# Typesetting Ladder Diagrams with LaTeX and Ti*k*Z

Luis Paulo Laus
e-mail: laus@utfpr.edu.br

Version 1 of 10 January 2018

## 1 Abstract

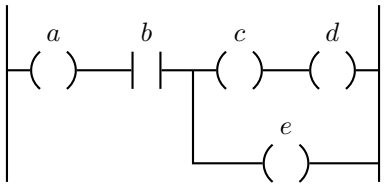
Ladder diagram (LD) is a graphical programming language that has evolved from electrical wiring diagrams for relay control systems used with programmable controllers (PLC[1]) as described in the international standard IEC-61 131-3. A LD program enables the programmable controller to test and modify data by means of standardized graphic symbols. These symbols are laid out in networks in a manner similar to a "rung" of a relay ladder logic diagram. This library provides Ti*k*Z symbols to draw high quality ladder diagrams. All standard and some non-standard symbols are possible, including all kids of contacts, coils and blocks. I decided to write this package, despite of the fact that there is available another package named `ladder` that also uses Ti*k*Z to typeset ladder diagrams, because that package seems[2] to lack support for blocks. The `tikz-ladder`, on the contrary, supports all features described in IEC-61 131-3, namely, blocks (for functions and function blocks), contacts and coils.

## 2 Ladder Diagram

According to IEC-61 131-3, the usage of letters, semigraphic or graphic for the representation of graphical elements is implementer specific and not a normative requirement. This poses a problem for creating a package for typesetting ladder diagrams in agreement to a standard that should be used by everyone: you can do whatever you want. Thus, this package provides Ti*k*Z symbols for typesetting ladder diagram as close as possible to the standard, but not to close since a program in the standard would look like:

```
|   a    b       c     d   |
+--( )--| |--+--( )---( )--+
|           |      e       |
|           +-----( )-----+
|                          |
```

and it is probably not what you want. With this package, you can produce something like[3]:




## 3 Ladder Diagram Library

**Ti*k*Z Library** ladder

```
\usepgflibrary{ladder}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[ladder]  % ConTeXt and pure pgf
\usetikzlibrary{ladder} % LaTeX and plain TeX when using TikZ
```

[1] Formerly known as programable logic controllers.

[2] Sorry, but the documentation is in French and I limited myself to look at the figures.

[3] This slightly awkward example was extracted from [IEC-61 131-3/2013, p. 218]; and explained by: "In the rung shown above, the value of the Boolean output *a* is always `TRUE`, while the value of outputs *c*, *d* and *e* upon completion of an evaluation of the rung is equal to the value of the input *b*.". In 2013 version there is typo: the *a* is missing. The 2003 version is correct.

```
\usetikzlibrary[ladder] % ConTEXt when using TikZ
```

This library provides graphics for ladder diagram related to programable controllers (PLC) and according to the international standard IEC-61 131-3. The library was written to extend the standard TikZ circuit library. The reader is urged to read the Section "Circuit Libraries" of TikZ manual. This library defines the following key:

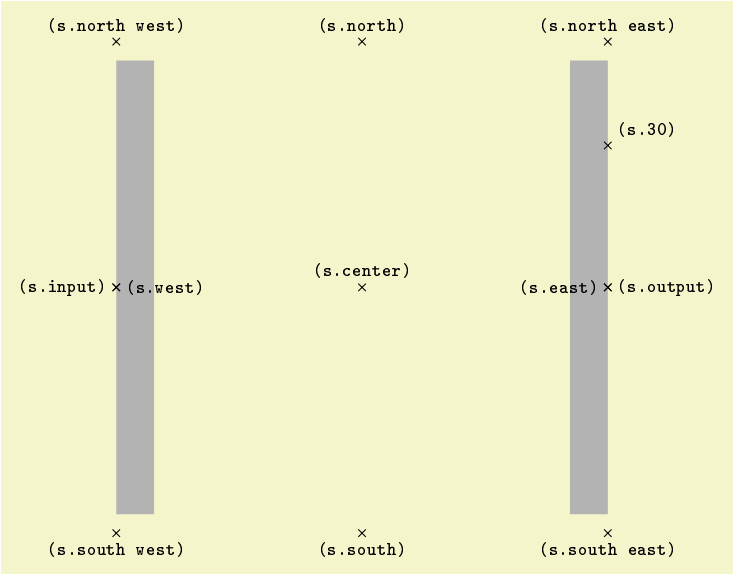
/tikz/ladder (no value)

This style calls `ladder` and installs ladder diagram graphics for symbols like contacts, coils and blocks.

In the next sections a description of the library features is provided.

# 4 Contacts

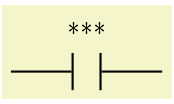
A contact is an element which imparts a state to the horizontal link on its right side which is equal to the Boolean AND of the state of the horizontal link at its left side with an appropriate function of an associated Boolean input, output, or memory variable. A contact does not modify the value of the associated Boolean variable.




```
\begin{tikzpicture}[ladder]
  \node[name=s, shape=contact ladder, shape example, inner xsep=1cm, inner ysep=1cm, minimum
width=6cm, minimum height=6cm]{};
\foreach \anchor/\placement in {center/above, 30/above right, north/above, south/below, east/left,
 west/right, north east/above, south east/below, south west/below, north west/above,input/left,output/right}
  \draw[shift=(s.|anchor)] plot[mark=x] coordinates{(0,0)} node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

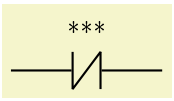
All kinds of standardized contacts can be represented: normally open contact (NO); normally closed contact (NC); positive transition-sensing contact (P); negative transition-sensing contact (N); compare contacts both typed and overloaded (for typesetting purposes it makes no difference.) In the following examples, the Boolean variable associated with the contact is indicated by "***".

Normally open contact (NO):



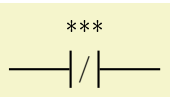

```
\tikz[ladder,thick] \draw(0,0) to [contact NO={info={***}}] ++(2,0);
```

Normally closed contact (NC):



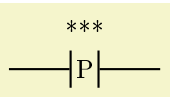

```
\tikz[ladder,thick] \draw(0,0) to [contact NC={info={***}}] ++(2,0);
```

Variation of normally closed contact (NC):



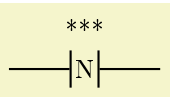

```
\tikz[ladder,thick] \draw(0,0) to [var contact NC={info={***}}] ++(2,0);
```

Positive transition-sensing contact (P):



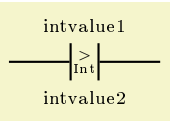

```
\tikz[ladder,thick] \draw(0,0) to [contact P={info={***}}] ++(2,0);
```

Negative transition-sensing contact (N):



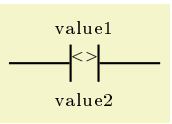

```
\tikz[ladder,thick] \draw(0,0) to [contact N={info={***}}] ++(2,0);
```

Compare contact (typed):




```
\tikz[ladder,thick] \draw(0,0) to [contact NO={
  info={|scriptsize intvalue1},
  info'={|scriptsize intvalue2},
  symbol={|tiny$|genfrac{}{}{0pt}{}{>}{|text{Int}}$}}] ++(2,0);
```

Compare contact (overloaded):




```
\tikz[ladder,thick] \draw(0,0) to [contact NO={
  info={|scriptsize value1},
  info'={|scriptsize value2},
  symbol={|tiny$|genfrac{}{}{0pt}{}{<>}{}$}}] ++(2,0);
```

There are two possibilities for normally closed contact. It is not recommended to mix them in the same document unless to explain their equivalence.

### 4.1 Keys for contacts

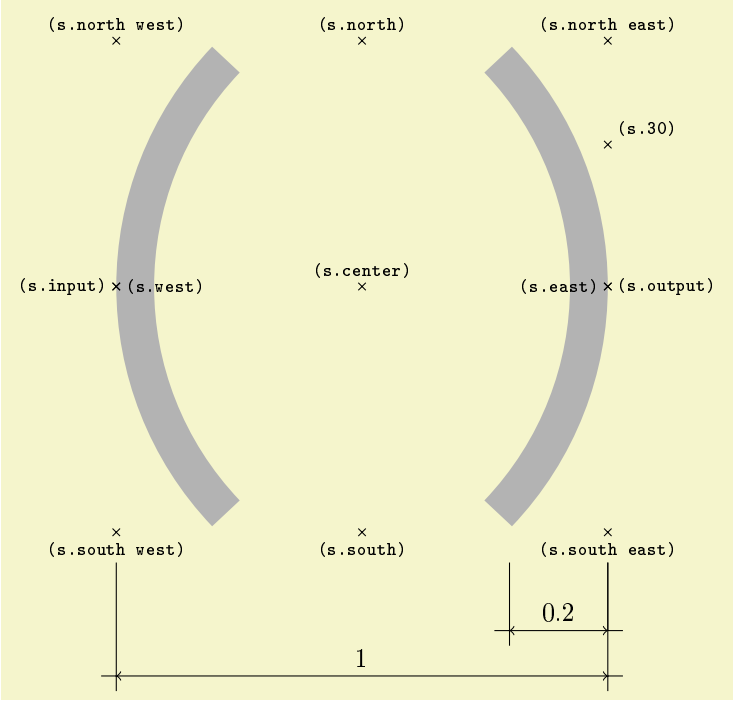
The most common key used with contacts is `info` which sets the variable name associated with the contacts. One may also need `info'` and `name`; all standard keys. In addition to the keys described in Section "Circuit Libraries" of "The TikZ and PGF Packages – Manual for version 3.0.1a", contacts accept:

`/tikz/symbol`=⟨*name*⟩ (no default)

This key sets the information, usually a single letter or comparison symbol, that will appear between the vertical lines. Usable for drawing compare contacts.

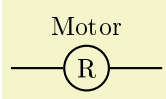
## 5 Coils

A coil copies the state of the link on its left to the link on its right without modification, and stores an appropriate function of the state or transition of the left link into the associated Boolean variable.




```
\begin{tikzpicture}[ladder]
  \node[name=s, shape=coil ladder, shape example, fill=none, inner xsep=1cm, inner ysep=1cm, minimum
width=6cm, minimum height=6cm]{};
\foreach \anchor/\placement in {center/above, 30/above right, north/above, south/below, east/left,
 west/right, north east/above, south east/below, south west/below, north west/above,input/left,output/right}
  \draw[shift=(s.|anchor)] plot[mark=x] coordinates{(0,0)} node[\placement] {\scriptsize\texttt{(s.\anchor)}};
  \draw[thin]([yshift=-4mm]s.south east) -- ++(0,-0.9) coordinate(x1)  -- ++(0.2,0)
     ([yshift=-4mm]$0.2*(s.south west) + 0.8*(s.south east)$) -- ++(0,-0.9) coordinate(x2) -- ++(-0.2,0)
     (x2) -- ++(0,-0.2);
  \draw[thin,<->] (x1) -- (x2) node[midway,above]{0.2};
  \draw[thin]([yshift=-4mm]s.south east) -- ++(0,-1.5) coordinate(x1)  -- ++(0.2,0)
     (x1) -- ++(0,-0.2) ([yshift=-4mm]s.south west)  -- ++(0,-1.5) coordinate(x2) -- ++(-0.2,0)
     (x2) -- ++(0,-0.2);
  \draw[thin,<->] (x1) -- (x2) node[midway,above]{1};
\end{tikzpicture}
```

The `coil ladder curvature` controls how round the coils look like. The default value is 0.2 as indicated above; 0.5 makes a round coil like[4]:




```
\tikz[ladder,thick]
  \draw(0,0) to [coil R={info={Motor},coil ladder curvature=0.5,
     minimum size=2.4|tikzcircuitssizeunit}] ++(2,0);
```

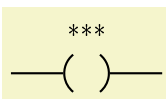
Note that it is also necessary to set the `minimum size` because coils have a proportion of $2.4 \times 2$. If you do this frequently, you can set the style for the coils like:

```
every coil R/.style={coil ladder curvature=0.5,minimum size=2.4\tikzcircuitssizeunit}
```

Values above 0.5, although possible, lead to estrange figures.

All kind of standardized coils are supported: coil (normal); negated coil (normally activated, NA); set (latch) coil; reset (unlatch) coil; positive transition-sensing coil; and negative transition-sensing coil. In the following examples, the Boolean variable associated with the coil is indicated by "***".

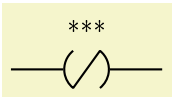
Coil (normally deactivated):




```
\tikz[ladder,thick] \draw(0,0) to [coil={info={***}}] ++(2,0);
```

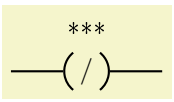
Negated coil (normally activated, NA):

[4] It can be used to draw relay coils according to NEMA – National Electrical Manufacturers Association.



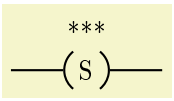

```
\tikz[ladder,thick] \draw(0,0) to [coil NA={info={***}}] ++(2,0);
```

Variation of negated coil (normally activated, NA):



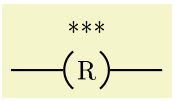

```
\tikz[ladder,thick] \draw(0,0) to [var coil NA={info={***}}] ++(2,0);
```

Set (latch) coil:



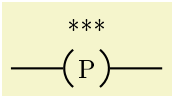

```
\tikz[ladder,thick] \draw(0,0) to [coil S={info={***}}] ++(2,0);
```

Reset (unlatch) coil:



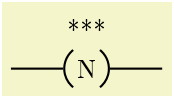

```
\tikz[ladder,thick] \draw(0,0) to [coil R={info={***}}] ++(2,0);
```

Positive transition-sensing coil:




```
\tikz[ladder,thick] \draw(0,0) to [coil={info={***},symbol=P}] ++(2,0);
```

Negative transition-sensing coil:

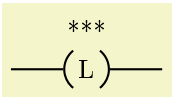



```
\tikz[ladder,thick] \draw(0,0) to [coil={info={***},symbol=N}] ++(2,0);
```

Note that positive and negative transition-sensing coils are not supported directly because, to be honest, no one uses them. Their symbols have to be coined using a normal coil and the parameter `symbol`.

There are two possibilities for negated coil (NA). It is not recommended to mix them in the same document unless to explain their equivalence.

It is possible, though not recommend because it disagrees with IEC-61 131-3, to used non-standard coils, e.g., some people uses L (for latch) and U (for unlatch) instead of, S and U, respectively. This is achieved by changing the `symbol` of a `coil` like:




```
\tikz[ladder,thick] \draw(0,0) to [coil={info={***},symbol=L}] ++(2,0);
```

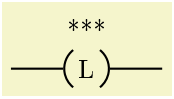
The same trick used for positive and negative transition-sensing coils.

## 5.1 Keys for coils

The most common key used with coils is `info` which sets the variable name associated with the coil. One may also need `info'`, `name` and `minimum size`; all standard keys. In addition to the keys described in Section "Circuit Libraries" of "The TikZ and PGF Packages – Manual for version 3.0.1a", coils accept:

`/tikz/symbol`=⟨*name*⟩ (no default)

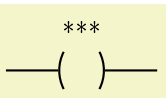
This key sets the information, usually a single letter, that will appear between the parenthesis. Usable for non-standard or rarely used coils lake positive and negative transition-sensing coils.



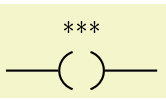

```
\tikz[ladder,thick] \draw(0,0) to [coil={info={***},symbol=L}] ++(2,0);
```

`/tikz/coil ladder curvature=`⟨*curvature index*⟩ (no default, initially `0.5`)

This key sets the curvature index, a number between 0.001 and 0.5 (in practice, though higher values are permitted) that defined how much the parentheses will be bent. It is the fraction of the total coil width occupied by one parenthesis. Usable for drawing electric coils in NEMA standard. In this case, `minimum size` will have to adjusted in order to correct the coil aspect ratio.




```
\tikz[ladder,thick] \draw(0,0) to [coil={info={***},
  coil ladder curvature=0.1}] ++(2,0);
```

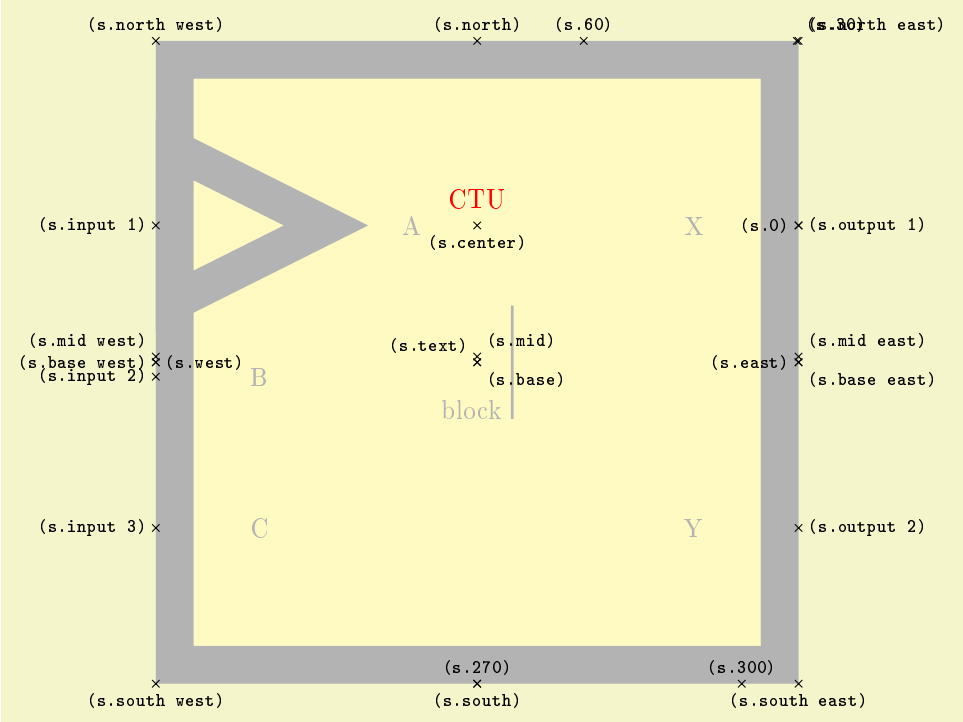



```
\tikz[ladder,thick] \draw(0,0) to [coil={info={***},
  coil ladder curvature=0.3}] ++(2,0);
```

# 6 Blocks

Blocks are used to represent all other features besides contacts and coils, namely functions and function blocks, manly: timers, counters, communication, string treatment, arithmetic and logical operations. Blocks can have many inputs and outputs. The first input and output shall be aligned with the rung line. This is done automatically and explains why the `center` anchor is not on the centre of the rectangle. For the exact rectangle centre use the `text` or `base` anchor[5].

Next, a block for a counter is represented with the anchors, you can see the input `>A` causes a clock input indication and the symbol `>` itself is gobbled. The input and output names are not accordingly to the standard IEC-61 131-3.




```
\begin{tikzpicture}[ladder]
  \node[name=s, shape=block ladder, shape example, minimum width=8cm, minimum height=8cm, inner xsep=1cm,
   inner ysep=1cm, input sep=2cm, output sep=4cm, inputs={>A,B,C}, clksize=2cm,  outputs={X,Y}, symbol
   color=black!30, symbol={|\textcolor{red}{CTU}|}] {block \vrule width1pt height1.5cm};
  \foreach \anchor/\placement in {center/below, text/above left, 0/left, 30/above right, 60/above,
   270/above, 300/above, mid/above right, mid east/above right, mid west/above left, base/below right,
   base east/below right, base west/left, north/above, south/below, east/left, west/right,
   north east/above right, south east/below, south west/below, north west/above, input 1/left,
   input 2/left, input 3/left, output 1/right, output 2/right}
  \draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)} node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

[5] I am not happy with this and it might be change in the future.

In the following subsections the standard names are employed accordingly to IEC-61 131-3.

## 6.1 Keys for blocks

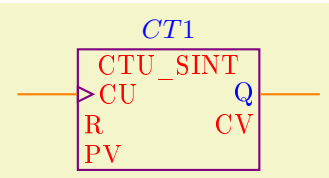
The three most common keys used with blocks are `symbol` which sets the block type, `name` which sets a Ti*k*Z label to be used in future references, particularly to access the inputs and outputs, and `info` which sets the variable name associated with the POU represented by the block. One may also need `info'` and `minimum width`; all standard keys. In addition to the keys described in Section "Circuit Libraries" of "The TikZ and PGF Packages – Manual for version 3.0.1a", blocks accept:

`/tikz/symbol`=⟨*name*⟩ (no default)

This key sets the information that appears inside the block rectangle, on the top. It specifies the POU type represented by the block.

`/tikz/symbol color`=⟨*colour*⟩ (no default)

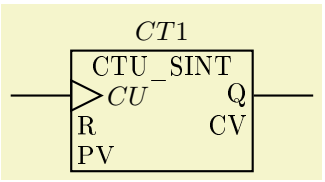
This key sets the colour used for all texts inside the block: symbol, inputs and outputs. In a `beamer` presentation it can be override for a particular input/output by forcing the text colour like this: `{\textcolor{blue}{Q}}`. For other document classes, you need to use a box:




```
\newsavebox{\myeqbox} % only once, preferable in the document preamble
\savebox{\myeqbox}{\textcolor{blue}{Q}}
\tikz[ladder,thick] \draw[orange](0,0) to [block={violet,
   info={[blue]$CT1$},inputs={>CU,R,PV}, outputs={\usebox{\myeqbox},CV},
   symbol=CTU\_SINT, symbol color=red, minimum width=2.4cm}] ++(4,0);
```

`/tikz/clksize`=⟨*width*⟩ (no default, initially `0.8` `tikzcircuitssizeunit`)

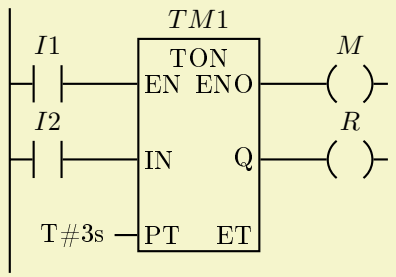
This key sets the size for the clock input indicator (`>`).




```
\tikz[ladder,thick] \draw(0,0) to [block={
   info=$CT1$,inputs={>$CU$,R,PV}, outputs={Q,CV},
   symbol=CTU\_SINT, clksize=0.4cm,
   minimum width=2.4cm}]  ++(4,0);
```

`/tikz/input sep`=⟨*width*⟩ (no default, initially `1.6` `tikzcircuitssizeunit`)

This key sets the vertical distance between two consecutive inputs.




```
\begin{tikzpicture}[ladder,thick]
  \draw(0,0) to [contact NO={info={$I1$}}] ++(1,0)
     to [block={info=$TM1$,symbol=TON,
       inputs={EN,IN,PT},outputs={ENO,Q,ET},
       name=TM1,minimum width=1.6cm,
       input sep=1cm,output sep=1cm}] ++(3,0)
    to [coil={info={$M$}}] ++(1,0);
  \draw(0,-1) to [contact NO={info={$I2$}}] ++(1,0) -- (TM1.input 2)
    (TM1.output 2) -- (4,-1) to [coil={info={$R$}}] ++(1,0)
    (TM1.input 3) -- +(-3mm,0)node[left]{T\#3s} (0,1) -- +(0,-3.5);
\end{tikzpicture}
```

`/tikz/output sep`=⟨*width*⟩ (no default, initially `1.6` `tikzcircuitssizeunit`)

This key sets the vertical distance between two consecutive outputs (see example above).

`/tikz/input`=⟨*inputs*⟩ (no default, initially `IN`)

This key sets the input names that appear inside the block. It is a comma separated list of inputs. Clock inputs are indicated by the first character being `>`. Coordinates for future external connection

are automatically generated in the form *name*.`input` *n*, where *n* is the input number starting in 1. An empty input can be generated by `{~}`. The minimal number of inputs is one, an empty list of inputs generates an error.

`/tikz/output`=⟨*outputs*⟩ (no default, initially `Q`)

This key sets the output names that appear inside the block. It is a comma separated list of outputs. Coordinates for future external connection are automatically generated in the form *name*.`output` *n*, where *n* is the output number starting in 1. An empty output can be generated by `{}`. The minimal number of outputs is one, an empty list of outputs generates an error.

## 6.2 Timers

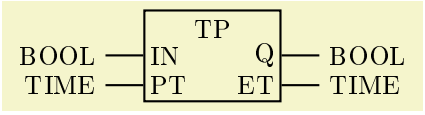
The standard IEC-61 131-3 specifies three timers as follows:
On-delay:


```
\node[block,inputs={IN,PT},outputs={Q,ET},
   symbol=TON,minimum width=18mm] (tp1) {};
\draw (tp1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (tp1.input 2) -- +(-5mm,0) node[left]{TIME}
      (tp1.output 1) -- +(5mm,0) node[right]{BOOL}
      (tp1.output 2) -- +(5mm,0) node[right]{TIME};
```

Off-delay:


```
\node[block,inputs={IN,PT},outputs={Q,ET},
   symbol=TOF,minimum width=18mm] (tp1) {};
\draw (tp1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (tp1.input 2) -- +(-5mm,0) node[left]{TIME}
      (tp1.output 1) -- +(5mm,0) node[right]{BOOL}
      (tp1.output 2) -- +(5mm,0) node[right]{TIME};
```

Pulse:

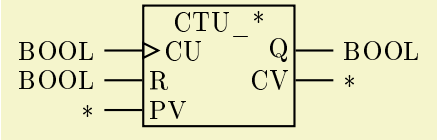



```
\node[block,inputs={IN,PT},outputs={Q,ET},
   symbol=TP,minimum width=18mm] (tp1) {};
\draw (tp1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (tp1.input 2) -- +(-5mm,0) node[left]{TIME}
      (tp1.output 1) -- +(5mm,0) node[right]{BOOL}
      (tp1.output 2) -- +(5mm,0) node[right]{TIME};
```

## 6.3 Counters

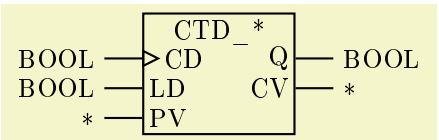
The clock input is indicated by the character > which need to be the very first one in the input description of a clock input. You can use any number of clock inputs and they can appear in any order. For instance, the inputs of an up-down counter with enable input shall be declare as `inputs={EN,>CU,>CD,R,LD,PV}`.

The standard IEC-61 131-3 specifies three counters. In the following examples, the symbol "*" indicates the numerical type of the counter (like `INT`, `DINT`, etc.).
Up-Counter:



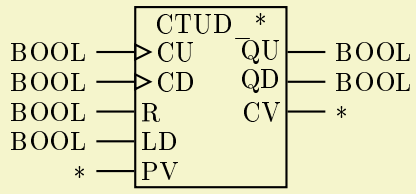

```
\node[block,inputs={>CU,R,PV},outputs={Q,CV},
   symbol=CTU\_*,minimum width=20mm] (ct1) {};
\draw (ct1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (ct1.input 2) -- +(-5mm,0) node[left]{BOOL}
      (ct1.input 3) -- +(-5mm,0) node[left,yshift=-0.4ex]{*}
      (ct1.output 1) -- +(5mm,0) node[right]{BOOL}
      (ct1.output 2) -- +(5mm,0) node[right,yshift=-0.4ex]{*};
```

Down-counters:




```
\node[block,inputs={>CD,LD,PV},outputs={Q,CV},
   symbol=CTD\_*,minimum width=20mm] (ct1) {};
\draw (ct1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (ct1.input 2) -- +(-5mm,0) node[left]{BOOL}
      (ct1.input 3) -- +(-5mm,0) node[left,yshift=-0.4ex]{*}
      (ct1.output 1) -- +(5mm,0) node[right]{BOOL}
      (ct1.output 2) -- +(5mm,0) node[right,yshift=-0.4ex]{*};
```

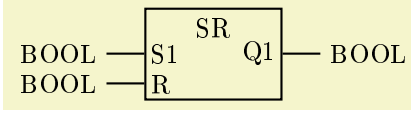
Up-down counters:




```
\node[block,inputs={>CU,>CD,R,LD,PV},outputs={QU,QD,CV},
     symbol=CTUD|_*,minimum width=20mm] (ct1) {};
\draw (ct1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (ct1.input 2) -- +(-5mm,0) node[left]{BOOL}
      (ct1.input 3) -- +(-5mm,0) node[left]{BOOL}
      (ct1.input 4) -- +(-5mm,0) node[left]{BOOL}
      (ct1.input 5) -- +(-5mm,0) node[left,yshift=-0.4ex]{*}
      (ct1.output 1) -- +(5mm,0) node[right]{BOOL}
      (ct1.output 2) -- +(5mm,0) node[right]{BOOL}
      (ct1.output 3) -- +(5mm,0) node[right,yshift=-0.4ex]{*};
```

## 6.4 Standard bistable function blocks

Bistable function block (set dominant): `RS(S1,R, Q1)`

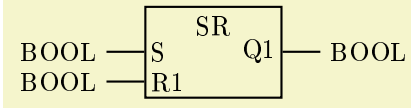



```
\node[block,inputs={S1,R},outputs={Q1},
   symbol=SR,minimum width=18mm] (sr1) {};
\draw (sr1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (sr1.input 2) -- +(-5mm,0) node[left]{BOOL}
      (sr1.output 1) -- +(5mm,0) node[right]{BOOL};
```

Bistable function block (set dominant) with long input names: `RS(SET1,RESET, Q1)`

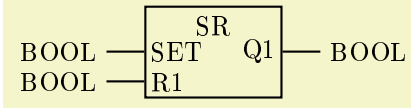

```
\node[block,inputs={SET1,RESET},outputs={Q1},
   symbol=SR,minimum width=18mm] (sr1) {};
\draw (sr1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (sr1.input 2) -- +(-5mm,0) node[left]{BOOL}
      (sr1.output 1) -- +(5mm,0) node[right]{BOOL};
```

Bistable function block (reset dominant): `RS(S,R1, Q1)`




```
\node[block,inputs={S,R1},outputs={Q1},
   symbol=SR,minimum width=18mm] (sr1) {};
\draw (sr1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (sr1.input 2) -- +(-5mm,0) node[left]{BOOL}
      (sr1.output 1) -- +(5mm,0) node[right]{BOOL};
```

Bistable function block (reset dominant) with long input names[6]: `RS(SET,RESET1, Q1)`




```
\node[block,inputs={SET,R1},outputs={Q1},
   symbol=SR,minimum width=18mm] (sr1) {};
\draw (sr1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (sr1.input 2) -- +(-5mm,0) node[left]{BOOL}
      (sr1.output 1) -- +(5mm,0) node[right]{BOOL};
```

## 6.5 Standard edge detection function blocks

Rising edge detector: `R_TRIG(CLK, Q)`


```
\node[block,inputs={CLK},outputs={Q},
   symbol=R|_TRIG,minimum width=20mm] (ed1) {};
\draw (ed1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (ed1.output 1) -- +(5mm,0) node[right]{BOOL};
```

Falling edge detector: `F_TRIG(CLK, Q)`

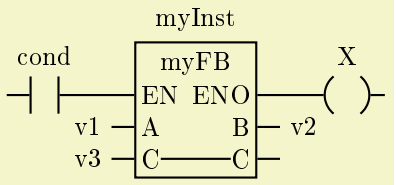

```
\node[block,inputs={CLK},outputs={Q},
   symbol=F|_TRIG,minimum width=20mm] (ed1) {};
\draw (ed1.input 1) -- +(-5mm,0) node[left]{BOOL}
      (ed1.output 1) -- +(5mm,0) node[right]{BOOL};
```

[6] Here we have a clear inconsistency between the text description that presents the input `RESET1` and graphical representation where it is `R1`. It is probably just a typo, but the form presented in [IEC-61 131-3/2013, p. 112] was kept unchanged.

### 6.6 Call representation

A call is used to execute a function, a function block instance, or a method of a function block or class. They are represented by blocks.

The proper map for in-out variables (`VAR_IN_OUT`) need special care. This map is represented by a line internal to the block connecting the left and right side of the in-out variable. The problem is that `tikz-ladder` only creates anchors at the left side of the inputs and right side of outputs. We need the opposite. This situation can be overcome in either two ways: using `calc` library to add and subtract a suitable distance to/from the standard anchors; creating two coordinates related to the input and output anchors but dislocated a suitable amount towards the interior of the block. Using `calc` library, one examples is:




```
\draw(0,0)
  to [contact NO={info={cond}}]  ++(1,0)
  to [block={inputs={EN,A,C},outputs={ENO,B,C},
      symbol=myFB, info=myInst, name=mf1,
      minimum width=1.6cm,
      input sep=1.2em, output sep=1.2em}] ++(3,0)
  to [coil={info={X}}] +(1,0);
\draw (mf1.input 2) -- +(-0.3cm,0) node[left]{v1}
      (mf1.input 3) -- +(-0.3cm,0) node[left]{v3}
      (mf1.output 2) -- +(0.3cm,0) node[right]{v2}
      (mf1.output 3) -- +(0.3cm,0)
      ($(mf1.input 3) + (1em,0)$) -- ($(mf1.output 3) - (1em,0)$);
```

If you prefer to use the `\coordinate` command, place:

```
\coordinate[xshift=1em] (p1) at (mf1.input 3);
\coordinate[xshift=-1em] (p2) at (mf1.output 3);
```

between the two `\draw` commands and replace de last code line by `(p1) --(p2);`.

# 7 Design Guidance

This section brings some recommendations that reflect the way I produce ladder diagrams. It may or may not work for you. Feel free to e-mail me if you have better ideas.

The first to consider is that `tikz-ladder` uses the `\tikzcircuitssizeunit` to keep all figures proportional. Therefore, when you consider any dimension related to symbol size it is good idea to set that dimension in respect to `\tikzcircuitssizeunit`, i.e., using `\tikzcircuitssizeunit` as the length unit[7]. The default value of `\tikzcircuitssizeunit` is 7 pt or approximately 2.46 mm and it can be set by the `circuit symbol unit` key among several other keys. Even better, you can establish your own length unit and set $x$ and $y$ to that length unit. In this way you will be working on a grid; if it is too big or too small you will have to change a single declaration. The `\tikzcircuitssizeunit` is too small for that purpose, so I use and recommend `5\tikzcircuitssizeunit` instead. Moreover, I like to leave a small space between rungs, so a second new length can be used to keep this amount. Again, you can adjust it globally. Thus, just after loading the `tikz-ladder` library I use to declare:

```
\newlength{\ladderskip}
\setlength{\ladderskip}{5\tikzcircuitssizeunit} % 5|tikzcircuitssizeunit = 35pt
\newlength{\ladderrungsep}
\setlength{\ladderrungsep}{.2\ladderskip}
\def\ladderrungend#1{\pgftransformyshift{-#1\ladderskip-\ladderrungsep}}
```

where:

`ladderskip` is the length that controls all distances in the diagram;

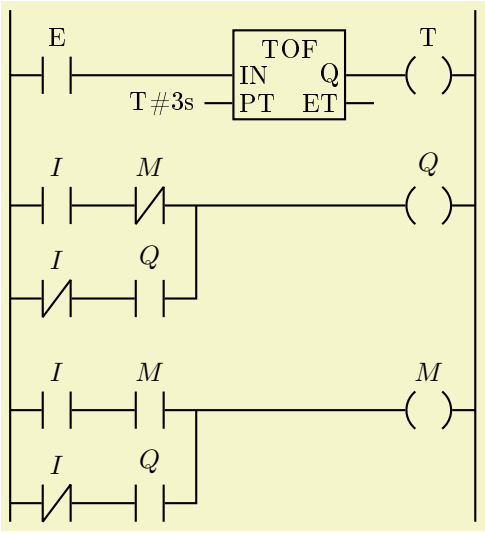
`ladderrungsep` is the additional separation between two successive rungs; and

`ladderrungend` is a macro that marks the end of a rung. Actually, it leaves a space and set the new vertical datum. The parameter of this macros is the number of rows the ending rung has plus some extra space if you like it.

[7] Therefore, all the examples in the previous sections should be reviewed.

To keeps track of a few rows in a rung is easy, but to place manually everything in the diagram is not. So, I devised a strategy: draw one rung at a time and move the datum (reset the origin) at the end of every rung, thus I can start fresh another rung. This is done by macro `\ladderrungend` which resets the vertical reference to a vertical position $n$ rows below the current position (plus some inter rung space). The macro also serves to *mark* rung end in the code (Ti*k*Z "program") that generates the diagram making it more readable.

The options passed to the Ti*k*Z environment in the next example are `[ladder,thick,x=\ladderskip,y=\ladderskip]` meaning it is a ladder diagram and $x$ and $y$ length unit are both set to `1\ladderskip`.




```
\begin{tikzpicture} [ladder,thick,x=|ladderskip,y=|ladderskip]
  \draw(0,0)
    to [contact NO={info={E}}]  ++(1,0) --++(1,0)
    to [block={inputs={IN,PT},outputs={Q,ET},symbol=TOF,name=tp1,
        minimum width=1.2|ladderskip,
        input sep=0.3|ladderskip, output sep=0.3|ladderskip}] ++(2,0)
    to [coil={info={T}}] +(1,0) coordinate(laddertopright);
  \draw (tp1.output 2) -- +(0.3\ladderskip,0)
        (tp1.input 2) -- +(-0.3\ladderskip,0) node[left]{T\#3s};
\ladderrungend{1.2}
  \draw(0,0)
    to [contact NO={info={$I$}}] ++(1,0)
    to [contact NC={info={$M$}}] ++(1,0) coordinate(laddercoil) -- ++(2,0)
    to [coil={info={$Q$}}] ++(1,0);
  \draw(0,-1)
    to [contact NC={info={$I$}}] ++(1,0)
    to [contact NO={info={$Q$}}] ++(1,0) -- (laddercoil);
\ladderrungend{2}
  \draw(0,0)
    to [contact NO={info={$I$}}] ++(1,0)
    to [contact NO={info={$M$}}] ++(1,0) coordinate(laddercoil) -- ++(2,0)
    to [coil={info={$M$}}] ++(1,0);
  \draw(0,-1)
    to [contact NC={info={$I$}}] ++(1,0)
    to [contact NO={info={$Q$}}] ++(1,0) -- (laddercoil);
\ladderrungend{2}
% power rails
  \draw let \p1=(laddertopright) in
    (0,\y1+0.7\ladderskip) -- (0,\ladderskip)
    (\x1,\y1+0.7\ladderskip) -- (\x1,\ladderskip);
\end{tikzpicture}
```

In the example, the first row has a timer, so an extra space is needed and for this 1.2 is given to `\ladderrungend`. Note that the `\draw (tp1.output 2)`... is not the start of a new row (the first rung has only one row); it is used solely to place the timer terminals for PT and ET.

Every rung starts with `\draw(0,0)` to mark the first position. The next row of the same rung will start with `\draw(0,-1)` and so forth.

All contacts and coils are placed by something like, e.g., `to [contact NC={info={$I$}}] ++(1,0)`. The `to` command places the element between the current position and the next position which is one length unit at the right of the current position. Blocks are bigger and need more space, so after a block use

`++(2,0)`. When a row is connected to a row above it, it is wise to mark the connection point on the row above using, e.g., `coordinate(laddercoil)`. Thus, the current row can be connected by `--(laddercoil)` or `-| (laddercoil)`. If there are several rows, connect only the last one.

You also may need some fillers like `--++(2,0)` because the row (or rung) is shorter than the others or because you want to leave some space before a block.

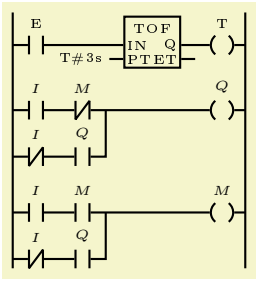
The last thing to do is to draw the power rails. For this, the end of the first row of the first rung was marked with `coordinate(laddertopright)`. In the example, both left and right power rails were drawn. To draw only the left one (mandatory[8]), remove the `(\x1,\y1+0.7\ladderskip) --(\x1,\ladderskip)`.

You may need to change the diagram size. There are a few options: place it into `\resizebox`; use the key `scale`; and change the `\tikzcircuitssizeunit` through the key `circuit symbol unit`. Placing into a `\resizebox` changes everything and it is the preferable option for presentations. The key `scale` only changes the space between symbols, but the font size, line width and symbol sizes are all kept the same. Changing the `\tikzcircuitssizeunit` keeps the font size and line width, but changes the symbols size and, if you were careful, the distance between symbols.

To shrink the diagram to half its normal size, place these commands before your `\begin{tikzpicture}`. Note that you will have to undo this after the diagram.

```
\tikzset{circuit symbol unit=3.5pt}
\setlength{\ladderskip}{5\tikzcircuitssizeunit} % 5\tikzcircuitssizeunit = 35pt
\setlength{\ladderrungsep}{.2\ladderskip}
\tiny
```

Both `\ladderskip` and `\ladderrungsep` were redefined, also the font size was changed. Incidentally, `\tiny` means 50% of the current size. The result should be:




This may not be the best way to perform big adjustments, but it can help in some occasions where a small adjust is needed. Also, if you do it frequently, consider to write a macro to encapsulate the feature.

# 8 Known Issues

I am not happy with the block symbol anchors, particularly `text`, `mid` and `base` anchors. Also, `center` is not in the centre and it is disturbing, to say the least. That might be changed in the future. The way the library was written also annoys me: it seems that the official libraries are written in two separated files: one for Ti*k*Z stuff and another for PGF, but I don't know how to separate it, thus we have a single file (at least for now).

# 9 Final Remarks

This package has been tested and used for more than three years, so I do believe it is mature by now and I decided to share it. On the other hand, I was the only person who used it[9], therefore idiosyncrasies were not detected.

Any comments, suggestions, and feedbacks are welcomed. I will do my best to answer as soon as possible. My contact e-mail is in the first page.

It should be great if someone with experience in writing Ti*k*Z libraries could have a look in the code and point out error or improvements to be made.

Typesetting ladder diagrams may be boring and time consuming. One thing you can try is `JQM - Java Quine McCluskey` for minimization of Boolean functions available on `https://sourceforge.`

[8] Actually, the standard reads "The right power rail may be explicit or implied".

[9] Not entirely true, two people asked me for the package, but I never heard from them again.

net/projects/jqm-java-quine-mccluskey/. It can generate the solution and create the corresponding ladder diagram based on a given truth table. Unfortunately, it does not place blocks, so the last example was generated with it, but the rung with the timer had to be done manual and also some fillers had to be added.