Typesetting a Process and Instrumentation Diagram with LaTeX and Ti*k*Z

Jelle Spijker
e-mail: spijker.jelle@gmail.com

Version: 1.0, Version date: 02/11/2018

1 Abstract

This package contains a collection of symbols, intended to be used with a *Process & Instrumentation Diagram*. The symbols are meant to be in agreement with international standard ISO-14617. It extends and modifies, when needed, the Ti*k*Z-library `circuits`. A drafting class is also included, which allows a Ti*k*Z-library `circuits.pid` diagrams, to be presented as a standalone document.

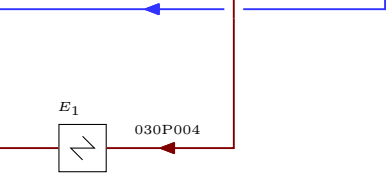
2 Introduction

P&IDs are used in a variety a industries, to design and illustrate the workings of various processes. The purpose of ISO 14617 in its final form is the creation of a library of harmonized graphical symbols for diagrams used in technical applications. Their work has been, and will be, performed in close cooperation between ISO and IEC. The ultimate result is intended to be published as a standard common to ISO and IEC, which their technical committees responsible for specific application fields can use in preparing International Standards and manuals. This Ti*k*Z-library is a translation of these norms, such that the can be used in LaTeXdocuments.

The author has access to the current ISO version, via his employer **Royal IHC**, these norms were bought by them and cannot be redistribute to the general public. However all individual ISO-14617 symbols can also be found at ISO.org, just search for 14617 and go to the graphical symbols tab.

The following sub norms are used for this Ti*k*Z-library:

- 14617-1 General information and indexes
- 14617-2 Symbols having general application
- 14617-3 Connections and related devices
- 14617-4 Actuators and related devices
- 14617-5 Measurement and control devices
- 14617-6 Measurement and control functions
- 14617-7 Basic mechanical components
- 14617-8 Valves and dampers
- 14617-9 Pumps, compressors and fans
- 14617-10 Fluid power converters
- 14617-11 Devices for heat transfer and heat engines
- 14617-12 Devices for separating, purification and mixing
- 14617-13 Devices for material processing
- 14617-14 Devices for transport and handling of material
- 14617-15 Installation diagrams and network maps



3 Example

```
\begin{tikzpicture}[circuit pid ISO14617, every info/.style={black, font=\tiny}]
\begin{complex device}[name=solar absorption cooler, xshift=-160mm, yshift=100mm, show
border, show label]
\coordinate (solar absorption cooler base) at (0,0);
\node[envelope={name=SC1, light, quantity transition to higher, info={[xshift=-2, yshift=-10]T}
info'={[xshift=10, yshift=30]\( SC_{1} \)}}] at (solar absorption cooler base) {};
\node[pressure vessel={name=T1, info={[yshift=-15]right:\( T_{1} \)}, rotate=90, anchor=north,
circuit symbol size=width 5 height 3}] at ($(SC1.output)+(\d,0)$) {};
\node[pump={name=P1, displacement, info=\( P_{1} \)}] at ($(SC1)!0.5!(T1)-(0,\d)$) {};
\node[envelope={name=G1, with={heating coil}{0}{0},info={[xshift=10]\( G_{1} \)}}] at ($(T1)+(2
\node[pump={name=P2, displacement, info=\( P_{2} \)}] at ($(T1)!0.5!(G1)+(0,\d)$) {};
\node[envelope={name=R1, device for separating, info={[yshift=-15]CH}, info'=above:\( R_{1} \)}
at ($(G1)+(\d,0)$) {};
\node[envelope={name=C1, with={heating coil}{0}{0},info={[xshift=-5]\( C_{1} \)}}] at ($(R1)+(\
\node[valve={name=V1, info={\( V_{1} \)}, rotate=90}] at ($(C1.south)-(-\d,2*\d)$) {};
\node[envelope={name=E1, with={heating coil}{0}{0},info={[xshift=-5]\( E_{1} \)}}] at ($(V1)-(\
\node[cooling tower={name=CT1, with={spray nozzle}{0}{2.5}, info=\( CT_{1} \)}] at ($(C1)+(2*\d
\node[pump={name=P3, displacement, rotate=-90, info={\( P_{3} \)}}] at (CT1 |- V1) {};
\node[coil shaped heat exchanger={name=H1, rotate=90, info=\( H_{1} \), huge circuit symbols},
anchor=refrigerant in] at ($(G1)-(0,\d+0.125*\d)$) {};
\node[pump={name=P4, displacement, info=\( P_{4} \), rotate=90}] at (V1 -| H1.input) {};
\node[valve={name=V2, info=\( V_{2} \), rotate=90}] at (P4 -| H1.refrigerant in) {};
\node[envelope={name=A1, with={heating coil}{0}{0}, info={[xshift=5]\( A_{1} \)}}] at ($(V2)-(0

% === Solar refrigerant system ===
\begin{flow system}[color=red!80!white, id=010]
\draw[flow path, id=P001, show'] (T1.north) to (SC1);
\draw[flow path, id=P002, show, pos=0.8] (SC1) to (P1 -| SC1) to (P1);
\draw[flow path, id=P003, show, pos=0.2] (P1) to (P1 -| T1) to (T1.west);
\draw[flow path, id=P004, show', pos=0.85] (G1-heating coil.south) to ($(G1-heating coil.south
\draw[flow path, id=P005, show, pos=0.75] (T1) to (T1 |- P2) to (P2);
\draw[flow path, id=P006, show, pos=0.3] (P2) to (P2 -| G1) to (G1-heating coil.north);
\end{flow system}

% == Cooling water system ===
\begin{flow system}[id=020, color=blue!80]
\draw[flow path, id=P001, show] (C1-heating coil.north) to (CT1-spray nozzle.north -| C1)
to (CT1-spray nozzle.north);
\draw[flow path, id=P002, show'] (CT1.south) to (P3);
\draw[flow path, id=P003, show'] (P3) to (P3 |- A1-heating coil.south) to (A1-heating coil.sout
\draw[flow path, id=P004, show, pos=0.3] (A1-heating coil.north) to (A1-heating coil.north -| C
to (C1-heating coil.south);
\end{flow system}

% === NH3 refrigerant
\begin{flow system}[id=030, color=red!50!black]
\draw[flow path, id=P001, show] (R1) to (C1);
\draw[flow path, id=P002, show', pos=0.75] (C1) to (C1 -| V1) to (V1);
\draw[flow path, id=P004, show', pos=0.85] (V1) to (V1 |- E1) to (E1);
\draw[flow path, id=P005, show'] (E1) to (E1 -| A1) to (A1);
\crossings{020P003}{030P004}
\end{flow system}

% === NH3-H20 refrigerant
\begin{flow system}[id=040, color=green!50!blue]
\draw[flow path, id=P001, show] (G1) to (R1);
\draw[flow path, id=P002, show'] (R1) to ++(0,-0.75\d) to ([xshift=5]\currentcoordinate -| G1)
to ([xshift=5]G1.south);
\draw[flow path, id=P003, show'] (H1.refrigerant in) to (V2);
\draw[flow path, id=P004, show'] (V2) to (A1);
\draw[flow path, id=P005, show, pos=0.75] (A1) to (A1 -| P4) to (P4);
\draw[flow path, id=P006, show] (P4) to (H1.input);
\draw[flow path, id=P007, show] (H1.output) to (G1.south -| H1.output);
\draw[flow path, id=P008, show', pos=0.8] (G1) to (G1 -| H1.refrigerant out) to (H1.refrigerant
\crossings{010P004}{040P008};
\end{flow system}
\end{complex device}
\end{tikzpicture}
```

4 P&ID library

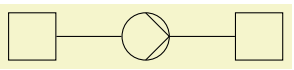
The `circuit pid` libraries can be used to draw different kind of *Process & Instrumentation Diagrams*. It is not a single library, but a hierarchy of libraries that work in concert. The main goal was to convert the ISO-14617 symbols and drawing conventions to a LaTeX/ Ti*k*Z environment. This library is setup in such a way that it can easily be extended with different drawing conventions and norms.

A P&ID circuit typically consist of numerous elements, such as pumps, valves, heaters, actuators, sensors, tanks etc; which are connected through a flow of a certain quantity, usually: mass, volume, electrical charge or information. In PGF/Ti*k*Z, we use nodes for these elements and normal lines for the flow of a quantity. Ti*k*Z offers a large number of different ways of positioning and connecting nodes in general, all of which can be used here. Additionally, the `circuits.pid` library defines additional useful commands, such as `to`-path, `with`-node and `at`-node. Those are useful for elements such as actuators en sensors, or for the combination of various nodes configurations.

A simple example shows how a P&ID could be created. A user has to decide which symbol graphics he would like to use. Which only option (at the time) is the ISO-14617 norm at the moment. By include the library `circuits.pid.ISO14617` in the preamble of his document.

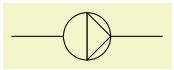
```
\usetikzlibrary{circuits.pid.ISO14617}
```

To create a Ti*k*Z-picture illustrating the process of pumping a fluid between two tanks the following code suffices.



```
\begin{tikzpicture}[circuit pid ISO14617]
\node[tank={name=T1}] at (0,0) {};
\node[tank={name=T2}] at (3,0) {};
\draw (T1) to [pump] (T2);
\end{tikzpicture}
```

The example above can be extended with the use of *annotations*, these are extensions for nodes, that give supplementary information. The pump used in the previous example can easily desribed as a displacement pump, with the use of annotations.



```
\begin{tikzpicture}[circuit pid ISO14617]
\draw (0,0) to [pump=displacement] (2,0);
\end{tikzpicture}
```

4.1 The Base Circuit Library

The following sections is directly copied out of the PGF/Ti*k*Z-manual, such that this manual can be used as a stand-alone document. The code and examples are converted from electrical type circuits to P&ID circuits.

Ti*k*Z Library `circuits`

```
\usetikzlibrary{circuits} % LaTeX and plain TeX
\usetikzlibrary[circuits] % ConTeXt
```

This library is a base library that is included by other circuit libraries. You do not include it directly, but you will typically use some of the general keys, described below.

`/tikz/circuits` (no value)

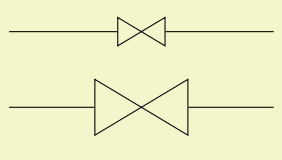
This key should be passed as an option to a picture or a scope that contains a circuit. It will do some internal setups. This key is normally called by more specialized keys like `circuit ee IEC` or `circuit pid ISO14617`.

4.1.1 Symbol Size

`/tikz/circuit symbol unit`=⟨*dimension*⟩ (no default, initially `7pt`)

This dimension is a "unit" for the size of symbols. The libraries generally define the sizes of symbols relative to this dimension. For instance, the symbol of a pump, by default, in the ISO14617 library equal to two and a half times this ⟨*dimension*⟩. When you change this ⟨*dimension*⟩, the size of all symbols will automatically change accordingly.

Note, that it is still possible to overwrite the size of any particular symbol. These settings apply only to the default sizes.



```
\begin{tikzpicture}[circuit pid ISO14617]
\draw (0,1) to [valve] (3.5,1);
\draw[circuit symbol unit=14pt]
(0,0) to [valve] (3.5,0);
\end{tikzpicture}
```

/tikz/huge circuit symbols (style, no value)

This style sets the default circuit symbol unit to `10pt`.

/tikz/large circuit symbols (style, no value)

This style sets the default circuit symbol unit to `8pt`.

/tikz/medium circuit symbols (style, no value)

This style sets the default circuit symbol unit to `7pt`.

/tikz/small circuit symbols (style, no value)

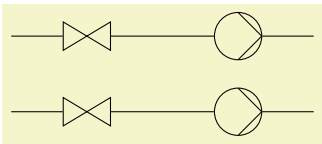
This style sets the default circuit symbol unit to `6pt`.

/tikz/tiny circuit symbols (style, no value)

This style sets the default circuit symbol unit to `5pt`.

/tikz/circuit symbol size=`width` ⟨*width*⟩ `height` ⟨*height*⟩ (no default)

This key sets `minimum height` to ⟨*height*⟩ times the current value of the circuit symbol unit and the `minimum width` to ⟨*width*⟩ times this value. Thus, this option can be used with a node command to set the size of the node as a multiple of the circuit symbol unit.



```
\begin{tikzpicture}[circuit pid ISO14617]
\draw (0,1) to [valve] (2,1) to[pump] (4,1);

\begin{scope}
[every valve/.style={circuit symbol size=width 4 height 1}]
\draw (0,0) to [valve] (2,0) to[pump] (4,0);
\end{scope}
\end{tikzpicture}
```

4.1.2 Declaring New Symbols

/tikz/circuit declare symbol=⟨*name*⟩ (no default)

This key is used to declare a symbol. It does not cause this symbol to be shown nor does it set a graphic to be used for the symbol, it simply "prepares" several keys that can later be used to draw a symbol and to configure it.

In detail, the first key that is defined is just called ⟨*name*⟩. This key should be given as an option to a `node` or on a `to` path, as explained below. The key will take options, which can be used to influence the way the symbol graphic is rendered.

Let us have a look at an example. Suppose we want to define a symbol called `foo`, which just looks like a simple rectangle. We could then say

```
\tikzset{circuit declare symbol=foo}
```

The symbol could now be used like this:

```
\node [foo]       at (1,1) {};
\node [foo={red}] at (2,1) {};
```

However, in the above example we would not actually see anything since we have not yet set up the graphic to be used by `foo`. For this, we must use a key called `set foo graphic` or, generally, `set` ⟨*name*⟩ `graphic`. This key gets graphic options as parameter that will be set when a symbol `foo` should be shown:

```
\begin{tikzpicture}
[circuit pid ISO14617,
circuit declare symbol=foo,
set foo graphic={draw,shape=rectangle,minimum size=5mm}]

\node [foo]       at (1,1) {};
\node [foo={red}] at (2,1) {};
\end{tikzpicture}
```

In detail, when you use the key ⟨*name*⟩=⟨*options*⟩ with a node, the following happens:

1. The `inner sep` is set to `0.5pt`.
2. The following style is executed:

   `/tikz/every circuit symbol` (style, no value)

   Use this style to set up things in general.
3. The graphic options that have been set using `set` ⟨*name*⟩ `graphic` are set.
4. The style `every` ⟨*name*⟩ is executed. You can use it to configure the symbol further.
5. The ⟨*options*⟩ are executed.

The key ⟨*name*⟩ will have a different effect when it is used on a `to` path command inside a `circuit` environment (the `circuit` environment sets up `to` paths in such a way that the use of a key declared using `circuit declare symbol` is automatically detected). When ⟨*name*⟩ is used on a `to` path, the above actions also happen (setting the inner separation, using the symbol graphic, and so on), but they are passed to the key `circuit handle symbol`, which is explained next.

`/tikz/circuit handle symbol`=⟨*options*⟩ (no default)

This key is mostly used internally. Its purpose is to render a symbol. The effect of this key differs, depending on whether it is used as the optional argument of a `to` path command or elsewhere.

If the key is not used as an argument of a `to` path command, the ⟨*options*⟩ are simply executed.

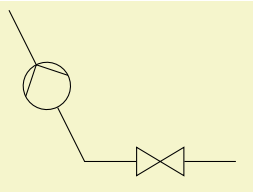
The more interesting case happens when the key is given on a `to` path command. In this case, several things happen:

1. The `to` path is locally changed and set to an internal path (which you should not try to change) that consists mostly of a single straight line.
2. The ⟨*options*⟩ are tentatively executed with filtering switched on. Everything is filtered out, except for the key `pos` and also the styles `at start`, `very near start`, `near start`, `midway`, `near end`, `very near end`, and `at end`. If none of them is found, `midway` is used.
3. The filtered option is used to determine a position for the symbol on the path. At the given position (with `pos=0` representing the start and `pos=1` representing the end), a node will be added to the path (in a manner to be described presently).
4. This node gets ⟨*options*⟩ as its option list.
5. The node is added by virtue of a special `markings` decoration. This means that a `mark` command is executed that causes the node to be placed as a mark on the path.
6. The marking decoration will automatically subdivide the path and cause a line to be drawn from the start of the path to the node's border (at the position that lies on a line from the node's center to the start of the path) and then from the node's border (at a position on the other side of the node) to the end of the path.
7. The marking decoration will also take care of the case that multiple marks are present on a path, in this case the lines from and to the borders of the nodes are only between consecutive nodes.
8. The marking decoration will also rotate the coordinate system in such a way that the x-axis points along the path. Thus, if you use the `transform shape` option, the node will "point along" the path.
9. In case a node is at `pos=0` or at `pos=1` some special code will suppress the superfluous lines to the start or end of the path.

4.1.3 Pointing Symbols in the Right Direction

Unlike normal nodes, which generally should not be rotated since this will make their text hard to read, symbols often need to be rotated. There are two ways of achieving such rotations:

1. When you place a symbol on a `to` path, the graphic symbol is automatically rotated such that it "points along the path." Here is an examples that shows how the pump shape is automatically rotated around:



```
\tikz [circuit pid ISO14617]
\draw (3,0) to[valve] (1,0) to[pump] (0,2);
```

2. Many shapes cannot be placed "on" a path in this way, namely whenever there are more than two possible inputs. Also, you may wish to place the nodes first, possibly using a matrix, and connect them afterwards. In this case, you can simply add rotations like `rotate=90` to the shapes to rotate them. The following four keys make this slightly more convenient:

`/tikz/point up` (no value)

This is the same as `rotate=90`.

```
\tikz [circuit pid ISO14617] \node [pump,point up] {};
```

`/tikz/point down` (no value)

This is the same as `rotate=-90`.

```
\tikz [circuit pid ISO14617] \node [pump,point down] {};
```

`/tikz/point left` (no value)

This is the same as `rotate=-180`.

```
\tikz [circuit pid ISO14617] \node [pump,point left] {};
```

`/tikz/point right` (no value)

This key has no effect.

```
\tikz [circuit pid ISO14617] \node [pump,point right] {};
```

4.1.4 Info Labels

Info labels are used to add text to a circuit symbol. Unlike normal nodes like a rectangle, circuit symbols typically do not have text "on" them, but the text is placed next to them (like the text "$3\,\mathrm{m\,s^{-1}}$" next to a pump).

TikZ already provides the `label` option for this purpose. The `info` option is built on top of this option, but it comes in some predefined variants that are especially useful in conjunction with circuits.

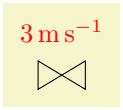
`/tikz/info=[`⟨*options*⟩`]`⟨*angle*⟩`:`⟨*text*⟩ (no default)

This key has nearly the same effect as the `label` key, only the following style is used additionally automatically:

/tikz/every info (style, no value)

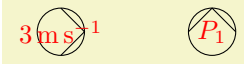
Set this style to configure the styling of info labels. Since this key is *not* used with normal labels, it provides an easy way of changing the way info labels look without changing other labels.

The ⟨*options*⟩ and ⟨*angle*⟩ are passed directly to the `label` command.




```
\begin{tikzpicture}[circuit pid ISO14617, every info/.style=red]
\node [valve,info=$\SI{3}{\meter\per\second}$] {};
\end{tikzpicture}
```

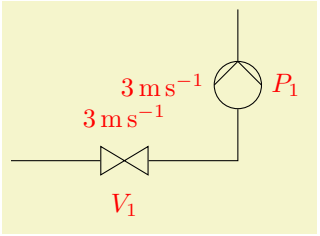
Hint: To place some text *on* the main node, use `center` as the ⟨*angle*⟩:




```
\begin{tikzpicture}[circuit pid ISO14617,every info/.style=red]
\node [pump,info=center:$\SI{3}{\meter\per\second}$] {};
\node [pump,point up,info=center:$P_1$] at (2,0) {};
\end{tikzpicture}
```

/tikz/info'=[⟨*options*⟩]⟨*angle*⟩:⟨*text*⟩ (no default)

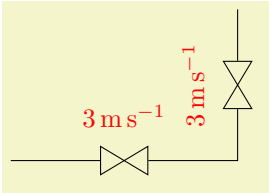
This key works exactly like the `info` key, only in case the ⟨*angle*⟩ is missing, it defaults to `below` instead of the current value of `label position`, which is usually `above`. This means that when you use `info`, you get a label above the node, while when you use the `info'` key you get a label below the node. In case the node has been rotated, the positions of the info nodes are rotated accordingly.




```
\begin{tikzpicture}[circuit pid ISO14617,every info/.style=red]
\draw (0,0) to[valve={info={$\SI{3}{\meter\per\second}$},info'={$V_1$}}] (3,0)
to[pump={info={$\SI{3}{\meter\per\second}$},info'={$P_1$}}] (3,2);
\end{tikzpicture}
```

/tikz/info sloped=[⟨*options*⟩]⟨*angle*⟩:⟨*text*⟩ (no default)

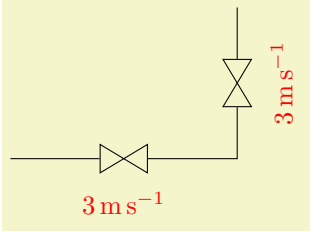
This key works like `info`, only the `transform shape` option is set when the label is drawn, causing it to follow the sloping of the main node.




```
\begin{tikzpicture}[circuit pid ISO14617,every info/.style=red]
\draw (0,0) to[valve={info sloped={$\SI{3}{\meter\per\second}$}}] (3,0)
to[valve={info sloped={$\SI{3}{\meter\per\second}$}}] (3,2);
\end{tikzpicture}
```

/tikz/info' sloped= (no default)

This is a combination of `info'` and `info sloped`.



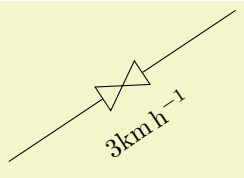

```
\begin{tikzpicture}[circuit pid ISO14617,every info/.style=red]
\draw (0,0) to[valve={info' sloped={$\SI{3}{\meter\per\second}$}}] (3,0)
to[valve={info' sloped={$\SI{3}{\meter\per\second}$}}] (3,2);
\end{tikzpicture}
```

/tikz/circuit declare unit={⟨*name*⟩}{⟨*unit*⟩} (no default)

This key is used to declare keys that make it easy to attach physical units to nodes. The idea is that instead of `info=$\SI{3}{\meter\per\second}$` you can write `speed=3` or instead of

info'=$\SI{1}{\cubic\meter\per\second}$ you can write volume flow'=5. Although units can be described as $\frac{m^3}{s}$ it is good practice to make us of the siunitx package.

In detail, four keys are defined, namely /tikz/⟨*name*⟩, /tikz/⟨*name*⟩', /tikz/⟨*name*⟩ sloped, and /tikz/⟨*name*⟩' sloped. The arguments of all of these keys are of the form [⟨*options*⟩]⟨*angle*⟩:⟨*value*⟩ and it is passed (slightly modified) to the corresponding key info, info', info sloped, or info' sloped. The "slight modification" is the following: The text that is passed to the, say, info key is not ⟨*value*⟩, but rather $\mathrm{⟨*value*⟩⟨*unit*⟩}$ This means that after you said circuit declare unit={*speed*}{\si{\meter\per\second}}, then speed=5 will have the same effect as info={[every ohm]$\mathrm{5\si{\meter\per\second}}$}. Here, every speed is a style that allows you to configure the appearance of this unit. Since the info key is used internally, by changing the every info style, you can change the appearance of all units infos.




```
\begin{tikzpicture}[circuit pid ISO14617,
circuit declare unit={my speed}{\si{\kilo\meter\per\hour}}]
\draw (0,0) to[valve={my speed' sloped=3}] (3,2);
\end{tikzpicture}
```

The circuit.pid library predefines the following unit keys:

Key	*Appearance of* 1 *unit*
/tikz/speed	$1\,\mathrm{m\,s^{-1}}$
/tikz/mass flow	$1\,\mathrm{kg\,s^{-1}}$
/tikz/volume flow	$1\,\mathrm{m^3\,s^{-1}}$

4.2 The P&ID library

TikZ Library circuits.pid

```
\usetikzlibrary{circuits.pid} % LaTeX and plain TeX
\usetikzlibrary[circuits.pid] % ConTeXt
```

This library defines general keys for the creation of *Process & Instrumentation Diagrams*. This library is written as an extension of TikZ-library circuits. It allows for the creation of new type of circuit diagram, using the same syntax and conventions established by TikZ-library circuits. The reader is urged to read the Section "Circuit Libraries" of TikZ manual.

The following keys are defined:

/tikz/circuit pid (no value)

This key should be passed as an option to a picture or a scope that contains a diagram. It will do some internal setups. At the time of writing only ISO-14617 P&ID are implemented. You normally do not use this library directly since it does not define any symbol graphics; This is done with the sublibrary circuits.pid.ISO14617

/tikz/measure=⟨*options*⟩ (no default)

A

Symbol		Measured or initiating variable	Modifier	Function
1051	A			alarming
1052	B			displaying discrete state
1053	C			controlling
1054	D	density	difference	
1055	E	electric variable		sensing
1056	F	flow rate	ratio, fraction	
1057	G	gauge, position, length		viewing
1058	H	hand		
1059	I			indicating
1060	J	power	scanning	
1061	K	time	time rate of change	
1062	L	level		
1063	M	moisture, humidity	momentarily	
1064	N	user's choice		user's choice
1065	O	user's choice		
1066	P	pressure, vacuum		connection of test point
1067	Q	quality	integral, total	integrating, summing
1068	R	radiation		registering, recording
1069	S	speed, frequency		switching
1070	T	temperature		transmitting
1071	U	multi-variable		multi-function
1072	V	user's choice		impact on process by valve, pump, etc.
1073	W	weight, force	Multiplying	
1074	X	unclassified		unclassified
1075	Y	user's choice		converting, computing
1076	Z	number of events, quantity		emergency or safety acting

/tikz/at (no value)

test

/tikz/with (no value)

test

Decoration /tikz/flow path

test

/tikz/id (no value)

test

/tikz/show (no value)

test

/tikz/show' (no value)

test

/tikz/pos (no value)

test

/tikz/pos slope (no value)

test

4.3 Implementation: The P&ID-symbols shape library

TikZ Library shapes.gates.pid

```
\usepgflibrary{shapes.gates.pid}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[shapes.gates.pid]  % ConTeXt and pure pgf
\usetikzlibrary{shapes.gates.pid} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[shapes.gates.pid] % ConTeXt when using TikZ
```

This library defines basic shapes that can be used by all pid-circuit libraries. Currently, it defines the following shapes:

- rectangle pid
- circle pid
- direction pid

Additionally, the library defines the following arrow tip: The direction pid arrow tip is basically the same as a `triangle 45` arrow tip with rounded joins.

`direction pid` yields thick ⟷ and thin ⟷

However, unlike normal arrow tips, its size does *not* depend on the current line width. Rather, it depends on the value of its arrow options, which should be set to the desired size. Thus, you should say something like `\pgfsetarrowoptions{direction pid}{5pt}` to set the size of the arrow.

Shape rectangle pid

This shape is completely identical to a normal `rectangle`, only there are two additional anchors: The `input` anchor is an alias for the `west` anchor, while the `output` anchor is an alias for the `east` anchor.

Shape circle pid

Like the rectangle pid shape, only for circles.

Shape direction pid

This shape is rather special. It is intended to be used to "turn an arrow tip into a shape." First, you should set the following key to the name of an arrow tip:

/pgf/direction pid arrow=⟨*right arrow tip name*⟩ (no default)

The value of this key will be used for the arrow tip depicted in an direction pid shape.

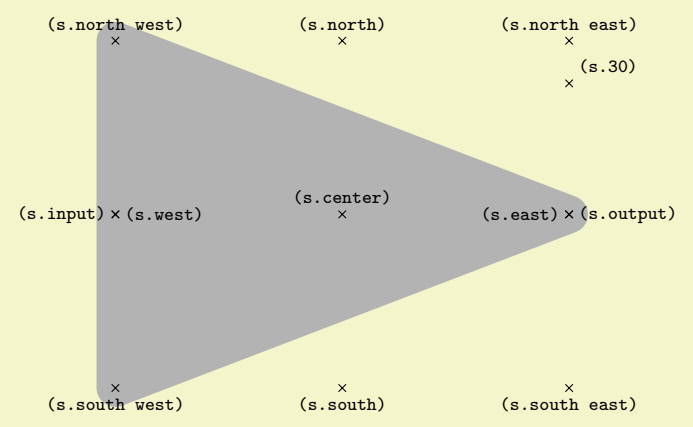
When a node of shape direction pid is created, several things happen:

1. The size of the shape is computed according to the following rules: The width of the shape is set up so that the left border of the shape is at the left end of the arrow tip and the right border is at the right end of the arrow tip. These left and right "ends" of the arrow are the tip end and the back end specified by the arrow itself. You usually need not worry about this width setting.

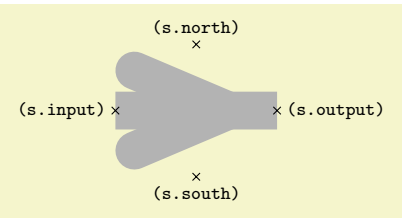
   By comparison, the height of the arrow is given by the current setting of `minimum height`. Thus, this key must have been set up correctly to reflect the "real" height of the arrow tip. The reason is that the height of an arrow is not specified when arrows are declared and is, thus, not available, here.

   Possibly, the height computation will change in the future to reflect the real height of the arrow, so you should generally set up the `minimum height` to be the same as the real height.
2. A straight line from left to right inside the shape's boundaries is added to the background path.
3. The arrow tip, pointing right, is drawn before the background path.

The anchors of this shape are just the compass anchors, which lie on a rectangle whose width and height are the above-computed height and width.




```
\begin{tikzpicture}
\pgfsetarrowoptions{direction pid}{6cm}
\node[name=s,shape=direction pid,shape example,minimum height=0.7654*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}
```




```
\begin{tikzpicture}[direction pid arrow=angle 45]
\node[name=s,shape=direction pid,shape example,minimum height=1.75cm] {};
\foreach \anchor/\placement in {north/above, south/below,
output/right, input/left}
\draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

4.4 Implementation: The ISO-14617 style P&ID-symbol shape library

TikZ Library shapes.gates.pid.ISO14617

```
\usepgflibrary{shapes.gates.pid.ISO14617}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[shapes.gates.pid.ISO14617]  % ConTeXt and pure pgf
\usetikzlibrary{shapes.gates.pid.ISO14617} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[shapes.gates.pid.ISO14617] % ConTeXt when using TikZ
```

This library defines shapes for depicting pid symbols according to the ISO14617 recommendations. These shapes will typically be used in conjunction with the graphic mechanism detailed earlier, but you can also used them directly.

Shape generic circle ISO14617

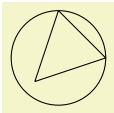
This shape inherits from circle pid, which in turn is just a normal `circle` with additional `input` and `output` anchors at the left and right ends. However, additionally, this shape allows you to specify a path that should be added before the background path using the following key:

/pgf/generic circle ISO14617/before background=⟨*code*⟩ (no default)

When a node of shape generic circle ISO14617 is created, the current setting of this key is used as the "before background path." This means that after the circle's background has been drawn/filled/whatever, the ⟨*code*⟩ is executed.

When the ⟨*code*⟩ is executed, the coordinate system will have been transformed in such a way that the point (1pt, 0pt) lies at the right end of the circle and (0pt, 1pt) lies at the top of the circle. (More precisely, these points will lie exactly on the middle of the radial line.)

Here is an examples of how to use this shape:



```
\tikz \node [generic circle ISO14617,
circle symbol size = width 5 height 5,
/pgf/generic circle ISO14617/before background={
\pgfpathmoveto{\pgfpoint{-0.5pt}{-0.5pt}}
\pgfpathlineto{\pgfpoint{1pt}{0pt}}
\pgfpathlineto{\pgfpoint{0pt}{1pt}}
\pgfpathlineto{\pgfpoint{-0.5pt}{-0.5pt}}
\pgfusepath{draw}
},
transform shape,
draw] {};
```

Shape generic rectangle ISO14617

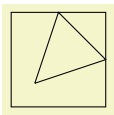
The same as generic circle ISO14617, only now it inherits from rectangle pid, and it allows for non-uniform scaling along the x and y-axis. This shape also allows you to use you to specify a path that should be added before the background path using the following key:

/pgf/generic rectangle ISO14617/before background=⟨*code*⟩ (no default)

When a node of shape generic rectangle ISO14617 is created, the current setting of this key is used as the "before background path." This means that after the rectangle's background has been drawn/filled/whatever, the ⟨*code*⟩ is executed.

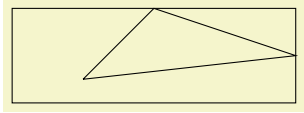
When the ⟨*code*⟩ is executed, the coordinate system will have been transformed in such a way that the point (1pt, 0pt) lies at the right end of the rectangle and (0pt, 1pt) lies at the top of the rectangle.

Here is an examples of how to use this shape:



```
\tikz \node [generic rectangle ISO14617,
circle symbol size = width 5 height 5,
/pgf/generic rectangle ISO14617/before background={
\pgfpathmoveto{\pgfpoint{-0.5pt}{-0.5pt}}
\pgfpathlineto{\pgfpoint{1pt}{0pt}}
\pgfpathlineto{\pgfpoint{0pt}{1pt}}
\pgfpathlineto{\pgfpoint{-0.5pt}{-0.5pt}}
\pgfusepath{draw}
},
transform shape,
draw] {};
```

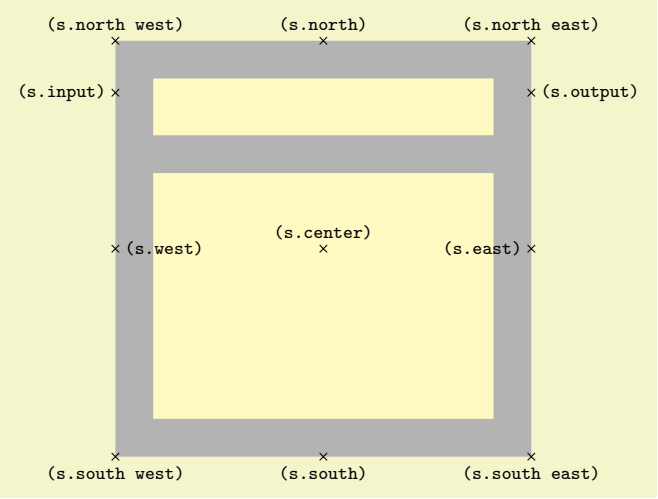
When the circuit symbol size key is changed the scale of the before background path is altered as well.



```
\tikz \node [generic rectangle ISO14617,
circle symbol size = width 15 height 5,
/pgf/generic rectangle ISO14617/before background={
\pgfpathmoveto{\pgfpoint{-0.5pt}{-0.5pt}}
\pgfpathlineto{\pgfpoint{1pt}{0pt}}
\pgfpathlineto{\pgfpoint{0pt}{1pt}}
\pgfpathlineto{\pgfpoint{-0.5pt}{-0.5pt}}
\pgfusepath{draw}
},
transform shape,
draw] {};
```

Shape 2531 ISO14617

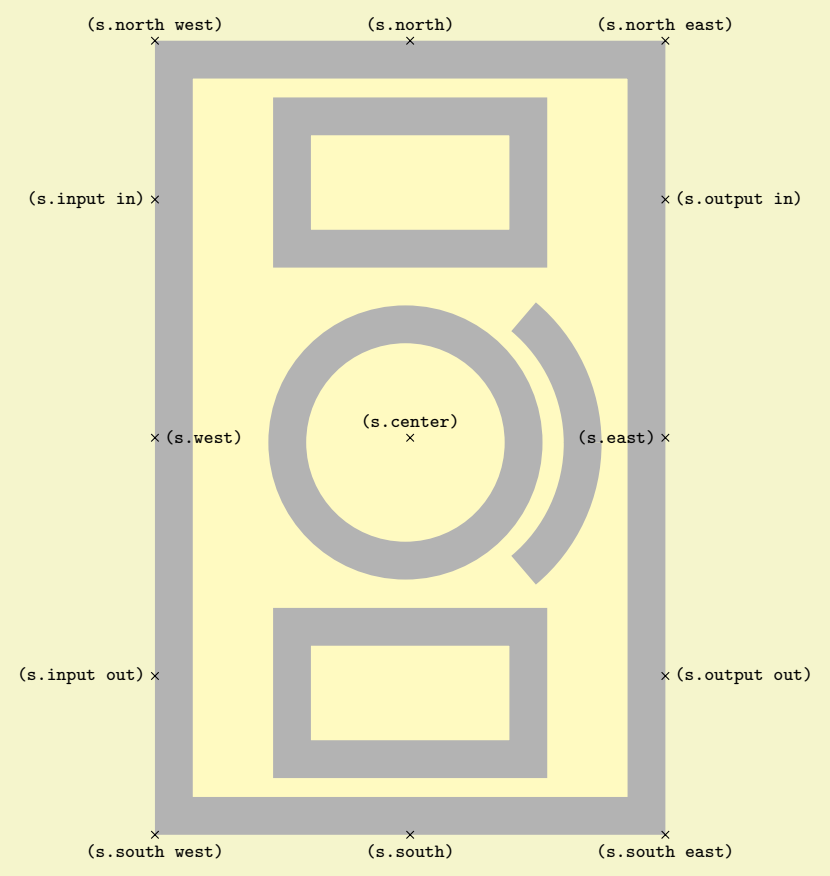
A shape based on the `rectangle` with a top compartment, where the `input` and `output` anchors are placed at the top part of the rectangle. This shape is used for symbol 2531 which is a boiler/steam generator.




```
\begin{tikzpicture}
\node [2531 ISO14617,
circuit symbol size = width 20 height 20,
shape example,
draw] (s) {};
\foreach \anchor/\placement in
{center/above,
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}
```

Shape 2518 ISO14617

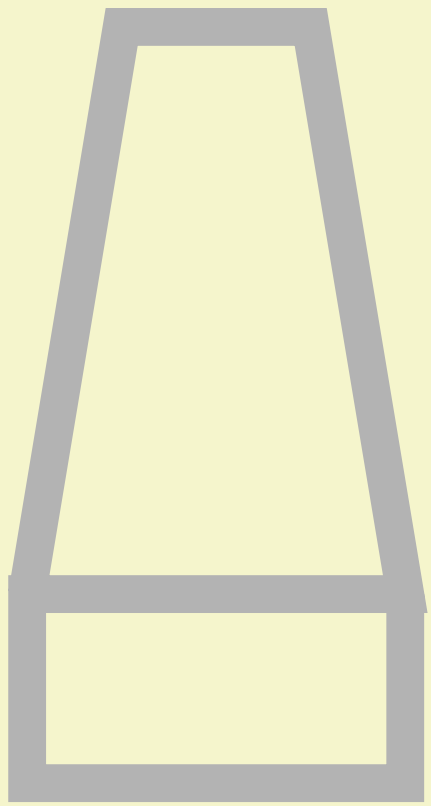
A shape based on the `rectangle` with a top compartment, where the following anchors are defined `input in`, `output in`, `input out` and `output out`. These anchors represent the ingoing flow and the return flow for a regenerative pre-heater. *ToDo:* issue #1




```
\begin{tikzpicture}
\node [2518 ISO14617,
circuit symbol size = width 25 height 40,
shape example,
draw] (s) {};
\foreach \anchor/\placement in
{center/above,
north/above, south/below, east/left, west/right,
north east/above, south east/below, south west/below, north west/above,
input in/left,output in/right, input out/left,output out/right}
\draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

Shape 2521 ISO14617

A shape based on the `rectangle` with a bottom compartment, where the following anchors are defined `input` and `output`. This shape is used for symbol 2521 which is a cooling tower.



```
\begin{tikzpicture}
\node [2521 ISO14617,
circuit symbol size = width 20 height 40,
shape example,
draw] (s) {};
% \foreach \anchor/\placement in
%  {center/above,
%   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}
```

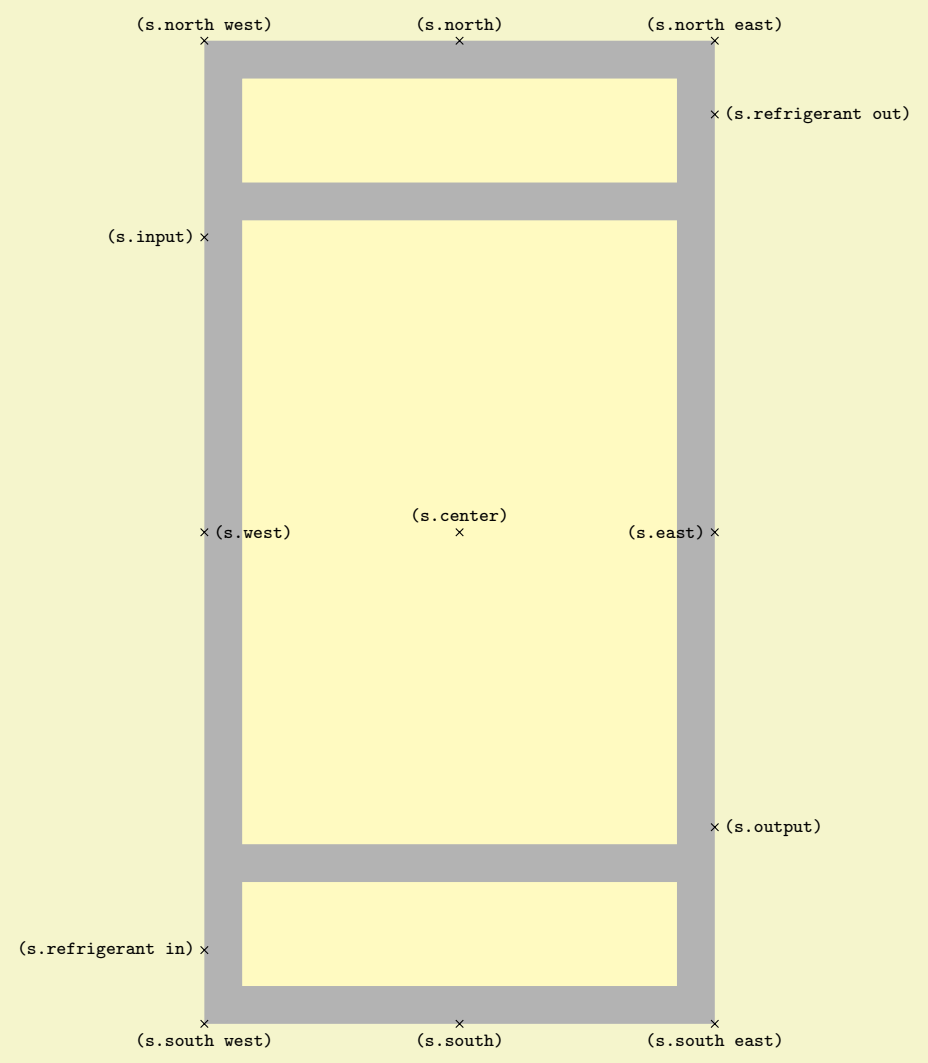
Shape `heat exchanger long ISO14617`

A shape based on the `rectangle` with a top and a bottom compartment, where the following anchors are defined `input`, `output`, `refrigerant in` and `refrigerant out`. This shape is used for symbol `2511` and `2514` which heat exhangers. This shape allows the use for a custum background path using the following key:

`/pgf/heat exchanger long ISO14617/before background`=⟨*code*⟩ (no default)

When a node of shape `heat exchanger long ISO14617` is created, the current setting of this key is used as the "before background path." This means that after the shapes background has been drawn/filled/whatever, the ⟨*code*⟩ is executed.

When the ⟨*code*⟩ is executed, the coordinate system will have been transformed in such a way that the point (1pt, 0pt) lies at the right end of the rectangle and (0pt, 1pt) lies at the top of the rectangle.




```
\begin{tikzpicture}
\node [heat exchanger long ISO14617,
circuit symbol size = width 25 height 50,
shape example,
draw] (s) {};
\foreach \anchor/\placement in
{center/above,
north/above, south/below, east/left, west/right,
north east/above, south east/below, south west/below, north west/above,
input/left,output/right,refrigerant in/left,refrigerant out/right}
\draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

4.5 ISO-14617 P&ID library

TikZ Library circuits.pid.ISO14617

```
\usetikzlibrary{circuits.pid.ISO14617} % LaTeX and plain TeX
\usetikzlibrary[circuits.pid.ISO14617] % ConTeXt
```

This library provides graphics for *Process & Instrumentation Diagrams* according to the international standard ISO-14617.

This library defines the following key:

/tikz/circuit pid ISO14617 (no value)

This key should be passed as an option to a picture or a scope that contains a diagram. It will do some internal setups.

/tikz/sensor (no value)

test

alias //tikz/751/sensor


```
\tikz[circuit pid ISO14617] \node[sensor={info=center:W}] {};
```

5 Example symbols

5.1 ISO-14617-7 Basic mechanical components

ISO name	*Base Key*	*Modifiers*	*Appearance*
open store	/tikz/open store		
tank	/tikz/tank		
tank atmospheric pressure	/tikz/tank atmospheric pressure		
container	/tikz/container		
cistern	/tikz/cistern		
pressure vessel	/tikz/pressure vessel		
vacuum vessel	/tikz/vacuum vessel		

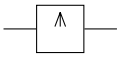
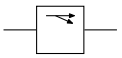
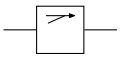
5.2 ISO-14617-9 Pumps, compressors and fans

ISO name	*Base Key*	*Modifiers*	*Appearance*
pump	/tikz/pump		
displacement pump	/tikz/pump	displacement	
adjustable displacement pump	/tikz/pump	displacement, adjustable	
centrifugal pump	/tikz/pump	centrifugal	
rotodynamic pump	/tikz/pump	rotodynamic	
compressor	/tikz/compressor		
fan	/tikz/fan		

5.3 ISO-14617-11 Devices for heat transfer and heat engines

ISO name	*Base Key*	*Modifiers*	*Appearance*
tank	/tikz/tank	with=heating coil	

5.4 ISO-14617-12 Devices for separating, purification and mixing

ISO name	*Base Key*	*Modifiers*	*Appearance*
`humidifier`	`/tikz/tank`	`with={spray nozzle}{0}{0.5}`	
`device for separating`	`/tikz/envelope`	`device for separating`	
`device for mixing`	`/tikz/envelope`	`device for mixing`	

Index