tiaoma

A barcode generator for typst that provides type safe API bindings for Zint (GitHub) library through a WASM plugin.

See official Zint manual for a more in-depth description of supported functionality.

API

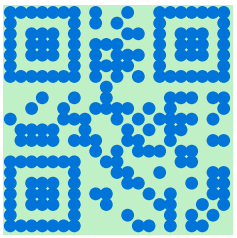
Some generators require additional configuration (such as composite codes), this can be achieved by passing options to Zint.

barcode

Draw a barcode SVG of any supported `symbology`.

Example:

```
tiaoma.barcode("12345678", "QRCode", options: (
  scale: 2.0,
  fg-color: blue,
  bg-color: green.lighten(70%),
  output-options: (
    barcode-dotty-mode: true
  ),
  dot-size: 1.2,
))
```



Parameters

barcode(
  data: `str`,
  symbology: `str`,
  options: `dictionary`,
  ..args: `any`
) -> `content`

data `str`

Data to encode.

symbology `str`

Symbology type name; must be one of supported types.

Example values: `"Code11"`, `"C25Standard"`, ...

options `dictionary`

Additional options to pass to Zint.

See the configuration section for details on available options and how to use them.

Default: `(:)`

..args `any`

Any additional arguments to forward to `image.decode` function.

dm-size

Returns `int` option value for given Data Matrix *width* and *height*.

Zint allows square and rectangular values to be enforced with `DM_SQUARE` and `DM_DMRE` Option 3 values.

Parameters

```
dm-size(
  height: int,
  width: int
) -> int
```

height `int`

Data Matrix height

width `int`

Data Matrix width

Shortcut functions

Most barcodes are supported through shortcut functions. They accept the same arguments as `barcode` function but don't require `symbology` to be specified.

Zint configuration

All exported functions support optionally providing the `options` dictionary which is passed to Zint. This provides means to fully configure generated images.

The following values are valid for the `options` dictionary:

Field	Type	Description	Default
height	`float`	Barcode height in X-dimensions (ignored for fixed-width barcodes)	`none`
scale	`float`	Scale factor when printing barcode, i.e. adjusts X-dimension	`1.0`
whitespace-width	`int`	Width in X-dimensions of whitespace to left & right of barcode	`0`
whitespace-height	`int`	Height in X-dimensions of whitespace above & below the barcode	`0`
border-width	`int`	Size of border in X-dimensions	`0`
output-options	`int` or `array` or `dictionary`	Various output parameters (bind, box etc, see below)	`0`
fg-color	`color`	foreground color	`black`
bg-color	`color`	background color	`white`
primary	`str`	Primary message data (MaxiCode, Composite)	`""`
option-1	`int`	Symbol-specific options (see manual)	`-1`
option-2	`int`	Symbol-specific options (see manual)	`0`
option-3	`int` or `str`	Symbol-specific options (see manual)	`0`
show-hrt	`bool`	Whether to show Human Readable Text (HRT)	`true`
input-mode	`int` or `string` or `array` or `dictionary`	Encoding of input data	`0`
eci	`int`	Extended Channel Interpretation.	`0`
dot-size	`float`	Size of dots used in BARCODE_DOTTY_MODE.	`4.0 / 5.0`
text-gap	`float`	Gap between barcode and text (HRT) in X-dimensions.	`1.0`
guard-descent	`float`	Height in X-dimensions that EAN/UPC guard bars descend.	`5.0`

Input Mode

Input mode options allow specifying how `Zint` should handle input data. `Zint` uses `int` bitflags for these, but `tiaoma` allows you to specify them using several other formats as documented below.

The following options are supported:

Input format (mutually exclusive)			
Constant	`int`	`str`	**Description**
`DATA_MODE`	`0`	`"data"`	Use full 8-bit range interpreted as binary data.
`UNICODE_MODE`	`1`	`"unicode"`	Use UTF-8 input.
`GS1_MODE`	`2`	`"gs1"`	Encode GS1 data using FNC1 characters.

Behavior customization			
Constant	`int`	`str`	**Description**
`ESCAPE_MODE`	`8`	`"escape"`	Process input data for escape sequences.
`GS1PARENS_MODE`	`16`	`"gs1-parentheses"`	Parentheses (round brackets) used in GS1 data instead of square brackets to delimit Application Identifiers (parentheses must not otherwise occur in the data).
`GS1NOCHECK_MODE`	`32`	`"gs1-no-check"`	Do not check GS1 data for validity, i.e. suppress checks for valid AIs and data lengths. Invalid characters (e.g. control characters, extended ASCII characters) are still checked for.
`HEIGHTPERROW_MODE`	`64`	`"height-per-row"`	Interpret the `height` variable as per-row rather than as overall height.
`FAST_MODE`	`128`	`"fast"`	Use faster if less optimal encodation for symbologies that support it (currently Data Matrix only).
`EXTRA_ESCAPE_MODE`	`256`	`"extra-escape"`	Undocumented.

String Value

`input_mode` of `str` type is assumed to be a *input format* value from the first table.

Array Value

`input_mode` of `array` type is assumed to be a list of `str` values from the above tables; individual constants will be converted to `int`s and unioned together.

Dictionary Value

`input_mode` of `dictionary` type is assumed to be `str`-`bool` pairs where keys are constants from the above table.

Additionally, *input format* can be specified as a `str` value paired to `"format"` key.

In other words, columns of the following table are equivalent:

`dictionary`	`array`	`str`	`int`
`("format": "data")`	`("data")`	`"data"`	`0`
`("unicode": true)`	`("unicode")`	`"unicode"`	`1`
`("gs1": true, "gs1-no-check": true)`	`("gs1", "gs1-no-check")`	N/A	`34`

Output Options

Output options allow specifying how `Zint` should generate the barcode/symbol.

Constant	`int`	`str`	Description
`BARCODE_BIND_TOP`	`1`	`"barcode-bind-top"`	Boundary bar *above* the symbol and between rows if stacking multiple symbols.
`BARCODE_BIND`	`2`	`"barcode-bind"`	Boundary bars *above* and *below* the symbol and between rows if stacking multiple symbols.

BARCODE_BOX	4	"barcode-box"	Add a box surrounding the symbol and whitespace.
SMALL_TEXT	32	"small-text"	Use a smaller font for the Human Readable Text.
BOLD_TEXT	64	"bold-text"	Embolden the Human Readable Text.
CMYK_COLOUR	128	"cmyk-color"	Select the CMYK colour space option for Encapsulated PostScript and TIF files.
BARCODE_DOTTY_MODE	256	"barcode-dotty-mode"	Plot a matrix symbol using dots rather than squares.
GS1_GS_SEPARATOR	512	"gs1-gs-separator"	Use GS instead of FNC1 as GS1 separator (Data Matrix only).
BARCODE_QUIET_ZONES	2048	"barcode-quiet-zones"	Add compliant quiet zones (additional to any specified whitespace).
BARCODE_NO_QUIET_ZONES	4096	"barcode-no-quiet-zones"	Disable quiet zones, notably those with defaults.
COMPLIANT_HEIGHT	8192	"compliant-height"	Warn if height not compliant and use standard height (if any) as default.
EANUPC_GUARD_WHITESPACE	16384	"ean-upc-guard-whitespace"	Add quiet zone indicators ("<" / ">") to HRT whitespace (EAN/UPC)
EMBED_VECTOR_FONT	32768	"embed-vector-font"	Embed font in vector output.

Array Value

`output_options` of `array` type assumed to be `str` values from the above table.

Dictionary Value

`output_options` of `dictionary` type assumed to be `str`-`bool` pairs where keys are constants from the above table.

Option 3

As there's constants associated with `option_3` values, this package allows specifying the value as either an `int` or a `str`.

The following table documents supported values and their `str` representations:

Constant	int	str	Description
DM_SQUARE	100	"square"	Only consider square versions on automatic symbol size selection
DM_DMRE	101	"rect"	Consider DMRE versions on automatic symbol size selection
DM_ISO_144	128	"iso-144"	Use ISO instead of "de facto" format for 144x144 (i.e. don't skew ECC)
ZINT_FULL_MULTIBYTE	200	"full-multibyte"	Enable Kanji/Hanzi compression for Latin-1 & binary data
ULTRA_COMPRESSION	128	"compression"	Enable Ultracode compression **(experimental)**

Examples

EAN (European Article Number)

EANX

Example:

```
#eanx("1234567890")
```

Result:

EAN-14

Example:

```
#ean14("1234567890")
```

Result:

EAN-13

Example:

```
#eanx("6975004310001")
```

Result:

EAN-8

Example:

```
#eanx("12345564")
```

Result:

EAN-5

Example:

```
#eanx("12345")
```

Result:

EAN-2

Example:

```
#eanx("12")
```

Result:

PDF417

Micro PDF417
Example:

```
#micro-pdf417("1234567890")
```

Result:

PDF417
Example:

```
#pdf417("1234567890")
```

Result:

Compact PDF417
Example:

```
#pdf417-comp("1234567890")
```

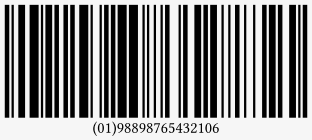
Result:

GS1

GS1-128
Example:

```
#gs1-128("[01]98898765432106")
```

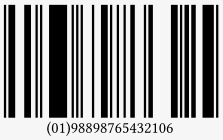
Result:



GS1 DataBar Omnidirectional
Example:

```
#dbar-omn("98898765432106")
```

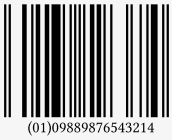
Result:



GS1 DataBar Limited
Example:

```
#dbar-ltd("988987654321")
```

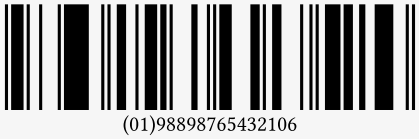
Result:



GS1 DataBar Expanded
Example:

```
#dbar-exp("[01]98898765432106")
```

Result:



GS1 DataBar Stacked
Example:

```
#dbar-stk("1234567890")
```

Result:

GS1 DataBar Stacked Omnidirectional
Example:

```
#dbar-omn-stk("1234567890")
```

Result:

Zint supports Omnidirectional, Limited, Expanded, Stacked and Composite Code variants of GS1. See configuration section for information on how to use them.

C25

Standard
Example:

```
#c25-standard("123")
```

Result:

Interleaved
Example:

```
#c25-inter("1234567890")
```

Result:

IATA
Example:

```
#c25-iata("1234567890")
```

Result:

Data Logic
Example:

```
#c25-logic("1234567890")
```

Result:

Industrial
Example:

```
#c25-ind("1234567890")
```

Result:

UPC (Universal Product Code)

UPC-A

Example:

```
#upca("01234500006")
```

Result:

UPC-A w/ Check Digit

Example:

```
#upca-chk("012345000065")
```

Result:

UPC-E

Example:

```
#upce("123456")
```

Result:

UPC-E w/ Check Digit

Example:

```
#upce-chk("12345670")
```

Result:

HIBC (Health Industry Barcodes)

Code 128
Example:

```
#hibc-128("1234567890")
```

Result:

Code 39
Example:

```
#hibc-39("1234567890")
```

Result:

Data Matrix
Example:

```
#hibc-dm("1234567890")
```

Result:

QR
Example:

```
#hibc-qr("1234567890")
```

Result:

PDF417
Example:

```
#hibc-pdf("1234567890")
```

Result:

Micro PDF417
Example:

```
#hibc-mic-pdf("1234567890")
```

Result:

Codablock-F
Example:

```
#hibc-codablock-f("1234567890")
```

Result:

Aztec
Example:

```
#hibc-aztec("1234567890")
```

Result:

Postal

Australia Post Redirection

Example:

```
#aus-redirect("12345678")
```

Result:

Australia Post Reply Paid

Example:

```
#aus-reply("12345678")
```

Result:

Australia Post Routing

Example:

```
#aus-route("12345678")
```

Result:

Australia Post Standard Customer

Example:

```
#aus-post("12345678")
```

Result:

Brazilian CEPNet Postal Code

Example:

```
#cepnet("1234567890")
```

Result:

DAFT Code

Example:

```
#daft("DAFTFDATATFDTFAD")
```

Result:

Deutsche Post Identcode

Example:

```
#dp-ident("1234567890")
```

Result:

Deutsche Post Leitcode

Example:

```
#dp-leitcode("1234567890")
```

Result:

Deutsher Paket Dienst

Example:

```
#dpd("01234567890123456789012345 67")
```

Result:

Dutch Post KIX Code

Example:

```
#kix("1234567890")
```

Result:

Japanese Postal Code
Example:

```
#japan-post("1234567890")
```

Result:

Korea Post
Example:

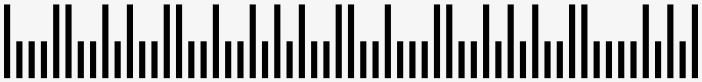
```
#korea-post("123456")
```

Result:

POSTNET
Example:

```
#postnet("1234567890")
```

Result:



Royal Mail 2D Mailmark (CMDM)
Example:

```
#mailmark-2d(
  32, 32,
  "JGB 011123456712345678CW14NJ1T 0EC2M2QS
REFERENCE1234567890QWERTYUIOPASDFGHJKLZXCVBNM"
)
```

Result:

Royal Mail 4-State Customer Code
Example:

```
#rm4scc("1234567890")
```

Result:

Royal Mail 4-State Mailmark
Example:

```
#mailmark-4s("21B2254800659JW5O9QA6Y")
```

Result:

Universal Postal Union S10
Example:

```
#upus10("RR072705659PL")
```

Result:

UPNQR (Univerzalnega Plačilnega Naloga QR)
Example:

```
#upnqr("1234567890")
```

Result:

USPS Intelligent Mail
Example:

```
#usps-imail("01300123456123456789")
```

Result:

Other Generic Codes

Aztec Code
Example:

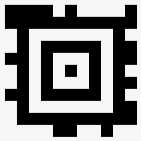
```
#aztec("1234567890")
```

Result:

Aztec Rune
Example:

```
#azrune("122")
```

Result:



Channel Code
Example:

```
#channel("123456")
```

Result:

Codabar
Example:

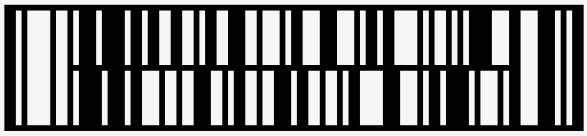
```
#codabar("A123456789B")
```

Result:

Codablock-F
Example:

```
#codablock-f("1234567890")
```

Result:



Code 11
Example:

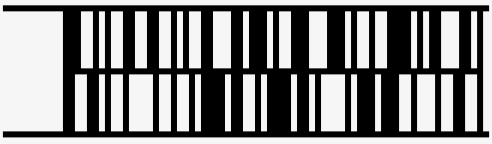
```
#code11("0123452")
```

Result:

Code 16k
Example:

```
#code16k("1234567890")
```

Result:



Code 32
Example:

```
#code32("12345678")
```

Result:

Code 39
Example:

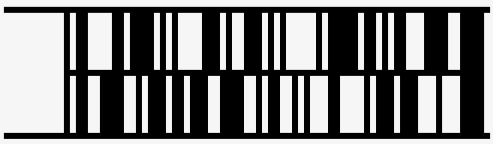
```
#code39("1234567890")
```

Result:

Code 49
Example:

```
#code49("1234567890")
```

Result:



Code 128
Example:

```
#code128("1234567890")
```

Result:

Code 128 (AB)
Example:

```
#code128ab("1234567890")
```

Result:

Code One
Example:

```
#code-one("1234567890")
```

Result:

Data Matrix (ECC200)
Example:

```
#data-matrix("1234567890")
```

Result:

DotCode
Example:

```
#dotcode("1234567890")
```

Result:

Extended Code 39
Example:

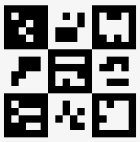
```
#ex-code39("1234567890")
```

Result:

Grid Matrix
Example:

```
#grid-matrix("1234567890")
```

Result:



Han Xin (Chinese Sensible)
Example:

```
#hanxin("abc123 全 ń 全濯")
```

Result:

IBM BC412 (SEMI T1-95)
Example:

```
#bc412("1234567890")
```

Result:

ISBN
Example:

```
#isbnx("9789861817286")
```

Result:

ITF-14

Example:

```
#itf14("1234567890")
```

Result:

LOGMARS

Example:

```
#logmars("1234567890")
```

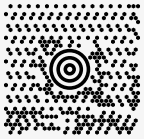
Result:

MaxiCode

Example:

```
#maxicode("1234567890")
```

Result:



Micro QR

Example:

```
#micro-qr("1234567890")
```

Result:

MSI Plessey

Example:

```
#msi-plessey("1234567890")
```

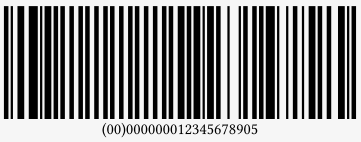
Result:

NVE-18 (SSCC-18)

Example:

```
#nve18("1234567890")
```

Result:

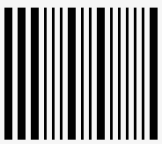


Pharmacode One-Track

Example:

```
#pharma("123456")
```

Result:



Pharmacode Two-Track

Example:

```
#pharma-two("12345678")
```

Result:

Pharmazentralnummer

Example:

```
#pzn("12345678")
```

Result:

Planet

Example:

```
#planet("1234567890")
```

Result:

Plessey
Example:

```
#plessey("1234567890")
```

Result:

QR Code
Example:

```
#qrcode("1234567890")
```

Result:

Rectangular Micro QR Code (rMQR)
Example:

```
#rmqr("1234567890")
```

Result:

Telepen Numeric
Example:

```
#telepen-num("1234567890")
```

Result:

Telepen
Example:

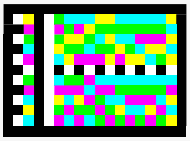
```
#telepen("ABCD12345")
```

Result:

Ultracode
Example:

```
#ultra("1234567890")
```

Result:



Vehicle Identification Number
Example:

```
#vin("2GNFLGE30D6201432")
```

Result:

Facing Identification Mark
Example:

```
#fim("A")
```

Result:

Flattermarken
Used for marking book covers to indicate volume order.

Example:

```
#flat("123")
```

Result:

Symbology Values

Following symbology values are supported:

"Code11"	"C25Standard"	"C25Inter"	"C25IATA"	"C25Logic"	"C25Ind"
"Code39"	"ExCode39"	"EANX"	"EANXChk"	"GS1128"	"Codabar"
"Code128"	"DPLEIT"	"DPIDENT"	"Code16k"	"Code49"	"Code93"
"Flat"	"DBarOmn"	"DBarLtd"	"DBarExp"	"Telepen"	"UPCA"
"UPCAChk"	"UPCE"	"UPCEChk"	"Postnet"	"MSIPlessey"	"FIM"
"Logmars"	"Pharma"	"PZN"	"PharmaTwo"	"CEPNet"	"PDF417"
"PDF417Comp"	"MaxiCode"	"QRCode"	"Code128AB"	"AusPost"	"AusReply"
"AusRoute"	"AusRedirect"	"ISBNX"	"RM4SCC"	"DataMatrix"	"EAN14"
"VIN"	"CodablockF"	"NVE18"	"JapanPost"	"KoreaPost"	"DBarStk"
"DBarOmnStk"	"DBarExpStk"	"Planet"	"MicroPDF417"	"USPSIMail"	"Plessey"
"TelepenNum"	"ITF14"	"KIX"	"Aztec"	"DAFT"	"DPD"
"MicroQR"	"HIBC128"	"HIBC39"	"HIBCDM"	"HIBCQR"	"HIBCPDF"
"HIBCMicPDF"	"HIBCCodablockF"	"HIBCAztec"	"DotCode"	"HanXin"	"Mailmark2D"
"UPUS10"	"Mailmark4S"	"AzRune"	"Code32"	"EANXCC"	"GS1128CC"
"DBarOmnCC"	"DBarLtdCC"	"DBarExpCC"	"UPCACC"	"UPCECC"	"DBarStkCC"
"DBarOmnStkCC"	"DBarExpStkCC"	"Channel"	"CodeOne"	"GridMatrix"	"UPNQR"
"Ultra"	"RMQR"	"BC412"			