# Barcode Writer in Pure PostScript

http://bwipp.terryburton.co.uk

Barcode Writer in Pure Postscript is an award-winning open source barcode maker that facilitates the printing of all major barcode symbologies entirely within level 2 PostScript, ideal for variable data printing. The complete process of generating printed barcodes is performed entirely within the printer (or print system) so that it is no longer the responsibility of your application or a library. There is no need for any barcode fonts and the flexibility offered by direct PostScript means you can avoid re-implementing barcode generator code or migrating to new libraries whenever your project language needs change.

The project homepage is at http://bwipp.terryburton.co.uk.

This is the main resource for the project providing the latest downloads of code and documentation, as well as access to the support and development mailing list.

To make it as easy as possible to incorporate this project into your own systems, whether they be freely available or proprietary, the software is licensed under the permissive MIT/X-Consortium License.

This documentation is auto-generated from the BWIPP wiki at https://github.com/bwipp/postscriptbarcode/wiki using the pandoc-based build system in the `wikidocs/__pandoc/` directory of the BWIPP repository: https://github.com/bwipp/postscriptbarcode.git

# Contents

# Chapter 1

# Barcode Writer in Pure PostScript

Useful links:

- Homepage: http://bwipp.terryburton.co.uk
- Documentation: https://github.com/bwipp/postscriptbarcode/wiki
- Documentation in PDF format for print: http://goo.gl/PBFNbv
- Download: https://github.com/bwipp/postscriptbarcode/releases/latest
- Source: https://github.com/bwipp/postscriptbarcode.git
- Issue tracker: https://github.com/bwipp/postscriptbarcode/issues
- Mailing list: http://groups.google.co.uk/group/postscriptbarcode
- Presentation: Slides: http://goo.gl/WqYB6A Materials: http://goo.gl/dth54z

Barcode Writer in Pure Postscript (BWIPP) generates all barcode formats entirely within PostScript so that the process of converting the input data into the printed output can be performed by the printer or RIP itself. This is ideal for variable data printing (VDP) and avoids the need to re-implement the barcode generation process whenever your language needs change.

Since this resource is written in PostScript and interpreted within the virtual machine of a printer it is compatible with any operating system and hardware platform.

It makes including any barcode within a PostScript document as simple as inserting the following directive:

```
0 0 moveto (978-1-56581-231-4) (includetext)
/isbn /uk.co.terryburton.bwipp findresource exec
```

There is a web-based demonstration of the project here:

https://the-burtons.xyz/barcode-generator/

This project is dedicated to the memory of Craig K. Harmon.

## “Flavours” of Named Resources

BWIPP is essentially a set of generic PostScript Level 2 named resources that are provided in four flavours for ease of use. The one to use depends on how you intend to deploy the library.

- “Packaged” or “unpackaged”: The named resources have been packaged for DSC conformance, portability and ease of distribution. You will most likely want to use a packaged flavour in production, however the unpackaged versions of the resources are useful for understanding the code, developing the library and debugging.
- “Separate files” or “monolithic”: The resource is provided as separate files that are formatted for direct use by Adobe Distiller, GhostScript, a printer hard disk or a document manager. The monolithic flavours contain all of the resources in a single file that is suitable for inclusion in the Prolog section of a each PostScript document or installing to a printer’s initial job VM to provide persistence between jobs until the device is reset.

This leads to the following set of four files.

For production use:

- `postscriptbarcode-packaged-resource` – Packaged; Separate files.
- `postscriptbarcode-monolithic-package` – Packaged; Monolithic file.

For BWIPP development:

- `postscriptbarcode-resource` – Unpackaged; Seperate files.
- `postscriptbarcode-monolithic` – Unpackaged; Monolithic file.

## Downloading

You can download prepared packages and the sources from here:

https://github.com/bwipp/postscriptbarcode/releases/latest

Alternatively you can get and build the latest from version control:

```
git clone https://github.com/bwipp/postscriptbarcode.git
cd postscriptbarcode
make
```

The flavours are built into subdirectories of the `build/` directory.

The build requirements are Perl, GNU Make and GhostScript.

# Chapter 2

# Quick Guide

Using Barcode Writer in Pure PostScript requires only some basic PostScript knowledge that is easily learned by experimentation. If you do not want to get your hands messy playing with PostScript then you can use one of the project's frontends which hide many of the details.

The best way to get familiar with using the code is to download the *monolithic* flavour of the latest release and open the `barcode_with_sample.ps` file with a text editor.

This file consists of the following sections:

- A PostScript language indicator beginning `%!PS`.
- Comments as lines beginning `%`.
- A definition for the `uk.co.terryburton.bwipp` category of named resources.
- A small set of named resource definitions for the renderers which generate the graphical output for the symbols, delimited by `% --BEGIN RENDERER ...--` and `% --END RENDERER ...--`.
- A large set of named resource definitions for the encoders which convert the input into a data structure suitable for printing using a PostScript renderer or importing into a frontend application, delimited by `% --BEGIN ENCODER ...--` and `% --END ENCODER ...--`.
- A set of sample barcode invocations delimited by `% --BEGIN SAMPLE--` and `% --END SAMPLE--`.

This is one example from the samples:

```
150 750 moveto (0123456789) (includetext height=0.75)
/interleaved2of5 /uk.co.terryburton.bwipp findresource exec
```

The meaning of each component of the invocation is as follows:

```
150 750 moveto             % The position of the symbol on the canvas
(0123456789)               % The data field: Data content represented by the symbol
(includetext height=0.75)  % The options field: Properties of the symbol
/interleaved2of5           % The type of barcode, often called the "symbology"
/uk.co.terryburton.bwipp findresource exec  % A call to plot the symbol on the canvas
```

The acceptable contents of the data field varies between symbologies as defined in the symbology reference.

The acceptable contents of the options field is for the most part common across all of the symbologies as defined in the options reference, however encoder-specific options do exist in many cases and the default values of options vary across symbologies.

Using the references mentioned above you should now be able to experiment by carefully amending the sample section of the file and observing the effect on the graphical output.

You will want to view the result of your changes regularly (since bugs may be hard to track down once introduced) either by using a software PostScript interpreter alongside a viewer or by sending the file to a PostScript-enabled printer. Alternatively you can use the web-based generator.

- GhostScript is an open source PostScript interpreter that is available for both Windows and Linux.
- Adobe Distiller is a commercial PostScript interpreter that is available for Windows and MacOS.
- gsview is a viewer for PostScript files on Windows which requires that GhostScript be installed.
- gv is a viewer for PostScript files on Linux which requires that GhostScript be installed.
- The Preview application on Mac OS X is able to view PostScript files.

- Most laser printers have native support for PostScript. Look for either Adobe PostScript Level 2 or Adobe PostScript 3 compatibility.
- CUPS, the Common Unix Printing System, adds PostScript support for non-PostScript printers by filtering PostScript documents through GhostScript.

To directly print a file to an installed, PostScript-enabled printer in Windows by printer name use the following command:

```
PRINT [/D:device] barcode_with_sample.ps
```

Alternatively for a printer attached directly to the first parallel port:

```
COPY /B barcode_with_sample.ps LPT1:
```

To directly print a file to a PostScript-enabled printer in Linux use the following command:

```
lpr -Pdevice -o raw barcode_with_sample.ps
```

Once you are comfortable with amending the `barcode_with_sample.ps` file you may want to simplify the file by removing definitions for barcode formats that you do not require bearing in mind the following points:

- You need only include the named resource definitions for the symbologies of the symbols that you are actually intending to create and must include any dependencies as specified in the resource file by the `% --REQUIRES ...` metadata. Examining the contents of the PS files created by the web-based generator at https://the-burtons.xyz/barcode-generator/ illustrates this point. If you have downloaded the source distribution of BWIPP you can create a standalone file containing only the resources required for a particular symbology by running something like `make build/standalone/code39.ps` or `make build/standalone_package/code39.ps`.
- If you intend to create an application whose purpose is to generate documents containing a variety of barcodes it is suggested that you start with the monolithic `barcode.ps` file and use the `% --BEGIN/END ENCODER ...--` and `% --BEGIN/END RENDERER ...--` delimiters to extract the relevant named resource definitions into your documents. This will allow you to simply update your project to the latest version of the BWIPP resource by just replacing your `barcode.ps` with the latest version.
- BWIPP includes a C library with bindings for various languages specifically for this purpose.
- More information is available in the Developer Notes.

# Chapter 3

# Monolithic Flavours

The monolithic `barcode.ps` file provides Barcode Writer in Pure PostScript as generic PostScript Level 2 named resources shipped in a single file for ease of inclusion within the Prolog section of a PostScript document template or for installing into a printer's initial job VM.

Prepared tarballs of BWIPP packages into the monolithic flavours are available from https://github.com/bwipp/postscriptbarcode/releases/latest with filenames such as `postscriptbarcode-monolithic` and `postscriptbarcode-monolithic-package`. Alternatively you can build these flavours from source with `make monolithic` or `make monolithic_package`.

## Inclusion Within the Prolog Section of a Document

An application will first include the contents of `barcode.ps` in the Prolog section of a PostScript file and then generate code like the following.

In the file's `Setup` or `PageSetup` section:

```
/qrcode dup /uk.co.terryburton.bwipp findresource def
```

and in the page description where a barcode is needed:

```
0 0 moveto (BWIPP) (eclevel=M) qrcode
```

If the application needs to import the resource under a different name to avoid a conflict, then the setup could be:

```
/foo /qrcode /uk.co.terryburton.bwipp findresource def
```

followed by:

```
0 0 moveto (BWIPP1) (eclevel=M) foo
0 0 moveto (BWIPP2) (eclevel=M) foo
...
```

(The above is analogous to `from uk.co.terryburton.bwipp import qrcode as foo` in other languages.)

Or, to generate a few barcodes with no setup section or local name at all:

```
0 0 moveto (BWIPP) (eclevel=M) /qrcode /uk.co.terryburton.bwipp findresource exec
```

This technique also reduces the possibility of namespace collision when using the library's procedures with other code.

## Installing to a Printer Initial Job VM

Send barcode.ps to the printer with the line `true () startjob` added at the top where the parentheses contain the printer's `startjob` password.

The named resources will remain available between jobs but will not persist accoss power cycles.

# Chapter 4

# Named Resource Flavours

The contents of the `Resource` directory provides Barcode Writer in Pure PostScript as generic PostScript Level 2 named resources split into seperate files structured for ease of deployment.

This standard delivery mechanism allows BWIPP resources to be added to a PostScript virtual machine's resource search path, or pre-downloaded to a printer's memory or permanent storage, or supplied by a document manager, all without any change in the code an application generates to use the resources.

Prepared tarballs of BWIPP packaged into the named resource flavours are available from https://github.com/bwipp/postscriptbarcode/releases/latest with filenames such as `postscriptbarcode-resource` and `postscriptbarcode-packaged-resource`. Alternatively you can build these flavours from source with `make resource` or `make packaged_resource`.

## Deploying the Named Resource

An application or administrator must first make the BWIPP resources available to the print system as described for a variety of situations below. An application will then generate code like the following.

In the file's `Setup` or `PageSetup` section:

```
/qrcode dup /uk.co.terryburton.bwipp findresource def
```

and in the page description where a barcode is needed:

```
0 0 moveto (BWIPP) (eclevel=M) qrcode
```

If the application needs to import the resource under a different name to avoid a conflict, then the setup could be:

```
/foo /qrcode /uk.co.terryburton.bwipp findresource def
```

followed by:

```
0 0 moveto (BWIPP1) (eclevel=M) foo
0 0 moveto (BWIPP2) (eclevel=M) foo
...
```

(The above is analogous to `from uk.co.terryburton.bwipp import qrcode as foo` in other languages.)

Or, to generate a few barcodes with no setup section or local name at all:

```
0 0 moveto (BWIPP) (eclevel=M) /qrcode /uk.co.terryburton.bwipp findresource exec
```

This technique also reduces the possibility of namespace collision when using the library's procedures with other code.

If the definitions for the routines that generate and render the barcode are not already resident in memory then they will be fetched from a standard resource location in a way that is transparent to the user.

### GhostScript

Unpack the contents of the Resource directory to somewhere accessible to the application.

Specify the location of the Resource files using the `-I` or `-sGenericResourceDir` parameters. Where the resource search path defaults to `./Resource` (or equivalent) you can omit this parameter when running GhostScript from the location of the Resource files.

Example for Windows users:

```
gswin64c.exe -dSAFER -I%rom%Resource/;C:/bwipp/Resource/ "C:/create-barcode.ps"
```

### Adobe Distiller

Unpack the contents of the Resource directory to somewhere accessible to the application. Ensure that `PSRESOURCEPATH` contains the directory containing the `.upr` file when Distiller is run. The contents should be a list of directories separated by colons, to be searched in order with two consecutive colons to indicate where the default location should fall within the search order.

### Printer Hard Disk

If a printer with a hard disk option is used, the resources can be downloaded once and remain available across power cycles. Resources can be downloaded with a vendor-specific tool, or by sending them to the printer with a snippet of PostScript at the top that queries the printer for the correct file name and creates the file.

### PostScript Document Manager

Unpack the contents of the Resource directory to somewhere accessible to your document manager software then include the `%%DocumentNeededResources` and `%%IncludeResource` DSC comments at the appropriate locations within your PostScript output. The document manager software can be configured to transparently insert the requested resources as necessary.

Any specific instuctions for common document manager software are welcome.

# Chapter 5

# Symbology Reference

## Point of Sale

### EAN-13

**EAN-13** is an extension of the UPC-A barcode symbology that usually carries a GTIN-13. It was designed by the International Article Numbering Association in 1976 for identification of retail goods at point of sale outside of the US.

Also known as: EAN, UCC-13, European Article Number, International Article Number, JAN, JAN-13, IAN, WPC, SAAN, UCCET, ABAC, BCCI, ICA, MANA, KANC, ANA, ANC.

Variants:

- EAN-13+2 is an extension of EAN-13 that includes a two-digit add-on.
- EAN-13+5 is an extension of EAN-13 that includes a five-digit add-on.
- EAN-99 is a special form of EAN-13 starting with `99` that is used as an in-store coupon.
- EAN-8 is a barcode symbology derived from EAN-13 that is designed for small packaging. It uses a distinct numbering system based on GTIN-8.
- ISBN is a variant of EAN-13 used to identify books.
- ISMN is a variant of EAN-13 used to identify printed music.
- ISSN is a variant of EAN-13 used to identify periodicals.
- EAN-13 Composite is a variant of EAN-13 that should be used when a CC-A or CC-B GS1 Composite 2D component is required.

Standards: ISO/IEC 15420, BS EN 797, GS1 General Specifications.

#### Data and Options

- The data field for a EAN-13 may contain twelve or thirteen digits, optionally followed by a space then two or five digits if an EAN-2 or EAN-5 add-on is required.
- If twelve digits of primary data are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.
- The **includetext** option should normally be supplied.
- The **guardwhitespace** option enables the display of whitespace guard marks.

#### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    9771473968012
Options: includetext guardwhitespace
Encoder: ean13
```

```
Data:    977147396801
Options: includetext guardwhitespace
```

```
Encoder: ean13
```


A symbol that includes a five-digit add-on:

```
Data:    9771473968012 54499
Options: includetext guardwhitespace
Encoder: ean13
```


## EAN-8

**EAN-8** is derived from the EAN-13 barcode symbology and is designed for small packaging. It usually carries a GTIN-8.

Also known as: UCC-8, JAN-8.

Variants:

- EAN-8+2 is an extension of EAN-8 that includes a two-digit add-on.
- EAN-8+5 is an extension of EAN-8 that includes a five-digit add-on.
- EAN-Velocity is a special form of EAN-8 starting with `0` that is used for in-store coupons.
- EAN-13 is a longer variant of EAN-8 which has a distinct number system based on GTIN-13.
- EAN-8 Composite is a variant of EAN-8 that should be used when a CC-A or CC-B GS1 composite 2D component is required.

Standards: ISO/IEC 15420, BS EN 797, GS1 General Specifications.

### Data and Options

- The data field takes either seven or eight digits, optionally followed by a space then two or five digits if an EAN-2 or EAN-5 add-on is required.
- If seven digits of primary data are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.
- The **includetext** option should normally be supplied.
- The **guardwhitespace** option enables the display of white space guard marks.

### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    01335583
Options: includetext
Encoder: ean8
```

```
Data:    0133558
Options: includetext
Encoder: ean8
```


Truncated with white space guards:

```
Data:    01335583
Options: includetext height=0.5 guardwhitespace
Encoder: ean8
```


## UPC-A

The **UPC-A** barcode symbology is used for identification of retail goods at point of sale inside of the US. It usually carries a GTIN-12.

Also known as: UPC, UCC-12, Universal Product Code.

Variants:

- UPC-A+2 is an extension of UPC-A that includes a two-digit add-on.
- UPC-A+5 is an extension of UPC-A that includes a five-digit add-on.
- UPC-E is a barcode symbology derived from UPC-A that is designed for small packaging.
- UPC-A Composite is a variant of UPC-A that should be used when a CC-A or CC-B GS1 composite 2D component is required.
- A UPC-A symbol can be converted to an EAN-13 symbol by prefixing the GTIN-12 with 0 to make the equivalent GTIN-13.

Standards: ISO/IEC 15420, BS EN 797, GS1 General Specifications.

### Data and Options

- The data field for a UPC-A may contain eleven or twelve digits, optionally followed by a space then two or five digits if an EAN-2 or EAN-5 add-on is required.
- Alternatively, the data field may contain seven or eight digits of a UPC-E to produce the equivalent UPC-A symbol.
- If eleven digits of primary data are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.
- The **includetext** option should normally be supplied.

### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    788581014974
Options: includetext
Encoder: upca
```

```
Data:    78858101497
Options: includetext
Encoder: upca
```


A symbol that includes a five-digit add-on:

```
Data:    788581014974 54499
Options: includetext guardwhitespace
Encoder: upca
```


## UPC-E

**UPC-E** is a compacted form of the UPC-A barcode symbology that usually carries a GTIN-12 with a number system of *0* or *1* that has been zero compressed.

Variants:

- UPC-E0 is a UPC-E with a number system of *0*.
- UPC-E1 is a UPC-E with a number system of *1*.
- UPC-E+2 is an extension of UPC-E that includes a two-digit add-on.
- UPC-E+5 is an extension of UPC-E that includes a five-digit add-on.
- UPC-A is the full size form of UPC-E.
- UPC-E Composite is a variant of UPC-E that should be used when a CC-A or CC-B GS1 Composite 2D component is required.

Standards: ISO/IEC 15420, BS EN 797, GS1 General Specifications.

### Data and Options

- The data field takes either seven or eight digits, optionally followed by a space then two or five digits if an EAN-2 or EAN-5 add-on is required.
- Alternatively, the data field may contain eleven or twelve digits of a UPC-A to produce the equivalent UPC-E symbol, provided that the input can be zero suppressed.
- If seven digits of primary data are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.
- The **includetext** option should normally be supplied.

### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    01234565
Options: includetext
Encoder: upce
```

```
Data:    0123456
Options: includetext
Encoder: upce
```


A truncated symbol:

```
Data:    01234565
Options: includetext height=0.5
Encoder: upce
```


## ISBN

An **ISBN** barcode is a variant of EAN-13 that is used to identify books.

Also known as: ISBN-13, International Standard Book Number, Bookland EAN-13.

Variants:

- ISBN-10 is a legacy format that was depreciated for public use after 1st January 2007.

Standards: ISO 2108, ISO/IEC 15420, BS EN 797, GS1 General Specifications.

### Data and Options

- The data should contain twelve or thirteen digits separated appropriately by dash characters -.
- The data can also be provided in legacy ISBN-10 format as nine or ten digits separated appropriately by dash characters -. This will be automatically upgraded to the ISBN-13 format.
- If the last digit of the primary data is not given then the ISBN check digit is calculated automatically.
- The **legacy** option prevents ISBN-10 input from being upgraded to ISBN-13 and will result in a symbol that is obsolete and should not be used at point of sale.
- The primary data can optionally be followed by a space then two or five digits if an EAN-2 or EAN-5 add-on is required.
- The **includetext** option should normally be supplied.
- The **guardwhitespace** option enables the display of white space guard marks.
- The following options are also relevant to this barcode symbology:
  - **isbntextfont**: Font name for text above symbol
  - **isbntextsize**: Font size for the text above symbol, in points
  - **isbntextxoffset**: Horizontal position of ISBN text, in points
  - **isbntextyoffset**: Vertical position of ISBN text, in points

### Example ISBN

Identical symbols, input provided with and without an ISBN check digit:

```
Data:    978-1-873671-00-9
Options: includetext
Encoder: isbn
```

```
Data:    978-1-873671-00
Options: includetext
Encoder: isbn
```

ISBN 978-1-873671-00-9


An ISBN with a five-digit add-on:

```
Data:    978-1-873671-00-9 54499
Options: includetext guardwhitespace
Encoder: isbn
```

ISBN 978-1-873671-00-9


The following ISBN-10 input will be automatically upgraded to a valid ISBN-13 symbol:

```
Data:    1-86074-271-2
Options: includetext
Encoder: isbn
```

```
Data:    1-86074-271
Options: includetext
Encoder: isbn
```

ISBN 978-1-86074-271-2


**Example ISBN-10**

Note that ISBN-10 is legacy format not for use at P.O.S.

The following will generate an obsolete ISBN-10 symbol:

```
Data:    1-86074-271-8
Options: legacy includetext guardwhitespace
Encoder: isbn
```

```
Data:    1-86074-271
Options: legacy includetext guardwhitespace
Encoder: isbn
```

ISBN 1-86074-271-8

## ISMN

An **ISMN** barcode is a variant of EAN-13 with a prefix *979* that is used to identify printed music.

Also known as: International Standard Music Number, ISMN-13.

Variants:

- ISMN-10 is a legacy format that was depreciated for public use.

Standards: ISO 10957, ISO/IEC 15420, BS EN 797, GS1 General Specifications.

### Data and Options

- The data should contain twelve or thirteen digits separated appropriately by dash characters -.
- The data can also be provided in legacy ISMN-10 format start *M-* then eight or nine digits separated appropriately by dash characters -. This will be automatically upgraded to the ISMN-13 format.
- The **legacy** option prevents ISMN-10 input from being upgraded to ISMN-13 and will result in a symbol that is obsolete and should not be used at point of sale.
- If the last digit of the primary data is not given then the ISMN check digit is calculated automatically.
- The primary data can optionally be followed by a space then two or five digits if an EAN-2 or EAN-5 add-on is required.
- The **includetext** option should normally be supplied.
- The **guardwhitespace** option enables the display of white space guard marks.
- The following options are also relevant to this barcode symbology:
  - **ismntextfont**: Font name for text above symbol
  - **ismntextsize**: Font size for the text above symbol, in points
  - **ismntextxoffset**: Horizontal position of ISMN text, in points
  - **ismntextyoffset**: Vertical position of ISMN text, in points

### Example ISMN

Identical symbols, input provided with and without an ISMN check digit:

```
Data:    979-0-2600-0043-8
Options: includetext
Encoder: ismn
```

```
Data:    979-0-2600-0043
Options: includetext
Encoder: ismn
```


The following ISMN-10 input will be automatically upgraded to a valid ISMN-13 symbol:

```
Data:    M-345-24680-5
Options: includetext
Encoder: ismn
```

```
Data:    M-345-24680
Options: includetext
Encoder: ismn
```


#### Example ISMN-10

Note that ISMN-10 is a legacy format not for use at P.O.S.

The following will generate an obsolete ISMN-10 symbol:

```
Data:    M-345-24680-5
Options: legacy includetext guardwhitespace
Encoder: ismn
```

```
Data:    M-345-24680
Options: legacy includetext guardwhitespace
Encoder: ismn
```


### ISSN

An **ISSN** barcode is an EAN-13 with prefix *977* used to identify periodicals.

Also known as: International Standard Serial Number.

Standards: ISO 3297, ISO/IEC 15420, BS EN 797, GS1 General Specifications.

#### Data and Options

- The data should contain the seven or eight digits ISSN separated by a dash characters -, followed by a two-digit sequence variant, optionally followed by two or five digits if a two-digit add-on or five-digit add-on is required.
- If the last digit of the ISSN data is not given then the ISSN check digit is calculated automatically.
- The **includetext** option should normally be supplied.
- The **guardwhitespace** option enables the display of white space guard marks.
- The following options are also relevant to this barcode symbology:
  - **issntextfont**: Font name for text above symbol
  - **issntextsize**: Font size for the text above symbol, in points
  - **issntextxoffset**: Horizontal position of ISSN text, in points
  - **issntextyoffset**: Vertical position of ISSN text, in points

A sequence variant is a two-digit number that usually starts at zero and is incremented whenever the recommended retail price is amended, where applicable.

#### Example

Identical symbols, input provided with and without an ISSN check digit and having sequence number *00*:

```
Data:    0317-8471 00
Options: includetext guardwhitespace
Encoder: issn
```

```
Data:    0317-847 00
Options: includetext guardwhitespace
Encoder: issn
```


An ISSN with sequence number *03* and a two-digit add-on representing issue number *17*:

```
Data:    0317-8471 03 17
Options: includetext guardwhitespace
Encoder: issn
```

```
Data:    0317-847 03 17
Options: includetext guardwhitespace
Encoder: issn
```


# Two-Dimensional

## Aztec Code

**Aztec Code** is a 2D matrix-style barcode symbology. It can encode full 256-character extended-ASCII.

Variants:

- Aztec Runes are a set of small barcode symbols that are used for special applications.

Standards: ISO/IEC 24778, ANSI/AIM BC13 - ISS Aztec Code.

### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The **eclevel** option is used to specify the percentage of error correction to be applied when expanding the data, by default `23`.
- The **ecaddchars** option is used to specify how many additional error correction characters to apply the data once expanded by the eclevel percentage, by default `3`.
- The **layers** option is used to specify a particular number of layers in which to encode the data, between `1` and `32`. By default the encoder will create a symbol with be minimal number of layers to encode the given data.
- The **format** option is used to select between `format=full` and `format=compact` symbol types. By default the encoder will choose the most appropriate format to create a symbol of minimal size.

- *Deprecated: Use Aztec Runes instead. The **format** option can also be used to create Aztec Code "runes", using `format=rune`. In this case the rune symbol number should be given in the data field.*
- The **readerinit** option denotes that the symbol is used for programming the barcode reader.
- The **raw** option denotes that the data field is providing the input as a pre-encoded bitstream suitable for direct low-level encoding.

#### Examples

```
Data:    This is Aztec Code
Options:
Encoder: azteccode
```

```
Data:    This is ^065ztec Code
Options: parse eclevel=50 ecaddchars=0
Encoder: azteccode
```

```
Data:    ABC123
Options: layers=3 format=full
Encoder: azteccode
```

```
Data:    ABC123
Options: format=compact
Encoder: azteccode
```

```
Data:    00100111001000000101001101111000010100111100101000000110
Options: raw
Encoder: azteccode
```

## Aztec Runes

**Aztec Runes** are a set of small barcode symbols that are used for special applications.

Variants:

Aztec Code is a 2D matrix-style barcode symbology that can encode full 256 character extended-ASCII.

Standards: ISO/IEC 24778, ANSI/AIM BC13 - ISS Aztec Code.

#### Data and Options

- The data field contains the rune number `0` to `255`.

**Examples**

```
Data:    25
Options:
Encoder: aztecrune
```

## Data Matrix

The **Data Matrix** symbology is 2D matrix-style barcode that can encode full 256 character extended-ASCII.

Also known as: Data Matrix ECC 200.

Variants:

- GS1 DataMatrix is a variant of Data Matrix that should be used when encoding data that is in GS1 Application Identifier standard format.
- HIBC Data Matrix is a variant of Data Matrix that should be used when encoding HIBC formatted data.

Standards: ISO/IEC 16022, ANSI/AIM BC11 - ISS Data Matrix.

**Data and Options**

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value. This is useful for specifying unprintable characters.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1`, `^PROG`, `^MAC5`, `^MAC6`.
- The **format** option is used to specify the shape of the symbol, either `square` (default) or `rectangle`.
- The **dmre** option enable Data Matrix Rectangular Extension with increases the number of rectangular symbol sizes available.
- The **columns** and **rows** options are used to specify the size of the symbol.
- The **version** option can also be used to specify the symbol size, as `version=RxC`. Valid options are:
  - With `format=square`: *10x10*, *12x12*, *14x14*, *16x16*, *18x18*, *20x20*, *22x22*, *24x24*, *26x26*, *32x32*, *36x36*, *40x40*, *44x44*, *48x48*, *52x52*, *64x64*, *72x72*, *80x80*, *88x88*, *96x96*, *104x104*, *120x120*, *132x132*, *144x144*
  - With `format=rectangle`: *8x18*, *8x32*, *12x26*, *12x36*, *16x36*, *16x48*
  - With `format=rectangle` and `dmre`: *8x18*, *8x32*, *8x48*, *8x64*, *12x26*, *12x36*, *12x64*, *16x36*, *16x48*, *16x64*, *24x32*, *24x36*, *24x48*, *24x64*, *26x32*, *26x40*, *26x48*, *26x64*
- If **columns**, **rows** and **version** are unspecified the encoder will default to creating a symbol of the specified **format** that is the minimum size to represent the given data.

**Examples**

```
Data:    This is Data Matrix
Options:
Encoder: datamatrix
```

```
Data:    This is ^068ata Matrix
Options: parse
Encoder: datamatrix
```

```
Data:    Fixed size
Options: rows=48 columns=48
Encoder: datamatrix
```

```
Data:    Rectangular
Options: format=rectangle version=16x48
Encoder: datamatrix
```

## Han Xin Code

The **Han Xin Code** symbology is a 2D matrix-style barcode symbology that can encode full 256 character extended-ASCII.

Also known as: Chinese Sensible.

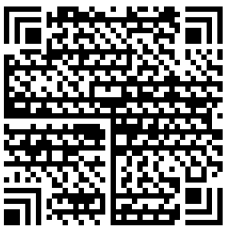
Standards: GB/T 21049-2007.

### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The **eclevel** option is used to specify the error correction level:
  - `eclevel=L1` - Lowest
  - `eclevel=L2`
  - `eclevel=L3`
  - `eclevel=L4` - Highest
- The **version** option is used to specify the size of the symbol, 1 to 84.
- If unspecified the encoder will select the version of the symbol that is the minimum size to represent the given data at the selected error correction level.

### Examples

```
Data:    Han Xin Code
Options: version=10 eclevel=L4
Encoder: hanxin
```



## MicroPDF417

The **MicroPDF417** barcode symbology is 2D stacked-linear barcode based on PDF417 that can encode full 256 character extended-ASCII.

Variants:

- PDF417 is a larger variant of the MicroPDF417 barcode.
- HIBC MicroPDF417 is a variant of MicroPDF417 that should be used when encoding HIBC formatted data.

Standards: ISO/IEC 24728, AIM ISS - MicroPDF417.

#### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The **columns** and **rows** options are used to specify the size of the symbol. Valid values are:
  - *1x11*, *1x14*, *1x17*, *1x20*, *1x24*, *1x28*, *2x8*, *2x11*, *2x14*, *2x17*, *2x20*, *2x23*, *2x26*, *3x6*, *3x8*, *3x10*, *3x12*, *3x15*, *3x20*, *3x26*, *3x32*, *3x38*, *3x44*, *4x4*, *4x6*, *4x8*, *4x10*, *4x12*, *4x15*, *4x20*, *4x26*, *4x32*, *4x38*, *4x44*
- If the **columns** and **rows** are unspecified the encoder will default to creating a symbol that is the minimum size to represent the given data.
- The **rowmult** option is used to specify how tall each bar is, with respect to the minimum module width. The default is 3.
- The **raw** option denotes that the data field is providing the input as a pre-encoded codewords in `^NNN` format, suitable for direct low-level encoding.
- The **cca** option identifies this symbol as a *CC-A* 2D component of a GS1 Composite symbol.
- The **ccb** option identifies this symbol as a *CC-B* 2D component of a GS1 Composite symbol.
- Note: Special size rules apply when the **cca** option is given, in which case the **columns** and **rows** options that are used to specify the size of the symbol must be one of:
  - *2x5*, *2x6*, *2x7*, *2x8*, *2x9*, *2x10*, *2x12*, *3x4*, *3x5*, *3x6*, *3x7*, *3x8*, *4x3*, *4x4*, *4x5*, *4x6*, *4x7*

#### Examples

```
Data:    MicroPDF417
Options:
Encoder: micropdf417
```

```
Data:    MicroP^068F417
Options: parse rows=15 columns=4
Encoder: micropdf417
```

## PDF417

The **PDF417** barcode symbology is 2D stacked-linear barcode that can encode full 256 character extended-ASCII.

Variants:

- Compact PDF417 is a shortened form of the PDF417 barcode that is used in applications where the space for the symbol is restricted.
- MicroPDF417 is a smaller variant of the PDF417 barcode.
- HIBC PDF417 is a variant of PDF417 that should be used when encoding HIBC formatted data.

Standards: ISO/IEC 15438, DD ENV 12925, AIM USS - PDF417.

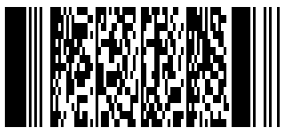
### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The **eclevel** option is used to specify the error correction level, from 1 to 5. The default is to choose a standard level of error correction that is determined by the encoded data length.
- The **columns** option specifies the number of columns (or groups of bars) in the output symbol, from 1 to 30.
- The **rows** option specifies the minimum number of rows in the symbol, from 3 to 90.
- If **rows** is unspecified the encoder will select a number that creates a symbol that is the minimum size to represent the given data.
- The **rowmult** option is used to specify how tall each bar is, with respect to the minimum module width. The default is 3.
- *Deprecated: Use Compact PDF417 instead. The* ***compact*** *option is used to create a compact/truncated PDF417 symbol that has fewer bars per row that a standard symbol and hence is more narrow.*
- The **raw** option denotes that the data field is providing the input as a pre-encoded codewords in `^NNN` format, suitable for direct low-level encoding.
- The **ccc** option identifies this symbol as a *CC-C* 2D component of a GS1 Composite symbol.

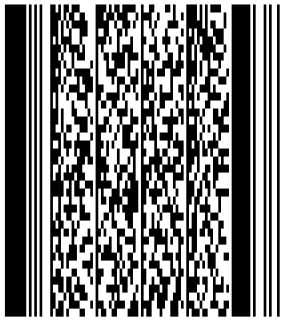
### Examples

```
Data:    PDF417
Options:
Encoder: pdf417
```

```
Data:    P^068F417
Options: parse columns=2 rows=15
Encoder: pdf417
```



```
Data:    Strong error correction
Options: columns=2 eclevel=5
Encoder: pdf417
```



```
Data:    ^453^178^121^239
Options: raw columns=2
Encoder: pdf417
```

## Compact PDF417

**Compact PDF417** is a shortened form of the PDF417 barcode that is used in applications where the space for the symbol is restricted.

Also known as: Truncated PDF417

Variants:

- PDF417 is the larger, more popular variant.
- MicroPDF417 is a smaller variant of the PDF417 barcode.
- HIBC PDF417 is a variant of PDF417 that should be used when encoding HIBC formatted data.

Standards: ISO/IEC 15438, DD ENV 12925, AIM USS - PDF417.

### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The **eclevel** option is used to specify the error correction level, from 1 to 5. The default is to choose a standard level of error correction that is determined by the encoded data length.
- The **columns** option specifies the number of columns (or groups of bars) in the output symbol, from 1 to 30.
- The **rows** option specifies the minimum number of rows in the symbol, from 3 to 90.
- If **rows** is unspecified the encoder will select a number that creates a symbol that is the minimum size to represent the given data.
- The **rowmult** option is used to specify how tall each bar is, with respect to the minimum module width. The default is 3.
- The **raw** option denotes that the data field is providing the input as a pre-encoded codewords in `^NNN` format, suitable for direct low-level encoding.

### Examples

```
Data:    A truncated PDF417
Options: columns=4
Encoder: pdf417compact
```

## QR Code

The **QR Code** symbology is a 2D matrix-style barcode symbology that can encode full 256 character extended-ASCII.

Also known as: Quick Response Code.

Variants:

- Micro QR Code is a small QR Code that is used in applications that require a small symbol space.
- GS1 QR Code is a variant of Data Matrix that should be used when encoding data that is in GS1 Application Identifier standard format.
- HIBC QR Code is a variant of QR Code that should be used when encoding HIBC formatted data.

Standards: ISO/IEC 18004, JIS X 0510, ITS - QR Code, AIM ISS - QR Code.

### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.

- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The **eclevel** option is used to specify the error correction level:
  - `eclevel=L` - Low
  - `eclevel=M` - Medium (default)
  - `eclevel=Q` - Quality
  - `eclevel=H` - High
- The **version** option is used to specify the size of the symbol, 1 to 40.
- If unspecified the encoder will select the version of the symbol that is the minimum size to represent the given data at the selected error correction level.
- The **format** option is used to select between `format=full` and `format=micro` (deprecated) symbol types. Alternatively, `format=any` will select the optimal symbol format for the given data. By default *full* format symbols will be generated.
- Note: It is recommended that the Micro QR Code encoder is used for such symbols.

#### Examples

```
Data:    QR Code
Options:
Encoder: qrcode
```

```
Data:    QR ^067ode
Options: parse
Encoder: qrcode
```

```
Data:    QR CODE 1234
Options: version=10 eclevel=Q
Encoder: qrcode
```

## Micro QR Code

The **Micro QR Code** symbology is a smaller variant of QR Code that is used in applications that require a small symbol space.

Also known as: Micro Quick Response Code.

Variants:

- QR Code is the more popular, larger variant.

Standards: ISO/IEC 18004, JIS X 0510, ITS - QR Code, AIM ISS - QR Code.

#### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.

- An appropriate size will be selected to work around the following restrictions:
  - An M1 symbol is only compatible with numeric data.
  - An M2 symbol is only compatible with alphanumeric data.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The **version** option is used to specify the size of the symbol, either `version=M1`, `version=M2`, `version=M3` or `version=M4`.
- The **eclevel** option is used to specify the error correction level:
  - `eclevel=L` - Low (default)
  - `eclevel=M` - Medium; Not compatible with M1 symbols
  - `eclevel=Q` - Quality; Only compatible with M4 symbols
- If unspecified the encoder will select the version of the symbol that is the minimum size to represent the given data at the selected error correction level.

#### Examples

```
Data:    01234567
Options:
Encoder: microqrcode
```

# One-Dimensional

## Code 128

**Code 128** is an arbitrarily long, high-density barcode symbology that can be used to encode full 256 character extended-ASCII.

Also known as: USD-6, USS-128, Code 128A, Code 128B, Code 128C.

Variants:

- GS1-128 is a variant of Code 128 that should be used when encoding data that is in GS1 Application Identifier standard format.
- HIBC Code 128 is a variant of Code 128 that should be used when encoding HIBC formatted data.

Standards: ISO/IEC 15417, ANSI/AIM BC4 - ISS Code 128, BS EN 799.

#### Data and Options

- The input can consist of any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII or extended-ASCII value, useful for specifying unprintable characters, e.g. `^029` for *GS*, `^209` for *Ñ*, etc.
- *FNC4* function characters will be inserted automatically to allow the encoding of extended ASCII characters.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1` through `^FNC3`.
- When the **parsefnc** option is specified, the special pseudo characters `^LNKA` and `^LNKC` at the end of the symbol indicate that a GS1-128 symbol includes a *CC-A/B* or *CC-C* GS1 composite 2D component.
- The **raw** option denotes that the data field is providing the input as pre-encoded codewords in `^NNN` format, suitable for direct low-level encoding.
- The mandatory check digit is calculated automatically.

#### Example

```
Data:    Count0123456789!
Options: includetext
Encoder: code128
```


## Code 39

The **Code 39** barcode symbology is discrete, variable length and self-checking.

Also known as: Code 3 of 9, LOGMARS, Alpha39, USD-3, USS-39.

Variants:

- Code 39 Extended is a variant of Code 39 that can be used to encode full 128 character ASCII with the use of shift character combinations.
- HIBC Code39 is a variant of Code 39 that should be used when encoding HIBC formatted data.
- AIM USD-2 is a subset of Code 39 containing the characters A-Z, 0-9, *space*, `-` and `.`.

Standards: ISO/IEC 16388, ANSI/AIM BC1 - USS Code 39, BS EN 800, MIL STD 1189.

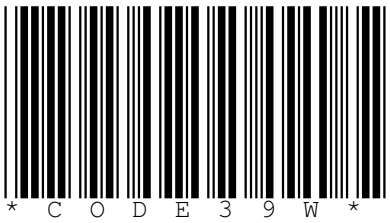
#### Data and Options

- The data field can hold any of the following:
  - Numbers 0-9
  - Capital letters A-Z
  - Symbols `-.$/+%*` and *space*
- The **includecheck** option calculates the check digit.
- The **includecheckintext** option makes the calculated checksum appear in the human readable text.
- The **hidestars** option suppresses the asterisks in the human readable text.

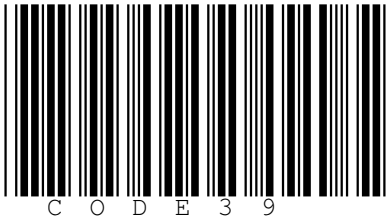
#### Examples

```
Data:    CODE39
Options: includetext
Encoder: code39
```


```
Data:    CODE39
Options: includecheck includetext includecheckintext
Encoder: code39
```




```
Data:    CODE39
Options: hidestars includecheck includetext
Encoder: code39
```




## Code 39 Extended

The **Code 39 Extended** barcode symbology is discrete, variable length and self-checking. It is based on Code 39 but can encode full 128 character ASCII by using shift combinations.

Also known as: Code 39 Full ASCII.

Variants:

- Code 39 is a simpler variant of Code 39 Extended.

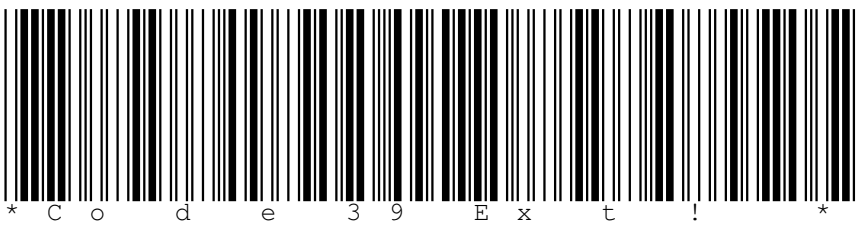
Standards: ISO/IEC 16388, ANSI/AIM BC1 - USS Code 39, BS EN 800.

### Data and Options

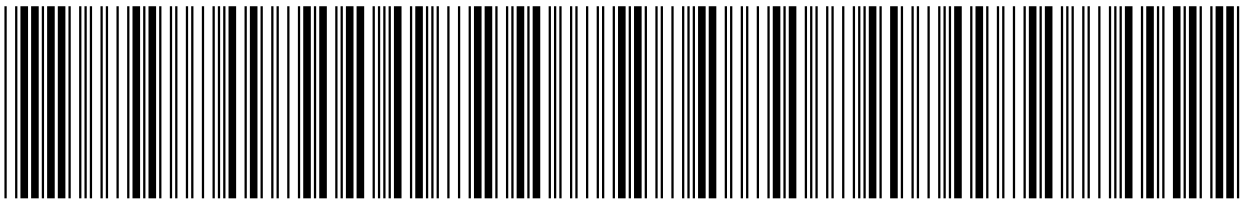
- The data field can consist of any ASCII data.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters, e.g. `^029` for *GS*, etc.
- The **includecheck** option calculates the check digit.
- The **includecheckintext** causes the calculated check digit to appear in the human readable text.
- The **hidestars** option suppresses the asterisks in the human readable text.

### Examples

```
Data:    Code39 Ext!
Options: includetext includecheck
Encoder: code39ext
```




```
Data:    Code39^029Extended
Options: parse includecheck
Encoder: code39ext
```



## Code 93

**Code 93** is a continuous, variable length, self-checking barcode symbology.

Also known as: USD-7, USS-93.

Variants:

- Code 93 Extended is a variant of Code 93 that can be used to encode full 128 character ASCII with the use of special shift character combinations.

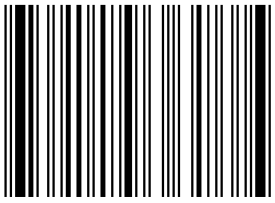
Standards: ANSI/AIM BC5 - USS Code 93, ITS 93i.

### Data and Options

- The data field can hold any of the following:
  - Numbers 0-9
  - Capital letters A-Z
  - Symbols `-.$/+%*` and *space*
- The **parsefnc** option allows the special shift characters to be supplied as `^SFT$`, `^SFT%`, `^SFT/` and `^SFT+`.
- The **includecheck** option calculates the two check digits.

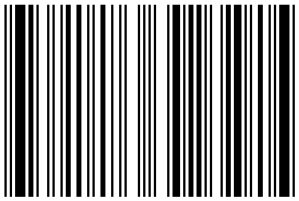
### Examples

```
Data:    CODE93
Options: includecheck
Encoder: code93
```



Code 93 including a special shift combination `(/)A` representing *!*:

```
Data:    CODE93^SFT/A
Options: parsefnc includecheck
Encoder: code93
```



## Code 93 Extended

The **Code 93 Extended** barcode symbology is continuous, variable length and self-checking. It is based on Code 93 but can encode full 128 character ASCII using four additional shift characters: *($) (%) (/) (+)*

Also known as: Code 93 Full ASCII.

Variants:

- Code 93 is a simpler variant of the Code 93 Extended barcode symbology.

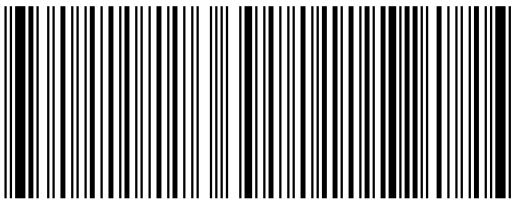
Standards: ANSI/AIM BC5 - USS Code 93, ITS 93i.

#### Data and Options

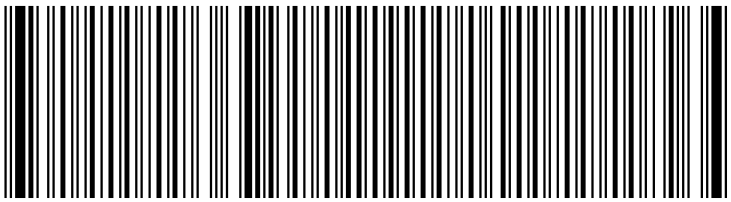
- The data field can consist of any ASCII data.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters, e.g. `^029` for *GS*, etc.
- The **includecheck** option calculates the two check digits.

#### Examples

```
Data:    Code93Ext!
Options: includecheck
Encoder: code93ext
```



```
Data:    Code93^029Extended
Options: parse includecheck
Encoder: code93ext
```



### Interleaved 2 of 5

**Interleaved 2 of 5** is a high-density numeric barcode symbology.

Also known as: ITF, Code 2 of 5 Interleaved, USD-1, USS-Interleaved 2 of 5.

Variants:

- ITF-14 is a variant of Interleaved 2 of 5 that should be used when encoding a fourteen-digit GTIN.

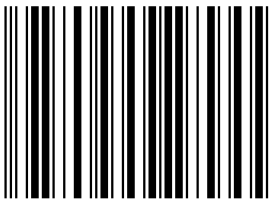
Standards: ISO/IEC 16390, ANSI/AIM BC2 - USS Interleaved 2 of 5, BS EN 801.

#### Data and Options

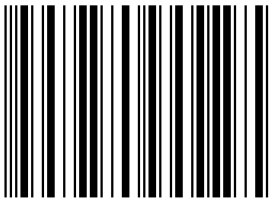
- The data can consist of any number of digits.
- The **includecheck** option calculates the check digit.
- The **includecheckintext** option makes the calculated checksum appear in the human readable text.
- If the length of the symbol including the possible check digit would be odd then the data is prefixed by *0*.

#### Examples

```
Data:    0123456789
Options:
Encoder: interleaved2of5
```



```
Data:    2401234567
Options: includecheck includetext includecheckintext
Encoder: interleaved2of5
```



# Supply Chain

## GS1 DataMatrix

**GS1 DataMatrix** is an implementation of the Data Matrix (ECC 200) barcode symbology with GS1 formatted data.

Standards: ISO/IEC 16022, ANSI/AIM BC11 ISS, GS1 General Specifications.

### Data and Options

- The data field input is provided in GS1 Application Identifier standard format starting with the mandatory *(01)* Application Identifier.
- The **format** option is used to specify the shape of the symbol, either `square` (default) or `rectangle`.
- The **columns** and **rows** options are used to specify the size of the symbol.
- The **version** option can also be used to specify the symbol size, as `version=RxC`. Valid options are:
  - With `format=square`: *10x10*, *12x12*, *14x14*, *16x16*, *18x18*, *20x20*, *22x22*, *24x24*, *26x26*, *32x32*, *36x36*, *40x40*, *44x44*, *48x48*, *52x52*, *64x64*, *72x72*, *80x80*, *88x88*, *96x96*, *104x104*, *120x120*, *132x132*, *144x144*
  - With `format=rectangle`: *8x18*, *8x32*, *12x26*, *12x36*, *16x36*, *16x48*
- If **columns**, **rows** and **version** are unspecified the encoder will default to creating a symbol of the specified **format** that is the minimum size to represent the given data.

### Example

```
Data:    (01)95012345678903(3103)000123
Options:
Encoder: gs1datamatrix
```

## GS1 QR Code

**GS1 QR Code** is an implementation of the QR Code barcode symbology with GS1 formatted data.

Standards: ISO/IEC 18004, ITS - QR Code, GS1 General Specifications.

**Data and Options**

- The data field input is provided in GS1 Application Identifier standard format starting with the mandatory *(01)* and *(8200)* Application Identifiers.
- The **eclevel** option is used to specify the error correction level:
  - `eclevel=L` - Low
  - `eclevel=M` - Medium (default)
  - `eclevel=Q` - Quality
  - `eclevel=H` - High
- The **version** option is used to specify the size of the symbol, 1 to 40.
- If the **version** is unspecified the encoder will default to creating a symbol that is the minimum size to represent the given data at the selected error correction level.

**Example**

```
Data:    (01)03453120000011(8200)http://www.abc.net
Options:
Encoder: gs1qrcode
```

```
Data:    (01)03453120000011(8200)http://abc.net(10)XYZ(410)9501101020917
Options:
Encoder: gs1qrcode
```

## GS1-128

**GS1-128** is an implementation of the Code 128 barcode symbology which carries GS1 formatted data, including a GTIN-14.

Also known as: UCC/EAN-128, EAN-128, UCC-128.

Variants:

- GS1-128 Composite is a variant of GS1-128 that should be used when a CC-A, CC-B or CC-C GS1 composite 2D component is required.
- EAN-14 is a variant of GS1-128 that should be used when encoding a fourteen-digit GTIN.
- SSCC-18 is a variant of GS1-128 that should be used when encoding an eighteen-digit SSCC.

Standards: ISO/IEC 15417, ANSI/AIM BC4-1999 ISS, BS EN 799, GS1 General Specifications.

**Data and Options**

- The data field input is provided in GS1 Application Identifier standard format.
- The **linkagea** option specifies that the symbol includes a *CC-A* or *CC-B* GS1 composite 2D component.
- The **linkagec** option specifies that the symbol includes a *CC-C* GS1 composite 2D component.

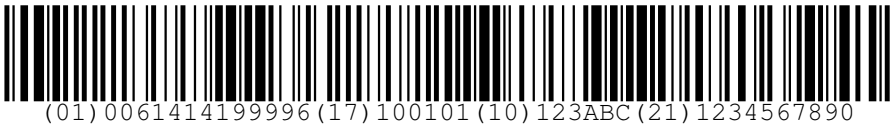
### Examples

GTIN *95012345678903*; Weight *0.123kg*:

```
Data:    (01)95012345678903(3103)000123
Options: includetext
Encoder: gs1-128
```


GTIN *00614141999996*; Expiration date *1st Jan 2010*; Batch *123ABC*; Serial *1234567890*:

```
Data:    (01)00614141999996(17)100101(10)123ABC(21)1234567890
Options: includetext
Encoder: gs1-128
```




## EAN-14

**EAN-14** is an implementation of the GS1-128 barcode symbology with *AI (01)* that is typically used to encode a GTIN-14.

Also known as: UCC-14.

Standards: ISO/IEC 15417, ANSI/AIM BC4-1999 ISS, BS EN 799, GS1 General Specifications.

### Data and Options

- The data field input is provided in GS1 Application Identifier standard format and must be a solitary *AI (01)* with thirteen or fourteen digits of a GTIN, i.e. `(01)...`.
- Arbitrary spacing may be placed between the digits to format the human readable text without interfering with the encoded data.
- If thirteen digits of primary data are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.

### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    (01)04601234567893
Options: includetext
Encoder: ean14
```

```
Data:    (01)0460123456789
Options: includetext
Encoder: ean14
```

## ITF-14

**ITF-14** is an implementation of the Interleaved 2 of 5 barcode symbology that is typically used to encode a GTIN-14, GTIN-13 or GTIN-12.

Also known as: UPC Shipping Container Symbol, SCS, UPC Case Code.

Standards: ISO/IEC 16390, ANSI/AIM BC2-1995 USS, BS EN 801, GS1 General Specifications.

### Data and Options

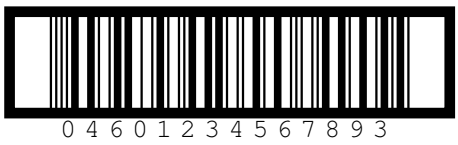
- The data consists of either thirteen or fourteen digits.
- Arbitrary spacing may be placed between the digits to format the human readable text without interfering with the encoded data.
- If thirteen digits are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.

### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    04601234567893
Options: includetext
Encoder: itf14
```

```
Data:    0460123456789
Options: includetext
Encoder: itf14
```




## SSCC-18

**SSCC-18** is an implementation of the GS1-128 barcode symbology with *AI (00)* that is typically used to encode an eighteen-digit shipping container serial number.

Also known as: EAN-18, NVE.

Standards: ISO/IEC 15417, ANSI/AIM BC4-1999 ISS, BS EN 799, GS1 General Specifications.

### Data and Options

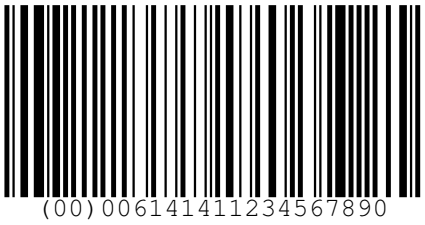
- The data field input is provided in GS1 Application Identifier standard format and must be a solitary *AI (00)* with seventeen or eighteen digits of a Serial Shipping Container Code, i.e. `(00)...`.
- Arbitrary spacing may be placed between the digits to format the human readable text without interfering with the encoded data.
- If seventeen digits of primary data are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.

### Example

Identical symbols, input provided with and without a check digit:

```
Data:    (00)006141411234567890
Options: includetext
Encoder: sscc18
```

```
Data:    (00)00614141123456789
Options: includetext
Encoder: sscc18
```




# GS1 DataBar Family

## GS1 DataBar Omnidirectional

**GS1 DataBar Omnidirectional** is a fixed-length, linear barcode symbology that can be used to encode a GTIN-14 for use at point of sale.

Also known as: RSS-14

Variants:

- GS1 DataBar Stacked Omnidirectional is a variant of GS1 DataBar Omnidirectional for use where a taller, narrower symbol is required.
- GS1 DataBar Omnidirectional Composite is a variant of GS1 DataBar Omnidirectional that should be used when a CC-A or CC-B GS1 composite 2D component is required.

Standards: ISO/IEC 24724, ITS Reduced Space Symbology (RSS), AIM ISS - Reduced Space Symbology (RSS), GS1 General Specifications.

### Data and Options

- The data field input is provided in GS1 Application Identifier standard format and must be a solitary *AI (01)* with thirteen or fourteen digits of a GTIN, i.e. `(01)...`.
- If thirteen digits of *AI (01)* data are supplied then the check digit is calculated automatically, otherwise the digits are encoded as supplied.
- The **linkage** option signifies the presence of a GS1 composite 2D component.

### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    (01)24012345678905
Options:
Encoder: databaromni
```

```
Data:    (01)2401234567890
Options:
Encoder: databaromni
```

## GS1 DataBar Stacked Omnidirectional

**GS1 DataBar Stacked Omnidirectional** is a fixed-length, stacked linear barcode symbology that can be used to encode a GTIN-14 for use a point of sale.

Also known as: RSS-14 Stacked Omnidirectional.

Variants:

- GS1 DataBar Omnidirectional is a variant of GS1 DataBar Stacked Omnidirectional for use where a shorter, wider symbol is required.
- GS1 DataBar Stacked Omnidirectional Composite is a variant of GS1 DataBar Stacked Omnidirectional that should be used when a CC-A or CC-B GS1 composite 2D component is required.

Standards: ISO/IEC 24724, ITS Reduced Space Symbology (RSS), AIM ISS - Reduced Space Symbology (RSS), GS1 General Specifications.

#### Data and Options

- The data field input is provided in GS1 Application Identifier standard format and must be a solitary *AI (01)* with thirteen or fourteen digits of a GTIN, i.e. `(01)...`.
- If thirteen digits of *AI (01)* data are supplied then the check digit is calculated automatically, otherwise the digits are encoded as supplied.
- The **linkage** option signifies the presence of a GS1 composite 2D component.

#### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    (01)24012345678905
Options:
Encoder: databarstackedomni
```

```
Data:    (01)2401234567890
Options:
Encoder: databarstackedomni
```

## GS1 DataBar Expanded

**GS1 DataBar Expanded** is a variable-length, linear barcode symbology that can be used to encode a GTIN-14 alongside a number of other application identifiers for use at point of sale.

Also known as: RSS Expanded.

Variants:

- GS1 DataBar Expanded Stacked is a variant of GS1 DataBar Expanded for use where a taller, narrower symbol is required.
- GS1 DataBar Expanded Composite is a variant of GS1 DataBar Expanded that should be used when a CC-A or CC-B GS1 composite 2D component is required.

Standards: ISO/IEC 24724, ITS Reduced Space Symbology (RSS), AIM ISS - Reduced Space Symbology (RSS), GS1 General Specifications.

#### Data and Options

- The data field input is provided in GS1 Application Identifier standard format.
- If the data contains a number of application identifiers matching any of the specifications below then they should be provided in this given order for maximum encoding efficiency:
  - `(01)9...(3103)...`
  - `(01)9...(3202)...`
  - `(01)9...(3203)...`

  - `(01)9...(310x/320x)...(11/13/15/17)...`
  - `(01)9...(310x/320x)...`
  - `(01)9...(392x)...`
  - `(01)9...(393x)...`
  - `(01)...`
- The **linkage** option signifies the presence of a GS1 composite 2D component.

#### Examples

```
Data:    (01)95012345678903(3103)000123
Options:
Encoder: databarexpanded
```

## GS1 DataBar Expanded Stacked

**GS1 DataBar Expanded Stacked** is a variable-length, stacked-linear barcode symbology that can be used to encode a GTIN-14 alongside a number of other application identifiers for use at point of sale.

Also known as: RSS Expanded Stacked.

Variants:

- GS1 DataBar Expanded is a variant of GS1 DataBar Expanded Stacked for use where a shorter, wider symbol is required.
- GS1 DataBar Expanded Stacked Composite is a variant of GS1 DataBar Expanded Stacked that should be used when a CC-A or CC-B GS1 composite 2D component is required.

Standards: ISO/IEC 24724, ITS Reduced Space Symbology (RSS), AIM ISS - Reduced Space Symbology (RSS), GS1 General Specifications.

#### Data and Options

- The data field input is provided in GS1 Application Identifier standard format.
- If the data contains a number of application identifiers matching any of the specifications below then they should be provided in this given order for maximum encoding efficiency:
  - `(01)9...(3103)...`
  - `(01)9...(3202)...`
  - `(01)9...(3203)...`
  - `(01)9...(310x/320x)...(11/13/15/17)...`
  - `(01)9...(310x/320x)...`
  - `(01)9...(392x)...`
  - `(01)9...(393x)...`
  - `(01)...`
- The **segments** option is used to specify the maximum number of segments per row which must be an even number. The default is *4*.
- The **linkage** option signifies the presence of a GS1 composite 2D component.

#### Examples

```
Data:    (01)95012345678903(3103)000123
Options: segments=4
Encoder: databarexpandedstacked
```

## GS1 DataBar Truncated

**GS1 DataBar Truncated** is a fixed-length, linear barcode symbology that can be used to encode a GTIN-14 for in-house applications.

Also known as: RSS-14 Truncated.

Variants:

- GS1 DataBar Stacked is a variant of GS1 DataBar Truncated for use where a taller, narrower symbol is required.
- GS1 DataBar Truncated Composite is a variant of GS1 DataBar Truncated that should be used when a CC-A or CC-B GS1 composite 2D component is required.

Standards: ISO/IEC 24724, ITS Reduced Space Symbology (RSS), AIM ISS - Reduced Space Symbology (RSS), GS1 General Specifications.

### Data and Options

- The data field input is provided in GS1 Application Identifier standard format and must be a solitary *AI (01)* with 13 or 14 digits of a GTIN, i.e. `(01)...`.
- If thirteen digits of *AI (01)* data are supplied then the check digit is calculated automatically, otherwise the digits are encoded as supplied.
- The **linkage** option signifies the presence of a GS1 composite 2D component.

### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    (01)24012345678905
Options:
Encoder: databartruncated
```

```
Data:    (01)2401234567890
Options:
Encoder: databartruncated
```

## GS1 DataBar Stacked

**GS1 DataBar Stacked** is a fixed-length, stacked linear barcode symbology that can be used to encode a GTIN-14 for in-house applications.

Also known as: RSS-14 Stacked.

Variants:

- GS1 DataBar Truncated is a variant of GS1 DataBar Stacked for use where a shorter, wider symbol is required.
- GS1 DataBar Stacked Composite is a variant of GS1 DataBar Stacked that should be used when a CC-A or CC-B GS1 composite 2D component is required.

Standards: ISO/IEC 24724, ITS Reduced Space Symbology (RSS), AIM ISS - Reduced Space Symbology (RSS), GS1 General Specifications.

#### Data and Options

- The data field input is provided in GS1 Application Identifier standard format and must be a solitary *AI (01)* with thirteen or fourteen digits of a GTIN, i.e. `(01)...`.
- If thirteen digits of *AI (01)* data are supplied then the check digit is calculated automatically, otherwise the digits are encoded as supplied.
- The **linkage** option signifies the presence of a GS1 composite 2D component.

#### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    (01)24012345678905
Options:
Encoder: databarstacked
```

```
Data:    (01)24012345678890
Options:
Encoder: databarstacked
```

## GS1 DataBar Limited

**GS1 DataBar Limited** is fixed-length, linear barcode symbology that can be used to encode a GTIN-14 beginning with *0* or *1* for in-house applications.

Also known as: RSS Limited.

Variants:

- GS1 DataBar Limited Composite is a variant of GS1 DataBar Limited that should be used when a CC-A or CC-B GS1 composite 2D component is required.

Standards: ISO/IEC 24724, ITS Reduced Space Symbology (RSS), AIM ISS - Reduced Space Symbology (RSS), GS1 General Specifications.

#### Data and Options

- The data field input is provided in GS1 Application Identifier standard format and must be a solitary *AI (01)* with thirteen or fourteen digits of a GTIN starting with *0* or *1*, i.e. `(01)0...` or `(01)1...`.
- If thirteen digits of *AI (01)* data are supplied then the check digit is calculated automatically, otherwise the digits are encoded as supplied.
- The **linkage** option signifies the presence of a GS1 composite 2D component.

#### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    (01)15012345678907
Options:
Encoder: databarlimited
```

```
Data:    (01)1501234567890
Options:
Encoder: databarlimited
```

# Postal Symbols

## Australia Post 4 State Customer Code

The **Australia Post 4 State Customer Code** is a barcode used by the Australian Postal Service to encode the data on letter mail.

### Data and Options

- The first two characters of the data field are digits used to specify the manditory FCC type of the symbols, either `11`, `45`, `59` or `67`.
- The next eight characters are digits that specify the manditory DPID.
- The number of remaining characters varies according to the given FCC code and these specify the contents of the customer information field in one of two alphabets:
  - The **custinfoenc** option should be supplied as `custinfoenc=numeric` if the customer information field is to be encoded using the numeric alphabet which can contain the digits 0-9.
  - Otherwise the customer information field is encoded using the default character encoding, `custinfoenc=character`, which permits any of the following characters:
  - Upper case letters `A-Z`
  - Lower case letters `a-z`
  - Digits `0-9`
  - Symbols `space` and `#`
- The maditory Reed-Solomon check bars are calculated automatically.

### Examples

FCC 62 symbol with character customer data:

```
Data:    6279438541AaaB 155
Options: custinfoenc=character
Encoder: auspost
```


FCC 59 symbol with numeric customer data:

```
Data:    593221132401234567
Options: custinfoenc=numeric
Encoder: auspost
```


## Deutsche Post Identcode

**Deutsche Post Identcode** is an implementation of the Interleaved 2 of 5 barcode symbology that is used by German Post for mail routing.

Also known as: DHL Identcode.

### Data and Options

- The data consists of a consecutive string of eleven or twelve digits consisting of:
  - Two-digit primary distribution centre identifier
  - Three-digit customer identifier
  - Six-digit mail piece identifier
  - One-digit check digit (may be omitted)
- If eleven digits are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.

#### Example

Identical symbols, input provided with an without a check digit:

```
Data:    563102430313
Options: includetext
Encoder: identcode
```

```
Data:    56310243031
Options: includetext
Encoder: identcode
```


## Deutsche Post Leitcode

The **Deutsche Post Leitcode** barcode symbology is an implementation of the Interleaved 2 of 5 barcode that is used by German Post for mail routing.

Also known as: DHL Leitcode.

#### Data and Options

- The data consists of a consecutive string of thirteen or fourteen digits consisting of:
  - Five-digit postal code
  - Three-digit street identifier
  - Three-digit house number
  - Two-digit product code
  - One-digit check digit (may be omitted)
- If thirteen digits are supplied then the check digit is calculated automatically. Otherwise the provided check digit must be correct.

#### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    21348075016401
Options: includetext
Encoder: leitcode
```

```
Data:    2134807501640
Options: includetext
Encoder: leitcode
```

## Japan Post 4 State Barcode

The **Japan Post 4 state barcode** symbology is used by the Japan Post service to encode the delivery point identifier on letter mail.

### Data and Options

- The data may contain any of the following characters:
  - Capital letters `A-Z`
  - Digits `0-9`
  - Hyphen `-`

### Example

```
Data:    6540123789-A-K-Z
Options:
Encoder: japanpost
```

## MaxiCode

The **MaxiCode** barcode symbology is a 2D barcode based on a hexagonal matrix surrounding a bulls eye pattern. It can encode a structured carrier message and full 256 character extended-ASCII.

Also known as: UPS Code, Code 6, Dense Code.

Standards: ISO/IEC 16023, ANSI/AIM BC10 - ISS MaxiCode.

### Data and Options

- The **mode** option is used to specify how the data is structured in the symbol:
  - `mode=2` - Formatted data containing a Structured Carrier Message with a numeric (US domestic) postal code.
  - `mode=3` - Formatted data containing a Structured Carrier Message with an alphanumeric (international) postal code.
  - `mode=4` - Unstructured extended ASCII data using standard error correction.
  - `mode=5` - Unstructured extended ASCII data using enhanced error correction.
  - `mode=6` - Barcode reader programming.
- If **mode** is unspecified the encoder will default to selecting `mode=5` if the encoded length of the input data permits enhanced error correction, otherwise it will select `mode=4` which provides standard error correction.
- The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- If `mode=4`, `mode=5` or `mode=6` the data field may contain any extended ASCII data.
- If `mode=2` or `mode=3` the data field must begin with a properly structured carrier message, followed by any extended ASCII data.
- The structured carrier message contains a postal code, three-digit class of service and a three-digit ISO country code separated by *GS* (ASCII 29) characters. It is formatted in the data field as follows: `[postal code]^029[country code]^029[service class]^029`. If `mode=2` the postcode must be numeric, whilst if `mode=3` the postcode may contain up to six digits, upper case letters and spaces.
- Alternatively, messages may begin with the special application field identifier `[)>{RS}01{GS}yy` where `{RS}` represents ASCII value 30, `{GS}` represents ASCII value 29 and `yy` is a two-digit year. In parse mode this is represented as `[)>^03001^0299`. If `mode=2` or `mode=3` this must be immediately followed by the structured carrier message.

#### Examples

```
Data:    This is MaxiCode
Options:
Encoder: maxicode
```

```
Data:    This is Maxi^067ode
Options: parse
Encoder: maxicode
```

```
Data:    152382802^029840^029001^0291Z00004951^029UPSN^02906X610^029159^0291234567^0291/1^029^029Y^0296
Options: mode=2 parse
Encoder: maxicode
```

```
Data:    ABC123^029840^029001^0291Z00004951^029UPSN^02906X610^029159^0291234567^0291/1^029^029Y^029634
Options: mode=3 parse
Encoder: maxicode
```

```
Data:    [\)>^03001^02996152382802^029840^029001^0291Z00004951^029UPSN^02906X610^029159^0291234567^0291
Options: mode=2 parse
Encoder: maxicode
```

## Royal Mail 4 State Customer Code

The **Royal Mail 4 State Customer Code** is a barcode symbology used by the British Postal Service to encode the postcode and delivery point identifier on letter mail.

Also known as: RM4SCC, CBC.

#### Data and Options

- The data may contain any of the following characters from the postcode and DPID:
  - Capital letters A-Z
  - Digits 0-9

- The maditory checksum digit is calculated automatically and must not be included in the data field

#### Example

```
Data:    LE28HS9Z
Options: includetext
Encoder: royalmail
```


## Royal TNT Post 4 state barcode

The **Royal TNT Post 4 state barcode** symbology is used by the Dutch Postal Service to encode the delivery point identifier on letter mail.

Also known as: KIX.

#### Data and Options

- The data may contain any of the following characters from the DPID:
  - Capital letters `A-Z`
  - Digits `0-9`

#### Example

```
Data:    1231FZ13XHS
Options: includetext
Encoder: kix
```


## USPS Intelligent Mail

The **USPS Intelligent Mail** barcode is used by the US Postal service to encode the delivery and sender information on letter mail.

Also known as: USPS OneCode.

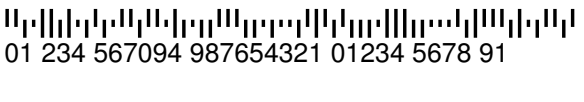
Standards: USPS-STD-11.

#### Data and Options

- The data contains 31 digits representing the following:
  - Barcode Identifier - two digits
  - Service Type Identifier - three digits
  - Mailer ID, Sequence Number - either six then nine digits respectively or nine then six digits respectively
  - Delivery Point ZIP Code - eleven digits
- The mandatory checksum digit is calculated automatically and must not be included in the data field.

#### Example

```
Data:    01234567094987654321012345678 91
Options: includetext
Encoder: onecode
```



## USPS POSTNET

The **USPS POSTNET** barcode symbology is used by the US Postal service to encode the ZIP code information on letter mail.

### Data and Options

- The data field contains the digits from the ZIP code, without dashes.
- The mandatory checksum is calculated automatically and must not be included in the data field.

### Example

```
Data:    12345123412
Options:
Encoder: postnet
```

## USPS PLANET

The **USPS PLANET** barcode symbology is used by the US Postal service to encode the ZIP code information on letter mail.

### Data and Options

- The data field contains eleven or thirteen digits, without dashes.
- The mandatory checksum is calculated automatically and must not be included in the data field.

### Example

```
Data:   01234567890
Options:
Encoder: planet
```

## USPS FIM Symbols

The **USPS FIM** encoder is used to generate static predefined barcode symbols.

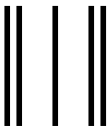
### Data and Options

- The data field accepts one of the following values:
  - `fima` - US Postal Service FIM-A symbol
  - `fimb` - US Postal Service FIM-B symbol
  - `fimc` - US Postal Service FIM-C symbol
  - `fimd` - US Postal Service FIM-D symbol

### Examples

A USPS FIM A symbol:

```
Data:    fima
Options:
Encoder: symbol
```

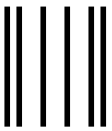


A USPS FIM B symbol:

```
Data:    fimb
Options:
Encoder: symbol
```

A USPS FIM C symbol:

```
Data:    fimc
Options:
Encoder: symbol
```



A USPS FIM D symbol:

```
Data:    fimd
Options:
Encoder: symbol
```

# Pharmaceutical Symbols

## Italian Pharmacode

**Italian Pharmacode** is a discrete, fixed length, self-checking barcode symbology used for pharmaceutical products in Italy.

Also known as: Code 32, IMH, Radix 32.

### Data and Options

- The data field must contain either eight or nine digits from the code. The leading *A* which is provided in some applications must be omitted.
- The mandatory check digit is calculated automatically if it is not provided.

### Examples

Identical symbols, input provided with and without a check digit:

```
Data:    012345676
Options: includetext
Encoder: code32
```

```
Data:    01234567
Options: includetext
Encoder: code32
```


## Pharmacode

**Pharmacode** is a binary barcode symbology that is used by the Pharmaceutical industry.

Also known as: Pharmaceutical Binary Code.

Variants:

- Two-track Pharmacode is a variant of the Pharmacode barcode.

### Data and Options

- The data field must contain a number between `3` and `131070` inclusive.
- The **nwidth**, **wwidth** and **swidth** options can be used to specify a custom width (in points) for the narrow bars, wide bars and inter-bar spaces respectively.

### Example

```
Data:    117480
Options:
Encoder: pharmacode
```

## Two-Track Pharmacode

**Two-Track Pharmacode** is a binary barcode symbology used by the Pharmaceutical industry.

Also known as: Two-track Pharmaceutical Binary Code.

Variants:

- Pharmacode is a variant of the Two-track Pharmacode barcode.

### Data and Options

- The data field must contain a number between `4` and `64570080` inclusive.

### Example

```
Data:    117480
Options:
Encoder: pharmacode2
```

## PZN

**PZN** is a discrete, fixed length, self-checking barcode symbology used for pharmaceutical products in Germany.

Also known as: Pharmazentralnummer.

Variants:

- PZN-7.
- PZN-8.

### Data and Options

- For the default PZN7 encoding, the data field must contain six digits or seven digits.
- The **pzn8** option specifies that a PZN8 symbol is required, in which case the data field must contain seven digits or eight digits.
- The mandatory check digit is calculated automatically if not provided.
- Note: by definition, not all six-digit or seven-digit number sequences are valid inputs.

### Examples

Identical PZN7 symbols, input provided with and without a check digit:

```
Data:    1234562
Options: includetext
Encoder: pzn
```

```
Data:    123456
Options: includetext
Encoder: pzn
```


Identical PZN8 symbols, input provided with and without a check digit:

```
Data:    0275808
Options: pzn8 includetext
Encoder: pzn
```

```
Data:    02758089
Options: pzn8 includetext
Encoder: pzn
```


# HIBC Symbols

**HIBC barcodes** use a number of general symbologies as carrier symbols for data structured according to the LIC and PAS structured data definitions.

Variants:

- HIBC Code 39 is a variant of Code 39.
- HIBC Code 128 is a variant of Code 128.
- HIBC PDF417 is a variant of PDF417.
- HIBC MicroPDF417 is a variant of MicroPDF417.
- HIBC QR Code is a variant of QR Code.
- HIBC Data Matrix is a variant of Data Matrix.
- HIBC Codablock F is a variant of Codablock F.

Standards: ANSI/HIBC Provider Applications Standard, ANSI/HIBC Supplier Labelling Standard, ANSI/HIBC Positive Identification for Patient Safety, ANSI/HIBC Syntax Standard.

#### Data and Options

- The data should be pre-encoded to describe the intended barcode content.
- The HIBC + character is prefixed automatically.
- The mandatory HIBC check character is automatically appended to the input.

## HIBC Code 39

```
Data:    A123BJC5D6E71
Options: includetext
Encoder: hibccode39
```


## HIBC Code 128

```
Data:    A123BJC5D6E71
Options: includetext
Encoder: hibccode128
```


## HIBC PDF417

```
Data:    A123BJC5D6E71
Options:
Encoder: hibcpdf417
```

## HIBC MicroPDF417

```
Data:    A123BJC5D6E71
Options:
```

```
Encoder: hibcmicropdf417
```

### HIBC QR Code

```
Data:    A123BJC5D6E71
Options:
Encoder: hibcqrcode
```

### HIBC Data Matrix

```
Data:    A123BJC5D6E71
Options:
Encoder: hibcdatamatrix
```

### HIBC Codablock F

```
Data:    A123BJC5D6E71
Options:
Encoder: hibccodablockf
```

## Less-used Symbols

### BC412

The **BC412** barcode symbology is single width, variable length barcode that is used for silicon wafer identification by the semiconductor manufacturing industry.

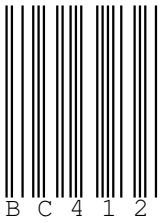
Also known as: BC412 SEMI, BC412 IBM.

#### Data and Options

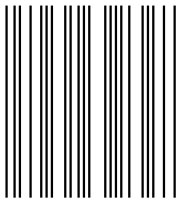
- The data field can hold any of the following:
  - Numbers 0-9
  - Capital letters A-Z, excluding O
- The **includestartstop** option enables the display of start and stop bars.
- The **includecheck** option calculates the check character.
- The **includecheckintext** option makes the calculated checksum appear in the human readable text.
- The **semi** option enables conformance to the SEMI standard by enabling start and stop bars as well as a check character.
- The **inkspread** option can be used to adjust the width of the bars.

#### Examples

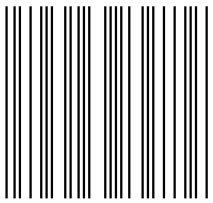
```
Data:    BC412
Options: includecheck
Encoder: bc412
```




```
Data:    BC412
Options: includestartstop
Encoder: bc412
```



```
Data:    BC412
Options: semi
Encoder: bc412
```



## Channel Code

**Channel Code** is a linear, continuous, self-checking, bidirectional barcode symbology that encodes between two and seven digits in a short space.

Standards: ANSI/AIM BC12 - USS Channel Code.

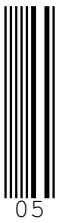
#### Data and Options

- The data field can hold zero prefixed values from any of the following ranges:
  - Channel 3: `00-26`
  - Channel 4: `000-292`
  - Channel 5: `0000-3493`
  - Channel 6: `00000-44072`
  - Channel 7: `000000-576688`
  - Channel 8: `0000000-7742862`
- The channel is determined to be one more than the number of digits given in the data field.
- The **shortfinder** option generates a symbol with a shortened finder pattern.
- The **includecheck** option appends an optional check bar suffix.

#### Examples

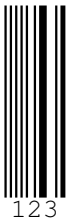
A channel 3 symbol holding the value five:

```
Data:    05
Options: includetext
Encoder: channelcode
```




A channel 4 symbol holding the value 123:

```
Data:    123
Options: includetext
Encoder: channelcode
```




A channel 4 symbol holding the value five including optional check bars:

```
Data:    005
Options: includetext includecheck
Encoder: channelcode
```


A channel 3 symbol holding the value 26 with a shorteded finder pattern:

```
Data:    26
Options: shortfinder includetext
Encoder: channelcode
```


## Codabar

**Codabar** is a linear, discrete, self-checking, bidirectional barcode symbology that can encode digits, six symbols and four delimiter characters. It is primarily used by libraries and blood banks, photo labs and FedEx airbills.

Also known as: Rationalized Codabar, Ames Code, NW-7, USD-4, USS-Codabar, ABC Codabar, Monarch, Code 2 of 7.

Standards: ANSI/AIM BC3 - USS Codabar, BS EN 798.

### Data and Options

- The data field must start and stop with one of the following delimiters
  - `ABCD`
  - `TNE*` (with the *altstartstop* option)
- The data field can otherwise hold any of the following
  - Digits 0-9
  - Symbols `-$:/.+`
- The **altstartstop** option specifies that the alternative set of delimiter characters is in use.
- The **includecheck** option calculates the check digit.
- The **includecheckintext** option makes the calculated check characters appear in the human readable text.

### Example

```
Data:    A0123456789B
Options: includecheck includetext includecheckintext
Encoder: rationalizedCodabar
```


## Codablock F

The **Codablock F** barcode symbology is 2D stacked-linear barcode that consists of a number of stacked Code 128 symbols. It can encode full 256 character extended-ASCII.

Variants:

- HIBC Codablock F is a variant of Codablock F that should be used when encoding HIBC formatted data.

Standards: USS Codablock F.

### Data and Options

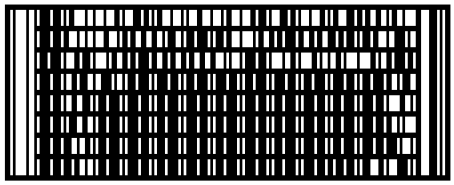
- The data field can consist of any extended-ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- *FNC4* function characters will be inserted automatically to allow the encoding of extended ASCII characters.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1` or `^FNC3`.
- The **columns** option specifies the number of columns in the symbol, default 8.
- The **rows** option specifies the number of rows in the symbol, between 2 and 44.
- If **rows** is unspecified the encoder will default to the smallest number of rows that can hold the given data.
- The **raw** option denotes that the data field is providing the input as a pre-encoded codewords in `^NNN` format, suitable for direct low-level encoding.
- The **rowheight** option specifies the height of the bars in each row in points. The default is 10.
- The **sepheight** option specifies the height of the separator bars enclosing the rows in points. The default is 1.

### Examples

```
Data:    Codablock F
Options:
Encoder: codablockf
```

```
Data:    CODABLOCK F 345678901234567890100 40digit
Options: columns=8 rows=5
Encoder: codablockf
```

```
Data:    Short bars, fat seperators
Options: columns=10 rows=8 rowheight=6 sepheight=2
Encoder: codablockf
```



## Code 11

**Code 11** is a linear, discrete, non-self-checking, bidirectional, numeric barcode symbology that is primarily used for labelling telecommunication equipment.

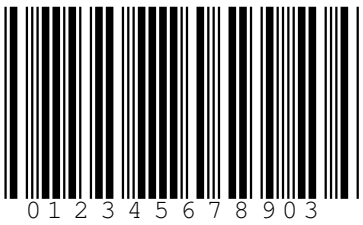
Also known as: USD-8.

### Data and Options

- The data consists of digits and the dash character -.
- The **includecheck** option calculates the check digits.
- For less than 10 data digits a single check digit is used.
- For 10 or more data digits two check digits are used.

### Example

```
Data:    0123456789
Options: includecheck includetext includecheckintext
Encoder: code11
```




## Code 16K

The **Code 16K** barcode symbology is 2D stacked-linear barcode that can encode full 256 character extended-ASCII with the use of the *FNC4* shift character.

Also known as: USS-16K

Standards: ANSI/AIM BC7 - USS Code 16K, BS EN 12323.

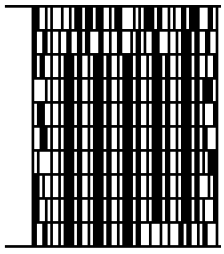
### Data and Options

- The data field can consist of any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- *FNC4* function characters will be inserted automatically to allow the encoding of extended ASCII characters.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1` through `^FNC3`.
- The **mode** option specifies the mode for the symbol. It is usual to leave this unspecified in which case the most appropriate mode that results in the shortest symbol is automatically selected based in the input data.
  - `mode=0` - Starting code set A
  - `mode=1` - Starting code set B
  - `mode=2` - Starting code set C
  - `mode=3` - Starting code set B with implied *FNC1*
  - `mode=4` - Starting code set C with implied *FNC1*
  - `mode=5` - Starting code set C with implied *Shift B*
  - `mode=6` - Starting code set C with implied *Double Shift B*
- The **pos** option specifies this symbol to be part of multi-part structured data. For example `pos=25` specifies this to be the second symbol in a group of five symbols.
- The **rows** option specifies the number of rows in the symbol, between two and sixteen.
- If **rows** is unspecified the encoder will default to the smallest number of rows that can hold the given data.
- The **raw** option denotes that the data field is providing the input as a pre-encoded codewords in `^NNN` format, suitable for direct low-level encoding.
- The **rowheight** option specifies the height of the bars in each row in points. The default is 10.
- The **sepheight** option specifies the height of the separator bars enclosing the rows in points. The default is 1.

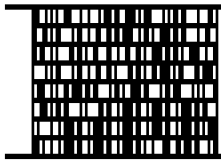
### Examples

```
Data:    Abcd-1234567890-wxyZ
Options:
Encoder: code16k
```

```
Data:    Code 16K
Options: rows=10
Encoder: code16k
```



```
Data:    Short bars, fat seperators
Options: rows=8 rowheight=5 sepheight=2
Encoder: code16k
```



## Code 25

**Code 2 of 5** is a simple low density numeric barcode symbology.

Also known as: Code 25, Industrial 2 of 5, Standard 2 of 5

Variants:

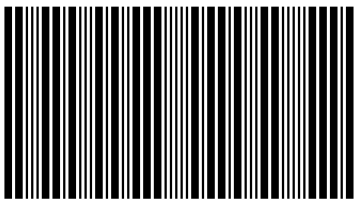
- IATA 2 of 5, Computer Identics 2 of 5.
- Datalogic 2 of 5.
- Matrix 2 of 5.
- COOP 2 of 5.

### Data and Options

- The data consists of any number of digits.
- The **includecheck** option calculates the check digit.
- The **version** option determines which variant to use:
  - `version=industrial` (default) - Industrial 2 of 5.
  - `version=iata` - *Deprecated: Use IATA 2 of 5*
  - `version=datalogic` - *Deprecated: Use Datalogic 2 of 5*
  - `version=matrix` - *Deprecated: Use Matrix 2 of 5*
  - `version=coop` - *Deprecated: Use COOP 2 of 5*

### Examples

```
Data:    01234567
Options:
Encoder: code2of5
```



## DotCode

The **DotCode** symbology is 2D matrix-style barcode that can encode full 256 character extended-ASCII.

Standards: AIM - ISS DotCode.

### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value. This is useful for specifying unprintable characters.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1`, `^FNC2`, `^FNC3`.
- The **fast** option is used to enable the optional high-speed symbol masking algorithm.
- The **ratio** is used to specify the aspect ratio of the symbol. The default is 1.5.
- The **columns** and **rows** options are used to specify the size of the symbol. When these are not specified a symbol is generated that maintains the selected aspect ratio.

#### Examples

```
Data:    This is DotCode
Options: fast inkspread=0.3
Encoder: dotcode
```

## Ultracode

The **Ultracode** symbology is a colour, 2D matrix-style barcode symbology that can encode full 256 character extended-ASCII.

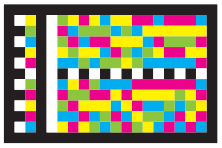
Standards: AIM ISS - Ultracode.

#### Data and Options

- The data field can contain any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1`, `^FNC3`.
- The **eclevel** option is used to specify the error correction level:
  - `eclevel=EC0` - Error detection only
  - `eclevel=EC1` - Low
  - `eclevel=EC2` - Medium (default)
  - `eclevel=EC3`
  - `eclevel=EC4`
  - `eclevel=EC5` - Highest

#### Examples

```
Data:    Nice colours!
Options: eclevel=EC3
Encoder: ultracode
```



## IATA 2 of 5

**IATA 2 of 5** is a variant of the Code 2 of 5 barcode symbology.

Also known as: Computer Identics 2 of 5.

Variants:

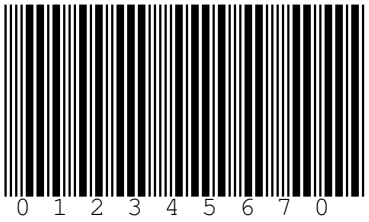
- Industrial 2 of 5, Standard 2 of 5.
- Datalogic 2 of 5.
- Matrix 2 of 5.
- COOP 2 of 5.

**Data and Options**

- The data consists of any number of digits.
- The **includecheck** option calculates the check digit.

**Examples**

```
Data:    01234567
Options: includetext includecheck includecheckintext
Encoder: iata2of5
```




## Matrix 2 of 5

**Matrix 2 of 5** is a variant of the Code 2 of 5 barcode symbology.

Variants:

- Industrial 2 of 5, Standard 2 of 5.
- IATA 2 of 5, Computer Identics 2 of 5.
- Datalogic 2 of 5.
- COOP 2 of 5.

**Data and Options**

- The data consists of any number of digits.
- The **includecheck** option calculates the check digit.

## Datalogic 2 of 5

**Datalogic 2 of 5** is a variant of the Code 2 of 5 barcode symbology.

Variants:

- Industrial 2 of 5, Standard 2 of 5.
- IATA 2 of 5, Computer Identics 2 of 5.
- Matrix 2 of 5.
- COOP 2 of 5.

**Data and Options**

- The data consists of any number of digits.
- The **includecheck** option calculates the check digit.

## COOP 2 of 5

**COOP 2 of 5** is a variant of the Code 2 of 5 barcode symbology.

Variants:

- Industrial 2 of 5, Standard 2 of 5.
- IATA 2 of 5, Computer Identics 2 of 5.
- Datalogic 2 of 5.

- Matrix 2 of 5.

#### Data and Options

- The data consists of any number of digits.
- The **includecheck** option calculates the check digit.

## Code 49

The **Code 49** barcode symbology is 2D stacked-linear barcode that can encode 128 character ASCII.

Also known as: USS-49.

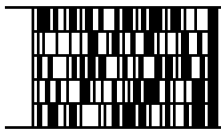
Standards: ANSI/AIM BC6 - USS Code 49.

#### Data and Options

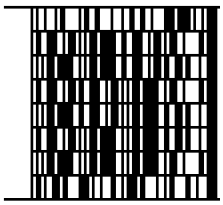
- The input can consist of any ASCII data.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1` through `^FNC3`.
- The **mode** option specifies the mode for the symbol. It is usual to leave this unspecified in which case the most appropriate mode that results in the shortest symbol is automatically selected based in the input data.
  - `mode=0` - regular alphanumeric mode
  - `mode=1` - append mode
  - `mode=2` - numeric mode
  - `mode=3` - group alphanumeric mode
  - `mode=4` - alphanumeric mode starting shift 1
  - `mode=5` - alphanumeric mode starting shift 2
  - `mode=6` - reserved
- The **pos** option specifies this symbol to be part of multi-part structured data, i.e. selecting `mode=3`. For example `pos=25` specifies this to be the second symbol in a group of five symbols.
- The **rows** option specifies the number of rows in the symbol, between *2* and *8*.
- If **rows** is unspecified the encoder will default to the smallest number of rows that can hold the given data.
- The **rowheight** option specifies the height of the bars in each row in points. The default is *10*.
- The **sepheight** option specifies the height of the separator bars enclosing the rows in points. The default is *1*.

#### Examples

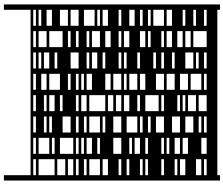
```
Data:    MULTIPLE ROWS IN CODE 49
Options:
Encoder: code49
```



```
Data:    CODE 49
Options: rows=8
Encoder: code49
```



```
Data:    Short bars, fat seperators
Options: rows=8 rowheight=6 sepheight=2
Encoder: code49
```



## Code One

**Code One** was the earliest public domain 2D matrix-style barcode. It is used by the health care and recycling industry and can encode full 256 character extended-ASCII.

Also known as: Code 1, Code 1S.

Standards: AIM USS - Code One.

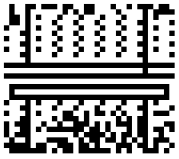
### Data and Options

- The data field can consist of any ASCII data for *standard* and *T-type* symbols.
- Note: *S-type* symbols are special in that they represent a numeric value so may only contain digits.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters, e.g. `^029` for *GS*, etc.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1`, `^FNC3`.
- The **version** option is used to specify the size and type of the symbol:
  - `A`, `B`, `C`, `D`, `E`, `F`, `G`, `H` - for standard format symbols (default automatic selection)
  - `version=T-16`, `version=T-32`, `version=T-48` - *T-type* symbols
  - `version=S-10`, `version=S-20`, `version=S-30` - *S-type* symbols

### Examples

```
Data:    Code One
Options:
Encoder: codeone
```

```
Data:    Code One
Options: version=C
Encoder: codeone
```



```
Data:    Code One
Options: version=T-32
Encoder: codeone
```

```
Data:    406990
Options: version=S-10
Encoder: codeone
```

## MSI Plessey

**MSI Plessey** is a continuous, non-self-checking, arbitrary length, numeric barcode symbology.

Also known as: MSI, MSI Modified Plessey.

Variants:

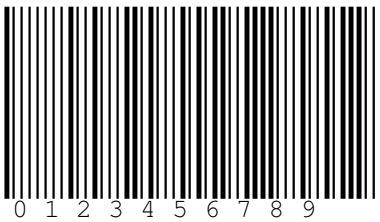
- Plessey (UK) is the original barcode upon which MSI Modified Plessey was based.

### Data and Options

- The data can consist of any number of digits.
- The **includecheck** option calculates the check digit or check digits.
- The **includecheckintext** option makes the calculated check characters appear in the human readable text.
- The **checktype** option is used to specify the type of checksum, either:
  - `checktype=mod10` (default)
  - `checktype=mod1010`
  - `checktype=mod11`
  - `checktype=ncrmod11`
  - `checktype=mod1110`
  - `checktype=ncrmod1110`
- The **badmod11** option allows a `checktype=mod11` checksum value of 10 to be encoded with a pair of check digits *10*. Normally in `checktype=mod11`, any input whose checksum evaluates to *10* is considered invalid having no correct representation.

### Examples

```
Data:    0123456789
Options: includecheck includetext
Encoder: msi
```




```
Data:    0123456789
Options: includecheck checktype=mod1110 includetext includecheckintext
Encoder: msi
```

```
Data:    0123456785
Options: includecheck checktype=mod11 badmod11 includetext includecheckintext
Encoder: msi
```


## Plessey

**Plessey** is a continuous, arbitrary length barcode symbology for encoding hexadecimal data.

Also known as: Anker Code.

Variants:

- MSI Modified Plessey is a variant of the Plessey (UK) barcode developed by the MSI Data Corporation.

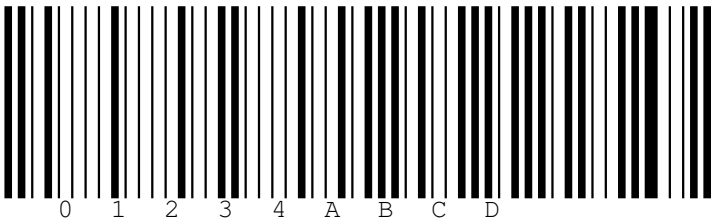
### Data and Options

- The data can contain any of the following:
  - Numbers `0-9`
  - Capital letters `A-F`
- Two manditory check characters implementing a CRC check are automatically included.
- The **includecheckintext** option makes the calculated check characters appear in the human readable text.
- The **unidirectional** option generates a unidirectional Plessey symbol.

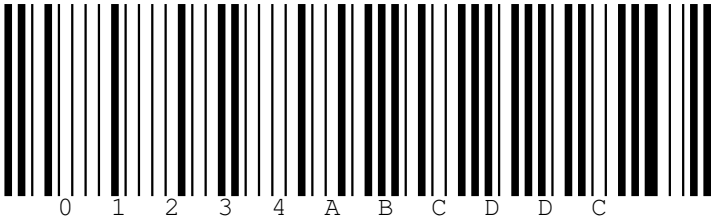
### Examples

Equivalent symbols, the latter displaying the two mandatory check characters:

```
Data:    01234ABCD
Options: includetext
Encoder: plessey
```

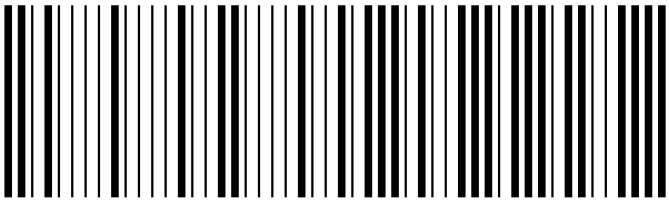



```
Data:    01234ABCD
Options: includetext includecheckintext
Encoder: plessey
```




A unidirectional symbol:

```
Data:    01234ABCD
Options: unidirectional
Encoder: plessey
```



## PosiCode

**PosiCode** is a continuous, variable length, non-self-checking, bidirectional barcode symbology that is designed for use within printing processes where it is difficult to precisely control the width of a bar.

Standards: ITS PosiCode.

### Data and Options

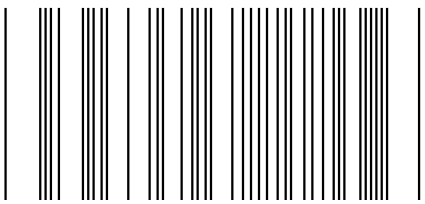
- The data field can hold the following:
  - For *standard* symbols: Any extended ASCII data. The default interpretation of data by readers is in accordance with ISO/IEC 8859-1.
  - *FNC4* function characters will be inserted automatically to allow the encoding of extended ASCII characters.
  - For *limited* symbols: letters `A-Z`, digits `0-9`, symbols `-` and `.`
- The **version** option is used to specify the variant of the symbol, either:
  - `version=a` (default)
  - `version=b`
  - `version=limiteda`
  - `version=limitedb`
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters, e.g. `^029` for *GS*, etc.
- When the **parsefnc** option is specified, non-data function characters can be specified by `^FNC1` through `^FNC3`.
- The **inkspread** option can be used to adjust the width of the bars.

### Example PosiCode

Equivalent ways to generate a PosiCode A symbol:

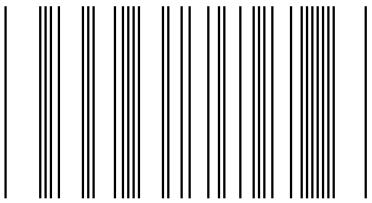
```
Data:    Abc123
Options:
Encoder: posicode
```

```
Data:    Abc123
Options: version=a
Encoder: posicode
```



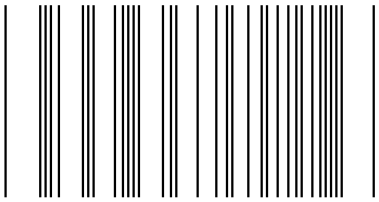
PosiCode A including a *GS* (ASCII 29) character:

```
Data:    AB^029CD
Options: parse
Encoder: posicode
```



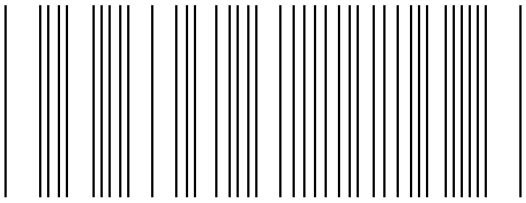
PosiCode A including an *FNC2* special character:

```
Data:    AB^FNC2CD
Options: parsefnc
Encoder: posicode
```



PosiCode B symbol with widened bars:

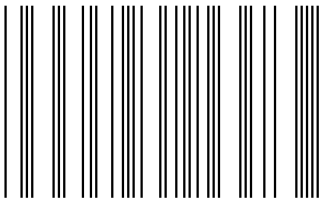
```
Data:    Abc123
Options: version=b inkspread=-1
Encoder: posicode
```



**Example Limited PosiCode**

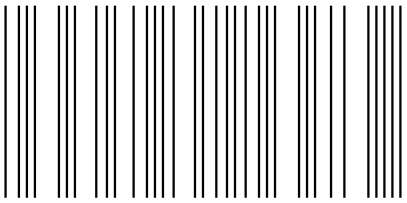
Limited PosiCode A with narrowed bars:

```
Data:    ABC-12.3
Options: version=limiteda
Encoder: posicode
```



Limited PosiCode B:

```
Data:    ABC-12.3
Options: version=limitedb
Encoder: posicode
```



## Telepen

**Telepen** is an arbitrary length barcode symbology for encoding all 128 ASCII characters without the need for shift characters.

Also known as: Telepen Alpha, Telepen Full ASCII.

Variants:

- Telepen Numeric.

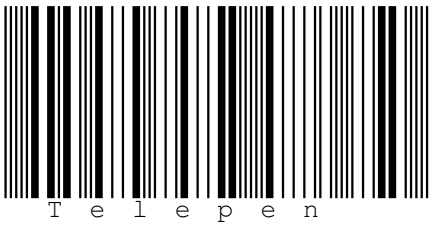
Standards: USS Telepen.

### Data and Options

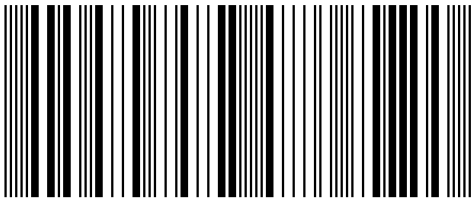
- The data can contain any standard ASCII data, values 0-127.
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The mandatory check digit is automatically included.
- *Deprecated: Use Telepen Numeric instead. When the **numeric** option is given, the data is read as either pairs of digits or `0X`, `1X`, etc. The singular values `^000` to `^016` can also be encoded using the* parse *option.*

### Examples

```
Data:    Telepen
Options: includetext
Encoder: telepen
```




```
Data:    Telepen^013
Options: parse
Encoder: telepen
```



## Telepen Numeric

**Telepen Numeric** is a variant of the Telepen symbology for efficient encoding of numeric data.

Variants:

- Telepen Alpha.

Standards: USS Telepen.

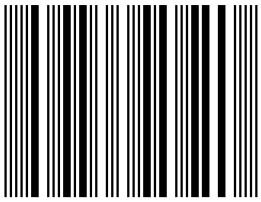
### Data and Options

- The data is provided as either pairs of digits or `0X`, `1X`, etc. The singular values `^000` to `^016` can also be encoded using the *parse* option.

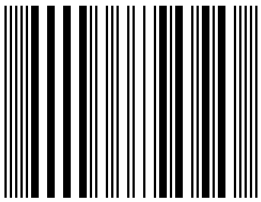
- When the **parse** option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.
- The mandatory check digit is automatically included.

**Examples**

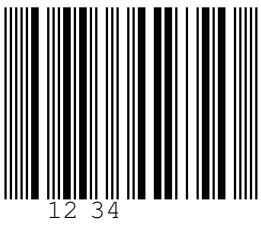
```
Data:    123456
Options:
Encoder: telepennumeric
```



```
Data:    1X345X
Options:
Encoder: telepennumeric
```



```
Data:    1234^005
Options: parse includetext
Encoder: telepennumeric
```




## GS1 Composite Symbols

**GS1 Composite** barcode symbologies consist of a primary component beneath a 2D component (variations of MicroPDF417 and PDF417) used to encode supplementary GS1 formatted data.

Variants:

- EAN-13 Composite is a variant of EAN-13.
- EAN-8 Composite is a variant of EAN-8.
- UPC-A Composite is a variant of UPC-A.
- UPC-E Composite is a variant of UPC-E.
- GS1 DataBar Omnidirectional Composite is a variant of GS1 DataBar Omnidirectional.
- GS1 DataBar Stacked Omnidirectional Composite is a variant of GS1 DataBar Stacked Omnidirectional.
- GS1 DataBar Expanded Composite is a variant of GS1 DataBar Expanded.
- GS1 DataBar Expanded Stacked Composite is a variant of GS1 DataBar Expanded Stacked.
- GS1 DataBar Truncated Composite is a variant of GS1 DataBar Truncated.
- GS1 DataBar Stacked Composite is a variant of GS1 DataBar Stacked.
- GS1 DataBar Limited Composite is a variant of GS1 DataBar Limited.
- GS1-128 Composite is a variant of GS1-128.

Standards: ISO/IEC 24723, ITS EAN.UCC Composite Symbology, AIM ISS - EAN.UCC Composite Symbology, GS1 General Specifications.

**Data and Options**

- The data field consists of a primary and secondary component separated by a pipe `|` character.
- The data for the primary component (preceding the pipe) is entered in a format identical to the corresponding non-composite barcode symbology.
- The data for the 2D component (following the pipe) is entered in GS1 Application Identifier standard format.
- For maximum efficiency, if the data for the 2D component contains a number of application identifiers matching any of the specifications below then they should be provided in this given order:
  - `(11)...(10)...`
  - `(17)...(10)...`
  - `(90){0-3 digits not starting 0}{upper alpha}...`
- The **ccversion** option is used to select a specific 2D component:
  - `ccversion=a` - CC-A
  - `ccversion=b` - CC-B
  - `ccversion=c` - CC-C (GS1-128 Composite only)
- If **ccversion** is not specified a CC-A component will be selected if the data will fit, otherwise a CC-B component will be used. In the case of GS1-128 Composite a CC-C component will be used if the data does not fit within either a CC-A or CC-B component.

## EAN-13 Composite

```
Data:    331234567890|(99)1234-abcd
Options: includetext guardwhitespace
Encoder: ean13composite
```


## EAN-8 Composite

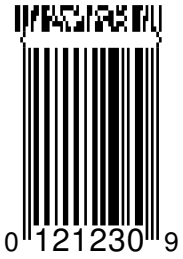
```
Data:    12345670|(21)A12345678
Options: includetext guardwhitespace
Encoder: ean8composite
```


## UPC-A Composite

```
Data:    01600033610|(99)1234-abcd
Options: includetext
Encoder: upcacomposite
```


### UPC-E Composite

```
Data:    0121230|(15)021231
Options: includetext
Encoder: upcecomposite
```




### GS1 DataBar Omnidirectional Composite

```
Data:    (01)03612345678904|(11)990102
Options:
Encoder: databaromnicomposite
```

### GS1 DataBar Stacked Omnidirectional Composite

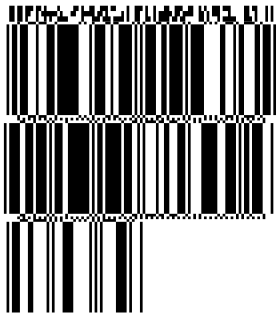
```
Data:    (01)03612345678904|(11)990102
Options:
Encoder: databarstackedomnicomposite
```

### GS1 DataBar Expanded Composite

```
Data:    (01)93712345678904(3103)001234|(91)1A2B3C4D5E
Options:
Encoder: databarexpandedcomposite
```

### GS1 DataBar Expanded Stacked Composite

```
Data:    (01)00012345678905(10)ABCDEF|(21)12345678
Options: segments=4
Encoder: databarexpandedstackedcomposite
```



### GS1 DataBar Truncated Composite

```
Data:    (01)03612345678904|(11)990102
Options:
Encoder: databartruncatedcomposite
```

### GS1 DataBar Stacked Composite

```
Data:    (01)03412345678900|(17)010200
Options:
Encoder: databarstackedcomposite
```

### GS1 DataBar Limited Composite

```
Data:    (01)03512345678907|(21)abcdefghijklmnopqrstuv
Options:
Encoder: databarlimitedcomposite
```

### GS1-128 Composite

GS1-128 Composite with a CC-A 2D component:

```
Data:    (01)03212345678906|(21)A1B2C3D4E5F6G7H8
Options:
Encoder: gs1-128composite
```

GS1-128 Composite with a CC-C 2D component:

```
Data:    (00)030123456789012340|(02)13012345678909(37)24(10)1234567ABCDEFG
Options: ccversion=c
Encoder: gs1-128composite
```

### CC-A

Isolated CC-A 2D component:

```
Data:    (01)95012345678903
Options: ccversion=a cccolumns=3
Encoder: gs1-cc
```

### CC-B

Isolated CC-B 2D component:

```
Data:    (01)95012345678903(3103)000123
Options: ccversion=b cccolumns=4
Encoder: gs1-cc
```

### CC-C

Isolated CC-C 2D component:

```
Data:    (02)13012345678909(37)24(10)1234567ABCDEFG
Options: ccversion=c cccolumns=5
Encoder: gs1-cc
```

## Raw Symbols

### DAFT

**DAFT** is an encoder for directly specifying the descender, ascender, full-height, tracker-bar succession for a custom 4 state barcode symbol.

#### Data and Options

- The data field contains a sequence of the characters `D`, `A`, `F` or `T` to denote the descender, ascender, full-height and tracker bars of a custom 4 state symbol.

#### Example

```
Data:    FATDAFTDAD
Options:
Encoder: daft
```

### Flattermarken

**Flattermarken** are identification marks used in book production that facilitate the proper arrangement of bound sections by a book binder.

#### Data and Options

- The data field can holding any sequence of digits corresponds to a 9 module width with the following meaning:
  - `1-9`: a single mark exists in the corresponding module position
  - `0`: unmarked sequence of modules
- The **inkspread** option can be used to adjust the width of the bars.
- If greater fidelity is required then the raw encoder should be used instead.

#### Example

```
Data:    1304
Options: inkspread=-1
Encoder: flattermarken
```

### Raw

The **raw** encoder is used for directly specifying the space/bar succession of a custom barcode symbol.

#### Data and Options

- The data field contains an alternating sequence of widths (1 to 9) for the bars and spaces of a custom symbol.

#### Example

```
Data:    331132131313411122131311333213114131131221323
Options: height=0.5
Encoder: raw
```

## Partial Symbols

### EAN-2

**EAN-2** is the two-digit add-on code that accompanies a EAN or UPC type barcode symbol such as an ISBN or ISSN.

Also known as: Two-Digit Add-On, Two-Digit Supplement, UPC-2

#### Data and Options

- The data field must contain two digits.
- The **includetext** option should normally be supplied.

#### Example

```
Data:    05
Options: includetext guardwhitespace
Encoder: ean2
```


### EAN-5

**EAN-5** is the five-digit add-on code that accompanies an EAN or UPC type barcode symbol such as an ISBN or ISSN.

Also known as: Five-Digit Add-On, Five-Digit Supplement, UPC-5

#### Data and Options

- The data field must contain five digits.
- The **includetext** option should normally be supplied.

#### Example

```
Data:    90200
Options: includetext guardwhitespace
Encoder: ean5
```


## GS1 Application Identifier Standard Format

Certain barcode symbologies (including GS1-128, GS1 DataBar Omnidirectional, GS1 DataMatrix, GS1 QR Code and GS1 Composite Symbols) represent standardized GS1 data and require that their data field is provided in GS1 Application Identifier standard format, consisting of a concatenated string of *AIs* along with their corresponding values.

The AIs are a set of approximately one hundred two-, three- or four-digit prefixes written within parentheses that represent physical attributes and business information, e.g.

- *(00)* is an eighteen-digit SSCC.
- *(01)* is a fourteen-digit GTIN.
- *(403)* is a variable-length routing code.

The following input represents GTIN *0061414199996*; Expiration Date *1 January 2010*; Batch *123ABC*; Serial *1234567890*:

```
(01)00614141999961(17)100101(10)123ABC(21)1234567890
```

Encoders for barcode symbologies that expect data in GS1 Application Identifier standard format will take care of parsing the input and inserting any necessary *FNC1* characters to delimit variable length fields.

### GS1 Application Identifier Definitions

The Application Identifier definitions are provided in the GS1 General Specifications. A summary is available here however this may be out of date.

# Chapter 6

# Options Reference

## Check Digits

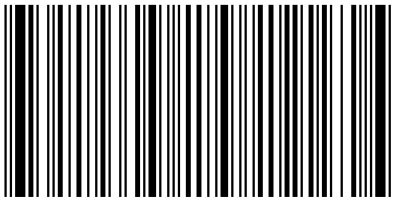
### includecheck

Generate check digit(s) for symbologies where the use of check digits is optional.

#### Example

Calculate the optional check characters of this Code 93 symbol:

```
Data:    CHECK ME OUT
Options: includecheck
Encoder: code93
```



### includecheckintext

Show the calculated check digit in the human readable text.

#### Notes

- For barcode symbologies where the check digit is not mandatory, this option must be used in combination with **includecheck**.
- If any part of the checksum does not have a printable representation then that part is not displayed.

#### Example

Display the check digit of this Royal Mail barcode:

```
Data:    LE28HS9Z
Options: includetext includecheckintext
Encoder: royalmail
```

# Input Processing

## parse

In supporting barcode symbologies, when the *parse* option is specified, any instances of `^NNN` in the data field are replaced with their equivalent ASCII value, useful for specifying unprintable characters.

### Example

Equivalent symbols:

```
Data:    This is Data Matrix
Options:
Encoder: datamatrix
```

```
Data:    This is ^068ata Matrix
Options: parse
Encoder: datamatrix
```

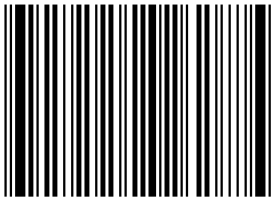
## parsefnc

In supporting barcode symbologies, when the **parsefnc** option is specified, non-data function characters can be specified by escaped combinations such as `^FNC1`, `^FNC4` and `^SFT/`.

### Example

Code 93 including a special shift combination `(/)A` representing *!*:

```
Data:    TERRY^SFT/A
Options: parsefnc includecheck
Encoder: code93
```



# Symbol Dimensions

## height

Height of longest bar, in inches.

### Example

A 1/2 inch tall EAN-13:

```
Data:    977147396801
Options: includetext height=0.5
Encoder: ean13
```


### width

Stretch the symbol to precisely this width, in inches.

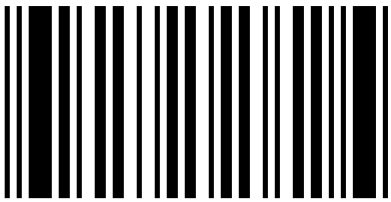
#### Notes

- This parameter literally stretches the symbol and text to the desired width which will may distort the human readable text.
- For information about resizing symbols read the article on resizing symbols.

#### Example

A 2 inch wide Code 93 symbol:

```
Data:    TERRY
Options: width=2
Encoder: code93
```



## Bar Properties

### inkspread

For linear barcodes, the amount by which to reduce the bar widths to compensate for inkspread, in points.

For matrix barcodes, the amount by which the reduce the width and height of dark modules to compensate for inkspread, in points.

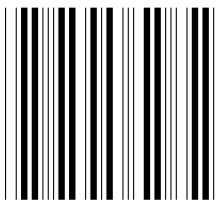
#### Notes

- Negative values will increase the bar width.

#### Example

Code 39 for a printer with very runny ink:

```
Data:    TEZ
Options: inkspread=0.6
Encoder: code39
```



### inkspreadh

For matrix barcodes, the amount by which the reduce the width of dark modules to compensate for inkspread, in points.

Note: inkspreadh is most useful for stacked-linear type barcodes such as PDF417 and Codablock F.

### inkspreadv

For matrix barcodes, the amount by which the reduce the height of dark modules to compensate for inkspread, in points.

## Text Properties

### includetext

Show human readable text for data in symbol.

#### Notes

- If a character in the data does not have a printable representation then it is not displayed

#### Example

Display the text encoded in this Code 39 symbol:

```
Data:    SEE ME
Options: includetext
Encoder: code39
```


### textfont

The font name for text.

#### Notes

- The font name must be the literal name of a PostScript font that is available to the system.
- This option should be used in combination with the **includetext** option.

#### Example

Customise the human readable text of this USPS POSTNET symbol

```
Data:    64501
Options: includetext textfont=Times-Roman textsize=9
Encoder: postnet
```

### textsize

The font size of the text in points.

#### Note

- This option should be used in combination with the **includetext** option.

#### Example

Customise the human readable text of this USPS POSTNET symbol

```
Data:    64501
Options: includetext textfont=Times-Roman textsize=9
Encoder: postnet
```


### textgaps

The inter-character spacing of the text.

#### Note

- This option should be used in combination with the **textxalign** option.

## Text Positioning

### textxalign

The **textxalign** option is used to specify where to horizontally position the text.

- `textxalign=offleft`
- `textxalign=left`
- `textxalign=center`
- `textxalign=right`
- `textxalign=offright`
- `textxalign=justify`

#### Notes

- By default (in the absence of **textxalign** or **textyalign**), each character of text is placed immediately below the corresponding modules where this is possible.
- Where there isn't such a direct relationship then the default is to position the text centrally beneath the symbol.

### textyalign

The **textyalign** option is used to specify where to vertically position the text.

- `textyalign=below`
- `textyalign=center`
- `textyalign=above`

#### Notes

- By default (in the absence of **textxalign** or **textyalign**), each character of text is placed immediately below the corresponding modules where this is possible.
- Where there isn't such a direct relationship then the default is to position the text centrally beneath the symbol.

### textxoffset

The horizontal position of the text in points relative to the default position.

### textyoffset

The vertical position of the text in points relative to the default position.

## Border Properties

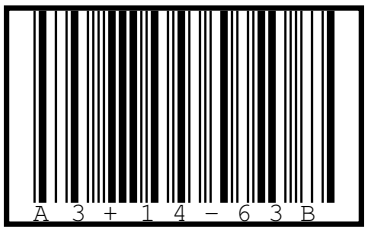
### showborder

Display a border around the symbol.

#### Example

Display a customised border around this Codabar symbol:

```
Data:    A3+14-63B
Options: includetext showborder borderwidth=2 borderbottom=8
Encoder: rationalizedCodabar
```




### borderwidth

Width of the border, in points.

### borderleft

Gap between the left edge of the border and the symbol, in points.

### borderright

Gap between the right edge of the border and the symbol, in points.

### bordertop

Gap between the top edge of the border and the symbol, in points.

### borderbottom

Gap between the bottom edge of the border and the symbol, in points.

## Symbol Colors

### barcolor

Color of the bars or dark modules, either as a hex `RRGGBB` value or a hex `CCMMYYKK` value.

### backgroundcolor

Color of the light background or light modules, either as a hex `RRGGBB` value or a hex `CCMMYYKK` value.

### bordercolor

Color of the border, either as a hex `RRGGBB` value or a hex `CCMMYYKK` value.

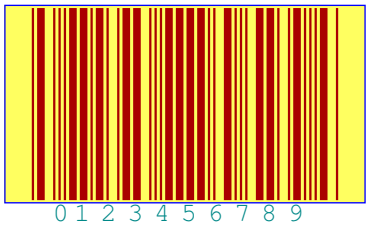
### textcolor

Color of the respective component, either as a hex `RRGGBB` value or a hex `CCMMYYKK` value.

#### Examples

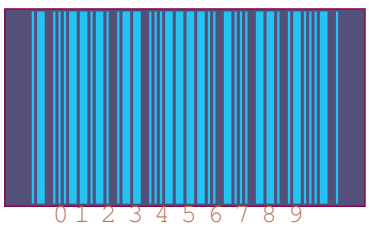
Colorized Code 11 symbol using the RGB colour space:

```
Data:    0123456789
Options: includetext barcolor=AA0000 textcolor=008888 backgroundcolor=FFFF60 showborder bordercolor=000
Encoder: code11
```




Colorized Code 11 symbol using the CMYK colour space:

```
Data:    0123456789
Options: includetext barcolor=AA000000 textcolor=00888844 backgroundcolor=CCCC6000 showborder bordercol
Encoder: code11
```




## EAN-UPC Add Ons

### addontextfont

The font name of the add on text.

#### Notes

- The font name must be the literal name of a PostScript available font.
- These options must be used in combination with the **includetext** option.

### addontextsize

The size of the add on text, in points

#### Notes

- These options must be used in combination with the **includetext** option.

#### Example

Customise the human readable text of this USPS POSTNET symbol

```
Data:    64501
Options: includetext textfont=Times-Roman textsize=9
Encoder: postnet
```


### addontextxoffset

Overrides the default horizontal positioning of the add on text.

### addontextyoffset

Overrides the default vertical positioning of the add on text.

## EAN-UPC Guards

### guardwhitespace

Display white space guards.

#### Example

ISBN-13 with tiny white space guard:

```
Data:    978-1-873671-00
Options: includetext guardwhitespace guardwidth=3 guardheight=3
Encoder: isbn
```

### guardwidth

Width of the whitespace guards, in points.

### guardheight

Height of the whitespace guards, in points.

### guardleftpos

Override the default horizontal position of the left whitespace guard.

### guardrightpos

Override the default horizontal position of the right whitespace guard.

### guardleftypos

Override the default vertical position of the left whitespace guard.

### guardrightypos

Override the default vertical position of the right whitespace guard.

#### Example

EAN-8 with very mangled white space guards:

```
Data:    01335583
Options: includetext guardwhitespace guardleftpos=14 guardrightpos=7 guardleftypos=15 guardrightypos=4
Encoder: ean8
```

# Chapter 7

# Knowledge Base

## FAQs

**How do I resize symbols without stretching the text?**

See this article on resizing symbols.

**Scanning ISBNs**

When an ISBN symbol is read by a barcode scanner that echos digits to a PC, the data string that is returned is most likely going to be the plain contents of the EAN-13 encoded symbol, i.e. 9781565924796, not 1-56592-479-7. Whether the scanner returns the former string or the latter they nevertheless represent the one same value despite appearing somewhat different.

**How do I integrate barcodes into my website or application?**

BWIPP is essentially a versatile library and is not necessarily a "turn key" solution by itself. https://groups.google.com/d/topic/postscriptbarcode/UOmONFc6cGQ/discussion

If you are a programmer then there are several language bindings that allow you to work with BWIPP without direct knowledge of PostScript. If you are not a programmer then there are a number of frontends that may be useful.

## Resizing Symbols

To create a barcode of some required width and height (without stretching the text) perform the following steps, in order.

Starting with this example:

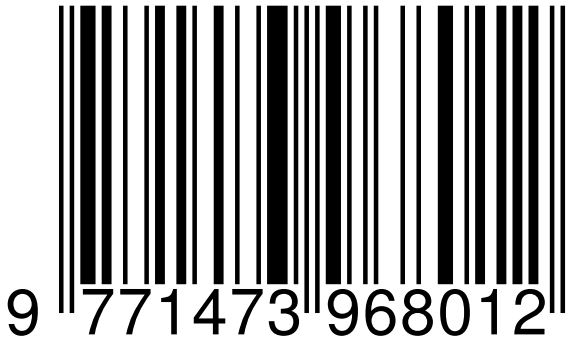
```
0 0 moveto (977147396801) (includetext)
/ean13 /uk.co.terryburton.bwipp findresource exec
```


Find the uniform (same x and y) scale factor that makes your output of the required **width**:

```
gsave
2 2 scale      %  <-- Add a line like this
0 0 moveto (977147396801) (includetext)
/ean13 /uk.co.terryburton.bwipp findresource exec
grestore
```


Add a height option that adjusts the bar height appropriately (taking the scaling into account):

```
gsave
2 2 scale
% Added height=0.8 option to adjust height
0 0 moveto (977147396801) (includetext height=0.8)
/ean13 /uk.co.terryburton.bwipp findresource exec
grestore
```




The result should now be of the intended dimensions with properly scaled (not stretched) text.

## Developing a Frontend to BWIPP

There are a number of frontends to BWIPP that vary in terms of the functionality that they expose and the way that they express this through their API or GUI, etc.

It would be nice to unify some of these projects but in the meantime this document attempts to provide some guidelines to apply when developing something that places BWIPP in the hands of developers and users.

The author would ideally like any language binding, library or graphical frontend to be representative of the complete functionality of the BWIPP resource and to be maintainable with minimal effort and these guideline help to achieve this goal.

#### Make Early Contact with the BWIPP Author

Contact the author of BWIPP whilst you're still experimenting. I will try not to insist on my own way as it's you that will end up supporting your creation so I want you to be happy with it, but it will help everyone if there is some consistency between your code and the next person's.

**Author's commitment:** If I know about your project then I will make a best efforts commitment to assist with end user support and developer support for any library or application that makes a genuine attempt to adopt the principles given here. Such projects should also feel free to adopt the BWIPP mailing list if they are so inclined and to request access to extend this wiki.

### Use the BWIPP C helper library and bindings...

Be aware that we have produced a C library and language-specific bindings with a common API to help with manipulating the BWIPP resources: https://github.com/bwipp/postscriptbarcode/tree/master/libs

You should attempt to use these where possible as it takes most of the pain out of working with the PostScript. If the API doesn't support something that you need then we can extend the interface as necessary.

### ... or at least parse the BWIPP metadata

If you choose to work directly with the PostScript then it is better to parse the inline metadata rather than embedding a load of static data in your code.

You should support new barcode formats automatically by scanning the barcode.ps metadata for `BEGIN/END ENCODER` blocks. From these extract descriptions, example data, options, etc. by using the `DESC`, `EXAM`, `EXOP`, ... stanzas within the `BEGIN/ENCODER ENCODER` blocks.

Example BWIPP metadata for an encoder:

```
% --BEGIN ENCODER ean8--
% --REQUIRES preamble raiseerror renlinear ean5 ean2--
% --DESC: EAN-8
% --EXAM: 02345673
% --EXOP: includetext guardwhitespace
% --RNDR: renlinear
... PostScript resource definition here ...
% --END ENCODER ean8--
```

The best strategy is for libraries and graphical frontends to be light on compiled-in data and can therefore be enhanced by simply replacing the barcode.ps file.

To fully meet this objective may require extending the barcode.ps metadata to describe the individual options that are available for each encoder. The BWIPP author is certainly interested in having such a discussion so please make contact regarding your requirements.

### Let Users Drive BWIPP Directly

Whether part of your design or as a fall back, allow advanced users to specify the data, options and encoder directly. This will allow them to access BWIPP functionality that you haven't anticipated or chosen to expose via your API or GUI.

### Use BWIPP's Error Reporting

Use the BWIPP error reporting mechanism to provide specific error messages to users so that they can understand why a given input is invalid.

The preferred way to do this is to wrap the BWIPP invocation in a "stopped context" which allows you to handle BWIPP-specific exceptions. For example, the following will invoke BWIPP and on error will emit formatted, descriptive text of the error (e.g. `BWIPP ERROR: EAN-13 must be 12 or 13 digits`) to STDERR which the calling program can recognise as an error and prompt the user:

```
{  % "try" BWIPP invocation
  0 0 moveto (ABC) () /code39 /uk.co.terryburton.bwipp findresource exec
  showpage
} stopped {  % "catch" all exceptions
  $error /errorname get dup length string cvs 0 6 getinterval (bwipp.) ne {
    stop  % Rethrow non-BWIPP exceptions
  } if
  % Handle BWIPP exceptions, e.g. emit formatted error to stderr
  (%stderr) (w) file
  dup (\nBWIPP ERROR: ) writestring
  dup $error /errorname get dup length string cvs writestring
  dup ( ) writestring
```

```
  dup $error /errorinfo get dup length string cvs writestring
  dup (\n) writestring
  dup flushfile
} if
```

Less advised, but which may be useful in some circumstances, it is possible to override the PostScript VM's default handleerror procedure to recognise and take some special action when handling BWIPP-specific exceptions. For example, the following will invoke `barcode.ps` and on error will emit formatted, descriptive text of the error (e.g. `BWIPP ERROR: EAN-13 must be 12 or 13 digits`) to STDERR which the calling program can recognise as an error and prompt the user:

```
%!PS
errordict begin
/handleerror {
  $error begin
  errorname dup length string cvs 0 6 getinterval (bwipp.) eq {
    (%stderr) (w) file
    dup (\nBWIPP ERROR: ) writestring
    dup errorname dup length string cvs writestring
    dup ( ) writestring
    dup errorinfo dup length string cvs writestring
    dup (\n) writestring
    dup flushfile end quit
  } if
  end //handleerror exec
} bind def
end

% If necessary, set up anything else specific to the environment just here.

% Include the BWIPP resource, either directly or from PS
(barcode.ps) run

% Now make the calls to BWIPP
0 0 moveto (ABC) () /code39 /uk.co.terryburton.bwipp findresource exec
```

### Locating the Resource

Allow the location of the `barcode.ps` file to be configured by the user so that non-admins users can provide a local version and distributions that deprecate bundled libraries can provide a separately packaged version.

In any case, use the following search order to locate the `barcode.ps` resource:

1. `[%USER_SPECIFIED_LOCATION%]`
2. `~/.[%APP_RC_DIRECTORY%]` (a user's own replacement)
3. `[%APP_INSTALL_DIR%]` (a version you have bundled)
4. `/usr/share/postscriptbarcode` (Fedora's postscriptbarcode package)
5. `/usr/share/libpostscriptbarcode` (Debian's libpostscriptbarcode package)

### Displaying the List of Supported Symbologies

To make the presentation of the list of barcode formats manageable any such list of barcodes should be rendered in the same/similar way as the web-based generator.

### Refer Users to the BWIPP Documentation

Point your users at the online BWIPP symbologies and options references.

The reference is written these in a way that is intended to be fairly environment agnostic but if you have any ideas or want to improve them in some way then please contribute.

- https://github.com/bwipp/postscriptbarcode/wiki/Symbologies-Reference

- https://github.com/bwipp/postscriptbarcode/wiki/Options-Reference

### Safe Argument Passing

Pass arguments to BWIPP in an injection-proof way that does not allow users to invoke arbitrary PostScript commands by means of un-escaped `)` or otherwise.

The best way is to "hexify" the data, options and encoder string data in your output, for example:

```
0 0 moveto
<3032333435363733>                          <-- Instead of (02345673)
<696e636c75646574657874>                    <-- Instead of (includetext)
<65616e38> cvn                              <-- Instead of /ean8
/uk.co.terryburton.bwipp findresource exec
```

Example Python:

```
import binascii, textwrap
def hexify(input):
  return textwrap.TextWrapper(subsequent_indent=' ', width=72). \
    fill('<' + binascii.hexlify(string) + '>')
```

Example Perl:

```
sub hexify {
  return '<'.(join "\n ", unpack '(A72)*', unpack 'H*', shift).'>';
}
```

# Chapter 8

# Acknowledgements

Many thanks for the technical contributions from:

- Martin Bailey
- Jean-François Barbeau
- Eric Belshaw
- Tim Brunson
- Hann-Huei Chiou
- John Desrosiers
- Lars DIECKOW
- Chapman Flack
- Anders Hammarquist
- Matthew Harmon
- Adrian Head
- Christian Hitz
- Ray Johnson
- Janos Juhasz
- Reinhold Kainhofer
- Gerd Knops
- Michael Landers
- Ross McFarland
- Rudolf Meier
- Greg Menke
- Matthew Newton
- Harald Oehlmann
- Herbert Voss
- Jörg Walter

Apologies to anybody that has been forgotten. Send a pull request or contact the author if that is the case.

# Chapter 9

# Cited-By

The following is a list of known references to Barcode Writer in Pure PostScript project and its derivatives.

If you are aware of any noteworthy additions to this list then please send a patch.

**Barcode Writer in Pure PostScript**


http://bwipp.terryburton.co.uk

Martínez, Juan J. (2004) "A Barcode Generator in Pure PostScript," http://blackshell.usebox.net/archive/a-barcode-generator-in-pure-postscript.html

Flack, Chapman. (2006) "Direct Use of the PostScript Language," http://www.anastigmatix.net/postscript/direct.html

Janssen, Mark. (2008) "Creating LTO barcodes," Foobar's Blog and Linkdump. http://blog.maniac.nl/2008/05/28/creating-lto-barcodes/

Janssen, Mark. (2009) "Webbased (PDF) LTO Barcode Generator," Foobar's Blog and Linkdump. http://blog.maniac.nl/webbased-pdf-lto-barcode-generator/

Rocholl, Johann C. (2009) "Robust 1D Barcode Recognition on Mobile Devices," http://www.vis.uni-stuttgart.de/uploads/tx_vispublications/thesis.pdf

Scarso, Luigi. (2009) "Una estensione di luatex: luatex lunatic," http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.188.5052&rep=rep1&type=pdf

Pluimers, Jeroen W. (2009) "Generating EAN-13 barcode EPS files for your article numbers," http://wiert.me/2009/11/30/generating-ean-13-barcode-eps-files-for-your-article-numbers/

Russell, Robert. (2010) "Barcodes in SAP with the Barcode Writer in Pure Postscript," http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/21446

Seichter, Dominik. (2010) "KBarcode4 Light Released," http://domseichter.blogspot.com/2010/08/kbarcode4-light-released.html

Willis, Nathan. (2010) "Barcode Writer in Pure PostScript," Worldlabel.com Incorporated. http://blog.worldlabel.com/2010/barcode-writer-in-pure-postscript.html

Russell, Robert. (2010) "How to Print(PDF) QR Codes in standard SAP," http://www.rjruss.info/2010/09/how-to-printpdf-qr-codes-in-standard.html

Zhao, Y., Sun, W. (2010) "Practice of Imposition and Illustrator Variable Data Plate Making with Barcode," Proceedings of 17th IAPRI World Conference on Packaging. ISBN 978-1-935068-36-5.

Russell, Robert. (2011) "More Barcodes with Barcode Writer in Pure Postscript," http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/22827

McNulty, John. (2011) "The Secret Lives of Objects," Sonic Arts Research Centre. Queen's University Belfast. http://www.robotmouth.com/papers_files/TSLOO.pdf

Russell, Robert. (2013) "Add FREE barcodes to the SAP Latin2 HP printer Driver," http://www.rjruss.info/2013/02/add-free-barcodes-to-sap-latin2-hp.html

Ehlenbroker J., Lohweg V. (2014) “microIDENT - A System for Simple Coding and Authentication of Documents,” Optical Document Security - The Conference on Optical Security and Counterfeit Detection IV.

Russell, Robert. (2014) “Example SAP Smartform with QRcodes using the Barcode Writer in Pure Postscript,” http://www.rjruss.info/2014/09/example-sap-smartform-with-qrcodes.html

Scarso, Luigi. (2015) “Two applications of SWIGLIB: GraphicsMagick and Ghostscript,” The TUGboat Journal, Volume 36, Number 3. p. 237.

Ehlenbroker J., Lohweg V. (2016) “System for simple coding, authentication and copy detection of printed documents,” U.S. Patent Application 15/114,422, published December 2016.

**pst-barcode**

http://www.ctan.org/tex-archive/graphics/pstricks/contrib/pst-barcode/

Goossens, M., Mittelbach F., Rahtz, S., Roegel, D. (2007) “The LaTeX Graphics Companion,” Addison Wesley. ISBN 978-0-321-50892-8.

Robbers, Yuri & Skjold, Annemarie. (2007) “Creating Book Covers using PSTricks,” The PracTex Journal, Number 1.

Thompson, Paul A. (2008) “Clinical trials management on the internet - II. Using LATEX, PostScript, and SAS to produce barcode label sheets,” The PracTeX Journal, Number 3.

Secondo, Stefano. (2009) “Cover Letter With Style - Part Six,” http://stefano.italians.nl/archives/65

Voß, Herbert. (2010) “The current state of the PSTricks project,” The TUGboat Journal, Volume 31i, Number 1. p. 36.

Brampton, Andrew. (2010) “LaTeX QR Based Business Card,” The Website of Andrew Brampton. http://bramp.net/blog/latex-qr-based-business-card

Pascal. (2011) “QR Code with Latex,” http://xaphire.de/recipes/?p=344

Voß, Herbert. (2011) “Ch 26: pst-barcode - Bar codes” in “PSTricks. Graphics and PostScript for TeX and LaTeX,” Cambridge: UIT Cambridge. pp. 497-508.

Molnar, Peter. (2013) “Processing class assignments on paper (Part I),” Blog: I’d rather be programming... http://blog.petermolnar.us/2013/03/processing-class-assignments-on-paper.html

Voß, Herbert. (2013) “QR-Codes im Rand ausgeben [QR Codes in the Margin],” Die TEXnische Komödie 4/2013. pp. 34–37. http://archiv.dante.de/DTK/PDF/komoedie_2013_4.pdf

Fischer, Ulrike. (2014) “biblatex variations,” The TUGboat Journal, Volume 35, Number 3. pp. 256-260.

Voß, Herbert. (2016) “Ch 25: pst-barcode: Barcodes” in “PSTricks. Grafik mit PostScript für TeX und LaTeX,” Lehmanns Media. pp. 535-548.

Tibi, Daniel. (2016) “Easy Codes: Generating Barcodes and QR Codes With LaTeX.” Ubuntu User, Issue 28 (Spring 2016). Linux New Media USA LLC. pp. 45-47.

**Barcode Writer in Pure JavaScript**

https://github.com/metafloor/bwip-js

Russell, Robert. (2015) “Barcodes in SAP with the Barcode Writer in Pure Postscript Updated and Also Available in JavaScript,” http://www.rjruss.info/2015/04/barcodes-in-sap-with-barcode-writer-in.html

**Scribus Barcode Generator Plugin**

http://documentation.scribus.net/index.php/Barcode_Generator

“Libre Graphics: Scribus. Open Source Desktop Publishing Turns Pro,” Linux Format, Issue 80 (June 2006). Future Publishing. p. 54.

Byfield, Bruce. (2011) “Four Hidden Tools in Scribus,” Linux Journal, November 2011. Belltown Media Inc. http://www.linuxjournal.com/content/four-hidden-tools-scribus

Willis, Nathan. (2013) “Scribus 1.4.3 adds color models and more,” LWN.net. Eklektix, Inc. https://lwn.net/Articles/563035/

White, Robert. (2015) “Beginning Scribus,” Apress Media, LLC. pp 300-301. ISBN 978-1-4842-0722-2.

### KBarcode

http://www.kbarcode.net/

Willis, Nathan. (2010) “Generating Barcodes with KBarcode,” Worldlabel.com Incorporated. http://blog.worldlabel.com/2010/generating-barcodes-with-kbarcode.html