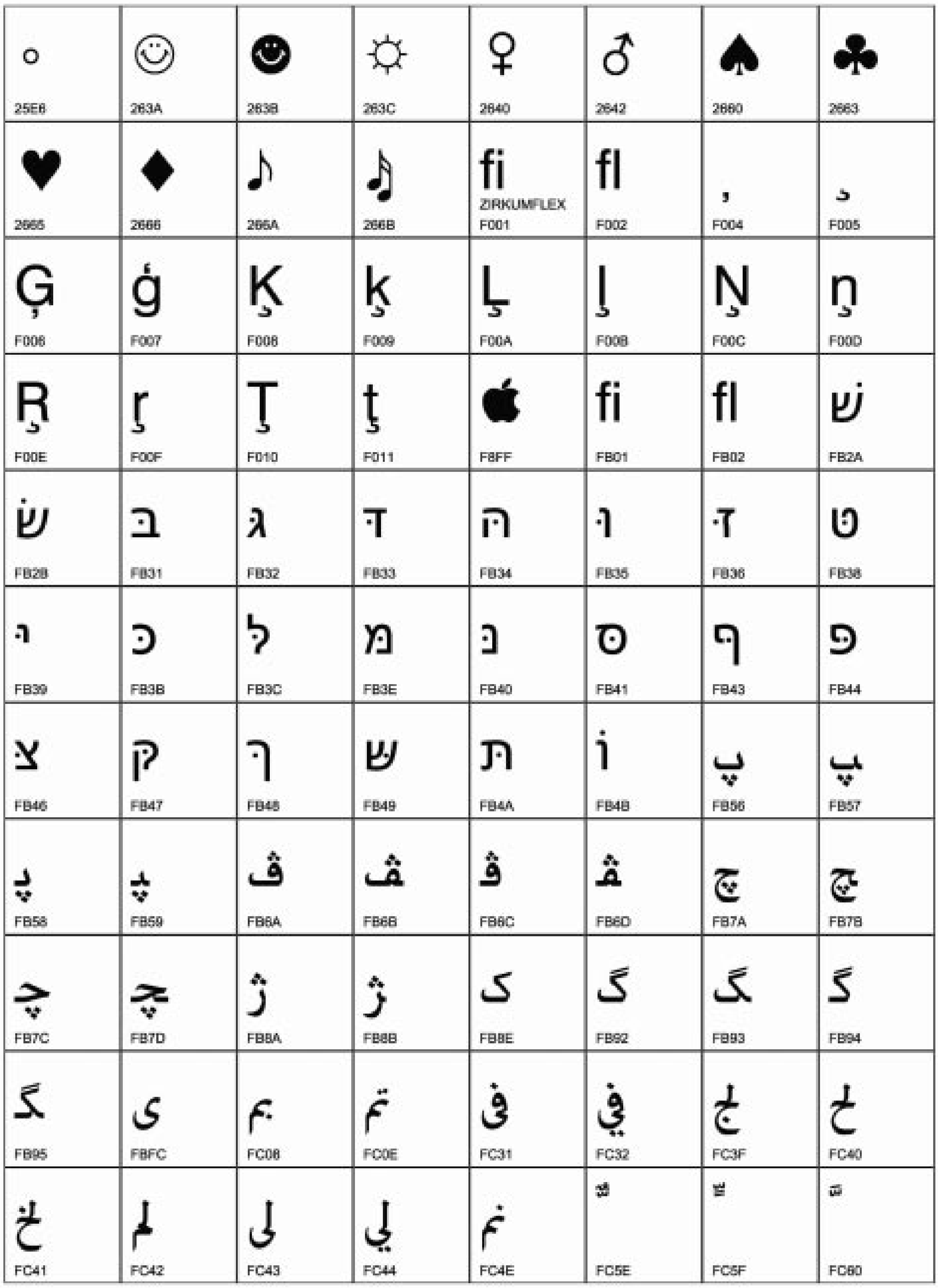
Character list Swiss 721

ێ 06CE	ې 06D0	ۑ 06D1	ے 06D2	ۓ 06D3	۔ 06D4	ە 06D5	۰ 06F0
۱ 06F1	۲ 06F2	۳ 06F3	۴ 06F4	۵ 06F5	۶ 06F6	۷ 06F7	۸ 06F8
۹ 06F9	Ẁ 1E80	ẁ 1E81	Ẃ 1E82	ẃ 1E83	Ẅ 1E84	ẅ 1E85	Ỳ 1EF2
ỳ 1EF3	– 2013	— 2014	― 2015	‗ 2017	‘ 2018	’ 2019	‚ 201A
‛ 201B	“ 201C	” 201D	„ 201E	† 2020	‡ 2021	• 2022	… 2026
‰ 2030	′ 2032	″ 2033	‹ 2039	› 203A	‼ 203C	‾ 203E	⁄ 2044
ⁿ 207F	₀ 2080	₁ 2081	₂ 2082	₃ 2083	₄ 2084	₅ 2085	₆ 2086
₇ 2087	₈ 2088	₉ 2089	₣ 20A3	₤ 20A4	₧ 20A7	₪ 20AA	€ AltGr + E 20AC
℅ 2105	ℑ 2111	ℓ 2113	№ 2116	ℜ 211C	™ 2122	Ω 2126	℮ 212E
ℵ 2135	⅓ 2153	⅔ 2154	⅛ 215B	⅜ 215C	⅝ 215D	⅞ 215E	← 2190
↑ 2191	→ 2192	↓ 2193	↔ 2194	↕ 2195	↨ 21A8	↵ 21B5	⇐ 21D0

Character list Swiss 721

⇑ 21D1	⇒ 21D2	⇓ 21D3	⇔ 21D4	∂ 2202	∆ 2206	∏ 220F	∑ 2211
− 2212	∕ 2215	∙ 2219	√ 221A	∞ 221E	∟ 221F	∩ 2229	∫ 222B
≈ 2248	≠ 2260	≡ 2261	≤ 2264	≥ 2265	⌂ 2302	⌐ 2310	⌠ 2320
⌡ 2321	␡ 2421	─ 2500	│ 2502	┌ 250C	┐ 2510	└ 2514	┘ 2518
├ 251C	┤ 2524	┬ 252C	┴ 2534	┼ 253C	═ 2550	║ 2551	╒ 2552
╓ 2553	╔ 2554	╕ 2555	╖ 2556	╗ 2557	╘ 2558	╙ 2559	╚ 255A
╛ 255B	╜ 255C	╝ 255D	╞ 255E	╟ 255F	╠ 2560	╡ 2561	╢ 2562
╣ 2563	╤ 2564	╥ 2565	╦ 2566	╧ 2567	╨ 2568	╩ 2569	╪ 256A
╫ 256B	╬ 256C	▀ 2580	▄ 2584	█ 2588	▌ 258C	▐ 2590	░ 2591
▒ 2592	▓ 2593	■ 25A0	□ 25A1	▪ 25AA	▫ 25AB	▬ 25AC	▲ 25B2
► 25BA	▼ 25BC	◄ 25C4	◊ 25CA	○ 25CB	● 25CF	◘ 25D8	◙ 25D9

Character list Swiss 721



◦ 25E6	☺ 263A	☻ 263B	☼ 263C	♀ 2640	♂ 2642	♠ 2660	♣ 2663
♥ 2665	♦ 2666	♪ 266A	♫ 266B	fi ZIRKUMFLEX F001	fl F002	, F004	F005
Ģ F006	ģ F007	Ķ F008	ķ F009	Ļ F00A	ļ F00B	Ņ F00C	ņ F00D
Ŗ F00E	ŗ F00F	Ţ F010	ţ F011	F8FF	ﬁ FB01	ﬂ FB02	שׁ FB2A
שׂ FB2B	בּ FB31	גּ FB32	דּ FB33	הּ FB34	וּ FB35	זּ FB36	טּ FB38
יּ FB39	כּ FB3B	לּ FB3C	מּ FB3E	נּ FB40	סּ FB41	ףּ FB43	פּ FB44
צּ FB46	קּ FB47	רּ FB48	שּ FB49	תּ FB4A	וֹ FB4B	ﭖ FB56	ﭗ FB57
ﭘ FB58	ﭙ FB59	ﭪ FB6A	ﭫ FB6B	ﭬ FB6C	ﭭ FB6D	ﭺ FB7A	ﭻ FB7B
ﭼ FB7C	ﭽ FB7D	ﮊ FB8A	ﮋ FB8B	ﮎ FB8E	ﮒ FB92	ﮓ FB93	ﮔ FB94
ﮕ FB95	ﯼ FBFC	ﰈ FC08	ﰎ FC0E	ﰱ FC31	ﰲ FC32	ﰿ FC3F	ﱀ FC40
ﱁ FC41	ﱂ FC42	ﱃ FC43	ﱄ FC44	ﱎ FC4E	ﱞ FC5E	ﱟ FC5F	ﱠ FC60

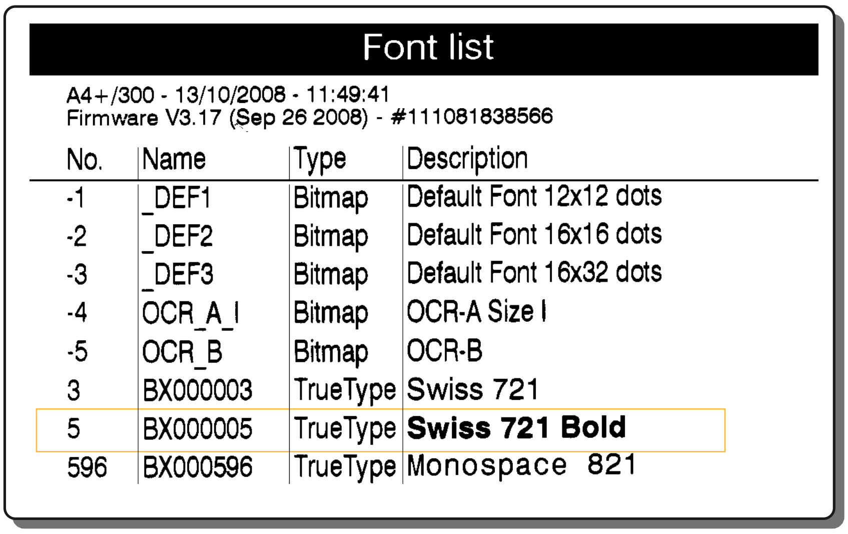
Character list Swiss 721

ﲁ FC81	ﲂ FC82	ﲊ FC8A	ﲍ FC8D	ﱯ FC6F	ﱰ FC70	ﱳ FC73	ﱵ FC75
ﲆ FC86	ﲇ FC87	ﲏ FC8F	ﲑ FC91	ﲒ FC92	ﲜ FC9C	ﲝ FC9D	ﲞ FC9E
ﲟ FC9F	ﲡ FCA1	ﲢ FCA2	ﲣ FCA3	ﲤ FCA4	ﲨ FCA8	ﲪ FCAA	ﲬ FCAC
ﲰ FCB0	ﳉ FCC9	ﳊ FCCA	ﳋ FCCB	ﳌ FCCC	ﳍ FCCD	ﳎ FCCE	ﳏ FCCF
ﳐ FCD0	ﳑ FCD1	ﳒ FCD2	ﳓ FCD3	ﳔ FCD4	ﳕ FCD5	ﳚ FCDA	ﳛ FCDB
ﳜ FCDC	ﳝ FCDD	ﳥ FCE5	ﳻ FCFB	ﳼ FCFC	ﳽ FCFD	ﳾ FCFE	ﴅ FD05
ﴆ FD06	ﴇ FD07	ﴈ FD08	ﴍ FD0D	ﴎ FD0E	ﴏ FD0F	ﴐ FD10	ﴗ FD17
ﴘ FD18	ﴙ FD19	ﴚ FD1A	ﴡ FD21	ﴢ FD22	ﴣ FD23	ﴤ FD24	ﴩ FD29
ﴪ FD2A	ﴫ FD2B	ﴬ FD2C	ﴰ FD30	﴾ FD3E	﴿ FD3F	ﶈ FD88	ﷲ FDF2
ﷺ FDFA	ﹰ FE70	ﹲ FE72	ﹴ FE74	ﹶ FE76	ﹸ FE78	ﹺ FE7A	ﹼ FE7C
ﹽ FE7D	ﹾ FE7E	ﺀ FE80	ﺁ FE81	ﺂ FE82	ﺃ FE83	ﺄ FE84	ﺅ FE85

Character list Swiss 721

ﺆ FE86	ﺇ FE87	ﺈ FE88	ﺉ FE89	ﺊ FE8A	ﺋ FE8B	ﺌ FE8C	ﺍ FE8D
ﺎ FE8E	ﺏ FE8F	ﺐ FE90	ﺑ FE91	ﺒ FE92	ﺓ FE93	ﺔ FE94	ﺕ FE95
ﺖ FE96	ﺗ FE97	ﺘ FE98	ﺙ FE99	ﺚ FE9A	ﺛ FE9B	ﺜ FE9C	ﺝ FE9D
ﺞ FE9E	ﺟ FE9F	ﺠ FEA0	ﺡ FEA1	ﺢ FEA2	ﺣ FEA3	ﺤ FEA4	ﺥ FEA5
ﺦ FEA6	ﺧ FEA7	ﺨ FEA8	ﺩ FEA9	ﺪ FEAA	ﺫ FEAB	ﺬ FEAC	ﺭ FEAD
ﺮ FEAE	ﺯ FEAF	ﺰ FEB0	ﺱ FEB1	ﺲ FEB2	ﺳ FEB3	ﺴ FEB4	ﺵ FEB5
ﺶ FEB6	ﺷ FEB7	ﺸ FEB8	ﺹ FEB9	ﺺ FEBA	ﺻ FEBB	ﺼ FEBC	ﺽ FEBD
ﺾ FEBE	ﺿ FEBF	ﻀ FEC0	ﻁ FEC1	ﻂ FEC2	ﻃ FEC3	ﻄ FEC4	ﻅ FEC5
ﻆ FEC6	ﻇ FEC7	ﻈ FEC8	ﻉ FEC9	ﻊ FECA	ﻋ FECB	ﻌ FECC	ﻍ FECD
ﻎ FECE	ﻏ FECF	ﻐ FED0	ﻑ FED1	ﻒ FED2	ﻓ FED3	ﻔ FED4	ﻕ FED5
ﻖ FED6	ﻗ FED7	ﻘ FED8	ﻙ FED9	ﻚ FEDA	ﻛ FEDB	ﻜ FEDC	ﻝ FEDD

Character list Swiss 721 bold

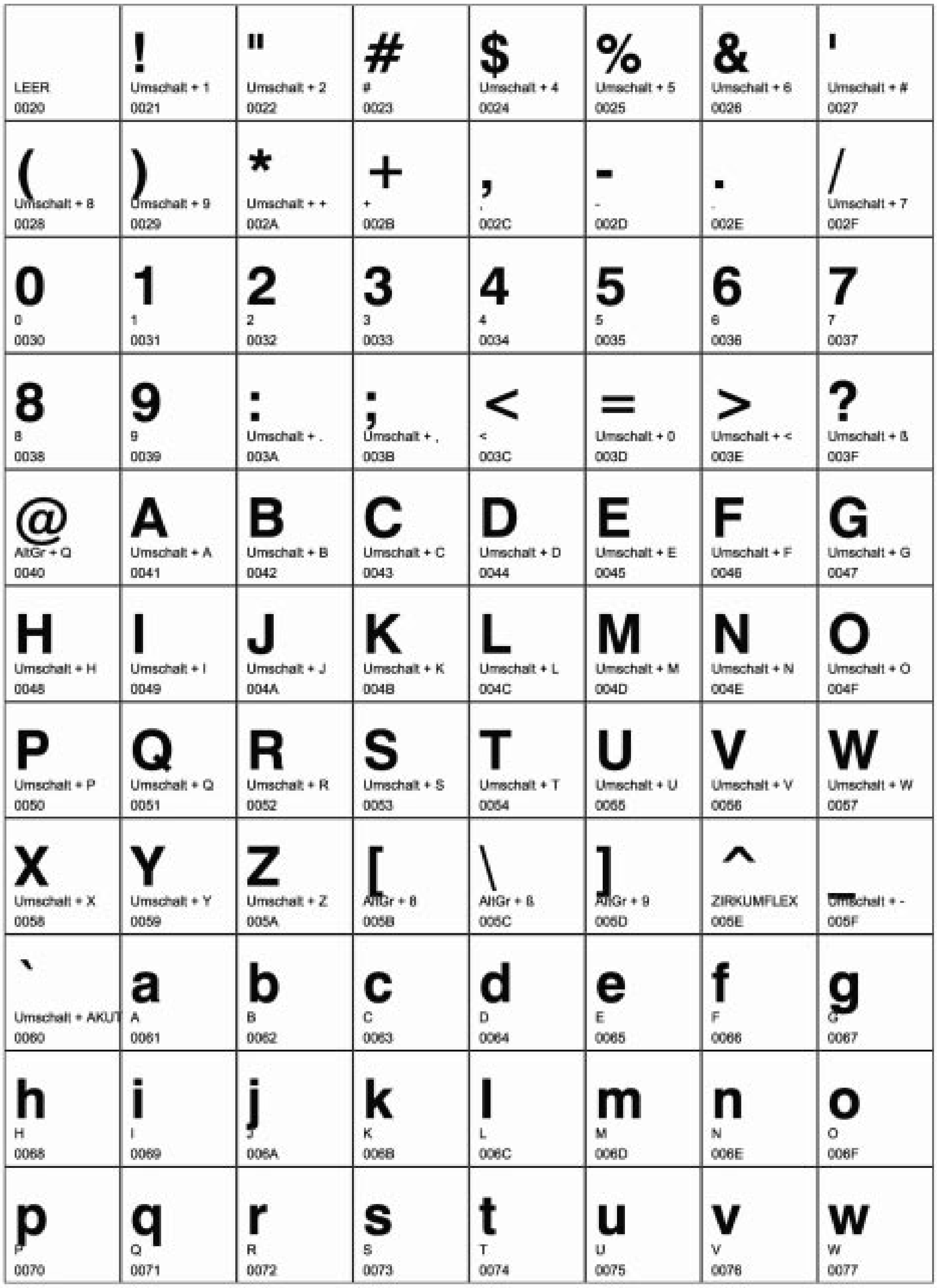


Font list

A4+/300 - 13/10/2008 - 11:49:41
Firmware V3.17 (Sep 26 2008) - #111081838566

No.	Name	Type	Description
-1	_DEF1	Bitmap	Default Font 12x12 dots
-2	_DEF2	Bitmap	Default Font 16x16 dots
-3	_DEF3	Bitmap	Default Font 16x32 dots
-4	OCR_A_I	Bitmap	OCR-A Size I
-5	OCR_B	Bitmap	OCR-B
3	BX000003	TrueType	Swiss 721
5	BX000005	TrueType	**Swiss 721 Bold**
596	BX000596	TrueType	Monospace 821

Character list Swiss 721 bold

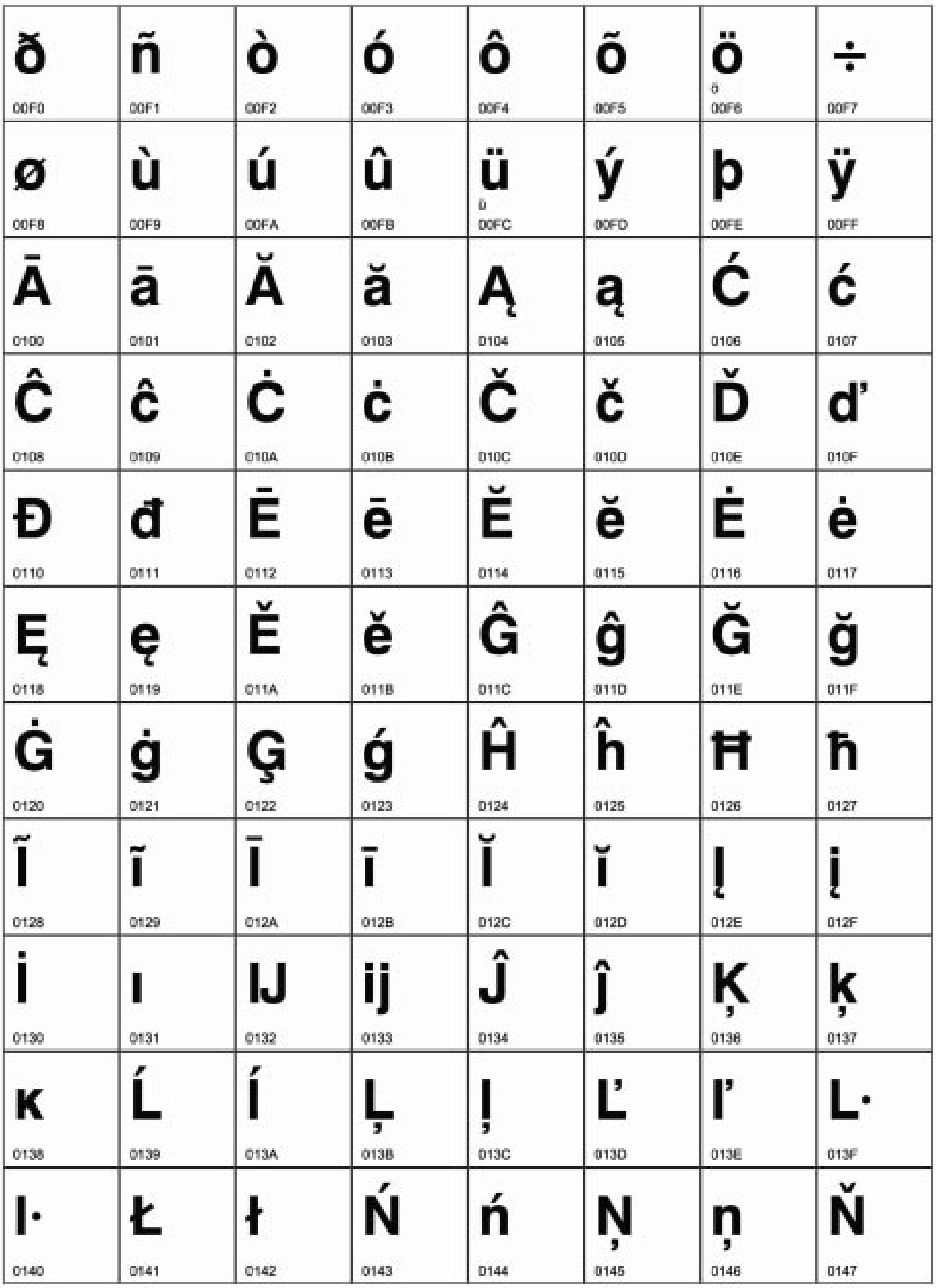


	!	"	#	$	%	&	'
LEER	Umschalt + 1	Umschalt + 2	#	Umschalt + 4	Umschalt + 5	Umschalt + 6	Umschalt + #
0020	0021	0022	0023	0024	0025	0026	0027
(	)	*	+	,	-	.	/
Umschalt + 8	Umschalt + 9	Umschalt + +	+	,	-	.	Umschalt + 7
0028	0029	002A	002B	002C	002D	002E	002F
0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7
0030	0031	0032	0033	0034	0035	0036	0037
8	9	:	;	<	=	>	?
8	9	Umschalt + .	Umschalt + ,	<	Umschalt + 0	Umschalt + <	Umschalt + ß
0038	0039	003A	003B	003C	003D	003E	003F
@	A	B	C	D	E	F	G
AltGr + Q	Umschalt + A	Umschalt + B	Umschalt + C	Umschalt + D	Umschalt + E	Umschalt + F	Umschalt + G
0040	0041	0042	0043	0044	0045	0046	0047
H	I	J	K	L	M	N	O
Umschalt + H	Umschalt + I	Umschalt + J	Umschalt + K	Umschalt + L	Umschalt + M	Umschalt + N	Umschalt + O
0048	0049	004A	004B	004C	004D	004E	004F
P	Q	R	S	T	U	V	W
Umschalt + P	Umschalt + Q	Umschalt + R	Umschalt + S	Umschalt + T	Umschalt + U	Umschalt + V	Umschalt + W
0050	0051	0052	0053	0054	0055	0056	0057
X	Y	Z	[	\	]	^	_
Umschalt + X	Umschalt + Y	Umschalt + Z	AltGr + 8	AltGr + ß	AltGr + 9	ZIRKUMFLEX	Umschalt + -
0058	0059	005A	005B	005C	005D	005E	005F
`	a	b	c	d	e	f	g
Umschalt + AKUT	A	B	C	D	E	F	G
0060	0061	0062	0063	0064	0065	0066	0067
h	i	j	k	l	m	n	o
H	I	J	K	L	M	N	O
0068	0069	006A	006B	006C	006D	006E	006F
p	q	r	s	t	u	v	w
P	Q	R	S	T	U	V	W
0070	0071	0072	0073	0074	0075	0076	0077

Character list Swiss 721 bold

x X 0078	y Y 0079	z Z 007A	{ AltGr + 7 007B	| AltGr + < 007C	} AltGr + 0 007D	~ AltGr + + 007E	€ 0080
00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ Umschalt + 3 00A7
¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
° Umschalt + ZIRKUMFLEX 00B0	± 00B1	² AltGr + 2 00B2	³ AltGr + 3 00B3	´ AKUT 00B4	µ AltGr + M 00B5	¶ 00B6	· 00B7
¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä Umschalt + ä 00C4	Å 00C5	Æ 00C6	Ç 00C7
È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö Umschalt + ö 00D6	× 00D7
Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü Umschalt + ü 00DC	Ý 00DD	Þ 00DE	ß ß 00DF
à 00E0	á 00E1	â 00E2	ã 00E3	ä ä 00E4	å 00E5	æ 00E6	ç 00E7
è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF

Character list Swiss 721 bold



ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö ö 00F6	÷ 00F7
ø 00F8	ù 00F9	ú 00FA	û 00FB	ü ü 00FC	ý 00FD	þ 00FE	ÿ 00FF
Ā 0100	ā 0101	Ă 0102	ă 0103	Ą 0104	ą 0105	Ć 0106	ć 0107
Ĉ 0108	ĉ 0109	Ċ 010A	ċ 010B	Č 010C	č 010D	Ď 010E	ď 010F
Đ 0110	đ 0111	Ē 0112	ē 0113	Ĕ 0114	ĕ 0115	Ė 0116	ė 0117
Ę 0118	ę 0119	Ě 011A	ě 011B	Ĝ 011C	ĝ 011D	Ğ 011E	ğ 011F
Ġ 0120	ġ 0121	Ģ 0122	ģ 0123	Ĥ 0124	ĥ 0125	Ħ 0126	ħ 0127
Ĩ 0128	ĩ 0129	Ī 012A	ī 012B	Ĭ 012C	ĭ 012D	Į 012E	į 012F
İ 0130	ı 0131	Ĳ 0132	ĳ 0133	Ĵ 0134	ĵ 0135	Ķ 0136	ķ 0137
ĸ 0138	Ĺ 0139	ĺ 013A	Ļ 013B	ļ 013C	Ľ 013D	ľ 013E	Ŀ 013F
ŀ 0140	Ł 0141	ł 0142	Ń 0143	ń 0144	Ņ 0145	ņ 0146	Ň 0147

Character list Swiss 721 bold

ň 0148	ŉ 0149	Ŋ 014A	ŋ 014B	Ō 014C	ō 014D	Ŏ 014E	ŏ 014F
Ő 0150	ő 0151	Œ 0152	œ 0153	Ŕ 0154	ŕ 0155	Ŗ 0156	ŗ 0157
Ř 0158	ř 0159	Ś 015A	ś 015B	Ŝ 015C	ŝ 015D	Ş 015E	ş 015F
Š 0160	š 0161	Ţ 0162	ţ 0163	Ť 0164	ť 0165	Ŧ 0166	ŧ 0167
Ũ 0168	ũ 0169	Ū 016A	ū 016B	Ŭ 016C	ŭ 016D	Ů 016E	ů 016F
Ű 0170	ű 0171	Ų 0172	ų 0173	Ŵ 0174	ŵ 0175	Ŷ 0176	ŷ 0177
Ÿ 0178	Ź 0179	ź 017A	Ż 017B	ż 017C	Ž 017D	ž 017E	ſ 017F
ƒ 0192	Ǧ 01E6	ǧ 01E7	Ǻ 01FA	ǻ 01FB	Ǽ 01FC	ǽ 01FD	Ǿ 01FE
ǿ 01FF	ʼ 02BC	ʽ 02BD	ˆ 02C6	ˇ 02C7	ˉ 02C9	˘ 02D8	˙ 02D9
˚ 02DA	˛ 02DB	˜ 02DC	˝ 02DD	; 037E	΄ 0384	΅ 0385	Ά 0386
· 0387	Έ 0388	Ή 0389	Ί 038A	Ό 038C	Ύ 038E	Ώ 038F	ΐ 0390

Character list Swiss 721 bold

Α 0391	Β 0392	Γ 0393	Δ 0394	Ε 0395	Ζ 0396	Η 0397	Θ 0398
Ι 0399	Κ 039A	Λ 039B	Μ 039C	Ν 039D	Ξ 039E	Ο 039F	Π 03A0
Ρ 03A1	Σ 03A3	Τ 03A4	Υ 03A5	Φ 03A6	Χ 03A7	Ψ 03A8	Ω 03A9
Ϊ 03AA	Ϋ 03AB	ά 03AC	έ 03AD	ή 03AE	ί 03AF	ΰ 03B0	α 03B1
β 03B2	γ 03B3	δ 03B4	ε 03B5	ζ 03B6	η 03B7	θ 03B8	ι 03B9
κ 03BA	λ 03BB	μ 03BC	ν 03BD	ξ 03BE	ο 03BF	π 03C0	ρ 03C1
ς 03C2	σ 03C3	τ 03C4	υ 03C5	φ 03C6	χ 03C7	ψ 03C8	ω 03C9
ϊ 03CA	ϋ 03CB	ό 03CC	ύ 03CD	ώ 03CE	Ё 0401	Ђ 0402	Ѓ 0403
Є 0404	Ѕ 0405	І 0406	Ї 0407	Ј 0408	Љ 0409	Њ 040A	Ћ 040B
Ќ 040C	Ў 040E	Џ 040F	А 0410	Б 0411	В 0412	Г 0413	Д 0414
Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C

Character list Swiss 721 bold

Н 041D	О 041E	П 041F	Р 0420	С 0421	Т 0422	У 0423	Ф 0424
Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C
Э 042D	Ю 042E	Я 042F	а 0430	б 0431	в 0432	г 0433	д 0434
е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C
н 043D	о 043E	п 043F	р 0440	с 0441	т 0442	у 0443	ф 0444
х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C
э 044D	ю 044E	я 044F	ё 0451	ђ 0452	ѓ 0453	є 0454	ѕ 0455
і 0456	ї 0457	ј 0458	љ 0459	њ 045A	ћ 045B	ќ 045C	ў 045E
џ 045F	Ґ 0490	ґ 0491	ְ 05B0	ֱ 05B1	ֲ 05B2	ֳ 05B3	ִ 05B4
ֵ 05B5	ֶ 05B6	ַ 05B7	ָ 05B8	ֹ 05B9	ֻ 05BB	ּ 05BC	ֽ 05BD
־ 05BE	ֿ 05BF	׀ 05C0	ׁ 05C1	ׂ 05C2	׃ 05C3	ׄ 05C4	א 05D0

Character list Swiss 721 bold

ב 05D1	ג 05D2	ד 05D3	ה 05D4	ו 05D5	ז 05D6	ח 05D7	ט 05D8
י 05D9	ך 05DA	כ 05DB	ל 05DC	ם 05DD	מ 05DE	ן 05DF	נ 05E0
ס 05E1	ע 05E2	ף 05E3	פ 05E4	ץ 05E5	צ 05E6	ק 05E7	ר 05E8
ש 05E9	ת 05EA	װ 05F0	ױ 05F1	ײ 05F2	׳ 05F3	״ 05F4	، 060C
؛ 061B	؟ 061F	ء 0621	آ 0622	أ 0623	ؤ 0624	إ 0625	ئ 0626
ا 0627	ب 0628	ة 0629	ت 062A	ث 062B	ج 062C	ح 062D	خ 062E
د 062F	ذ 0630	ر 0631	ز 0632	س 0633	ش 0634	ص 0635	ض 0636
ط 0637	ظ 0638	ع 0639	غ 063A	ـ 0640	ف 0641	ق 0642	ك 0643
ل 0644	م 0645	ن 0646	ه 0647	و 0648	ى 0649	ي 064A	ً 064B
ٌ 064C	ٍ 064D	َ 064E	ُ 064F	ِ 0650	ّ 0651	ْ 0652	٠ 0660
١ 0661	٢ 0662	٣ 0663	٤ 0664	٥ 0665	٦ 0666	٧ 0667	٨ 0668

Character list Swiss 721 bold

٩ 0669	٪ 066A	٫ 066B	٭ 066D	ٷ 0677	ٸ 0678	ٹ 0679	ٺ 067A
ٻ 067B	ټ 067C	ٽ 067D	پ 067E	ٿ 067F	ڀ 0680	ځ 0681	ڂ 0682
ڃ 0683	ڄ 0684	څ 0685	چ 0686	ڇ 0687	ڈ 0688	ډ 0689	ڊ 068A
ڋ 068B	ڌ 068C	ڍ 068D	ڎ 068E	ڏ 068F	ڐ 0690	ڑ 0691	ڒ 0692
ړ 0693	ڔ 0694	ڕ 0695	ږ 0696	ڗ 0697	ژ 0698	ڙ 0699	ښ 069A
ڛ 069B	ڜ 069C	ڝ 069D	ڞ 069E	ڟ 069F	ڠ 06A0	ڡ 06A1	ڢ 06A2
ڣ 06A3	ڤ 06A4	ڥ 06A5	ڦ 06A6	ڧ 06A7	ڨ 06A8	ک 06A9	ڪ 06AA
ګ 06AB	ڬ 06AC	ڭ 06AD	ڮ 06AE	گ 06AF	ڰ 06B0	ڱ 06B1	ڲ 06B2
ڳ 06B3	ڴ 06B4	ڵ 06B5	ڶ 06B6	ڷ 06B7	ں 06BA	ڻ 06BB	ڼ 06BC
ڽ 06BD	ھ 06BE	ۀ 06C0	ہ 06C1	ۂ 06C2	ۃ 06C3	ۄ 06C4	ۅ 06C5
ۆ 06C6	ۇ 06C7	ۈ 06C8	ۉ 06C9	ۊ 06CA	ۋ 06CB	ی 06CC	ۍ 06CD

Character list Swiss 721 bold

ێ 06CE	ې 06D0	ۑ 06D1	ے 06D2	ۓ 06D3	۔ 06D4	ە 06D5	۰ 06F0
۱ 06F1	۲ 06F2	۳ 06F3	۴ 06F4	۵ 06F5	۶ 06F6	۷ 06F7	۸ 06F8
۹ 06F9	Ẁ 1E80	ẁ 1E81	Ẃ 1E82	ẃ 1E83	Ẅ 1E84	ẅ 1E85	Ỳ 1EF2
ỳ 1EF3	– 2013	— 2014	― 2015	‗ 2017	‘ 2018	’ 2019	‚ 201A
‛ 201B	“ 201C	” 201D	„ 201E	† 2020	‡ 2021	• 2022	… 2026
‰ 2030	′ 2032	″ 2033	‹ 2039	› 203A	‼ 203C	‾ 203E	⁄ 2044
n 207F	0 2080	1 2081	2 2082	3 2083	4 2084	5 2085	6 2086
7 2087	8 2088	9 2089	Fr 20A3	₤ 20A4	Pt 20A7	₪ 20AA	€ AltGr + E 20AC
℅ 2105	ℑ 2111	ℓ 2113	№ 2116	ℜ 211C	™ 2122	Ω 2126	℮ 212E
ℵ 2135	⅓ 2153	⅔ 2154	⅛ 215B	⅜ 215C	⅝ 215D	⅞ 215E	← 2190
↑ 2191	→ 2192	↓ 2193	↔ 2194	↕ 2195	↨ 21A8	↵ 21B5	⇐ 21D0

Character list Swiss 721 bold

⇑ 21D1	⇒ 21D2	⇓ 21D3	⇔ 21D4	∂ 2202	∆ 2206	∏ 220F	∑ 2211
− 2212	∕ 2215	∙ 2219	√ 221A	∞ 221E	∟ 221F	∩ 2229	∫ 222B
≈ 2248	≠ 2260	≡ 2261	≤ 2264	≥ 2265	⌂ 2302	⌐ 2310	⌠ 2320
⌡ 2321	␡ 2421	─ 2500	│ 2502	┌ 250C	┐ 2510	└ 2514	┘ 2518
├ 251C	┤ 2524	┬ 252C	┴ 2534	┼ 253C	═ 2550	║ 2551	╒ 2552
╓ 2553	╔ 2554	╕ 2555	╖ 2556	╗ 2557	╘ 2558	╙ 2559	╚ 255A
╛ 255B	╜ 255C	╝ 255D	╞ 255E	╟ 255F	╠ 2560	╡ 2561	╢ 2562
╣ 2563	╤ 2564	╥ 2565	╦ 2566	╧ 2567	╨ 2568	╩ 2569	╪ 256A
╫ 256B	╬ 256C	▀ 2580	▄ 2584	█ 2588	▌ 258C	▐ 2590	░ 2591
▒ 2592	▓ 2593	■ 25A0	□ 25A1	▪ 25AA	▫ 25AB	▬ 25AC	▲ 25B2
► 25BA	▼ 25BC	◄ 25C4	◊ 25CA	○ 25CB	● 25CF	◘ 25D8	◙ 25D9

Character list Swiss 721 bold

◦ 25E6	☺ 263A	☻ 263B	☼ 263C	♀ 2640	♂ 2642	♠ 2660	♣ 2663
♥ 2665	♦ 2666	♪ 266A	♫ 266B	fi ZIRKUMFLEX F001	fl F002	, F004	¸ F005
Ģ F006	ģ F007	Ķ F008	ķ F009	Ļ F00A	ļ F00B	Ņ F00C	ņ F00D
Ŗ F00E	ŗ F00F	Ţ F010	ţ F011	F8FF	fi FB01	fl FB02	שׁ FB2A
שׂ FB2B	בּ FB31	גּ FB32	דּ FB33	הּ FB34	וּ FB35	זּ FB36	טּ FB38
יּ FB39	כּ FB3B	לּ FB3C	מּ FB3E	נּ FB40	סּ FB41	ףּ FB43	פּ FB44
צּ FB46	קּ FB47	רּ FB48	שּ FB49	תּ FB4A	וֹ FB4B	ﭖ FB56	ﭗ FB57
ﭘ FB58	ﭙ FB59	ﭪ FB6A	ﭫ FB6B	ﭬ FB6C	ﭭ FB6D	ﭺ FB7A	ﭻ FB7B
ﭼ FB7C	ﭽ FB7D	ﮊ FB8A	ﮋ FB8B	ﮎ FB8E	ﮒ FB92	ﮓ FB93	ﮔ FB94
ﮕ FB95	ﯼ FBFC	ﰈ FC08	ﰎ FC0E	ﰱ FC31	ﰲ FC32	ﰿ FC3F	ﱀ FC40
ﱁ FC41	ﱂ FC42	ﱃ FC43	ﱄ FC44	ﱎ FC4E	ﱞ FC5E	ﱟ FC5F	ﱠ FC60

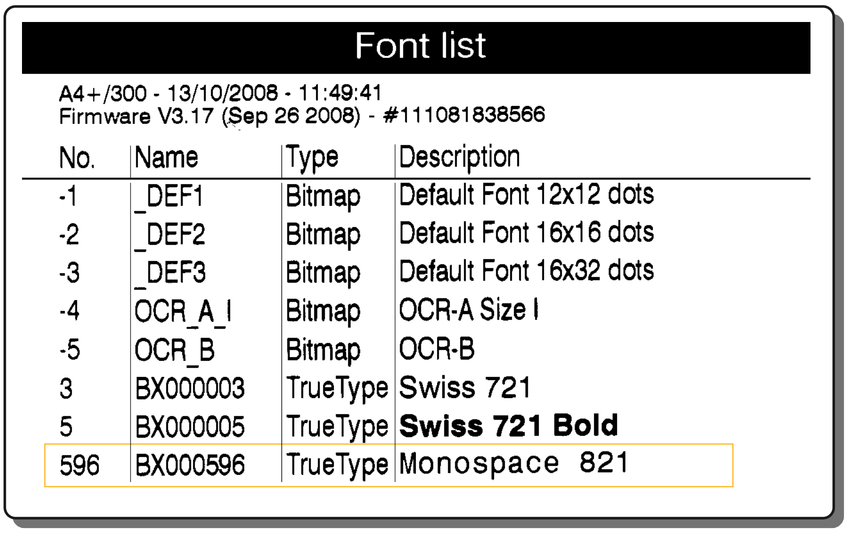
Character list Swiss 721 bold

ﱡ FC61	ﱢ FC62	ﲊ FC8A	ﲍ FC8D	ﱯ FC6F	ﱰ FC70	ﱳ FC73	ﱵ FC75
ﲆ FC86	ﲇ FC87	ﲏ FC8F	ﲑ FC91	ﲒ FC92	ﲜ FC9C	ﲝ FC9D	ﲞ FC9E
ﲟ FC9F	ﲡ FCA1	ﲢ FCA2	ﲣ FCA3	ﲤ FCA4	ﲨ FCA8	ﲪ FCAA	ﲬ FCAC
ﲰ FCB0	ﳉ FCC9	ﳊ FCCA	ﳋ FCCB	ﳌ FCCC	ﳍ FCCD	ﳎ FCCE	ﳏ FCCF
ﳐ FCD0	ﳑ FCD1	ﳒ FCD2	ﳓ FCD3	ﳔ FCD4	ﳕ FCD5	ﳚ FCDA	ﳛ FCDB
ﳜ FCDC	ﳝ FCDD	ﳥ FCE5	ﳻ FCFB	ﳼ FCFC	ﳽ FCFD	ﳾ FCFE	ﴅ FD05
ﴆ FD06	ﴇ FD07	ﴈ FD08	ﴍ FD0D	ﴎ FD0E	ﴏ FD0F	ﴐ FD10	ﴗ FD17
ﴘ FD18	ﴙ FD19	ﴚ FD1A	ﴡ FD21	ﴢ FD22	ﴣ FD23	ﴤ FD24	ﴩ FD29
ﴪ FD2A	ﴫ FD2B	ﴬ FD2C	ﴰ FD30	﴾ FD3E	﴿ FD3F	ﶈ FD88	ﷲ FDF2
ﷺ FDFA	ﹰ FE70	ﹲ FE72	ﹴ FE74	ﹶ FE76	ﹸ FE78	ﹺ FE7A	ﹼ FE7C
ﹽ FE7D	ﹾ FE7E	ﺀ FE80	ﺁ FE81	ﺂ FE82	ﺃ FE83	ﺄ FE84	ﺅ FE85

Character list Swiss 721 bold

ﺆ FE86	ﺇ FE87	ﺈ FE88	ﺉ FE89	ﺊ FE8A	ﺋ FE8B	ﺌ FE8C	ﺍ FE8D
ﺎ FE8E	ﺏ FE8F	ﺐ FE90	ﺑ FE91	ﺒ FE92	ﺓ FE93	ﺔ FE94	ﺕ FE95
ﺖ FE96	ﺗ FE97	ﺘ FE98	ﺙ FE99	ﺚ FE9A	ﺛ FE9B	ﺜ FE9C	ﺝ FE9D
ﺞ FE9E	ﺟ FE9F	ﺠ FEA0	ﺡ FEA1	ﺢ FEA2	ﺣ FEA3	ﺤ FEA4	ﺥ FEA5
ﺦ FEA6	ﺧ FEA7	ﺨ FEA8	ﺩ FEA9	ﺪ FEAA	ﺫ FEAB	ﺬ FEAC	ﺭ FEAD
ﺮ FEAE	ﺯ FEAF	ﺰ FEB0	ﺱ FEB1	ﺲ FEB2	ﺳ FEB3	ﺴ FEB4	ﺵ FEB5
ﺶ FEB6	ﺷ FEB7	ﺸ FEB8	ﺹ FEB9	ﺺ FEBA	ﺻ FEBB	ﺼ FEBC	ﺽ FEBD
ﺾ FEBE	ﺿ FEBF	ﻀ FEC0	ﻁ FEC1	ﻂ FEC2	ﻃ FEC3	ﻄ FEC4	ﻅ FEC5
ﻆ FEC6	ﻇ FEC7	ﻈ FEC8	ﻉ FEC9	ﻊ FECA	ﻋ FECB	ﻌ FECC	ﻍ FECD
ﻎ FECE	ﻏ FECF	ﻐ FED0	ﻑ FED1	ﻒ FED2	ﻓ FED3	ﻔ FED4	ﻕ FED5
ﻖ FED6	ﻗ FED7	ﻘ FED8	ﻙ FED9	ﻚ FEDA	ﻛ FEDB	ﻜ FEDC	ﻝ FEDD

Character list Monospace

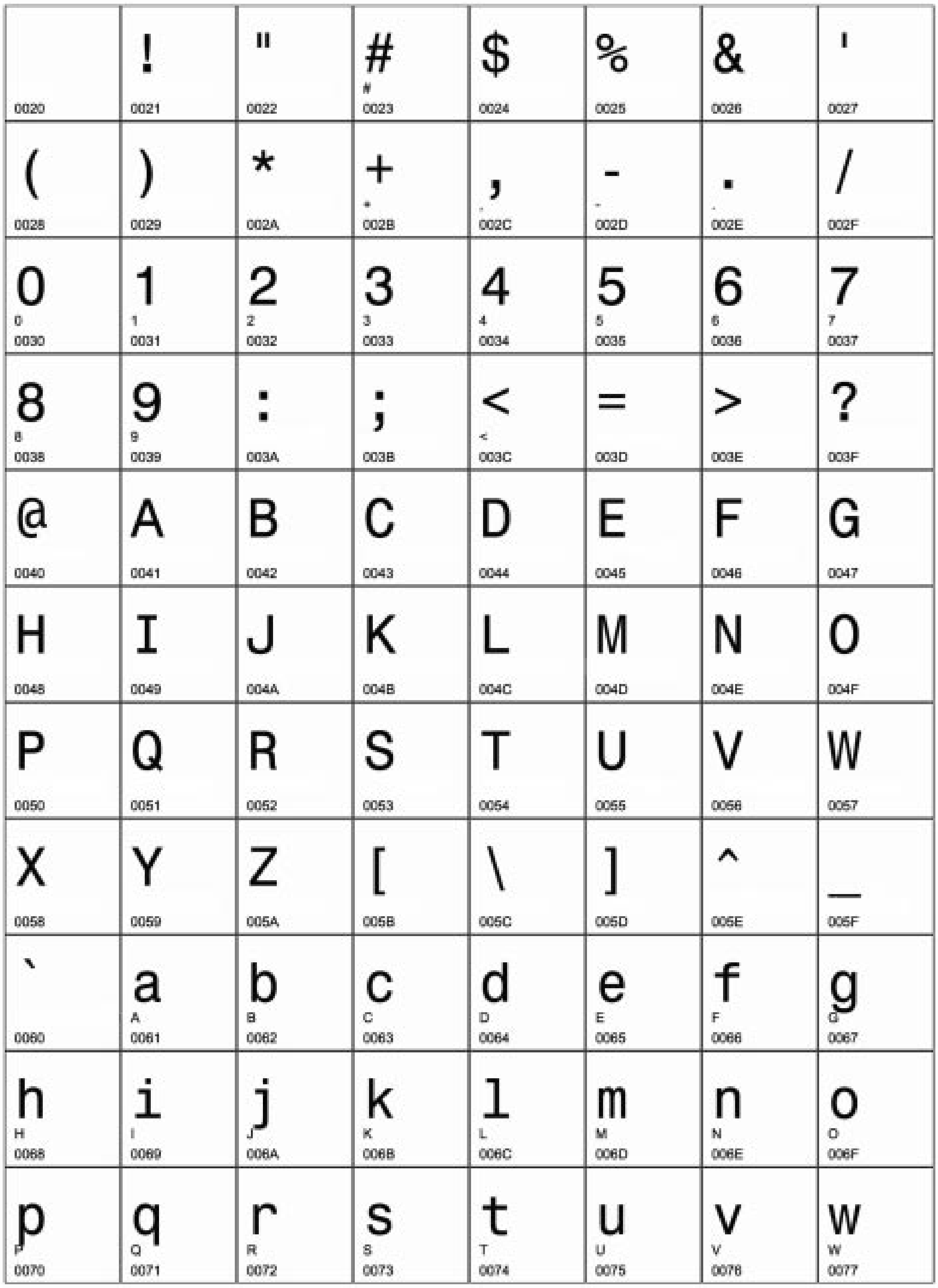


Font list

A4+/300 - 13/10/2008 - 11:49:41
Firmware V3.17 (Sep 26 2008) - #111081838566

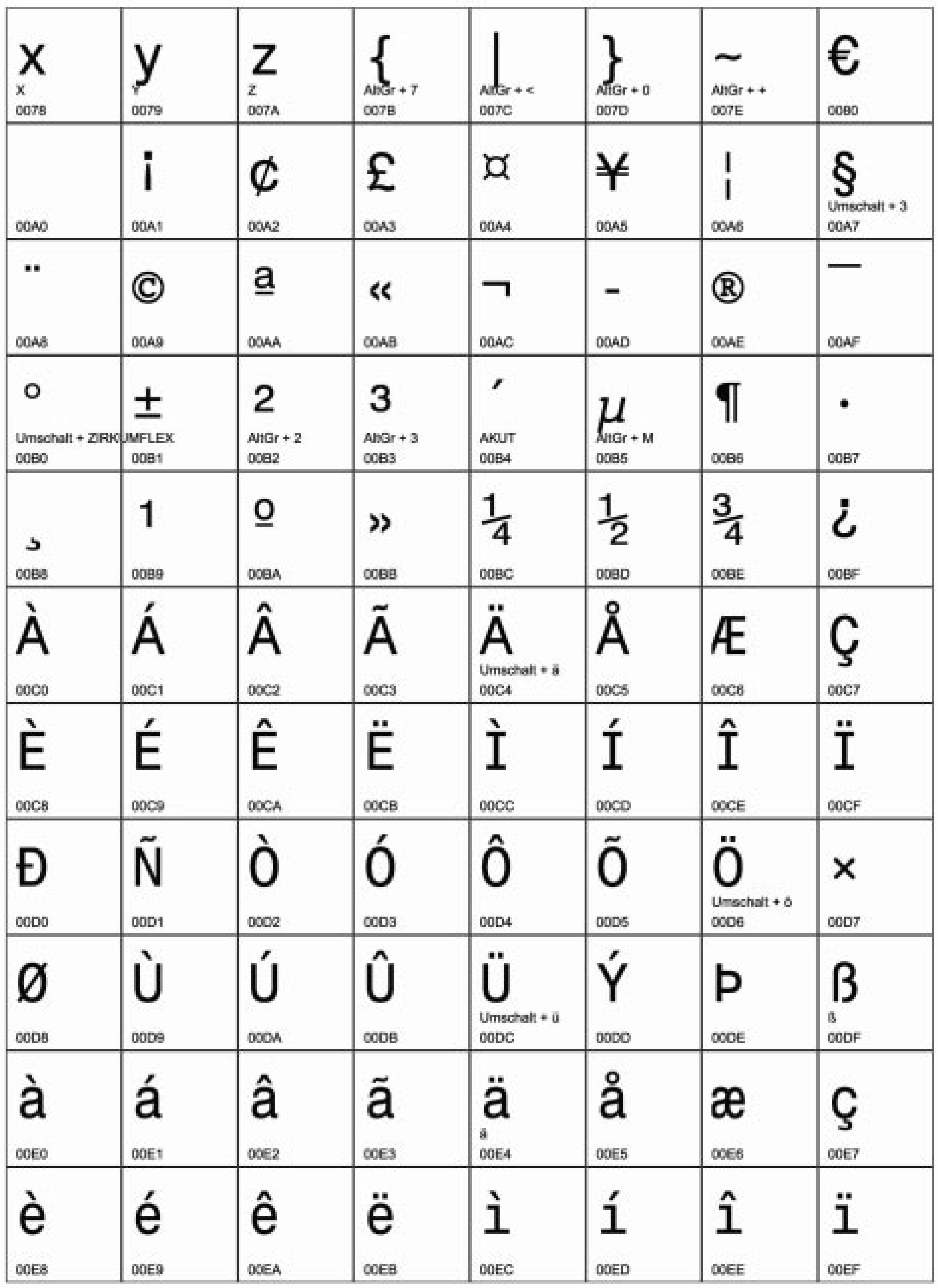
No.	Name	Type	Description
-1	_DEF1	Bitmap	Default Font 12x12 dots
-2	_DEF2	Bitmap	Default Font 16x16 dots
-3	_DEF3	Bitmap	Default Font 16x32 dots
-4	OCR_A_I	Bitmap	OCR-A Size I
-5	OCR_B	Bitmap	OCR-B
3	BX000003	TrueType	Swiss 721
5	BX000005	TrueType	**Swiss 721 Bold**
596	BX000596	TrueType	Monospace 821

Character list Monospace



0020	! 0021	" 0022	# # 0023	$ 0024	% 0025	& 0026	' 0027
(0028	) 0029	* 002A	+ + 002B	, , 002C	- - 002D	. . 002E	/ 002F
0 0 0030	1 1 0031	2 2 0032	3 3 0033	4 4 0034	5 5 0035	6 6 0036	7 7 0037
8 8 0038	9 9 0039	: 003A	; 003B	< < 003C	= 003D	> 003E	? 003F
@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047
H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057
X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
` 0060	a A 0061	b B 0062	c C 0063	d D 0064	e E 0065	f F 0066	g G 0067
h H 0068	i I 0069	j J 006A	k K 006B	l L 006C	m M 006D	n N 006E	o O 006F
p P 0070	q Q 0071	r R 0072	s S 0073	t T 0074	u U 0075	v V 0076	w W 0077

Character list Monospace



x X 0078	y Y 0079	z Z 007A	{ AltGr + 7 007B	| AltGr + < 007C	} AltGr + 0 007D	~ AltGr + + 007E	€ 0080
00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ Umschalt + 3 00A7
¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
° Umschalt + ZIRKUMFLEX 00B0	± 00B1	² AltGr + 2 00B2	³ AltGr + 3 00B3	´ AKUT 00B4	µ AltGr + M 00B5	¶ 00B6	· 00B7
¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä Umschalt + ä 00C4	Å 00C5	Æ 00C6	Ç 00C7
È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö Umschalt + ö 00D6	× 00D7
Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü Umschalt + ü 00DC	Ý 00DD	Þ 00DE	ß ß 00DF
à 00E0	á 00E1	â 00E2	ã 00E3	ä ä 00E4	å 00E5	æ 00E6	ç 00E7
è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF

Character list Monospace

ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö ö 00F6	÷ 00F7
ø 00F8	ù 00F9	ú 00FA	û 00FB	ü ü 00FC	ý 00FD	þ 00FE	ÿ 00FF
Ā 0100	ā 0101	Ă 0102	ă 0103	Ą 0104	ą 0105	Ć 0106	ć 0107
Ĉ 0108	ĉ 0109	Ċ 010A	ċ 010B	Č 010C	č 010D	Ď 010E	ď 010F
Đ 0110	đ 0111	Ē 0112	ē 0113	Ĕ 0114	ĕ 0115	Ė 0116	ė 0117
Ę 0118	ę 0119	Ě 011A	ě 011B	Ĝ 011C	ĝ 011D	Ğ 011E	ğ 011F
Ġ 0120	ġ 0121	Ģ 0122	ģ 0123	Ĥ 0124	ĥ 0125	Ħ 0126	ħ 0127
Ĩ 0128	ĩ 0129	Ī 012A	ī 012B	Ĭ 012C	ĭ 012D	Į 012E	į 012F
İ 0130	ı 0131	Ĳ 0132	ĳ 0133	Ĵ 0134	ĵ 0135	Ķ 0136	ķ 0137
ĸ 0138	Ĺ 0139	ĺ 013A	Ļ 013B	ļ 013C	Ľ 013D	ľ 013E	Ŀ 013F
ŀ 0140	Ł 0141	ł 0142	Ń 0143	ń 0144	Ņ 0145	ņ 0146	Ň 0147

Character list Monospace

ň 0148	ŉ 0149	Ŋ 014A	ŋ 014B	Ō 014C	ō 014D	Ŏ 014E	ŏ 014F
Ő 0150	ő 0151	Œ 0152	œ 0153	Ŕ 0154	ŕ 0155	Ŗ 0156	ŗ 0157
Ř 0158	ř 0159	Ś 015A	ś 015B	Ŝ 015C	ŝ 015D	Ş 015E	ş 015F
Š 0160	š 0161	Ţ 0162	ţ 0163	Ť 0164	ť 0165	Ŧ 0166	ŧ 0167
Ũ 0168	ũ 0169	Ū 016A	ū 016B	Ŭ 016C	ŭ 016D	Ů 016E	ů 016F
Ű 0170	ű 0171	Ų 0172	ų 0173	Ŵ 0174	ŵ 0175	Ŷ 0176	ŷ 0177
Ÿ 0178	Ź 0179	ź 017A	Ż 017B	ż 017C	Ž 017D	ž 017E	ſ 017F
ƒ 0192	Ǧ 01E6	ǧ 01E7	Ǻ 01FA	ǻ 01FB	Ǽ 01FC	ǽ 01FD	Ǿ 01FE
ǿ 01FF	ʼ 02BC	ʽ 02BD	ˆ 02C6	ˇ 02C7	ˉ 02C9	˘ 02D8	˙ 02D9
˚ 02DA	˛ 02DB	˜ 02DC	˝ 02DD	; 037E	΄ 0384	΅ 0385	Ά 0386
· 0387	Έ 0388	Ή 0389	Ί 038A	Ό 038C	Ύ 038E	Ώ 038F	ΐ 0390

Character list Monospace

Α 0391	Β 0392	Γ 0393	Δ 0394	Ε 0395	Ζ 0396	Η 0397	Θ 0398
Ι 0399	Κ 039A	Λ 039B	Μ 039C	Ν 039D	Ξ 039E	Ο 039F	Π 03A0
Ρ 03A1	Σ 03A3	Τ 03A4	Υ 03A5	Φ 03A6	Χ 03A7	Ψ 03A8	Ω 03A9
Ϊ 03AA	Ϋ 03AB	ά 03AC	έ 03AD	ή 03AE	ί 03AF	ΰ 03B0	α 03B1
β 03B2	γ 03B3	δ 03B4	ε 03B5	ζ 03B6	η 03B7	θ 03B8	ι 03B9
κ 03BA	λ 03BB	μ 03BC	ν 03BD	ξ 03BE	ο 03BF	π 03C0	ρ 03C1
ς 03C2	σ 03C3	τ 03C4	υ 03C5	φ 03C6	χ 03C7	ψ 03C8	ω 03C9
ϊ 03CA	ϋ 03CB	ό 03CC	ύ 03CD	ώ 03CE	Ё 0401	Ђ 0402	Ѓ 0403
Є 0404	Ѕ 0405	І 0406	Ї 0407	Ј 0408	Љ 0409	Њ 040A	Ћ 040B
Ќ 040C	Ў 040E	Џ 040F	А 0410	Б 0411	В 0412	Г 0413	Д 0414
Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C

Character list Monospace

Н 041D	О 041E	П 041F	Р 0420	С 0421	Т 0422	У 0423	Ф 0424
Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C
Э 042D	Ю 042E	Я 042F	а 0430	б 0431	в 0432	г 0433	д 0434
е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C
н 043D	о 043E	п 043F	р 0440	с 0441	т 0442	у 0443	ф 0444
х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C
э 044D	ю 044E	я 044F	ё 0451	ђ 0452	ѓ 0453	є 0454	ѕ 0455
і 0456	ї 0457	ј 0458	љ 0459	њ 045A	ћ 045B	ќ 045C	ў 045E
џ 045F	Ґ 0490	ґ 0491	ְ 05B0	ֱ 05B1	ֲ 05B2	ֳ 05B3	ִ 05B4
ֵ 05B5	ֶ 05B6	ַ 05B7	ָ 05B8	ֹ 05B9	ֻ 05BB	ּ 05BC	ֽ 05BD
־ 05BE	ֿ 05BF	׀ 05C0	ׁ 05C1	ׂ 05C2	׃ 05C3	ׄ 05C4	א 05D0

Character list Monospace

ב 05D1	ג 05D2	ד 05D3	ה 05D4	ו 05D5	ז 05D6	ח 05D7	ט 05D8
י 05D9	ך 05DA	כ 05DB	ל 05DC	ם 05DD	מ 05DE	ן 05DF	נ 05E0
ס 05E1	ע 05E2	ף 05E3	פ 05E4	ץ 05E5	צ 05E6	ק 05E7	ר 05E8
ש 05E9	ת 05EA	װ 05F0	ױ 05F1	ײ 05F2	׳ 05F3	״ 05F4	، 060C
؛ 061B	؟ 061F	ء 0621	آ 0622	أ 0623	ؤ 0624	إ 0625	ئ 0626
ا 0627	ب 0628	ة 0629	ت 062A	ث 062B	ج 062C	ح 062D	خ 062E
د 062F	ذ 0630	ر 0631	ز 0632	س 0633	ش 0634	ص 0635	ض 0636
ط 0637	ظ 0638	ع 0639	غ 063A	ـ 0640	ف 0641	ق 0642	ك 0643
ل 0644	م 0645	ن 0646	ه 0647	و 0648	ى 0649	ي 064A	ً 064B
ٌ 064C	ٍ 064D	َ 064E	ُ 064F	ِ 0650	ّ 0651	ْ 0652	٠ 0660
١ 0661	٢ 0662	٣ 0663	٤ 0664	٥ 0665	٦ 0666	٧ 0667	٨ 0668

Character list Monospace

٩ 0669	ی 06CC	۰ 06F0	۱ 06F1	۲ 06F2	۳ 06F3	۷ 06F7	۸ 06F8
۹ 06F9	Ẁ 1E80	ẁ 1E81	Ẃ 1E82	ẃ 1E83	Ẅ 1E84	ẅ 1E85	Ỳ 1EF2
ỳ 1EF3	– 2013	— 2014	― 2015	‗ 2017	‘ 2018	’ 2019	‚ 201A
‛ 201B	“ 201C	” 201D	„ 201E	† 2020	‡ 2021	• 2022	… 2026
‰ 2030	′ 2032	″ 2033	‹ 2039	› 203A	‼ 203C	‾ 203E	⁄ 2044
n 207F	0 2080	1 2081	2 2082	3 2083	4 2084	5 2085	6 2086
7 2087	8 2088	9 2089	₣ 20A3	₤ 20A4	₧ 20A7	₪ 20AA	€ AltGr + E 20AC
℅ 2105	ℑ 2111	ℓ 2113	№ 2116	ℜ 211C	™ 2122	Ω 2126	℮ 212E
ℵ 2135	⅓ 2153	⅔ 2154	⅛ 215B	⅜ 215C	⅝ 215D	⅞ 215E	← 2190
↑ 2191	→ 2192	↓ 2193	↔ 2194	↕ 2195	↨ 21A8	↵ 21B5	⇐ 21D0
⇑ 21D1	⇒ 21D2	⇓ 21D3	⇔ 21D4	∂ 2202	∆ 2206	∏ 220F	∑ 2211

Character list Monospace

− 2212	∕ 2215	∙ 2219	√ 221A	∞ 221E	∟ 221F	∩ 2229	∫ 222B
≈ 2248	≠ 2260	≡ 2261	≤ 2264	≥ 2265	⌂ 2302	⌐ 2310	⌠ 2320
⌡ 2321	␡ 2421	─ 2500	│ 2502	┌ 250C	┐ 2510	└ 2514	┘ 2518
├ 251C	┤ 2524	┬ 252C	┴ 2534	┼ 253C	═ 2550	║ 2551	╒ 2552
╓ 2553	╔ 2554	╕ 2555	╖ 2556	╗ 2557	╘ 2558	╙ 2559	╚ 255A
╛ 255B	╜ 255C	╝ 255D	╞ 255E	╟ 255F	╠ 2560	╡ 2561	╢ 2562
╣ 2563	╤ 2564	╥ 2565	╦ 2566	╧ 2567	╨ 2568	╩ 2569	╪ 256A
╫ 256B	╬ 256C	▀ 2580	▄ 2584	█ 2588	▌ 258C	▐ 2590	░ 2591
▒ 2592	▓ 2593	■ 25A0	□ 25A1	▪ 25AA	▫ 25AB	▬ 25AC	▲ 25B2
► 25BA	▼ 25BC	◄ 25C4	◊ 25CA	○ 25CB	● 25CF	◘ 25D8	◙ 25D9
◦ 25E6	☺ 263A	☻ 263B	☼ 263C	♀ 2640	♂ 2642	♠ 2660	♣ 2663

Character list Monospace

♥ 2665	♦ 2666	♪ 266A	♫ 266B	fi ZIRKUMFLEX F001	fl F002	, F004	د F005
Ģ F006	ģ F007	Ķ F008	ķ F009	Ļ F00A	ļ F00B	Ņ F00C	ņ F00D
Ŗ F00E	ŗ F00F	Ţ F010	ţ F011	F8FF	ﬁ FB01	ﬂ FB02	שׁ FB2A
שׂ FB2B	בּ FB31	גּ FB32	דּ FB33	הּ FB34	וּ FB35	זּ FB36	טּ FB38
יּ FB39	כּ FB3B	לּ FB3C	מּ FB3E	נּ FB40	סּ FB41	ףּ FB43	פּ FB44
צּ FB46	קּ FB47	רּ FB48	שּ FB49	תּ FB4A	וֹ FB4B	ی FBFC	ﹰ FE70
ﹲ FE72	ﹴ FE74	ﹶ FE76	ﹸ FE78	ﹺ FE7A	ﹼ FE7C	ﹾ FE7E	ﺁ FE81
ﺍ FE8D	ﺏ FE8F	ﺓ FE93	ﺕ FE95	ﺙ FE99	ﺝ FE9D	ﺡ FEA1	ﺥ FEA5
ﺩ FEA9	ﺫ FEAB	ﺭ FEAD	ﺯ FEAF	ﺱ FEB1	ﺵ FEB5	ﺹ FEB9	ﺽ FEBD
ﻁ FEC1	ﻅ FEC5	ﻉ FEC9	ﻍ FECD	ﻑ FED1	ﻕ FED5	ﻙ FED9	ﻝ FEDD
ﻡ FEE1	ﻥ FEE5	ﻫ FEEB	ﻭ FEED	ﻯ FEEF	ﻱ FEF1	� FFFD	

Index

Index

Index

Index

Index

Index

Index