

Contents

1 Introduction	2
2 Usage	2
3 Config presets	2
4 Reference	3
4.1 config	3
4.1.1 config	3
4.1.2 dark	6
4.1.3 blueprint	6
4.2 schema	7
4.2.1 load	7
4.2.2 render	7

1 Introduction

This package provides a way to make beautiful register diagrams using the CeTZ package. It can be used to document Assembly instructions or binary registers

This is a port of the [homonymous Python script](#) for Typst. For more information on the schema format, please check out the original project's [format.md](#)

2 Usage

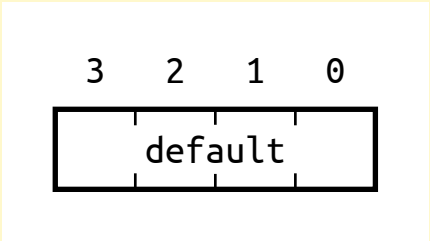
Simply import schema from `src/lib.typ` and call `schema.load` to parse a schema description. Then use `schema.render` to render it, et voilà !

```
#import "src/lib.typ": schema
#let doc = schema.load("path/to/schema.typ")
#schema.render(doc)
```

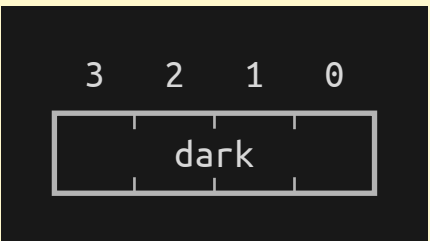
3 Config presets

Aside from the default config, some example presets are also provided:

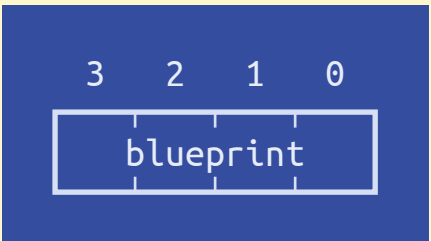
- `config.config()`: the default theme, black on white

	<pre>let ex = schema.load(`yaml structures: main: bits: 4 ranges: 3-0: name: default `) schema.render(ex, config: config.config())</pre>
--	--

- `config.dark()`: a dark theme, with white text and lines on a black background

	<pre>let ex = schema.load(`yaml structures: main: bits: 4 ranges: 3-0: name: dark `) schema.render(ex, config: config.dark())</pre>
---	---

- `config.blueprint()`: a blueprint theme, with white text and lines on a blue background

	<pre>let ex = schema.load(`yaml structures: main: bits: 4 ranges: 3-0: name: blueprint `) schema.render(ex, config: config.blueprint())</pre>
---	---

4 Reference

4.1 config

- [config\(\)](#)
- [dark\(\)](#)
- [blueprint\(\)](#)

4.1.1 config

Creates a dictionary of all configuration parameters

Parameters

```
config(
  default-font-family: str,
  default-font-size,
  italic-font-family: str,
  italic-font-size: length,
  background: color,
  text-color: color,
  link-color: color,
  bit-i-color: color,
  border-color: color,
  bit-width: float,
  bit-height: float,
  description-margin: float,
  dash-length: float,
  dash-space: float,
  arrow-size: float,
  margins,
  arrow-margin: float,
  values-gap: float,
  arrow-label-distance: float,
  force-descs-on-side: bool,
  left-labels: bool,
  width: float,
  height: float,
  full-page: bool
) -> dictionary
```

default-font-family str

The default font family

- default font-size (length): The absolute default font size

Default: "Ubuntu Mono"

italic-font-family str

The italic font family (for value descriptions)

Default: "Ubuntu Mono"

italic-font-size length

The absolute italic font size

Default: 12pt

background color

The diagram background color

Default: white

text-color color

The default color used to display text

Default: black

link-color color

The color used to display links and arrows

Default: black

bit-i-color color

The color used to display bit indices

Default: black

border-color color

The color used to display borders

Default: black

bit-width float

The width of a bit

Default: 30

bit-height float

The height of a bit

Default: 30

description-margin float

The margin between descriptions

Default: 10

dash-length float

The length of individual dashes (for dashed lines)

Default: 6

dash-space float

The space between two dashes (for dashed lines)

Default: 4

arrow-size float

The size of arrow heads

- margins (tuple[float]): TODO -> remove

Default: 10

arrow-margin float

The margin between arrows and the structures they link

Default: 4

values-gap float

The gap between individual values

Default: 5

arrow-label-distance float

The distance between arrows and their labels

Default: 5

force-descs-on-side bool

If true, descriptions are placed on the side of the structure, otherwise, they are placed as close as possible to the bit

Default: false

left-labels `bool`

If true, descriptions are put on the left, otherwise, they default to the right hand side

Default: `false`**width** `float`

TODO -> remove

Default: `1200`**height** `float`

TODO -> remove

Default: `800`**full-page** `bool`

If true, the page will be resized to fit the diagram and take the background color

Default: `false`

4.1.2 dark

Dark theme config

Parameters

`dark(..args: any)`**..args** `any`see [config\(\)](#)

4.1.3 blueprint

Blueprint theme config

Parameters

`blueprint(..args: any)`**..args** `any`see [config\(\)](#)

4.2 schema

- [load\(\)](#)
- [render\(\)](#)

4.2.1 load

Loads a schema from a file or a raw block. This function returns a dictionary of structures

Supported formats: `.yaml`, `.json`, `.xml`

Parameters

```
load(path-or-schema: str raw) -> dictionary
```

path-or-schema `str` or `raw`

If it is a string, defines the path to load.

If it is a raw block, its content is directly parsed (the block's language will define the format to use)

4.2.2 render

Renders the given schema This functions

Parameters

```
render(
  structures: dictionary,
  config: auto dictionary
)
```

structures `dictionary`

A schema dictionary, as returned by [load\(\)](#)

config `auto` or `dictionary`

The configuration parameters, as returned by [config\(\)](#)

Default: `auto`