Barcode & OCR Printing

Technical Reference Manual

Version: 1.51

Abstract

This document contains platform-independent and solution-independent information about OCR text and Barcode printing on our devices, as well as general background information about PCL fonts and barcodes, and about the printer languages PCL and HP-GL/2. For each barcode symbology supported, important information about the symbology (attributes, encoding), the checksum algorithms, and the code construction procedure, is provided.

C o n t e n t s

Notice

RICOH COMPANY LTD. MAKE NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Ricoh Company Ltd. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

No part of this document may be photocopied, reproduced, or translated to another language without the prior written consent of Ricoh Company Ltd.

This document is only available in English.

Trademarks
Microsoft®, Windows®, Microsoft Windows™, Windows 95™, MS Windows and MS are registered trademarks of Microsoft Corporation in the United States and/or other countries.
PCL® is a registered trademark of Hewlett-Packard Company.
All named company names and/or products are trade names or registered trade names of the named companies.
Other product names used herein are for identification purposes only and might be trademarks of their respective companies. We disclaim any and all rights in those marks.

Important notice
Parts of this document are subject to change without prior notice.

1   Introduction

This document contains platform-independent and solution-independent information about OCR text and barcode printing on our devices, and about the barcode symbologies supported.

It also contains general background information about PCL and HP-GL/2 programming, because barcoding solutions may require changes in the PDL data stream. Specifically, it also contains information about PCL fonts, because there exist font-based barcoding solutions.

For each barcode symbology or barcode application supported, it contains important information about the symbology attributes and encoding, the checksum algorithms, and the code construction procedure.

For information related to barcode printing on a specific platform (operating system, application, etc), please refer to corresponding separate documentation.

For information related to using a specific solution, please refer to corresponding separate documentation.

For information related to a specific device, please refer to Appendix D.

1.1   Terminology and Notation

This section defines the terminology and notational conventions which are used throughout this document.
For a full list, refer to the Glossary in Appendix G.

Acronyms

CHK = checksum or check digit (if clear from context) or check character or checksum method
CTX = clear text (human readable text), or clear text style
HDD = hard disk (drive/download)
DIMM = "Dual In-line Memory Module"
SDcard = "Secure Digital Card"
RAM = "Random Access Memory"

Font characters or Message characters

Characters of the Latin-1 codepage are coded in 1 byte.
Therefore, the ASCII codes of characters are in the range 000..255 (dec) = 00..FF (hex).
The following notation is used:
  <ddd> = **dec** = decimal
  <xx>   = **hex** = hexadecimal

You can enter any character in an **editor**, by entering its decimal ASCII code on the keyboard as **<Alt> + <0ddd>**.
In order to see all characters as printed in this document, the editor needs to be in Latin-1 codepage (Windows codepage 1252).

Special characters

Notation	Name	(hex)	(dec)	Meaning / Usage / Remarks
<esc>	Escape	<1B>	<027>	in PCL commands: start of command
<SI>	Shift In	<0F>	<015>	in PCL commands: select primary font
<SO>	Shift Out	<0E>	<014>	in PCL commands: select secondary font
<CR>	Carriage Return	<0D>	<013>	carriage return
<LF>	Line Feed	<0A>	<010>	line feed
<FF>	Form Feed	<0C>	<012>	page feed
<SP>	Space	<20>	<032>	blank space
<DEL>	Delete	<7F>	<127>	DEL character
<UEL>	Universal Exit Language			<esc>%-12345X

For acronyms and notation related to barcodes, please refer to the chapter "Barcodes" below.

1.2   Technical support

For any technical support, please **contact your local representative**.

You may **obtain technical support** If you have general questions about or problems with barcode printing on our devices.

1.3   Limited warranty

Documentation

The general information provided in this document about fonts, printer languages, and barcodes (especially the CHK algorithms and encodings), has been carefully researched and assembled. However, it is only intended to serve as a quick reference, it is not to replace any official specification. We cannot guarantee the correctness and completeness of the information given.
In case of doubt, please refer always to the official specification.

The correctness of the **CHK algorithms** of the following symbologies or applications cannot be guaranteed, for the reason indicated:
- (a compatible barcode reader is not available for verification):
  - MSI mod 11 mod 10
  - USPS FIM
  - Danish PTT 39
  - French Postal A/R 39
  - 4-state postal codes
- (a compatible decoding application was not available for verification):
  - UPC-128
  - UCC/EAN-128
  - UPS-128
  - PZN (Pharma-Zentral-Nummer)
  - German Post AG Leitcode / Identcode
- (an inconsistent behaviour of different barcode readers was observed):
  - 2 of 5 Industrial
  - 2 of 5 Matrix

2   Fonts (General information)

This chapter contains general information about **printer device fonts**, and specifically it explains the support of soft PCL device fonts on our printer devices.

The following issues are explained:
- The nature of PCL fonts (font locations, font characteristics), the font selection methods, and the concept of primary and secondary fonts.
- The syntax of PCL soft font data (font header, character descriptor).
- The PCL Font list.
- For RAM and HDD fonts, the download mechanism and the command syntax.
- For HDD download fonts, the HDD Directory list.
- For DIMM fonts, the Font DIMM.
- For SDcard fonts, the Font SDcard.

2.1   PCL fonts

A **font** can be defined as a set of character bitmaps or as an algorithm (which may take certain parameters) which can generate a glyph bitmap for each character with certain attributes.

This chapter describes the following issues, with particular focus on PCL:
- Where can fonts be stored ?
- Attributes / characteristics of a font
- Selection / usage of a font

2.1.1   Font location

Fonts can be **stored/located** at different places, either outside or inside the printer, either in printer ROM or on some medium attached to the printer.

The following list contains the possible locations of fonts, listed in order of ascending priority.
Fonts are searched for in order of this priority.

Term	Location	Prio
• **Internal / Resident font (ROM font)**	Printer ROM	low
• **SIMM/DIMM font**	Font SIMM/DIMM	|
• **SDcard font**	Font SDcard	|
• **Cartridge font**	Font cartridge	|
• **HDD download fonts**	Printer hard disk (HDD)	|
• **RAM Soft / Download fonts ("permanent")** (lowest ID has highest priority)	Printer RAM	| |
• **RAM Soft / Download fonts ("temporary")** (lowest ID has highest priority)	Printer RAM	| high

To **use/access** a non-ROM font, it must exist in the printer RAM.
Therefore, the firmware first needs to load/transfer/copy it from the medium (SIMM/DIMM, SDcard, HDD, etc) into the printer RAM.

2.1.2   Font characteristics

The following **font characteristics** exist.
They are listed in order of decreasing priority, as when used by the font selection by characteristics mechanism.

- **Symbol set**   = the set of printable symbols
- **Spacing**   = fixed / proportional inter-character spacing
- **Pitch**   = horizontal size; in cpi (# chars / inch)
- **Height**   = vertical size; in point (1 pt = 1/72")
- **Style**   = posture (upright/italic), width (condensed, normal, expanded), structure (solid, outline, shadow)
- **Stroke weight**   = normal / bold / etc.
- **Typeface family** = the overall appearance / shape of characters
- **Resolution**   = in dpi (dots per inch)
- **Location**   = (see above)
- **Orientation**   = logical page with respect to the physical page

The combination of these attributes determines the character bitmap, as it will appear on the page.

2.1.3 Font selection methods

Once the font is available to the device (printer controller), and properly listed in the device's font list, it can be **accessed/selected** through various methods.

PCL also provides the concept of **primary** and **secondary** fonts. See below.

Selection by Characteristics

The command sequence to be used is:
```
  <esc> ( <symbol set> <esc> ( s <spacing> p [<pitch(*)> h] [<height(*)> v] <style> s <weight> b
<typeface> T
```

Example: <esc>(8U <esc>(s 1p 12v 1s 3b 4148T

The syntax for each font is indicated in the device's PCL Font list.
Please refer to the section "Selecting the font" in the chapter "PCL" below.

2.1.4 The device's PCL Font list

Every controller maintains a **PCL Font list**, i.e. a list of (device) fonts available under PCL.

Alternative names are:
- Font list
- PCL Configuration page

Some models even combine both lists into one.

The method to print out these lists, and their contents and appearance, differ from model to model.

The procedure to print out the PCL Font list

Usually you press the <Menu> button and navigate to something like "Printer settings" --> "PCL" --> "Font list" --> "List print" --> ...

For the **exact procedure to print out the PCL Font list** for a particular device, please refer to the device-specific information in <u>Appendix DPF</u>, or look it up in the corresponding Operating Instructions manual of the device.

The contents of the PCL Font list

The fonts are categorized into:
- Internal fonts = ROM fonts = Resident fonts (- Scalable, - Bitmapped)
- "Permanent" soft fonts (RAM)
- "Temporary" soft fonts (RAM)
- HDD / Disk fonts = Download fonts (Printer hard disk) (- Scalable, - Bitmapped)
- DIMM/SIMM fonts or SDcard fonts
- Cartridge fonts

The RAM fonts are sorted by the IDs which were used in the download command.
The HDD fonts are sorted alphabetically by the external name with which each font was downloaded.
The DIMM fonts and SDcard fonts are sorted by the order in which they are contained on the DIMM or SDcard, respectively.

The list shows, for each available PCL font, several of these attributes (device-dependent):
- font #/no.
- font ID
- symbol set
- spacing fixed/proportional
- typeface
- style
- (stroke) weight
- pitch (cpi) [not for proportional fonts]
- point size (font height) [not for scalable fonts]
- (default) orientation
- sample printed text, usually "012..ABC..abc..."
- font select command syntax

2.2 [SUPPORT ONLY]

2.3 [SUPPORT ONLY]

2.4 HDD fonts / The Printer HDD

One possibility to store a soft font on the printer is to do so on the printer's internal hard disk (HDD).

2.4.1 [SUPPORT ONLY]

2.4.2 The device's HDD Directory list

Every printer controller maintains a **HDD Directory list**, i.e. a list of all files residing on the HDD (including the PCL font files).

Alternative names are: "PCL Configuration page".
Some models combine both lists into one.

The procedure to print out the HDD Directory list

The procedure to print out this list, and their contents and appearance, differs from model to model.
Usually you press the <Menu> button, and navigate to something like "Printer settings" --> "PCL" --> "Directory list" --> "List print" --> ...

For the exact **procedure to print out the HDD Directory list** for a particular device, please refer to the device-specific information in Appendix DPF, or look it up in the corresponding Operating Instructions manual of the device.

The contents of the HDD Directory list

These attributes are listed:
- total disk capacity, disk free space, # Directories, # Files
- for each downloaded file:
  - Directory path
  - File name (case-sensitive)
  - File size (# bytes)

The HDD fonts are sorted alphabetically by the external name with which each font was downloaded.

Note: Though paths are sometimes displayed on the printout with a "/" (forward slash), however, in all commands above always a "\" (backslash) needs to be used !

Note: There is no indication of which font a file corresponds to, unless the name was chosen in a mnemonic fashion.

2.5 DIMM fonts / The Font DIMM

Another possibility is to store a set of soft fonts on a DIMM ("Dual In-line Memory Module"), which is plugged into a slot on the printer controller.

The printer controller knows how to access these fonts.
From the PCL controller's view they can be used transparently as any other device font:
- They can be accessed/selected as any other device font.
- They appear on the PCL Font list as any other device font.

2.5.1 Our Font DIMM

The Font DIMM provided is named "Barcode Font DIMM Type A".
It can hold 4 MB of data.
It fits into a slot on the printer controller board.

For a list of the models that support a Font DIMM, refer to Appendix D.

All PCL font formats are supported.

2.5.2 DIMM insertion procedure

Install the DIMM module
To install the DIMM module in the printer device, it needs to be inserted in the free DIMM slot on the printer controller board.
For the exact **procedure to install the DIMM module** for a particular device, please refer to the device-specific information in Appendix DPD.

Uninstall the DIMM module
Remove the DIMM module from the printer device.

2.5.3 [SUPPORT ONLY]

2.6 SDcard fonts / Font SDcards

Yet another possibility is to store a set of soft fonts on an SDcard ("Secure Digital Card"), which is plugged into a slot on the printer controller.

The printer controller knows how to access these fonts.
From the PCL controller's view they can be used transparently as any other device font:
• They can be accessed/selected as any other device font.
• They appear on the PCL Font list as any other device font.

2.6.1 Our Font SDcard

There is a Font SDcard ("Barcode & OCR Font SDcard") provided with "Barcode & OCR Package (BOP) for SAP - SDcard package -".
It fits into a slot on the printer controller board.

For a list of the models that support a Font SDcard, refer to Appendix D.

All PCL font formats are supported.

2.6.2 SDcard insertion procedure

Install the SDcard
To install the SDcard in the printer device, it needs to be inserted in a free suitable SDcard slot on the printer controller board.
For the exact **procedure to insert the SDcard** for a particular device **and which of the slots to use**, please refer to Appendix DPS.

Uninstall the SDcard
Remove the SDcard from the printer device.

2.6.3 [SUPPORT ONLY]

2.7 [SUPPORT ONLY]

3   The Printer languages (General information)

The **parameters** for the OCR text or barcode determine eventually the **command sequence** that has to be sent to the printer device.
The **printer languages** supported by our printer devices and supporting PCL soft fonts, are **PCL5** and **HP-GL/2**.
The following sections contain the necessary background information about drawing text for each language.
For each language this information is provided:
- General information (command syntax, coordinate system)
- Selecting a font
- Drawing text
- Sizing text
- Rotating text
- Positioning text
- Complete command sequence

Note: (HP-GL/2 versus PCL)
The main reason to use HP-GL/2 besides PCL is that it allows independent scaling of fonts in both dimensions.
Moreover, the HP-GL/2 command sequence is actually even easier to understand and to specify than PCL, because the width and height parameters can be directly specified in cm, without any conversion into pt, and because HP-GL/2 doesn't use ASCII control characters (below <032>).

For detailed information about PJL, PCL5, and HP-GL/2, refer to the standard literature:
• **[ ]** HP PCL 5e Technical Reference Manual : chs 9-11
• **[ ]** HP PCL 5 Comparison Guide
• **[ ]** HP PJL Technical Reference Manual

3.1   PCL

The HP **PCL Printer Language** (PCL) is a **page description language** (PDL).
That means it is used to specify the contents and format of a page.
"PCL" is the acronym for "Printer Command Language", which is the name of the printer language developed by Hewlett-Packard.

PCL5e is the version that our black-and-white devices use.
PCL5c is the version that our color devices use. It is a superset of PCL5e.

Full documentation for PCL commands is available in the "PCL5 Printer Language Technical Reference Manual" by HP.

3.1.1   General information

The Command syntax

The general syntax of a PCL command is:
```
 <esc> <character> <letter> <value> <letter>
```

Note:
• Please be aware that the syntax is case-sensitive.
• Please be aware that PCL commands may contain both a zero (0) or an uppercase 'O', which may look the same, depending on the font.
  Similar for "l" = lowercase L and the digit "1".
    E.g. in "\e&l^^^H" etc., "l" is the lowercase letter "L" not the digit "1" !
    E.g. in "\e&l^^^O" etc., "O" is the uppercase letter "O" not the digit "0" !
• Do NOT introduce any extra blanks.
• Do NOT introduce any line feeds. The PCL statements have to be entered in one line! Any line break will cause the printer also to move to the next line, which may void any previous cursor positioning.

Note: (Condensing commands)
Multiple successive PCL commands with the same initial character can be condensed into one single command, by converting the terminating character of the previous command to lowercase and omitting the first three characters of each command.
Example: "<esc>(s1P <esc>(s12V <esc>(s1S <esc>(s3B <esc>(s4148T" can be shortened to "<esc>(s1p12v1s3b4148T".

Note: (Using an editor)
If the file contains hex <00>, a normal editor such as NotePad would convert them into spaces <20>. In such case using a Hex editor is mandatory. Normally you can also use the DOS Editor. You can enter any character by first pressing <Ctrl>+<P>, then enter the decimal ASCII code on the numeric key pad while keeping the <Alt> key pressed.
Example: At the end of the text you may want to perform a Form Feed. Enter <Ctrl>+<P>, then <Alt>+12. The "female" sign appears. An <esc> sign is displayed as an arrow to the left.

Note: (Macros)
PCL allows you to define and execute or call **macros**. A macro is identified by a number (<#>).
• `<esc>&f<#>y0X`   start define macro #
• `<esc>&f<#>y1X`   stop define macro #
• `<esc>&f<#>y2X`   execute macro #
• `<esc>&f<#>y3X`   call macro #

The Coordinate system

For PCL, the origin of the logical page is at the upper left corner. Positive directions are, therefore, right and down.

Measurements

Units under PCL for vertical and horizontal measurements can be:
- **PCL unit**   = (user-definable through the <esc>&u#D command)
- **deci-point**   = 1/720 inch (resolution-independent)
- **dots**   = (resolution-dependent)
- **Columns / Rows**   = (depending on current HMI/VMI)

3.1.1.1   [SUPPORT ONLY]

3.1.2 Selecting the font

A typical PCL **font select command sequence** looks like this.
```
<esc>(0N <esc>(s1p40v0s0b3T
```

It is the condensed version of the sequence of these commands:
```
<esc>(0N
<esc>(s1P
<esc>(s40V
<esc>(s1S
<esc>(s3B
<esc>(s4148T
```

The syntax and meaning of each is explained in the table below.

`<esc> ( <ID>`	Symbol set (e.g. "0N" = Latin-1)
`<esc> ( s <#> P`	Spacing (0 = fixed, 1 = proportional)
`<esc> ( s <#> H`	Pitch (in cpi = characters per inch) (*)
`<esc> ( s <#> V`	Height (in points, 1 pt = 1"/72) (*)
`<esc> ( s <#> S`	Style (0 = upright, 1 = italic)
`<esc> ( s <#> B`	Stroke weight (0 = medium, 3 = bold)
`<esc> ( s <#> T`	Typeface family (e.g. 4099 = "Courier")

Note: (*) (<height> versus <pitch>)
Whether the "<pitch> **h**" or the "<height> **v**" command needs to be specified in a font select command, depends on the following scheme.

Scalability: Spacing:		**Bitmapped** (format 0,20)	**Scalable** (format 10,11,15)
- fixed	0p	both **h** and **v**	only **h**
- proportional	1p	only **v**	only **v**

Note: (Full versus short font selection method)
All of these characteristics should always be specified (**full font selection method**).
As a minimum, only the characteristics of the new font that differ from those of the previously designated font must be sent (**short font selection method**). The other characteristics are then retained from the previously selected font, where applicable. E.g. "`<esc>(s25V`" would just change the size of the currently selected primary font.

Note: (Font selection of non-resident fonts)
An HDD download or a DIMM or SDcard font appears in the PCL Font list.
It can be selected in the same way as if it were a ROM resident font, e.g. by its characteristics.
The command is listed on the PCL Font list.
I.e. the selection method is transparent with respect to the location of the font.

Using the Primary/Secondary Font mechanism

PCL has the concept of a "primary" and a "secondary" font, which allows you to quickly toggle between two frequently used fonts, without having to reselect them every time anew.

To <u>define</u> each of them, use the above font select commands with "`<esc>(...`" for the primary font, and "`<esc>)...`" for the secondary font.

To <u>toggle / switch</u> between them, just send the following character/byte:

`<SI>` `= <0F>` `= <015>`	"Shift In". Switch to the primary font.

`<SO>    = <0E>   = <014>`	"Shift Out". Switch to the secondary font.

3.1.3 Drawing text

To draw text, it just needs to be sent to the printer as is, i.e. the sequence of ASCII codes of its characters.
It will be drawn in the currently active font.

To send **non-printable** ASCII control character codes (range <00> .. <1F>), the following "Transparent Print Data" command needs to be prefixed:

`<esc> & p <# bytes> X`	"Transparent Print Data"

3.1.4 Sizing text

The font select command sequence specifies a certain size (height and/or pitch).
To change it, these commands can be used:

`<esc> ( s <height> V`	
`<esc> ( s <pitch> H`	

3.1.5 Orienting/Rotating text

To achieve rotation, specify the `<esc>&a^^^P` command with one of the parameters 90,180,270.
To cancel rotation, the `<esc>&a0P` command needs to be sent.

`<esc> & a ^^^ P`	

3.1.6 Positioning text

The positioning of text and graphics is done through an internal "PCL cursor".
Its current position is identified through horizontal and vertical coordinates.
It is automatically **advanced implicitly**, e.g. after printing a character or a blank space or a tab or a line feed.
It can also be **set explicitly** through the following commands.

Cursor positioning

For positioning the cursor on the page, PCL offers three different possibilities.
The first method is recommended, because it is fully independent of any current printer setting.

`<esc> & a ^^^ H`	Horizontal move. The value ^^^ is measured in **deci-pt** = 1/720 inch (1 inch = 2.54 cm). A **relative** movement can be performed by putting a plus(+) or minus(-) sign in front of the positioning value. For example `<esc>&a-720H` positions the cursor 1 inch towards the left.
`<esc> & a ^^^ V`	Vertical move. Analogous to the horizontal move. Positive direction moves down. <u>Example:</u> `<esc>&a566.9h1133.86V` positions the cursor about 2 cm from the left and 4 cm from the top margin.
`<esc> * p ^^^ X`	Horizontal move. The value ^^^ is measured in **PCL units**. This method depends on the actual resolution (300/600 dpi).
`<esc> * p ^^^ Y`	Vertical move. Analogous to the horizontal move.
`<esc> & a ^^^ C`	Horizontal move. The value ^^^ is specified as number of **columns**. This method depends on the current HMI, i.e. the currently used font.
`<esc> & a ^^^ R`	Vertical move. The value ^^^ is specified as number of **rows**. This method depends on the current VMI.

Moreover, the current cursor position can be saved by pushing it on a stack, and later be reverted to by popping it off again.
This is useful if you want to leave the actual printing position, do some output somewhere else, and then return to the previous position.

`<esc> & f 0 S`	Push Cursor. The actual cursor position is stored on the stack.
`<esc> & f 1 S`	Pop Cursor. Returns to the last pushed cursor position.

Note:
Note that the origin of the PCL character cell is at its upper left corner:

3.1.7 The complete command sequence

Based on the previous sections, here is a typical PCL command sequence to draw text in a certain size, orientation, and position.

Example

```
...
<esc>&f0S                             -- Save the actual cursor position.
<esc>&a720h720V                       -- Set the cursor to 25.4 mm off the top and left page margin.
<esc>)0Y<esc>)s1p20v0s0b9012T         -- Define Code 39 as secondary font.
<esc>&a90P                            -- Rotate the printing position by 90 degrees counterclockwise.
Serial Nr.                            -- Print this text using the primary font.
<0E>                                  -- Activate the secondary (barcode) font.
*12345*                               -- Barcode message data, including Start/Stop character.
<0F>                                  -- Return to the primary font.
<esc>&a0P                             -- Return to the original printing direction.
<esc>&f1S                             -- Return to the original cursor position.
...
```

3.2 HP-GL/2

HP-GL/2 is a subset of PCL5, originally used for plotter operation.

3.2.1 General information

The Command syntax

An HP-GL/2 command is a 2-letter mnemonic operation code, followed by a list of parameters, separated by commas, and optionally terminated by a semicolon.

Note:
Semicolons are delimiters for commands, but can be omitted in most cases.
They should be kept for better readability, but can be omitted to save space.

A typical HP-GL/2 **initialization** sequence is:

`<esc>%1B`	-- enter HP-GL/2 mode
`IN;`	-- initialize
`SP1;`	-- select black pen
`...`	
`<esc>%0A`	-- exit HP-GL/2 mode

The Coordinate system

For HP-GL/2, the origin of the logical page is at the lower left corner. Positive directions are, therefore, right and up.
(This is different from PCL, where the origin is at the upper left corner.)

Note: (HP-GL/2 Picture Frame)
HP-GL/2 draws only those parts of images that lie inside the "HP-GL/2 Picture Frame".
By default, it has the considerably large distance of 0.5 inch from the top and bottom edge.
In order to be able to draw closer to the paper's edges, the Picture Frame has to be enlarged first.
To do so, the following command has to be sent, under PCL.

```
<esc>&l0O <esc>&l0E <esc>*c5953x8419Y <esc>&a0h0V <esc>*c0T
```

It has to be (re-)sent after any change of orientation, i.e. after every "`<esc>&l__O`".
The values above are specific to allowing edge-to-edge printing (i.e. for the maximum possible width and height) of a Portrait document on DIN A4 paper (e.g. width = 210 mm = 210 mm x 1 inch / 25.4 mm x 720 deci-pt / inch = 5953 deci-pt).

Measurements

Coordinates/offsets and measures are specified in "**current units**".
These can be:
- "**plotter units**" (**plu**): 1 plu = 0.025 mm; => 400 plu = 1 cm, 1016 plu = 1 inch
- "**user units**" = can be defined through the SC command;

A table of all useful HP-GL/2 commands can be found below.

3.2.1.1   HP-GL/2 commands

Only those commands needed for text printing are listed here.
For full information, refer to the HP-GL/2 related chapters of the "HP PCL5 Technical Reference Manual".
For each command, the [chapter-page] of its definition is indicated below.

HP-GL/2 commands by Category

Code	Description
	=== Characters / Text ===
LB \<text\> \<delimiter\>	"Label"; [23-71] -- draw text -- This command implies an automatic Pen Down.
DT \<delimiter\> ,1	"Define Label Terminator"; [23-54] -- 1 = non-printing -- default = \<ETX\> = hex \<03\>
	=== Character Sizing === -- (only possible w/ scalable fonts)
SI \<width\> ,\<height\>	"Absolute Character Size"; [23-89] -- nominal character width and CAP height (in cm)
SR \<width\> ,\<height\>	"Relative Character Size"; [23-98] -- dto (as percentage of P1-P2)
ES \<width\> [,\<height\>]	"Extra Space"; [23-62] -- modify inter-character spacing and line spacing
	=== Pen positioning (Horizontal/Vertical) ===
PU [\<x\> ,\<y\>]	"Pen Up"; [20-54] -- pen up, then move to (x,y): Horizontal/Vertical cursor positioning
PD [\<x\> ,\<y\>]	"Pen Down"; [20-39] -- pen up, then draw line to (x,y)
PA [\<x\> ,\<y\>]	"Plot Absolute"; [20-37] -- enter absolute move mode
PR [\<x\> ,\<y\>]	"Plot Relative"; [20-52] -- enter relative move mode
LO	"Label Origin"; [23-74] --
CP	"Character Plot"; [23-30] -- only in terms of spaces and lines
	=== Coordinate system ===
RO \<angle\>	"Rotate coordinate system"; [19-38] -- rotates Print direction by (0/90/180/270) degrees
SC 0,\<xFac\>,0,\<yFac\>,2	"Scale"; [19-45,19-52] -- define user units in terms of plotter units (plu); with scale factors \<xFac\>, \<yFac\>; Example: SC 0,400,0,400,2 => set 1 user-unit = 400 plu = 1 cm
	=== Control ===
\<esc\> % 1 **B**	enter/switch to HP-GL/2; [18-16] - 0= revert to previous HP-GL/2 pen position - **1**= place pen to current PCL cursor position
\<esc\> % 0 **A**	exit HP-GL/2, switch to PCL; [18-17] - **0**= revert to previous PCL cursor position - 1= place cursor to current HP-GL/2 pen position
CO " \<text\> "	"Comment"; [19-21]
DF	"Default values"; [19-21]
IN	"Initialize"; [19-24] -- initialize HP-GL/2
SP \<pen #\>	"Select Pen"; [22-48] - 1 = black pen
	== Fonts: -- select fonts
SS	"Select Standard Font"; [23-103] -- select/make primary font as current -- requires previous assignment with SD or FI
SA	"Select Alternate Font"; [23-79] -- select/make secondary font as current -- requires previous assignment with AD or FN

`<SI>`	"Shift-in" (hex <0F>) -- switch to primary font (only works inside of label text)
`<SO>`	"Shift-out" (hex <0E>) -- switch to secondary font (only works inside of label text)
`FI <fontID>`	"Select Primary Font"; [23-65] -- assign PCL font as primary -- requires previous assignment of <fontID> under PCL (using *cD (s...T *cF)) **Note:** This does not yet select / switch to the primary font.
`FN <fontID>`	"Select Secondary Font"; [23-68] -- dto., as secondary
	== Fonts: -- define/specify fonts
`SD <kind>,<value>, ...`	"Standard Font Definition"; [23-82] -- select primary font by attributes; analogous to AD
`AD <kind>,<value>, ...`	"Alternate Font Definition"; [23-23] -- select secondary font by attributes
`AD 1 ,<#>`	Font: Symbol set (default= 277 / Roman-8)
`AD 2 ,<#>`	Font: Spacing (0= fixed, 1= proportional) (default= 0 / fixed)
`AD 3 ,<#>`	Font: Pitch (in cpi) (default= 9 cpi)
`AD 4 ,<#>`	Font: Height (in pt) (default= 11.5 pt)
`AD 5 ,<#>`	Font: Style / Posture (1= italic) (default= 0 / upright)
`AD 6 ,<#>`	Font: Stroke weight (3= bold) (default= 0 / medium)
`AD 7 ,<#>`	Font: Typeface family (default= 48 / "Stick")

3.2.2 Selecting the font

There are several possibilities to select a font.

	select by attributes	call by ID	set current
● primary:	`SD ...;`	`FI ...;`	`SS;`
● secondary:	`AD ...;`	`FN ...;`	`SA;`

The recommended sequence is:

```
...
<SO>                        -- switch to secondary font
<esc>)...                   -- normal PCL font select command sequence for secondary font
<esc>*c99d6F                -- copy to RAM and assign font ID # 99
<esc>%1B                    -- enter HP-GL/2 mode
SA                          -- select alternate/secondary font
FN99;                       -- select secondary font ID # 99
...
```

3.2.3 Drawing text

To just draw a first draft of the barcode, regardless of its size, orientation, and position, the following command sequence needs to be sent to the printer.

```
... LB <text> <delimiter> ; ...
```

By default, the **<delimiter>** is the character/byte <ETX> = hex <03>.
On some platforms, this ASCII code may raise problems; then a different <delimiter> needs to be specified.
It needs to be chosen in such a way that it is:
- supported by the platform's data stream architecture, and
- never occurring as part of the <text>

The command to redefine it is:

```
... DT <delimiter> ; ...
```

Example
```
...
DT <9F> ;
...
LB <text> <9F> ;
...
```

3.2.4 Sizing text

With a pure PCL font select command, the width and height of the font can only be scaled proportionally; i.e. their ratio always remains the same.
The HP-GL/2 language offers the possibility to scale a font in both dimensions independently.
This allows you to customize the look of the barcodes to match very closely to those printed with your previously existing barcode applications.
This will ensure continued scannability, and they will continue to fit in your predesigned forms.

The command to specify the **width** and the **height** of subsequently drawn text is:

```
... SI <width> , <height> ; ...
```

Both measures have to specified in centimeters **[cm]**.

Example: ...SI0.13,1.5;...

Note: (SI command)
(a) The SI command may not be omitted.
  Otherwise the size of any previously specified SI command will remain effective.
(b) The width and height should always be explicitly specified in the SI command.
  Normally, if the SI command is specified without parameters, the PCL specified size is used.
  Unfortunately this may not work with some devices.
(c) If two SI commands are sent, the second takes precedence over the first.

Note: In the case of a barcode font, the clear text underneath the barcode is scaled in the same way ! So it may actually look somewhat distorted.

Determining the size parameters

For some of the barcode fonts, the clear text and in some cases the bars as well extend below the baseline.
Therefore, the measured value is not always the same as specified.

It is possible to compute exactly the width and height parameters necessary to specify in the SI command, using the font metrics information of the fonts. Since this is rather awkward to do, the following approach is much easier:

1. First draw a draft barcode at some reasonable nominal size, using the HP-GL/2 command

```
SI0.25,1.0;
```

2. Then measure the observed size.

3. Then compute the required parameters by this straightforward formula (rule of three):

```
(Required parameter)  =  (Nominal parameter) x (Desired size) / (Measured size)
```

This needs to be done for both the width and the height independently. It is much quicker than operating with tables.

Inter-character space

In case your barcode reader needs some extra **inter-character space** (for discrete symbologies), this can be achieved using the ES (Extra Space) command:

```
... ES <factor> ; ...
```

This should be left at zero (ES0,0;).
The unit is the default font HMI/pitch (width of the space character).

Example: ...ES0.12;... => +12% of normal space character width is added between characters.

3.2.5 Orienting/Rotating text

For specifying **rotation**, these commands can be used:

```
...RO90;...
...RO180;...
```

```
...RO270;...
```

To switch back to no rotation, this command needs to be issued:

```
...RO0;...
```

Example: `...RO90;LBHello<9F>;RO0;...`

Note: (RO command)
If two RO commands are sent, the second takes precedence over the first.

Note: The PCL rotate command (`<esc>&a#P`) has no effect on HP-GL/2 drawing.

3.2.6 Positioning text

Due to the design of the barcode fonts, the reference point of the font characters does not always coincide with the lower left corner of the first bar. Therefore, some additional adjustment of the position may be required.

For fine-tuning the positioning of text, you can use this HP-GL/2 command sequence to specify a **relative position offset**:

```
...
PR ;
PU <horz.offset> , <vert.offset> ;
PD ;
...
```

PR switches to "Plot Relative" mode. (It suffices to specify this once at the beginning of the job; so it can be omitted later.)

PU raises the pen and moves it by the specified (relative) offset.
The 1st parameter specifies a horizontal offset, the 2nd parameter specifies a vertical offset.
All values can have a decimal point and a minus sign.
Positive direction is towards the right and up.
The offsets can be specified in centimeters [cm], if previously, e.g. at the beginning of the job, the following **scaling** command was sent, which defines a user unit as 400 plu ("plotter units") = 1 cm.
Due to the "PR" (Plot relative) command sent before, the PU parameters are interpreted as relative offsets instead of absolute coordinates.

```
...
SC0,400,0,400,2;
...
```

PD lowers the pen again. (This is not really necessary since eventually the LB command will imply an automatic pen-down.)

Example: `...PU1.2,-0.5;...` moves the pen to the right and down.

Note: (Origin of character cell)
Note that the origin of the HP-GL/2 character cell is at its lower left corner: (This is different from PCL !)

Note: (PU command)
If two PU commands are sent, the second takes precedence over the first.

Note: (Order of PU and RO commands)
The PU command should always precede a RO command, if any. For a PU command is interpreted relative to the current coordinate system, which is changed by the RO command.

Note:
When entering and exiting HP-GL/2, the cursor position may remain at the current position or revert to the position before entering the language. That behaviour can be controlled through the parameters of the Enter/Exit HP-GL/2 commands `<esc>%<#>A/B`.

3.2.7 The complete command sequence

As a result of the discussion of the previous sections, the **typical command sequence** to draw text using the secondary font, is depicted below.

Command	Meaning
...	**(enlarge HP-GL/2 Picture Frame)**
`<esc>&l0O`	-- set orientation (here: Portrait)
`<esc>&l0E`	-- clear top margin

`<esc>*c`**`5953`**`x`**`8419`**`Y`	-- set HP-GL/2 Picture frame size (here: for DIN A4 Portrait)
`<esc>&a0h0V`	-- set cursor to (0,0)
`<esc>*c0T`	-- set HP-GL/2 Picture frame anchor point
`...`	
`...`	**(re-initialize HP-GL/2)**
`<esc>%1B`	-- enter HP-GL/2 mode; pen position at current PCL cursor position;
`IN;`	-- initialize HP-GL/2
`SP1;`	-- select black pen
`PR;`	-- set default plotting mode to "Relative"
`DT<9F>;`	-- define delimiter (any character not used in the message text)
`SC0,400,0,400,2;`	-- define 1 user-unit as 400 plu = 1 cm
`RO0;`	-- set rotation to 0
`<esc>%0A`	-- exit HP-GL/2 mode
`...`	
`...`	**(actual printing of text)**
`<SO>`	-- switch to secondary font
`<font select command>`	-- normal PCL font select sequence
`<esc>*c99d6F`	-- copy to RAM and assign font ID # 99
`<esc>%1B`	-- enter HP-GL/2 mode
`SA`	-- select alternate/secondary font
`FN99;`	-- select secondary font ID # 99
`PU<xoffs>,<yoffs>;`	-- **position** cursor (relative offset in cm)
`RO<angle>;`	-- **rotate**
`SI<width>,<height>;`	-- **sizing** font: specify width and height (in cm)
`LB`	-- print data
`<text>`	-- actual message text
`<9F>;`	-- delimiter defined above
`<esc>%0A`	-- exit HP-GL/2 mode and return to PCL; the cursor is positioned at the previous PCL cursor position (!)
`<SI>`	-- switch to primary font
`...`	

Note:
If necessary (for command length reasons), recurring or frequently used parts of this sequence may need to be stored in a PCL macro.

Note:
The following choices are arbitrary and can be replaced, but are subject to some restrictions:

● 99	Temporary font ID assigned under PCL to allow access from HP-GL/2.
● <9F>	Character to be used as the text delimiter for the LB command; shows as "Ÿ". The default is hex <03> = <ETX>. Any byte can be used that is supported by the environment but will not occur as part of the <text>.

Note:
A piece of HP-GL/2 code should always be "self-contained", i.e. be independent of the influence of possible preceding alien HP-GL/2 code.
To achieve that, the "IN" command may be (re-)sent and/or other commands required for the proper functioning may have to be (re-)sent.

4   Barcodes (General information)

This chapter provides general background information about barcodes that may be needed to understand the subsequent chapters.

4.1 Introduction

Barcodes are used for conveying information related to an object in a reliably machine-readable form attached to its outside surface.

This can be an ID code to identify the object (e.g. an article number or invoice number), or additional infomation about the object, such as routing information (e.g. postal code), price, production date, weight, etc.

Other machine-readable forms besides barcodes are:
- magnetic stripe
- transponder
- OCR characters

Each of these has its own advantages and disadvantages related to the field where it is to be used.

A 1-D barcode encodes such information in a black-and-white rectangle-shaped pattern ("bars" and "spaces").
A 2-D barcode can potentially use arbitrary patterns.
From here on, the terms "barcode" or "symbology" shall refer to the narrower sense of 1-D barcodes.

Usually the actual data is preceded by a unique start character and followed by an optional check character and a unique stop character.
Different types of barcodes, so-called symbologies, exist to accommodate the different requirements of the area of application, and to allow for more than one barcode per object. They differ by several characteristics.

The most common 1-D symbologies are:
  Code 128, Code 39, Code 93, Codabar, 2 of 5 Interleaved, 2 of 5 Industrial, 2 of 5 Matrix, MSI/Plessey, UPC/EAN-x, etc.

4.1.1 [SUPPORT ONLY]

4.2 Definitions / Terminology

To avoid misunderstandings, a clear terminology is needed.
For a quick definition of a term, see the Glossary in Appendix G of this document.

A **barcode** encodes information in a pattern of rectangles of different brightness (usually black-and-white).
The information is encoded binarily in the relative physical widths of the (dark) bars and (light) spaces.
It can thus be optically read and decoded.
(Compared to OCR characters the error rate is much lower.)

Different types of barcodes, so-called **symbologies**, have been developed to accommodate the different requirements of the application, and to allow for more than one barcode on an object. They differ by several characteristics, which are described below. The definition of a symbology comprises the definition of the character set and an algorithm describing how data is encoded.

The following two phases have to be considered:
- **Tagging**/Encoding:
  To tag an object with a barcode, the data needs to be **encoded**, possibly a check character needs to be calculated, and then the actual barcode pattern needs to be **printed** and **affixed** to the surface of the object.
- **Reading**/Decoding:
  At the point and time where the information is needed, the barcode needs to be **read** by a scanning unit (barcode reader), and then the binary stream obtained needs to be **decoded** and interpreted as data.

Barcodes can be constructed in various ways:

1-dimensional barcodes (**1-D**, **linear** barcode)
A (1-dimensional) **barcode** is a linear sequence of vertical **bars** (of essentially the same height) separated by **spaces** (gaps).
The barcode itself starts with a special **start character** (guard bar) and ends with a special **stop character**.
The actual **data characters** may be followed by one or more **check characters**.
Every character or pair of characters is encoded as a sequence of bars and spaces.
The widths of the bars and spaces are measured in multiples of a "module".

2-dimensional barcodes (**2-D**)
A 2-dimensional barcode is a rectangular pattern consisting of smaller rectangles or other shapes (hexagons, ...) in different shades of grey or colors or just black-and-white.
2-D barcodes have a much higher information density and can thus convey information from a database, rather than just the object ID as a key into the database.
The following possibilities exist:
- A **stacked (multi-row)** barcode is just like a long 1-D barcode wrapped into a rectangle.
- A **matrix code** is a true 2-D barcode (potentially an arbitrary pattern of dots).
- **Bar-height encoding** is used e.g. by POSTNET barcodes. Information is not encoded in the widths of bars and spaces, but the bars may have 2 or 3 different heights.

A barcode has to be delimited from other non-blank areas (e.g. other barcodes on the same surface) by a sufficiently wide blank **space zone / quiet zone** on both sides.

4.3 Characteristics of a symbology

These determine its redundancy and error-checking and -correcting capabilities, and the recognition rates.
These are usually designed to meet the requirements of a specific application area (such as retail POS, mail delivery, library, etc), scanning technology, and data (ID, article number, weight, etc).

A **symbology** is characterized by the following attributes.

=== (by definition) ===

- The **dimensionality** indicates if it is a 1-D or 2-D barcode.

- The **character set** (**alphabet**) describes the set of encodable characters.
  E.g. numeric, alphanumeric/text, full ASCII, special characters.
  Some characters are non-ASCII and may be neither printable nor human readable, they only have code-internal relevance (e.g. start/stop characters, <CODEx>, <FNCx>, etc). They only convey information for the decoding algorithm, not for the message data.

- The **encoding scheme** is an algorithm describing how data is encoded and decoded.
  Sometimes this can be as simple as a table assigning a pattern of bars and spaces to every character, or just a binary value.
  This implies the following two attributes.

- In a **continuous** symbology all spaces convey information.
  In a **discrete** symbology the inter-character spaces convey no information. They are always Narrow.

- With each character a numerical reference value (**reference number**) is associated, which is used for checksum calculation.

- A symbology is **multi-level** if bars and spaces have different widths.
  In an **N-level** symbology the widths of the elements are multiples 1,2,..,N of the module width. Typical values are 2 or 4.

- The **# elements per character** describes how many elements (bars + spaces) are used to encode a particular character. It can be fixed (the same for all characters) or variable.
  In a continuous symbology it is even, in a discrete symbology it is odd.

- The **length** of the barcode defines how many characters can be encoded in 1 barcode. It can be fixed or variable.

- A symbology is **self-checking** if the code contains a certain degree of redundancy that allows the reader to detect or even correct read errors. This means that the binary barcode encoding of each character already includes some parity bit, which allows to detect or correct the misreading of any single character. If this is not the case, an explicit check character may be needed.

- The optional or mandatory existence of 1 or more **check characters**. They are usually appended behind the data characters, before the stop character. Check characters are calculated using a **checksumming algorithm**. It is parametrized by a sequence of **weights** and the **modulus** (usually the number of codable characters). For each data character its numerical representation (e.g. the number corresponding to the binary representation of its code) is multiplied with the weight corresponding to its position in the data sequence, and the overall sum is then reduced with respect to the modulus.
  Adding a check character makes the code even safer against possible misreadings. Most codes are, however, self-checking.

- **Bidirectional reading**:
  Can the barcode be read in both directions ? This requires the start and stop characters to be distinguishable.
  All major symbologies are bidirectional.

=== (by typical usage/application) ===

- Areas / Environments of usage:
  e.g. commerce (POS), photo laboratories, industry, health industry, medical/pharma, mail/courier, production line, library checkout, invoicing, etc.

- Values/Data/Information encoded:
  e.g. ID (object ID, personal ID), article number, price, weight, customer number, destination, postal code, routing information, etc.

- Region of usage:
  i.e. world-wide, USA/Canada, Europe, Japan, etc.

A (basic) **symbology** defines the character set and its encoding scheme.

A symbology **application** is based on a **base symbology,** but may deviate by
- the character set (extended or restricted)
- additional check characters / special checksum algorithm
- special interpretation of characters at certain positions (e.g. <FNC1> in UCC/EAN-128)
- restrictions of length
- etc.
E.g. an **extended symbology** extends the character set by pairing or escaping characters from the character set of its base symbology.

4.4 Encoding schemes

The encoding scheme of an <u>application</u> maps each character into a sequence of characters of the message character set of the corresponding base symbology.

The encoding scheme of a <u>symbology</u> maps each character of its character set into a <u>numerical representation</u>, e.g. a bit sequence.
This numerical sequence can be visualized graphically in different ways:

1-D -- Linear

As a sequence of bars and spaces in different thicknesses (module widths).
There are two different ways/schemes to denote this in a numeric fashion. Both can be translated 1:1 into each other.

Binary/Bit notation
Only digits 0 and 1 are used. A "1" indicates a bar, a "0" indicates a space. Occasionally also "B" and "S" are used.
The number of consecutive digits indicates the thickness of the element.
E.g. 1000011100 = bssssbbbsss.
Often this encoding is just the binary representation of the reference value of the character, plus some extra redundancy/self-check bit(s).

Width notation
This notation counts the number of consecutive '0' and '1' bits.
The values for bars and spaces are alternating. Each value indicates the thickness of the element (expressed in # modules).
The smallest element can be expressed as "1". If there are only 2 thicknesses, also "N"(narrow) and "W"(wide) could be used.
E.g.
 - 2211 or WWNN   = wide bar - wide space - narrow bar - narrow space   = bW-sW-bN-sN   = 110010
 - 1432   = bar(1) - space(4) - bar(3) - space(2)   = b1-s4-b3-s2   = 1000011100

<u>Note</u>:
If spaces convey no information (all narrow) (e.g. 2 of 5 Industrial), they may be omitted from the sequence in this notation.
E.g. "a1b1c1d1..." --> "abcd..."

1-D -- Bar-height encoded

As a sequence of equidistant bars with (2 or 3) different heights.
A **2-state** code uses 2 baselines and has 2 possible values (e.g. POSTNET).
A **4-state** code uses 3 baselines and has 4 possible values (e.g. KIX).

1-D -- Bar-width encoded

As a sequence of bars of different width, separated by narrow spaces.
E.g. 2 of 5 Industrial.

1-D -- Space-width encoded

As a sequence of narrow bars, separated by spaces of different width.
E.g. German Post encoding of the address on letter mail.

2-D

In a 2-dimensional barcode the information is encoded as a rectangular pattern consisting of smaller rectangles or other shapes (hexagons, ...) in different shades of grey or colors or just black-and-white.

2-D -- Stacked

A **stacked (multi-row)** barcode is just like a long 1-D barcode wrapped into a rectangle.
E.g. PDF417.

2-D -- Matrix

A **matrix code** is a true 2-D barcode (potentially an arbitrary pattern of dots).
E.g. DataMatrix or UPS Maxicode.

4.5 [SUPPORT ONLY]

4.6 [SUPPORT ONLY]

4.7 Attributes of a barcode

An actual **barcode** (i.e. as generated and printed for a particular message), has the following **attributes**.

- the **symbology** of the barcode

- the **encoded data**

- the presence or absence of 1 or more **check characters**, and the **checksum method** with which they were generated

- the **height** of the barcode, i.e. the height of the bars

- the **module width X** or **density**
  The physical width of the smallest element (bar/space).
  The width of each element (bars, spaces) is a multiple of X.
  The tolerances (admissible range of values) are defined in the specification of the symbology.
  A high density means a small value of X, and vice-versa.
  The density has a tradeoff of readability, i.e. recognition rate, versus the overall width.
  The module width X is usually specified in units of mil = 1/1000 inch. Typical values are 10 mil or 13 mil.
  For symbologies with a fixed # modules per character, the density can also be expressed in cpi (characters per inch).

- the **Narrow:Wide ratio (N:W ratio)**
  The ratio of the width of the smallest element to the width of the next wider element.
  In order to be able to distinguish between wide and narrow elements, a minimum narrow-to-wide ratio is needed, within a certain range of tolerance.
  This ratio has a tradeoff of readability, i.e. recognition rate, versus the overall width.
  Typically it lies in the range 1:2.0 .. 1:3.0.

- the **overall width** of the barcode:
  It is determined by the length of the message (# characters), the number of elements per character, and the module width / density.

- the **orientation** of the barcode:
  The barcode can be printed **rotated** (vertical) or in normal (horizontal) orientation.

- the **clear text style** of the barcode:
  Is the data in addition printed as **human readable text** or not.
  If yes: where (position relative to the barcode, e.g. below or on top), and in which font.

Therefore, to print data as a barcode, these attributes have to be specified as **parameters**.
The command syntax to specify them, and the place(s) where to do it, depends on the barcode mechanism of the platform and the mechanism of the solution looked at.

4.8 [SUPPORT ONLY]

4.9 [SUPPORT ONLY]

4.10 Barcode quality / readability issues

Depending of the quality/tolerance of the barcode reader, the reading environment conditions (air pollution/dust, distance of object from reader, speed, shape of surface, etc), and the maximum admissible failure rate, a certain minimum quality of a barcode is required.

There are several factors affecting the quality of the barcode itself, and thus its readability.
Besides the physical attributes of the barcode, there is also some influence from the printing environment.

Factors inherent to the barcode attributes

Height
A large height makes it easier to aim or increases the chance to be hit by the laser beam.

Module width / Density, N:W ratio, Total width
The bigger the module width, the better the readability.
The bigger the N:W ratio, the better the readability.
However, if the total width of the barcode becomes too large, the barcode may not fit on the object anymore, or it may be too wide for the barcode reader. So there is always a reasonable compromise to be found for this tradeoff.

Factors inherent to the printing environment

For rectangular bars, it makes not much of a difference if they were generated from a font or through draw rectangle commands.

Physical thickness of bars
Depending on certain conditions, bars may actually appear thicker than their nominal width (as specified by the font size or the rectangle size command). Since this makes the spaces even smaller, the bar:space width ratio can become heavily biased, resulting in non-readability.

This is influenced by the **toner density** (Toner saving mode) and the **toner quality**. It may also be influenced by other printer controller settings.

Contrast / Reflexion
The better the contrast (ratio of reflexion) of the white background to the black toner, the better the readability.
To increase the whiteness of the background, use white paper, and make sure the development unit (OPC drum, corona, etc) does not cause any black spots.
To increase the blackness of the toner, use fresh toner and do not use Toner saving.

Sharpness of edges
If the edges are uneven or fuzzy, the barcode reader will measure/observe random deviations from the actual widths.
This can be caused by a bad engine (laser timing, drum rotation, etc). Switching between SEF and LEF paper feed direction may help.

Resolution
If the resolution is too low, the width ratios of narrow and wide bars and spaces become biased, due to rounding errors.
This may become critical with high density codes. A resolution of 600 dpi is recommended.

4.11 Limitations of barcode solutions

A particular barcode solution may have several limitations. They can be classified/categorized as follows.

4.11.1 Platform-specific limitations

Limitations specific to the platform/environment of the issuing system. These could affect:
- the max. number of message characters
- the message character set (some preliminary mapping may be needed)
- intermediate character sets involved

4.11.2 Device-specific limitations

Limitations inherent in the capabilities of the printer device (printer controller, HDD, DIMM, SDcard, etc). E.g.:
- proper support of required PDL commands

4.11.3 Solution-specific related limitations

Limitations specific to the particular solution implementing barcode printing.
For example, if barcode fonts are employed:
- limitations generally inherent in the (font-based) approach and technology used
- limitations particular to the specific fonts of this product
E.g.
- independent scalability of width and height of bars
- individual scaling of the widths of bars and spaces of each thickness
- clear text style and placement
- special clear text requirements /with Code 39/93 Extended; with check characters; ...)
- special clear text styles/conventions (e.g. German Post Leitcode: "40472.074.006.00" or Pharma-Zentral-Nummer (PZN): "PZN-1234567")

4.12 Troubleshooting barcode printing

This section contains information about troubleshooting barcode printing, both in general and specific to the font-based approach.

For **barcode reader specific** information, please refer to its handbook.
For **device-specific** information, please refer also to the corresponding Operation Instructions manual for the particular device.
For **platform-specific** information, please refer to corresponding separate documentation (printing mechanism, barcode mechanism, font mechanism).
For **solution-specific** information, please refer to the Troubleshooting section in separate documentation.
For **font-specific** information, please refer to the section about the particular barcode font.

As always:
1. identify the different **stages** the **data stream** goes through (from the barcode specification on the issuing system to the printed image), and
2. identify the **active** and **passive components** which apply changes to the data stream.

Data stream: Stages
- printed image
- PDL command stream
- intermediate data
- barcode specification in original document

Active Components:
- barcode reader
- printer device controller
- platform-specific mechanisms
- solution-specific mechanisms

Passive Components:
- fonts (this: solution)
- command fragments (platform-specific)

- command fragments (solution-specific)

To determine where things go wrong, try to intercept/capture the data stream at different stages.
Always confirm that the firmware version of the printer controller of the device is within specification.

4.12.1 The isolation steps

This section contains a list of symptoms, listed in chronological order.
Starting from the beginning, try to find the first symptom that is applicable to your situation.
For each symptom, possible reasons are given, with some explanation and a description how to verify or refute that it is actually the case, and how to fix/solve it.

=== Symptom: No font files on HDD Directory list === [BOP-HDD only]

Reason: **The HDD fonts are not downloaded correctly.**
Try downloading again. Check the connection to the printer.

=== Symptom: No "Status Indication Files" on HDD Directory list === [IBS/BOCR only]

Reason: **BOCR was not installed correctly.**
Try installing again. Check the connection to the printer.

=== Symptom: No barcode/OCR fonts on PCL Font list ===

Reason: **The DIMM or SDcard is not correctly installed.**
Check its proper seat, or re-insert it firmly. Reboot the printer.

Reason: **The HDD fonts are not downloaded correctly.**
Reason: **The font files are corrupted.**
Reason: **The font data is invalid.**
Make sure the correct original font files are used.
Try downloading/installing again.

=== Symptom: No barcode characters in barcode font samples on PCL Font list ===

Reason: **The font select sequence fails.**
Contact support.

Note: The sample barcode characters on the PCL Font list cannot be read by a barcode reader, because there are no start/stop characters.

=== Symptom: No barcode characters on Font Test sheet / Barcode Test Sheet ===

Reason: **The font Test sheets may be corrupted.**
Make sure the original Font test sheet file is used.
Try sending it again. Check the connection to the printer.

Note: The barcode characters in the ASCII table on the front page cannot be read by a barcode reader, because there are no start/stop characters.

=== Symptom: Barcodes on Barcode Test Sheet not readable ===

Reason: **The print or paper quality is beyond tolerance.**
Check the paper quality. Check the toner quality.

Reason: **The barcode reader is not working properly.**
The barcode reader may be broken.
Does it scan any barcode at all ?
Check the cabling.
Try using a different barcode reader.

Reason: **The barcode reader does not support this symbology and CHK method.**
Use a suitable barcode reader.

Reason: **The barcode reader is not configured properly.**
Confirm that the symbology is enabled.
Confirm that the proper CHK method for this symbology is supported by and enabled on the barcode reader.
Does it scan other barcodes ?

Reason: **The environment conditions are beyond tolerance of the barcode reader.**
Try cleaning up.

<u>**Reason**</u>**: The quality of the background is beyond tolerance.**
Try using white paper for optimal reflexion and contrast.

<u>**Reason**</u>**: A wrong font is being selected.**
Compare the relative widths of the bars and spaces with the Logical encoding specification of the symbology.

=== Symptom: A particular barcode is not printed at all or not printed properly ===

<u>**Reason**</u>**: The command sequence is wrong.**
There may be something wrong with the command sequence. Check the command sequence arriving at the printer is correct.

-- Some characters are missing appearing **blank**.

-- Some characters appear as a **void marker** (Ø).

=== Symptom: The printed barcode looks "strange" and is not readable ===

<u>**Reason**</u>**: The quality of the bars is beyond tolerance.**

The **sharpness of the edges** of the bars is fuzzy.
If this always occurs only on one particular edge of the bars, switching the feed direction (SEF vs LEF) may help.

The bars/rectangles **appear too black** (too thick/wide).
Try to decrease the blackness by using a different Toner Saving mode.

The bars/rectangles **appear too grey**.
Try to increase the blackness by using a different Toner Saving mode.
Check the toner quality at other/normal text.

=== Symptom: The printed barcode looks OK but is not readable ===

The barcode reader does not beep.

<u>**Reason**</u>**: The barcode reader is not suitable for the N:W ratio or density/width used.**
Try using a different barcode reader.

<u>**Reason**</u>**: The barcode reader is not configured properly.**
Confirm that the required check method for that symbology is enabled (VERIFY CHK ON/OFF).
Confirm that the minimum/maximum length setting is not in conflict with the data.
In the handbook of the barcode reader, check for other related settings that may be configured wrongly.

<u>**Reason**</u>**: The logical barcode pattern is invalid.**
(Profound check): Decode the barcode manually to check if it is a legal sequence of that symbology. Contact support.

<u>**Reason**</u>**: The start or stop or other control characters may be missing.**
Confirm that the correct start/stop pattern is present.

<u>**Reason**</u>**: The check character is invalid.**
The calculation may be wrong, or a wrong font character is being used.
Perform a manual calculation of the checksum.
Contact support.

<u>**Reason**</u>**: Some other barcode character is wrong.**
A wrong font may have been selected, or the font has become corrupted.

<u>**Reason**</u>**: The command sequence is wrong.**
There may be something wrong with the command sequence.
Confirm that the command sequence arriving at the printer is correct. (Profound check): Perform the code construction manually.

<u>**Reason**</u>**: The surrounding quiet zone is too small.**
Leave enough empty space around the barcode according to the specification.

<u>**Reason**</u>**: The physical widths of bars or spaces are beyond tolerance of the symbology spec's.**
The bars and/or spaces may be too wide or too small.
The density may be too high (X too small).
The N:W ratio may be beyond tolerance.
The total width may be too large.
The inter-character spacing may be beyond tolerance.

<u>**Reason**</u>**: The physical widths of bars or spaces are beyond tolerance of the reader's spec's.**
Use a suitable barcode reader.

=== Symptom: The printed barcode is read but with a wrong result ===

The barcode reader beeps, but the result displayed does not correspond to the expected/original message data.

<u>**Reason**</u>**: The barcode reader is not configured correctly for that symbology.**

Some parameter of that symbology has not been activated or configured correctly.
Confirm if for this symbology "Extended" ("Full ASCII") should be enabled on the barcode reader.
A wrong output character set is selected.
  (E.g. in the case of a keyboard wedge, the keyboard language setting has to be configured correctly on the barcode reader.)
If the CHK is not displayed: Check the "Transmit CHK" setting.
If the 2/5-digit supplement for EAN/UPC is missing:

Reason: The logical barcode pattern encodes wrong data.
Decode the barcode manually and check which message data it actually encodes.
This can be due to one of the two following reasons.

Reason: Some barcode character is wrong.
A wrong font may have been selected, or the font has become corrupted.

Reason: The command sequence is wrong.
There may be something wrong with the command sequence.
Confirm that the command sequence arriving at the printer is correct.
Perform the code construction manually.
Contact support.

=== Symptom: The printed barcode is read correctly, but not always ===

Reason: The barcode measurements are slightly beyond tolerance.
Confirm that the barcode measurements (quiet zone, density, N:W ratio, bar:space width ratio ~ 1:1, etc) are within the tolerance of the symbology specification and of the barcode reader.

Reason: The barcode quality is slightly beyond tolerance.
Confirm that the barcode quality (blackness, contrast, sharpness of edges, etc) are within the tolerance of the symbology specification and of the barcode reader.

Reason: The barcode reader is critical to the barcode measurements or quality.
Confirm that the barcode measurements and quality are within the tolerance of the barcode reader.
Try using a different barcode reader.

=== Symptom: The printed barcode is always read correctly, but still not appearing as intended ===

Refer to the corresponding one of the following symptoms.

=== Symptom: The barcode size is wrong ===

Reason: A wrong size parameter was specified (for the font).
See the main procedure for information on how to configure the correct size.

=== Symptom: The barcode position is wrong ===

Reason: A wrong positioning command was specified.
See the main procedure for information on how to achieve the correct position.

=== Symptom: The clear text appears wrong ===

Reason: A wrong CTX parameter was specified.
Confirm that the CTX parameters are as desired.

Reason: The wrong barcode font was selected.
Check that the proper font for the desired CTX style was selected.

Reason: The character code for a clear text character is wrong.
Confirm that the character is chosen from the correct pool.

=== Symptom: Everything is fine ===

4.12.2 [SUPPORT ONLY]

4.12.3 Contacting support

If you cannot solve the problem by yourself, contact support.
To contact support, the following information has to be provided.

- The symptom/problem observed.

-   The actual barcode image.
    To scan in a barcode image with a digital scanner (fax, copier), it should be placed diagonally(!) on the scanner's platen glass, to avoid that the bars are parallel or perpendicular to the scan direction / paper edges. Otherwise the rounding applied by the scanner would distort the absolute thicknesses of bars and spaces, thus arriving at wrong relative widths/ratios, which would make it unusable for problem analysis.
-   The desired/intended barcode parameters (symbology, CHK method, CTX style, N:W ratio, etc).
-   The message data.
-   The platform used and relevant settings (see platform-specific documentation).
-   The solution used and relevant settings (see solution-specific documentation).
-   Any of your findings and steps you already attempted yourself to isolate the problem.
-   Captured data, as being requested by support staff. (See next section below.)

Capturing data

If necessary, the command sequence may need to be analyzed.
This will be explicitly requested by support staff.

To capture the final printer language data stream, as it reaches the printer, follow the procedure given by support staff.

To capture the data stream at any intermediate stage, follow the procedure given by support staff.

5   The Barcode Symbologies and Applications

This chapter contains one section of information for each **symbology** or **application**.
A symbology / application **family** summarizes the specifications of related symbologies /applications.

Symbologies

A **Symbology** defines a character set, some checksum methods, and a logical encoding scheme.

The **Message character set** defines the **alphabet** of the encodable **message data**.
Besides the actual **message data characters**, there are some (message) **control characters** which can also be part of the message (e.g. <FNCx>, <Shift>, (+), (%), etc).

Reference number

A **Reference number** is assigned to each message character, which is used in calculating checksums.
In numeric character sets, the reference number of a digit is usually equal to its value.

Check characters

One or more **Check characters** (**CHK**) may be required to be appended to the data to reduce read errors.
Since this is not necessarily always a digit, the term **check digit** should only be used for numeric symbologies.
The CHK is/are usually appended to the end of the message, before the [stop] character.
There may be one or more different **checksum methods** for how to calculate them.
A checksum is usually calculated by some formula applied to the message data. In most cases, it is calculated as a weighted sum of the reference numbers of all message data characters, reduced modulo the size of the alphabet. The check character is then the character (from the message alphabet) whose reference number equals the checksum.

CHK formula

The algorithm can be specified concisely in a **CHK formula**, for which the following syntax is used:
Example: #CHK = + [...,*3,*2] x #M mod 47
The "CHK" is the resulting check character.
  The "#CHK" denotes the checksum, i.e. the reference number of the check character.
The "M" is the message data, viewed as a sequence of message characters.
  The "#M" denotes the sequence of the reference numbers of the message characters.
A "*7" specifies a single weight factor, e.g. 7.
The "[ ]" is to group the weight factors.
An "..." inside the "[ ]" defines the entire sequence of weight factors as a logical continuation of the partial sequence given, to the left or to the right. E.g.
  - [*7, ..., *3, *2]   ==> [*7 *6 *5 *4 *3 *2] -- finite
  - [*1, *2, *3, ...]   ==> [*1 *2 *3 *4 *5 *6 *7 *8 *9 *10 ...] -- infinite, starting from the left end
  - [..., *3, *2, *1]   ==> [... *10 *9 *8 *7 *6 *5 *4 *3 *2 *1] -- infinite, starting from the right end
An "..." outside the "[ ]" indicates an (alternating) infinite repetition of the (finite) sequence inside the "[ ]", to the left or to the right. E.g.
  - ... [*3, *9]   ==> ... *3 *9 *3 *9 *3 *9 *3 *9 *3 *9 *3 *9 *3 *9 *3 *9
  - ... [*7, ..., *3, *2] ==> ... *7 *6 *5 *4 *3 *2 *7 *6 *5 *4 *3 *2 *7 *6 *5 *4 *3 *2
The "x #M" indicates component-wise multiplication of each message character reference number with the corresponding element of the weight factor sequence, and subsequent summing of all products. E.g. [..., *3, *2, *1] x "10 17 9 8 5" = 5*10 + 4*17 + 3*9 + 2*8 + 1*5.
The "mod N" means reduction of the sum "modulo N", i.e. taking the remainder upon division by N, thus mapping it into the range {0, ..., N-1}.
The "+/-" indicates the sign.
  A "+" means that the remainder has to be taken as is, i.e. the difference to the next lower multiple of N.
  A "-" means that instead the difference to the next higher mutliple of N has to be taken.

Encoding scheme

The **Encoding scheme** defines a mapping of the message data M = "<c#1>...<c#n>" into eventually a barcode image.
This scheme is multi-tiered; the stages and conversion steps are explained below.
Note: The notation "<c#i>" is used for characters in general; in the case of purely numeric digits, the notation "<d#i>" is preferred.

Symbology character set

The **Symbology character set** is a set of **Symbology characters**, which are an abstract representation of portions of a barcode, that essentially correspond to the message characters ([A], [1], etc) or pairs thereof ([63], etc), plus some extra symbology-specific (symbology) **control characters** ([startX], [stop], etc). A symbology character can usually be identified by the name or the reference number of the message character (or pair thereof) which it represents.

Logical encoding

Each symbology character has a **Logical encoding**, which can be given in two equivalent representations:
The **Binary representation** is a binary number (e.g. 110001110101), in which a "1" represents a bar, and a "0" represents a space. A sequence of w consecutive "11...1" or "00...0" represent bars or spaces of corresponding width w. This binary number is usually somehow related to the binary representation of the reference number of the corresponding character.
In the **Width representation** (e.g. 2331111) a digit "i" corresponds to an element (bar or space) consisting of i modules (i = 1, 2, 3, 4); the digits are alternating for bars and spaces. This representation is more suitable for directly realizing the barcode image.
Please note that the logical barcode encoding does not specify any physical measurements.

Note:
Depending on the Inter-character spacing, the pattern in the encoding scheme looks different:

		Binary representation	Width representation
- Discrete		"1 ... 1"	"b s ... s b"
- Continuous		"1 ... 0" (right-hand)	"b s ... b s"
	or	"0 ... 1" (left-hand)	"s b ... s b"

Imaging / Physical realization

The **Imaging / Physical realization** of this encoding can be implemented as a mapping
- to a series of rectangle drawing commands in some printer language, or
- to the ASCII code of a barcode font character.
This is called the **Physical encoding**.

Physical attributes

The physical encoding is subject to constraints given by **Physical attributes**, such as the specification and tolerances of minimum and maximum measurements of bars and spaces (bar height, overall width, module width X, density, N:W ratio, inter-character spaces, etc) and of the quiet zones, as well as conventions about the clear text style (placement, format, font, etc), and other conventions (bearer bars, etc).

Applications

An **Application** maps its message data to the message character set of some existing **base symbology**.
Usually there is some **semantical interpretation** of the data.
It specifies which of the CHK methods of the base symbology has to be applied, if any.
In addition or instead, it may have its own CHK method.
And it may have its own conventions about the physical barcode layout and clear text style.

Code construction procedure

The **Code construction procedure** defines how the original message data M = "<c#1> ... <c#n>" is step-wise converted into a series of symbology characters (including start/stop and possible check characters), e.g. "[start] [c#1] ... [c#n] [CHK] [stop]", and eventually the barcode image. This is a multi-step procedure.

The following table shows the different **Stages during the Code construction procedure**.

Stage	Level name	Data	Alphabet
- (-I)	**Application**	original (application) message data	Application message character set
- (0)	**Message**	(original) (symbology) message data	(Symbology) Message character set
- (I)	**Symbology** / Logical	Symbology characters with CHK and [start/stop]	Symbology character set
- (II)	**Physical** / Realization	- sequence of barcode font characters	- Font character set (ASCII codes)
		- PDL rectangle draw commands	- PDL syntax
- (III)	**Image**	black rectangles on paper	--

The following table shows the different **Steps of the Code construction procedure**.

Step	Stage	Step name	Modifications applied	Components involved
- (0)	(-I) -> (0)	Application level	Convert the original application message data into the symbology message alphabet. This may include: - calculate and append application-specific CHK - insert application control characters: <FNC1>, <StartX>, etc.	Encoding (-1) (application-specific) CHK algorithm
- (I)	(0) -> (I)	Symbology / Logical level	Convert the original (symbology) message data into a sequence of symbology characters. This may include: - alphabet **Extending**: ($), (%), (/), (+), or $, %, /, + e.g. Escaping: a -> (+)A - determine [startX], [CodeX], [Shift] (e.g. Code 128 Autoswitch) - calculate and append symbology-specific CHK - plain 1:1 mapping: <c> |--> [<c>] - **Interleaving/Pairing**: <d> <d> |--> [<d><d>] - insert symbology control characters: [start], [stop], [guard], etc.	Encoding (0) Reference numbers and CHK algorithm
- (II)	(I) -> (II)	Physical / Realization level	Map each symbology character into a physical realization of the black-and-white pattern, using one of these realizations: - convert to barcode font character - convert to PCL rectangle commands	Encoding (II) (Encoding (I))
- (III)	(II) -> (III)	Imaging / Printing level	Convert - font character --> glyph bitmap - rectangle command --> rectangle	Printer controller / PCL Interpreter

The following **notation scheme** is used (<c#i> is the message character at position i):

[startX] [<c#1>] ... [<c#n>] [<CHK>] [stopY]

If different pools are involved, the following scheme may be used instead:

byte #	1	2	3	4	5	6	7	8	9	10	11
symb. char.	[start]	[<c#1>]	[<c#2>]	[<c#3>]	[<c#4>]	[guard]	[<c#5>]	[<c#6>]	[<c#7>]	[<CHK>]	[stop]
pool		A	A	A	A		C	C	C	C	

The following **encoding tables** are used in the code construction procedure.

Encoding (-I): (Application level)

This describes the conversion of
  <application message data> --> **<symbology message data>**.

 In the easiest case, it can be described as a 1:1 mapping
  <application message character> --> <symbology message character>

The technique of **escaping** can be described as a mapping
  <application char> |--> **<escape char> <letter>**
This is used for **extending** the alphabet.

Encoding (0): (Message level)

Such mapping describes the conversion of
  <message data> |--> <symbology data>.

In the easiest case, in can be described as a 1:1 mapping
  <message character> |--> <symbology character> = **[<message character>]**

In the case of **interleaving/pairing**, it can be described as a mapping
  <message character> <message character> |--> <symbology character> = **[<message character> <message character>]**

A message character may also be mapped to different sets of symbology characters, depending on certain context conditions.
These sets are called **pools**.
This is denoted as
  <message character> |--> <symbology character> = **[<message character>]:<pool>**
Example
The EAN/UPC family defines pools A,B,C.
Depending on the value of the 1st digit and/or of the check digit, and depending on the position of a message digit, e.g. "3", within the message,
it would be mapped into one of the symbology characters [3]:A or [3]:B or [3]:C.

Encoding (CHK): (Reference number)

Such table describes the mapping
  <message character> |--> <reference number>

Encoding (I): (Logical level)

Such table describes the mapping
  <symbology character> |--> <logical barcode encoding>

Encoding (II): (Physical level)

Such table describes a physical realization of the logical width representation.
This is not part of the symbology specification !
Depending on the solution/approach, a symbology character may be mapped into
- the ASCII code of a barcode font character which images the black rectangles, i.e.
  <symbology character> |--> <font character ASCII code>
- a series of PCL commands which draw black rectangles of the given width, i.e.
  <symbology character> |--> <PCL code>

This information is described in a separate chapter.

Encoding (III): (Image level)

For the mapping of a <symbology character> to the actual **barcode image**, please refer to the Test sheet printout of a corresponding barcode font.

Chapter structure

For each symbology or application or family thereof, the information is given by the following structure.

The heading contains the main name which is used throughout this documentation.

The **Overview** section gives an overview of the symbology, its **typical usage**, its characteristics and its particularities. It also lists **alternative names**, and lists **special applications** of this symbology. It includes a **sample barcode** and refers to the **specification** and other **sources of information**.

The **Character set** section gives a brief description of the **message character set** and the **symbology character set**.
It also lists other **logical attributes**, e.g. possible message **length restrictions** (min/max, parity odd/even), and **other restrictions** related to the message.
In the case of a symbology, this also explains the function of the control characters (e.g. <Shift>, <FNCx>, <(%)>, etc).
In the case of an application, this also includes the **semantical interpretation** of the message characters, especially at particular positions (e.g. "RA" + <8 digits> + <internal CHK> + "FR").

Next, the **Reference numbers** are defined for all message characters. They are denoted as "#...".

Then all possible (and supported) **Checksum methods** are presented. Check characters may be mandatory or optional, and their required number may differ. For each method the algorithm is specified verbosely and as a concise formula, showing the weight factors and the "modulo". Besides the abstract definition, an evidentiary example is given.
Note: The message characters to which the algorithm is being applied are numbered as <c#1> ... <c#n>. The terms "odd-positioned" and "even-positioned" refer to these indices.

The **Encoding scheme** section summarizes the principles of the encoding. First, it contains information such as the **dimensionality** (1-D/2-D), and the degree of information contained in the width and the height of the characters' bars and spaces and the inter-character spaces. Next it contains information about the meaning of "x of y", Interleaving, the number of modules per element (narrow/wide, or multi-level), the number of elements per character, and the number of modules per character. Last, it states whether the symbology is **self-checking** (i.e. without an explicit check character, through internal redundancy e.g. a parity bit), and whether it is **bi-directionally** readable.

Then the **Encoding table** defines the actual mapping between the symbology characters into the logical pattern of bars and spaces, both in the **binary representation** and in the **width representation**.

Follow the **Physical attributes**, as enumerated above.

The **Code construction** section defines all steps how the message data is converted into symbology data.
Essentially this includes appending the CHK and adding [start] and [stop] and possibly other symbology control characters.

The chapter ends with a full **example**, showing all stages of the code construction, starting with the message data, and ending with the resulting barcode image.

Notation

The following barcode-specific acronyms and notational conventions are used.

Acronyms
 - **ch**   = character
 - **el**   = element
 - **b**   = bar
 - **s**   = space
 - **N**   = narrow
 - **W**   = wide
 - **e<i> / b<i> / s<i>**   = element / bar / space of width <i> (where <i> = 1,2,3,4, N,W)
                             e.g. - b1,bN = narrow bar, sN = narrow space, bW = wide bar, sW = wide space
 - **md**   = module
 - **mb**   = bar module (= a module belonging to a bar)
 - **ms**   = space module (= a module belonging to a space)
 - **#<...>**   = number of <...> per character

Note:
For the number of **elements per character**, the following equations hold:
el = #b + #s =
        = (#e1 + #e2 +#e3 + #e4) or (#eN + #eW)
        = (#b1 + #b2 + #b3 + #b4) + (#s1 + #s2 + #s3 + #s4) or (#bN + #bW) + (#sN + #sW)

Note:
For the number of **modules per character**, the following equations hold:
#mb   = (1*#b1 + 2*#b2 + 3*#b3 + 4*#b4) or (1*#bN + 2*#bW)
#ms   = (1*#s1 + 2*#s2 + 3*#s3 + 4*#s4) or (1*#sN + 2*#sW)
#md = #mb + #ms =
        = (1*#e1 + 2*#e2 + 3*#e3 + 4*#e4) or (1*#eN + 2*#eW)

For symbology characters the following notation is used:

  [<character name>]   = symbology character (by name)

E.g. **[A]** indicates the symbology character for message character "A". This different notation is necessary to avoid any confusion with the barcode font characters used; they may not necessarily be coded as font character "A".

 [# <reference number>]   = the symbology character for the message character (or pair thereof) with the reference number indicated

E.g. **[#19]** indicates the symbology character for a message character with reference number 19. This different notation is necessary to avoid confusion in the case of multiple codes for a symbology, such as Code 128.

For <u>message characters</u> the notation "..." (i.e. with double quotes) is used.

If the meaning of a symbol is clear from the context, the [...] or <...> or "..." may be omitted to save space.

5.1 Symbology: **Code 128**

5.1.1 Overview

This symbology is known as:
- Code 128

Code 128 actually defines 3 codes:
- **Code A**:   defines all characters except "a" to "z" from the ASCII table, including the control characters <00>..<1F>.
- **Code B**:   same as Code A, but the control characters <00>..<1F> are replaced by the lower case characters "a".."z".
- **Code C**:   represents all digit pairs from "00" to "99"
In addition, there are some control characters defined for switching between these codes.

Usage
Application:	many applications: industrial, video rental, etc.
Regional use:	world-wide
Popularity:	widely used

Remarks
Characteristics: (+) very high density; (+) high reliability;

Example
This is a sample of a typical barcode of this symbology.

Further information

The specification of this symbology is maintained at:
- USA: ANSI/AIM BC4-1995 (ISS)
- Europe: EN 799

Other sources of information can be found at Appendix L.

Applications

Applications based on this symbology are:
- UCC/EAN-128 (*)
- UPC-128 (*)
- UPS-128
Those marked (*) are directly supported by this product; they are explained in separate sections right after this symbology.

5.1.2 Character set (Alphabet)

Message character set

Message data characters

The message data character set consists of **128** characters (Full ASCII).

The full ASCII support is achieved by defining 3 character sets (encoded in 3 different modes) and switching among them:

- **Code A =**   - uppercase alphanumeric <20>..<5F> = {0..9, A..Z} plus some special characters, plus
  - ASCII control chars <00>..<1F> = {NUL,SOH,ETX,BEL,BS,...,US}
- **Code B =**   - uppercase alphanumeric <20>..<5F> = {0..9, A..Z} plus some special characters, plus
  - all lowercase chars <60>..<7F> = {a..z} plus some special characters
- **Code C =**   - double density numeric, all digit pairs {"00", ..., "99"}

A character is said to be in "**mode X**", if it is interpreted according to Code X.

Message control characters

The message control character set consists of **8** characters, 4 to switch between modes, and 4 "Function codes".

Control character	Function
<CodeA>	switch to Mode A; i.e. characters following this will be interpreted according to Code A
<CodeB>	switch to Mode B; i.e. characters following this will be interpreted according to Code B
<CodeC>	switch to Mode C; i.e. characters following this will be interpreted according to Code C
<Shift>	toggle Mode A/B for next character only
<FNC1>	Application-specific interpretation: - [startX] [FNC1] ... -- start sequence reserved for UCC/EAN-128 applications - <letter> <FNC1> -- "Application Indicator" - <digit><digit> <FNC1> -- dto
<FNC2>	concatenate / multi-read (concatenate instructions to barcode reader)
<FNC3>	reset barcode reader
<FNC4>	Alphabet extension: (upper ASCII half: <128>..<255>) - <FNC4> <char> -- shift next character only to Extended (+128) - <FNC4> <FNC4> -- toggle Extended / Non-extended mode (affects all following characters)

Note: (Specifying the control characters in the message data stream)
These control characters are usually generated and inserted automatically in the data stream, but they may also be explicitly specified as part of the message, e.g. to force a certain mode.
Since they do not have any ASCII equivalent, they need to mapped to some arbitrary ASCII codes in the input message data stream.
A "de facto" standard, as employed e.g. by JetCAPS, is:
- <80>   <128>   <Shift>
- <81>   <129>   <FNC1>
- <82>   <130>   <FNC2>
- <83>   <131>   <FNC3>
- <84>   <132>   <FNC4>
- <85>   <133>   <CodeA>
- <86>   <134>   <CodeB>
- <87>   <135>   <CodeC>
This mapping may, however, not be appropriate for some environments. Refer to separate documentation for deviations from this scheme.

Message length restrictions
The message length is variable and not limited.
Code C requires an even number of digits, since they are encoded in pairs. Otherwise the data may be padded with a leading '0'.

Symbology character set

The symbology character set consists of **106** = **102** (message) **+ 4** (control) characters.

Symbology data characters
There is a symbology character for every message (data or control) character (Code A,B) or pair thereof (Code C). See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[startA]	start of barcode in Mode A
[startB]	start of barcode in Mode B
[startC]	start of barcode in Mode C
[stop]	stop of barcode

The switching among the codes makes it possible to encode all ASCII characters in one single barcode. Moreover, clever switching to Code C for at least 2 pairs of digits can minimize the length of the barcode.

The term "**Code 128 autoswitch**" is used if the solution determines automatically when to use which code and inserts the necessary [CODEx] and [Shift] characters.

5.1.3 Reference numbers

This table assigns a unique reference number to each message character.

Ref. #	A	B	C	Ref. #	A	B	C	Ref. #	A	B	C
#0	SP	SP	00	#36	D	D	36	#72	BS	h	72
#1	!	!	01	#37	E	E	37	#73	HT	i	73
#2	„	„	02	#38	F	F	38	#74	LF	j	74
#3	#	#	03	#39	G	G	39	#75	VT	k	75
#4	$	$	04	#40	H	H	40	#76	FF	l	76
#5	%	%	05	#41	I	I	41	#77	CR	m	77
#6	&	&	06	#42	J	J	42	#78	SO	n	78
#7	'	'	07	#43	K	K	43	#79	SI	o	79
#8	(	(	08	#44	L	L	44	#80	DLE	p	80
#9	)	)	09	#45	M	M	45	#81	DC1	q	81
#10	*	*	10	#46	N	N	46	#82	DC2	r	82
#11	+	+	11	#47	O	O	47	#83	DC3	s	83
#12	,	,	12	#48	P	P	48	#84	DC4	t	84
#13	-	-	13	#49	Q	Q	49	#85	NAK	u	85
#14	.	.	14	#50	R	R	50	#86	SYN	v	86
#15	/	/	15	#51	S	S	51	#87	ETB	w	87
#16	0	0	16	#52	T	T	52	#88	CAN	x	88
#17	1	1	17	#53	U	U	53	#89	EM	y	89
#18	2	2	18	#54	V	V	54	#90	SUB	z	90
#19	3	3	19	#55	W	W	55	#91	ESC	{	91
#20	4	4	20	#56	X	X	56	#92	FS	|	92
#21	5	5	21	#57	Y	Y	57	#93	GS	}	93
#22	6	6	22	#58	Z	Z	58	#94	RS	~	94
#23	7	7	23	#59	[	[	59	#95	US	DEL	95
#24	8	8	24	#60	\	\	60	#96	<FNC3>	<FNC3>	96
#25	9	9	25	#61	]	]	61	#97	<FNC2>	<FNC2>	97
#26	:	:	26	#62	^	^	62	#98	<Shift>	<Shift>	98
#27	;	;	27	#63	_	_	63	#99	<CodeC>	<CodeC>	99
#28	<	<	28	#64	NUL	`	64	#100	<CodeB>	<FNC4>	<CodeB>
#29	=	=	29	#65	SOH	a	65	#101	<FNC4>	<CodeA>	<CodeA>
#30	>	>	30	#66	STX	b	66	#102	<FNC1>	<FNC1>	<FNC1>
#31	?	?	31	#67	ETX	c	67				
#32	@	@	32	#68	EOT	d	68	#103	[startA]	[startA]	[startA]
#33	A	A	33	#69	ENQ	e	69	#104	[startB]	[startB]	[startB]
#34	B	B	34	#70	ACK	f	70	#105	[startC]	[startC]	[startC]
#35	C	C	35	#71	BEL	g	71	--	[stop]	[stop]	[stop]

5.1.4 Checksum methods

Code 128 defines 1 mandatory check character (mod 103), weighted by position (1,2,3,... from left to right).
The start character is also taken into account.

CHK Formula

1 mandatory CHK

#CHK = + [*1, *1, *2, *3, ...] x #M mod 103

CHK Algorithm

The checksum is the **modulo 103** remainder of the sum of all reference values multiplied with their position.

Example (CHK calculation)

M = "C O D E <SP> 1 2 8"
-->

Message character	<startB>	C	O	D	E	<SP>	1	2	8		
Reference number	104	35	47	36	37	0	17	18	24		

Weight factor	* 1	* 1	* 2	* 3	* 4	* 5	* 6	* 7	* 8		
Product -> Sum	104	+ 35	+ 94	+ 108	+ 148	+ 0	+ 102	+ 126	+ 192		$= 909 = 8x\ \underline{103} + \mathbf{85}$

-->
CHK = #85 = "u" (in mode B)

5.1.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces)
Dimensionality:	1-D
(Continuous / Discrete):	Continuous (inter-character space contains information)
Logical encoding type:	
# Modules per element:	4-level = 1, 2, 3, 4 md
# Elements per character:	**6 el** = 3 b + 3 s
# Modules per character:	**11 md** = (various) - # bar modules (= 1x b1 + 2x b2 + 3x b3 + 4x b4) = even - # space modules (= 1x s1 + 2x s2 + 3x s3 + 4x s4) = odd
Self-checking:	yes
Bi-directional:	yes

5.1.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.1.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.1.8 Code construction: Steps

The message data M = "<c#1>...<c#n>" is converted into the symbology character set according to the following scheme.

> **[startX] [<c#1>] ... [<c#n>] [<CHK>] [stop]**

1. Convert the message characters into symbology characters

There are 4 different algorithms how to do the conversion.

Code 128A:
Allows only Code A message characters.
Convert each message character <c#i> into the symbology character [<c#i>].
Prepend [startA].

Code 128B:
Allows only Code B message characters.
Convert each message character <c#i> into the symbology character [<c#i>].
Prepend [startB].

Code 128C:

Allows only Code C message characters.
Convert each pair of message digits "<d#i><d#i+1>" into the single symbology character [<d#i><d#i+1>].
Prepend [startC].

Code 128 Autoswitch:
Apply the following **"auto-switch" algorithm**.
===

General rules:
- Whenever possible, use Code B.
- Stay inside a mode as long as possible. Switch only if necessary, as indicated below:
- When 4 consecutive digits or more => use Code C
- When <00> .. <1F> => requires Code A
- When <60> .. <7F> => requires Code B

More explicitly:

Condition	Reaction
Initially: Determine start mode:	
- if "<digit><digit><any>"	=> send [startC] and switch to mode C
- if "<00..1F><any>"	=> send [startA] and switch to mode A
- else	=> send [startB] and switch to mode B
When in mode A:	
- if "<digit><digit><digit><digit><any>"	=> send [CodeC] and switch to mode C
- if "<60..7F><60..7F><any>"	=> send [CodeB] and switch to mode B
- if "<60..7F><not 60..7F><any>"	=> send [Shift] and stay in mode A
- else	=> stay in mode A
When in mode B:	
- if "<digit><digit><digit><digit><any>"	=> send [CodeC] and switch to mode C
- if "<00..1F><00..1F><any>"	=> send [CodeA] and switch to mode A
- if "<00..1F><not 00..1F><any>"	=> send [Shift] and stay in mode B
- else	=> stay in mode B
When in mode C:	
- if "<00..1F><any>" OR "<digit><00..1F><any>"	=> send [CodeA] and switch to mode A
- if "<not digit><any>" OR "<digit><not digit><any>"	=> send [CodeB] and switch to mode B
- else	=> stay in mode C

In mode A and B, code the characters individually.
In mode C, code the digits in pairs.

===

2. Apply check characters

Calculate the CHK according to the algorithm above and append it to the message.

3. Insert symbology control characters

Append the [stop] character.

5.1.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.1.10 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		C O D E <SP> 1 2 8
		-- (*)
		<startB> C O D E <SP> 1 2 8

		`-- (*)`
		`<startB> C O D E <SP> 1 2 8 `**`u`**
		`[startB] [C] [O] [D] [E] [SP] [1] [2] [8] [u] [stop]`
		CODE 128

(*) For the calculation of the CHK see above.

5.2 Application: UCC/EAN-128

5.2.1 Overview

This application is known under these names:
- EAN/UCC-128
- EAN-128
- UCC-128
- USS-128
This is not to be confused with UPC-128 or with UPS-128.

UCC/EAN-128 is a special application of Code 128. It defines data formats for commerce.

Usage
Application:	palette labels, shipping containers, not for retail checkout
Regional use:	world-wide
Popularity:	very common

Example
This is a sample of a typical barcode of this application.

Further information

The specification of this application is maintained at:
- USA / North America: Uniform Code Council (UCC), Uniform Symbology Specification (USS)
- Europe: EN 799

Other sources of information can be found at Appendix L.

Base symbology

Code 128

5.2.2 Character set

Message character set

Message data characters
The message data character set is the same as for Code 128.

Message control characters

The <FNC1> control character has a special meaning here.
The barcode start sequence "[startX] [FNC1] ..." is reserved to be used exclusively with UCC/EAN-128.
Moreover, <FNC1> is used to delimit a value of variable length. See at "Semantics" below.

Note: The encoding of the <FNC1> character as an application message character, if necessary, depends on the definition/convention of the solution.

Message length restrictions
The message length is variable. It depends on the AI.
The maximum number of message characters is limited to 48.
Taking into account the interleaving, there is another restriction to a maximum of 35 barcode characters, including the start, stop and control characters.

Other message data restrictions
None.

Semantics (Interpretation of message characters)

An arbitrary UCC/EAN-128 code is a sequence
  [startX] <FNC1><AI_1><data_1> [<FNC1>] <AI_2><data_2> ... [<FNC1>] <AI_N><data_N> ...
where <data_i> is a value specific for the "Application Identifier" <AI_i>, typically an article number, price, weight, date, etc.
That is, one code can contain multiple <AI>.
Subsequent <AI> need to be preceded by a <FNC1> only if the previous value is of variable length.
Each <AI> can be 2..4 digits long. Some valid <AI> are defined in the table below.

Application Identifiers <AI> (excerpt)

AI	Application	# char.	Values
00	SSCC-18 (*1)	(18)	numeric
01	SSCC-14	(14)	numeric
11	Production date	(6)	(yymmdd)
13	Packaging date	(6)	(yymmdd)
15	Sell by date	(6)	(yymmdd)
17	Expiration date	(6)	(yymmdd)
21 250	Serial #	(2) (variable)	numeric
22	HIBC ()	(variable)	alphanumeric
31xy	Measures (net) (Weight, Length, Width, Depth, Area, Volume, etc)	(6)	numeric
33xy	Measures (gross) (dto)	(6)	numeric
37	# units contained	(variable)	numeric
400	Order number	(variable)	alphanumeric
420	Domestic postal routing	(variable)	alphanumeric
421	International postal routing	(variable)	alphanumeric
9x	(company-specific)	(max. 30)	alphanumeric

Note: (*1)
SSCC-18 requires the last digit to be a proprietary CHK. It is also known as "UPC-128". Please see the chapter "UPC-128" below.

5.2.3 Encoding (0): (Application level)

There is no specific conversion necessary at this stage.

5.2.4 Checksum methods

There is no application-specific check character.
The (mandatory) checksum method of the base symbology is being applied.

5.2.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Width (overall)	= max. 165 mm (incl. quiet zone)

Imaging conventions

In the clear text, each Application identifier <AI> (behind the [FNC1]) is shown in parentheses, e.g. ...(11)021231(17)031231... .

5.2.6 Code construction: Steps

The application message data M = "<c#1> ... <c#n>" is converted to the symbology message character set as follows.

> M(I) = **<FNC1> <c#1> ... <c#n>**

1. Insert application control characters

Insert the <FNC1> control character in front of the message.

2. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the Code 128 symbology.

For the CHK calculation, note that the message starts with <FNC1> at position 1. The first original message character has therefore to be multiplied with weight factor 2, etc.
Which <startX> character to choose, depends on the data.

5.2.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		Production date = 24 December 2001; Packaging date = 10 January 2002;
		(11) 011224 **(13)** 020110
		<startC> <FNC1> 11 01 12 24 13 02 01 10
		(*)
		[startC] [FNC1] [11] [01] [12] [24] [13] [02] [01] [10] **[#75] [stop]**
		(11) 011224 (13) 020110

(*) Checksum calculation:

Message character	<startC>	<FNC1>	11	01	12	24	13	02	01	10	
Reference number	105	102	11	01	12	24	13	02	01	10	
Weight factor	* 1	* 1	* 2	* 3	* 4	* 5	* 6	* 7	* 8	* 9	
Product -> Sum	105	+ 102	+ 22	+ 3	+ 48	+ 120	+ 78	+ 14	+ 8	+ 90	= 590 = 5x 103 + **75**

5.3 Application: UPC-128

5.3.1 Overview

This application is known under these names:
- UPC-128
- UPC-128 Shipping Container Code
- Code 128 UPC Shipping Container Code
- SSCC-18
- UCC-128 (!)

This is a special version of UCC/EAN-128, with AI = 00.

Usage
It is used to identify shipping containers by a serial number.

Example
This is a sample of a typical barcode of this application.

Further information

The specification of this application is maintained at UPC.

Other sources of information can be found at Appendix L.

Base symbology

Code 128 (Code 128 C)

5.3.2 Character set

Message character set

The message character set consists of **10** characters.

Message data characters
Numeric only = {0,...,9}.

Message control characters
None.

Message length restrictions
The message length is fixed to = **19 digits** + 1 CHK.

Other message data restrictions
The first two digits must be "00".

Semantics (Interpretation of message characters)

An arbitrary UPC-128 message has a fixed length of 2+17+1 digits, with this structure:
  0 0 <data> <CHK>

The "00" is the Application Identifier which identifies the data as SSCC-18.
The <data> is a fixed length data structure of 17 digits. This is a unique number consisting of these parts:
 - digit 3   (1)   Extension digit
 - digits 4..10   (7)   UCC/EAN Company Prefix
 - digits 11..19   (9)   Serial Reference Number

The **Extension digit** has no predetermined logic; it is freely available to the member company (a member of the national EAN or UCC organization).

The EAN/UPC **Company Prefix** identifies the shipping company. It is assigned by the UCC or EAN to a member company. (In Germany this is called "BBN" ("Bundeseinheitliche Betriebsnummer").)
The **Serial Reference Number** is assigned by the member company to identify an individual shipping container.

The <CHK> is the application-specific mod 10 check digit.

5.3.3 Encoding (0): (Application level)

There is no specific conversion necessary at this stage.

5.3.4 Checksum methods

There is 1 mandatory application-specific check digit mod 10, with alternating weights *3 and *1.
It has to be applied before and in addition to the Code 128 mod 103 CHK algorithm.

CHK Formula

1 mandatory CHK

> **#CHK = - [*3, ..., *1,*3] x #M mod 10**
>   = - (3*<c#3>+ 1*<c#4>+ ... + 1*<c#18> + 3*<c#19>) mod 10
>   = - [3* (<c#3>+ <c#5>+ ... + <c#19>) + 1* (<c#4> + <c#6> + ... + <c#18>)] mod 10

CHK Algorithm
This special mod 10 algorithm is only applied to the 19 message digits.
1. Sum up all odd-positioned digits. Multiply the sum by 3.
2. Add the remaining (even-positioned) digits.
3. The check digit is the difference to the next multiple of 10.

Example (CHK calculation)

M = "0 0 9 0 0 0 9 0 0 9 1 2 3 4 5 6 7 8 9"
-->
CHK = - [3* (0+9+0+9+0+1+3+5+7+9) + 1* (0+0+0+0+9+2+4+6+8)] mod 10 = - [3* 43 + 29] = - 158 mod 10 = **2**

5.3.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

In the clear text, the Application Identifier (behind the <FNC1>) is shown in parentheses: "(00)".
The extension digit, the company prefix, the serial number and the check digit are usually separated from each other with a gap.
E.g. "`(00) x pp ppppp ssssssss c`".

5.3.6 Code construction: Steps

The application message data M = "0 0 <c#3> ... <c#19>" is converted to the symbology message character set as follows.
(That means there are 19 application message characters.)

> **<startC> <FNC1> 0 0 <c#3> ... <c#19> <CHK mod 10>**

(That means there are 21 symbology message characters.)

1. Apply application-specific check characters

Calculate the CHK according to the mod 10 algorithm above and append it to the message.

2. Insert application control characters

Insert the <FNC1> control character in front of the message.

3. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the Code 128 symbology, using Code C.
Note that the symbology-specific mod 103 CHK method also has to be applied.

[startC] [FNC1] [00] [<c#3><c#4>] ... [<c#19><CHK mod 10>] [<CHK mod 103>] [stop]

(That means there are 14 symbology characters.)

5.3.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		Application Identifier: 00 Extension digit: 9 UCC/EAN Company prefix: 0009009 Serial Reference Number: 123456789
		(00) 9 0 0 0 9 0 0 9 1 2 3 4 5 6 7 8 9
		-- (*)
		0 0 9 0 0 0 9 0 0 9 1 2 3 4 5 6 7 8 9 **2**
		<startC> <FNC1> 00 90 00 90 09 12 34 56 78 92
		-- (**)
		[startC] [FNC1] [00] [90] [00] [90] [09] [12] [34] [56] [78] [92] **[#28] [stop]**

(*) For the calculation of the CHK for UPC-128 see above.

(**) <u>Checksum calculation</u> (Code 128 CHK)

Message character	<startC>	<FNC1>	00	90	00	90	09	12	34	56	78	92	
Reference number	105	102	0	90	0	90	09	12	34	56	78	92	
Weight factor	*1	*1	*2	*3	*4	*5	*6	*7	*8	*9	*10	*11	
Product -> Sum	105	+ 102	+ 0	+ 270	+ 0	+ 450	+ 54	+ 84	+ 272	+ 504	+ 780	+ 1012	= 3633

-->
#CHK = 35x <u>103</u> + **28**

5.4 Application: UPS-128

5.4.1 Overview

This application is known under these names:
- UPS-128

Usage
It is used to encode Tracking numbers of shipments by UPS (United Parcel Service).

Example
This is a sample of a typical barcode of this application.
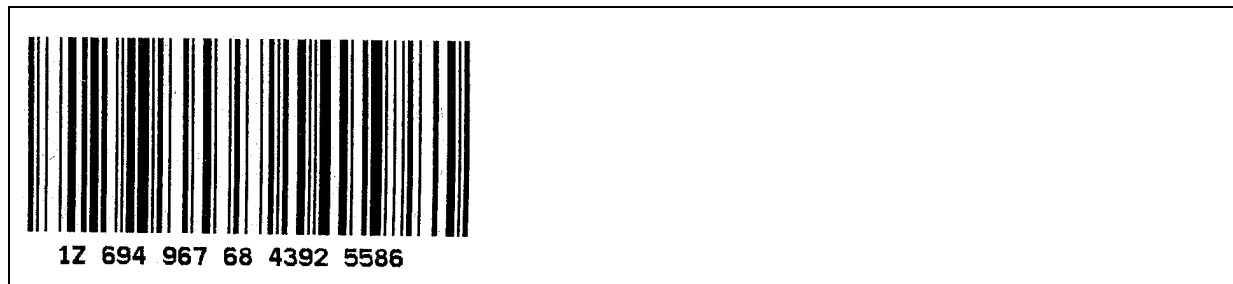
Further information

The specification of this application is maintained at UPS.

Other sources of information can be found at Appendix L.

Base symbology

Code 128

5.4.2 Character set

Message character set

The message character set consists of **10** characters.

Message data characters
Numeric only = {0,...,9} + "Z".

Message control characters
None.

Message length restrictions
The message length is fixed to = 2 ("1Z") + **15 digits** + 1 CHK.

Other message data restrictions
The first two digits must be "1Z".

Semantics (Interpretation of message characters)

An arbitrary UPS-128 message has a fixed length of 2+15+1 digits, with this structure:
  1 Z <data> <CHK>

The "1Z" is the "**FACT ID**" which identifies the data as UPS-128 Tracking number.
The <data> is a fixed length data structure of 15 digits. This is a unique number consisting of these parts:
 - digits 3..8   (6)   UPS client ID
 - digits 9..10   (2)   service code
 - digits 11..17   (7)   sequential number

The **Service code** identifies the type of service:
- 53 = UPS -Standard collecting
- 54 = UPS -Express plus

- 55 = UPS -Express
- 56 = UPS -Standard

The <CHK> is the application-specific mod 10 check digit.

5.4.3 Encoding (0): (Application level)

There is no specific conversion necessary at this stage.

5.4.4 Checksum methods

There is 1 mandatory application-specific check digit mod 10, with alternating weights *1 and *2.
It has to be applied before and in addition to the Code 128 mod 103 CHK algorithm.

CHK Formula

1 mandatory CHK

$$
\begin{aligned}
\#CHK &= - [*1,*2]... \text{ x } \#M \text{ mod } 10 \\
&= - (1*<c\#3> + 2*<c\#4> + ... + 2*<c\#16> + 1*<c\#17>) \text{ mod } 10 \\
&= - [2* (<c\#4> + <c\#6> + ... + <c\#16>) + 1* (<c\#3> + <c\#5> + ... + <c\#17>)] \text{ mod } 10
\end{aligned}
$$

CHK Algorithm

This special mod 10 algorithm is only applied to the 15 message digits.
1. Sum up all even-positioned digits. Multiply the sum by 2.
2. Add the remaining (odd-positioned) digits.
3. The check digit is the difference to the next multiple of 10.

Example (CHK calculation)

M = "1 Z 9 0 4 7 1 1 5 6 1 2 3 4 5 6 7"
-->
CHK = - [2* (0+7+1+6+2+4+6) + 1* (4+1+5+1+3+5+7)] mod 10 = - [2*26 + 26] = - 8 mod 10 = **2**

5.4.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

In the clear text, the Application Identifier (behind the <FNC1>) is shown in parentheses: "(00)".
The extension digit, the company prefix, the serial number and the check digit are usually separated from each other with a gap.

5.4.6 Code construction: Steps

The application message data M = "1 Z <c#3> ... <c#17>" is converted to the symbology message character set as follows.
(That means there are 17 application message characters.)

<startB> 1 Z <CodeC> <c#3> ... <c#17> <CHK mod 10>

(That means there are 20 symbology message characters.)

1. Apply application-specific check characters

Calculate the CHK according to the mod 10 algorithm above and append it to the message.

2. Insert application control characters

Insert the <codeC> control character before the digits.

3. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the Code 128 symbology, using Code B.
Note that the symbology-specific mod 103 CHK method also has to be applied.

[startB] [1] [Z] [CodeC] [<c#3><c#4>] ... [<c#17><CHK mod 10>] [<CHK mod 103>] [stop]

(That means there are 14 symbology characters.)

5.4.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		"1 Z 9 0 4 7 1 1 5 6 1 2 3 4 5 6 7"
		-- (*)
		1 Z 9 0 4 7 1 1 5 6 1 2 3 4 5 6 7 **2**
		<startB> 1 Z **<CodeC>** 90 47 11 56 12 34 56 72
		-- (**)
		[startB] 1 Z [CodeC] [90] [47] [11] [56] [12] [34] [56] [72] **[#45] [stop]**

(*) For the calculation of the CHK for UPS-128 see above.

(**) <u>Checksum calculation</u> (Code 128 CHK)

Message character	<startB>	1	Z	<CodeC>	90	47	11	56	12	34	56	72	
Reference number	104	17	58	99	90	47	11	56	12	34	56	72	
Weight factor	*1	*1	*2	*3	*4	*5	*6	*7	*8	*9	*10	*11	
Product -> Sum	104	+ 17	+ 116	+ 297	+ 360	+235	+ 66	+ 392	+ 96	+ 306	+ 560	+ 792	= 3341

-->
#CHK = 32x <u>103</u> + **45**

5.5 Symbology: **Code 39**

5.5.1 Overview

This symbology is known as:
- Code 3/9
- Code 3 of 9
- USD-3
- Alpha39

Usage

Application:	industrial, tracking, general-purpose; not for retail;
Regional use:	world-wide;
Popularity:	very widely used;

Remarks

The historically first alphanumeric symbology. The most popular symbology used for ID, inventory, and tracking purposes.
The full 128 lower ASCII characters can be represented by combining two characters. See "Code 39 Extended" below.
Characteristics: (-) needs a lot of space; (-) low tolerance;

Example

This is a sample of a typical barcode of this symbology.
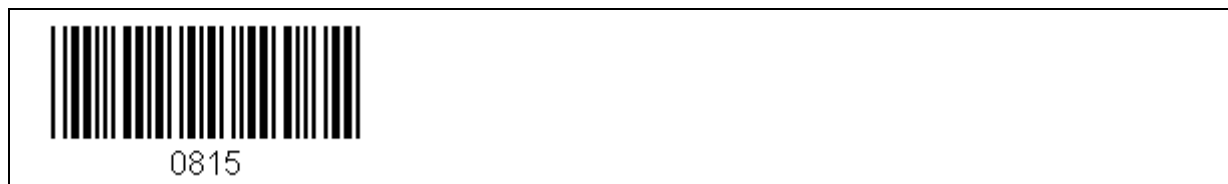
Further information

The specification of this symbology is maintained at:
- USA: ANSI
  - (ANSI MH10.8-1983, Chapter 4.4)
  - (ANSI/AIM BC1-1995 (USS))
- Europe: EN 800

Other sources of information can be found at Appendix L.

Applications

Applications based on this symbology are:
- Code 39 Extended (*)
- PZN (Pharma-Zentral-Nummer) (*)
- Danish PTT 39
- French Postal 39 A/R
Those marked (*) are directly supported by this product; they are explained in separate sections right after this symbology.

5.5.2 Character set (Alphabet)

Message character set

The message character set consists of **43** characters.

Message data characters
Alphanumeric =
- 10 numeric/digits {0,...,9}
- 26 uppercase letters {A,...,Z}
- 7 special symbols { - | . | $ | / | + | % | <space> }

Message control characters

The 4 characters {$, %, /, +} are also used for extending the character set. See "Application: Code 39 Extended" below.

Message length restrictions

The message length is variable and not limited.

Symbology character set

The symbology character set consists of **44 = 43** (message) **+ 1** (control) characters.

Symbology data characters
There is a symbology character for every message character. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start/stop]	start and stop of barcode

5.5.3 Reference numbers

This table assigns a unique reference number to each message character.

Character	Ref. #	Character	Ref. #	Character	Ref. #	Character	Ref. #
'0' .. '9'	#0 .. #9						
A	#10	J	#19	S	#28	'−' minus	#36
B	#11	K	#20	T	#29	'.' point	#37
C	#12	L	#21	U	#30	' ' space	#38
D	#13	M	#22	V	#31	'$' dollar	#39
E	#14	N	#23	W	#32	'/' slash	#40
F	#15	O	#24	X	#33	'+' plus	#41
G	#16	P	#25	Y	#34	'%' percent	#42
H	#17	Q	#26	Z	#35		
I	#18	R	#27				

There is no reference number assigned to the [start/stop] character.

5.5.4 Checksum methods

Code 39 defines 1 optional check character (mod 43), weighted with all factors = 1.

CHK Formula

1 optional CHK

> **#CHK = + ... [*1] x #M mod 43**

CHK Algorithm
The check character is calculated as the modulo 43 remainder of the sum of all reference numbers.

Example (CHK calculation)

M = "C O D E <SP> 3 9"
-->

Message character	C	O	D	E	<sp>	3	9	#CHK
Reference number	12	24	13	14	38	3	9	
Weight factor	* 1	* 1	* 1	* 1	* 1	* 1	* 1	
Product -> Sum	12	+ 24	+ 13	+ 14	+ 38	+ 3	+ 9	=113 = $2 \times \underline{43} + \mathbf{27}$

-->
CHK = #27 = "**R**"

5.5.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces)
Dimensionality:	1-D

(Continuous / Discrete):	Discrete (inter-character space contains no information)	
Logical encoding type:	"3 of 9" = 3 elements out of 9 are wide	
# Modules per element:	2-level = N, W	
# Elements per character:	**9 el** = 5 b + 4 s	
# Modules per character:	**12 md** = 6 eN + 3 eW	-- 3 elements wide
	= 3 bN + 2 bW + 3 sN + 1 sW	-- 2 bars wide, 1 space wide
	= 7 mb + 5 ms	
Self-checking:	yes	
Bi-directional:	yes	

5.5.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.5.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Height	Recommended are min. 20 mm or 25% of the width.
N:W ratio	1:2.0 .. 3.0; recommended: 1:2.25

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.5.8 Code construction: Steps

The message data M = "<c#1>...<c#n>" is converted into the symbology character set according to the following scheme.

[start/stop] [<c#1>] ... [<c#n>] [<CHK>] [start/stop]

---

1. Apply check characters

If required, calculate the CHK according to the algorithm above and append it to the message.

2. Convert the message characters into symbology characters

Convert each message character <c#i> into the symbology character [<c#i>].

3. Insert symbology control characters

Prepend and append the [start/stop] character.

5.5.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.5.10 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		C O D E <SP> 3 9
		-- (*)
		C O D E <SP> 3 9 **R**
		[start/stop] [C] [O] [D] [E] [SP] [3] [9] [R] [start/stop]
		 CODE 39

(*) For the calculation of the CHK see above.

5.6 Application: Code 39 Extended

5.6.1 Overview

This symbology is known as:
- Code 39 Extended
- Code 39 Full ASCII

Usage
This application is used to encode all characters of the lower half of the ASCII table, from hex <00> to <7F>.

Example
This is a sample of a typical barcode of this application.

Further information

The specification of this application is maintained at: (see base symbology).

Other sources of information can be found at Appendix L.

Base symbology

Code 39 (with or without CHK)

5.6.2 Character set

Message character set

The message character set consists of **128** characters (Full ASCII).

Message data characters
Full ASCII character set = <00> .. <7F>.

Message control characters
None.

Message length restrictions
The message length is variable and not limited.

5.6.3 Encoding (0): (Application level)

An **application message** is converted to a **symbology message** according to the following scheme.

The ASCII control characters and lowercase characters and special characters are encoded by prefixing Code 39 characters with one of the characters { $ | % | / | + }, according to the table below.

Char.	Encod.	Char.	Encod.	Char.	Encod.	Char.	Encod.	Char.	Encod.	Char.	Encod.
NUL	%U	DLE	$P	SP	space	0	/P 0	@	%V	'	%W
SOH	$A	DC1	$Q	!	/A	...	...	A	A	a	+A
STX	$B	DC2	$R	"	/B	9	/Y 9	...	...	...	...
ETX	$C	DC3	$S	#	/C			Z	Z	z	+Z
EOT	$D	DC4	$T	$	/D						
ENQ	$E	NAK	$U	%	/E						
ACK	$F	SYN	$V	&	/F						
BEL	$G	ETB	$W	`	/G						

BS	$H	CAN	$X	(	/H						
HT	$I	EM	$Y	)	/I						
LF	$J	SUB	$Z	*	/J	:	/Z				
VT	$K	ESC	%A	+	/K +	;	%F	[	%K	{	%P
FF	$L	FS	%B	,	/L	<	%G	\	%L	|	%Q
CR	$M	GS	%C	−	/M −	=	%H	]	%M	}	%R
SO	$N	RS	%D	.	/N .	>	%I	^	%N	~	%S
SI	$O	US	%E	/	/O /	?	%J	_	%O	DEL	%T

The following table can be used for quick decoding.

letter	pre-$	pre-%	pre-/	pre-+
A	SOH	ESC	!	a
B	STX	FS	"	b
C	ETX	GS	#	c
D	EOT	RS	$	d
E	ENQ	US	%	e
F	ACK	;	&	f
G	BEL	<	'	g
H	BS	=	(	h
I	HT	>	)	i
J	LF	?	*	j
K	VT	[	+	k
L	FF	\	,	l
M	CR	]	−	m
N	SO	^	.	n
O	SI	_	/	o
P	DLE	{	0	p
Q	DC1	|	1	q
R	DC2	}	2	r
S	DC3	~	3	s
T	DC4	DEL	4	t
U	NAK	NUL	5	u
V	SYN	@	6	v
W	ETB	`	7	w
X	CAN		8	x
Y	EM		9	y
Z	SUB		:	z

Note: (Code 39 Extended ambiguity 1)
The four escape/prefix characters "$", "%", "/", "+" used, are actually characters of the symbology alphabet.
Therefore, a sequence like "/E" is ambiguous: is it to mean the characters "/E" or "%" ?
For this reason, a barcode reader always has to be explicitly configured to be in Normal or in "Extended" ("Full ASCII") mode.

Note: (Code 39 Extended ambiguity 2)
Some characters can be coded in two different ways. E.g. "$" can be coded as either "$" or "/D".
The choice is entirely up to the implementor.
The characters affected are: "$" = /D, "%" = /E, "/" = /O, "+" = /K, "-" = /M, "." = /N, "0" = /P, ..., "9" = /Y.
It is important to be aware that, depending on the choice made, a different CHK will result !
The barcode reader, however, will always decode the original message correctly.

5.6.4 Checksum methods

There is no application-specific CHK defined for Code 39 Extended.
The normal Code 39 checksum method can (optionally) be applied, after the Extending has been constructed.

5.6.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.
Note that clear text for Extended is difficult to achieve with a barcode font.

5.6.6 Code construction: Steps

The application message data M = "<c#1> ... <c#n>" is converted to the symbology message character set, by pairing one of {$ % / +} and an uppercase letter, as indicated in the table above.

Perform extending by escaping

Convert each message character by expanding it into a pair consisting of an escape character (one of '$', '%', '/', '+') and an uppercase letter, according to the Encoding (0) above.

Apply the Code construction steps of the base symbology

Apply the Code construction steps of the Code 39 symbology.
The CHK may or may not be required.

5.6.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.
Here without CHK.

		Data
		C o d e <SP> 3 9 <SP> e x
		C **+ O + D + E** <SP> 3 9 <SP> **+ E + X**
		[start/stop] [C] [+] [O] [+] [D] [+] [E] [SP] [3] [9] [SP] [+] [E] [+] [X] **[start/stop]**
		Code 39 ex

5.7 Application: PZN

5.7.1 Overview

This application is known under these names:
- PZN, PZN8
- Pharma-Zentral-Nummer

Usage

Application:	Distribution of pharmaceutical / health care products
Regional use:	Germany only

Remarks

From March 2013, the former PZN specification (with 6 digits + 1 CHK, now a.k.a. "**PZN7**")
 has been superseded by **PZN8**, which has 1 digit more (7 digits + 1 CHK).
Legacy 6+1-digit PZN numbers just receive a leading "0".

Example

This is a sample of a typical barcode of this application (with PZN7).

PZN-1234562

Further information

The specification of this application is maintained at:
 Informationsstelle für Arzneispezialitäten GmbH (IFA)
 Hamburger Allee 26-28
 D-60486 Frankfurt am Main
 Germany

 Phone/Fax: +49-(0)69-979919-0/39
 Email: <ifa@ifaffm.de>

Other sources of information can be found at Appendix L.

Base symbology

Code 39 (without CHK)

5.7.2 Character set

Message character set

The message character set consists of **11** characters.

Message data characters

{0, ..., 9, - } = numeric / digits, minus sign

Message control characters

None.

Message length restrictions

For PZN8, the message length is fixed to = 7 digits + 1 CHK.
(For PZN7, the message length was = 6 digits + 1 CHK.)

5.7.3 Encoding (0): (Application level)

An **application message** is converted to a **symbology message** according to the following scheme.

The application character set is part of the symbology message character set.
Therefore, there is no conversion necessary.

5.7.4 Checksum methods

PZN defines 1 mandatory PZN-specific CHK mod 11, which is applied to the 7 (formerly 6) message digits.
The Code 39 CHK is not used.

CHK Formula

1 mandatory CHK

- **PZN8:   #CHK = + [*1, *2, *3, *4, *5, *6, *7] x #M mod 11**

- **PZN7:   #CHK = + [*2, *3, *4, *5, *6, *7] x #M mod 11**

CHK Algorithm
Each of the seven message digits has to be multiplied with its position.

The check digit is calculated as the remainder **modulo 11** of the sum of the products.
Any PZN whose check digit yields 10 is not used.

Example (CHK calculation)

(With PZN7): M = "1 2 3 4 5 6"
-->

Message character Reference number	0	1	2	3	4	5	6	
Weight factor	*1	*2	*3	*4	*5	*6	*7	
Product -> Sum	0	+ 2	+ 6	+ 12	+ 20	+ 30	+ 42	= 112 = 10 * 11 + **2**

-->
#CHK = **2**

5.7.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Height	The minimum code height is 6 mm.
N:W ratio	The code should be used with ratio 1:3. If the module width is larger than 0.5 mm, the ratio 1:2 is recommended.

Imaging conventions

There are no specific conventions about the physical layout of the image.

The clear text should be "P Z N - <d#1> ... <d#6> <d#7> <CHK>".
I.e. the text "PZN-" is shown in front, and the check digit is shown at the end as well.
Note that this cannot be achieved with a barcode font, but it needs to be created separately underneath the non-CTX barcode using a normal text font.

5.7.6 Code construction: Steps

The application message data M = "<d#1> ... <d#6>" is converted to the symbology message character set as follows.

<minus> <d#1> ... <d#6> <d#7> <CHK_11>

1. Apply application-specific check characters

Calculate the CHK according to the algorithm above and append it to the message.

2. Insert application control characters

Insert a minus sign in front of the message.
Note: The characters "PZN" are not barcoded.

[start] [-] [<d#1>] ... [<d#6>] [<d#7>] [<CHK_11>] [stop]

---

3. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the Code 39 symbology.
The symbology-specific CHK method must NOT be applied.

5.7.7 Code construction: The complete procedure (Example)

This section contains a full example (with PZN7) of all stages of the Code construction.

		Data
		1 2 3 4 5 6
		-- (*)
		1 2 3 4 5 6 **2**
		- 1 2 3 4 5 6 2
		[start] [-] [1] [2] [3] [4] [5] [6] [2] [stop]
		PZN-1234562

(*) For the calculation of the CHK see above.

5.8 Application: Danish PTT 39

5.8.1 Overview

This application is known under these names:
- Danish PTT 39
- Danish Postal 39

Usage
Application: Danish Post: parcel labels : tracking number ("trace & track") for shipment through postal services
Regional use: Denmark only

Example
This is a sample of a typical barcode of this application.

Further information

The specification of this application is maintained at:
  Post Danmark: //www.postdanmark.dk/

Other sources of information can be found at Appendix L.

Base symbology

Code 39 (without CHK)

5.8.2 Character set

Message character set

Message data characters
{0, ..., 9} + {C,K,O,U}

Message control characters
None.

Message length restrictions
The message length is fixed to 10 or 8 digits.

Semantics (Interpretation of message characters)
- **10 digits** ("Consignment #") or
- { CC | CK | CO | CU } (for abroad) + **8 digits** ("Consignment #")
Behind this, the CHK and "DK" are automatically added.

5.8.3 Encoding (0): (Application level)

An **application message** is converted to a **symbology message** according to the following scheme.

The application character set is part of the symbology message character set.
Therefore, there is no conversion necessary.

5.8.4 Checksum methods

Danish PTT 39 defines 1 mandatory application-specific CHK mod 11, which is applied to the 10 message digits.
The Code 39 CHK is not used.

CHK Formula

1 mandatory CHK

#CHK = - [*9, *7, *8, *6, *4, *2, *3, *5, *9, *7] x #M mod 11

Further: A result of 0 becomes 5. A result of 10 becomes 0.

CHK Algorithm

The 10 message digits have to be multiplied with the factors.
If the first two characters are non-numeric (CC,CK,CO,CU), their reference values in this CHK algorithm are 0.
The check digit is calculated as the difference of the sum of the products to the next multiple of 11.
Further: A result of 0 becomes 5. A result of 10 becomes 0.

Example (CHK calculation)

M = "1 2 3 4 5 6 7 8 9 0"
-->

Message character Reference number	1	2	3	4	5	6	7	8	9	0	
Weight factor	*9	*7	*8	*6	*4	*2	*3	*5	*9	*7	
Product -> Sum	9	+ 14	+ 24	+ 24	+ 20	+ 12	+ 21	+ 40	+ 81	+ 0	= 245 = 23 * 11 - 8

-->
#CHK = **8**

5.8.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Imaging conventions

There are no specific conventions about the physical layout of the image.

The clear text should be "<d#1> ... <d#10> <CHK_11> D K".
I.e. the check digit is shown as well.
Note that this cannot be achieved with a barcode font, but it needs to be created separately underneath the non-CTX barcode using a normal text font.

5.8.6 Code construction: Steps

The application message data M = "<d#1> ... <d#10>" is converted to the symbology message character set as follows.

<d#1> ... <d#10> <CHK_11> D K

1. Apply application-specific check characters

Calculate the CHK according to the algorithm above and insert it into the message.

2. Insert application control characters

Append "DK".

[start] [<d#1>] ... [<d#10>] [<CHK_11>] [D] [K] [stop]

3. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the Code 39 symbology.
The symbology-specific CHK method must NOT be applied.

5.8.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		1 2 3 4 5 6 7 8 9 0
		-- (*)
		1 2 3 4 5 6 7 8 9 0 **8**
		1 2 3 4 5 6 7 8 9 0 8 **D K**
		[start] [1] [2] [3] [4] [5] [6] [7] [8] [9] [0] [8] [D] [K] [stop]

(*) For the calculation of the CHK see above.

5.9 Application: French Postal 39 A/R

5.9.1 Overview

This application is known under these names:
- French Postal A/R 39

Usage
 Application: French Post: Used on registered letter forms ("Recommandé").
 Regional use: France only

Example
This is a sample of a typical barcode of this application.

Further information

The specification of this application is maintained at:
  La Poste France: //www.laposte.fr/

Other sources of information can be found at Appendix L.

Base symbology

Code 39 (without CHK)

5.9.2 Character set

Message character set

Message data characters
{0, ..., 9} + {A, B, R}

Message control characters
None.

Message length restrictions
The message length is fixed to **10** characters: = ("**RA**" or "**RB**") + **8 digits**
Behind this, the following is automatically added: + 1 CHK + "**FR**".

5.9.3 Encoding (0): (Application level)

An **application message** is converted to a **symbology message** according to the following scheme.

The application character set is part of the symbology message character set.
Therefore, there is no conversion necessary.

5.9.4 Checksum methods

French Postal A/R 39 defines 1 mandatory application-specific CHK mod 11, which is applied to the 8 message digits.
The Code 39 CHK is not used.

CHK Formula

1 mandatory CHK

  #CHK = - [*3, *5, *7, *9, *8, *6, *2, *4] x #M mod 11

Further: A result of 0 becomes 5. A result of 10 becomes 0.

CHK Algorithm

The 8 message digits have to be multiplied with the factors. (The "RA/RB" are not taken into account.)
The check digit is calculated as the difference of the sum of the products to the next multiple of 11.
Further: A result of 0 becomes 5. A result of 10 becomes 0.

Example (CHK calculation)

M = "R A 1 2 3 4 5 6 7 8"
-->

Message character Reference number	1	2	3	4	5	6	7	8			
Weight factor	*3	*5	*7	*9	*8	*6	*2	*4			
Product -> Sum	3	+ 10	+ 21	+ 36	+ 40	+ 36	+ 14	+ 32			= 192 = 18 * 11 - **6**

-->
#CHK = **6**

5.9.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

N:W ratio	1: 2.0 .. 1:3.0

Imaging conventions

There are no specific conventions about the physical layout of the image.

The clear text should be "R A/B <d#1> ... <d#8> <CHK> F R".
I.e. the check digit is shown as well.

5.9.6 Code construction: Steps

The application message data M = "R A/B <d#1> ... <d#8>" is converted to the symbology message character set as follows.

R <A/B> <d#1> ... <d#8> <CHK_11> F R

---

1. Apply application-specific check characters

Calculate the CHK according to the algorithm above and insert it into the message.

2. Insert application control characters

Append "FR".

[start] [R] [A/B] [<d#1>] ... [<d#8>] [<CHK_11>] [F] [R] [stop]

---

3. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the Code 39 symbology.
The symbology-specific CHK method must NOT be applied.

5.9.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data

		R A 1 2 3 4 5 6 7 8
		-- (*)
		R A 1 2 3 4 5 6 7 8 **6**
		R A 1 2 3 4 5 6 7 8 6 **F R**
		[start] [R] [A] [1] [2] [3] [4] [5] [6] [7] [8] [6] [F] [R] [stop]

(*) For the calculation of the CHK see above.

5.10 Symbology: **Code 93**

5.10.1 Overview

This symbology is known as:
- Code 93
- USS-93

This alpha-numeric code is quite similar to Code 39, but more compact (compressed version).

Usage
 Popularity: not as widely used

Remarks
The full 128 lower ASCII characters can be represented by escaping characters. See "Code 93 Extended" below.
Characteristics: (+) less space than Code 39; (-) low tolerance;

Example
This is a sample of a typical barcode of this symbology.

Further information

The specification of this symbology is maintained at:
(ANSI/AIM BC5-1995 (USS))

Other sources of information can be found at Appendix L.

Applications

Applications based on this symbology are:
- Code 93 Extended (*)
Those marked (*) are directly supported by this product; they are explained in separate sections right after this symbology.

5.10.2 Character set (Alphabet)

Message character set

The message character set consists of **47 = 43** (message) **+ 4** (control) characters.

This is the same character set as Code 39, enhanced by 4 control characters.

The full set of all 128 lower ASCII characters can be encoded by escaping characters, more specifically by prefixing uppercase letters with one of the special control characters. See "Code 93 Extended" below.

Message data characters
Alphanumeric:
- 10 numeric digits {0,...,9}
- 26 uppercase letters {A,...,Z}
- 7 special characters { - | . | $ | / | + | % | <SP> }

Message control characters
There are 4 unique control characters: { ($), (%), (/), (+) }.

Control character	Function
($)	Escape characters for Code 93 Extended.
(%)	They cannot stand alone by themselves, but they must be in front of an uppercase letter.
(/)	
(+)	

Message length restrictions
The message length is variable and not limited.

Symbology character set

The symbology character set consists of **49 = 47** (message) **+ 2** (control) characters.

Symbology data characters
There is a symbology character for every message character. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start]	The start of a barcode.
[stop]	The stop of a barcode.

5.10.3 Reference numbers

This table assigns a unique reference number to each message character.

Char.	Ref. #	Char.	Ref. #	Char.	Ref. #	Char.	Ref. #	Char.	Ref. #
0	#0	A	#10	K	#20	U	#30	/	#40
1	#1	B	#11	L	#21	V	#31	+	#41
2	#2	C	#12	M	#22	W	#32	%	#42
3	#3	D	#13	N	#23	X	#33		
4	#4	E	#14	O	#24	Y	#34	($)	#43
5	#5	F	#15	P	#25	Z	#35	(%)	#44
6	#6	G	#16	Q	#26	–	#36	(/)	#45
7	#7	H	#17	R	#27	.	#37	(+)	#46
8	#8	I	#18	S	#28	SP	#38		
9	#9	J	#19	T	#29	$	#39		

There is no reference number assigned to the [start] and [stop] characters.

5.10.4 Checksum methods

Code 93 requires 2 mandatory check characters, named "C" and "K". They are calculated as the modulo 47 sum of the reference numbers, weighted from right to left with factors 1,...,20 (C), and 1,...,15 (K), respectively.

CHK Formula

1 mandatory CHK

> **#C= + ... [*20, *19, ..., *2, *1] x #M mod 47**
>
> **#K= + ... [*15, *14, ..., *2, *1] x #(MC) mod 47**

CHK Algorithm
The 1st check character is calculated as the modulo 47 remainder of the sum of the multiplication from right to left of each reference value with its position number. After each 20th position, the multiplication factor is set to 1 again.
Put it at the end of the message.
For the 2nd check character, apply the same algorithm to the new message. The first check character is already multiplied with position 1, and the multiplication factor is set to 1 after each 15th character.

Example (CHK calculation)

M = "C O D E <SP> 9 3"
-->
"CHK-C"

Message character	C	O	D	E	<SP>	9	3		CHK =>
Reference number	12	24	13	14	38	9	3		=>
Weight factor	*7	*6	*5	*4	*3	*2	*1		
Product -> Sum	84	+ 144	+ 65	+ 56	+ 114	+ 18	+ 3		= 484 = 10 x <u>47</u> + **14**

-->
CHK-C = #14 = "**E**"
-->
M' = "C O D E <SP> 9 3 **E**"

"CHK-K"

Message character	C	O	D	E	<SP>	9	3	**E**		CHK =>
Reference number	12	24	13	14	38	9	3	14		=>
Weight factor	*8	*7	*6	*5	*4	*3	*2	*1	*	
Product -> Sum	96	+ 168	+ 78	+ 70	+ 152	+ 27	+ 6	+ 14	+	= 611 = $13 \times \underline{47} + 0$

-->
CHK-K = #0 = "**0**"
-->
M'' = "C O D E <SP> 9 3 **E 0**"

5.10.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces)
Dimensionality:	1-D
(Continuous / Discrete):	Continuous (inter-character space contains information)
Logical encoding type:	"3 wide out of 9"
# Modules per element:	multi-level: - b = 1, 2, 3 md - s = 1, 2, 3, 4 md - [start/stop] = b4 (4 modules bar)
# Elements per character:	**6 el** = 3 b + 3 s -- 3 bars + 3 spaces
# Modules per character:	**9 md** = (various) = 3x 2 md + 3x 1 md, or = 1x 3 md + 1x 2 md + 4x 1 md
Self-checking:	no ! (needs CHK)
Bi-directional:	yes

5.10.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.10.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.10.8 Code construction: Steps

The message data M = "<c#1>...<c#n>" is converted into the symbology character set according to the following scheme.

 [start] [<c#1>] ... [<c#n>] [<CHK-C>] [<CHK-K>] [stop]

---

1. Apply check characters

Calculate the 2 CHK according to the algorithm above and append them to the message.

2. Convert the message characters into symbology characters

Convert each message character <c#i> into the symbology character [<c#i>].

3. Insert symbology control characters

Prepend the [start] character and append the [stop] character.

5.10.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.10.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		C O D E <SP> 9 3
		-- (*)
		C O D E <SP> 9 3 **E 0**
		[start] [C] [O] [D] [E] [SP] [9] [3] [E] [0] [stop]
		 CODE 93

(*) For the calculation of the CHK see above.

5.11 Application: Code 93 Extended

5.11.1 Overview

This application is known under these names:
- Code 93 Extended
- Code 93 Full ASCII

Example
This is a sample of a typical barcode of this application.

Code 93 ex

Further information

The specification of this application is maintained at: (see base symbology).

Other sources of information can be found at Appendix L.

Base symbology

Code 93

5.11.2 Character set

Message character set

The message character set consists of **128** characters (Full ASCII).

Message data characters
Full ASCII character set = <00> .. <7F>.

Message control characters
None.

Other than in Code 39 Extended, the extra control characters ($), (%), (/), (+) are used for Extending.

Message length restrictions
The message length is variable and not limited.

5.11.3 Encoding (0): (Application level)

An **application message** is converted to a **symbology message** according to the following scheme.

The ASCII control characters and lowercase characters and special characters are encoded by prefixing uppercase letters with one of the symbology control characters { (%) | ($) | (/) | (+) }, according to the table below.

Char.	Encod.	Char.	Encod.	Char.	Encod.	Char.	Encod.	Char.	Encod.	Char.	Encod.
NUL	(%)U	DLE	($)P	SP	space	0	(/)P 0	@	(%)V	'	(%)W
SOH	($)A	DC1	($)Q	!	(/)A	...	...	A	A	a	(+)A
STX	($)B	DC2	($)R	"	(/)B	9	(/)Y 9	...	...	...	...
ETX	($)C	DC3	($)S	#	(/)C			Z	Z	z	(+)Z
EOT	($)D	DC4	($)T	$	(/)D $						
ENQ	($)E	NAK	($)U	%	(/)E %						
ACK	($)F	SYN	($)V	&	(/)F						
BEL	($)G	ETB	($)W	`	(/)G						
BS	($)H	CAN	($)X	(	(/)H						

HT	($)I	EM	($)Y	)	(/)I									
LF	($)J	SUB	($)Z	*	(/)J			:	(/)Z					
VT	($)K	ESC	(%)A	+	(/)K	+		;	(%)F	[	(%)K	{	(%)P
FF	($)L	FS	(%)B	,	(/)L			<	(%)G	\	(%)L	|	(%)Q
CR	($)M	GS	(%)C	–	(/)M	–		=	(%)H	]	(%)M	}	(%)R
SO	($)N	RS	(%)D	.	(/)N	.		>	(%)I	^	(%)N	~	(%)S
SI	($)O	US	(%)E	/	(/)O	/		?	(%)J	_	(%)O	DEL	(%)T

The following table can be used for quick decoding.

letter	($) ...	(%) ...	(/) ...	(+) ...
A	SOH	ESC	!	a
B	STX	FS	"	b
C	ETX	GS	#	c
D	EOT	RS	$	d
E	ENQ	US	%	e
F	ACK	;	&	f
G	BEL	<	'	g
H	BS	=	(	h
I	HT	>	)	i
J	LF	?	*	j
K	VT	[	+	k
L	FF	\	,	l
M	CR	]	–	m
N	SO	^	.	n
O	SI	_	/	o
P	DLE	{	0	p
Q	DC1	|	1	q
R	DC2	}	2	r
S	DC3	~	3	s
T	DC4	DEL	4	t
U	NAK	NUL	5	u
V	SYN	@	6	v
W	ETB	`	7	w
X	CAN		8	x
Y	EM		9	y
Z	SUB		:	z

Note: (Code 93 Extended escape characters)
This is essentially the same table as for Code 39 Extended, but with a separate set of escape/control characters which cannot be confused with the alphabet characters.
They are denoted "($), (%), (/), (+)" instead of "$, %, /, +".
Thus a barcode reader can automatically determine if some given text is supposed to be Normal or Full ASCII.

Note: (Code 93 Extended ambiguity)
Some characters can be coded in two different ways. E.g. "$" can be coded as either "$" or "(/)D".
The choice is entirely up to the implementor.
The characters affected are: "$" = (/)D, "%" = (/)E, "/" = (/)O, "+" = (/)K, "-" = (/)M, "." = (/)N, "0" = (/)P, ..., "9" = (/)Y.
It is important to be aware that, depending on the choice made, a different CHK will result !
The barcode reader, however, will always decode the original message correctly.

5.11.4 Checksum methods

There is no application-specific CHK for Code 93 Extended.
The normal Code 93 checksum method is used, after the Extending has been applied.

5.11.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.
Note that clear text for Extended is difficult to achieve with a barcode font.

5.11.6 Code construction: Steps

The application message text M = "<c#1> ... <c#n>" is converted to the symbology message character set as indicated in the Encoding table above.

1. Perform extending by escaping

Convert each message character by expanding it into a pair consisting of an escape character (one of '$', '%', '/', '+') and an uppercase letter, according to the Encoding (0) above.

2. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the Code 93 symbology.

5.11.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		C o d e <SP> 9 3 <SP> e x
		C **(+) O (+) D (+) E** <SP> 9 3 <SP> **(+) E (+) X**
		-- (*)
		C (+) O (+) D (+) E <SP> 9 3 <SP> (+) E (+) X **6 P**
		[start] [C] [(+)] [O] [(+)] [D] [(+)] [E] [SP] [9] [3] [SP] [(+)] [E] [(+)] [X] [6] [P] **[stop]**
		Code 93 ex

(*) Calculations for **"CHK-C"** and **"CHK-K"**

Calculation for **"CHK-C"**

Message character	C	(+)	O	(+)	D	(+)	E	<SP>	9	3	<SP>	(+)	E	(+)	X		
Reference number	12	46	24	46	13	46	14	38	9	3	38	46	14	46	33		
Weight factor	*15	*14	*13	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1		
Product -> Sum	180	+ 644	+ 312	+ 552	+ 143	+ 460	+ 126	+ 304	+63	+18	+190	+184	+42	+92	+33		= 3343

-->
#CHK-C = 3343 = 71 x <u>47</u> + **6** --> CHK-C = #6 = **"6"**

Calculation for **"CHK-K"**

Message character	C	(+)	O	(+)	D	(+)	E	<SP>	9	3	<SP>	(+)	E	(+)	X	**6**	
Reference number	12	46	24	46	13	46	14	38	9	3	38	46	14	46	33	6	
Weight factor	*1	*15	*14	*13	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	
Product -> Sum	12	+ 690	+ 336	+ 598	+ 156	+ 506	+ 140	+ 342	+72	+21	+228	+230	+56	+138	+66	+6	= 3597

-->
#CHK-K = 3597 = 76 x <u>47</u> + **25** --> CHK-K = #25 = **"P"**

5.12 Symbology: **Codabar**

5.12.1 Overview

This symbology is known as:
- Codabar, CodaBar, USD-4, NW-7, Code 2 of 7, Monarch, Code-27, Ames code, etc.

Usage

Application:	medical / pharmaceutical, libraries, blood banks (to be replaced by ISBT128), air parcel, courier services, photography (photo print industry), shipping; etc.
Popularity:	This is an older code, it has been in existence since 1972.

Remarks

Characteristics: (+) high tolerance (very easy to read);

Example

This is a sample of a typical barcode of this symbology.
This one has <startA> and <stopA>.

0815

Further information

The specification of this symbology is maintained at:
- USA: (ANSI/AIM BC3-1995 (USS)), at AIM USA
- Europe: EN 798, by AIM Deutschland

Other sources of information can be found at Appendix L.

Applications

Applications based on this symbology are:
- AIM Codabar (*)
- FedEx (using <start B> + <data (16)> + stop D>)
Those marked (*) are directly supported by this product; they are explained in separate sections right after this symbology.

5.12.2 Character set (Alphabet)

Message character set

The message character set consists of **20** = **16** (message) **+ 4** (control) characters.

Message data characters
- 10 numeric {0,...,9}
- 6 special chars { - | : | . | $ | / | + }

Message control characters

There are 4 different start/stop characters { A | B | C | D } . They can be used in any combination.
Since the start/stop characters actually encode information, they are considered part of the message alphabet.

Control character	Function
<startX>	specifies the start of the barcode (X = A,B,C,D)
<stopY>	specifies the stop of the barcode (Y = A,B,C,D)

Note:
These control characters have to be actually specified as part of the message, in order to specify which start and stop character should be used, unless they are fixed by some convention. In the message they are usually written as "A", "B", "C", "D", if they need to be specified.

Message length restrictions

The message length is variable and not limited.

Symbology character set

The symbology character set consists of **20** = **16** (message data) **+ 4** (control) characters.

Symbology data characters
There is a symbology character for every message data character. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start/stopA]	start and stop of barcode
[start/stopB]	dto.
[start/stopC]	dto.
[start/stopD]	dto.

5.12.3 Reference numbers

This table assigns a unique reference number to each message character.

Char.	Ref. #	Char.	Ref. #	Char.	Ref. #	Char.	Ref. #	Char.	Ref. #
0	#0	4	#4	8	#8	:	#12	[start/stopA]	#16
1	#1	5	#5	9	#9	/	#13	[start/stopB]	#17
2	#2	6	#6	–	#10	.	#14	[start/stopC]	#18
3	#3	7	#7	$	#11	+	#15	[start/stopD]	#19

The reference number of each message digit is the value of the digit itself.
Here both the start and stop characters need a reference number.

5.12.4 Checksum methods

The Codabar specification does not require any check characters nor define any checksum method.

A derived specification called "**AIM Codabar**", however, defines the following mod 16 check character.

CHK Formula

1 mandatory/optional
- mandatory for AIM Codabar
- optional for standard Codabar

> **#CHK = - ... [*1] x #M mod 16**

CHK Algorithm
Add the reference numbers of all characters, including <startX> and <stopY>.
The check character is the character whose reference number is the difference to the next multiple of 16.

Example (CHK calculation)

M = "<startA> 0 1 2 3 4 <stopB>"
-->
CHK = 16 + 0 + 1 + 2 + 3 + 4 + 17 = 43 = 3 * <u>16</u> - **5**
-->
M' = "<startA> 0 1 2 3 4 **5** <stopB>"

5.12.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces)
Dimensionality:	1-D
(Continuous / Discrete):	Discrete (inter-character space contains no information)

Logical encoding type:	
# Modules per element:	2-level = N, W
# Elements per character:	**7 el** = 4 b + 3 s
# Modules per character:	variable (!): 9 md = 5 eN + 2 eW , or 10 md = 4 eN + 3 eW
Self-checking:	yes
Bi-directional:	yes

5.12.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.12.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

N:W ratio	1:2.0 .. 1:3.0; recommended 1:2.25;

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.12.8 Code construction: Steps

The message data M = "<startX> <c#1>...<c#n> <stopY>" (where X,Y are members of A,B,C,D) is converted into the symbology character set according to the following scheme.

> **[start/stopX] [<c#1>] ... [<c#n>] [[<CHK>]] [start/stopY]**

1. Apply check characters

If required, calculate the CHK according to the algorithm above and insert it at the end of the message, before the <stopY>.
Note that the <startX> and <stopY> are considered to be part of the message.

2. Convert the message characters into symbology characters

Convert each message character <c#i> into the symbology character [<c#i>].
Convert the <startX> into the symbology character [start/stopX].
Convert the <stopY> into the symbology character [start/stopY].

5.12.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.12.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data

		<startA> 0 1 2 3 4 <stopB>
		-- (*)
		<startA> 0 1 2 3 4 **5** <stopB>
		[start/stopA] [0] [1] [2] [3] [4] [5] [start/stopB]
		-- here: for **N**ormal CTX style
		A 0 1 2 3 4 5 B

(*) For the calculation of the CHK see above.

5.13 Symbology: **2 of 5 Interleaved**

5.13.1 Overview

This symbology is known as:
- 2 of 5 Interleaved
- Interleaved 2 of 5
- 2/5 Interleaved
- I-2/5
- ITF

Usage
Application:	industrial, air cargo, hospitals, courier services, shipping industry
Regional use:	world-wide
Popularity:	very common; e.g. used by the German Post (Deutsche Post AG), and the United States Postal Services (USPS)

Remarks
Characteristics: (+) little space / extremely compact, (-) low tolerance;

Example
This is a sample of a typical barcode of this symbology.

Further information

The specification of this symbology is maintained at:
- USA: AIM/USS-I2/5, ANSI/AIM BC2-1995 (USS)
- Europe: EN 801 by AIM

Other sources of information can be found at Appendix L.

Applications

Applications based on this symbology are:
- Deutsche Post AG Ident-/Leitcode (*)
- USPS 25 Tray/Sack label (*)
Those marked (*) are directly supported by this product; they are explained in separate sections right after this symbology.

5.13.2 Character set (Alphabet)

Message character set

The message character set consists of **10** characters.

Message data characters
Numeric only {0,...,9}.

Message control characters
None.

Message length restrictions
The message length is variable and not limited.
Since the digits are encoded in pairs, the number of digits must be even, otherwise the message may be padded with a leading 0.
If a CHK needs to be added, the number of digits needs to be odd.

Symbology character set

The symbology character set consists of **102 = 10x10** (message) **+ 2** (control) characters.

Symbology data characters

There is a symbology character for every pair of message digits. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start]	start of barcode
[stop]	stop of barcode

5.13.3 Reference numbers

The reference number of each message digit is the value of the digit itself.
There is no reference number assigned to the [start] and [stop] characters.

5.13.4 Checksum methods

This symbology defines 1 optional mod 10 check character, weighted with alternating factors *3 and *1.
The algorithm is the same as for 2 of 5 Industrial.

CHK Formula

1 optional CHK

$$
\begin{aligned}
\text{\#CHK} &= - \ldots [\text{*1, *3}] \times \text{\#M mod 10} \\
&= - [\text{*3, *1, ..., *1, *3}] \times \text{\#M mod 10} \qquad\qquad \text{-- since length is odd} \\
&= - \{ (\text{<d\#1> + <d\#3> + ...}) \text{ *3} + (\text{<d\#2> + <d\#4> + ...}) \text{ *1} \} \text{ mod 10}
\end{aligned}
$$

CHK Algorithm

1. Sum all values of the odd-positioned digits (beginning with the 1st digit).
2. Multiply the result by 3.
3. Add the values of the remaining (even-positioned) digits.
4. The check digit is the difference to the next multiple of 10.

Example (CHK calculation)

M = "1 2 3 4 5 6 9"
-->
CHK = - { (1 + 3 + 5 + 9) *3 + (2 + 4 + 6) *1 } mod 10 = - 66 mod 10 = - 7 x <u>10</u> + **4**

5.13.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces) **"Interleaved":** In order to achieve the compactness, the symbology interleaves pairs of digits. The 1st digit in a pair is encoded in the widths of the 5 bars. The 2nd digit in a pair is encoded in the widths of the 5 spaces.
Dimensionality:	1-D
(Continuous / Discrete):	Continuous (inter-character space contains information)
Logical encoding type:	"2 of 5" = 2 out of 5 elements are wide
# Modules per element:	2-level = N, W
# Elements per character:	per digit pair: 10 el = 5 b + 5 s per digit : 5 el = 3 eN + 2 eW
# Modules per character:	per digit pair: 14 md per digit: 7 md = 3 eN + 2 eW
Self-checking:	yes

Bi-directional:	yes

5.13.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.13.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.13.8 Code construction: Steps

The message data M = "<c#1>...<c#n>" is converted into the symbology character set according to the following scheme.

---

if without CHK (n even)
  [start] [<c#1><c#2>] [<c#3><c#4>] ... [<c#n-1><c#n>] [stop]

if with CHK (n odd)
  [start] [<c#1><c#2>] [<c#3><c#4>] ... [<c#n><CHK>] [stop]

---

1. Apply check characters

If required, calculate the CHK according to the algorithm above and append it to the message.
Note that now the number of digits should be even.

2. Convert the message characters into symbology characters

Starting from the left, convert each pair of message digits "<d#i> <d#i+1>" into the symbology character [<d#i> <d#i+1>].

3. Insert symbology control characters

Prepend the [start] character and append the [stop] character.

5.13.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.13.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		1 2 3 4 5 6 9
		-- (*)
		1 2 3 4 5 6 9 **4**
		[start] [12] [34] [56] [94] [stop]

(*) For the calculation of the CHK see above.

5.14 Application: Deutsche Post AG (German Postal) Leitcode and Identcode

5.14.1 Overview

This application is known under these names:
- German Postal 2 of 5 Leitcode
- German Postal 2 of 5 Identcode
- Deutsche Post AG Leit-/Identcode
- Deutsche Frachtpost Leit-/Identcode

There are two codes: the Leitcode and the Identcode.
The **Leitcode** gives an indication of the destination.
The **Identcode** contains a tracking number providing an identification of the customer (sender) and the mailpiece.

Usage
Application: parcel shipments
Regional use: German Post (Deutsche Post AG) (Deutsche Frachtpost)

Example
This is a sample of a typical barcode of this application.

Further information

The specification of this application is maintained at:
 Deutsche Post AG
 Generaldirektion, Dienststelle 221b-1
 D-64276 Darmstadt
 Germany

 Phone: +49-(0)6151-9084735
 WWW: www.deutschepost.de

Other sources of information can be found at Appendix L.

Base symbology

2 of 5 Interleaved (without CHK)

5.14.2 Character set

Message character set
Same as the base symbology.

Message length restrictions
The message length is fixed:
- Leitcode: = 13 digits + 1 CHK
- Identcode: = 11 digits + 1 CHK

Other message data restrictions

None.

<u>**Semantics**</u> (Interpretation of message characters)

Leitcode:	(13)	
- digits 1..5	= (5)	Postal code (Postleitzahl, PLZ)
- digits 6..8	= (3)	Street ID/number
- digits 9..11	= (3)	House number
- digits 12..13	= (2)	Product code

Identcode:	(11)	
- digits 1..2	= (2)	ID of primary distribution center
- digits 3..5	= (3)	Customer ID
- digits 6..11	= (6)	Mailing number

5.14.3 Encoding (0): (Application level)

An **application message** is converted to a **symbology message** according to the following scheme.

There is no application-specific encoding of the application message text into the symbology message character set.

5.14.4 Checksum methods

This application defines 1 mandatory mod 10 check digit, weighted with alternating factors *4 and *9.
The CHK of the base symbology is not used.

<u>**CHK Formula**</u>

1 mandatory (proprietary) CHK

> **#CHK = - ... [*9, *4] x #M mod 10**   -- since message length odd
>   = - [*4, *9, ..., *9, *4] x #M mod 10
>   = - { (<d#1> + <d#3> + ...) *4 + (<d#2> + <d#4> + ...) *9 } mod 10

<u>**CHK Algorithm**</u>
1. Sum the values of the odd-positioned digits.
   Multiply the result by 4.
2. Sum the values of the even-positioned digits.
   Multiply the result by 9.
3. Add the two results.
   The check digit is the difference to the next multiple of 10.

<u>**Example (CHK calculation)**</u> (here: Identcode)

M = "9 8 7 6 5 4 3 2 1 0 1"
-->
CHK = - { (9+7+5+3+1+1) *4 + (8+6+4+2+0) *9 } = - 284 = - 29 x <u>10</u> + **6**

5.14.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

<u>**Measurements**</u>

Please refer to the official specification.

<u>**Imaging conventions**</u>

In the clear text the digits are grouped with dots and spaces according to their interpretation.
The CHK is shown in parentheses or in a smaller font size.

<u>Leitcode</u>	21348.075.016.40 (1)
<u>Identcode</u>	98.765 432.101 (6)

Note that this cannot be achieved with a barcode font. Instead, a non-CTX barcode font should be used, and the CTX has to be drawn as separate text underneath the barcode using a normal text font.

5.14.6 Code construction: Steps

The application message data M = "<d#1> ... <d#11/13>" is converted to the symbology message character set as follows.

<d#1> ... <d#11/13> <CHK>

1. Apply application-specific check characters
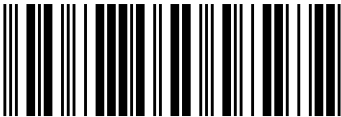
Calculate the CHK according to the algorithm above and append it to the message.

2. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the 2 of 5 Interleaved symbology.
The symbology-specific CHK method must NOT be applied.

5.14.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction (here: Identcode).

		Data
		9 8 7 6 5 4 3 2 1 0 1
		-- (*)
		9 8 7 6 5 4 3 2 1 0 1 **6**
		[start] [98] [76] [54] [32] [10] [16] [stop]

(*) For the calculation of the CHK see above.

5.15 Application: USPS 25 Tray label and Sack label

5.15.1 Overview

This application is known under these names:
- USPS 25 Tray/Sack Label

There are two codes: the **Tray label** and the **Sack label**.
Both contain the "ZIP code" of the receiver and a "Content Identifier Number".

Usage
Application: labeling of postal trays and sacks; for automation rate mailings; for periodicals and standard mail (letter size & flat size pieces);
Regional use: United States Postal Services (USPS) only; - since July 1997

Example
This is a sample of a typical barcode of this application.

Tray label

2230034401

Sack label

55116572

Further information

The specification of this application is maintained at:
 US Postal Services
See http://pe.usps.gov.

Other sources of information can be found at Appendix L.

Base symbology

2 of 5 Interleaved (without CHK)

5.15.2 Character set

Message character set

Same as the base symbology.

Message length restrictions
The message length is fixed:
- Tray label: = 10 digits
- Sack label: = 8 digits

Other message data restrictions
None.

Semantics (Interpretation of message characters)

The "**Tray label**" has 10 message digits:
- digits 1..5   = (5)   "ZIP code"
- digits 6..8   = (3)   "Content Identifier Number" [CIN]
- digits 9..10   = (2)   "Processing Code"

The "**Sack label**" has 8 message digits:
- digits 1..5   = (5)   "ZIP code"
- digits 6..8   = (3)   "Content Identifier Number" [CIN]

The **"Content Identifier Number" [CIN]** indicates this information:
- mailing class (express, priority, first-class, periodicals, standard, package)
- presorting / automation process
- sack/tray

The **"Processing code"** indicates this information:
- 07 = other

5.15.3 Encoding (0): (Application level)

An **application message** is converted to a **symbology message** according to the following scheme.

There is no application-specific encoding of the application message text into the symbology message character set.

5.15.4 Checksum methods

This application does not require any check characters.

5.15.5 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Quiet zone	= 10X
Height	= 0.65" - 0.75" (Sack: min.= 0.7")
Module width X	= 0.012" - 0.016" = 12..16 mil (optimal = 0.015")
N:W ratio	= 1:2.3..3.0 (optimal = 1:3.0)

Imaging conventions

For the conventions of the physical layout and the clear text, please refer to the official specifications.

5.15.6 Code construction: Steps

1. Application-specific conversions

There is no specific conversion of the application message data M = "<d#1> ... <d#n>" into the symbology message character set.
No CHK is used.

2. Apply the Code construction steps of the base symbology

Apply the Code construction steps of the 2 of 5 Interleaved symbology.
The symbology-specific CHK method must NOT be applied.

5.15.7 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction (here: Tray label).

		Data
		2 2 3 0 0 3 4 4 0 1

		-- (*)
		[start] [22] [30] [03] [44] [01] **[stop]**
		2230034401

5.16 Symbology: **2 of 5 Industrial**

5.16.1 Overview

This symbology is known as:
- 2 of 5 Industrial, Industrial 2 of 5, 2/5 Industrial
- 2 of 5 Standard, Standard 2 of 5, 2/5 Standard
- Normal 2/5
- Code 2/5, 2 of 5

Usage

Application:	industrial
Popularity:	(older code)

Remarks
Characteristics: (+) high tolerance,

Example
This is a sample of a typical barcode of this symbology.

Further information

Other sources of information can be found at Appendix L.

Applications

There are no special applications based on this symbology.

5.16.2 Character set (Alphabet)

Message character set

The message character set consists of **10** characters.

Message data characters
Numeric only {0,...,9}.

Message control characters
None.

Message length restrictions
The message length is variable and not limited.

Symbology character set

The symbology character set consists of **12** = **10** (message) **+ 2** (control) characters.

Symbology data characters
There is a symbology character for every message digit. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start]	left-hand side / start of barcode
[stop]	right-hand side / stop of barcode

5.16.3 Reference numbers

The reference number of each message digit is the value of the digit itself.
There is no reference number assigned to the [start] and [stop] characters.

5.16.4 Checksum methods

This symbology defines 1 optional mod 10 check character, weighted with alternating factors *3 and *1.

CHK Formula

1 optional CHK

> **#CHK = - ... [*1, *3] x #M mod 10**
>   = - { (... + <d#n-2> + <d#n>) *3 + (... + <d#n-3> + <d#n-1>) *1 } mod 10

CHK Algorithm

1. Sum all digits from right to left of the odd-positioned digits (beginning with the last digit).
   Multiply the result by 3.
2. Add the remaining (even-positioned) digits.
3. The check digit is the difference to the next multiple of 10.

Example (CHK calculation)

M = "7 8 9 0 1 2"
-->
CHK = - { (8 + 0 + 2) *3 + (7 + 9 + 1) *1 } = - 47 = - 5 x <u>10</u> + **3**

5.16.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	"bar width encoding" = no information in spaces (all narrow)
Dimensionality:	1-D
(Continuous / Discrete):	Discrete (inter-character space contains no information)
Logical encoding type:	"2 of 5" = 2 out of 5 bars are wide
# Modules per element:	2-level: b: N, W s: N -- (no info in spaces)
# Elements per character:	**9 el** = 5 b + 4 s
# Modules per character:	**11 md** = 7 eN + 2 eW = = 3 bN + 2 bW + 4 sN -- 5 bars (2 wide, 3 narrow) = 7 mb + 4 ms
Self-checking:	yes
Bi-directional:	yes

5.16.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.16.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

N:W ratio	1:2.0 .. 3.0

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.16.8 Code construction: Steps

The message data M = "<d#1>...<d#n>" is converted into the symbology character set according to the following scheme.

[start] [<d#1>] ... [<d#n>] [<CHK>] [stop]

---

1. Apply check characters

If required, calculate the CHK according to the algorithm above and append it to the message.

2. Convert the message characters into symbology characters

Convert each message digit <d#i> into the symbology character [<d#i>].

3. Insert symbology control characters

Prepend the [start] character and append the [stop] character.

5.16.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character. Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.16.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		7 8 9 0 1 2
		-- (*)
		7 8 9 0 1 2 **3**
		[start] [7] [8] [9] [0] [1] [2] [3] **[stop]**
		789012

(*) For the calculation of the CHK see above.

5.17 Symbology: **2 of 5 Matrix**

5.17.1 Overview

This symbology is known as:
- 2 of 5 Matrix
- 2/5 Matrix
- Matrix 2 of 5

Usage
 Application: industrial

Remarks
Characteristics: (+) high tolerance; (-) not self-checking;

Example
This is a sample of a typical barcode of this symbology.

Further information

Other sources of information can be found at Appendix L.

Applications

There are no special applications based on this symbology.

5.17.2 Character set (Alphabet)

Message character set

The message character set consists of **10** characters.

Message data characters
Numeric only {0,...,9}.

Message control characters
None.

Message length restrictions
The message length is variable and not limited.

Symbology character set

The symbology character set consists of **11** = **10** (message) **+ 1** (control) characters.

Symbology data characters
There is a symbology character for every message digit. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start/stop]	start and stop of barcode

5.17.3 Reference numbers

The reference number of each message digit is the value of the digit itself.
There is no reference number assigned to the [start] and [stop] characters.

5.17.4 Checksum methods

This symbology defines 1 optional mod 10 check character, weighted with alternating factors *3 and *1.

CHK Formula

1 optional CHK

> **#CHK = - ... [*1, *3] x #M mod 10**
>   = - { (... + <d#n-2> + <d#n>) *3 + (... + <d#n-3> + <d#n-1>) *1 } mod 10

CHK Algorithm
Same as for "2 of 5 Industrial".

Example (CHK calculation)
See the Example for 2 of 5 Industrial.

5.17.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces)	
Dimensionality:	1-D	
(Continuous / Discrete):	Discrete (inter-character space contains no information)	
Logical encoding type:	"2 of 5" = 2 elements out of 5 are wide	
# Modules per element:	2-level = N, W; 3 (only [start/stop])	
# Elements per character:	**5 el** = 3 b + 2 s	
# Modules per character:	**7 md** = 3 eN + 2 eW -- except [start/stop]	
Self-checking:	no ! (needs CHK)	
Bi-directional:	yes	

5.17.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.17.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.17.8 Code construction: Steps

The message data M = "<d#1>...<d#n>" is converted into the symbology character set according to the following scheme.

[start/stop] [<d#1>] ... [<d#n>] [<CHK>] [start/stop]

---

1. Apply check characters

If required, calculate the CHK according to the algorithm above and append it to the message.

2. Convert the message characters into symbology characters

Convert each message digit <d#i> into the symbology character [<d#i>].

3. Insert symbology control characters

Prepend and append the [start/stop] character.

5.17.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.17.10　　　Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		7 8 9 0 1 2
		-- (*)
		7 8 9 0 1 2 **3**
		[start/stop] [7] [8] [9] [0] [1] [2] [3] [start/stop]
		 789012

(*) For the calculation of the CHK see above.

5.18 Symbology: **Code 11**

5.18.1 Overview

This symbology is known as:
- Code 11
- USD-8

Usage

 Application: labeling of telecommunications equipment

Remarks

Characteristics: (-) low tolerance; discrete, high-density;

Example

This is a sample of a typical barcode of this symbology.

Further information

Other sources of information can be found at Appendix L.

Applications

There are no special applications based on this symbology.

5.18.2 Character set (Alphabet)

Message character set

The message character set consists of **11** characters.

Message data characters

{ 0, ..., 9, - (dash) }

Message control characters

None.

Message length restrictions

The message length is variable.

Symbology character set

The symbology character set consists of **12** = **11** (message) **+ 1** (control) characters.

Symbology data characters

There is a symbology character for every message digit. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start/stop]	start/start of barcode

5.18.3 Reference numbers

The reference number of each message digit is the value of the digit itself.
The "-" has reference number 10.
There is no reference number assigned to the [start/stop] character.

5.18.4 Checksum methods

This symbology defines several 2 different checksum methods for 1 or 2 check digits, which are calculated by a special "mod 11" algorithm.
- 1 check digit "C"
- 2 check digits "CK"
For messages with less than 10 characters, only 1 CHK is used. For more than 10 characters, 2 CHK should be used.

CHK Formula

1 or 2 optional CHK: "[mod11] mod11"

C = + ... [*10, *9, ..., *2, *1] x #M mod 11
K = + ... [*9, *8, ..., *2, *1] x #MC mod 11

5.18.4.1　Method 1: ("mod11")

CHK Algorithm
Calculate C and append it to the message.

5.18.4.2　Method 2: ("mod11 mod11")

CHK Algorithm
1. Calculate the 1st check digit C according to the first "mod 11" algorithm.
2. To calculate the 2nd check digit K, append the 1st check digit C to the original message, then apply the second "mod 11" algorithm to it.

Example (CHK calculation)

M = "1 2 3 - 5 6 7 8 9 4 1" --> C = **5** --> K = **0** --> M" = "1 2 3 - 5 6 7 8 9 4 1 **5 0**"

	<#1>	<#2>	<#3>	<#4>	<#5>	<#6>	<#7>	<#8>	<#9>	<#10>	<#11>	C	CHK =>
Message digit M = (Reference number)	1	2	3	-	5	6	7	8	9	4	1		
Weight factor	1	10	9	8	7	6	5	4	3	2	1		
Product -> Sum	1	+ 20	+ 27	+ 80	+ 35	+ 36	+ 35	+32	+27	+8	+1		= 302 = 27 x 11 + **5**
M' =	1	2	3	-	5	6	7	8	9	4	1	**5**	
Weight factor	3	2	1	9	8	7	6	5	4	3	2	1	
Product -> Sum	3	+4	+3	+90	+40	+42	+42	+40	+36	+12	+2	+5	= 319 = 29 x 11 + **0**
M" =	1	2	3	-	5	6	7	8	9	4	1	**5**	**0**

5.18.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces)
Dimensionality:	1-D
(Continuous / Discrete):	Discrete (inter-character space contains no information)
Logical encoding type:	
# Modules per element:	2-level = {1,2}
# Elements per character:	**5 el** = 3 b + 2 s
# Modules per character:	**7 md** = 3 eN + 2 eW **6 md** = 4 eN + 1 eW
Self-checking:	no ! (needs CHK)
Bi-directional:	yes

5.18.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.18.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.18.8 Code construction: Steps

The message data M = "<d#1>...<d#n>" is converted into the symbology character set according to the following scheme.

[start/stop] [<d#1>] ... [<d#n>] [<CHK>] [[<CHK>]] [start/stop]

---

1. Apply check characters

According ot the CHK method required, calculate the CHK according to the algorithm above and append it/them to the message.

2. Convert the message characters into symbology characters

Convert each message digit <d#i> into the symbology character [<d#i>].

3. Insert symbology control characters

Prepend and append the [start/stop] character.

5.18.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.19 Symbology: **MSI**

5.19.1 Overview

This symbology is known as:
- MSI
- MSI/Plessey
- Modified Plessey

Note:
This code is a variation of the Plessey code used in the US. It is different from the original "Plessey" code, which is also named "UK Plessey".

Usage
Application:	grocery store shelf tags, for inventory control;
Popularity:	This is an older code, it has been in existence since 1970.

Remarks
Characteristics: (+) high tolerance, (-) takes a lot of space, (-) not self-checking;
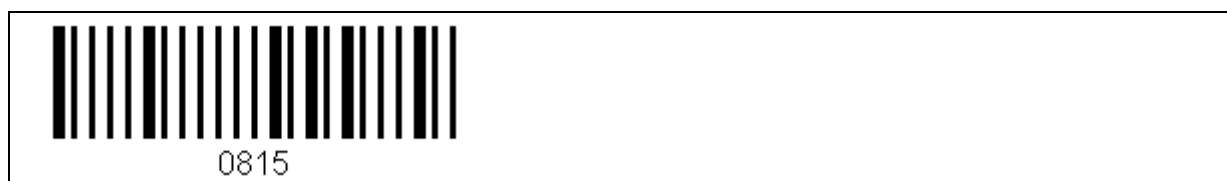
Example
This is a sample of a typical barcode of this symbology.

Further information

Other sources of information can be found at Appendix L.

Applications

There are no special applications based on this symbology.

5.19.2 Character set (Alphabet)

Message character set

The message character set consists of **10** characters.

Message data characters
Numeric only {0,...,9}.

Message control characters
None.

Message length restrictions
The message length is variable, but limited to **13 digits**.

Symbology character set

The symbology character set consists of **12** = **10** (message) **+ 2** (control) characters.

Symbology data characters
There is a symbology character for every message digit. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start]	start of barcode

[stop]	stop of barcode

5.19.3 Reference numbers

The reference number of each message digit is the value of the digit itself.
There is no reference number assigned to the [start] and [stop] characters.

5.19.4 Checksum methods

This symbology defines several different checksum methods for 1 or 2 optional check digits, which are calculated by a combination of two special "mod 10" and "mod 11" algorithms.
- no check digit
- 1 check digit: "mod 10"
- 2 check digits: "mod 10" and "mod 10"
- 2 check digits: "mod 11" and "mod 10"

<u>CHK Formula</u>

1 or 2 optional CHK: "[mod11|mod10] mod10"

<u>CHK Algorithm</u>
See below.

<u>Example (CHK calculation)</u>
See below.

5.19.4.1 The "mod 10" algorithm

<u>CHK Formula</u>

```
N1    = [..., *100, *0, *10, *0, *1] x #M
N2    = ... [*1, *0] x #M + [*1] x (2 * N1)
#CHK = - N2 mod 10
```

<u>CHK Algorithm</u>

1. From the message number, generate a new number consisting of every other digit, starting with the rightmost digit.
2. Multiply this number by 2. Then sum all its digits.
3. Add all remaining message digits (those not used in step 1).
4. The check digit is the difference to the next multiple of 10.

<u>Example (CHK calculation)</u>

M = "1 2 3 4 5 6"
-->
N1 = 246 --> 2 x N1 = 492
N2 = (1+3+5) + (4+9+2) = 24 = 3 x <u>10</u> - **6**
CHK = **6**

5.19.4.2 The "mod 11" algorithm

<u>CHK Formula</u>

```
#CHK = - ... [*7, *6, *5, *4, *3, *2] x #M mod 11
```

If #CHK = 10, the CHK digit is to be omitted !

<u>CHK Algorithm</u>

1. From right to left multiply all digits by their position plus 1, i.e. beginning with factor 2;

reset the factor to 2 after the factor 7.
I.e. the weights are (...,7,6,...,3,2,7,6,...,3,2).
2. The check digit is the difference to the next multiple of 11.

Example (CHK calculation)

M = "1 2 3 4 5 6"
-->

		<d#1>	<d#2>	<d#3>	<d#4>	<d#5>	<d#6>		CHK =>
Message digit (Reference number)		1	2	3	4	5	6		
Weight factor		7	6	5	4	3	2		
Product -> Sum		7	+ 12	+ 15	+ 16	+ 15	+ 12		= 77 = 7 x 11 - **0**

-->
CHK = **0**

5.19.4.3 Method 1: ("mod10")

CHK Algorithm
Apply the "mod 10" algorithm above.

Example (CHK calculation)

M = "1 2 3 4 5 6"
--> (see above)
M'(mod10) = "1 2 3 4 5 6 **6**"

5.19.4.4 Method 2: ("mod10 mod10")

CHK Algorithm
1. Calculate the 1st check digit according to the "mod 10" algorithm.
2. To calculate the 2nd check digit, append the 1st check digit to the original message, then apply the "mod 10" algorithm to it.

Example (CHK calculation)

M = "1 2 3 4 5 6"
--> (see above)
M'(mod10) = "1 2 3 4 5 6 **6**"
-->
N1 = 1356 --> 2 x N1 = 2712
N2 = (2+4+6) + (2+7+1+2) = 24 = 3 x 10 - **6**
CHK = **6**
-->
M''(mod10 mod10) = "1 2 3 4 5 6 **6 6**"

5.19.4.5 Method 3: ("mod11 mod10")

CHK Algorithm

1. Calculate the 1st check digit according to the "mod 11" algorithm.
2. To calculate the 2nd check digit, append the 1st check digit to the original message, then apply the "mod 10" algorithm to it.

Example (CHK calculation)

M = "1 2 3 4 5 6"
--> (see above)
M'(mod11) = "1 2 3 4 5 6 **0**"
-->
N1 = 1350 --> 2 x N1 = 2700
N2 = (2+4+6) + (2+7+0+0) = 21 = 3 x 10 - **9**
CHK = **9**
-->
M''(mod11 mod10) = "1 2 3 4 5 6 **0 9**"

5.19.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces)
Dimensionality:	1-D
(Continuous / Discrete):	Continuous (inter-character space contains information)
Logical encoding type:	"pulse width modulated":
	Each digit as 4-bit binary representation. Then
	- bit "1" --> "bW sN" -- wide bar + narrow space
	- bit "0" --> "bN sW" -- narrow bar + wide space
# Modules per element:	2-level = {1,2}
# Elements per character:	**8 el** = 4 b + 4 s
# Modules per character:	**12 md** = 4 eN + 4 eW
Self-checking:	no ! (needs CHK)
Bi-directional:	yes

5.19.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.19.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Please refer to the official specification.

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.19.8 Code construction: Steps

The message data M = "<d#1>...<d#n>" is converted into the symbology character set according to the following scheme.

> **[start] [<d#1>] ... [<d#n>] [[<CHK>]] [[<CHK>]] [stop]**

1. Apply check characters

According ot the CHK method required, calculate the CHK according to the algorithm above and append it/them to the message.

2. Convert the message characters into symbology characters

Convert each message digit <d#i> into the symbology character [<d#i>].

3. Insert symbology control characters

Prepend the [start] character and append the [stop] character.

5.19.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.19.10 Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		1 2 3 4 5 6
		-- (*)
		1 2 3 4 5 6 **0 9**
		[start] [1] [2] [3] [4] [5] [6] [0] [9] [stop]

(*) For the calculation of the CHK see above; here: CHK "mod11 mod10".

5.20 Application family: **USPS Postal codes**

See at:
- USPS 25 Tray label and Sack label
- USPS POSTNET
- USPS FIM and ZEBRA
- USPS IMb (Intelligent Mail barcode)

5.21 Symbology: **USPS POSTNET**

5.21.1 Overview

This symbology is known as:
- USPS POSTNET, POSTNET, USPS Postnet, PostNet, Postnet 3 of 5

Usage

　Application:　　　　USA postal code (ZIP code), for automatic mail sorting
　Regional use:　　　by United States Postal Service (USPS)
Superseded by "Intelligent Mail barcode" (IMb).

"**PostNET**" is an acronym of "**Post**al **N**umeric **E**ncoding **T**echnique".
If the PostNet code is preprinted on a mailpiece, a discount applies to the rate.

It provides information about the address of the receiver.

It comes in 3 different lengths.
POSTNET 5 encodes only the 5-digit **ZIP** code of the address. It is also known as "ZIP", or "POSTNET 5-digit ZIP code".
POSTNET 9 encodes the enhanced 9-digit **ZIP+4** code of the address. It is also known as "ZIP+4", or "POSTNET 9-digit ZIP+4 code".
POSTNET 11 also encodes the 2-digit **Delivery Point Code (DPC)**, which is usually the last 2 digits of the house # or the Post box #. It is also known as "DPC PostNet", "DPBC PostNet", or "POSTNET 11-digit **D**elivery **P**oint (**B**ar)**C**ode".
The **Delivery Point Code (DPC)** is usually the last 2 digits of the house # or the Post box #.

Remarks
Characteristics: fixed height and width;

Example
This is a sample of a typical barcode of this symbology.

Further information

The specification of this symbology is maintained at:
　United States Postal Services (USPS): www.usps.com; (document DMM C840)

Other sources of information can be found at .

Applications

There are no special applications based on this symbology.

5.21.2 Character set (Alphabet)

Message character set

The message character set consists of **10** characters.

Message data characters
Numeric only {0,...,9}.

Message control characters
None.

Message length restrictions

The message length is fixed, as follows:
- **PostNET 5**:   = 5 + 1 CHK   = ZIP code (5) + CHK (1)
- **PostNET 9**:   = 9 + 1 CHK   = ZIP+4 code (9) + CHK (1)
- **PostNET 11**:   = 11 + 1 CHK   = ZIP+4 code (9) + DPC (2) + CHK (1)

Symbology character set

The symbology character set consists of **11** = **10** (message) **+ 1** (control) characters.

Symbology data characters

There is a symbology character for every message digit. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start/stop]	"frame bar" = start and stop of barcode

5.21.3 Reference numbers

The reference number of each message digit is the value of the digit itself.
There is no reference number assigned to the [start] and [stop] characters.

5.21.4 Checksum methods

This symbology defines 1 mandatory mod 10 check digit, weighted with all factors = *1.

CHK Formula

1 mandatory CHK

#CHK = - ... [*1] x #M mod 10

CHK Algorithm

1. Sum the values of all digits.
2. The check digit is the difference to the next multiple of 10.

Example (CHK calculation)

M = "1 2 3 4 9"
-->
CHK = - {1+2+3+4+9} mod 10 = - 19 = - 2 x 10 + **1**

5.21.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	"2-state Bar height encoding"
	There is no information in the spaces or widths.
Dimensionality:	--
(Continuous / Discrete):	--
Logical encoding type:	
# Modules per element:	2 states of a bar: low (short) ("i") , high (long) ("I")
# Elements per character:	**5 b** = 2 high + 3 low
# Modules per character:	--
Self-checking:	no ! (needs CHK)
Bi-directional:	yes

5.21.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.21.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Height of full (tall) bar	= 0.115 .. 0.135 inch = 8.28 .. 9.72 pt
Height of half (short) bar	= 0.040 .. 0.060 inch = 2.88 .. 4.32 pt
Bar width	= 15 .. 25 mil
Space width	= 12 .. 40 mil
Total width	- PostNET 5 (32 bars) = 1.260 .. 1.625 inch - PostNET 9 (52 bars) = 2.090 .. 2.625 inch - PostNET 11 (62 bars) = 2.510 .. 3.125 inch

Imaging conventions

There are no specific conventions about the physical layout of the image and the clear text.

5.21.8 Code construction: Steps

The message data M = "<d#1>...<d#n>" is converted into the symbology character set according to the following scheme.

[start/stop] [<d#1>] ... [<d#n>] [<CHK>] [start/stop]

1. Apply check characters

Calculate the CHK according to the algorithm above and append it to the message.

2. Convert the message characters into symbology characters

Convert each message digit <d#i> into the symbology character [<d#i>].

3. Insert symbology control characters

Prepend and append the [start/stop] character.

5.21.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.21.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		1 2 3 4 9
		-- (*)
		1 2 3 4 9 **1**

		[start/stop] [1] [2] [3] [4] [9] [1] [start/stop]
		Iₙₐₗₗₐₗₗₐₗₗₐₗₗₐₗₗₗₐₗₗ

(*) For the calculation of the CHK see above.

5.22 Symbology: **USPS FIM**

5.22.1 Overview

This symbology is known as:
- USPS FIM

FIM stands for "**Facing Identification Mark**".

It is not actually a proper symbology.
It is a pattern of vertical bars printed in the upper right portion of the mailpiece, used as an orientation mark to identify certain classes of mail.

There are 5 patterns defined.
- **FIM-A** = for Courtesy Reply Mail (CRM) and Meter Reply Mail (MRM) with pre-printed ZIP+4 POSTNET
- **FIM-B** = for Business Reply Mail (BRM) (Penalty mail, Franked mail) without pre-printed POSTNET
- **FIM-C** = for Business Reply Mail (BRM) (Penalty mail, Franked mail) with pre-printed ZIP+4 POSTNET
- **FIM-D** = for OCR readable IBI (information based indicia) mail without pre-printed POSTNET
- **FIM-E** = for postcard-size and letter-size FirstClass Mail with customized services

Usage
Regional use: by United States Postal Service (USPS)

Example
This is a sample of a typical barcode of this symbology.
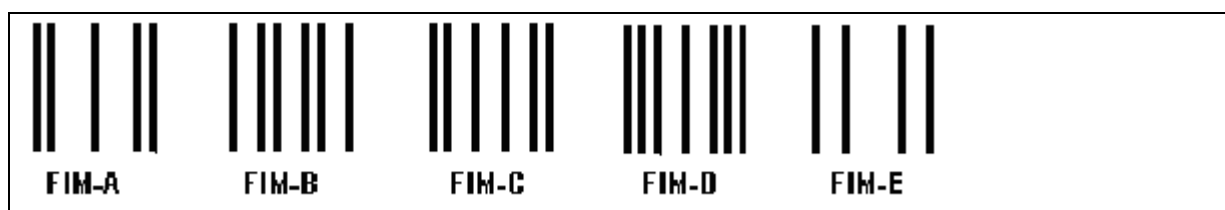
Further information

The specification((standardization)) of this symbology is maintained at:
  United States Postal Services (USPS): www.usps.com; (documents DMM C810 and C800)

Other sources of information can be found at Appendix L.

5.22.2 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	All bars and spaces are again separated by spaces.
Logical encoding type:	A symmetric pattern of '0' and '1'.
# Modules per character:	17

5.22.3 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.22.4 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Quiet zone	About the size of the quiet zones around the FIM, please refer to the official specification.

Height	= 0.50 .. 0.75 inch = 36 .. 54 pt
Width (each bar and space)	= 1/32" +/- 0.008" ~= 23.25 .. 39.25 mil
Total width	~= 0.52625 .. 0.53625 inch

5.23 Symbology: **USPS ZEBRA**

5.23.1 Overview

This symbology is known as:
- USPS ZEBRA

This is not a barcode; just a series of several diagonal or vertical black stripes/lines: "////" or "||||".

Usage
Application: Visual indication on a Tray label, next to the Tray label barcode, that a tray contains automation rate mailing.
Regional use: United States Postal Services (USPS) only; since July 1997;

Further information
Specification: See http://pe.usps.gov

Example
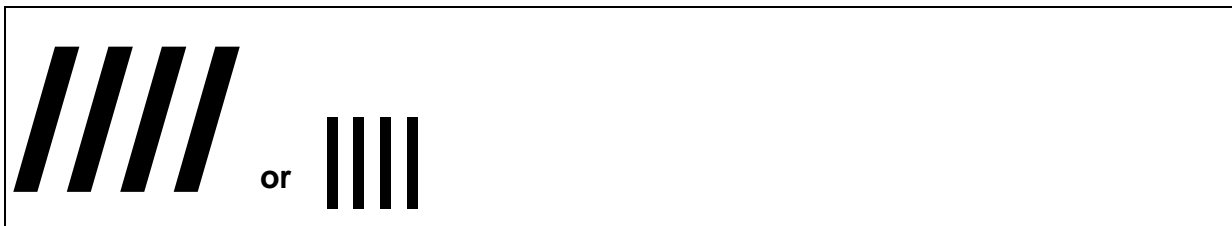
5.23.2 Physical attributes

This section specifies attributes related to the physical realization of the barcode.

Measurements

Height	= 0.25 .. 0.375"
Width (each mark and space)	= 0.125 .. 0.25" In a given symbol, all marks and spaces need to have the same width.

5.24 Symbology family: **The UPC/EAN family**

5.24.1 Overview

This symbology family is known under these names:
- UPC (Universal Product Code) : UPC-A/E
- EAN (European Article Number): EAN-x
- JAN (Japanese Article Number): JAN-x
- World code

Usage

Application:	retail product marking; article number
Regional use:	world-wide (UPC = USA/Canada, EAN/JAN = Europe, Japan, other countries)
Popularity:	omnipresent; exists since 1973

Remarks

Characteristics: (+) very little space; (-) small tolerances;

Example

For examples, please see at each member of this family below.

Further information

The specification of these symbologies is maintained at:
- Germany: DIN 66236 by DIN Germany
- Europe: EN 797 by EAN International

Other sources of information can be found at Appendix L.

Applications

There are no special applications based on this symbology.
There is the possibility of an optional supplemental 2- or 5-digit encoding (EAN/UPC-x+2/5).

5.24.2 Character set (Alphabet)

Message character set

The message character set consists of **10** characters.

Message data characters
Numeric only = {0,...,9}.

Message control characters
None.

Message length restrictions

The message length is fixed. See below at each symbology.

Symbology character set

The symbology character set consists of **35** = **3 x 10** (message) **+ 5** (control) characters.

Symbology data characters

There are 3 different symbology characters associated with every message digit.
See the table in the section Encoding (I) for the full list.

Thus, there are 3 different "pools" of symbology characters. For short, they are called pool A, B, and C.
A symbology character is denoted as
 [<digit>]:<pool>
For example: "[0]:B" means that the digit "0" is encoded in pool B.
The pool scheme is further explained below.

Symbology control characters

The symbology family has these 5 control characters.

Control character	Function
[start/stop]	Start and stop bars of main barcode for all symbologies.
[guard]	Intermediate guard bars of main barcode for all symbologies.
[stop UPC-E]	Special stop bars of main barcode for UPC-E symbology.
[start supp]	Start bars for 2- or 5-supplement.
[guard supp]	Guard bars for 2- or 5-supplement.

5.24.3 Reference numbers

The reference number of a message digit is the value of the digit itself.
There are no reference numbers assigned to the symbology control characters.

5.24.4 Checksum methods

The symbologies of this family define 1 mandatory mod 10 check digit, weighted with alternating factors *3 and *1.

CHK Formula

1 mandatory CHK

> **#CHK = - ... [*1,*3] x #M mod 10**
>   = - { (... + <d#n-2> + <d#n>) *3 + (... + <d#n-3> + <d#n-1>) *1 } mod 10

CHK Algorithm

1. Sum every other digit, starting with the right-most digit.
   Multiply the result by 3.
2. Add all other digits (those not used in step 1) to the result.
3. The check digit is the difference to the the next multiple of 10.

Example (CHK calculation)

M = "1 2 3 4 5 9"
-->
CHK = - { (2+4+9) *3 + (1+3+5) *1 } mod 10 = - 54 = - 6 x <u>10</u> + **6**

5.24.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	Linear (bars and spaces)
Dimensionality:	1-D
(Continuous / Discrete):	Continuous (inter-character space contains information)
Logical encoding type:	(see below)
# Modules per element:	4-level = 1, 2, 3, 4 md
# Elements per character:	**4 el** = 2 b + 2 s - left-hand = "sbsb" - right-hand = "bsbs"
# Modules per character:	**7 md** = (various) #mb is odd/even for pool A/B ("odd/even parity" encoding).
Self-checking:	no ! (needs CHK)
Bi-directional:	yes

The Pool scheme

The encoding scheme is quite complex:

Depending on the position of the digit within the message, it is encoded either as a "**left-hand**" or as a "**right-hand**" character.
Depending on the message context (e.g. the value of the 1st digit and/or the CHK digit) and the position of the digit within the message, a "left-hand" digit is encoded as either "**odd parity**" or "**even parity**". This means that in the binary encoding, the number of '1' is odd or even, respectively.
Thus, there are 3 different encodings for each digit. They are called "**pools**".

Using the following notation for the control characters:
- [start/stop]   = "("
- [guard]   = "-"
- [stop UPC-E]   = ")"
- [start supp]   = "#"
- [guard supp]   = "$"
the pool usage can be summarized as follows:

EAN-13:	**(A * * * * * - C C C C C <u>C</u> (**	-- where * = A/B depending on the value of <d#1>
EAN-8:	**(A A A A - C C C <u>C</u> (**	
UPC-A:	**(A A A A A A - C C C C C <u>C</u> (**	
UPC-E:	**(* * * * * *)**	-- where * = A/B depending on the value of <d#1> (= 0 or 1) and <CHK>
2-addon:	**# * $ ***	-- where * = A/B depending on the value of the <CHK>
5-addon:	**# * $ * $ * $ * $ ***	-- where * = A/B depending on the value of the <CHK>

For the pool to be used for each "*", please see the pool pattern information at the symbology in question below.

5.24.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.24.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

N:W ratio	1:2.0

Imaging conventions

Each symbology of this family has its own conventions about the physical layout of the image and the clear text. Please refer to the corresponding chapters below.

Taking into account the clear text conventions, and using the pools A',C',D,E,F, which are defined below, the pool usage looks like this. For the pool to be used for each "*", please see the pool pattern information at the symbology in question below.

```
              1       2  3  4  5  6  7     8  9  10 11 12 C
EAN-13:   D  (  A  *  *  *  *  *  -  C  C  C  C  C  C  (        -- * = A/B depends on the value of <d#1>

                 1  2  3  4     5  6  7  C
EAN-8:        (  A  A  A  A  -  C  C  C  C  (

              1     1  2  3  4  5  6     7  8  9  10 11 C     C
UPC-A:    D  (  A' A  A  A  A  A  -  C  C  C  C  C  C' (  D

              1     2  3  4  5  6  7     C
UPC-E:    D  (  *  *  *  *  *  *  )  D             -- * = A/B depends on the value of <d#1> (= 0 or 1)
                                                     and <CHK>

                 1     2
+2 addon:     #  *  $  *                           -- * = E/F depends on the value of the <CHK>

                 1     2     3     4     5
+5 addon:     #  *  $  *  $  *  $  *  $  *         -- * = E/F depends on the value of the <CHK>
```

5.24.8 Code construction: Steps

The conversion of the message data M = "<d#1>...<d#n>" into the symbology character set depends on the actual symbology of this family. Please see the corresponding chapter below.

The message characters are encoded in one of the pools A, B, C and are enclosed by a start and a stop character. An intermediate guard character may need to be used as well.

The following kind of scheme is used:

byte #	1	2	...	...	...	...	...	...
symb. char.	[start]	[<c#1>]	...	[guard]	...	[<c#n>]	[<CHK>]	[stop]
pool		A / B				C	C	

In general, the following sequence of steps is applied.

1. Apply check characters

A check digit is calculated according to the algorithm indicated in the section "Checksum methods".
(The result is either appended to the message, or used in determining the pattern of pools to be used.)

2. Convert the message characters into symbology characters

Each message digit <d#i> is converted into one or more symbology characters [<d#i>], using the pools indicated in Encoding (I), depending on the value of the 1st digit and/or the CHK digit.

3. Insert symbology control characters

The [start], [guard], and [stop] characters are inserted where indicated.

5.24.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.24.10   Code construction: The complete procedure (Example)

For a full example of all stages of the Code construction, please see the members of this family.

5.25 Symbology: **EAN-13**

5.25.1 Overview

This symbology is known as:
- EAN-13

Usage
Application: article numbers of retail products
Regional use: Europe/Japan; other countries
Popularity: This is probably the most commonly used barcode in the world. Whenever you purchase food, the barcode is on the product.

Remarks
This symbology can have an optional supplemental 2-digit or 5-digit encoding (for books and periodicals) (EAN-13+2, EAN-13+5).
This symbology is a superset of UPC-A.

Example
These are samples of typical barcodes of this symbology.

Further information

See the chapter "The UPC/EAN family" above.

The exact code specifications for EAN-13 and EAN-8 can be obtained from:
- Germany: Centrale für Coorganisation GmbH (CCG)

See: www.barcodeisland.com/ean13.phtml.

Applications

Applications based on this symbology are:

- "EAN 99" = EAN-13 with country code "99"

- **ISSN** ("International Standard Serial Number") = EAN-13 with country code "**977**" and an additional proprietary mod 11 CHK

- **ISBN** ("International Standard Book Number") = EAN-13 with country code "**978**" and an additional proprietary mod 11 CHK

- **ISMN** ("International Standard Music Number") = EAN-13 with country code "**979**" and an additional proprietary mod 11 CHK

Example: (ISBN)
- "**ISBN-13**" = "978-3-89864-405-**1**" corresponds to "**ISBN-10**" = "3-89864-405-**7**".

5.25.2 Character set (Alphabet)

Message character set

Refer to the corresponding information in the chapter for the symbology family above.

Message length restrictions
The message length is fixed to = **12 digits** + 1 CHK.

Other message data restrictions
None.

Semantics (Interpretation of message characters)

- Country code   (2 or 3)
- Supplier #   (6 or 5 or 4)
- Product/article #   (4 or 5)
- CHK   (1)

Country codes: (excerpt)

- 00..13	= US/CDN
- 30..37	= France
- 40..44	= Germany
- 45,49	= Japan
- 50	= UK
- 54	= Belgium, Luxemburg
- 560	= Portugal
- 57	= Denmark
- 590	= Poland
- 599	= Hungary
- 70	= Norway
- 73	= Sweden
- 76	= Switzerland
- 80..83	= Italy
- 84	= Spain
- 859	= Czechia
- 87	= Netherlands
- 90..91	= Austria

Symbology character set

Symbology data characters
Refer to the corresponding information in the chapter for the symbology family above.

Symbology control characters
Refer to the corresponding information in the chapter for the symbology family above.

5.25.3 Reference numbers

Refer to the corresponding information in the chapter for the symbology family above.

5.25.4 Checksum methods

Refer to the corresponding information in the chapter for the symbology family above.

The mod 10 CHK algorithm described there is applied to the message M = "<d#1>...<d#12>".

Example (CHK calculation)

M = "4 3 6 8 9 3 2 9 0 8 4 9"
-->
CHK = - { (3+8+3+9+8+9) *3 + (4+6+9+2+0+4) *1 } mod 10 = - 145 = - 15 x <u>10</u> + **5**

5.25.5 Encoding scheme

Refer to the corresponding information in the chapter for the symbology family above.

5.25.6 Encoding (I): The Logical encoding

For the logical encoding, refer to the chapter of the symbology family above.

The pool pattern for EAN-13 is:

	1		2	3	4	5	6	7		8	9	10	11	12	C	
D	**(**	**A**	*****	*****	*****	*****	*****	**-**	**C**	**C**	**C**	**C**	**C**	**C**	**(**

The first digit <d#1> is not part of the barcode itself.
The following 6 digits <d#2> .. <d#7> are coded using pools A and B, depending on the value of <d#1>. The pattern to be used is determined according to the following pool scheme.
The remaining digits and the CHK use pool C.

<d#1>	<d#2>	<d#3>	<d#4>	<d#5>	<d#6>	<d#7>
0 (UPC-A)	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Example:
If the 1st digit is "1", the pool pattern is: D(**AABABB**-CCCCCC(

5.25.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Height	The recommended height (incl. clear text) is 70 % of the overall width (incl. the quiet zone).
Width	30.50 .. 74.58 mm; see the table below
Module width X	10.6 .. 26.0 mil; see the table below
N:W ratio	1:2.0

Note: The overall width is 113 X = **95 X** (barcode) + 18 X (quiet zone).

Name of size	Scale factor	Module width [mm]	Module width [mil]	EAN-13 total width [mm]	EAN-13 height [mm]			
SC0	0.818	0.270	10.6	30.50	21.48			
SC1	0.90	0.297	11.7	33.56	23.63			
SC2 (*)	**1.00 (100%)**	**0.330**	**13.0**	**37.29**	**26.26**			
SC3	1.10	0.363	14.3	41.02	28.89			
SC4	1.20	0.396	15.6	44.75	31.51			
SC5	1.35	0.446	17.7	50.34	36.46			
SC6	1.50	0.495	19.5	55.94	39.39			
SC7	1.65	0.544	21.4	61.63	43.33			
SC8	1.85	0.610	24.0	68.99	48.58			
SC9	2.00	0.660	26.0	74.58	52.52			

The total width is including the quiet zone.
The height is including the clear text.
(*) SC2 is the nominal size.

Imaging conventions

Normally, EAN codes are printed with clear text underneath in a fully embedded style.
The 1st digit (which is not coded directly) is placed as clear text in front of the barcode.

The pool pattern for EAN-13 looks like this:

	1		2	3	4	5	6	7		8	9	10	11	12	C	
D	**(**	**A**	*****	*****	*****	*****	*****	**-**	**C**	**C**	**C**	**C**	**C**	**C**	**(**

5.25.8 Code construction: Steps

The message data M = "<d#1>...<d#12>" is converted into the symbology character set according to the following scheme.

byte #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
symb. char.	[d#1]	[start/ stop]	[d#2]	[d#3]	[d#4]	[d#5]	[d#6]	[d#7]	[guard]	[d#8]	[d#9]	[d#10]	[d#11]	[d#12]	[<CHK >]	[start/ stop]
pool	:D		:A	:A/B (*)	:A/B (*)	:A/B (*)	:A/B (*)	:A/B (*)		:C	:C	:C	:C	:C	:C	

Note: (*) The pools in which these characters are encoded depend on the value of <d#1>; see the table in section "Encoding (I)" above.

1. Apply check characters

Calculate the check digit according to the algorithm indicated above.
Append it to the message.

2. Convert the message characters into symbology characters

Depending on the value of the 1st digit <d#1>, determine the pool pattern to be used for digits <d#2> ... <d#7>, according to the table in section "Encoding (I)" above.
Convert the message digits <d#i> and the CHK into symbology characters [<d#i>], according to this pool pattern.

3. Insert symbology control characters
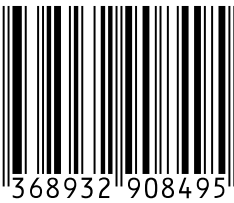
Insert the [start/stop] and [guard] characters where indicated.

5.25.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.25.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		4 3 6 8 9 3 2 9 0 8 4 9
		-- (*)
		4 3 6 8 9 3 2 9 0 8 4 9 **5**
		-- (**)
		[4]:D [start/stop] [3]:A [6]:B [8]:A [9]:A [3]:B [2]:B [guard] [9]:C [0]:C [8]:C [4]:C [9]:C [5]:C [start/stop]

(*) For the calculation of the CHK see above.

5.26 Symbology: **EAN-8**

5.26.1 Overview

This symbology is known as:
- EAN-8

Usage
Application: for small products with limited label space; e.g. books
Regional use: same as EAN-13
Popularity: same as EAN-13

Remarks
This symbology has nothing in common with UPC-E (!).
This symbology can have an optional supplemental 2-digit or 5-digit encoding (for books and periodicals) (EAN-8+2, EAN-8+5).

Example
These are samples of typical barcodes of this symbology.
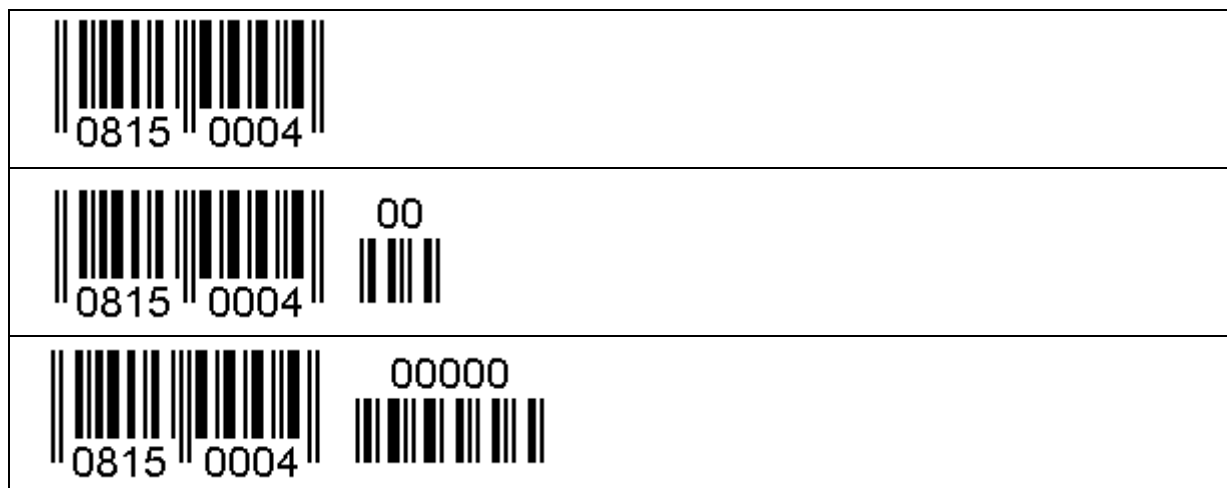
Further information

See the chapter "EAN-13" above.

Applications

Applications based on this symbology are:
- EAN Velocity (*)
Those marked (*) are directly supported by this product; they are explained in separate sections right after this symbology.

5.26.2 Character set (Alphabet)

Message character set
Refer to the corresponding information in the chapter for the symbology family above.

Message length restrictions
The message length is fixed to = **7 digits** + 1 CHK.

Other message data restrictions
None.

Semantics (Interpretation of message characters)

- Country code   (2 or 3)
- Supplier # & Product/article #   (5 or 4)
- CHK   (1)

For the country codes, refer to the chapter "Symbology: EAN-13" above.

<u>**Symbology character set**</u>

Symbology data characters
Refer to the corresponding information in the chapter for the symbology family above.

Symbology control characters
Refer to the corresponding information in the chapter for the symbology family above.

5.26.3 Reference numbers

Refer to the corresponding information in the chapter for the symbology family above.

5.26.4 Checksum methods

Refer to the corresponding information in the chapter for the symbology family above.

The mod 10 CHK algorithm described there is applied to the message M = "<d#1>...<d#7>".

<u>**Example**</u> **(CHK calculation)**

M = "4 3 6 8 9 3 2"
-->
CHK = - { (4+6+9+2) *3 + (3+8+3) *1 } mod 10 = - 77 = - 8 x <u>10</u> + **3**

5.26.5 Encoding scheme

Refer to the corresponding information in the chapter for the symbology family above.

5.26.6 Encoding (I): The Logical encoding

For the logical encoding, refer to the chapter of the symbology family above.

5.26.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

<u>**Measurements**</u>

Height	The recommended height (incl. clear text) is 70 % of the overall width (incl. the quiet zone).
Width	21.87 .. 53.46 mm; see the table below
Module width X	10.6 .. 26.0 mil; see the table below
N:W ratio	1:2.0

Note: The overall width is 81 X = **67 X** (barcode) + 14 X (quiet zone).

Name of size	Scale factor	Module width [mm]	Module width [mil]	EAN-8 total width [mm]	EAN-8 height [mm]		
SC0	0.818	0.270	10.6	21.87	17.70		
SC1	0.90	0.297	11.7	24.06	19.48		
SC2 (*)	**1.00 (100%)**	**0.330**	**13.0**	**26.73**	**21.64**		
SC3	1.10	0.363	14.3	29.40	23.80		
SC4	1.20	0.396	15.6	32.08	25.97		
SC5	1.35	0.446	17.7	36.09	29.21		
SC6	1.50	0.495	19.5	40.10	32.46		
SC7	1.65	0.544	21.4	44.10	35.71		
SC8	1.85	0.610	24.0	49.45	40.03		
SC9	2.00	0.660	26.0	53.46	43.28		

The total width is including the quiet zone.
The height is including the clear text.

(*) SC2 is the nominal size.

Imaging conventions

Normally, EAN codes are printed with clear text underneath in a fully embedded style.

The pool pattern for EAN-8 looks like this:

```
        1  2  3  4       5  6  7  C
    (   A  A  A  A  -  C  C  C  C  (
```

5.26.8 Code construction: Steps

The message data M = "<d#1>...<d#7>" is converted into the symbology character set according to the following scheme.

byte #	1	2	3	4	5	6	7	8	9	10	11
symb. char.	[start/stop]	[<d#1>]	[<d#2>]	[<d#3>]	[<d#4>]	[guard]	[<d#5>]	[<d#6>]	[<d#7>]	[<CHK>]	[start/stop]
pool		:A	:A	:A	:A		:C	:C	:C	:C	

1. Apply check characters

Calculate the check digit according to the algorithm indicated above.
Append it to the message.

2. Convert the message characters into symbology characters

Convert the message digits <d#i> and the CHK into symbology characters [<d#i>], using the pools A and C as indicated above.

3. Insert symbology control characters

Insert the [start/stop], and [guard] characters where indicated.

5.26.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.26.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		4 3 6 8 9 3 2
		-- (*)
		4 3 6 8 9 3 2 **3**
		-- (**)
		[start/stop] [4]:A [3]:A [6]:A [8]:A [guard] [9]:C [3]:C [2]:C [3]:C [start/stop]

(*) For the calculation of the CHK see above.

5.26.11 Application: EAN-Velocity

This is not a proper symbology, but just a shorter way to specify the message data.
It is a special version of EAN-8, with the first digit = 0.

Usage
It is used for internal article numbering.

Message length
The length of this code is fixed to **6 digits**.

Code construction
1. If less than 6 digits are specified, pad the number to 6 digits on the left with leading zeroes.
2. Then add another digit '0' at the beginning.
3. Then follow the code construction procedure for EAN-8.

5.27 Symbology: **UPC-A**

5.27.1 Overview

This symbology is known as:
- UPC-A
- UPC version A

Usage

Application:	article numbering for retail usage
Regional use:	USA / Canada
Popularity:	omnipresent

Remarks

This symbology is a subset of EAN-13 (with 1st digit = 0).
This symbology can have an optional supplemental 2-digit or 5-digit encoding (for books and periodicals) (UPC-A+2, UPC-A+5).

Example

These are samples of typical barcodes of this symbology.

Further information

See the chapter "The UPC/EAN family" above.

Applications

There are no special applications based on this symbology.

5.27.2 Character set (Alphabet)

Message character set

Refer to the corresponding information in the chapter for the symbology family above.

Message length restrictions

The message length is fixed to = **11 digits** + 1 CHK.

Other message data restrictions

None.

Semantics (Interpretation of message characters)

- category	(1) ("system number")
- data	(10) = supplier ID (ca. 5) + product ID (ca. 5)
- CHK	(1)

Symbology character set

Symbology data characters
Refer to the corresponding information in the chapter for the symbology family above.

Symbology control characters
Refer to the corresponding information in the chapter for the symbology family above.

5.27.3 Reference numbers

Refer to the corresponding information in the chapter for the symbology family above.

5.27.4 Checksum methods

Refer to the corresponding information in the chapter for the symbology family above.

The mod 10 CHK algorithm described there is applied to the message M = "<d#1>...<d#11>".

Example (CHK calculation)

M = "1 2 3 4 5 6 7 8 9 0 1"
-->
CHK = - { (1+3+5+7+9+1) *3 + (2 + 4 + 6 + 8 + 0) *1 } mod 10 = - 98 = - 10 x <u>10</u> + **2**

5.27.5 Encoding scheme

Refer to the corresponding information in the chapter for the symbology family above.

5.27.6 Encoding (I): The Logical encoding

For the logical encoding, refer to the chapter of the symbology family above.

5.27.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

N:W ratio	1:2.0

Note: The overall width is 116 **95 X** (barcode) .

Name of size	Scale factor	Module width [mm]	Module width [mil]	UPC-A total width [mm]	UPC-A height [mm]
SC2 (*)	1.00 (100%)	0.33	13.0	37.29	26.26

(*) SC2 is the nominal size. For the other scale factors, see the table at EAN-13.
The total width is including the quiet zone.
The height is including the clear text.

Imaging conventions

Normally, UPC codes are printed with clear text underneath in a fully embedded style.
The 1st digit (which is also coded directly), is additionally placed as clear text in front of the barcode.
The check digit (which is also coded directly), is additionally placed as clear text immediately behind the barcode.
The bar height of the first digit and the check digit is the same as of the start, guard, and stop characters.

The pool pattern for UPC-A looks like this:

```
          1      1 2 3 4 5 6      7 8 9 10 11 C        C
          D  (  A' A A A A A  -  C C C C C C'  (   D
```

For the definition of the pools, see "The Physical Encoding" below.

5.27.8 Code construction: Steps

The message data M = "<d#1>...<d#11>" is converted into the symbology character set according to the following scheme.

byte #	1	2	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
symb. char.	[d#1]	[start/ stop]	[d#1]	[d#2]	[d#3]	[d#4]	[d#5]	[d#6]	[guard]	[d#7]	[d#8]	[d#9]	[d#10]	[d#11]	[CHK]	[start/ stop]	[CHK]
pool	:D		:A'	:A	:A	:A	:A	:A		:C	:C	:C	:C	:C	:C'		:D

1. Apply check characters

Calculate the check digit according to the algorithm indicated above.
Append the result to the message.

2. Convert the message characters into symbology characters

Convert each message digit <d#i> into a symbology character [<d#i>], using the pools A and C, as indicated above.
Follow the CTX convention using pools D,A',C', as indicated in the scheme.

3. Insert symbology control characters

Insert the [start/stop] and [guard] characters as indicated above.

5.27.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.27.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		1 2 3 4 5 6 7 8 9 0 1
		-- (*)
		1 2 3 4 5 6 7 8 9 0 1 **2**
		-- (**)
		[1]:D [start/stop] [1]:A' [2]:A [3]:A [4]:A [5]:A [6]:A [guard] [7]:C [8]:C [9]:C [0]:C [1]:C [2]:C' [start/stop] [2]:D

(*) For the calculation of the CHK see above.

5.28 Symbology: **UPC-E**

5.28.1 Overview

This symbology is known as:
- UPC-E
- UPC version E

 UPC-E is a zero suppression version of UPC-A.
This symbology exists in 2 different variants:
- UPC-E0, E0
- UPC-E1, E1
Note: If only "UPC-E" is specified, most likely (the historically first) UPC-E0 is referred to.

Usage

Application:	compact code for small products/packages
Regional use:	same as UPC-A
Popularity:	same as UPC-A

Remarks
This symbology is a compressed version of UPC-A (using compression of 4 consecutive zeroes).
This symbology can have an optional supplemental 2-digit or 5-digit encoding (for books and periodicals) (UPC-E+2, UPC-E+5).

Example
These are samples of typical barcodes of this symbology.
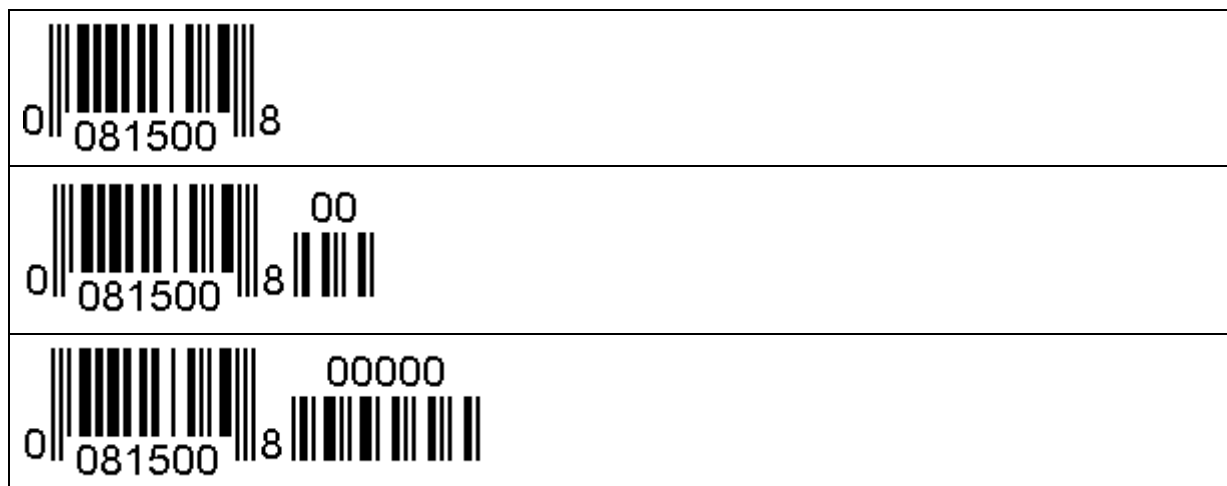
Further information

See the chapter "The UPC/EAN family" above.

Applications

There are no special applications based on this symbology.

5.28.2 Character set (Alphabet)

Message character set
Refer to the corresponding information in the chapter for the symbology family above.

Message length restrictions
The message length is fixed to = **7 digits** + 1 CHK.

Other message data restrictions
The 1st digit must be 0 (**UPC-E0**) or 1 (**UPC-E1**).

Semantics (Interpretation of message characters)
- {0,1}   (1)
- data   (6)
- CHK   (1)

Symbology character set

Symbology data characters
Refer to the corresponding information in the chapter for the symbology family above.

Symbology control characters

Refer to the corresponding information in the chapter for the symbology family above.
In addition, this control character is defined only for UPC-E:

Control character	Function
[stop UPC-E]	Special stop of barcode only for UPC-E.

5.28.3 Reference numbers

Refer to the corresponding information in the chapter for the symbology family above.

5.28.4 Checksum methods

In the case of UPC-E, the checksum algorithm is not directly applied to the message M = "<d#1> ... <d#7>" (where <d#1> = 0 or 1).
Instead, first the message data has to be expanded into the 11-digit UPC-A format by padding it with 4 or 5 consecutive zeroes. The position of these depends on the value of the last message digit x := <d#7>, according to the table below.
After the expansion, the mod 10 CHK algorithm is then applied to the extended message M' = "<c#1>...<c#11>", as described in chapter "Symbology: UPC-A" above.

x = <d#7>	UPC-E "######**x**" --> UPC-A **Expansion**:	Example: M = "198765**x**"
x = 0..2	###**x**0000###	198**x**0000765
x = 3	####00000##	19870000065
x = 4	#####00000#	19876000005
x = 5..9	######0000**x**	1987650000**x**

or

Original number (UPC-A Expansion)											Shortened version (UPC-E)						
d1	d2	d3	**0**	0	0	0	0	d4	d5	d6	d1	d2	d3	d4	d5	d6	**0**
d1	d2	d3	**1**	0	0	0	0	d4	d5	d6	d1	d2	d3	d4	d5	d6	**1**
d1	d2	d3	**2**	0	0	0	0	d4	d5	d6	d1	d2	d3	d4	d5	d6	**2**
d1	d2	d3	d4	0	0	0	0	0	d5	d6	d1	d2	d3	d4	d5	d6	**3**
d1	d2	d3	d4	d5	0	0	0	0	0	d6	d1	d2	d3	d4	d5	d6	**4**
d1	d2	d3	d4	d5	d6	0	0	0	0	**5**	d1	d2	d3	d4	d5	d6	**5**
d1	d2	d3	d4	d5	d6	0	0	0	0	**6**	d1	d2	d3	d4	d5	d6	**6**
d1	d2	d3	d4	d5	d6	0	0	0	0	**7**	d1	d2	d3	d4	d5	d6	**7**
d1	d2	d3	d4	d5	d6	0	0	0	0	**8**	d1	d2	d3	d4	d5	d6	**8**
d1	d2	d3	d4	d5	d6	0	0	0	0	**9**	d1	d2	d3	d4	d5	d6	**9**

Example (CHK calculation)

M = "0 1 2 3 4 5 **6**"
-->
M' = "0 1 2 3 4 5 0 0 0 0 6"
-->
CHK = - { (0+2+4+0+0+6) *3 + (1+3+5+0+0) *1 } mod 10 = - 45 = - 5 x 10 + **5**
-->
M'' = "0 1 2 3 4 5 6 **5**"

5.28.5 Encoding scheme

Refer to the corresponding information in the chapter for the symbology family above.

5.28.6 Encoding (I): The Logical encoding

For the logical encoding, refer to the chapter of the symbology family above.

The pool pattern for UPC-E is:

	1	2	3	4	5	6	7		C	
D	**(**	*	*	*	*	*	*	**)**		**D**

Neither the first digit <d#1> nor the CHK are encoded directly as barcode characters.
The following 6 digits <d#2> .. <d#7> are encoded using pools A and B, depending on the value of <d#1> (= 0 or 1) and CHK.
The pool pattern to be used is determined according to the following pool scheme.

	UPC-E0						UPC-E1					
<CHK>	<d#2>	<d#3>	<d#4>	<d#5>	<d#6>	<d#7>	<d#2>	<d#3>	<d#4>	<d#5>	<d#6>	<d#7>
0	B	B	B	A	A	A	A	A	A	B	B	B
1	B	B	A	B	A	A	A	A	B	A	B	B
2	B	B	A	A	B	A	A	A	B	B	A	B
3	B	B	A	A	A	B	A	A	B	B	B	A
4	B	A	B	B	A	A	A	B	A	A	B	B
5	B	A	A	B	B	A	A	B	B	A	A	B
6	B	A	A	A	B	B	A	B	B	B	A	A
7	B	A	B	A	B	A	A	B	A	B	A	B
8	B	A	B	A	A	B	A	B	A	B	B	A
9	B	A	A	B	A	B	A	B	B	A	B	A

Example: For UPC-E1, if <CHK> = 7, the pool pattern is "ABABAB".

Note:
The pool scheme for UPC-E1 is almost identical with the pool scheme for EAN-13, except for the row for "0".
The pool scheme for UPC-E0 can be derived from the pool scheme for UPC-E1 by simply interchanging "A" and "B".

5.28.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Height	The recommended height (incl. clear text) is 70 % of the overall width (incl. the quiet zone).
N:W ratio	1:2.0

Note: The overall width is 72 X = **51 X** (barcode) + 21 X (quiet zone).

Name of size	Scale factor	Module width [mm]	Module width [mil]	UPC-E total width [mm]	UPC-E height [mm]
SC2 (*)	**1.00 (100%)**	**0.33**	**13.0**	**23.76**	

(*) SC2 is the nominal size. For the other scale factors, see the table at EAN-13.
The total width is including the quiet zone.
The height is including the clear text.

Imaging conventions

Normally, UPC codes are printed with clear text underneath in a fully embedded style.
The 1st digit ('0' or '1', which is not coded directly), is placed as clear text in front of the barcode.
The check digit (which is not coded directly), is placed as clear text immediately behind the barcode.

The pool pattern for UPC-E looks like this:

	1	2	3	4	5	6	7		C	
D	**(**	*	*	*	*	*	*	**)**		**D**

5.28.8 Code construction: Steps

The message data M = "<d#1>...<d#7>" is converted into the symbology character set according to the following scheme.

byte #	1	2	3	4	5	6	7	8	9	10
symb. char.	[d#1]	[start/stop]	[d#2]	[d#3]	[d#4]	[d#5]	[d#6]	[d#7]	[stop UPC-E]	[CHK]
pool	:D		:A/B (*)	:A/B (*)	:A/B (*)	:A/B (*)	:A/B (*)	:A/B (*)		:D

Note: (*)
The pool (A or B) in which each character is encoded depends on the value of <d#1> (0 or 1) and the <CHK>.
See the pool scheme above.
Similarly to EAN-13, <d#1> and <CHK> are thus not directly coded as characters.

1. Apply check characters

Calculate the check digit according to the algorithm indicated above.
This digit is not appended to the message, but used to determine the pool pattern for digits <d#2> .. <d#7>.

2. Convert the message characters into symbology characters

Depending on the value of the 1st digit <d#1> and the <CHK>, determine the pool pattern to be used for digits <d#2> ... <d#7>, according to the pool scheme in section "Encoding (I)" above.
Convert the message digits <d#i> into symbology characters [<d#i>], for i = 2..7, according to this pool pattern.

3. Insert symbology control characters

Insert the [start/stop] and [stop UPC-E] characters as indicated above.

5.28.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.28.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		0 1 2 3 4 5 6
		-- (*)
		0 1 2 3 4 5 6 **5**
		-- (**)
		[0]:D [start/stop] [1]:B [2]:A [3]:A [4]:B [5]:B [6]:A [stop UPC-E] [5]:D

(*) For the calculation of the CHK see above.

5.29 Symbology: **The EAN/UPC 2-digit supplement**

5.29.1 Overview

This symbology is known as:
- EAN/UPC-x+2
- EAN/UPC-x +2-digit add-on
- EAN/UPC plus 2-digit supplement
. EAN 2

The 2-digit supplement extends the EAN/UPC codes.

Usage
The 2 additional digits specify the number of the week of the year (01-52) in weekly publications.
Application: weekly magazins, publishing
Regional use: world-wide; newspaper and magazine business
Popularity: omnipresent

Example
This is a sample of a typical barcode of this symbology. (This is EAN-8+2.)

More examples were given for each symbology at EAN-13, EAN-8, UPC-A, and UPC-E above.

Further information

See the chapter "The UPC/EAN family" above.

Applications

There are no special applications based on this symbology.

5.29.2 Character set (Alphabet)

Message character set
Refer to the corresponding information in the chapter for the symbology family above.

Message length restrictions
The message length is fixed, as follows:
- EAN-13 + 2   = 12 + 1 CHK + 2   = 14 + 1 CHK
- EAN-8 + 2   = 7 + 1 CHK + 2   = 9 + 1 CHK
- UPC-A + 2   = 11 + 1 CHK + 2   = 13 + 1 CHK
- UPC-E + 2   = 7 + 1 CHK + 2   = 9 + 1 CHK

Other message data restrictions
None.

Semantics (Interpretation of message characters)
The 2 additional digits specify the number of the week of the year (01-52) in weekly publications.

Symbology character set

Symbology data characters
Refer to the corresponding information in the chapter for the symbology family above.

Symbology control characters

Control character	Function
[start supp]	Start bars for 2- or 5-supplement.
[guard supp]	Guard bars for 2- or 5-supplement, to separate characters. ("01"="sb")

Refer also to the corresponding information in the chapter for the symbology family above.

5.29.3 Reference numbers

Refer to the corresponding information in the chapter for the symbology family above.

5.29.4 Checksum methods

The 2-digit supplement employs its own checksum method.
The CHK is not encoded directly as a barcode character, but instead determines the pool pattern to be used for the message digits.

CHK Formula

1 mandatory CHK

#CHK = S mod 4

CHK Algorithm
The number represented by the two supplement digits, S = "<d#1> <d#2>", is divided by 4.
The remainder determines which pool (E or F) is used to encode each of the two digits, according to the table below.

Example (CHK calculation)

S = "89"
-->
Remainder = 89 mod 4 = 22 x <u>4</u> + **1**

5.29.5 Encoding scheme

Refer to the corresponding information in the chapter for the symbology family above.

For the supplement, pools E and F are used.
They correspond to pools A and B, but in addition they have clear text on top.

5.29.6 Encoding (I): The Logical encoding

For the logical encoding, refer to the chapter of the symbology family above.

The pool pattern for the 2-digit supplement is:

	1	2
+2 addon: #	* $	*

The 2 digits <d#1> <d#2> are encoded using pools E and F, depending on the value of CHK. The pattern to be used is determined according to the following pool scheme.

Remainder	<d#1>	<d#2>
0	E	E
1	E	F
2	F	E
3	F	F

<u>Example</u>: Remainder = **1** --> Pool pattern = "**E F**"

5.29.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

The distance between (the stop character of) the main code and (the start character of) the supplement must lie in the range 7X .. 10X.

Example: (EAN-13+2)

Name of size	Scale factor	Module width [mm]	Module width [mil]	EAN-13+2 total width [mm]	EAN-13+2 height [mm]
SC2 (*)	**1.00 (100%)**	**0.33**	**13.0**	45.87	25.93

Imaging conventions

For the main portion of the barcode, see the corresponding chapter above.
For the supplement portion of the barcode, the clear text has to be on top of the bars.

5.29.8 Code construction: Steps

The complete barcode looks like this:
  <EAN-x or UPC-x main barcode> <space> <2-digit add-on>

The message data M = "<main message><d#1><d#2>" is converted into the symbology character set according to the following scheme.

<main code> [SP]:D [start supp] [<d#1>]:E/F [guard supp] [<d#2>]:E/F

---

or

symb. char.	<main code>	[SP]	[start supp]	[<d#1>]	[guard supp]	[<d#2>]
pool		**:D**		**:E/F (*)**		**:E/F (*)**

Note: ("[SP]:D")
The pseudo symbology character [SP]:D is a pseudo pool D character, which implements a blank space having the same width as the other pool D characters. It is used for separating the 2-digit supplement code from the main code.

Note: (*)
The pool (E or F) in which this character is encoded depends on the value of the CHK determined above.

1. Apply check characters

Calculate the check digit according to the algorithm indicated above.
The result is not appended to the message but determines the pool pattern to be used.

2. Convert the message characters into symbology characters

Convert the message digits <d#1>, <d#2> into the symbology characters [<d#1>] [<d#2>], using pools E and F, as indicated above, depending on the value of the CHK.
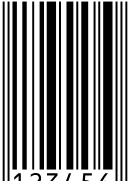
3. Insert symbology control characters

Insert the [start supp] and [guard supp] characters as indicated above.

5.29.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.29.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		... 8 9
		-- (*) (**)
		... [SP]:D [start supp] [8]:**E** [guard supp] [9]:**F**

		8 9

. . .

(*) For the calculation of the CHK see above.

5.30 Symbology: **The EAN/UPC 5-digit supplement**

5.30.1 Overview

This symbology is known as:
- EAN/UPC-x +5
- EAN/UPC plus 5-digit add-on
- EAN/UPC plus 5-digit supplement
. EAN 5

The 5-digit supplement extends the EAN/UPC codes.

<u>**Usage**</u>
The 5 additional digits specify the number of the day or the price of the publication.
 Application: journals, newspapers, books
 Regional use: world-wide; newspaper and book publishing business
 Popularity: omnipresent

<u>**Example**</u>
This is a sample of a typical barcode of this symbology. (This is EAN-8+5.)

More examples were given for each symbology at EAN-13, EAN-8, UPC-A, and UPC-E above.

<u>**Further information**</u>

See the chapter "The UPC/EAN family" above.

<u>**Applications**</u>

There are no special applications based on this symbology.

5.30.2 Character set (Alphabet)

<u>**Message character set**</u>
Refer to the corresponding information in the chapter for the symbology family above.

<u>**Message length restrictions**</u>
The message length is fixed, as follows:
- EAN-13 + 5   = 12 + 1 CHK + 5   = 17 + 1 CHK
- EAN-8 + 5   = 7 + 1 CHK + 5   = 12 + 1 CHK
- UPC-A + 5   = 11 + 1 CHK + 5   = 16 + 1 CHK
- UPC-E + 5   = 7 + 1 CHK + 5   = 12 + 1 CHK

<u>**Other message data restrictions**</u>
None.

<u>**Semantics**</u> (Interpretation of message characters)
The 5-digit supplement encodes the price.
- 1st digit = currency (0=Pound/DM, 5= US-$, 9=special)
- digits 2..5 = price

<u>**Symbology character set**</u>

Symbology data characters
Refer to the corresponding information in the chapter for the symbology family above.

Symbology control characters
Refer to the corresponding information in the chapter "2-digit Supplement" above.

5.30.3 Reference numbers

Refer to the corresponding information in the chapter "2-digit Supplement" above.

5.30.4 Checksum methods

The 5-digit supplement employs its own checksum method.
The CHK is not encoded directly as a barcode character, but instead determines the pool assignment to the message digits.

<u>CHK Formula</u>

1 mandatory CHK

#CHK = + [*3, *9, *3, *9, *3] x #S mod 10

<u>CHK Algorithm</u>

1. Sum all odd-positioned digits (#1,#3,#5).
   Multiply the result by 3.
2. Sum all even-positioned digits (#2,#4).
   Multiply the second sum by 9.
3. Add both results.
4. Divide the sum by 10.
The remainder determines which pool (E or F) is used to encode each of the five message digits.

<u>Example (CHK calculation)</u>

S = "12345"
-->
Remainder = (1+3+5) *3 + (2+4) *9 mod 10 = 27 + 54 = 81 = 8 x 10 + **1**

5.30.5 Encoding scheme

See the chapter "2-digit supplement" above.

5.30.6 Encoding (I): The Logical encoding

For the logical encoding, refer to the chapter of the symbology family above.

The pool pattern for the 5-digit supplement is:

		1		2		3		4		5
+5 addon:	#	*	$	*	$	*	$	*	$	*

The 5 digits <d#1> ... <d#5> are coded using pools E and F, depending on the value of CHK. The pattern to be used is determined according to the following pool scheme.

Remainder	<d#1>	<d#2>	<d#3>	<d#4>	<d#5>
0	F	F	E	E	E
1	F	E	F	E	E
2	F	E	E	F	E
3	F	E	E	E	F
4	E	F	F	E	E
5	E	E	F	F	E
6	E	E	E	F	F
7	E	F	E	F	E
8	E	F	E	E	F
9	E	E	F	E	F

<u>Example</u>: Remainder = **1** --> Pool pattern = "**F E F E E**"

5.30.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

The distance between (the stop character of) the main code and (the start character of) the supplement must lie in the range 7X .. 10X.

Example: (EAN-13+5)

Name of size	Scale factor	Module width [mm]	Module width [mil]	EAN-13+5 total width [mm]	EAN-13+5 height [mm]
SC2 (*)	**1.00 (100%)**	**0.33**	**13.0**	54.78	25.93

Imaging conventions

For the main portion of the barcode, see the corresponding chapter above.
For the supplement portion of the barcode, the clear text has to be on top of the bars.

5.30.8 Code construction: Steps

The complete barcode looks like this:
   <EAN-x or UPC-x barcode> <space> <5-digit add-on>

The message data M = "<main message><d#1>...<d#5>" is converted into the symbology character set according to the following scheme.

<main code> [SP]:D [start supp] [<d#1>]:E/F [guard supp] ... [guard supp] [<d#5>]:E/F

---

or

symb. char.	<main code>	[SP]	[start supp]	[d#1]	[guard supp]	[d#2]	[guard supp]	[d#3]	[guard supp]	[d#4]	[guard supp]	[d#5]
pool		:D		:E/F (*)		:E/F (*)		:E/F (*)		:E/F (*)		:E/F (*)

Note: ("[SP]:D")
The pseudo symbology character [SP]:D is a pseudo pool D character, which implements a blank space having the same width as the other pool D characters. It is used for separating the 5-digit supplement code from the main code.

Note: (*)
The pool (E or F) in which this character is encoded depends on the CHK value determined above.

1. Apply check characters

Calculate the check digit according to the algorithm indicated above.
The result is not appended to the message but determines the pool pattern.

2. Convert the message characters into symbology characters

Convert the message digits <d#1>, ..., <d#5> into symbology characters [<d#1>] ... [<d#5>], using the pools E and F, as indicated above, depending on the value of the CHK.

3. Insert symbology control characters

Insert the [start supp] and [guard supp] characters as indicated above.

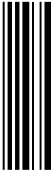## 5.30.9 Encoding (II): The Physical encoding

In the **physical realization** of the encoding as **barcode images**, every symbology character is mapped to a barcode character.
Depending on the barcode solution, this can be the ASCII code of a barcode font character, whose glyph makes up the black-and-white rectangular pattern, or it can be a sequence of Draw rectangle commands in some printer language, realizing the black-and-white rectangular pattern, corresponding to the logical width encoding defined above.

5.30.10   Code construction: The complete procedure (Example)

This section contains a full example of all stages of the Code construction.

		Data
		... 1 2 3 4 5
		-- (*) (**)
		... [SP]:D [start supp] [1]:F [guard supp] [2]:E [guard supp] [3]:F [guard supp] [4]:E [guard supp] [5]:E
		1 2 3 4 5 ...

(*) For the calculation of the CHK see above.

5.31 Symbology: **Australian 4-state postal**

5.31.1 Overview

This symbology is known as:
- Australian 4-state postal
- Australia Post 4-state

Since 1999.

This symbology comes in several versions:

Short Name	Proper Name	FCC code	Structure (# bars)
37-CUST	"Standard Customer Barcode" ("Customer Barcode 1")	FCC = **11**	DPID (16)
37-REPL	"Reply Paid Barcode"	FCC = **45**	dto.
37-ROUT	"Routing Barcode"	FCC = **87**	dto.
37-REDI	"Redirection Barcode"	FCC = **92**	dto.
52-FF-MET	"Customer Barcode 2"	FCC = **59**	DPID (16) + CustInfo (16)
67-FF-MET	"Customer Barcode 3"	FCC = **62**	DPID (16) + CustInfo (31)
67-FF-MAN	(reserved)	FCC = **44**	dto.

Note: The first number indicates the total number of bars in the barcode.
Note: FCC stands for "Format Control Code".

DPID = "Delivery Point Identifier" or "Sorting Code"
Identifies destination of a piece of mail = 16 bars = 8 digits (using the N table).

CustInfo = "Customer Info"
Free-format customer-specific data area, may contain any information not interpreted by Australia Post = 16 or 31 bars.
If it is numeric only, it can be encoded using the N table, thus accommodating 8 or 15 digits.
Otherwise the C table has to be used, thus accommodating 5 or 10 chars.

Usage
Application: Postal code, for automatic mail sorting
Regional use: Australia only

Remarks
Characteristics: fixed height and width;

Example
This is a sample of a typical barcode of this symbology.

Further information

The specification of this symbology is maintained at:
  Australia Post : //www.auspost.com.au/download/: { Australian4StateBarcodeSpec.pdf, GuideToPrint4StateBarcode.pdf, 4state.pdf }

Other sources of information can be found at Appendix L.

Applications

There are no special applications based on this symbology.

5.31.2 Character set (Alphabet)

Message character set

The message character set consists of **64 = 10+26+26+2** characters.

Message data characters
Alphanumeric {0,...,9, A, ..., Z, a, ..., z, <SP>, #}.

Message control characters
None.

Message length restrictions
The message length is fixed, as indicated above.

Symbology character set

Symbology data characters
There is a symbology character for every message character. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start/stop]	"frame bar" = start and stop of barcode

5.31.3 Reference numbers

Not applicable.

5.31.4 Checksum methods

This symbology employs a Reed-Solomon ECC algorithm which produces a 24-bit CHK encoded in 12 bars.

5.31.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	"4-state Bar height encoding" There is no information in the spaces or widths.
Dimensionality:	--
(Continuous / Discrete):	--
Logical encoding type:	Uses 3 different encoding schemes: - N table = digits only - C table = alphanumeric {A..Z, a..z, 0..9, <SP>, #} - ECC table = for Reed-Solomon
# Modules per element:	4 states of a bar: Descender, Tracker, Ascender, Full; (see above)
# Elements per character:	N table: = **2b** C table, ECC table: = **3b**
# Modules per character:	--
Self-checking:	no ! (needs CHK)
Bi-directional:	yes

5.31.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

Structure of barcode

	(# bars)	"pattern"	value
- <start>	(2)	"13"	

- <FCC>	(4)	var.	2 digits (N table)		
- <DPID>	(16)	var.	8 digits (N table)		
- <Filler bar> or	(1)	"3"	(*)		
- <CustInfo>	(16) or	var.	(*) 8 digits (N table) or 5 chars (C table)		
	(31)		(*) 15 digits (N table) or 10 chars (C table)		
- <Reed-Solomon ECC>	(12)	var.	(R/S table)		
- <stop>	(2)	"13"			

Note: (*)
If no CustInfo is present, 1 single Filler bar must be placed.
If the CustInfo encoding results in less than 16 or 31 bars, respectively, it must be padded using Filler bars.
Which encoding table (N or C) is to be used for CustInfo, is entirely up to the customer. A barcode solution could determine it automatically depending on whether CustInfo consists of only digits or is alphanumeric. Or, it could provide a parameter to specify it explicitly.

5.31.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Refer to the official specifications.

Imaging conventions

CTX is optional. By convention it is to be placed above the barcode outside the quiet zone, with a max. 8 point font size.

5.31.8 Code construction: Steps

The message data M = "<DPID> [, <Customer Info>]" is converted into the symbology character set according to the following scheme.

[start/stop] [<FCC>] [<DPID>] [[<Customer Info>]] [<ECC>] [start/stop]

---

1. Apply check characters

Calculate the ECC according to the algorithm above and append it to the message.

2. Convert the message characters into symbology characters

Convert each message character <d#i> into the symbology character [<d#i>].

3. Insert symbology control characters

Prepend and append the [start/stop] character.

5.32 Symbology (family): **Kix / RM4SCC / Singapore 4-state postal**

5.32.1 Overview

This symbology family comprises 3 members:
- **Kix**: "**K**i**x**" is an acronym of "**K**lanten**i**nde**x**", which is Dutch for "client index".
- **RM4SCC**: Royal Mail 4-State Customer Code
- **Singapore**: Singapore 4-state postal code

Usage
Application:	Postal code, for automatic mail sorting	
	It provides information about the address of the receiver.	
	If the postal code is preprinted on a mailpiece, a discount applies to the rate.	
Regional use:	- Kix:	by Dutch (Netherlands) post office
	- RM4SCC:	by Royal Mail (United Kingdom) post office
	- Singapore:	by Singapore post office

Remarks
Characteristics: fixed height and width;

Example
This is a sample of a typical barcode of this symbology.
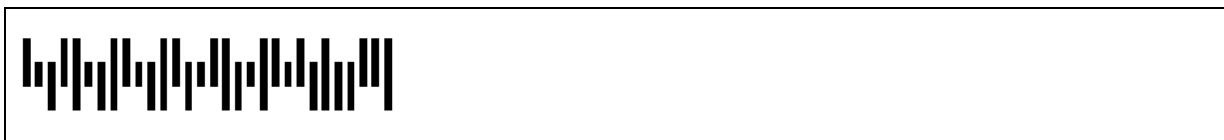
Further information

The specification of this symbology is maintained at:
- Kix: www.ptt-post.kix.nl
- RM4SCC: Royal Mail
- Singapore: Singapore Post Pte Ltd.

Other sources of information can be found at .

Applications

There are no special applications based on this symbology.

5.32.2 Character set (Alphabet)

Message character set

The message character set consists of 36 = **10+26** characters.

Message data characters
Alphanumeric {0,...,9, A, ..., Z}.

Message control characters
None.

Message length restrictions
The message length is fixed, as follows:

Kix: =
- <4 digits> + <2 alpha>   -- post code
  [[+ <1..5 digits>]   -- house #, postbus (P.O.box) #, reply #
    [+ 'X' + <1..6 chars>]]   -- house # supplement

RM4SCC: =
- 7, 8, or 9 chars   -- post code

Singapore: =
- 6 digits -- Postal code

<u>or</u>
- 4 chars   -- BRS (Business Reply Service) License Number code

Symbology character set

The symbology character set consists of **38** = **36** (message) **+ 2** (control) characters.

Symbology data characters
There is a symbology character for every message digit. See the table in the section Encoding (I) for the full list.

Symbology control characters

Control character	Function
[start]	start of barcode (only for Singapore)
[stop]	stop of barcode (only for Singapore)

5.32.3 Reference numbers

For the CHK algorithm, the Asc and Dsc values are used.

5.32.4 Checksum methods

- **Kix**:   = no CHK (!)
- **RM4SCC**:   = 1 CHK
- **Singapore**:   = 1 CHK

The CHK algorithms are pretty complicated. Refer to the proper spec's.

5.32.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	"4-state Bar height encoding" There is no information in the spaces or widths.
Dimensionality:	--
(Continuous / Discrete):	--
Logical encoding type:	easily derived from binary representation
# Modules per element:	4 states of a bar: Full, Ascender, Descender, Tracker
# Elements per character:	**4 b** (2 have Asc, 2 have Dsc)
# Modules per character:	**6** = 2 * 2(Asc)+ 2 * 1(Dsc)

5.32.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

5.32.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Singapore:

Height of Tracker bar	= 1.02 .. 1.52 mm
Height of Asc/Dsc bar	= 1.6 .. 2.16 mm
Pitch	20..24 bars/inch
Module width X (bar,space)	= 0.38 .. 0.63 mm

<u>**Imaging conventions**</u>

There is no clear text.

5.32.8 Code construction: Steps

Kix:

[<c1>] ... [<c6>] [[<c7>] ... [<c18>]]

RM4SCC:

[<c1>] ... [<c7/8/9>] [<CHK>]

Singapore:

[start] [<c1>] ... [<c4/6>] [<CHK>] [stop]

5.33 Symbology: **Intelligent Mail barcode (IMb) (USPS postal 4-state)**

5.33.1 Overview

This symbology is known as:
- Intelligent Mail barcode
- Intelligent Mail(R) Barcode
- USPS OneCode Solution
- USPS 4-state Customer Barcode (4CB, 4-CB, USPS4CB)

Since 01.Sep.2006.
Supersedes PLANET and POSTNET.

Usage

Application:	Postal code, for automatic mail sorting
Regional use:	by United States Postal Service (USPS)

Remarks
Characteristics: fixed height and width.

Example
This is a sample of a typical barcode of this symbology.

Further information

The specification of this symbology is maintained at United States Postal Service (USPS).
Documentation (by USPS):
- "Intelligent Mail Barcode 4-state Specification" (USPS-B-3200)
- "Intelligent Mail Barcode Technical Resource Guide"
- https://ribbs.usps.gov/index.cfm?page=intelligentmail

Other sources of information can be found at Appendix L.

Applications

There are no special applications based on this symbology.

5.33.2 Character set (Alphabet)

Message character set

The message character set consists of the 10 digits {0,..,9}.

Message data characters
Digits {0,...,9}.

Message control characters
None.

Message length restrictions

Variable: either 20 or 25 or 29 or 31 digits.

Message structure

# digits	item
- 20	**Tracking Code**
(2)	- (BI) Barcode ID
(3)	- (STID) Service Type ID
(6|9)	- (MID) Mailer ID
(9|6)	- Serial Number
0	-- **Routing Code**: empty

	5	or - ZIP
	9	or - ZIP + 4
	11	or - ZIP + 4 + delivery point

A solution may require a separator of between Tracking code and Routing code.

Symbology character set

There are no dedicated [start] and [stop] characters.
The characters are a complex interleaved encoding of the message data.

5.33.3 Reference numbers

Not applicable.

5.33.4 Checksum methods

This symbology employs a complex 11-bit CRC Frame Check Sequence (FCS).

5.33.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	"4-state bar height encoding" There is no information in the spaces or widths.
Dimensionality:	--
(Continuous / Discrete):	--
Logical encoding type:	(A complex algorithm, interleaving the max. 31 message digits.)
# Modules per element:	4 states of a bar: Descender, Tracker, Ascender, Full;
# Elements per character:	5 bars. . In total 65 bars.
# Modules per character:	n/a
Self-checking:	yes
Bi-directional:	yes

5.33.6 Encoding (I): The Logical encoding

This information is not necessary for the code construction, only for troubleshooting.
Please contact technical support.

Structure of barcode

The barcode consists of exactly **65 bars**.

There is no dedicated start/stop pattern.

There are 4 different possible bar heights:

0	F	T+A+D	"Full bar"	
1	A	T+A	"Ascender bar"	
2	D	T+D	"Descender bar"	
3	T	T	"Tracker" / "Clock bar"	

5.33.7 Physical attributes

This section specifies attributes related to the physical realization of a barcode.

Measurements

Item: Specification:	Range [inch]	Range [inch]	Range [mm]
- bar width	0.020 +/- 0.005 inch	=~ 0.015 .. 0.025 inch	=~ 0.381 .. 0.635 mm
- gap width	0.026 +/- 0.014 inch	~= 0.012 .. 0.040 inch	=~ 0.305 .. 1.016 mm
- center-to-center width	0.0458 +/- 0.0042 inch	~= 0.0416 .. 0.050 inch	=~ 1.056 .. 1.270 mm
- horizontal pitch => total width (65 bars)	22 +/- 2 bars/inch	=~ 20 .. 24 bars /inch => 2.71 .. 3.25 inch	=~ 6.88 .. 8.25 cm
- height of Full bar	0.145 +/- 0.020 inch ((~~ex: 0.182 +/- 0.048))~~	=~ 0.125 .. 0.165 inch	=~ 3.18 .. 4.19 mm
- height of Tracker bar	0.048 +/- 0.009 inch	~= 0.039 .. 0.057 inch	=~ 0.99 .. 1.45 mm

Imaging conventions

CTX is optional.
If put, then to be placed immediately above or below, at a distance of 0.028 .. 0.5 inch.
The text has to be left-aligned with the 1st bar.
Fields are to be separated by a space, e.g. according to 2+3+6+9+5+4+2.
The font has to be 10 .. 12 pt sans-serif.

The quiet zone ("clear zone") around the barcode must be at least
- 0.028 inch =~ 0.71 mm vertically, and
- 0.125 inch =~ 3.175 mm horizontally.

5.33.8 Code construction: Steps

Refer to the official specification.

5.34 [SUPPORT ONLY]

5.35 [SUPPORT ONLY]

5.36 Symbology (2D)**: PDF417**

5.36.1 Overview

This symbology is known as:
- PDF417

The name refers to the large amount of data it can encode ("**P**ortable **D**ata **F**ile"), the number of bars (and spaces) per codeword (**4**), and the number of modules per codeword (**17**).

Usage

Application:	Shipping labels in shipping/carrier environment.
Regional use:	World-wide.
Popularity:	The most commonly used 2D barcode.

Remarks

Characteristics:
A stacked high-density 2D symbology.
Developed 1989-1992 by Symbol Technologies.
Standardized and maintained by ANSI / AIM USA.
Typical size 4 x 8 cm.
Not sensitive to dirt or mutilation.
Built-in error correction capability.
Automatic data compression.
Full ASCII character set.
Can be read with CCD cameras.

Example

This is a sample of a typical barcode of this symbology.

Further information

The specification of this symbology is maintained at:
- ANSI / AIM USA.

Other sources of information can be found at <u>Appendix L</u>.

Applications

Macro PDF
Allows to spread information across multiple PDF417 symbols.

Truncated PDF
Omits the right-hand portion (right row indicator & stop pattern) in order to save space.

5.36.2 Character set (Alphabet)

Message character set

The message character set consists of any binary information.

Message data characters
Any binary information.

Message control characters
None.

Message length restrictions

The resulting number of codewords must not exceed 928.

5.36.3 Reference numbers

For calculating the Reed-Solomon Error correction, the codeword values (0..928) are used.

5.36.4 Checksum methods

This symbology defines a Reed-Solomon ECC algorithm.

5.36.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	
Dimensionality:	2D stacked
(Continuous / Discrete):	continuous
Logical encoding type:	codewords
# Modules per element:	**1 .. 6**
# Elements per codeword:	**4b + 4s**
# Modules per codeword:	**17**

5.36.6 Encoding (I): The Logical encoding

This information is only necessary for troubleshooting.
Please contact Technical support.

5.36.7 Encoding (II): The Physical encoding

This information is only necessary for troubleshooting.
Please contact Technical support.

5.36.8 [SUPPORT ONLY]

5.37 Symbology (2D): **DataMatrix™**

5.37.1 Overview

This symbology is known as:
- DataMatrix

Usage

Application: pharma, production, electronics industry (e.g. ICs)
Regional use: world-wide
Popularity: growing

Remarks

Characteristics:
- a 2D matrix code
- developed 1989 by International Data Matrix = IDMatrix or RVSI-Acuity CiMatrix
- very compact / high-density, information density 13 chars / 100 sq cm
- many different chars encodable
- 2 variants: ECC 000-140 vs ECC 200 (recommended / most secure)
- secure, error correction: ECC (Reed-Solomon) <-- only ECC 200, reconstructable after up to 25% destruction
- **shape**: ECC 000-140 are all square; ECC 200 can be square or non-square rectangular;
- layout: different # rows and # columns (see table below);
- **size** of symbol: depends on selected variant and size of symbol element;
- **symbol elements**: black & white squares;
- **size** per symbol element: variable .001 sq in (= 1 mil square) .. 14.0 sq in;

There exist several **variants** (**ECC-xxx)**.

The variants can be distinguished as follows:

variant	# rows / cols / modules per row	=>	upper right square
- ECC 000-140	odd		black
- ECC 200	even		white

Layout by variant

The variants can have different **layouts** (shape, # rows x columns).

variant	shape	# rows x # cols	
- ECC 000	square	9 x 9	... 49 x 49
- ECC 050	square	11 x 11	... 49 x 49
- ECC 080	square	13 x 13	... 49 x 49
- ECC 100	square	13 x 13	... 49 x 49
- ECC 140	square	17 x 17	... 49 x 49
- ECC 200	square	10 x 10	... 144 x 144
- ECC 200	rectangular	8 x 18	... 16 x 48

Maximally encodable information & error correction capabilities by variant & layout

ECC 200 (square)

# rows x # cols	max. # digits (numeric)	max. # chars (alpha-numeric)	max. # bytes (binary)	max. correctable errors / erasures
10 x 10	6	3	1	2 / 2
12 x 12	10	6	3	3 / 3
14 x 14	16	10	6	5 / 7
16 x 16	24	16	10	6 / 9
18 x 18	36	25	16	7 / 11
20 x 20	44	31	20	9 / 15
22 x 22	60	43	28	10 / 17
24 x 24	72	52	34	12 / 21
26 x 26	88	64	42	14 / 25
32 x 32	124	91	60	18 / 33
36 x 36	172	127	84	21 / 39
40 x 40	228	169	112	24 / 45
44 x 44	288	214	142	28 / 53
48 x 48	348	259	172	34 / 65
52 x 52	408	304	202	42 / 78
64 x 64	560	418	278	56 / 106
72 x 72	736	550	366	72 / 132
80 x 80	912	682	454	96 / 180
88 x 88	1152	862	574	112 / 212

96 x 96	1392	1042	694	136 / 260
104 x 104	1632	1222	814	168 / 318
120 x 120	2100	1573	1048	204 / 390
132 x 132	2608	1954	1302	248 / 472
144 x 144	3116	2335	1556	310 / 590

ECC 200 (rectangular)

# rows x # cols	max. # digits (numeric)	max. # chars (alpha-numeric)	max. # bytes (binary)	max. correctable errors / erasures
8 x 18	10	6	3	3 / 3
8 x 32	20	13	8	5 / 5
12 x 26	32	22	14	7 / 11
12 x 36	44	31	20	9 / 15
16 x 36	64	46	30	12 / 21
16 x 48	98	72	47	14 / 25

Example

These are samples of typical barcodes of this symbology (ECC 200 variant).
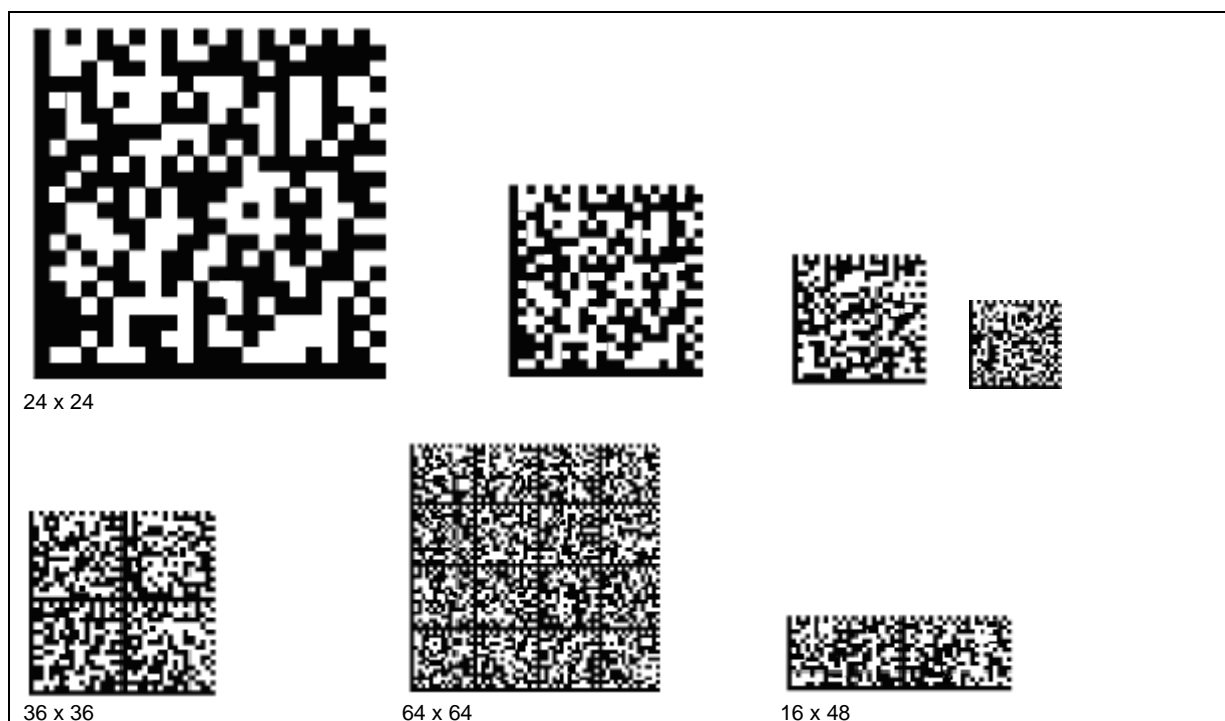
24 x 24

36 x 36   64 x 64   16 x 48

Further information

The specification of this symbology is maintained at:
  - AIM International: "ANSI/AIM BC11-ISS - DataMatrix"
  - ISO 16022 ;

Other sources of information can be found at <u>Appendix L</u>.

Applications

"Extended Channel Interpretation"
= A mechanism to enable other character sets & data interpretations.

"Structured Append"
= A mechanism to allow sequence of multiple (up to 16) symbols.

5.37.2 Character set (Alphabet)

Message character set (actual symbology **alphabet**)

The message character set consists of any byte sequence, e.g. ASCII characters.

Message data characters
Any binary information or ASCII text.

Message control characters
None.

Message length restrictions

Variable. Depends on the chosen ECC-xxx variant, the chosen layout (# rows x # columns), and the type of message data (numeric vs alpha-numeric vs binary). Refer to the table above.

5.37.3 Reference numbers

Not applicable.

5.37.4 Checksum methods

This symbology defines a Reed-Solomon ECC algorithm.

5.37.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	
Dimensionality:	2D

5.37.6 Encoding (I): The Logical encoding

This information is only necessary for troubleshooting.
Please contact Technical support.

Encoding/Compression methods:

encoding	description
- ASCII	1 alpha-numeric (lower 7-bit ASCII) char <u>or</u>
	2 numeric digits into 1 byte
- C40	3 alpha-numeric chars into 2 bytes (for numeric & upper-case)
- TEXT	3 alpha-numeric chars into 2 bytes (for numeric & lower-case)
- BASE256	(binary) bytes = (alpha-numeric) extended ASCII (8-bit) chars
- NONE	byte-wise (uncompressed)

5.37.7 Encoding (II): The Physical encoding

This information is only necessary for troubleshooting.
Please contact Technical support.

5.37.8 [SUPPORT ONLY]

5.38 Symbology (2D): **UPS Maxicode™**

5.38.1 Overview

This symbology is known as:
- UPS MaxiCode, - UPScode, - Maxicode

Usage
Application: parcel services, shipping/forwarding companies
Regional use: world-wide
Popularity: mainly by UPS

Remarks
Characteristics:
- 2D matrix code
- high information density : max. 100 ASCII chars per sq inch
- all ASCII
- built-in data compression
- by OmniPlanar; 1987/1989/1992 by UPS ("United Parcel Service")
- for parcel destination info for automatic sorting/routing/tracking

Example
These are samples of typical barcodes of this symbology.
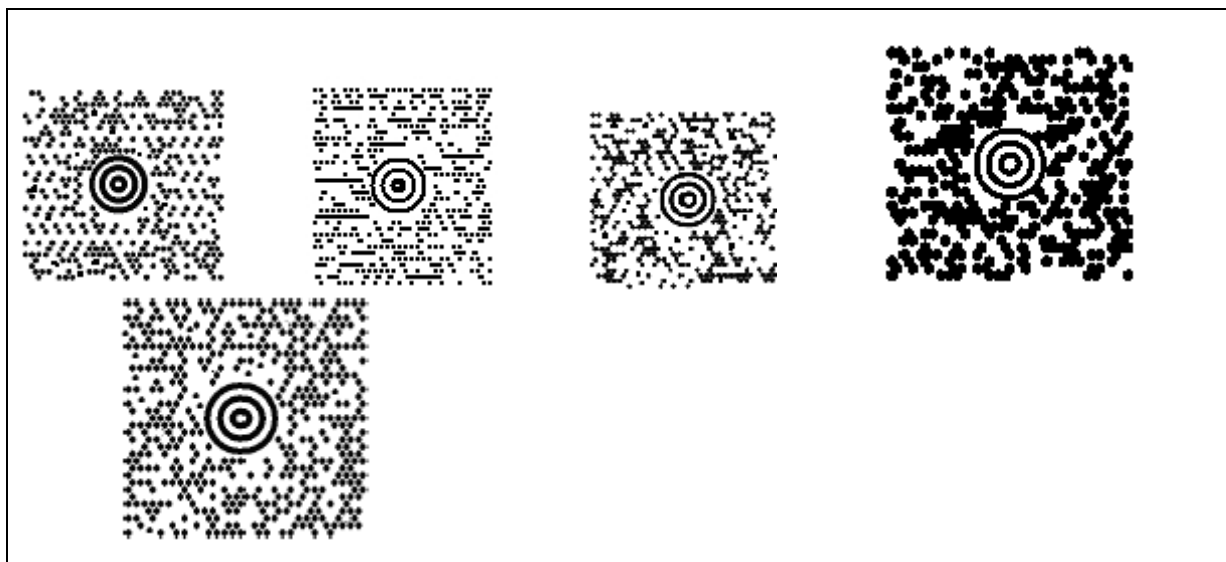
Further information

The specification((standardization)) of this symbology is maintained at:
- AIM International : "ANSI/AIM BC10-ISS - Maxicode"

Other sources of information can be found at Appendix L.

Applications

"Structured Message Append" = A mechanism to put information into a sequence of multiple symbols.

5.38.2 Character set (Alphabet)

Message character set (actual symbology **alphabet**)

The message character set consists of any binary information.

Message data characters
Any binary information.

Message control characters

None.

Message structure

- **primary** message   = "structured carrier msg" = package destination
- **secondary** message   = weight, etc.

All fields are required except where noted as optional [...].

<message> ::=
<**label number**> <FS> <**number of labels for shipment**> <FS> <**mode**> <FS> <**ship-to postal code**> <FS> <**ship-to country code**> <FS> <**class of service**> <FS> <ANSI message header> <Transportation Data Format Header> <**UPS Tracking Number**> <GS> <**SCAC**> <GS> [<**UPS Shipper Number**>] <GS> [<**Julian Day of Pickup**>] <GS> [<**Shipment ID**>] <GS> [<**Package # n/N**>] <GS> [<**Package Weight**>] <GS> [<**Address validation**>] <GS> [<**Ship-to Address**>] <GS> [<**Ship-to City**>] <GS> [<**Ship-to State**>] <ANSI message terminator>

tag / field	length (# chars)	values / type	sample
==== control codes			
<**FS**> -- (field separator)	1	"," (comma)	,
<**RS**> -- (record separator)	1 or 3	hex <1E> or "\RS"	\RS
<**GS**> -- (group separator)	1 or 3	hex <1D> or "\GS"	\GS
<**EOT**> -- (end-of-transmission)	1 or 4	hex <04> or "\EOT"	\EOT
<**ANSI message header**>	3 + x	"[)>" <RS>	[)>\RS
<**Transportation Data Format Header**>	2 + x + 2	"01" <GS> <yy>	01\GS96 or 01<1D>96
<**ANSI message terminator**>	x + x	<RS> <EOT>	\RS\EOT or <1E><04>
==== header fields			
<**label number**>	any	numeric	1
<**number of labels for shipment**>	any	numeric	1
==== primary msg fields	(= 60 bit)		
<**mode**>	1 (as 4 bit)	numeric (see *1)	2
<**ship-to postal code**>	if <mode>==2: = 5 or 9 digits if <mode>==3: = max. 6 chars	numeric alpha-num	152382802
<**ship-to country code**>	3	numeric (see ISO 3166)	840
<**class of service**>	3	numeric (see *2)	001
==== secondary msg fields	(total max 84 chars; => max. 43 chars for optional fields)		
<**UPS Tracking Number**>	10	alpha-num	1Z14647438
<**SCAC**> -- "Standard Carrier Alpha Code"	4	"UPSN" -- for production "UPSS" -- for test	UPSN
[<**UPS Shipper Number**>]	6	alpha-num	WX9031 or 410E1W
[<**Julian Day of Pickup**>]	3	num (001..366)	272
[<**Shipment ID**>]	0-30	alpha-num	1234567
[<**Package # n/N**>]	max. 7 = (1-3) + 1 + (1-3)	num "/" num -- package # in shipment total	1/3 or 001/003
[<**Package Weight**>]	3	num	15 or 015
[<**Address validation**>]	1	"Y" or "N"	Y
[<**Ship-to Address**>]	max. 36	alpha-num;	634 MAIN ST
[<**Ship-to City**>]	max. 21	alpha-num;	LITTLE ROCK
[<**Ship-to State**>]	2	alpha	AK

Note: (*1) (<mode>)
<mode> ::=
    | 2   = SCM (Structured Carrier Msg) with numeric postal code (for US domestic shipments)
    | 3   = SCM (Structured Carrier Msg) with alpha-numeric postal code (for international shipments)
    | 4   = general-purpose data with standard error correction (max. 90 chars)
    | 5   = general-purpose data with enhanced error correction (max. 74 chars)

Note: (*2) (<class of service>)
See UPS documentation.

5.38.3 Reference numbers

Not available.

5.38.4 Checksum methods

This symbology defines a Reed-Solomon ECC algorithm.
This allows the reconstructability of the symbol after up to 25% destruction.
There are 2 levels:
 - SEC = Standard Error Correction

- EEC = Enhanced Error Correction

5.38.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	
Dimensionality:	2D

- Length: 144 chars, max. 93 ASCII or 138 digits

- 144 codewords; 1 **codeword** = 6 hexagons

- max. 596 bits, incl. redundancy

5.38.6 Encoding (I): The Logical encoding

This information is only necessary for troubleshooting.
Please contact Technical support.

5.38.7 Encoding (II): The Physical encoding

This information is only necessary for troubleshooting.
Please contact Technical support.

5.38.8 [SUPPORT ONLY]

5.39 [SUPPORT ONLY]

5.40 Symbology (2D): **QR Code**

5.40.1 Overview

This symbology is known as:
- QR Code, QRcode, QR-Code, Standard QR code, ...

Usage

Application:	commercial tracking, entertainment & transport ticketing, product marking, in-store product labeling; storing addresses or URL e.g. on advertisements in magazines or on posters to link to sites with more information;
Regional use:	world-wide.
Popularity:	The most commonly used 2D barcode.

Remarks

The name refers to the usage ("**Q**uick **R**esponse Code").
A QR Code symbol can contain any numeric and alphanumeric data or Kanji text with high compression, and any binary data.
A matrix square-shaped high-density 2D symbology.
Developed 1994 in Japan by Denso Wave Corporation, a subsidiary of Toyota, originally for the automotive industry for tracking vehicles during the manufacturing process.
Very tolerant to dirt or mutilation, through built-in error correction capability.
Automatic data compression.
Full binary character set.
Can be read with CCD cameras.
License-free (patent rights owned by Denso Wave are not being exercised).

Example

These are samples of a typical symbol of this symbology.
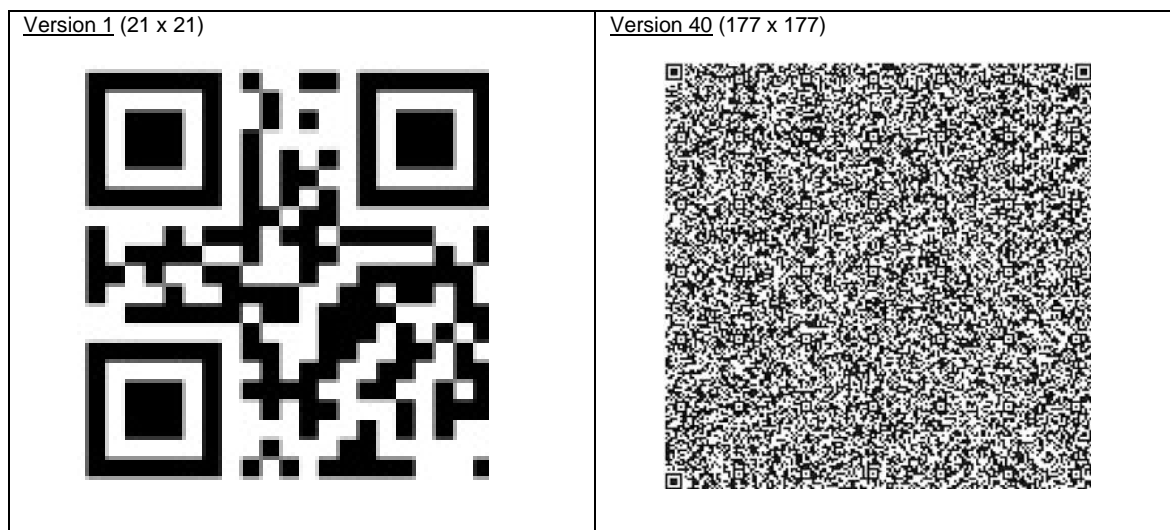
Version 1 (21 x 21)

Version 40 (177 x 177)

Further information

The specification of this symbology is maintained at:
- AIM International oct'1997 (ISS - QR Code)
- JIS X 0510
- JEIDA-55
- ISO/IEC 18004:2006

Other sources of information can be found at **www.qrcode.com** and Appendix L.

Applications

The above is referred to as "**Standard QR Code**".

"**Structured append**" allows to split a single symbol into up to 16 smaller ones.

"**Micro QR Code**" is designed to fit on small areas like printed circuit boards. It can encode max. 35 digits. It has only one position detect pattern. It exists in 4 versions with 11 x 11, 13 x 13, 15 x 15, and 17 x 17 # modules.

"**Swiss QR Code**" is a variant used in Switzerland for the A6 format "Zahlteil QR-Rechnung" (payment part) of a "QR-Rechnung" (QR-bill).
It uses fixed parameters: version 25 = 117 x 117 modules, Binary mode, ECC level M, and overall size 46 x 46 mm.
It has a fixed 7x7 mm size Swiss cross in the middle.
It needs a module size of at least 0.4 mm, and a quiet zone of at least 4 modules or 5 mm.

Version 1 of the specification has been obsoleted by version 2.
Besides the overall QR Code ECC, the "Reference" field has its own "mod 10 recursive" CHK.
This, as well as the CTX, should usually be provided by the calling application, not by the barcoding solution.
For the full specification refer to the document "Swiss Payment Standards 2018 - Swiss Implementation Guidelines QR-bill" ("ig-qr-bill-en Version 2.0 - 15.11.2018.pdf") by SIX.
Fields must be separated by <LF> or <CR+LF>, which must be omitted after the last field.

Example:

5.40.2 Character set (Alphabet)

Message character set

The message character set can consist of any binary information, i.e. a sequence of bytes (hex 00..FF).
In addition, there are 3 compression modes (Numeric, Alpha-numeric, Kanji), in which not all bytes are permitted.

Note:
For the ASCII control characters <00..1F> it is a priori not defined/clear how they can be passed to the encoding solution and from the barcoder reader, because they may be interpreted according to their control function rather than passed as mere binary data. Refer to corresponding documentation.

Message length restrictions

Refer to the table below.

5.40.3 Reference numbers

Refer to the official specification.

5.40.4 Checksum methods

This symbology defines a Reed-Solomon ECC algorithm.

5.40.5 Encoding scheme

A QR code symbol can be encoded in one of 40 "**versions**", named "Version 1 .. 40".
It is a square, consisting of 21 x 21 (version 1) .. 177 x 177 (version 40) **modules**, in steps of 4 modules, each of which is a little square either black or white.

There are 4 possible **encodings** (**compression modes**) of message data:

	Alphabet / Character set	Compression (1 code-word = 8 bit)
- **Numeric**	numeric / digits only { 0 .. 9 } = 10 digits;	compression factor = 2.4 ((3x8/10)) (3 digits into 10 bits) = (12 digits into 5 code-words)
- **Alpha-numeric**	digits + upper-case alpha + 9 special symbols: { 0-9 A-Z SP $ % * + - . / : } = 45 characters; -- no lower-case !	compression factor = 1.4545... ((2x8/11)) (2 characters into 11 bits) = (16 characters into 11 code-words)
- **Binary**	any 8-bit byte {hex 00..FF}	compression factor = 1 ((1x8/8)) none / uncompressed
- **Kanji**	(full-width Kana & Shift-JIS = 0x8140 - 9FFC, E040 - EAA4);	compression factor = 2 x 0.62 ((1x8/13)) (1 2-byte Kanji into 13 bits) = (8 Kanjis into 13 code-words)

Error correction capability

QR code defines these 4 **error correction levels** (**ECC levels**):

		ECC
- **L**	"Low"	~ 7 %
- **M**	"Medium"	~ 15 %
- **Q**	"Quality"	~ 25 %
- **H**	"High"	~ 30 %

ECC: Symbol is restorable if ... % damaged. Or: ... % of code-words can be restored if they are damaged.

Maximum data capacity

Encoding: \\ ECC level:		L	M	Q	H
- Numeric	(# digits):	7089	5596	3993	3057
- Alpha-numeric	(# chars):	4296	3391	2420	1852
- Binary	(# bytes):	2953	2331	1663	1273
- Kanji	(# chars):	1817	1435	1024	784

For version 40; for the other versions refer to the table below.

Sizing
  1. Version & hence # modules are determined from the message data.
  2. Module size is determined from the desired overall size, or vice versa.

5.40.6 Encoding (I): The Logical encoding

The following table shows the maximum number of characters that can be encoded in the given version, ECC level, and compression mode.
(Example: A version 19 symbol with ECC level H can accommodate max. 493 alpha-numeric characters.)

Version	Modules	ECC Level		Numeric	Alpha-num.	Binary	Kanji
1	21 x 21	L		41	25	17	10
		M		34	20	14	8
		Q		27	16	11	7
		H		17	10	7	4
2	25 x 25	L		77	47	32	20
		M		63	38	26	16
		Q		48	29	20	12
		H		34	20	14	8
3	29 x 29	L		127	77	53	32
		M		101	61	42	26
		Q		77	47	32	20
		H		58	35	24	15
4	33 x 33	L		187	114	78	48
		M		149	90	62	38
		Q		111	67	46	28
		H		82	50	34	21
5	37 x 37	L		255	154	106	65
		M		202	122	84	52
		Q		144	87	60	37
		H		106	64	44	27
6	41 x 41	L		322	195	134	82
		M		255	154	106	65
		Q		178	108	74	45
		H		139	84	58	36
7	45 x 45	L		370	224	154	95
		M		293	178	122	75
		Q		207	125	86	53
		H		154	93	64	39
8	49 x 49	L		461	279	192	118
		M		365	221	152	93
		Q		259	157	108	66
		H		202	122	84	52
9	53 x 53	L		552	335	230	141
		M		432	262	180	111
		Q		312	189	130	80
		H		235	143	98	60

#	Size	ECC						#	Size	ECC					
10	57 x 57	L		652	395	271	167	27	125 x 125	L		3514	2132	1465	902
		M		513	311	213	131			M		2701	1637	1125	692
		Q		364	221	151	93			Q		1933	1172	805	496
		H		288	174	119	74			H		1501	910	625	385
11	61 x 61	L		772	468	321	198	28	129 x 129	L		3669	2223	1528	940
		M		604	366	251	155			M		2857	1732	1190	732
		Q		427	259	177	109			Q		2085	1263	868	534
		H		331	200	137	85			H		1581	958	658	405
12	65 x 65	L		883	535	367	226	29	133 x 133	L		3909	2369	1628	1002
		M		691	419	287	177			M		3035	1839	1264	778
		Q		489	296	203	125			Q		2181	1322	908	559
		H		374	227	155	96			H		1677	1016	698	430
13	69 x 69	L		1022	619	425	262	30	137 x 137	L		4158	2520	1732	1066
		M		796	483	331	204			M		3289	1994	1370	843
		Q		580	352	241	149			Q		2358	1429	982	604
		H		427	259	177	109			H		1782	1080	742	457
14	73 x 73	L		1101	667	458	282	31	141 x 141	L		4417	2677	1840	1132
		M		871	528	362	223			M		3486	2113	1452	894
		Q		621	376	258	159			Q		2473	1499	1030	634
		H		468	283	194	120			H		1897	1150	790	486
15	77 x 77	L		1250	758	520	320	32	145 x 145	L		4686	2840	1952	1201
		M		991	600	412	254			M		3693	2238	1538	947
		Q		703	426	292	180			Q		2670	1618	1112	684
		H		530	321	220	136			H		2022	1226	842	518
16	81 x 81	L		1408	854	586	361	33	149 x 149	L		4965	3009	2068	1273
		M		1082	656	450	277			M		3909	2369	1628	1002
		Q		775	470	322	198			Q		2805	1700	1168	719
		H		602	365	250	154			H		2157	1307	898	553
17	85 x 85	L		1548	938	644	397	34	153 x 153	L		5253	3183	2188	1347
		M		1212	734	504	310			M		4134	2506	1722	1060
		Q		876	531	364	224			Q		2949	1787	1228	756
		H		674	408	280	173			H		2301	1394	958	590
18	89 x 89	L		1725	1046	718	442	35	157 x 157	L		5529	3351	2303	1417
		M		1346	816	560	345			M		4343	2632	1809	1113
		Q		948	574	394	243			Q		3081	1867	1283	790
		H		746	452	310	191			H		2361	1431	983	605
19	93 x 93	L		1903	1153	792	488	36	161 x 161	L		5836	3537	2431	1496
		M		1500	909	624	384			M		4588	2780	1911	1176
		Q		1063	644	442	272			Q		3244	1966	1351	832
		H		813	493	338	208			H		2524	1530	1051	647
20	97 x 97	L		2061	1249	858	528	37	165 x 165	L		6153	3729	2563	1577
		M		1600	970	666	410			M		4775	2894	1989	1224
		Q		1159	702	482	297			Q		3417	2071	1423	876
		H		919	557	382	235			H		2625	1591	1093	673
21	101 x 101	L		2232	1352	929	572	38	169 x 169	L		6479	3927	2699	1661
		M		1708	1035	711	438			M		5039	3054	2099	1292
		Q		1224	742	509	314			Q		3599	2181	1499	923
		H		969	587	403	248			H		2735	1658	1139	701
22	105 x 105	L		2409	1460	1003	618	39	173 x 173	L		6743	4087	2809	1729
		M		1872	1134	779	480			M		5313	3220	2213	1362
		Q		1358	823	565	348			Q		3791	2298	1579	972
		H		1056	640	439	270			H		2927	1774	1219	750
23	109 x 109	L		2620	1588	1091	672	40	177 x 177	L		7089	4296	2953	1817
		M		2059	1248	857	528			M		5596	3391	2331	1435
		Q		1468	890	611	376			Q		3993	2420	1663	1024
		H		1108	672	461	284			H		3057	1852	1273	784
24	113 x 113	L		2812	1704	1171	721								
		M		2188	1326	911	561								
		Q		1588	963	661	407								
		H		1228	744	511	315								
25	117 x 117	L		3057	1853	1273	784								
		M		2395	1451	997	614								
		Q		1718	1041	715	440								
		H		1286	779	535	330								
26	121 x 121	L		3283	1990	1367	842								
		M		2544	1542	1059	652								
		Q		1804	1094	751	462								
		H		1425	864	593	365								

For more information, please contact Technical support.

5.40.7 Encoding (II): The Physical encoding

The **low-level encoding** maps the codewords into a physical pattern of black or white module squares.

General structure

A QR code **symbol** is a square matrix symbol, consisting of little module squares.

It has 3 **position detection patterns** at 3 corners (lower-left, upper-left, upper-right).

For more information, please contact Technical support.

5.41 Symbology (2D): **Aztec Code**

5.41.1 Overview

This symbology is known as:
- Aztec Code
(A concrete symbol may be referred to as "an Aztec code".)

Invented 1995 by Welch-Allyn.
Uses less space than other matrix barcodes because it does not require a surrounding blank "quiet zone".
There are 2 variants: **compact** and **full**.
The symbol is built on a square grid with an Aztec pyramid shaped "**bull's-eye**" pattern at its centre for locating the code.
The term "**module**" refers to a single elementary-sized white or black square (which may consist of 1 or more pixels).
The central bull's-eye consists of **9×9** or **13×13** modules, respectively.
Together with a 1-module "**mode message**" row around it, which encodes basic coding parameters, this forms the "**core**", of thus **11×11** or **15×15** modules, respectively.
The message data is encoded around the core in concentric 2-module "**layers**", thus producing total sizes of 15×15, 19×19, etc.
The compact core (of 11 x 11 modules) may be surrounded by 1 to 4 layers, producing symbols from 15×15 through 27×27 modules.
The full core (of 15 x 15 modules) supports up to 32 layers, hence max. 151×151 modules, which can encode the maximum of 3832 digits, 3067 letters, or 1914 bytes of data.
The corners of the core include **orientation marks**, allowing the code to be read if rotated or reflected.

Usage
Application: - transport ticketing (e.g. Deutsche Bahn (German railway), IATA (aviation) boarding passes, ...);
         - invoices,
         - mobile apps

Regional use: World-wide.
Popularity: Growing; especially with mobile applications.

Example
Below are a few samples of this symbology.

Message: "stethos Systemhaus GmbH, Weimarer Str. 48, D-71065 Sindelfingen, Tel. +49-7031-860910"

ECC = min. **10%** => (Compact 4-layer = 27 x 27 modules)

ECC = min. **23%** => (Full 4-layer = 31 x 31 modules)

ECC = min. **36%** => (Full 5-layer = 37 x 37 modules)

ECC = min. **50%** => (Full 6-layer = 41 x 41 modules)

Further information

The specification of this symbology is maintained at:
- ISO/IEC 24778:2008

Other sources of information can be found at Appendix L.

5.41.2 Character set (Alphabet)

Message character set

Message data characters

Full ASCII set (7-bit: 0-127), plus any (8-bit) binary data.

Message control characters

There are 2 **escape codes**:

- **FNC1** = followed by an Application Identifier (AI), as of the GS1-128 standard (refer to the section "UCC/EAN-128" above).

- **ECI** = followed by a 6-digit Extended Channel Interpretation (ECI) code, which specifies the character set used to interpret the bytes following.
By default = ECI 000003: (0-127 = ANSI X3.4 (ASCII); 128-255 = ISO 8859-1 (Latin-1)).

Message length restrictions

The maximum number of elements that can be encoded in 1 symbol (Full, 32 layers => 151x151 modules) is:
(3832 digits) or (3067 letters) or (1914 bytes).

5.41.3 Reference codes

This information is only necessary for troubleshooting.
Please contact Technical support.

5.41.4 Checksum methods

This symbology defines a Reed-Solomon ECC algorithm.
There are 4 ECC levels: 10%, 23% (recommended), 36%, and 50% =: x%.
The number of check codewords then is = (symbol capacity) * x% + 3.

5.41.5 Encoding scheme

The following table summarizes **characteristics** of the encoding scheme of this symbology.
For the actual **encoding** for each symbology character, see the next section.

Physical encoding type:	
Dimensionality:	2D

5.41.6 Encoding (I): The Logical encoding

This information is only necessary for troubleshooting.
Please contact Technical support.

5.41.7 Encoding (II): The Physical encoding

This information is only necessary for troubleshooting.
Please contact Technical support.

5.41.8 [SUPPORT ONLY]

5.42 [SUPPORT ONLY]

6 OCR Fonts

<u>**Introduction**</u>

OCR (**"Optical Character Recognition"**) is a term collectively identifying technologies to automate conversion of text contained as an image on some surface (e.g. paper) into byte-coded text (e.g. ASCII), by using pattern recognition algorithms which rely on the sufficiently different appearance and characteristics of different symbols. While this may be attempted with any font or even handwritten text, there exist specific OCR fonts with particular high redundancy and maximum difference between any two symbols to minimize the chance of misreadings.

MICR (**"Magnetic Ink Character Recognition"**) is an analogous concept with recognition of characters printed with magnetizable toner with non-optical magnetic readers.

OCR fonts are human readable fonts whose characters can be well distinguished from each other and which can thus be recognized at a high recognition rate by a relatively simple reading technology. Other than barcodes they require a relatively friendly environment (no dirt, etc).

MICR fonts in addition use magnetic toner to allow for non-optical recognition.

Most commonly used are these:
- **OCR-A**
- **OCR-B**
- **MICR: CMC-7**
- **MICR: E-13B**

They are explained further in the following subsections.

For the <u>encoding</u> of the symbols as (ASCII) code-points in a specific font, refer to the User's Manual accompanying that font.

6.1 OCR-A

Alphabet
- **OCR-A** implements the standard ASCII symbol set {20, ..., 7E}.
- **OCR-A1** contains some additional characters used for banking.

Note: (OCR-A1 special characters)
Compared to the regular OCR-A, these non-ASCII characters below are OCR-A1 specific and used for banking / check printing.
While OCR-A implements the standard ASCII symbol set, OCR-A1 contains some additional characters used for banking, e.g. to print money transfers ("Überweisungen") and checks ("Schecks"). It is mainly used in Germany.

Symbol	Name	Unicode
● ∫	**<hook>**	U+2440
● Ч	**<fork>**	U+2442
● ⊓	**<inverted fork>**	U+2443
● ⊓	**<chair>**	U+2441
● ⫝	**<belt-buckle>**	U+2444

Specification
- ANSI X3.17-1981 (R2002), X3.17-2000, ANSI INCIT S17-1981 (R2000)
- ISO 1831; ISO 1073/11, ISO 1073/I-1976 (ISO 1073-1)
- DIN 66008

Samples

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9
 ! " # ⊊ % & ' ( ) * + ⌐ - . / : ; < = > ? @ ⟦ \ ⟧ ∧ { | } Ч ⫝
```

Usage
- print money transfers and bank checks
- mainly used in Germany, Scotland, Guayana, ...

Example

Example for printing checks with OCR-A1:

```
1234567890⊓00∫00
```

Measurements & Imaging conventions

Sizes by Usage

Which values of pitch/height need to be implemented, depends on the usage.
For check printing mostly only 10cpi / 12pt is required.

6.2 OCR-B

Alphabet
- **OCR-B** implements the standard ASCII symbol set {20, ..., 7E}.
- 121 characters

Specification
- ANSI X3.49-1982 (R2002), ANSI X3.49-1975 (R1982)
- ISO 1073/II-1976, ISO 1831, ECMA-11
- DIN EN 14603, DIN 66009
- since 1968

Samples

```
A B C D E F G H I J K L M N O R P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o r p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9
 !"#$%&'()*+,-./ :;<=>0?@ [\]^_`{|}~
¤ £ ¦ Ä Å Æ Ö Ø Ü   Ñ ä æ ø ÿ ß             § ¥
```

Usage
- for readable text in conjunction with barcode printing and for the address label of letter mailings, passports, serial tracking labels, postal mail, credit card imprints, cash registers, licence plates, ...
- in most European countries {Austria, Belgium, Denmark, Finland, France, Germany, Greece, Italy, Netherlands, Norway, Portugal, Switzerland, UK, ...}

Measurements & Imaging conventions

Sizes by Usage

Which values of pitch/height need to be implemented, depends on the usage.
For check printing mostly only 10cpi is required.

Example

```
   This is text in OCR B
```

6.3 MICR: CMC-7

This symbology is known as: CMC-7.

Alphabet
- 10 digits {0..9} + 5 special banking characters { "internal", "terminator", "amount", -, "routing"} + (optional) 26 uppercase letters {A..Z}

Note: (CMC-7 special characters)
These non-ASCII characters are CMC-7 specific and used for check printing.

Symbol	Name	Description
● ▐▌	**internal**	start of bank account information
● ▐▌	**terminator**	start of amount field
● ▐▌	**amount**	terminator for bank routing information
● ▐▌	(unused)	(not used)
● ▐▌	**routing**	institution which the check is drawn from

Specification
- ISO 1004, DIN 66007;
- since 1964

Encoding scheme

Each character consists of 7 vertical black lines.
Of a total of 8, 9, or 10 lines, 1, 2, or 3, respectively, are white.

Samples
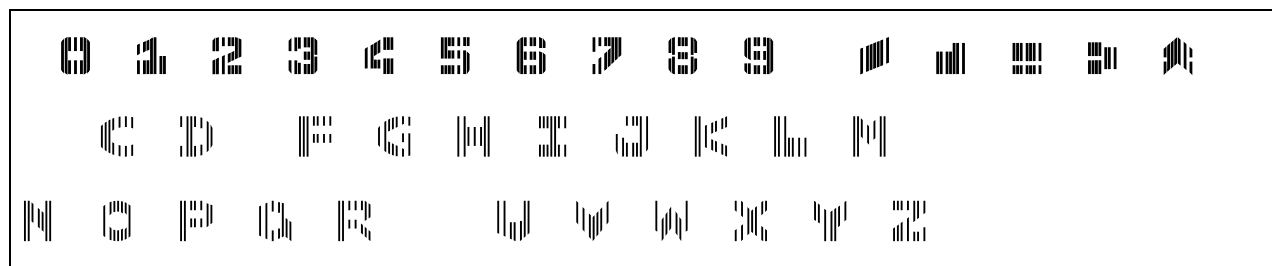
Usage/Application
- to print characters for magnetic recognition and optical character recognition systems in automated check processing
- in these countries:
  - France, Guadeloupe, Martinique
  - Spain and most Spanish speaking countries {Argentina, Bolivia, Chile, Ecuador, Mexico, Paraguay, Peru, ...}
  - Portugal and Brazil
  - some other European countries {Italy, Belgium, Netherlands, Denmark, Finland, Norway, Sweden}
  - Israel
  - Ivory Coast
  - Japan

6.4 MICR: E-13B

This symbology is known as: E-13B, E13B, E13-B.

Alphabet
- 10 digits { 0 .. 9 } + 4 special banking characters { transit, amount, on-us, dash }

Symbol	Name	Unicode	Description
● ▮▝	**Transit**	U+2446	Routing #; BSB, branch bank identification, institution where the check is drawn from; delimits (precedes/introduces & terminates) the bank routing # (bank branch routing transit number)
● ▖▘	**Amount**	U+2447	delimits (precedes/introduces & terminates) the transaction amount of the check;
● ▌▌▘	**On-us**	U+2448	delimits (precedes/introduces & terminates) the customer account number;
● ▮▮▮	**Dash**	U+2449	separator to delimits parts of numbers, e.g. inside routing number or account number;

Specification
- ANSI X9.27-2000 , ISO 2033, DIN 66226
- since 1958

Encoding scheme

Each character is based on a grid of 7 x 11 squares.

Samples
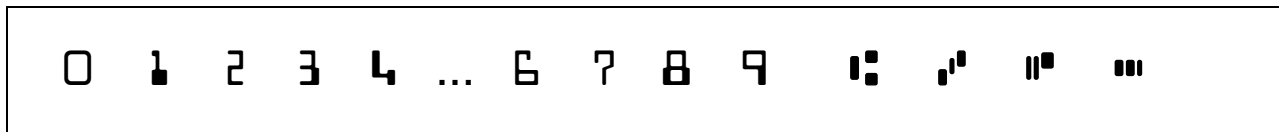
Usage/Application
- on bank checks and drafts for magnetic recognition and optical character recognition systems for check processing
- in these countries:
  - United States, Canada
  - Mexico, Puerto Rico, Panama
  - most of South America
  - UK
  - Iraq, Kuwait
  - Korea, Taiwan, Hongkong
  - India, Malaysia, Thailand, Singapore, Australia
  - and some other countries {...}

Example
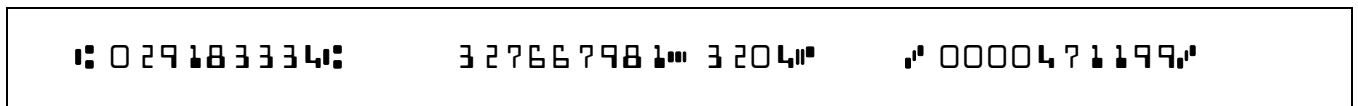
7 Appendices

The remainder of this document consists of several Appendices.

7.1 [SUPPORT ONLY]

7.2 [SUPPORT ONLY]

7.3 Appendix D: Device-specific information

Refer to the separate document "Device Information .pdf".

7.3.1 Appendices DPx: Device: Procedures

Refer to the separate document "Device Information .pdf".

7.3.2 Appendix DPD: (Procedure) Insert the DIMM

Refer to the separate document "Device Information .pdf".

7.3.3 Appendix DPF: (Procedure) Print out the PCL Font list and the HDD Directory list

Refer to the separate document "Device Information .pdf".

7.3.4 [SUPPORT ONLY]

7.3.5 Appendix DPS: (Procedure) Insert the SDcard

Refer to the separate document "Device Information .pdf".

7.4 Appendix G: Glossary

Terminology

The following table contains concise descriptions of general barcode related terminology as well as product-specific terminology.

Term	Explanation
	=== (Barcode / general) ===
1-D barcode	1-dimensional barcode. A rectangular area filled with a pattern of rectangles of different brightness encoding data. The dark rectangles are called "bars"; the gaps between the bars are called "spaces".
2-D barcode	2-dimensional barcode. A black-and-white pattern of arbitrary shape.
2-state code **4-state code**	(1-D) A form of --> Bar height encoding. A 2-state code uses 2 baselines and has 2 possible values. E.g. POSTNET. A 4-state code uses 3 baselines and has 4 possible values. E.g. KIX.
alphabet	See -> "character set".
annotated text	see "clear text"
(barcode) **application**	The concrete use of a (basic) symbology. Based on the basic symbology, with the following possible deviations: The character set may be extended. Additional check characters or a special checksum algorithm may be defined. Characters at certain positions may have a special interpretation (e.g. <FNC1> in EAN-128). The number of characters may be restricted.
application family	A family of similar or related applications.
bar	The dark element of a barcode. Usually a black rectangle. It has a certain width.
barcode	(1-D): See -> 1-D barcode. (2-D): See -> 2-D barcode.
barcode character	The character of a barcode font whose glyph image encodes one character.
barcode zone	The area containing the barcode surrounded by the Quiet zone and optional Clear text underneath.
bar-height encoding	(1-D) As a sequence of equidistant bars with different heights. Thus information is encoded not in the widths of the bars and spaces, but in the height of the bars. E.g. POSTNET, KIX, etc.
bar-width encoding	(1-D) As a sequence of bars of different widths, separated by narrow spaces. E.g. 2 of 5 Industrial.
basic symbology	The symbology on which an application is based.
binary representation	The representation of the logical encoding as a binary pattern of "0" and "1". In the case of a linear barcode, the "1" correspond to bars, the "0" correspond to spaces; this can be expressed more compactly in the -> width representation.
center guard pattern	Some intermediate non-data character of the barcode. (E.g. EAN-13.)
character	The smallest encodable quantity. A character can be either a start/stop character, a data(message) character, or a check character.
characters/inch (cpi)	Measure for the density of a barcode with a fixed number of modules.
character set	The set of encodable message characters (alphabet) of a symbology. E.g. numeric, alphanumeric, Full ASCII, etc.
check character	A character (from the message character set) which is obtained from the checksum. Usually the character whose reference value is the remainder of the checksum modulo N (where N is the size of the character set). Including it in the barcode as a check character (immediately before the [stop] character), allows for a certain degree of error detection or even error correction. A symbology may or may not require (one or more) check characters, as an option to increase the safety of reading.
check digit	The check character in the case of a numeric symbology.
checksum	A checksum is calculated from the values of all data characters, according to some algorithm. Usually all characters are assigned a value and weighted (i.e. multiplied by some weight factor) according to their position, and then summed up. For some type of barcodes a checksum is required, for others optional. Some have two check digits.
clear text	The human readable text version of the data printed together with the actual barcoded data, usually below the bars. Also called "annotated text" or "readable text".
code construction	The procedure of converting the original message data into a barcode image. It happens in a series of stages: Application, Logical/Symbology, Physical/Imaging.
continuous barcode	There are no inter-character spaces. Every space is part of the code and thus conveys information. The opposite of a -> discrete barcode.
control character	A -> message control character (e.g. <FNC1>, <startX>, etc) or a -> symbology control character (e.g. [startX]).
cpi	-> "characters per inch"
density	The classification of a barcode according to its module width (X). • Ultra-high density (UHD): X <= 0.19mm = 7.5 mil • High density: 0.19mm < X <= 0.24mm = 9.5 mil • Medium density: 0.24mm < X <= 0.30mm = 11.8 mil • Low density: 0.30mm < X <= 0.50mm = 19.7 mil • -- (only for long range/ distance): X > 0.50mm If the number of modules per character is fixed, the density for a symbology can also be expressed in

	terms of characters/inch (cpi).
discrete barcode	Every character starts and ends with a bar. The inter-character space is always narrow and does thus not convey any information. The opposite of a -> continuous barcode.
element	A bar or a space. Each element has a certain physical width.
encoding scheme	A scheme describing characteristics of the -> logical encoding and the -> physical encoding.
extended symbology	The character set may be extended by encoding additional characters as pairs or escaped characters of the basic symbology.
gap	-> "space"
guard bar	A symbology control character used by EANUPC. Sometimes also used as a synonym for the [start] / [stop] characters.
inter-character space/gap	The space between the last bar of a character and the first bar of the next character. For continuous symbologies this does not exist. For discrete symbologies its width is usually 1X.
linear barcode **linear encoding**	(1-D) A linear sequence of vertical bars of the same height separated by spaces. Information is encoded in the widths of both the bars and the spaces.
logical encoding	A scheme defining each symbology character as a logical pattern of "0" and "1". It doesn't specify how these are physically created.
matrix code	(2-D) A true 2-D barcode (potentially an arbitrary pattern of dots).
message	The original data to be barcoded. It consists of a sequence of -> message characters.
message character	A character of the message data. It can be an actual data character (e.g. "7", "A", "b", etc), or a message control character (e.g. <FNC1>, <startX>, etc).
message data	-> "message"
mil	Measure for the module width. 1 mil = 1/1000 inch.
mis-read	The barcode was wrongly decoded. E.g. substitution error.
module	The smallest element of a barcode. The width of each element can be expressed as a multiple of the module's width.
module width ("**X**")	The physical width of the smallest element (module). See also -> Density. It can be expressed in the unit "mil". E.g. Narrow bar = 1X.
multi-level barcode	Contains elements of different widths. These are a multiple of the module width.
N-level barcode	The widths of the elements are multiples 1,..,N of the module width.
N:W ratio **Narrow:Wide ratio**	The ratio of the (average) width of a narrow (smallest) element versus the (average) width of a wide (next thicker) element. Typically in the range 1:2.0 .. 3.0.
physical encoding	The specification how the logical encoding is implemented physically; e.g. in the widths or heights of bars and/or spaces.
pool	A collection of symbology characters assigned to each symbology message character. A symbology may define multiple pools which are to be used under different conditions. A character is encoded "in" a pool, depending on a certain condition.
pool pattern	The sequence of pools to be used for a sequence of message characters. E.g. a pattern "ABC" means that the 1st/2nd/3rd character is to be encoded in pool A/B/C.
pool scheme	A table assigning for each condition the pool pattern to be used.
quiet zone	Also "Space zone". The blank space in front of and behind the barcode. Needed for reading purposes.
read error	The barcode could not be decoded.
readable text	see "clear text"
recognition rate	Ratio of number of successful readings upon the first attempt, divided by the number of all attempted readings.
reference number	A numerical value associated to each element of the character set. It is used for checksum calculation. Usually numbered 0 .. (|character set| - 1).
self-checking barcode	Mis-read in any one character can be detected (even without check character) => The scan will fail.
self-correcting barcode	Mis-read in any one character can be detected and corrected (even without check character) => The scan will be successful.
space	The bright element of a barcode. The space between adjacent bars. Also known as "gap".
space-width encoding	(1-D) As a sequence of narrow bars, separated by spaces of different width. Information is encoded in the widths of the spaces, the bars are just separators. E.g. German Post encoding of the address on letter mail.
space zone	-> "quiet zone".
stacked barcode	(2-D) A 2-dimensional barcode which is actually just like a long 1-D barcode wrapped into a rectangle. Also called **multi-row** barcode.
start character	The first (non-data) character of a barcode. In order to distinguish barcodes of different symbologies, a barcode always starts with a special start bar combination.
stop character	The last (non-data) character of a barcode. In order to distinguish barcodes of different symbologies, a

	barcode always ends with a special stop bar combination. Some symbologies have the same start and stop character.
substitution error	A character has been mis-read as a different character. This can be largely avoided by the use of check digits.
symbology	The specification of a type of barcodes, defining the character set and a logical encoding scheme.
symbology character	An abstract definition of small independent non-composed portions of the barcodes. One can distinguish (symbology) data and control characters. It can be implemented as a barcode character using a font, or as a sequence of rectangle draw commands in some PDL.
symbology control character	A non-data symbology character. E.g. [start], [stop], etc.
symbology data character	A symbology character essentially corresponding to a message data character or pairs thereof. E.g. [7], [A], [b], [FNC1], etc.
symbology family	A family of similar or related symbologies. E.g. EAN/UPC.
weight	The factor with which the value of a particular character is multiplied when calculating the checksum.
width representation	For linear barcodes, a pattern of {1,2,3,4} or {1,2} or {N,W}, indicating alternatingly the widths of bars and spaces. This can be 1:1 translated into the binary representation of the logical encoding.

=== Acronyms ===	
	=== (general) ===
dec	The acronym for "decimal".
DIMM	The acronym for "Dual In-line Memory Module".
dto	"dito" = "same as previous/above"
HDD	The acronym for "Hard Disk (Drive)". Also used here for "Hard Disk Download".
hex	The acronym for "hexadecimal".
RAM	The acronym for "Random Access Memory".
SDcard	The acronym for "Secure Digital Card".
TRM	"Technical Reference Manual - Barcode & OCR Printing -" (this document)
	=== (Barcode / general) ===
AIM	The acronym for "Automatic Identification Manufacturers". A company in Pittsburgh, Pennsylvania, USA.
CEN	The acronym for "Commission for European Normalization".
CHK	An abbreviation of "checksum" or "check digit" (if clear from context) or "check character" or "checksum method".
CTX	An abbreviation of "clear text" (human readable text), or "clear text style" (if clear from context).
DIN	The acronym for "Deutsche Industrie-Norm" (German industry standard). (www.din.de)
EAN	The acronym for "European Article Number". Governed by "EAN International" in Brussels, Belgium.
ECC	The acronym for "Error Checking (Detection) & Correction".
EN	European Standard (Norm)
FCC	The acronym for "Format Control Code" (Australian Post).
GLN	The acronym for "Global Location Number". Identifies a company/organization or a physical location or a functional organization entity.
GTIN	The acronym for "Global Trade Item Number".
HIBC	The acronym for "Health Industry Bar Code".
IAN	The acronym for "International Article Numbering".
ISS	The acronym for "International Symbology Specification". Governed by AIM.
JAN	The acronym for "Japanese Article Number". Governed by "Japan Industrial Standard Organization" (JIS) in Tokyo, Japan.
MICR	The acronym for "Magnetic Ink Character Recognition".
OCR	The acronym for "Optical Character Recognition".
SSCC	The acronym for "Application Standard for Shipping Container Codes" or "Serial Shipping Container Code".
UCC	The acronym for "Uniform Code Council".
UPC	The acronym for "Uniform Product Code". Governed by the "Uniform (Product) Code Council" in Dayton, Ohio, USA.
USS	The acronym for "Uniform Symbology Specification". Governed by AIM.
WPC	The acronym for "World Product Code".
ZIP (code)	The USPS term for postal code. The acronym of "Zone Improvement Plan".

Notation

Notation	Meaning
<ddd>	decimal notation for ASCII codes of font characters or message characters

<xx>	hexadecimal notation for ASCII codes of font characters or message characters

Special ASCII characters

Notation	Name	(hex)	(dec)	Meaning / Usage / Remarks
<esc>	Escape	<1B>	<027>	in PCL commands: start of command
<SI>	Shift In	<0F>	<015>	in PCL commands: select primary font
<SO>	Shift Out	<0E>	<014>	in PCL commands: select secondary font
<CR>	Carriage Return	<0D>	<013>	line feed
<LF>	Line Feed	<0A>	<010>	line feed
<FF>	Form Feed	<0C>	<012>	page feed
<SP>	Space	<20>	<032>	blank space
<DEL>	Delete	<7F>	<127>	DEL character
<UEL>	UEL			"Universal Exit Language" ("<esc>%-12345X")

Special non-ASCII message characters

Notation	Meaning / Usage / Remarks
<FNCx>	Code 128 control character (x=A|B|C)
<Shift>	Code 128 control character
<CodeX>	Code 128 control character (X=A|B|C)

7.5 Appendix L: Literature / Bibliography

For **platform-specific** information, see the documentation provided separately.
- SAP systems (R/3, mySAP ERP): => "Volume 2B"

For **solution/product-specific** information, refer to the separately provided "User's Manual" of the product.

[Fonts] and [Printer languages]

[ ] HP PCL5e Printer Language Technical Reference Manual
[ ] HP PCL5 Comparison Guide
[ ] HP PCL5c Color Technical Reference Manual
[ ] HP PJL Technical Reference Manual

[Barcodes]

[ ] datalogic: Strichcode-Fibel (2nd edition) -- currently only available in German language
[ ] www.barcodeisland.com
[ ] www.datalogic.de
[ ] www.symbol.de
[ ] www.strichcode.com
[ ] www.sick.de
[ ] www.ruoss-kistler.ch/handel/hilfe/barcode_lexikon.htm

[Standardization organizations/institutes]

Organizat.	Address

[ ] AIM	[USA]: ISS, USS
	AIM USA
	  634 Alpha Drive
	  Pittsburgh, PA 15238-2802
	  USA
	  Phone: +1-(412) 963-8588; Fax: +1- (412) 963-8753
	
	[Germany]: EN xxx
	AIM Deutschland (AIM-D) e.V.
	  Herr Kretz
	  Akazienweg 26
	  D-68623 Lampertheim
	  Germany
	  Phone: 06206-13177
	  Email: (aim-d@t-online.de)

[ ] ANSI	[USA]:
	American National Standards Institute (ANSI)
	  Sales Dept.
	  1430 Broadway
	  New York, NY 10018
	  USA
	  Phone: +1-(212)642-4900

[ ] UPC	[USA]: UPC, UCC
	Uniform Product Code Council, Inc. (UPC)
	  7051 Corporate Way, Suite 201
	  Dayton, OH 45459-4294
	  USA
	  Phone: +1-(513)435-3870

[ ] JIS	[Japan]: JAN
	Japan Industrial Standard Organization (JIS)
	  The Distribution Code Center, No.2 TOC-Building
	  7-23-1, Nishigotanda, Shinagawa-ku, Tokyo 141

[ ] EAN	[Europe]: EAN
	EAN International
	  Rue de Colonies 54
	  BTE 8
	  B-1000 Bruxelles/Brussels

	Belgium Phone: +32-(02) 2187675 WWW: www.ean.be
[ ] CEN	[Europe]: Comité Européen de Normalisation / European Committee for Standardization 　WWW: www.cenorm.be
[ ] DIN	[Germany]: DIN, DIN EN Deutsches Institut für Normung e.V. (DIN) 　Zweigstelle Köln: Kamekestraße 8, D-50672 Köln; Phone: +49-(0)221-5713-0 　Zweigstelle Berlin: Burggrafenstraße 6, D-10787 Berlin; Phone: +49-(0)30-2601-0 　WWW: www.din.de, www.beuth.de
[ ] CCG	[Germany]: EAN Centrale für Coorganisation GmbH (CCG) 　Postfach 19 04 24 　D-50501 Köln/Cologne 　Germany 　Phone:　+49-(0)221-94714-0 　Fax:　　+49-(0)221-94714-990 　Email:　<info@ccg.de> 　WWW:　www.ccg.de

7.6 Appendix P: Programming tools/information

Font measurement unit definitions

These units are used in font measuring.

Unit	Definition	Remarks
1 **inch**	:= 25.4 mm	
1 **mil**	:= 1/1000 inch	-- used for module width
1 PCL **point** (**pt**)	:= 1/72 inch	-- used for bitmapped fonts design & positioning & scalable TrueType fonts
1 **deci-pt**	:= 0.1 pt = 1/720 inch	
1 **twip**	:= 1/1440 inch = 1/20 pt = 1/2 deci-pt	-- used for font metrics information
1 **typographic point** 1 **CG (CompuGraphic) point**	:= 1/72.307 inch	-- used for scalable fonts design (e.g. Intellifont, Compugraphic)
1 **dot** (@ 600 dpi)	:= 1/600 inch	-- used on physical printout (depending on resolution)

Font measurement unit conversion table

For the most frequently used units, here is the resulting conversion table.

to: from:	... mm	... inch	... mil	... pt	... deci-pt	... twips	... dots (@ 600 dpi)
1 **mm** =	1	0.03937	39.37	2.8346...	28.346...	56.692...	23.622...
1 **inch** =	25.4	1	1000	72	720	1440	600
1 **mil** =	0.0254	0.001	1	0.072	0.072	0.144	0.6
1 **pt** =	0.35277...	0.01388...	13.88...	1	10	20	8.333...
1 **deci-pt** =	0.035277...	0.001388...	1.388...	0.1	1	2	0.8333...
1 **twip** =	0.0176385...	0.00069444..	0.69444...	0.05	0.5	1	0.41666...
1 **dot** (@ 600 dpi) =	0.042333...	0.001666...	1.666...	0.12	1.2	2.4	1

SBCS code points

The **Hex/Dec conversion table** allows you to quickly convert Hex and Dec numbers.
For the sake of clarity, Hex numbers are always written as 2 digits <xx> and Dec numbers as 3 digits <ddd>.

The **ASCII table** visualizes the standard lower ASCII half, and the upper half using the character set of the Windows 1252 codepage.

Hex/Dec conversion table

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015
1x	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031
2x	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047
3x	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063
4x	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079
5x	080	081	082	083	084	085	086	087	088	089	090	091	092	093	094	095
6x	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111
7x	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8x	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9x	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
Ax	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
Bx	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
Cx	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
Dx	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
Ex	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
Fx	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

ASCII table (code page = 1252 "Windows Latin-1")

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	€	NOT USED	,	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	NOT USED	NOT USED	NOT USED
9x	NOT USED	'	'	"	"	•	–	—	~	™	š	›	œ	NOT USED	NOT USED	Ÿ
Ax		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	%	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

(end)