

# Typesetting Ladder Diagrams with $\text{\LaTeX}$ and $\text{TikZ}$

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

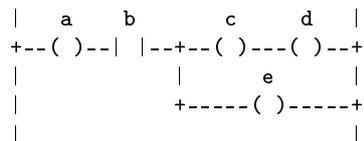
Version: 1.3, Version date: 2022-04-10

## 1 Abstract

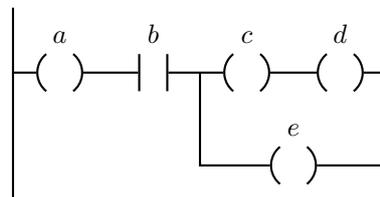
Ladder diagram (LD) is a graphical programming language that has evolved from electrical wiring diagrams for relay control systems used with programmable controllers (PLC<sup>1</sup>) as described in the international standard IEC-61131-3 [1]. An LD program enables the programmable controller to test and modify data using standard 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  $\text{TikZ}$  symbols to draw high-quality ladder diagrams. All standard and some non-standard symbols are possible, including all kinds of contacts, coils and blocks. I decided to write this package, although it is available another package named `ladder` that also uses  $\text{TikZ}$  to typeset ladder diagrams because that package seems<sup>2</sup> to lack support for blocks. The `tikz-ladder`, on the contrary, supports all features described in IEC-61131-3 [1], namely, blocks (for functions, function blocks, etc.), contacts and coils.

## 2 Ladder Diagram

According to IEC-61131-3 [1, sic, p. 211], the usage of letters, semi-graphic 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 with a standard that should be used by everyone: you can do whatever you want. Thus, this package provides  $\text{TikZ}$  symbols for typesetting ladder diagram as close as possible to the standard, but not too close since a program in the standard would look like:



and it is probably not what you want. With `tikz-ladder`, you can produce something like<sup>3</sup>:



The next section provides a formal introduction to the package and it is not meant to be complete. Section 4 gives some design guidance which is arguably the most important part of this document. The remaining contains more technical material and examples. The last sections address some specific issues and contain some final remarks, acknowledgements and references.

<sup>1</sup>Formerly known as programable logic controllers.

<sup>2</sup>Sorry, but the documentation is in French and I limited myself to looking at the figures.

<sup>3</sup>This slightly awkward example was extracted from reference [1, 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 input  $b$ .” In the 2013 version, there is a typo: the  $a$  is missing. The 2003 version is correct.

### 3 Ladder Diagram Library

#### TikZ Library `circuits.plc.ladder`

```
\usepgflibrary{circuits.plc.ladder} % LATEX and plain TEX and pure pgf
\usepgflibrary[circuits.plc.ladder] % ConTEXt and pure pgf
\usetikzlibrary{circuits.plc.ladder} % LATEX and plain TEX when using TikZ
\usetikzlibrary[circuits.plc.ladder] % ConTEXt when using TikZ
```

This library provides graphics for ladder diagrams related to programmable controllers (PLC) and according to the international standard IEC-61 131-3 [1]. The library was written to extend the standard TikZ-library `circuit`. The reader is urged to read Section “Circuit Libraries” of reference [2]. This library defines many symbols and the following keys, macros and lengths:

`/tikz/circuit plc ladder` (no value)

This style calls `circuit plc ladder` and installs ladder diagram graphics for symbols like contacts, coils and blocks. Users are urged to read Section 4 for discussed examples and strategies for typesetting ladder diagrams.

#### `\ladderskip`

A L<sup>A</sup>T<sub>E</sub>X length is automatically set by `circuit plc ladder` to the vertical length unit, `y`. It can be used to prescribe widths and heights, especially regarding `blocks`. This length will be accurate even if the vertical length unit is redefined, for instance, via an option like `y=1cm`.

#### `\ladderrungend{⟨number⟩}`

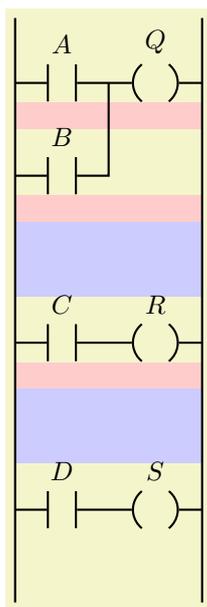
This macro skips a `⟨number⟩` of `\ladderskip` and resets the vertical coordinate so the next rung will start at (0,0). It is meant to be used as an *end of rung mark*. The `⟨number⟩` does not need to be the number of rows in the current rung. Some arbitrary space can be used.

#### `\ladderpowerrails`

This macro draws one or two power rails. It trusts `\ladderrungend` to draw the lines with the correct length. For two power rails, left and right, it also depends on the definition of point `laddertoprigh` at the end of the first row of the top rung.

`/tikz/ladderrungsep=⟨number⟩` (no default, initially 0.2)

This key sets the extra space left at the end of every rung given in `\ladderskip` length unit.



```
\begin{tikzpicture}[circuit plc ladder,thick,ladderrungsep=0.8]
\draw(0,0)
to [contact NO={info={A}},name=ca] ++(1,0)
to [coil={info={Q}}] ++(1,0) coordinate(laddertoprigh);
\fill[red!20] (laddertoprigh |- ca.south) rectangle +(-2,-1em);
\draw(0,-1)
to [contact NO={info={B}},name=cb] ++(1,0) -- +(0,1);
\fill[red!20] (laddertoprigh |- cb.south)
rectangle +(-2,-1em) coordinate (pb);
\fill[blue!20] (pb) rectangle ++(2,-0.8);
\ladderrungend{2}
\draw(0,0)
to [contact NO={info={C}},name=cc] ++(1,0)
to [coil={info={R}}] ++(1,0);
\fill[red!20] (laddertoprigh |- cc.south)
rectangle +(-2,-1em) coordinate (pc);
\fill[blue!20] (pc) rectangle ++(2,-0.8);
\ladderrungend{1}
\draw(0,0)
to [contact NO={info={D}}] ++(1,0)
to [coil={info={S}}] ++(1,0);
\ladderrungend{1}
\ladderpowerrails
\end{tikzpicture}
```

In this example, the commands `fill` draw pink rectangles to illustrate the free vertical space of about `1em` between one row and the next inside the same rung and blue rectangles to illustrate the extra space between consecutive rungs inserted by `ladderrungsep=0.8`. If `ladderrungsep` is

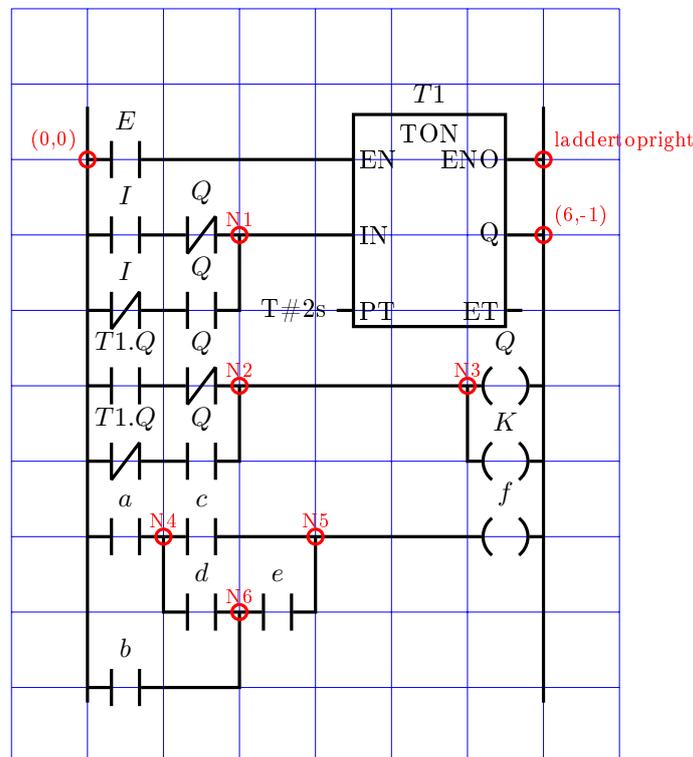
set to zero, the distance between rungs will be the same as the distance between rows inside a rung, namely `1 \ladderskip`. As a consequence, the tip of the letters *C*, *R*, *D* and *S* would go inside the ping rectangles just like the tip of the letter *B*.

The next section brings examples and strategies for typesetting ladder diagrams. The detailed description of the library features is left for the subsequent sections.

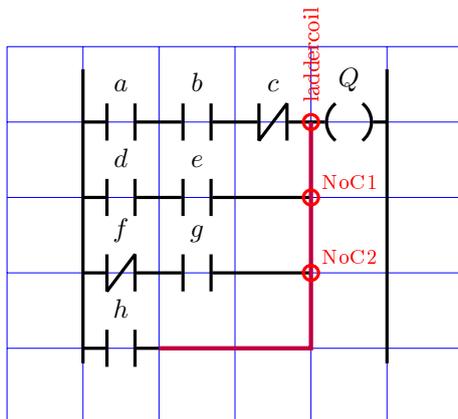
## 4 Design Guidance

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. One concept behind Section “Circuit Libraries” of reference [2] is that networks are drawn slightly different from other graphics in *TikZ*. In essence, we draw a line and place several symbols on this line. Therefore, although it is possible to draw a symbol from the `tikz-ladder` library using the command `\node`, in general, it is much easier to use a combination of commands `\draw` and `to`. The `tikz-ladder` has been written to work this way.

The first thing to consider is that, as the manual says [2], “*TikZ ist kein Zeichenprogramm*” which translates to “*TikZ is not a drawing program*”. You shall start with a draft of your diagram and then codify it using *TikZ*. Generally, a handmade pencil sketch on grid paper will do. In this draft, you shall use node names for future reference. So, an example draft would look like this:



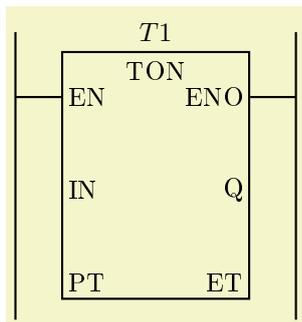
The diagram is constituted of horizontal wires or lines, where the components (contacts, coils and blocks) are inserted forming a row of elements, and vertical wires that joint with the horizontal ones. Nodes (coordinates) are placed at wire junctions and other meaningful points. Note that, in the example above, some nodes are indicated by name while others by their coordinates (see the red circles). Using the name instead of coordinates makes the diagram easier to modify, but the node position is needed. In most cases, the node position arises naturally from the construction of the diagram, but sometimes it is unknown and coordinates or cumbersome operations on coordinates have to be used instead. As a rule of thumb, *named nodes should be placed using the command coordinate at every junction in which one wire goes down then right*, like N3 and N4. Most of the time, for wires that go down then left like N1, N2, N5 and N6, the connection can be made using relative coordinates like in `-- ++(0,1)` or `-- +(0,1)` since those connections will be made when the next row, the row below, is coded. A situation when it is worth using named nodes, even if wires below the current row come exclusively from the left, is when there is more than one connection like this:



Here, node `laddercoil` is connected to contact `e`, `g` and `h`. In TikZ, only the last connection is made: contact `h` is connected to node `laddercoil`; horizontal fillers, `-- +(1,0)`, are placed after contacts `e` and `g` and the second and third rows end in those fillers; nodes `NoC1` and `NoC2` are not used and can be suppressed.

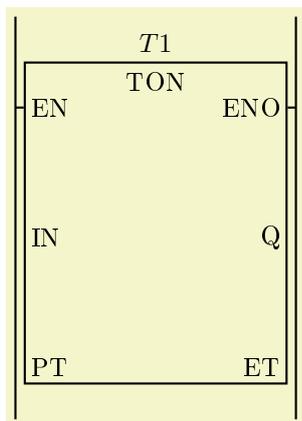
The second thing to consider is that `tikz-ladder` uses the `\tikzcircuitsizeunit` to keep all figures proportional. Therefore, when you consider any length related to symbol size, it is a good idea to specify that length using `\tikzcircuitsizeunit` as the length unit<sup>4</sup>. Even better, used the new length `\ladderskip`.

From version 1.3 on, `tikz-ladder` automatically sets `x` and `y` length units to `5\tikzcircuitsizeunit` or approximately 12.35 mm. It also sets a new length `\ladderskip` to `y` even if `y` is manually modified, for instance, via writing `y=7\tikzcircuitsizeunit` in the options list. The intention is to use `\ladderskip` as the vertical length unit to specify any length that depends on the diagram size. In particular, `\ladderskip` is used internally as the vertical distance between rows inside a rung and to draw power rails. Users can use `\ladderskip` to set `input sep` and `output set`, for instance. Compare:



```
\begin{tikzpicture}[circuit plc ladder,thick]
\draw(0,0)
to [block={info=$T1$,inputs={EN,IN,PT},
outputs={ENO,Q,ET},symbol={TON},
name=T1,minimum width=2\ladderskip,
input sep=\ladderskip,output sep=\ladderskip}]
++(3,0) coordinate(laddertopright);
\ladderrungend{3}
\ladderpowerrails
\end{tikzpicture}
```

with



```
\begin{tikzpicture}[circuit plc ladder,thick,y=7\tikzcircuitsizeunit]
\draw(0,0)
to [block={info=$T1$,inputs={EN,IN,PT},
outputs={ENO,Q,ET},symbol={TON},
name=T1,minimum width=2\ladderskip,
input sep=\ladderskip,output sep=\ladderskip}]
++(3,0) coordinate(laddertopright);
\ladderrungend{3}
\ladderpowerrails
\end{tikzpicture}
```

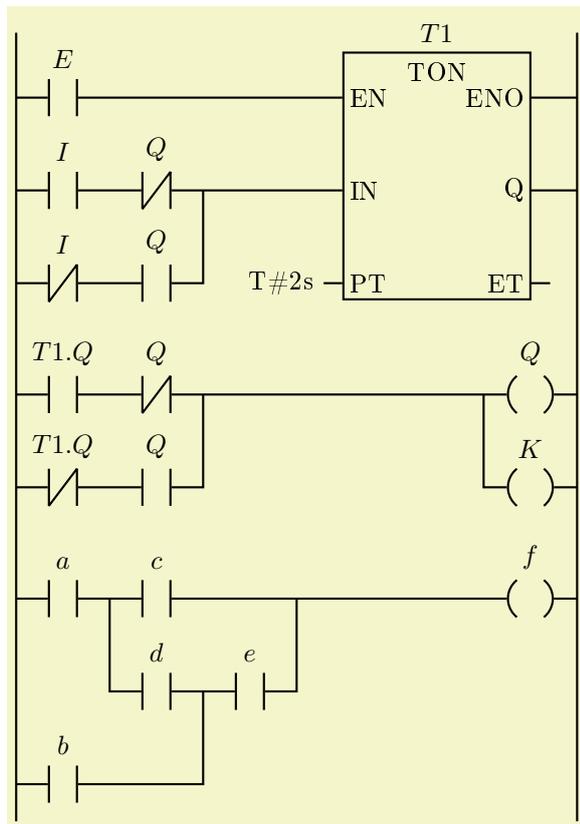
The width of the diagram remained the same because the horizontal length unit, `x`, was not changed, but `\ladderskip` was modified by `y=7\tikzcircuitsizeunit` affecting the block width and height: the width

<sup>4</sup>The default value of `\tikzcircuitsizeunit` is 7 pt or approximately 2.46 mm and it can be set by the `circuit symbol` unit key among several other keys.

because `minimum width` depends on `\ladderskip`; the height because of `output sep` and `output sep`. Also, the power rails' length was affected (see below for more on power rails).

Keeping track of a few rows in a rung is easy, but manually placing everything in the diagram is not, particularly, if this diagram contains many rungs. So, a strategy was devised: draw one rung at a time and move the datum (reset the origin) at the end of every rung. This is done by a new macro `\ladderrungend` which resets the vertical reference to a vertical position  $n$  rows below the current position (plus some inter-rung space) where  $n$  is passed as a macro parameter. Note that `\ladderrungend` relies on `\ladderskip`. It also serves to *mark* the rung end in the code (TikZ “program”) that generates the diagram making it more readable. The extra optional space between consecutive rungs is set as a fraction of `\ladderskip` by key `ladderrungsep`. One important advantage of this strategy is that this makes it possible to change the rung position by simply cutting and pasting it around because every node starts at (0,0). No further adjustments nor editing are required.

Returning to the example at the beginning of this section, we start coding at position (0,0) and use command `to` to place the symbols rung by rung, row by row:



```
\begin{tikzpicture}[circuit plc ladder,thick]
\draw(0,0)
  to [contact NO={info={\$E\$}}] ++(1,0) --++(2,0)
  to [block={info=\$T1$, inputs={EN,IN,PT},
    outputs={ENO,Q,ET}, symbol={TON}, name=T1,
    minimum width=2\ladderskip,
    input sep=\ladderskip,output sep=\ladderskip}]
    ++(3,0) coordinate(laddertoprigh);
\draw(0,-1)
  to [contact NO={info={\$I\$}}] ++(1,0)
  to [contact NC={info={\$Q\$}}] ++(1,0) coordinate(N1)
  -- (T1.input 2)
  (T1.output 2) -- (6,-1); % grid coordinate
\draw(0,-2)
  to [contact NC={info={\$I\$}}] ++(1,0)
  to [contact NO={info={\$Q\$}}] ++(1,0) -- (N1);
\draw (T1.input 3) -- +(-0.2,0) node[left]{T\#2s}
  (T1.output 3) -- +(0.2,0);
\ladderrungend{3}
\draw(0,0)
  to [contact NO={info={\$T1.Q\$}}] ++(1,0)
  to [contact NC={info={\$Q\$}}] ++(1,0) coordinate(N2)
  -- ++(3,0) coordinate(N3)
  to [coil={info={\$Q\$}}] ++(1,0);
\draw(0,-1)
  to [contact NC={info={\$T1.Q\$}}] ++(1,0)
  to [contact NO={info={\$Q\$}}] ++(1,0) -- (N2)
  (N3) -- ++(0,-1)
  to [coil={info={\$K\$}}] ++(1,0);
\ladderrungend{2}
\draw(0,0)
  to [contact NO={info={\$a\$}}] ++(1,0) coordinate(N4)
  to [contact NO={info={\$c\$}}] ++(1,0)
  -- ++(1,0) coordinate(N5)
  -- ++(2,0)
  to [coil={info={\$f\$}}] ++(1,0);
\draw(N4) -- ++(0,-1)
  to [contact NO={info={\$d\$}}] ++(1,0) coordinate(N6)
  to [contact NO={info={\$e\$}}] ++(1,0) -- (N5);
\draw(0,-2)
  to [contact NO={info={\$b\$}}] ++(1,0) -| (N6);
\ladderrungend{3}
\ladderpowerrails
\end{tikzpicture}
```

First, a normally open contact  $E$  is placed in the path by command `to`. This command interrupts the path, draws a node with the symbol corresponding to its options (a normally open contact in this case) and returns to the path drawing a line between the coordinate before it and the symbol and a second line between the symbol and the coordinate after the command `to`. So, each command `to` is written between two coordinates. It is very convenient to use relative coordinates to prescribe the second coordinate such that the  $x$  value will be the space reserved for the symbol. Thus, `++(1,0)` will give one length unit for the symbol (see the grid in the example draft). Next, a filler of two units is placed with command `-- ++(2,0)`. Then, a block for a timer is placed leaving three-length units. The `minimum width`, `input sep` and `output sep` are all specified in

terms of `\ladderskip`. The first row of the first rung ends with the command `coordinate(laddertopright)` which will be used by macro `\ladderpowerrails` to draw, quite obviously, the power rails.

The first rung continues at point  $(0, -1)$ , followed by contacts  $I$  and  $Q$ , both placed into a gap of one length unit each. Node  $N1$  is declared just after contact  $Q$ . Next, a line is drawn to the timer block second input, `-- (T1.input 2)`. The line connecting the second output to the right power rail is drawn using the absolute coordinate,  $(6, -1)$ . This point can also be specified using `(T1.output 2 -| laddertopright)` which is more flexible and less prone to mistakes, see Section Coordinates at Intersections of reference [2]. This circumstance, connecting an output to an unspecified point, is uncommon but can happen. Normally, only the first block output is connected and it is done by the command `to` which inserts the block.

Continuing at point  $(0, -2)$ , two contacts,  $I$  and  $Q$ , are placed into a gap of one length unit each and this row is connected to the row above by command `-- (N1)` or just `-- +(0,1)` because we know for sure this contact will be connected with the line right above it. Here, a single `+` is used since local coordinates do not need to be updated.

Finally, the preset time is written outside the block and little lines are drawn for the preset time and elapsed time – terminals for PT and ET. The first rung ends with macro `\ladderrungend{3}` and the next rung will start at  $(0,0)$ , just like the first one.

The second rung starts with two contacts, node  $N2$ , a filler `-- +(3,0)`, another node  $N3$  and a coil  $Q$ . Node  $N3$  is placed to mark the start point of the descending line connected to coil  $K$ . In the next row, starting at  $(0, -1)$ , two contacts are inserted and the line is connected with the row above by command `-- (N2)` and, just like the first rung, it can be replaced by `-- +(0,1)`. The path restarts at node  $N3$  moving down and coil  $K$  is inserted. The second rung ends with `\ladderrungend{2}`.

The third and last rung starts at  $(0,0)$ . Node  $N4$  is placed between contacts  $a$  and  $c$ . This node is used to place contact  $d$ . Node  $N5$  is placed between two fillers where a connection from contact  $e$  arrives. A single filler of three length unit can be used after contact  $c$  if node  $N5$  is suppressed because the connection is made after placing contact  $e$  by a command like `--+(0,1)`. Node  $N6$  is placed between contacts  $d$  and  $e$  to make it easy to connect the left side of contact  $b$ . This connection could be done by `-- +(1,0) --+(0,1)` or `-|++(1,1)` but if the previous row is modified it would also have to be changed accordingly. The use of named nodes avoids this need.

The third rung ends with `\ladderrungend{3}` and the macro `\ladderpowerrails` is used to draw the power rails on both sides of the diagram. This macro relies on node `laddertopright` placed in the very first row and the use of macro `\ladderrungend` to draw the power rails at the correct length.

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)` or even a higher value.

It is possible to pass a value different from the number of rows into a rung to macro `\ladderrungend`. This can be necessary, for instance, if the rung contains a big block. If, however, the user is not happy with the default separation between all rungs, the key `ladderrungsep` can be used to set a more suitable value.

While typesetting a complex ladder diagram, it might be useful to draw a grid at least until you are satisfied with the produced diagram. Care should be taken, however, because macro `\ladderrungend` leaves extra space between successive rungs which would lead to grid misalignment. This can be avoided by temporarily setting `ladderrungsep` to zero or another integer. For instance, start your diagram with:

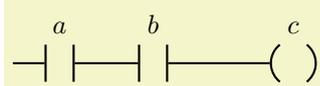
```
\begin{tikzpicture}[circuit plc ladder,thick,ladderrungsep=0]
  \draw[blue,very thin,step=\ladderskip] (0,0) grid (5,-7);
```

After completion, `ladderrungsep` can be returned to its original setting or any other suitable value and the grid can be erased or commented out.

Another hint that can help is to make named nodes visible. Assuming all nodes start with  $N$  followed by a sequential number, the following code can be placed at the end of the picture to display (draw) the first four nodes:

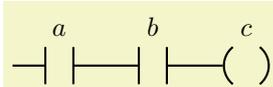
```
\foreach \x in {1,...,4}
  \node[label={[[label distance=-4pt,above, red,font=\tiny]N\x}],
  circle,inner sep=1pt,draw=red,thin] at (N\x) {};
```

The right power rail is optional and many people are refraining from using it. Thus, the line on the right side of some symbols should not be drawn. To prevent `TikZ` from drawing a line leaving the right side of the symbol, place this symbol at the end of the gap by typing `at end` or `pos=1` in the options list; like



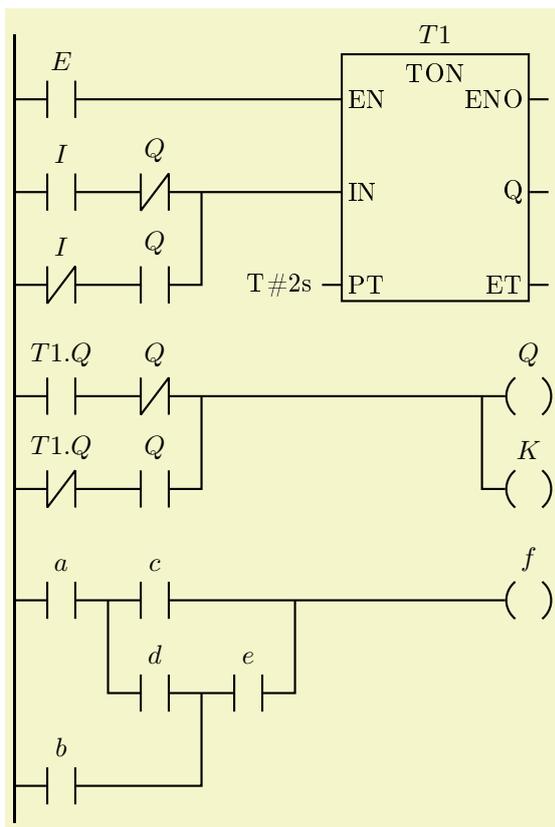
```
\tikz[circuit plc ladder,thick]
\draw(0,0) to[contact NO={info={\$a\$}}] ++(1,0)
to[contact NO={info={\$b\$}}] ++(1,0)
to [coil={info={\$c\$},pos=1}] ++(1,0);
```

The total gap for the coil can be reduced since the right side is not used anymore and this will make the space evenly distributed:



```
\tikz[circuit plc ladder,thick]
\draw(0,0) to[contact NO={info={\$a\$}}] ++(1,0)
to[contact NO={info={\$b\$}}] ++(1,0)
to [coil={info={\$c\$},pos=1}] ++(0.5,0);
```

Moreover, if no `laddertoprigh`t is set, `\ladderpowerrails` will draw only the left power rail. Returning once more to the example at the beginning of this section:



```
\begin{tikzpicture}[circuit plc ladder,thick]
\draw(0,0)
to [contact NO={info={\$E\$}}] ++(1,0) --++(2,0)
to [block={info=\$T1$, inputs={EN,IN,PT},
outputs={ENO,Q,ET}, symbol={TON}, name=T1,
minimum width=2\ladderskip, pos=1,
input sep=\ladderskip, output sep=\ladderskip}]
++(1.5,0);
\draw(0,-1)
to [contact NO={info={\$I\$}}] ++(1,0)
to [contact NC={info={\$Q\$}}] ++(1,0) coordinate(N1)
-- (T1.input 2);
\draw(0,-2)
to [contact NC={info={\$I\$}}] ++(1,0)
to [contact NO={info={\$Q\$}}] ++(1,0) -- (N1);
\draw (T1.input 3) -- +(-0.2,0) node[left]{T\#2s}
(T1.output 1) -- +(0.2,0)
(T1.output 2) -- +(0.2,0)
(T1.output 3) -- +(0.2,0);
\ladderrungend{3}
\draw(0,0)
to [contact NO={info={\$T1.Q\$}}] ++(1,0)
to [contact NC={info={\$Q\$}}] ++(1,0) coordinate(N2)
-- ++(3,0) coordinate(N3)
to [coil={info={\$Q\$},pos=1}] ++(0.5,0);
\draw(0,-1)
to [contact NC={info={\$T1.Q\$}}] ++(1,0)
to [contact NO={info={\$Q\$}}] ++(1,0) -- (N2)
(N3) -- ++(0,-1)
to [coil={info={\$K\$},pos=1}] ++(0.5,0);
\ladderrungend{2}
\draw(0,0)
to [contact NO={info={\$a\$}}] ++(1,0) coordinate(N4)
to [contact NO={info={\$c\$}}] ++(1,0)
-- ++(1,0) coordinate(N5)
-- ++(2,0)
to [coil={info={\$f\$},pos=1}] ++(0.5,0);
\draw(N4) -- ++(0,-1)
to [contact NO={info={\$d\$}}] ++(1,0) coordinate(N6)
to [contact NO={info={\$e\$}}] ++(1,0) -- (N5);
\draw(0,-2)
to [contact NO={info={\$b\$}}] ++(1,0) -| (N6);
\ladderrungend{3}
\ladderpowerrails
\end{tikzpicture}
```

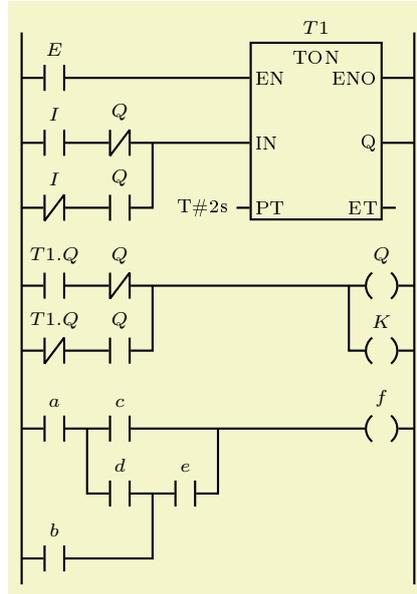
Note that the space meant for all symbols that would be connected to the right power rail, namely the timer `T1` and coils `Q`, `K` and `f`, is divided by two.

You may need to change the diagram size. There are a few options: place it into `\resizebox`; use the key `scale`; set `x` and `y` length unit; 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` or setting set `x` and `y` length unit keeps the font size and line width, but changes the size of the symbol and, if you were careful, the distance between symbols.

For example, to shrink the diagram to 70% of 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=4.9pt}
\scriptsize
```

Incidentally, `\scriptsize` means 70% of the current font 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 adjustment is needed. Also, if you do it frequently, consider writing a macro to encapsulate the feature. The next sections give details on the symbols supported by `tikz-ladder` divided in three main categories:

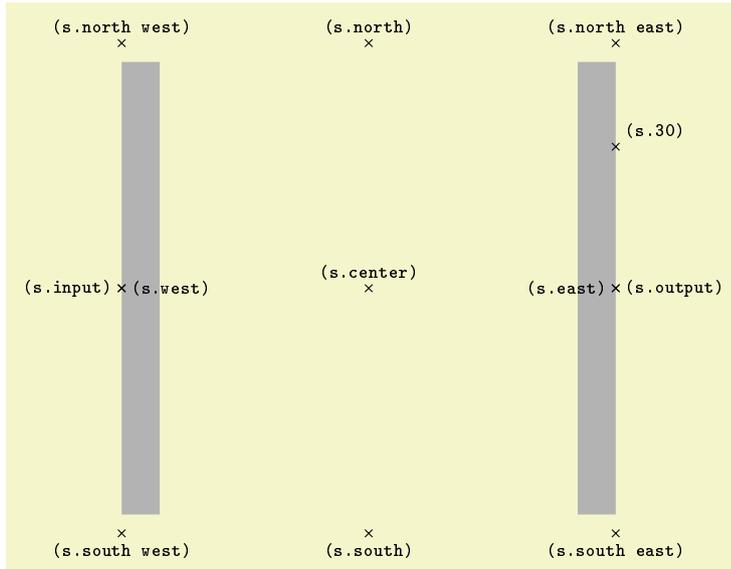
**contacts** all kinds of standard contacts;

**coils** all kinds of standard coils are supported and the curvature of a coil is customizable;

**blocks** all kinds of standard blocks are supported and examples of timers, counter, bistable functions blocks, edge detection functions blocks and call representation are given.

## 5 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 [1, sic, p. 216].



```
\begin{tikzpicture}[circuit plc 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 standard 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[circuit plc ladder, thick]
  \draw(0,0) to [contact NO={info={***}}] ++(2,0);
```

Normally closed contact (NC):



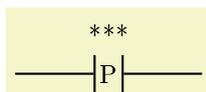
```
\tikz[circuit plc ladder, thick]
  \draw(0,0) to [contact NC={info={***}}] ++(2,0);
```

Variation of the normally closed contact (NC):



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

Positive transition-sensing contact (P):



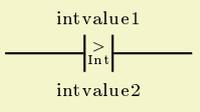
```
\tikz[circuit plc ladder, thick]
  \draw(0,0) to [contact P={info={***}}] ++(2,0);
```

Negative transition-sensing contact (N):

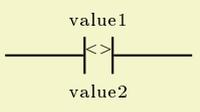


```
\tikz[circuit plc ladder, thick]
  \draw(0,0) to [contact N={info={***}}] ++(2,0);
```

Compare contact (typed):

	<pre>\tikz[circuit plc ladder,thick] \draw(0,0) to [contact NO={   info={\scriptsize intvalue1},   info'={\scriptsize intvalue2},   symbol={\tiny\$\genfrac{}{}{0pt}{}{&gt;}{}{}}}] ++(2,0);</pre>
---	--

Compare contact (overloaded):

	<pre>\tikz[circuit plc ladder,thick] \draw(0,0) to [contact NO={   info={\scriptsize value1},   info'={\scriptsize value2},   symbol={\tiny\$\genfrac{}{}{0pt}{}{&lt;&gt;}{}{}}}] ++(2,0);</pre>
---	--

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

## 5.1 Keys for contacts

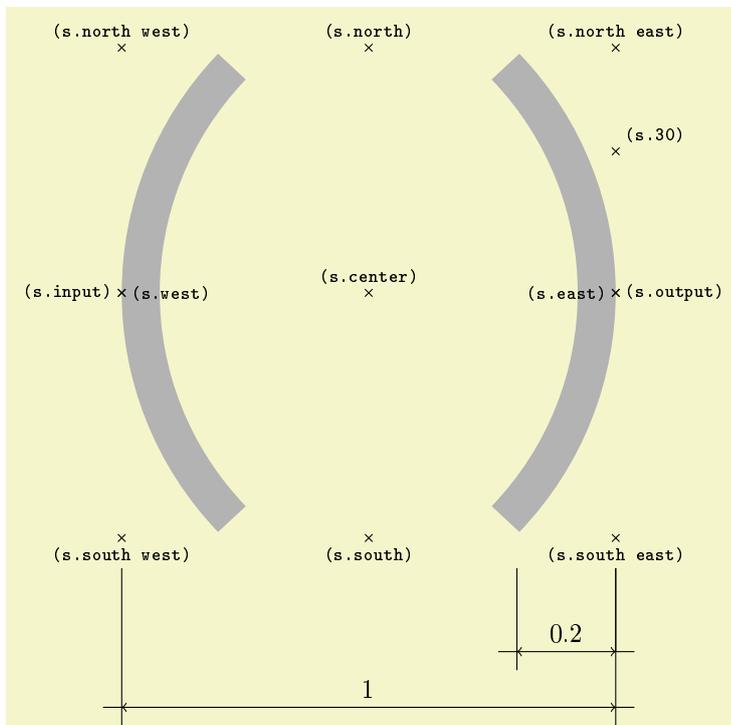
The most common key used with contacts is `info` which sets the variable name associated with the contacts. One may also need `info'` and `name`; both standard keys. In addition to the keys described in Section "Circuit Libraries" of reference [2], contacts accept:

`/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.

## 6 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 [1, sic, p. 218].

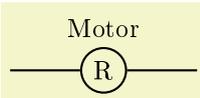


```

\begin{tikzpicture}[circuit plc ladder]
  \node[name=s, shape=coil 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=*] 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<sup>5</sup>:



```

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

```

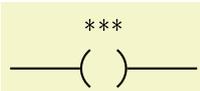
Note that it is also necessary to set the `minimum size` or `minimum height` because coils have a proportion of  $2.4 \times 2$ . If you do this frequently, you can set the style for the coils like:

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

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

All kinds of standard 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):

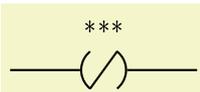


```

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

```

Negated coil (normally activated, NA):



```

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

```

Variation of the negated coil (normally activated, NA):

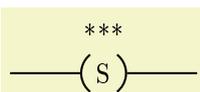


```

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

```

Set (latch) coil:

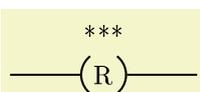


```

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

```

Reset (unlatch) coil:



```

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

```

<sup>5</sup>It can be used to draw relay coils according to NEMA – National Electrical Manufacturers Association.

Positive transition-sensing coil:

<pre>*** ——(P)——</pre>	<pre>\tikz[circuit plc ladder,thick] \draw(0,0) to [coil={info={***},symbol=P}] ++(2,0);</pre>
------------------------	--

Negative transition-sensing coil:

<pre>*** ——(N)——</pre>	<pre>\tikz[circuit plc ladder,thick] \draw(0,0) to [coil={info={***},symbol=N}] ++(2,0);</pre>
------------------------	--

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 recommended because it disagrees with IEC-61131-3 [1], to use non-standard coils, e.g., some people use L (for latch) and U (for unlatch) instead of, S and R, respectively. This is achieved by setting the `symbol` of a coil like:

<pre>*** ——(L)——</pre>	<pre>\tikz[circuit plc ladder,thick] \draw(0,0) to [coil={info={***},symbol=L}] ++(2,0);</pre>
------------------------	--

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

## 6.1 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 reference [2], 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 like positive and negative transition-sensing coils.

<pre>*** ——(L)——</pre>	<pre>\tikz[circuit plc ladder,thick] \draw(0,0) to [coil={info={***},symbol=L}] ++(2,0);</pre>
------------------------	--

`/tikz/coil ladder curvature=<curvature index>` (no default, initially 0.2)

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, the `minimum size` will have to be adjusted to correct the coil aspect ratio.

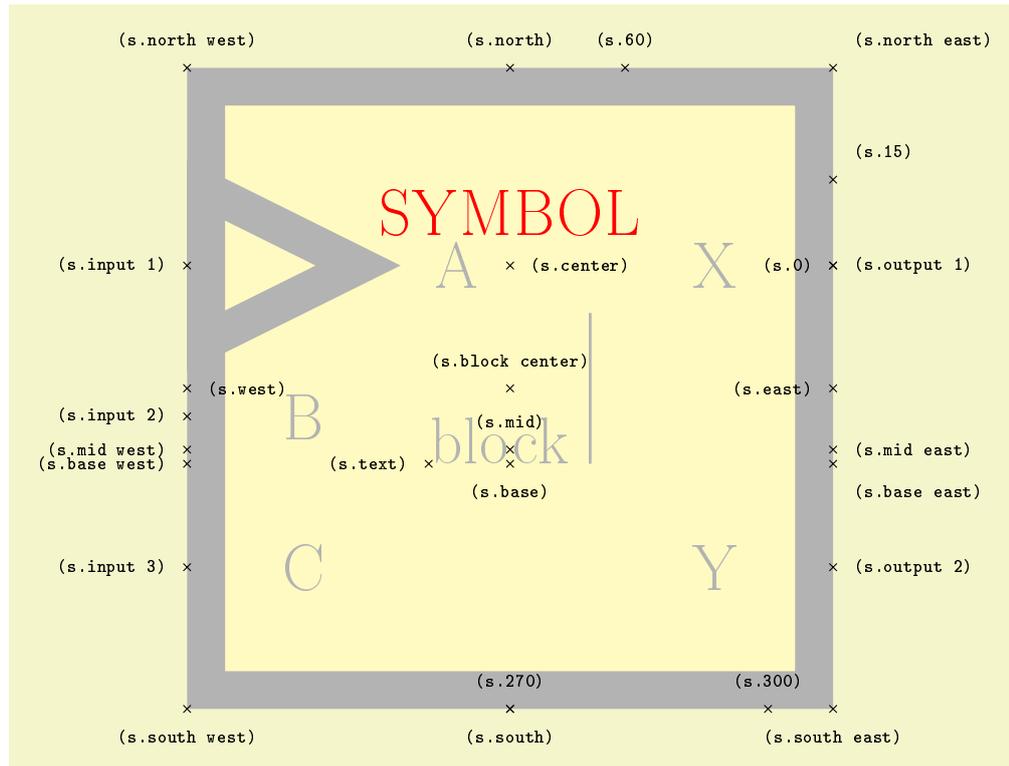
<pre>%Q0.0 ——( )——</pre>	<pre>\tikz[circuit plc ladder,thick] \draw(0,0) to [coil={info={%Q0.0}, coil ladder curvature=0.1}] ++(2,0);</pre>
--------------------------	--

<pre>%Q0.0 ——( )——</pre>	<pre>\tikz[circuit plc ladder,thick] \draw(0,0) to [coil={info={%Q0.0}, coil ladder curvature=0.3}] ++(2,0);</pre>
--------------------------	--

## 7 Blocks

Blocks are used to represent all other features besides contacts and coils, namely functions, methods and function blocks – call representation. The main functions and function blocks are: 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 in the centre of the rectangle. For the exact rectangle centre, use the `block center` anchor.

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 [1].



```
\Huge
\begin{tikzpicture}[circuit plc 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}{SYMBOL}}] {block \vrule width1pt height2cm};
  \foreach \anchor/\placement in {center/right, block center/above, text/left, 0/left, 15/above right, 60/above,
  270/above, 300/above, mid/above, mid east/right, mid west/left, base/below,
  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}
```

In the following subsections, the standard names are employed accordingly to IEC-61 131-3 [1].

### 7.1 Keys for blocks

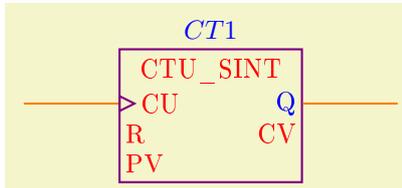
The three most common keys used with blocks are `symbol` which sets the block type, `name` which sets a TikZ label to be used in future references, particularly to access the inputs and outputs, and `info` which sets the variable name associated with the POU (program organization unit) represented by the block. One may also need the standard keys `info'` and `minimum width`. In addition to the keys described in Section “Circuit Libraries” of reference [2], 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)

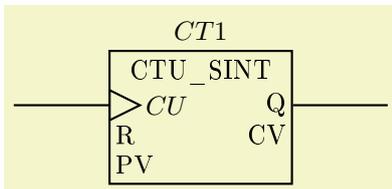
This key sets the colour used for all texts inside the block: symbol, inputs and outputs. In a beamer presentation it can be overridden for a particular input/output by forcing the text colour like this: `{\textcolor{blue}{Q}}`. For other document classes, you need to use a box:



```
\newsavebox{\myeqbox} % only once, preferable in the document preamble
\savebox{\myeqbox}{\textcolor{blue}{Q}}
\tikz[circuit plc ladder,thick] \draw[orange](0,0) to [block={violet,
info={\textcolor{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  $0.8 \backslash tikz\circuitssizeunit$ )

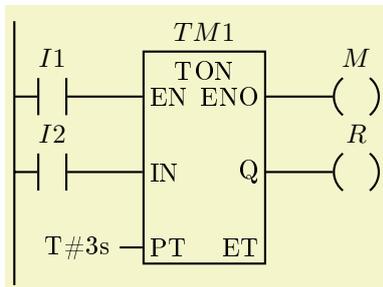
This key sets the size for the clock input indicator (>).



```
\tikz[circuit plc 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  $1.6 \backslash tikz\circuitssizeunit$ )

This key sets the vertical distance between two consecutive inputs.



```
\begin{tikzpicture}[circuit plc ladder,thick,x=1cm,y=1cm]
\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},pos=1}] ++(0.5,0);
\draw(0,-1) to [contact NO={info={I2}}] ++(1,0) -- (TM1.input 2)
(TM1.output 2) -- (4,-1) to [coil={info={R},pos=1}] ++(0.5,0)
(TM1.input 3) -- +(-0.3,0)node[left]{T#3s} (0,1) -- +(0,-3.5);
\end{tikzpicture}
```

Note that, in this example, `x`, `y`, `input sep` and `output sep` are all set to 1 cm. Usually, this is not a good idea and a multiple of `\tikz\circuitssizeunit` should be used instead. Even better, `\ladderskip` should be used in `input sep` and `output sep` because this length is automatically set to `y`. See Section 4 for details.

`/tikz/output sep=<width>` (no default, initially  $1.6 \backslash tikz\circuitssizeunit$ )

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 connections 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 `~` or `{}`. 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 connections 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 `~` or `{}`. The minimal number of outputs is one, an empty list of outputs generates an error.

## 7.2 Timers

The standard IEC-61 131-3 [1, p. 114] specifies three timers as follows:

On-delay:



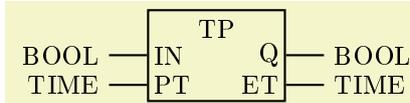
```
\node[block,inputs={IN,PT},outputs={Q,ET},
symbol=TON,minimum width=18mm] (tp1) {};
\draw (tp1.input 1) -- +(-5mm,0) node[left]{BOOL}
(tp1.input 2) -- +(-5mm,0) node[left]{TIME}
(tp1.output 1) -- +(5mm,0) node[right]{BOOL}
(tp1.output 2) -- +(5mm,0) node[right]{TIME};
```

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:



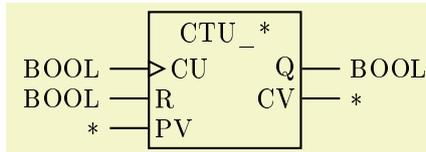
```
\node[block,inputs={IN,PT},outputs={Q,ET},
symbol=TP,minimum width=18mm] (tp1) {};
\draw (tp1.input 1) -- +(-5mm,0) node[left]{BOOL}
(tp1.input 2) -- +(-5mm,0) node[left]{TIME}
(tp1.output 1) -- +(5mm,0) node[right]{BOOL}
(tp1.output 2) -- +(5mm,0) node[right]{TIME};
```

## 7.3 Counters

The clock input is indicated by the character > which needs 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 declared as `inputs={EN,>CU,>CD,R,LD,PV}`.

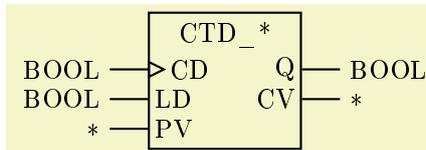
The standard IEC-61 131-3 [1, p. 113] specifies three counters. In the following examples, the symbol “\*” indicates the numerical type of the counter (like INT, DINT, etc.).

Up-Counter:



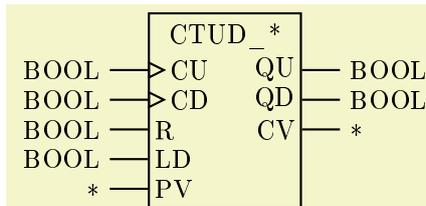
```
\node[block,inputs={>CU,R,PV},outputs={Q,CV},
symbol=CTU|_*,minimum width=20mm] (ct1) {};
\draw (ct1.input 1) -- +(-5mm,0) node[left]{BOOL}
(ct1.input 2) -- +(-5mm,0) node[left]{BOOL}
(ct1.input 3) -- +(-5mm,0) node[left,yshift=-0.4ex]{*}
(ct1.output 1) -- +(5mm,0) node[right]{BOOL}
(ct1.output 2)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

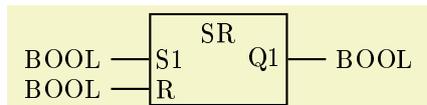
Up-down counters:



```
\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]{*};
```

## 7.4 Standard bistable function blocks

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



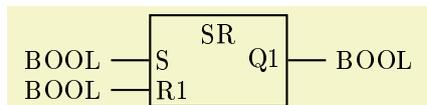
```
\node[block,inputs={S1,R},outputs={Q1},
symbol=SR,minimum width=18mm] (sr1) {};
\draw (sr1.input 1) -- +(-5mm,0) node[left]{BOOL}
(sr1.input 2) -- +(-5mm,0) node[left]{BOOL}
(sr1.output 1) -- +(5mm,0) node[right]{BOOL};
```

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<sup>6</sup>: 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};
```

## 7.5 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)



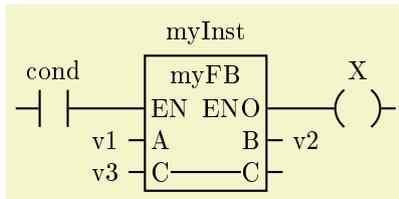
```
\node[block,inputs={CLK},outputs={Q},
symbol=F_TRIG,minimum width=20mm] (ed1) {};
\draw (ed1.input 1) -- +(-5mm,0) node[left]{BOOL}
(ed1.output 1) -- +(5mm,0) node[right]{BOOL};
```

## 7.6 Call representation

A call is used to execute a function, a function block instance or a method of a function block or class [1, sic, p. 60]. They are represented by blocks.

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

<sup>6</sup>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 [1, p. 112] was kept unchanged.



```

\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) -- +(-0.2,0) node[left]{v1}
      (mf1.input 3) -- +(-0.2,0) node[left]{v3}
      (mf1.output 2) -- +(0.2,0) node[right]{v2}
      (mf1.output 3) -- +(0.2,0)
      ($ (mf1.input 3) + (1em,0)$) -- ($ (mf1.output 3) - (1em,0)$);

```

The same effect can be obtained replacing the last code line by `([xshift=1em]mf1.input 3) -- ([xshift=-1em]mf1.output 3)`; but this time, the anchors are displaced by the command `xshift`. If, however, 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 the last code line by `(p1) -- (p2)`;

A lot of things can go wrong here. If, for instance, the `VAR_IN_OUT` label is modified or the `inner sep` is changed, you might need to modify the amount of space left for the labels on both sides.

## 7.7 Directly represented variables (%)

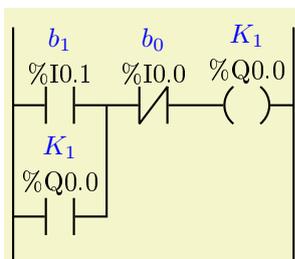
Direct representation of a single-element variable shall be provided by a special symbol formed by the concatenation of [1, sic, p. 54]:

- a per cent sign “%” and
- location prefixes I, Q or M and
- a size prefix X (or none), B, W, D or L and
- one or more (hierarchical addressing) unsigned integers that shall be separated by periods “.”.

EXAMPLES:

- %MW1.7.9
- %ID12.6
- %QL20

The character % must be preceded by a backslash “\%”. It is also possible to place an illustrative text along with the `info`, usually above it. The user-friendly variable names are placed in blue above each directly represented variable in the following example. The label distance is set globally to `-3pt` to save space. Even so, the distance between rows has to be increased to make room for the labels. So, `y` is set to `6\tikzcircuitssizeunit` or 20% above its default value.



```

\begin{tikzpicture} [circuit plc ladder,thick,
  label distance=-3pt,y=6\tikzcircuitssizeunit]
\draw(0,0)
  to [contact NO={info={label={blue}$b_1$}}\%IO.1}] ++(1,0)
  coordinate(laddercoil)
  to [contact NC={info={label={blue}$b_0$}}\%IO.0}] ++(1,0)
  to [coil={info={label={blue}$K_1$}}\%Q0.0}] ++(1,0)
  coordinate(laddertoprigh);
\draw(0,-1)
  to [contact NO={info={label={blue}$K_1$}}\%Q0.0}] ++(1,0)
  -- (laddercoil);
\ladderrungend{2}
\ladderpowerrails
\end{tikzpicture}

```

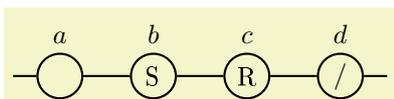
## 8 NEMA

It is possible to typeset NEMA – National Electrical Manufacturers Association – standard diagrams using this package. The main difference is that coils are represented by  $\text{---}\bigcirc\text{---}$  instead of  $\text{---}(\text{---})\text{---}$ .

You can replace all coil symbols for circles globally placing the following keys in the document preamble:

```
\tikzset{
  coil NA/.style={coil=#1,symbol={$/\$/}},
  every coil/.style={minimum size=2.4\tikzcircuitssizeunit,coil ladder curvature=0.5},
  every coil S/.style={minimum size=2.4\tikzcircuitssizeunit,coil ladder curvature=0.5},
  every coil R/.style={minimum size=2.4\tikzcircuitssizeunit,coil ladder curvature=0.5},
  every coil NA/.style={minimum size=2.4\tikzcircuitssizeunit,coil ladder curvature=0.5}
}
```

That would produce:



```
\begin{tikzpicture} [circuit plc ladder,thick]
\draw(0,0)
  to [coil={info={$a$}}] ++(1,0)
  to [coil S={info={$b$}}] ++(1,0)
  to [coil R={info={$c$}}] ++(1,0)
  to [coil NA={info={$d$}}] ++(1,0);
\end{tikzpicture}
```

If, however, you have lots of diagrams you better of redefining at least the `coil graphic` which is likely the most used one (not to say the only one). It is because the algorithm used to draw coils uses arcs, which are less efficient than circles. Thus, place this key in your document preamble:

```
\tikzset{
  set coil graphic={
    circuit symbol lines,
    fill=none,
    circuit symbol size=width 2.4 height 2.4,
    transform shape,
    shape=circle ee,
    node contents=\pgfkeysvalueof{/pgf/ladder symbol}
  }
}
```

## 9 Known Issues

The `center` anchor of the block symbol is not in the centre and it is disturbing, to say the least. This is due `TikZ` positioning algorithm, which is unlike to change.

The way the library was written is not exactly like the standard `TikZ` libraries. It seems that the official libraries are written in two distinct files: one for `TikZ` stuff and another for `PGF`, but I don't know how to separate it, thus we have a single file (at least for now).

## 10 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<sup>7</sup>, therefore idiosyncrasies were not detected.

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

It should be great if someone with experience in writing `TikZ` libraries could have a look at the code and point out errors 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.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.

<sup>7</sup>Not entirely true, a total of three people asked me about the package.

## 11 Acknowledgement

I would like to thank Martin Fabian with the Chalmers University of Technology for the strong encouragement to rewrite Section 4 and provide more examples.

## References

- [1] **International Electrotechnical Commission.** *IEC 61131-3:2013 – Programmable controllers - Part 3: Programming languages.* Geneva, 2013.
- [2] **Tantau,** Till. *The TikZ and PGF Packages* : Manual for version 3.1.9a-34-ga0d9cada, <https://github.com/pgf-tikz/pgf>, 2021.