The chemplants package

Elia Arnese Feffin*

Version 0.9.8 – 2019/11/19

Abstract

The chemplants package offers tools to draw simple or barely complex schemes of chemical processes. Process units and styles for streams and utilities are defined to be a sort of extension of the Ti*k*Z package, thus a basic knowledge of the logic of this powerful tool is required to profitably use chemplants.

Contents

1 Motivations **2**

2 Starting Point **3**
2.1 Licensing . . . 3
2.2 Installation . . . 3
2.3 Basic Knowledge Required . . . 3
2.4 Purposes of the Package . . . 4

3 Streams and Utilities **4**
3.1 Main Stream . . . 4
3.2 Secondary Stream . . . 5
3.3 Utility Stream . . . 5
3.4 Signal . . . 5
3.5 Hidden Streams and Components . . . 6

4 Process Units **7**
4.1 Understanding Symbols . . . 8
4.2 Fluids and Solids Storage . . . 10
4.3 Fluid Handling . . . 13
4.4 Heat Exchangers . . . 20
4.5 Separators . . . 25
4.6 Columns . . . 33
4.7 Reactors . . . 34
4.8 Associative Pics for Reactors . . . 37

*e-mail: elia24913@me.com

5 Process Utility Units **41**
5.1 Valves 42
5.2 Control Instruments 44
5.3 Process Inlets and Outlets 46
5.4 Nozzles 46
5.5 Blocks 47

6 Transforming Units **48**

7 Customisation **51**

8 Examples **55**

9 What Happens Next **66**

References **66**

1 Motivations

The chemplants package had birth during my bachelor's degree in Chemical Engineering at the University of Padova. I discovered LaTeX during the first year and I started using it to write my lecture notes, my reports and almost every document I had to produce; the more I used it, the more I explored the boundless universe of extensions available, both using them and studying relative documentation (and guides). Soon, I encountered one the most beautiful and complex packages of the LaTeX distribution: the Ti*k*Z package.

Programmed drawing really changed the way I look at technical drawings and schematic representation. Ti*k*Z gives the author the possibility to use one program only to produce written documents and the drawings they require, allowing a perfect integration between them. I used this extremely powerful tool to draw schematics of mechanics, sketches of diagrams and electrical circuits (with another powerful package based on Ti*k*Z: CircuiTi*k*Z).

At a certain point, as a chemical engineering student, I had the need to start drawing schematics of chemical processes in the forms of a block flow diagram (BFD) or of a process flow diagram (PFD). BFDs are not an issue since they are very simple, but PFDs require specific symbols for the representation of process units. I looked for a long time for a CircuiTi*k*Z-like package useful for chemical plants, but nothing seems to be available to this aim.

At that point chemplants kicked in, starting as a simple idea: to use Ti*k*Z, in particular the possibility to define custom styles and pics, to fix a standard set of symbols to be used with Ti*k*Z drawing commands, symbols meant to be easy-to-use and easy-to-modify. Initially, just the units I had the need to represent were considered, but then the set of definitions started growing and a more flexible code structure was required, thus the real birth of the package took place. This happened during the first year of my master's degree.

As already told, the motivations of the chemplants package are to fill a lack in the LaTeX packages tree and to give everyone the possibility to draw schematics of chemical processes, particularly PFDs, in a simple way. A basic knowledge of Ti*k*Z is clearly required.

All of the symbols and styles defined are based on the UNICHIM regulation, the Italian code to draw chemical processes diagrams. It takes its name from the homolog association: *associazione per l'unificazione del settore dell'industria chimica*. This package is not pretentious enough to strictly follow UNICHIM, also because this regulation defines parameters to draw schematics way more complex than PFDs. Anyway, UNICHIM is still the guiding light of the representation of units and streams defined by chemplants.

2 Starting Point

2.1 Licensing

The chemplants package is covered by the LaTeX Project Public License (LPPL), version 1.3c or later. Basically, this means that users are free to use, modify and distribute any part of the package. More accurate and detailed informations can be found into the license itself, the latest version of which is available at `http://www.latex-project.org/lppl.txt`.

2.2 Installation

The package is supplied as a simple zip archive containing the `chemplants.sty` file, the main code of the package, and the `chemplants_doc.pdf` file, the documentation (this file), together with its source code. The simplest way to make the package work is to place `chemplants.sty` into the same directory of the `.tex` file that uses chemplants, a solution useful to users who do not want to go along a full installation.

A better installation procedure for users who adopt the TeXlive distribution on a Linux-like system (including MacOS) consists in looking for the main directory of the distribution and following the path:

```
../texlive/texmf-local/tex/latex/
```

in which a new folder called `chemplants` should be created. The file `chemplants.sty` should be placed into that folder. After that, it is necessary to let TeX know that the tree structure is changed and that a new package is available, hence it is necessary to type in the terminal of the system:

```
sudo texhash
```

and to wait for the magic to be done (the insertion of the user password may be required after this instruction). Another option is to let the `tlmgr` utility do all the work, moving the terminal action to the directory in which `chemplants.sty` is (`cd` instruction) and typing:

```
sudo tlmgr install chemplants
```

For a Windows system running TeXlive it should work the same way, but commands have to be typed in the prompt removing the `sudo` prefix, used by Linux-like systems. Finally, MiKTeX should provide a custom package manager to handle the TeX tree structure, so chemplants have to be installed in the way MiKTeX manager usually handles new packages.

2.3 Basic Knowledge Required

In order to profitably use the chemplants package, a basic knowledge of the Ti*k*Z package is required. There are a lot of excellent introductory guides to this gigantic

package and for every doubt there is also the enormous and excellent documentation of the package: Tantau 2019. For impatient readers, Crémer 2011 (available on CTAN into the Ti*k*Z package directory) offers a short but useful introduction to Ti*k*Z.

Italian language users can find on the internet some very useful guides to learn the bases of Ti*k*Z (and more of what is needed to use chemplants). A short but effective introduction is given in a dedicated chapter of Pantieri 2017, derived from a previous article of the same author: Pantieri and Gordini 2014. Users who want to be really surprised by the capability of Ti*k*Z, besides the full documentation aforementioned, can check Fiandrino 2014, an excellent guide available on the GUIT website (the Italian TEX and LATEX users group).

Finally, readers interested on UNICHIM regulations can easily find some tables on the internet, or a more interesting source of information in Cacciatore and Calatozzolo 2018. This book reports a selection of tables coming form UNICHIM 1994, the official UNICHIM manual, mainly the ones concerning process units, styles for streams and control instrumentation; there are also some examples of PFDs.

2.4 Purposes of the Package

Having mentioned the UNICHIM regulation, it is important to spend a couple of words more about the aim of this package. The chemplants package is meant to help users which have a basic knowledge of Ti*k*Z in representing schemes of chemical processes and plants in a simple way. This requires to access to symbols for process units, styles for streams and, possibly, symbols and styles for control instrumentation. These three elements, plus a rudimental mechanism to set the main parameters of the drawing, are what chemplants provides.

This package is not meant to produce representation of complex units or of very specialised equipments such as the Linde column used in air distillation plants or the Casale reactor used in ammonia synthesis. A fine representation of units like the two just mentioned requires more than a simple symbol to be placed somewhere in a PFD, but a complex and detailed scheme, which goes beyond the scope of chemplants. Moreover, complex drawings like these are not that common, so they do not need to be defined as pics in order to be extensively used and easily modified. Users in the need to represent specialised schematics should exploit the basic and advanced features of the Ti*k*Z package in a more general way, rather than asking chemplants to do it for them.

3 Streams and Utilities

Streams to be used in BFDs and PFDs can be obtained by means of the `\draw` command of Ti*k*Z to represent lines with the operator `--`. The graphicas aspect of a stream is defined as a Ti*k*Z style and can be applied to any `\draw` command as an option to the command itself. Although not explicitly showed, all of the following example instructions are intended to be used within a `tikzpicture` environment.

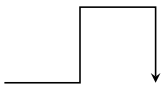
3.1 Main Stream

A main stream indicates the main path of a process, the one prime matters follow to be transformed into the desired products. It is defined as a style called `main stream`, to be applied to the `\draw` command:

```
\draw[main stream] (0,0) -- (2,0);
```

and yields an arrow of `semithick` thickness:

As for all of the Ti*k*Z arrows declared through the `->` option, the tip is present only on the last point of the path:



so every main stream (arrow) to be represented requires its own `\draw` command.

3.2 Secondary Stream

A secondary stream indicates a process stream different from the main one, still very important to the process, though not as the principal line (reactants recycle is an example). It is defined as a style called `secondary stream`, to be applied to the `\draw` command:

```
\draw[secondary stream] (0,0) -- (2,0);
```

and yields an arrow of `thin` thickness:

3.3 Utility Stream

A utility stream indicates all of the streams different from the main and secondary ones, but anyway useful to the process (at least in a PFD), such as heating steam or cooling water. It is defined as a style called `utility stream`, to be applied to the `\draw` command:

```
\draw[utility stream] (0,0) -- (2,0);
```

and yields an arrow of `very thin` thickness (the standard one in Ti*k*Z):

The standard arrow tip defined for `main stream`, `secondary stream` and `utility stream` styles (and for everything else in chemplants that has an arrow tip) is the `stealth` arrow of Ti*k*Z. This can be changed, as a lot of other graphical parameters can. The way such customisations can be achieved will be discussed after the introduction of all units.

3.4 Signal

In a PFD, it is possibile to sketch also the main paths of the control system of a chemical plant. Controllers are connected to units and to actuators through a signal line, which in a PFD is intended as a generic signal (no distinction between pneumatic, electric and so on). It is defined as a style called `signal`, to be applied to the `\draw` command:

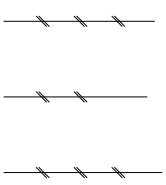
```
\draw[signal] (0,0) -- (2,0);
```

and yields a line of `very thin` thickness with parallel and oblique short lines placed at regular intervals:

It should be notice that `signal` style is not very flexible. Markings start 5 mm after the path initial point and are spaced by 5 mm, ending at least 5 mm before the final point of the path. Thus, the optimal result is obtained if a path has a length which is a multiple of 5 mm. Otherwise, there will be an "uncovered portion" of the path. For example, the code:

```
\draw[signal] (0,2) -- (2,2);
\draw[signal] (0,1) -- (1.8,1);
\draw[signal] (0,0) -- (2.2,0);
```

yields:



The top line only yields the correct graphical result. Notice also that the minimum length a `signal` path should have is 1 mm, which results into a short signal with a single mark on its middle point.

Sometimes the `signal` style is not flexible enough, such as for very short paths. For this specific aim, a `short signal` style is defined, to be used in the same way of the standard `signal`, but it places a single mark in the middle of the path:

so it is recommended when a signal shorter than 1 cm has to be drawn.

3.5 Hidden Streams and Components

Although not really considered by the UNICHIM regulation, it is sometimes useful to show streams and components within a unit. Two special styles are defined to this aim.

The `hidden stream` style is defined to be applied to the `\draw` command:

```
\draw[main stream, hidden stream] (0,0) -- (2,0);
```

and yields a `dashed` line with no specified thickness, so it has to be used with either `main stream` or `utility stream` to draw the arrow of the right thickness:

The `hidden component` style is defined to be applied to the `\pic` command to draw components within a unit (in the way that will be introduced later):

```
\pic[hidden component] at (0,0) {valve=main};
```

and yields the required unit represented not with a solid line, but with a `densely dotted` pattern:

4 Process Units

Process units are defined as pics. A pic is a Ti*k*Z object that represents a simple draw to be placed at certain coordinates through the `\pic` command. The advantages of using a pic to define a standard symbol are the easiness of use and of modification of the symbol itself, if needed.

In order to define a pic, it is important to specify the starting point and, of course, the code that draws the pic. The starting point is the most important point of the pic since it is its anchor, the point that will be placed to the coordinates declared by the `\pic` command. The drawing code defines, instead, the dimensions of the pic and its default orientation (and the line thickness, `thick` by default in chemplants).

The two features just introduced imply to specify what is the logic of the pics defined by chemplants. The main anchor is almost always defined as the centre of the symbol (or as an important point close to it). This choice let all of the transformation options defined by Ti*k*Z to be applied to the pics representing the units, so a high flexibility in placing and orienting units is granted. However, there is also a drawback in this approach: the coordinates of the main anchor of the pic cn be established, but the coordinates of the points in which streams touch the unit have to be calculated by hand.

The problem just highlighted was the main trouble of the chemplants package. This issue was fixed redefining units in a more cleaver way and using more advanced tools of the Ti*k*Z package. I would like to explain a little bit of history of the package before going on.

When I started to define units, since they where few and very simple (and since I was not that able in programmed drawing), pics contained the simple geometrical description of the symbols. This was easy to do, but it had the great disadvantage of forcing the user (only me at that time) to manually calculate coordinates for the placement of units and for streams connections. In a certain way, the problem is also an opportunity because it forces the drawer to accurately plan the layout of the schematics, but it is silly and very time-consuming in large and complex drawings.

My dream was to build a simple automatic interface like the CircuiTi*k*Z one, which uses the `to` operator of Ti*k*Z in a very clever way (within a custom environment of the package). Anyway, electric networks schematics are simpler than chemical processes drawing under this aspect: a great part of electrical components, such as resistors, capacitors and inductors, are bipoles, which means with one input and one output only, just two connections points. In chemical plants, most of the units are not simple bipoles, but can have a wide variety of inlet and outlet streams connected in variable points. Due to this necessity, I was not able to find a simple solution during the very first writing of the package code.

After some researches, I saw the light: coordinate nodes. The Ti*k*Z package offers the possibility to define a pic with some internal coordinate nodes, special points to which a name can be assigned. In this way it is not necessary to calculate by hand the position of the remarkable points of a unit, but it is enough to know the names of the coordinate nodes which represent them to snap a stream on those points. Moving a unit will then move also anchors, so the snap will be held. Finally, Ti*k*Z let a pic to be prefixed with a name that univocally identifies the pic. This name is prefixed also to anchors declared in the pic definition, hence also anchors are univocally determined, avoiding to snap a stream to the wrong unit.

4.1 Understanding Symbols

Units defined by chemplants can be two different kinds of pics:

- simple pic objects, which can be used by just calling them through a `\pic` command;
- pic objects with a mandatory argument, in which the pic name called through the `\pic` command has to be followed by a specification.

The second category is particularly useful to draw similar units distinguished by some details, such as different columns types, or to let units to be sensible to the context, for example units that have to be drawn with different thicknesses. A third possibility is to give a text argument to the pic that has to be represented inside the pic itself; this is particularly useful for control instruments.

The General Pic Syntax

A pic is a Ti*k*Z object which can be called by the `\pic` command. The general syntax of the command is more or less:

`\pic [`⟨*options*⟩`] (`⟨*identifier*⟩`) at (`⟨*coordinate*⟩`) {`⟨*name*⟩`};`

where:

- ⟨*options*⟩ is a list of options to be passed to the pic and it is, as the name says, an optional argument;
- ⟨*identifier*⟩ is a user-defined name assigned to the pic that should univocally determine it and that will be prefixed, together with a dash, `-`, to all of the nodes names (it is an optional argument for the pic, but mandatory if one wants to access the node features);
- ⟨*coordinates*⟩ is whichever expression Ti*k*Z recognises as a specification of the coordinates on its canvas;
- ⟨*name*⟩ is the name of the pic to be drawn.

This syntax holds true for simple pics only. Units defined as pics with arguments have a similar syntax, but with an extra argument:

`\pic [`⟨*options*⟩`] (`⟨*identifier*⟩`) at (`⟨*coordinate*⟩`) {`⟨*name*⟩`=`⟨*type*⟩`};`

where ⟨*type*⟩ is the argument to be passed to the ⟨*name*⟩ unit and which specifies some features of the unit itself. The usage of these syntaxes will be clearer in the future, when examples will be shown.

Common Nodes

In the following, units defined by chemplants are listed, described and shown. In order to profitably draw units it is important to know where their anchors and nodes are, what their dimensions are and which is their default orientation. Units will be drawn and some information will be given in the meanwhile:

- drawings will be shown in their default orientation;
- anchors will be represented on units by a little red cross;

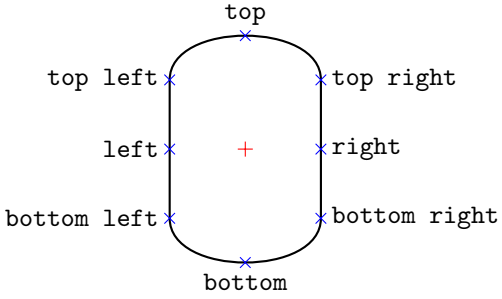
- remarkable nodes, the ones defined by the pic code, will be represented on units by a little blue cross with an abbreviation of the name of the node on its side;
- dimensions will be marked on drawings, both total dimensions and distances from the important points to the anchor.

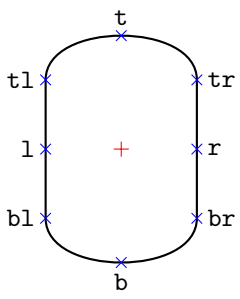
Only instructions to produce the "raw units" will be shown, so, in order to avoid confusion, three units will be presented: a "pure" unit; a unit marked with dimensions; a unit marked with the nodes.

What concerns the names of the remarkable nodes which will be used in the following require a clarification. As a general rule, nine standard names are used to identify chemplants nodes. These names are long for the sake of clearance, but they will be abbreviated in this discussion just to limit the space they require. A short list:

- the node `anchor` is always placed where the pic is anchored and it will be never marked by a name, but only by the small red cross mentioned above;
- the node `left` is placed on the left limit of the unit, in its centre, and will be indicated using the abbreviated name `l`;
- the node `bottom left` is placed on the left limit of the unit, in its lower useful point, and will be indicated using the abbreviated name `bl`;
- the node `bottom` is placed on the lower limit of the unit, in its centre, and will be indicated using the abbreviated name `b`;
- the node `bottom right` is placed on the right limit of the unit, in its lower useful point, and will be indicated using the abbreviated name `br`;
- the node `right` is placed on the right limit of the unit, in its centre, and will be indicated using the abbreviated name `r`;
- the node `top right` is placed on the right limit of the unit, in its upper useful point, and will be indicated using the abbreviated name `tr`;
- the node `top` is placed on the upper limit of the unit, in its centre, and will be indicated using the abbreviated name `t`;
- the node `top left` is placed on the left limit of the unit, in its upper useful point, and will be indicated using the abbreviated name `tl`.

Taking as example a tank, full names of the nodes, shown on the unit on the left, will be indicated using abbreviated names, as on the unit on the right:





It should be noticed that not all of the "boundary nodes" are defined for every unit. Usually they are all present in a rectangle-shaped unit, such as a tank, but in circle-shaped units the "corner nodes" are not present. Furthermore, units represented by strange symbols can have some special nodes to indicate their remarkable points. Special nodes will be discussed, both with extended and abbreviated names, describing units which require them.

It is important to recall that to access the node features of pics it is mandatory to assign an identifier to the `\pic` command. Assuming that such identifier is `T`, then one can access to all of the nodes inside the pic thanks to it using them as coordinates nodes (as they are). Also interposing a dash, `-`, is fundamental. For example `T-bottom right` used as coordinate identifies the point of the `bottom right` node of the pic identified (prefixed) by `T`.

A Command to Show Measures

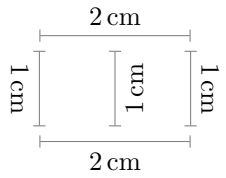
By the way, chemplants defines a command to draw dimensions: `\measure`. This command has to be used directly within a `tikzpicture` environment (also without the final semicolon) and requires three mandatory arguments: start point coordinates, end point coordinates and the text of the measure. Coordinates can be declared in any way recognised by Ti*k*Z, while the text can be anything that can be placed into a Ti*k*Z node.

The default appearance of a measure is a grey `thin` line with flat tips. The measure is yield as text placed in the middle of the line, sloped in its direction and always below it. Anyway, `\measure` accepts an optional argument in which any anchor specification that can be passed to a Ti*k*Z node can be used. The most useful is `above`, which moves the text of a measure above the line.

Some examples using the siunitx package (but also normal text will work). The code:

```
\measure{(0,0)}{(2,0)}{\SI{2}{\cm}}
\measure{(0,1.2)}{(0,0.2)}{\SI{1}{\cm}}
\measure{(1,0.2)}{(1,1.2)}{\SI{1}{\cm}}
\measure[above]{(2,1.2)}{(2,0.2)}{\SI{1}{\cm}}
\measure[above]{(0,1.4)}{(2,1.4)}{\SI{2}{\cm}}
```

yields:



It is important to notice that the value of the measure is not calculated automatically, but it must be passed as an argument. This is useful to indicate, for example, the length of a pipe or the real dimensions of a unit in a process scheme.

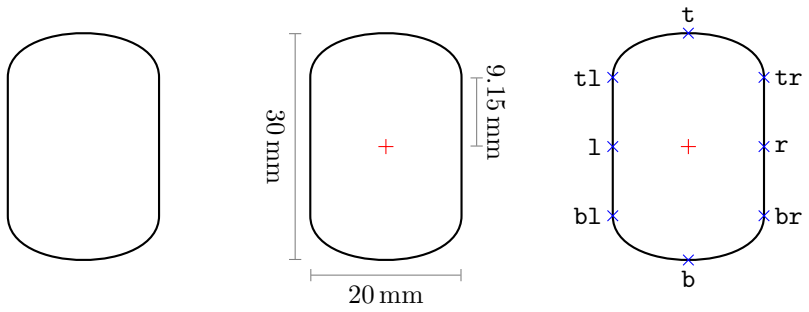
4.2 Fluids and Solids Storage

Tank

A tank is a generic recipient useful to store process fluids or solids. A generic symbol is defined for a process tank and it can be used with no distinction on its shape, which is represented by a rectangle with rounded bases. It is defined as a simple pic called `tank`:

```
\pic at (0,0) {tank};
```

and yields a vertical tank anchored in its centre:



where the measure on the right indicates the distance from the middle of the tank to the point where the curvature begins. This point is identified by the `top right` node and by its analogs.

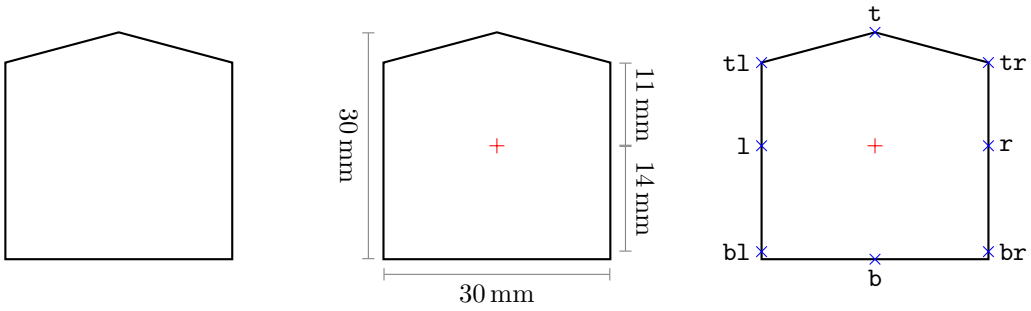
The `tank` pic is generic and it is useful to represent process tanks. Storage tanks can be represented with the same pic, but there are also some specific symbols.

Cone Roof Tank

A cone roof tank is a large tank placed on the ground and useful to store process fluids or solids. As the name says, it has a cone-shaped roof. It is defined as a simple pic called `cone tank`:

```
\pic at (0,0) {cone tank};
```

and yields a rectangle, in which centre there is the anchor, with a cone-shaped roof:



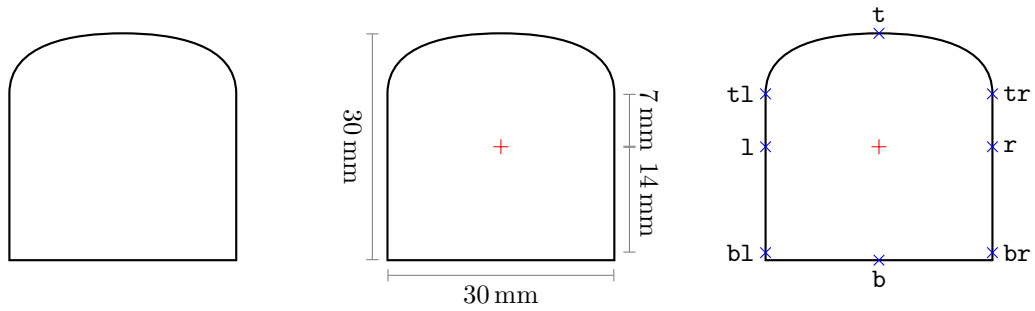
where the measure on the bottom right indicates the distance from the middle of the tank to the point where the outlet stream should be connected. This point is identified by the `bottom right` node and by its analogous node on the left.

Dome Roof Tank

A dome roof tank is the same of a cone roof tank, but, clearly, it has a dome-shaped roof. It is defined as a simple pic called `dome tank`:

```
\pic at (0,0) {dome tank};
```

and yields a rectangle, in which centre there is the anchor, with a dome-shaped roof:



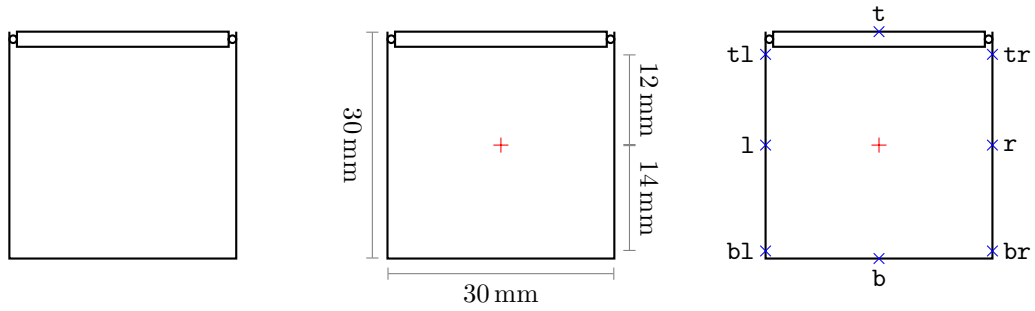
where the measure on the top right indicates the distance from the middle of the tank to the point where the curvature begins, while the measure on the bottom right indicates the distance from the middle of the tank to the point where the outlet stream should be connected. This last point is identified by the `bottom right` node and by its analogous node on the left.

Floating Roof Tank

A floating roof tank is a large tank placed on the ground and useful to store process fluids or solids. It has the advantage to change in volume depending on how much it is filled, which allows to compensate pressure unbalances during the filling and the draining of the tank, also to avoid the formation of a gas pocket above the stored substance. It is defined as a simple pic called `floating roof tank`:

```
\pic at (0,0) {floating roof tank};
```

and yields a rectangle, in which centre there is the anchor, with a sketch of the floating roof:



where the measure on the bottom right indicates the distance from the middle of the tank to the point where the outlet stream should be connected. This point is identified by the `bottom right` node and by its analogous node on the left.

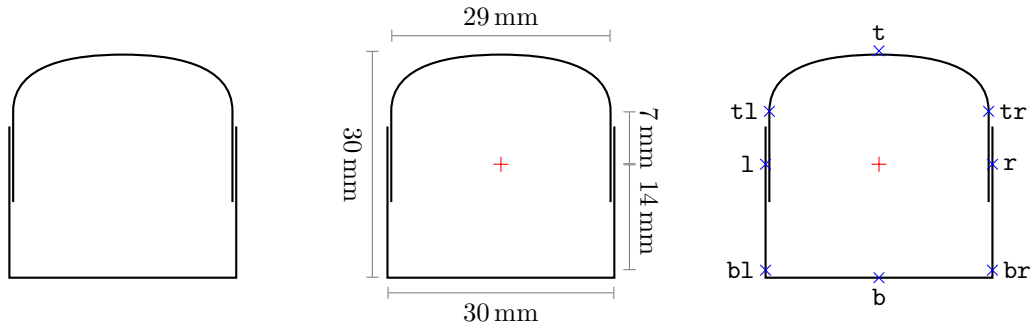
Bell GasHolder

Tanks introduced so far can be used to represent general systems to store either solids, liquids and gases. For gases storage, a specific kind of tanks exists, which is designed to control the gas pressure in a simple way: gasholders.

The most common gasholder uses a bell-shaped floating roof to expand and contract its volume depending on the amount of gas to be stored. It is defined as a simple pic called `bell gasholder`:

```
\pic at (0,0) {bell gasholder};
```

and yields a rectangle, in which centre there is the anchor, with a sketch of the bell-shaped floating roof:



where the measure on the top indicates the width of the moving roof, th measure on the top right indicates the distance from the middle of the gasholder to the point where the curvature begins and the measure on the bottom right indicates the distance from the middle of the gasholder to the point where a stream should be connected. This last point is identified by the `bottom right` node and by its analogous node on the left.

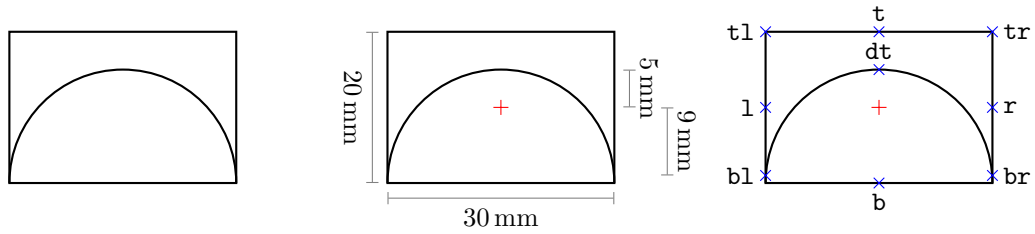
It should be quite obvious that `left` and `right` nodes, falling on the roof rails, are not meant to be used for stream connections, but just to labelling purposes or for control instrumentation connections.

Dry GasHolder

Another kind of gasholder is the so called dry gasholder, where there is a fixed structure that contains an "expandable ballon" in which the gas is stored. It is defined as a simple pic called `dry gasholder`:

```
\pic at (0,0) {dry gasholder};
```

and yields a rectangle, in which centre there is the anchor, with a sketch of the ballon inside it:



where the measure on the top right indicates the distance from the middle of the gasholder to the top of the internal balloon, while the measure on the bottom right indicates the distance from the middle of the gasholder to the point where a stream should be connected. This last point is identified by the `bottom right` node and by its analogous node on the left.

A special node is defined for the `dry gasholder`: usually the inlet stream is connected to one bottom side of the balloon, on the `bottom right` node or on the `bottom left` node, while the outlet is connected to the top of the dome (and not to the top of the containing structure). This special node is called `dome top` and it is marked in the above drawing using the abbreviated name `dt`. This node can be called in the usual way: for example, in a `dry gasholder` identified as `G` the node is `G-dome top`.

4.3 Fluid Handling

Fluid handling devices are essential to chemical plants because it is always necessary to move a fluid, from one place to another against pressure drops generated by pipes or units.

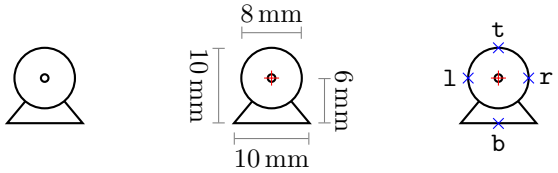
Fluid handling units used in the chemical industry are basically classified considering the kind of fluid they can move, either a liquid or a gas. On the basis of this classification, there is a main distinction between pumps for liquids and equipments to move gases. The latter are furthermore distinguished with respect to the pressure rise a unit can achieve: fans, blowers and compressors.

Centrifugal Pump

A pump is a mechanical machine useful to move a liquid and to increase its pressure. The most simple and widely used pump is a simple kinetic machine known as centrifugal pump. It is defined as a simple pic called `centrifugal pump`:

```
\pic at (0,0) {centrifugal pump};
```

and yields a circle, in which centre there is the anchor, supported by a triangular base and containing a little circle representing the inlet nozzle:



where the measure on the right indicates the distance from the inlet of the pump to its base, while the measure on the top indicates the diameter of the circle.

Nodes are defined only for the remarkable outlet points of the pump, but not for its inlet: this is placed on the anchor point, so it should not be difficult to find it (use the `anchor` node to snap purposes). Anyway, it should be noticed that the centrifugal pump, together with the fan that will be introduced later on, is the only unit that requires the inlet to be placed in the centre of the symbol, while other units do not. Finally, the `bottom` node is not really meant to connect streams, but it could be exploited to labelling purposes.

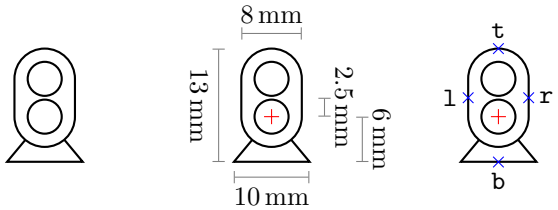
The centrifugal pump is for sure the most widely used one, but its applications are quite limiting with respect to the pressure rise and the kind of treatable liquid. Other pumps exist, such as the ones belonging to the volumetric machines: these are mainly rotary pumps and reciprocating pumps.

Rotary Pump

Rotary pumps are themselves a large family of machines, but UNICHIM defines a generic symbol to represent them all. It is defined as a simple pic called `rotary pump`:

```
\pic at (0,0) {rotary pump};
```

and yields an oblong circle supported by a triangular base and containing the sketch of two little circle, where the anchor is in the centre of the lowest one:



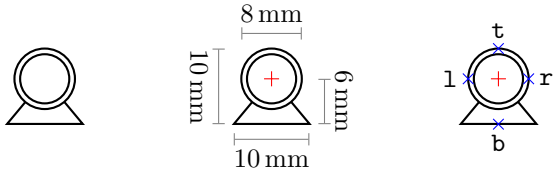
where measures on the right indicate the distance from the inlet of the pump (placed on its boundary, either `left` or `right` nodes) to its anchor, 2.5 mm, and the distance form the anchor of the pump to its base, 6 mm, while the measure on the top indicates the width of the body.

Liquid Ring Pump

As told before, the `rotary pump` is generic, so it can be used to represent gear pumps, lobe pumps, screw pumps, hollow disc pumps and other similar machines. A remarkable exception of a rotary pump not representable by means of the generic symbol is the liquid ring pump, useful to aspire a fluid rather than to compress it, hence to generate vacuum. It is defined as a simple pic called `liquid ring pump`:

```
\pic at (0,0) {liquid ring pump};
```

and yields a circle, in which centre there is the anchor, supported by a triangular base and containing a smaller circle representing the liquid ring:



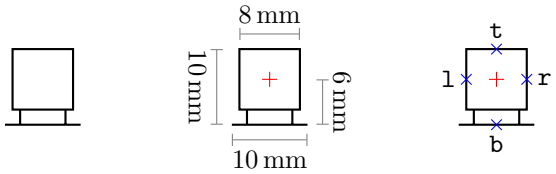
where the measure on the right indicates the distance from the inlet of the pump (placed on its boundary) to its base, while the measure on the top indicates the diameter of the circle.

Reciprocating Pump

The other category of volumetric pumps are the reciprocating ones, which work through the classical principle of cylinder and piston. A generic reciprocating pump is defined as a simple pic called `reciprocating pump`:

```
\pic at (0,0) {reciprocating pump};
```

and yields a square, in which centre there is the anchor, supported by a squared base:



where the measure on the right indicates the distance from the inlet of the pump (placed on its boundary) to its base, while the measure on the top indicates the length of the side side of the square.

Fan

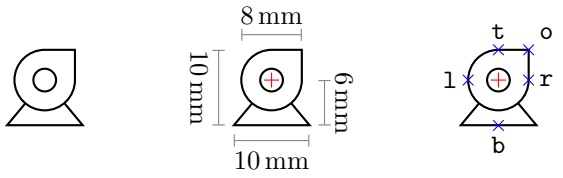
Liquid handling equipments are done, so the description can move to gas handling units. As told before, these are mainly distinguished on the basis of the pressure rise they can achieve, but also on their working principle.

If it is simply required to move a gas, without a significant pressure rise, a fan or, at most, a blower, is enough. These two units are simple kinetic machines similar to a

centrifugal pump and both of them are defined by UNICHIM as a single symbol, which in chemplants can be found under the name of `fan`:

```
\pic at (0,0) {fan};
```

and yields a circle, in which centre there is the anchor, with a squared vertex, supported by a triangular base and containing a little circle representing the inlet nozzle:



where the measure on the right indicates the distance from the inlet of the fan (this time again placed in its centre) to its base, while the measure on the top indicates the diameter of the circle.

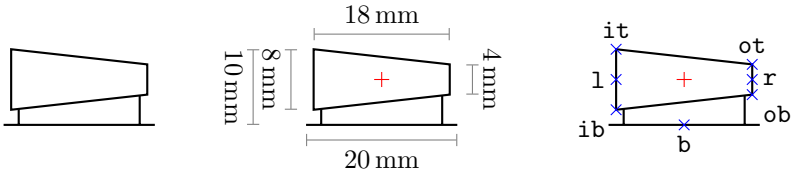
A special node is defined for the `fan`: the squared vertex indicates the outlet of the unit, thus it is identified by the `outlet` node, indicated as `o` in the above representation. As for the centrifugal pump, the `anchor` node should be used to snap the inlet stream to the fan.

Centrifugal Compressor

When a gas needs a high pressure rise, there is no other way but using a compressor. The only kinetic machine useful to compress a gas that has a symbol defined by chemplants is the centrifugal compressor. It is defined as a simple pic called `centrifugal compressor`:

```
\pic at (0,0) {centrifugal compressor};
```

and yields a cone frustum, in which middle there is the anchor, supported by a squared base:



Measures require some clarifications. The unit has a total width of 20 mm and a total height of 10 mm. The cone frustum has a length of 18 mm (the base protrudes horizontally 1 mm on each side, as for all of the other symbols defined as fluid handling units), its larger base has a height of 8 mm and the smaller one of 4 mm.

Also nodes are not that easy to understand. As always, the `bottom` node is defined to labelling purposes. The `left` and `right` nodes should not be used to connect streams, but can be exploited to connect control instrumentation or to represent the shaft of the compressor. Streams should be connected on the sides of the cone frustum: more precisely the inlet stream has to enter the unit on the side of the larger base and the outlet one has to leave the unit on the side of the smaller one, both of them vertically, in the direction perpendicular to the axis of the cone frustum. To this aim, four special nodes are defined:

- a node called `inlet bottom`, abbreviated in the picture above as `ib`;

- a node called `outlet bottom`, abbreviated in the picture above as `ob`;
- a node called `outlet top`, abbreviated in the picture above as `ot`;
- a node called `inlet top`, abbreviated in the picture above as `it`.

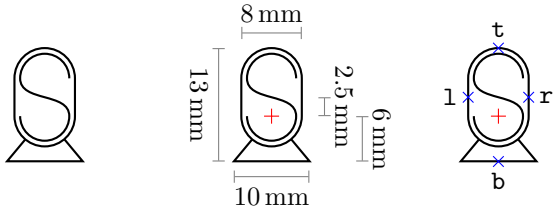
(At most, it is possible to tolerate the inlet placed on the left side of the unit, on the `left` node, but the outlet has to be compulsorily on the side of the smaller base.)

Rotary Compressor

As for pumps, also compressors are not kinetic machines only, but also volumetric, again rotary and reciprocating. Also in this case, a generic symbol is defined by UNICHIM to indicate all of the rotary compressors. It is defined as a simple pic called `rotary compressor`:

```
\pic at (0,0) {rotary compressor};
```

and yields an oblong circle supported by a triangular base and containing the sketch of a sort of S, where the anchor is in the centre of the lowest branch:



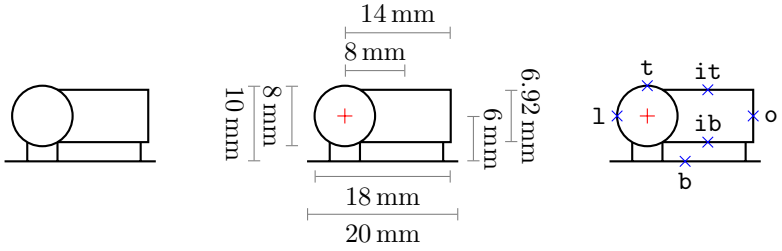
where measures on the right indicate the distance from the inlet of the compressor (placed on its boundary, either `left` or `right` nodes) to its anchor, 2.5 mm, and the distance from the anchor of the compressor to its base, 6 mm, while the measure on the top indicates the width of the body.

Reciprocating Compressor

Differently from what happens for pumps, reciprocating compressors are represented by means of two different symbols, one for single stage operations and one for multistage units. The simple stage unit is defined as a simple pic called `reciprocating compressor`:

```
\pic at (0,0) {reciprocating compressor};
```

and yields a circle, in which centre there is the anchor, merged with the rectangular body of the unit, all of which is supported by a squared base:



The circle is a sketch of the engine of the compressor, while the rectangle is the main body of the unit, where the gas gets effectively compressed.

Measures require some clarification. The unit has a total width of 20 mm and a total height of 10 mm. The figure obtained joining the circle and the rectangle has a length of 18 mm (the base protrudes horizontally 1 mm on each side). The circle has a diameter of 8 mm, while the rectangle has a height of 6.92 mm. The base of the unit is 6 mm below its anchor point.

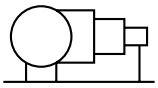
Since the "real" compressing part of the unit is the rectangle, nodes on the circle should be used to labelling purposes, to connect the control instrumentation or to sketch the shaft of the engine. The gas inlet stream should be connected to the `inlet bottom` or `inlet top` nodes, showed in the above drawing using the abbreviated names `ib` and `it` respectively, while the outlet stream has to be connected to the `outlet` node, `o` in the above representation. The two measures not cited yet are referred to these remarkable points of the unit: inlet nodes are 8 mm to the right of the anchor, while the outlet node is 14 mm to the right of the anchor.

MultiStage Compressor

Besides all of the above mentioned symbols, the most common unit used to indicate gas handling machine is the one that represents the multistage reciprocating compressor. Indeed, this is also widely used to represent a generic compression unit, despite its real meaning. It is defined as a simple pic called `multistage compressor`:

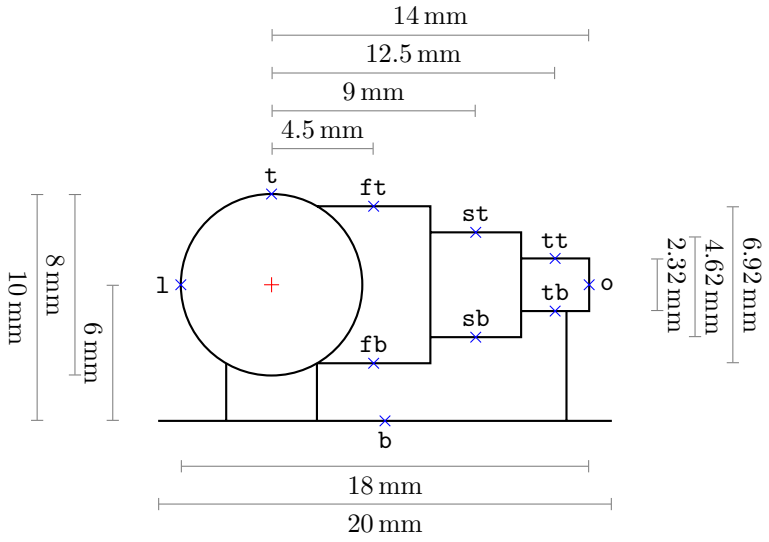
```
\pic at (0,0) {multistage compressor};
```

and yields a circle, in which centre there is the anchor, merged with three rectangles, all of which is supported by a squared base:



Again the circle is a sketch of the engine of the compressor, while the three rectangles are the main body of the unit and represent the compression stages. Rigorously, there should be as many rectangles as the compression stages are, but usually a symbol with three rectangles is used independently of the number of stages.

No measures or nodes are shown in the above drawing, that is the pure symbol, because of its complexity. In order to better understand measures and nodes, here is a huge `multistage compressor`:



where measure bars are just expanded, but numbers are correct and refer to the real dimensions of the little unit above.

The unit has a total width of 20 mm and a total height of 10 mm. The figure obtained joining the circle and the three rectangles has a length of 18 mm. The circle has a diameter of 8 mm, while the base of the unit is 6 mm below its anchor point. Measures on the right refer to the heights of the three rectangles: starting from the left of the symbol, the first rectangle has a height of 6.92 mm, the second one has a height of 4.62 mm and the third one is 2.32 mm high.

Understanding measures on the top requires to discuss first about nodes. As for the single stage reciprocating compressor, inlets are on the sides of the rectangles, which means on the top and on the bottom of each one, while the outlet is on the right side of the unit, point identified by the `outlet` node marked in the drawing as `o` and placed 14 mm to the right of the anchor.

Since rectangles represent compression stages, two nodes are defined for each one:

- the first stage is provided with a node called `first bottom` and with a node called `first top`, marked in the drawing as `fb` and `ft` respectively and both placed 4.5 mm to the right of the anchor;
- the second stage is provided with a node called `second bottom` and with a node called `second top`, marked in the drawing as `sb` and `st` respectively and both placed 9 mm to the right of the anchor;
- the third stage is provided with a node called `third bottom` and with a node called `third top`, marked in the drawing as `tb` and `tt` respectively and both placed 12.5 mm to the right of the anchor.

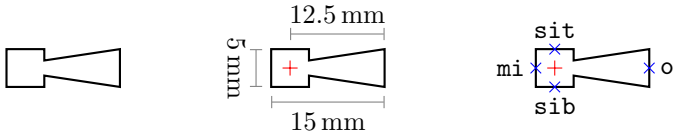
These nodes should be used as connection points for the inlet and are present to enable a flexible representation when complex necessities exist, such as the case of a stream which has to enter the compressor in the second stage, being mixed and compressed together with the one that enters in the first. Also complex paths can be produced thanks to these nodes, using them improperly as outlets to represent cooling operations applied to the fluid and interposed between compression stages.

Ejector

The last fluid handling unit defined by chemplants is neither a kinetic nor a volumetric machine, but it belongs to (and is the most remarkable example of) static machines category. The ejector works thanks to the relation of the pressure energy with the kinetic energy of a fluid, usually a gas, and most of the time is used to suck a secondary fluid into a driving stream or to generate vacuum thanks to this effect. It is defined as a simple pic called `ejector`:

```
\pic at (0,0) {ejector};
```

and yields a square, in which centre there is the anchor, jointed with a cone frustum:



where the measure on the top indicates the distance from the outlet of the ejector to its anchor.

None of the common nodes are defined for the `ejector` since its symbols recall its real shape, where every point has a specific function. The driving stream enters from the `main inlet` node, abbreviated as `mi`, while the secondary one is aspired through the `suck inlet bottom` node or the `suck inlet top` node, abbreviated as `sib` and `sit` respectively. The two streams, mixed together, leave the unit from the `outlet` node, identified as `o` in the above drawing.

4.4 Heat Exchangers

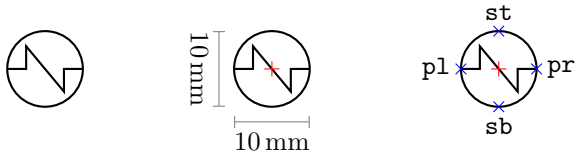
Many variants of heat exchangers are defined by chemplants. Among "simple symbols", two heat exchangers are available to general purposes, while three more are meant to be used to represent specific operations. There are also some symbols reserved to represent specific machines.

Heat Exchanger

The simplest possibile heat exchanger is useful to indicate thermal energy transfer between fluids which do not undergo phase change. It is defined as a simple pic called `heat exchanger`:

```
\pic at (0,0) {heat exchanger};
```

and yields a circle, in which centre there is the anchor, with a sketch of the path of internal pipes crossing the symbol in horizontal:



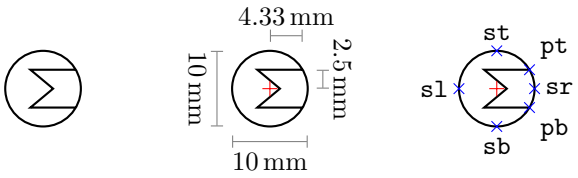
The `heat exchanger` has none of the common nodes, but it has some special ones. The stream crossing the exchanger through its internal pipes can be connected to the `pipes left` node, abbreviated in the picture above as `pl`, and to the `pipes right` node, abbreviated in the picture above as `pr`. In the same way, the stream crossing the exchanger through its shell can be connected to the `shell bottom` node, abbreviated in the picture above as `sb`, and to the `shell top` node, abbreviated in the picture above as `st`.

Heat Exchanger BiPhase

An alternative heat exchanger is useful to indicate thermal energy transfer between fluids where one of them, usually the main process stream, partially changes phase (but it can also be used in the place of a simple heat exchanger without any phase change). It is defined as a simple pic called `heat exchanger biphase`:

```
\pic at (0,0) {heat exchanger biphase};
```

and yields a circle, in which centre there is the anchor, with a sketch of the path of internal pipes which enter and leave the unit on the right side:



where measures on the right and on the top indicate the distances from the middle of the heat exchanger to the point where the internal pipes patch touches the boundary of the circle. It is also useful to remember that the same point can be identified in polar coordinates and lies 30° and 5 mm from to centre of the circle.

The `heat exchanger biphase` has none of the common nodes, but it has some special ones. The stream crossing the exchanger through its internal pipes can be connected to the `pipes bottom` node, abbreviated in the picture above as `pb`, and to the `pipes top` node, abbreviated in the picture above as `pt`. In the same way the stream crossing the exchanger through its shell can be connected to four nodes:

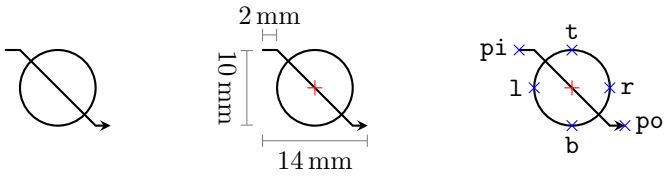
- the `shell left` node, abbreviated in the picture above as `sl`;
- the `shell bottom` node, abbreviated in the picture above as `sb`;
- the `shell right` node, abbreviated in the picture above as `sr`;
- the `shell top` node, abbreviated in the picture above as `st`.

Boiler and Condenser

The other two heat exchangers defined are the boiler and the condenser. Even though the name should be self explicative, the boiler is useful to vaporise a liquid, while the condenser is useful to condense a vapour (both totally or partially). The boiler is defined as a simple pic called `boiler`:

```
\pic at (0,0) {boiler};
```

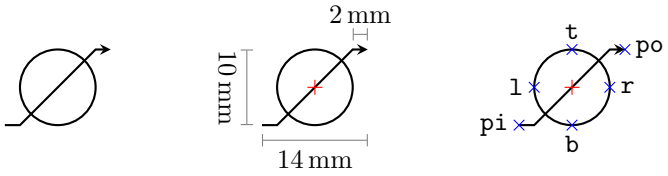
and yields a circle, in which centre there is the anchor, with a sketch of the path of internal pipes that crosses the circle falling down from the left to the right:



while the condenser is defined as a simple pic called `condenser`:

```
\pic at (0,0) {condenser};
```

and yields a circle, in which centre there is the anchor, with a sketch of the path of internal pipes that crosses the circle rising up from the left to the right:



Both the `boiler` and the `condenser` have some special nodes. Among the common ones, only `left`, `bottom`, `right` and `top` are defined (and here it is not explicitly specified that these nodes belong to the shell because of the more generality of the two pics). In addition there are two nodes more: a node called `pipes inlet`, abbreviated in the picture above as `pi`, and a node called `pipes outlet`, abbreviated in the picture above as `po`.

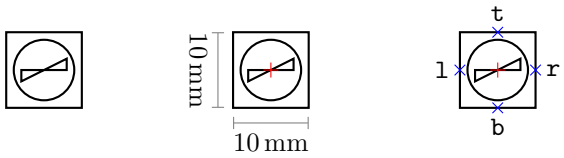
Boiler and condenser deserve a couple of words more. When looking at schemes coming from different sources, especially from different countries, understanding which one is the boiler and which one is the condenser is not obvious at all. In the UNICHIM regulation the arrows crossing the exchangers represent somehow the energy level variations of the utility fluids: for example, in a condenser the energy of the auxiliary fluid rises due to the enthalpy of condensation released by the vapour. In some books (especially if they come from the USA) it is not that rare to see the boiler symbol used in the place of the condenser one: in this case it should be interpreted as "the vapour is knocked down as a liquid". This can lead to confusion and misunderstandings, hence it is advisable to define preventively and clearly what are the meanings of the symbols used.

Air Heat Exchanger

Another kind of special heat exchanger defined by chemplants is a simple fan that blows on pipes, technically known as air heat exchanger. It is defined as a simple pic called `air heat exchanger`:

```
\pic at (0,0) {air heat exchanger};
```

and yields a square, in which centre there is the anchor, with a sketch of the fan:



The just introduced `air heat exchanger` is the last example of "simple symbol" defined as heat transfer equipments. This concept means that the units can be used to generic representation purposes, especially the `heat exchanger` and the `heat exchanger biphase`.

It is not seldom to find also more complex and realistic representation of equipments, in fact also UNICHIM indicates some variants of the simple symbols to be used to give a better idea of the unit or to identify a specific equipment in a more general family. This should not be a surprise after the many symbols to represent fluids handling machines discussed above.

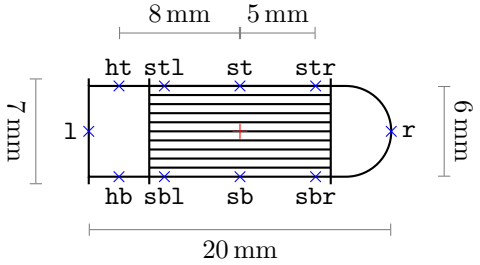
Tube Bundle Heat Exchanger

The most widely used heat transfer equipment in the chemical industry is for sure the tube bundle heat exchanger, made up of a bundle of pipes enclosed into a shell. It is defined as a simple pic called `tube bundle heat exchanger`:

```
\pic at (0,0) {tube bundle heat exchanger};
```

and yields a horizontal rectangle, in which centre there is the anchor, with a rounded end and a sketch of the internal pipes:

Only the pure symbol has been shown because, just like for the already discussed `multistage compressor`, there are a lot of measures and custom nodes to show:



Again, the symbol is bigger, but measures are referred to the one with the right dimensions. Measures should be clear. The only remark regards measures on the sides of the unit: the left one indicates the total height, while the right one ignores the little protrusions of the vertical lines.

Only the `left` and `right` nodes are defined among the common ones. The others are aimed to identify some remarkable points of the two main sections of the `tube bundle heat exchanger`: the head of the tube bundle and the shell that encloses it. Two nodes are defined for the head: the `head bottom` node, shown above as `hb`, and the `head top` node, shown above as `ht`. The fluid that passes through the internal pipes have to be connected using these nodes. Six more nodes are defined for the shell:

- the `shell bottom left` node is abbreviated in the drawing as `sbl`;
- the `shell bottom` node is abbreviated in the drawing as `sb`;
- the `shell bottom right` node is abbreviated in the drawing as `sbr`;
- the `shell top right` node is abbreviated in the drawing as `str`;
- the `shell top` node is abbreviated in the drawing as `st`;
- the `shell top left` node is abbreviated in the drawing as `stl`.

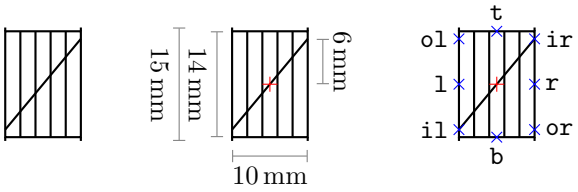
So many nodes are defined for the shell to allow the representation to be as flexible as possibile, in fact flow configuration can have a great impact on the heat exchanger performances and it is often useful to represent it also graphically.

Plate Heat Exchanger

The tube bundle heat exchanger is the most widely used, but it is not the only one. Another equipment useful to transfer thermal energy is the plate heat exchanger. It is defined as a simple pic called `plate heat exchanger`:

```
\pic at (0,0) {plate heat exchanger};
```

and yields a rectangle, in which centre there is the anchor, with a sketch of the plates pushed together:



where the measure on the inner left indicates the height of the rectangle without the small protruding vertical lines, while the measure on the right indicates the distance from the centre of the rectangle to the end of the oblique line.

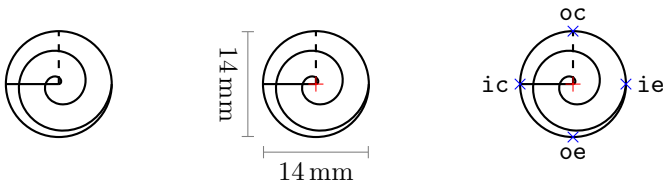
Some special nodes are defined for the `plates heat exchanger`. The construction of the machine does not imply the existence of two well defined chambers, but the two fluids travel into alternate plates. Anyway, an oblique line is sketched on the unit to identify one of the paths, which ends fall on `inner left` and `inner right` nodes, abbreviated above as `il` and `ir` respectively. The other fluid can be connected to the `outer left` node, abbreviated above as `ol`, and to the `outer right` node, abbreviated above as `or`. The remaining nodes should not be used for streams connections.

Spiral Heat Exchanger

Another common heat exchanger, somehow the "wrapped version" of the plate heat exchanger, is the spiral heat exchanger. It is defined as a simple pic called `spiral heat exchanger`:

```
\pic at (0,0) {spiral heat exchanger};
```

and yields a circle, in which centre there is the anchor, with a sketch of the spiral path:



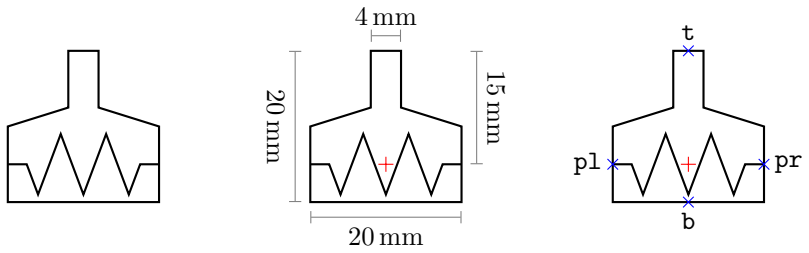
The `spiral heat exchanger` has none of the common nodes, but it has some special ones. Also in this case it is not possibile to individuate an internal and an external chamber, but this time the paths of the fluids are continuous and one is effectively internal to the other. Usually, one of the end of the path is placed on the centre of the cross section of the unit, while the other one is on its boundary: this justifies the names of the nodes. The `inner center` node is abbreviated above as `ic`, while the `inner edge` node is abbreviated above as `ie`; these two nodes should be used to connect the inner fluid. In the same way, but for the outer fluid, the `outer center` node is abbreviated above as `oc`, while the `outer edge` node is abbreviated above as `oe`.

Pipe Furnace

The last heat transfer equipment defined by chemplants (at least in this section of the manual) is not properly a heat exchanger, but it is a furnace. It is defined as a simple pic called `pipe furnace`:

```
\pic at (0,0) {pipe furnace};
```

and yields a rectangle, in which centre there is the anchor, with a sketch of the path of the internal pipes crossing the symbol in horizontal and a representation of the furnace chimneystack:



where the measure on the right indicates the distance from the anchor point to the top of the chimneystack.

The `pipe furnace` has some special nodes. Among the common ones, only `bottom` and `top` are defined and they may be useful to represent fuel inlet and stack gases outlet. In addition, there are two nodes more: a node called `pipes left`, abbreviated in the picture above as `pl`, and a node called `pipes right`, abbreviated in the picture above as `pr`.

4.5 Separators

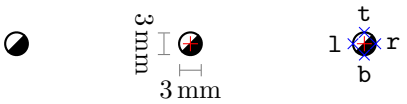
Separation unit operations are some of the most important operations in chemical engineering. Besides the separation of homogenous mixtures, that is usually done in "simple" equipments which will be introduced in the following, the separation of heterogeneous mixtures is practically much more simple and some specific equipments based on mechanical principles exist. Also more complex separation units will be introduced after the mechanical ones, mainly the ones based on thermal energy supply or removal and aimed to separate special kinds of homogenous mixtures. (Notice that the terms gas and vapour will be used indiscriminately, although not properly correct.)

Steam Trap

When the condensation of steam or of some other vapours is involved, it is always necessary to withdraw a liquid from somewhere. To avoid the steam to escape from the liquid outlet, a stream trap can be profitably used. It is defined as a simple pic called `steam trap`:

```
\pic at (0,0) {steam trap};
```

and yields a little half field circle anchored in its centre:



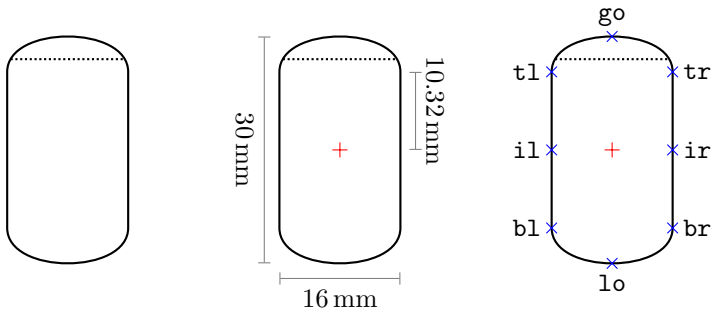
In the default orientation, which is the one commonly used in process diagrams and should not be changed, the inlet of the steam trap have to touch the empty half of the circle, while the outlet have to come out from the filled half.

Gas-Liquid Separator

When more sophisticated separations of gases and liquids mixtures are needed another unit is useful: the gas-liquid separator. It is defined as a simple pic called `gas-liquid separator`:

```
\pic at (0,0) {gas-liquid separator};
```

and yields a tank, in which centre there is the anchor, with a sketch of the anti-entrainment system on its top:



where the measure on the right indicates the distance from the middle of the tank to the point where the curvature begins.

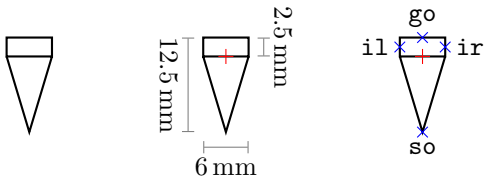
Besides the common nodes, special nodes are defined for stream connections. In the above drawing, `il` and `ir` indicate respectively the `inlet left` node and the `inlet right` node, which should be used to connect the inlet stream. On the top of the unit there is the `gas outlet` node, abbreviated as `go` in the figure, while on its bottom there is the `liquid outlet` node, abbreviated as `lo` in the figure. Names should be self-explicative.

Cyclone

A cyclone is a unit useful to separate solid powders entrained by a gas stream. It is defined as a simple pic called `cyclone`:

```
\pic at (0,0) {cyclone};
```

and yields a little rectangle, on which base there is the anchor, attached to a triangular funnel:



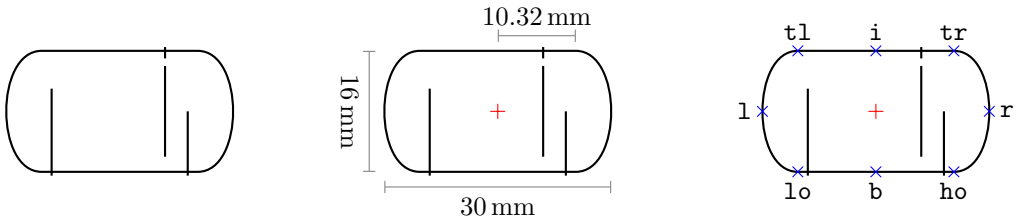
None of the common nodes are defined for the `cyclone`, but there are special nodes to be used to connect specific streams. In the above drawing, `il` and `ir` indicate respectively the `inlet left` node and the `inlet right` node, which should be used to connect the inlet stream. On the top of the unit there is the `gas outlet` node, abbreviated as `go` in the figure, while on its bottom there is the `solid outlet` node, abbreviated as `so` in the figure. Names should be self-explicative.

Stratifier

A common operation to be carried out in chemical processes is the separation of two immiscible liquids. If they have different densities, time and gravity will do the job, so a stratifier can be used. It is defined as a simple pic called `stratifier`:

```
\pic at (0,0) {stratifier};
```

and yields a horizontal tank, in which centre there is the anchor, with a sketch of the internal baffles:



where the measure on the top indicates the distance from the middle of the tank to the point where the curvature begins.

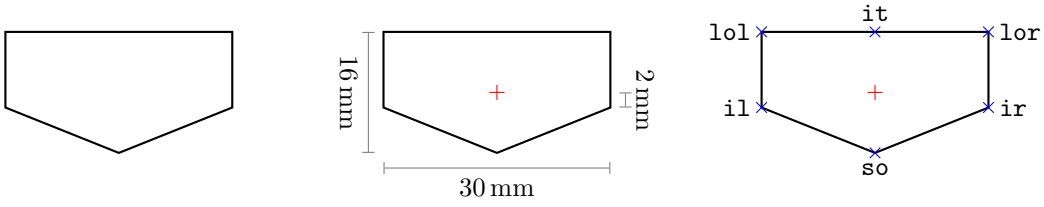
Besides the common nodes, special nodes are defined for stream connections. The inlet stream should enter from the `inlet` node, marked as `i` in the figure, and, after the stratification, the light liquid comes out from the `light outlet` node, marked as `lo` in the figure, while the heavy liquid comes out from the `heavy outlet` node, marked as `ho` in the figure.

Settler

Previously, the cyclone have been introduced. A unit with the same purposes, but useful when a solid suspended into a liquid has to be separated, is the settler. It is defined as a simple pic called `settler`:

```
\pic at (0,0) {settler};
```

and yields a horizontal rectangle, in which centre there is the anchor, with a triangular funnel end:



where the measure on the right indicates the distance from the middle of the unit to the point where the funnel begins.

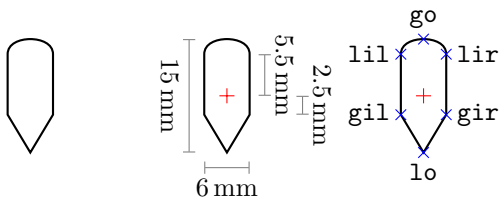
None of the common nodes are defined for the `settler`, but there are special nodes to be used to connect specific streams. In the above drawing, `il`, `ir` and `it` indicate respectively the `inlet left` node, the `inlet right` node and the `inlet top` node, which should be used to connect the inlet stream. On the top left of the unit there is the `liquid outlet left` node, abbreviated as `lol` in the figure, while on the top right of the unit there is the `liquid outlet right` node, abbreviated as `lor` in the figure. Finally, on the bottom of the unit there is the `solid outlet` node, abbreviated as `so` in the figure. Three inlets are defined to give flexibility to the representation, in fact the same symbol can be used to indicate circular settling basins as well as rectangular settling basins.

Scrubber

Another case in which the separation of a "gas-liquid" mixture is needed is the one where a condensable vapour must be knocked out from a gas stream by condensation, or when liquid droplets are entrained by a gas stream. There are a lot of possibilities to achieve this operation, but in some particular cases a simple wash of the gas using water (or some appropriate solvent) will be enough, thus a scrubber can be used. It is defined as a simple pic called `scrubber`:

```
\pic at (0,0) {scrubber};
```

and yields a little tank, in which centre there is the anchor, with a triangular funnel end:



where the measure on the top right indicates the distance from the middle of the tank to the point where the curvature begins, while the measure on the bottom right indicates the distance from the middle of the tank to the point where the funnel begins.

None of the common nodes are defined for the `scrubber`, but there are special nodes to be used to connect specific streams. In the above drawing, `gil` and `gir` indicate respectively the `gas inlet left` node and the `gas inlet right` node, which should be used to connect the inlet gas stream. In the same way `lil` and `lir` indicate respectively the `liquid inlet left` node and the `liquid inlet right` node, which should be used to connect the inlet liquid stream. On the top of the unit there is the `gas outlet` node, abbreviated as `go` in the figure, while on its bottom there is the `liquid outlet` node, abbreviated as `lo` in the figure. Names should be self-explicative.

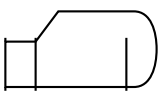
As a final remark, it should be noticed that the `scrubber` can be used also to represent a barometric condenser: it is sufficient to place it high enough above the ground reference and to draw a long vertical stream coming out from the `liquid outlet` to indicate the drain pipe.

Kettle Boiler

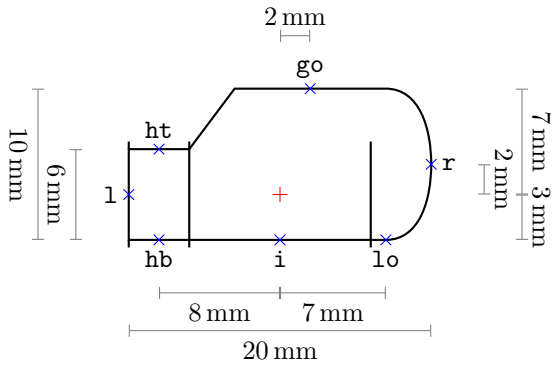
The scrubber is the first example of separation unit based on thermal energy exchange. In that case, the transfer is achieved by direct contact of the inlet gas stream using a liquid, but it is more common to use non-contact systems in which a pure energy transfer is achieved. One of the simplest, this time useful to concentrate a solution of a non-volatile solute (or to simply partially boil a liquid), is the kettle boiler. It is defined as a simple pic called `kettle boiler`:

```
\pic at (0,0) {kettle boiler};
```

and yields a strage symbol which recalls a tube bundle heath exchanger with an enlarged room for the vapour:



Only the pure symbol has been shown because, just like for the already discussed `tube bundle heat exchanger`, there are a lot of measures and custom nodes to show:



Again, the symbol is bigger, but measures are referred to the one with the right dimensions.

Only the `left` and `right` nodes are defined among the common ones. The others are aimed to identify some remarkable points of the two main sections of the `kettle boiler`: the head of the tube bundle and the shell of the boiling chamber. Two nodes are defined for the head: the `head bottom` node, shown above as `hb`, and the `head top` node, shown above as `ht`. Here the heating fluid has to be connected. Three more nodes are defined for the boiling chamber: the `inlet`, shown above as `i`, the `gas outlet`, shown above as `go`, and the `liquid outlet`, shown above as `lo`. Their usage should be clear.

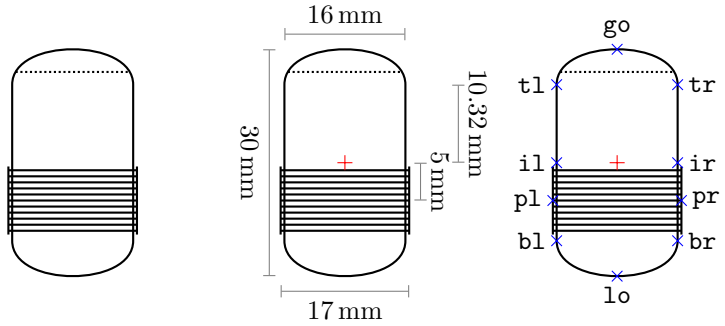
The kettle boiler is the simplest machine belonging to the family of evaporators. These units, like the name says, are meant to vaporise a liquid, so they are a special kind of heat exchangers. Anyway, they are described together with the separators because they are mainly used to concentrate solutions of non-volatile or poorly-volatile solutes, hence "separating" a part of the solvent from the solution.

Tube Bundle Evaporator

A slightly more complex evaporator, which indeed works in the same way of a kettle boiler, is the tube bundle evaporator. It is defined as a simple pic called `tube bundle evaporator`:

```
\pic at (0,0) {tube bundle evaporator};
```

and yields a tank, in which centre there is the anchor, with sketches of the tube bundle and of the anti-entrainment system on its top:



where the measure on bottom indicates the total width of the unit, the measure on top indicates the width of the tank, the measure on the right indicates the distance from the middle of the tank to the point where the curvature begins and the measure on bottom right refers to the distance from the anchor point to the middle of the pipe bundle.

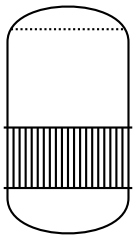
Some of the special nodes defined should be clear at this point, at least `inlet left`, `inlet right`, `gas outlet` and `liquid outlet`. Two more special nodes are defined for this pic: `pipes left`, abbreviated above as `pl`, and `pipes right`, abbreviated above as `pr`.

Basket Evaporator and Climbing Film Evaporator

A different evaporator is the one that uses the basket system. In this case the heating fluid passes through the shell of the tube bundle, while in the former one it passes through the pipes. It is defined as a simple pic called `basket evaporator`:

```
\pic at (0,0) {basket evaporator};
```

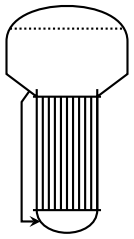
and yields a tank, in which centre there is the anchor, with sketches of the tube bundle and of the anti-entrainment system on its top:



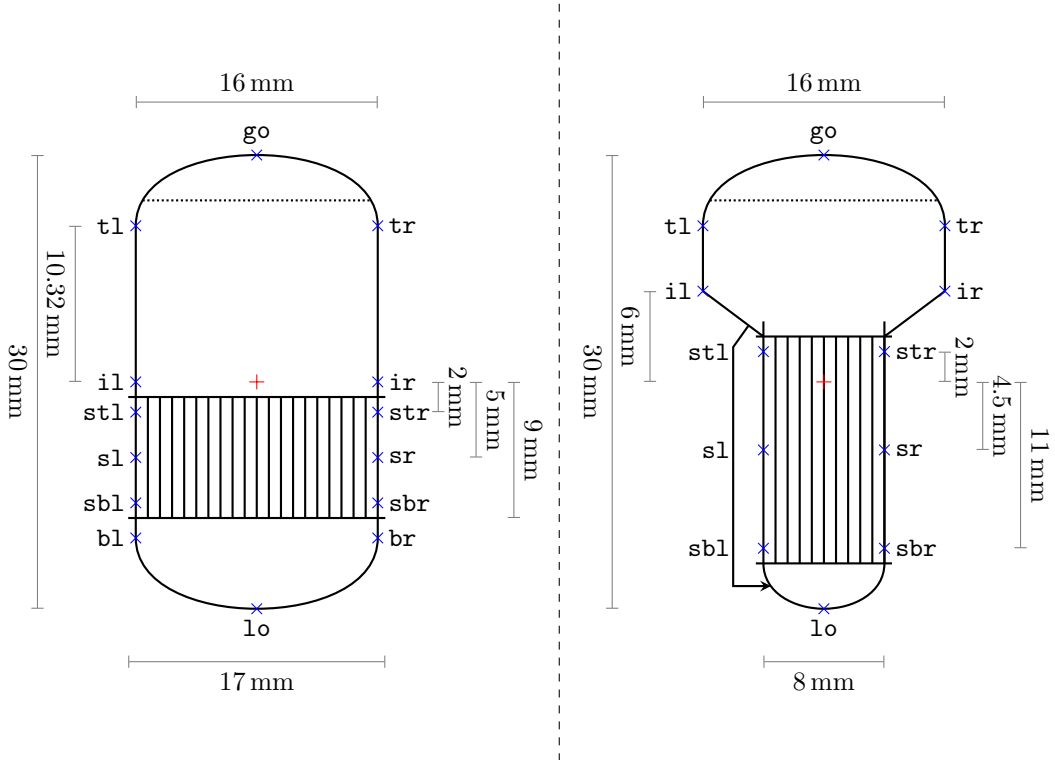
The `basket evaporator` has a pretty complex nodes structure, so it will be described later on, together with a unit that is defined more or less with the same nodes: the climbing film evaporator. It is defined as a simple pic called `climbing film evaporator`:

```
\pic at (0,0) {climbing film evaporator};
```

and yields a strange shape, , in which centre there is the anchor, with sketches of the tube bundle, of the anti-entrainment system on its top and of the internal liquid recirculation path:



In order to better understand, larger versions of both the `basket evaporator` and the `climbing film evaporator` are shown below with full markings of measures (which values, as usual, refer to the dimension of real units) and nodes:



Measures should be clear enough. The only remark is that the measure on the bottom of the `climbing film evaporator` refers to the width of the tube bundle part of the unit without taking into account the little protruding lines. Always for this unit, labels of the three nodes on the lower left part are far from their marks only to avoid overlaps with the internal recirculation path.

For both units there are two of the common nodes: `top left` and `top right`, while for the `basket evaporator` there are also `bottom left` and `bottom right`. Other nodes are common to both units. Two main inlet nodes: `inlet left` and `inlet right`, abbreviated above as `il` and `ir` respectively. Two outlet nodes: `gas outlet` and `liquid outlet`, abbreviated above as `go` and `lo` respectively. The remaining nodes are to be used to connect the heating fluid to the shell of the pipe bundle:

- the `shell top left` node is abbreviated in the drawing as `stl`;
- the `shell left` node is abbreviated in the drawing as `sl`;
- the `shell bottom left` node is abbreviated in the drawing as `sbl`;
- the `shell bottom right` node is abbreviated in the drawing as `sbr`;
- the `shell right` node is abbreviated in the drawing as `sr`;
- the `shell top right` node is abbreviated in the drawing as `str`.

Just like for the `tube bundle heat exchanger`, so many nodes are defined for the shell to allow the representation to be as flexible as possibile, in fact flow configuration can have a great impact on the evaporator performances and it is often useful to represent it also graphically.

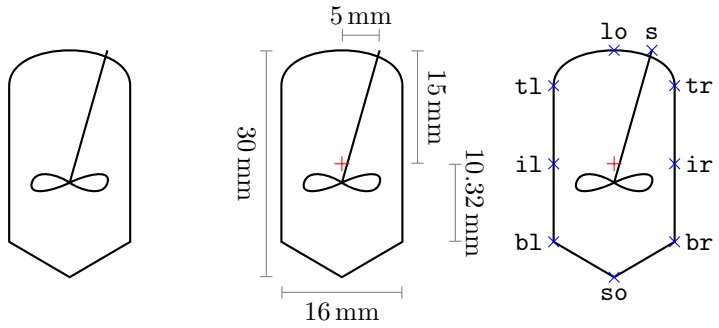
As a last remark on these two units, often also the `basket evaporator` has an internal recirculation path, but this is not marked on the default symbol. If one wants to show it, this should be done drawing an arrow of `thick` thickness and with a `stealth` tip from the `bottom right` node to the `inlet right` node (or the same using nodes on the left).

Stirred Crystallizer

The last kind of separator defined by chemplants is often used in conjunction with an evaporator and can push the operation of concentration far enough to achieve the saturation of the solution, causing consequently the precipitation of solids. A simple non-thermal crystallizer is a stirred tank with a cone-shaped bottom useful to collect solids in the direction of the outlet pipe. It is defined as a simple pic called `stirred crystallizer`:

```
\pic at (0,0) {stirred crystallizer};
```

and yields a vertical tank, in which centre there is the anchor, with the sketch of a mechanical stirrer:



where the measure on the bottom-right indicates the distance from the middle of the tank to the point where the cone begins, while measures on the top right indicate the distances from the stirrer end point to the anchor point in the middle of the tank.

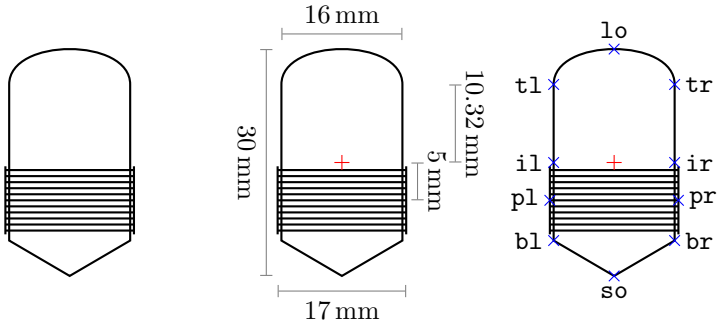
The end of the stirrer outside the tank is also identified by a special coordinate node, indicated in the above drawing as `s` and called `shaft`. It should be evident that this is not a point defined to connect streams, but it may be useful for some kind of control systems. In addition to this node, also the `inlet left` node and `inlet right` node are defined, shown above as `il` and `ir` respectively, and finally the `liquid outlet` node and `solid outlet` node, shown above as `lo` and `so` respectively.

Tube Bundle Crystallizer

A stirred crystallizer can be used to sketch diagrams of crystallization operation held in multiple units, such as a cooling in a heat exchanger and the following crystallization in a proper equipment. However, there is also a simple all-in-one machine that realizes heath exchange and crystallization at the same time. It is defined as a simple pic called `tube bundle crystallizer`:

```
\pic at (0,0) {tube bundle crystallizer};
```

and yields a tank, in which centre there is the anchor, with sketches of the tube bundle:



where the measure on bottom indicates the total width of the unit, the measure on top indicates the width of the tank, the measure on the right indicates the distance from the middle of the tank to the point where the curvature begins and the measure on bottom right refers to the distance from the anchor point to the middle of the pipe bundle.

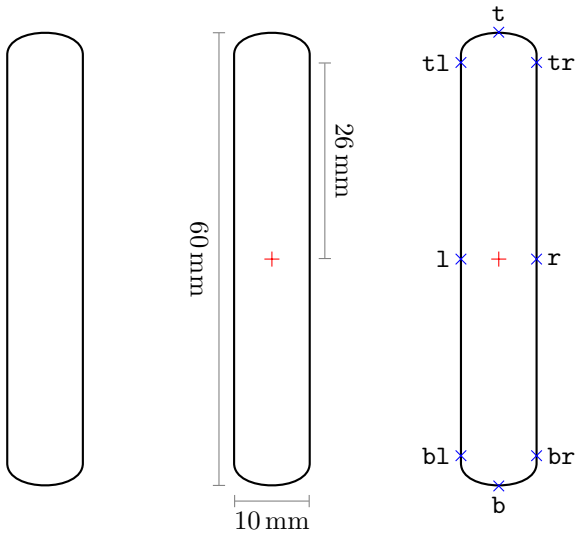
Some of the special nodes defined should be clear at this point, at least `inlet left`, `inlet right`, `liquid outlet` and `solid outlet`. Two more special nodes are defined for this pic: `pipes left`, abbreviated above as `pl`, and `pipes right`, abbreviated above as `pr`.

4.6 Columns

A column is the most characteristic piece of equipment of the chemical industry and is a vertical pipe with large diameter featured by various types of internals; it can be used in a wide set of applications. It is defined as a pic with arguments called `column`:

```
\pic at (0,0) {column=empty};
```

and yields a vertical column anchored in its centre:

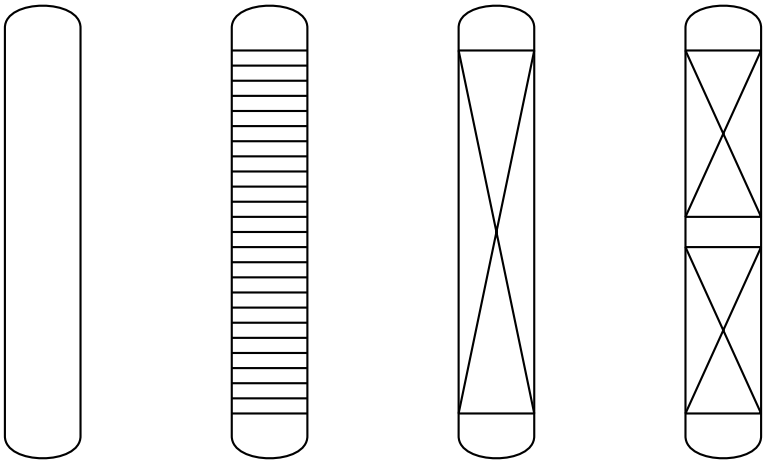


where the measure on the right indicates the distance from the middle of the column to the points where the nodes are placed. It should be noticed that this point is not where the curvature begins, like in tank-shaped units. The curvature begins 27 mm above the middle of the unit (in my opinion, a better graphical effect is obtained with a stream placed slightly below this point). Similar considerations apply the similar nodes on the bottom.

Four kinds of columns are defined: `empty`, `trayed`, `packed` and `packed double`, which represent respectively an empty column, a column with trays, a column with a single packed zone and a column with two packed zones (useful to represent distillation in packed columns). These keys should be used as arguments of the `column` pic. The code:

```
\pic at (0,0) {column=empty};
\pic at (3,0) {column=trayed};
\pic at (6,0) {column=packed};
\pic at (9,0) {column=packed double};
```

yields:



All of the columns have the same dimensions and the anchor mark is always at the centre of the pic (they are all defined as variants of the `empty` column). Finally, it is useful to remember that trays (or packings) start 24 mm above the middle of the column and are spaced 2 mm each.

4.7 Reactors

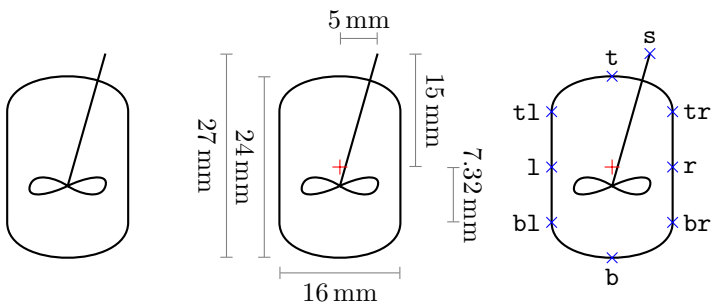
Besides columns, reactors are the other most characteristic equipments of the industrial chemistry, for which is essential to realize a reaction, but on a very large scale. The variety of reactors available in the industry is wide-ranged in every aspect: shape, phases within the reactor, operating principles, stream configurations and so on. For this reason, it is impossibile to summarise all of the existing reactors using one symbol only, but it is anyway possible to represent the most common ones in terms of operating principled, which is what is done by chemplants.

Stirred Reactor

One of the most common representation of a reactor used in chemical engineering is a simple vessel provided with a stirrer sketch, which is the core of the perfectly mixed reactor models. This kind of symbol is defined as a simple pic called `stirred reactor`:

```
\pic at (0,0) {stirred reactor};
```

and yields a vertical, in which centre there is the anchor, with the sketch of a mechanical stirrer:



Reading measures is not that easy, so it is better to give some clarification: the pic have an overall height of 27 mm and an overall width of 16 mm, while the tank (without the stirrer protruding part) is 24 mm high; the measure on the bottom right indicates the distance from the middle of the tank to the point where the curvature begins, while

measures on the top right indicate the distances from the stirrer end point to the anchor point in the middle of the tank.

The end of the stirrer outside the tank is also identified by a special coordinate node, indicated in the above drawing as `s` and called `shaft`. It should be evident that this is not a point defined to connect streams, but it may be useful for some kind of control systems.

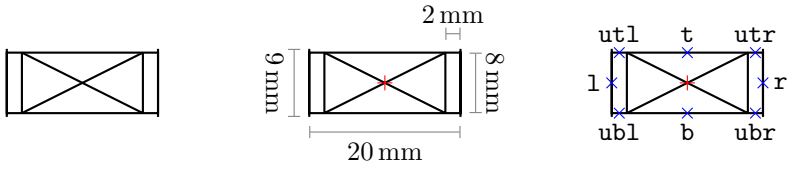
A stirred reactor can be used to represent two ideal reactor models among the most commonly used in the chemical engineering: the perfectly mixed batch reactor and the continuous stirred tank reactor (CSTR). Anyway, a `stirred reactor` is a very generic representation, so it will be appropriate for all of the reactors in which there is a mixer, regardless from the mixer type or of the phases within the reactor.

Packed Bed Crystallizer

The third most common ideal reactor model, the plug flow reactor (PFR), can be functionally represented by a packed bed reactor, that is clearly useful to represent also a real packed bed reactor. It is defined as a simple pic called `packed bed reactor`:

```
\pic at (0,0) {packed bed reactor};
```

and yields a rectangular reactor, in which centre there is the anchor, with the representation of the inside packing:



where the measure on the right indicates the height of the rectangle without the small protruding vertical edges, while the measure on the top indicates the width of the section without the packing.

The `packed bed reactor` has some special nodes. Among the common ones, only `left`, `bottom`, `right` and `top` are defined. In addition, there are four nodes more:

- a node called `utility bottom left`, abbreviated in the picture above as `ubl`;
- a node called `utility bottom right`, abbreviated in the picture above as `ubr`;
- a node called `utility top right`, abbreviated in the picture above as `utr`;
- a node called `utility top left`, abbreviated in the picture above as `utl`.

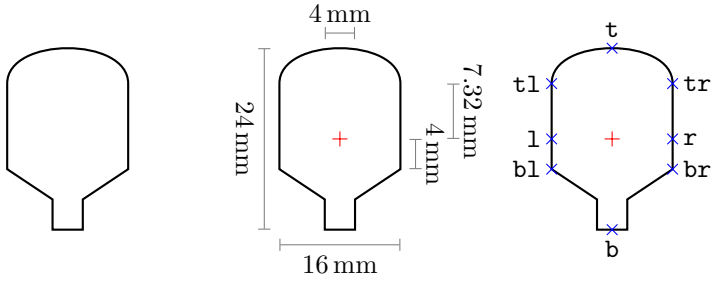
These nodes are useful to simulate the presence of a jacket associated to the reactor and can be used, for example, to connect utility streams carrying fluids of a temperature control system.

Fluidized Bed Reactor

Even though the two above mentioned reactors are the most common ones, there are many others and one example is the fluidized bed reactor. Its usage have the same aims of the packed bed reactor, but this time solids are free to move within the reactor entrained and mixed by the gas. It is defined as a simple pic called `fluidized reactor`:

```
\pic at (0,0) {fluidized bed reactor};
```

and yields a half tank-shaped reactor anchored in its centre:



where the measure on the right indicates the distance from the middle of the tank to the point where the curvature begins.

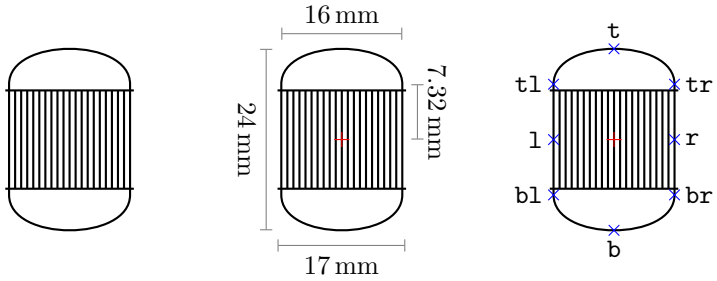
No special node is defined for the `fluidized reactor`. Anyway, it should be noticed that the restriction of the bottom represents a real section change in the tank, which allows the bed to be firstly entrained by a fast flow and then to fall down due to the slowing of the stream caused by the expansion of the cross section. For this reason, the main inlet should be placed on the `bottom` node, while the main outlet should be connected to the `top` node. Other nodes, in particular `bottom left`, `bottom right`, `top right` and `top left`, can be used for connections if one needs to represent a solid stream going through the reactor.

Tube Bundle Reactor

Another possibility to represent a reacting system is a tube bundle reactor, which can be used as a film reactor, thus to contact a liquid stream falling inside the reactor in the form of a film distributed on the walls of the internal pipes and a gas stream flowing in the internal spaces of the pipes, or as a multi-pipe packed bed reactor. It is defined as a simple pic called `tube bundle reactor`:

```
\pic at (0,0) {tube bundle reactor};
```

and yields a tank-shaped reactor, in which centre there is the anchor, with the sketch of the internal pipes:



where the measure on the right indicates the distance from the middle of the tank to the point where the curvature begins, while the measure on the top indicates the width of the reactor without the small protruding horizontal lines which represent the support plates of internal pipes.

No special node is defined for the `tube bundle reactor`, but it is a good idea to specify that `left` and `right` nodes identify two points which fall between the support plates of internal pipes, so they are the "access" to the outer side of the pipe bundle. For this reason these two points are perfect to connect utility streams carrying fluids of a temperature control system. Unlike evaporators, there are not so many nodes for the

shell of the pipes: an evaporator is a purely thermal unit and the flow configuration can be an important information to show, while for a reactor it is not that important.

Finally, another useful information is that both support plates of the pipes are placed to a vertical distance of 6.5 mm from the anchor point, even though not explicitly marked in the above drawing.

4.8 Associative Pics for Reactors

The just introduced reactors are only a little part of the huge variety of symbols one can find looking at process schemes. It is almost impossible to accomplish all of the requests about the representation of reactors: one wants to draw a jacket, another one wants a stirrer and a jacket, another one wants a stirrer but no jacket, another one wants a sprayer, a stirrer and a jacket and so on.

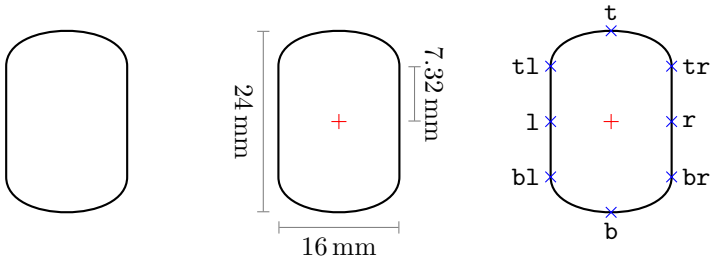
Even though the number of possibile combinations is huge, a smart and efficient solution can always be found. A possibile way is what I like to call associative pics, a palette of pics which can be overlapped and bonded together thanks to anchors. In this way, only the building blocks need to be defined and then users can play with them to produce symbols they like the most, or the ones they need. It should be clear that some pics will exclude the usage of others (for example a stirrer should not be represented in a packed bed reactor).

Tank Reactor

It is necessary to specify that this technique is adopted in chemplants only for reactors, more precisely for tank shaped reactors. The main building block is, in fact, a little tank which represents a generic empty reactor. It is defined as a simple pic called `tank reactor`:

```
\pic at (0,0) {tank reactor};
```

and yields a vertical tank anchored in its centre:



where the measure on the right indicates the distance from the middle of the tank to the point where the curvature begins.

This pic is formally analogue to the `tank`, the very first pic introduced, but smaller. It is useful to know that the scale factor to "convert" a `tank reactor` into a `tank` is 1.25. The reason why it is useful to know it will be explained in the future, when pics transformations will be introduced.

As told before, the `tank reactor` is the starting point of all of the representation. In particular, almost all of the other pics defined later on (apart two of them) can be bonded to the `tank reactor` using the `anchor` node. Assuming that the reactor is declared as:

```
\pic (R) at (0,0) {tank reactor};
```

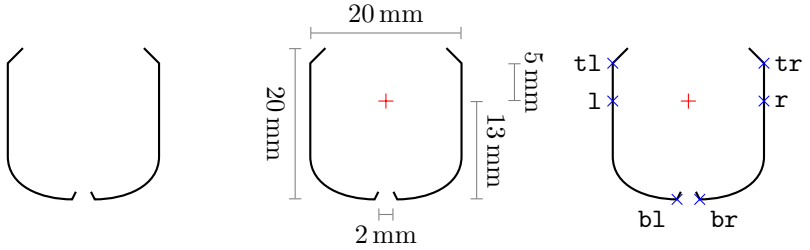
thus identified as `R`, the point in which other pics have to be declared is the node `R-anchor`.

Jacket

The first gadget for the `tank reactor` is a jacket for temperature control. It is defined as a simple pic called `jacket`:

```
\pic at (0,0) {jacket};
```

and yields the sketch of and external jacket anchored in its centre:

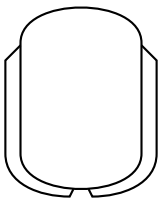


where the measure on the bottom right indicates the distance from the anchor of the jacket to its bottom, while the measure on the bottom indicates the width of the "bottom hole".

The `jacket` has to be bonded to the `tank reactor` using its `anchor` node:

```
\pic (R) at (0,0) {tank reactor};
\pic at (R-anchor) {jacket};
```

and the result is:



It should be noticed that the `jacket` extends the dimensions of the `tank reactor` by 2 mm in horizontal on both sides and by 1 mm in vertical on the bottom.

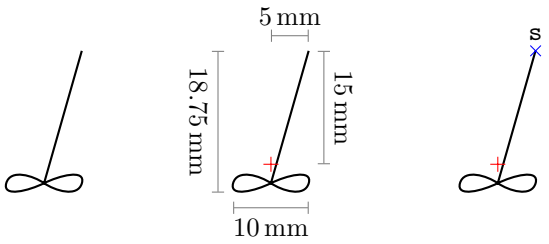
This solution is functional, but it introduces also a drawback: being the `jacket` and the `tank reactor` two different pics, they have to be identified using two different prefixes. In the above example, the node `R-left` identifies the `left` node of the reactor, but not the `left` node of the jacket. The drawback is that it is necessary to remember to use the correct prefix to identify nodes of the right pic, but this is also a safer approach.

Stirrer

The `jacket` is the only external gadget defined, but there are a lot of internal accessories, among which there is the already known stirrer. It is defined a simple pic called `stirrer`:

```
\pic at (0,0) {stirrer};
```

and yields the sketch of a mechanical stirrer:



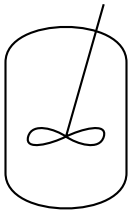
where measures on the right and on the top indicate the distances from the anchor of the stirrer to the end of its shaft.

The `stirrer` has a special node: the `shaft` node is placed at the end of the shaft and it is abbreviated above as `s`.

The `stirrer` has to be bonded to the `tank reactor` using its `anchor` node:

```
\pic (R) at (0,0) {tank reactor};
\pic at (R-anchor) {stirrer};
```

and the result is:



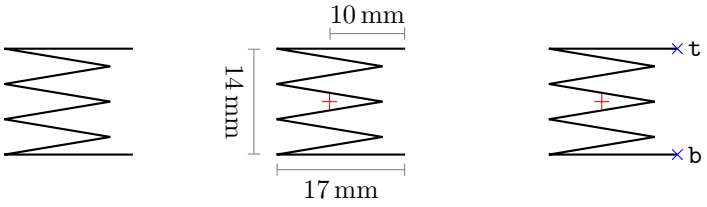
It should be noticed that the `stirrer` extends the dimensions of the `tank reactor` by 3 mm in vertical on the top. The pic obtained in this way is exactly the same of the `stirred reactor`, but this last one has still its own definition as a single pic due to its importance.

Coil

Instead of using a jacket to control the temperature of a reactor, there is another solution: a coil, either electrical or carrier of a temperature controlling fluid, placed within the reactor. It is defined as a simple pic called `coil`:

```
\pic at (0,0) {coil};
```

and yields the sketch of a coil anchored in its center:



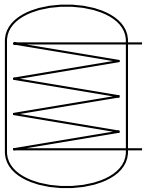
where the measure on the top indicates the distance from the anchor of the coil to the end of its path.

The two nodes declared for the `coil` are the `bottom` node and the `top` node, abbreviated above as `b` and `t` respectively.

The `coil` has to be bonded to the `tank reactor` using its `anchor` node:

```
\pic (R) at (0,0) {tank reactor};
\pic at (R-anchor) {coil};
```

and the result is:



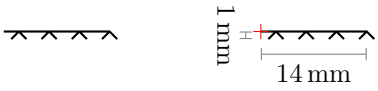
It should be noticed that the `coil` extends the dimensions of the `tank reactor` by 2 mm in horizontal on the right. Even though some overlapping is obtained, the `coil` and the `stirrer` can be used together.

Sprayer

When it is important to represent the distribution of a liquid fed to a reactor, a special pic comes in hand. It is defined as a simple pic called `sprayer`:

```
\pic at (0,0) {sprayer};
```

and yields the sketch of a sprayer anchored on its inlet end, the one on the right:

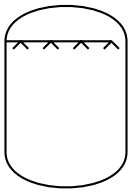


and no special or common nodes are defined for this pic, apart from the usual `anchor` node.

The `sprayer` has to be bonded to the `tank reactor` using its `top left` node (or its `top right` node using a trick that will be introduced in the following):

```
\pic (R) at (0,0) {tank reactor};
\pic at (R-top left) {sprayer};
```

and the result is:

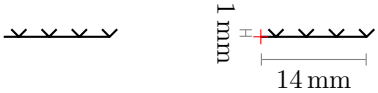


Bubbler

In the same way, when it is important to represent the distribution of a gas fed to a reactor, a special pic comes in hand. It is defined a simple pic called `bubbler`:

```
\pic at (0,0) {bubbler};
```

and yields the sketch of a bubbler anchored on its inlet end, the one on the right:

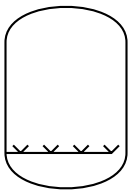


and no special or common nodes are defined for this pic, apart the usual `anchor` node.

The `bubbler` has to be bonded to the `tank reactor` using its `bottom left` node (or its `bottom right` node using a trick that will be introduced in the following):

```
\pic (R) at (0,0) {tank reactor};
\pic at (R-bottom left) bubbler;
```

and the result is:

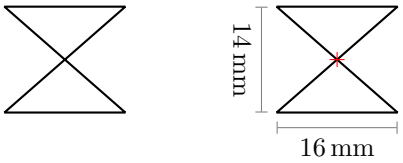


Packing

Finally, the last pic defined as a gadget for the `tank reactor` is useful to obtain an alternative representation of the `packed bed reactor`. It is defined as a simple pic called `packing`:

```
\pic at (0,0) {packing};
```

and yields the representation of a packing:

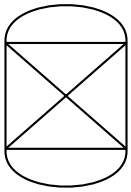


and no special or common nodes are defined for this pic, apart from the usual `anchor` node.

The `packing` has to be bonded to the `tank reactor` using its `anchor` node:

```
\pic (R) at (0,0) {tank reactor};
\pic at (R-anchor) {packing};
```

and the result is:



It should be clear enough that this pic has not to be used with any of the preceding ones, but, at most, with the `jacket`.

5 Process Utility Units

Process utilities is the name given to a set of special units useful to better specify what is going on in a process; such units are valves, pipe joints, equipment nozzles and so on. Among process utilities, there are also control instrumentation and "gadgets" for units to be used, for example, to let the control system automatically act on the process.

5.1 Valves

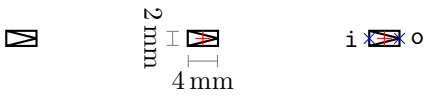
A utility valve should not be misconceived with a lamination valve (as it often happens). Despite the extremely common fact that the same symbol is used for both the units, for the sake of clearance UNICHIM indicates the lamination of a fluid through a valve using a different symbol with respect to the one used for regulation or interception valves.

Lamination Valve

A lamination valve is a unit useful to expand a fluid, hence to reduce its pressure. It can be represented using a simple pic called `lamination valve`:

```
\pic at (0,0) {lamination valve};
```

which yields a rectangle anchored in its centre in which there is an arrow sketch:



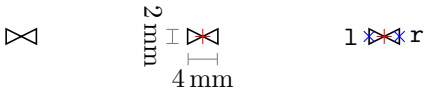
Nodes defined for the `lamination valve` are particular. A lamination valve is a unit much more similar to a simple electric bipole, in fact it has only two nodes (plus the `anchor`); moreover this unit is a "one way operation", so the nodes have a simple logic: one `inlet` and one `outlet`. These two are indicated in the drawing above as `i` and `o` respectively.

Valve

A generic valve, intended as a a regulation valve or as an interception valve, can be represented using a pic with arguments called `valve`:

```
\pic at (0,0) {valve=main};
```

which yields a horizontal valve anchored in its centre:

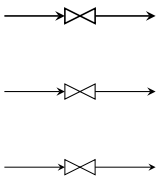


Note that the `anchor` node comes in hand with a `valve`. If one wants to specify that a valve has a sensor or an actuator (pics will be introduced in the following), its pic can be simply "bonded" to the valve placing it on the `anchor` node of the `valve`. For example, if a `valve` is identified as `V`, placing the actuator in the `V-anchor` node will do the job.

The `valve` pic is defined with an argument to make it sensible to the stream on which it is placed, in fact two kinds of valves are defined: `main`, `secondary` and `utility`, which are drawn respectively with `semithick` lines, `thin` lines and `very thin` lines. These keys should be used as arguments of the `valve` pic. The code:

```
\draw[main stream] (0,2) -- (0.8,2);
\pic at (1,2) {valve=main};
\draw[main stream] (1.2,2) -- (2,2);
\draw[secondary stream] (0,1) -- (0.8,1);
\pic at (1,1) {valve=secondary};
\draw[secondary stream] (1.2,1) -- (2,1);
\draw[utility stream] (0,0) -- (0.8,0);
\pic at (1,0) {valve=utility};
\draw[utility stream] (1.2,0) -- (2,0);
```

yields:



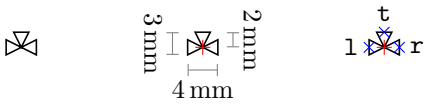
Conversely, `lamination valve` is defined as a simple pic because it is more similar to a unit operation rather than to a regulation valve, hence it is drawn with a `thick` line just like process units.

Three-Way Valve

Even though it is not properly a valve, the similarities of the symbols qualify a pipes joint also with the name of three-way valve. It is defined as a pic with arguments called `valve triple`:

```
\pic at (0,0) {valve triple=main};
```

which yields a horizontal valve anchored in its centre with the third way going out from the top:



where the measure on the right indicates the distance from the middle of the joint to its top.

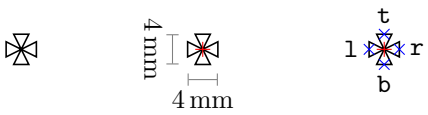
Just like for the valve, three kinds of three-way valves are defined: `main`, `secondary` and `utility`, which are drawn respectively with `semithick` lines, `thin` lines and `very thin` lines. These keys should be used as arguments of the `valve triple` pic.

Four-Way Valve

Just like three-way valve, also a joint of four pipes is not properly a valve, but the similarity of its symbols to the one used to denote valves gives it the informal name of four-way valve. It is defined as a pic with arguments called `valve quadruple`:

```
\pic at (0,0) {valve quadruple=main};
```

which yields a "cross valve" anchored in its centre:



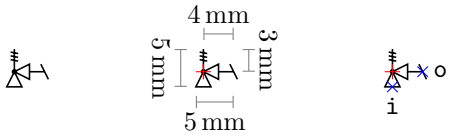
Just like for the valve, three kinds of four-way valves are defined: `main`, `secondary` and `utility`, which are drawn respectively with `semithick` lines, `thin` lines and `very thin` lines. These keys should be used as arguments of the `valve quadruple` pic.

Safety Valve

A special kind of valve is the one used to protect a vessel against pressure anomalies: the safety valve. The chemplants package defines a single symbols to represent both safety valves and relief valves. It is defined as a pic with arguments called `safety quadruple`:

```
\pic at (0,0) {safety valve=main};
```

which yields a bent valve, in which central node there is the anchor, with sketches of the spring and of the vent:



where the measure on the right indicates the distance from the anchor to the top of the spring, while the measure on the measure on the top indicates the distance from the anchor to the vent.

Nodes defined for the `safety valve` follow the same logic of the ones defined for the `lamination valve`. Being a safety valve a "directional" unit, the nodes have a simple logic: one `inlet` and one `outlet`. These two are indicated in the drawing above as `i` and `o` respectively. The `outlet` node may be used to represent the collection of the discharged stream into a particular treatment circuit.

Though a safety valve should never be included into the main process path, possibly even not into the secondary paths, three kinds of safety valves are defined: `main`, `secondary` and `utility`, which are drawn respectively with `semithick` lines, `thin` lines and `very thin` lines. These keys should be used as arguments of the `safety valve` pic.

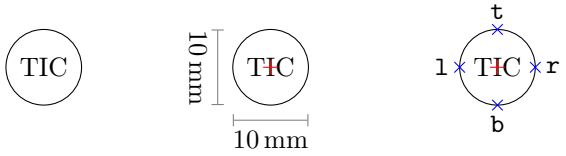
5.2 Control Instruments

Instrument

Often not present in a PFD, schematics of control instrumentation is a valuable integration to the process scheme. It can be shown in order to better describe a (simple) plant. A generic instrument symbol is defined as a pic with arguments called `instrument`:

```
\pic at (0,0) {instrument=TIC};
```

and yields a circle anchored in its centre, with `thin` line thickness and containing the designation of the instrument:



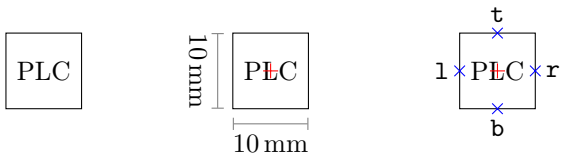
The behaviour of the `instrument` pic differs with respect to other pics with arguments and it is more similar to the third argument of the `\measure` command: the argument can be anything accepted by a Ti*k*Z node and, in the case of chemical plants control instrumentation, it should be a text specifying the type of instrument, as shown above (TIC stands for temperature indicator and controller).

Controller

Sometimes it is necessary to represent a different kind of instrumentation. A common case is a computer connected to a measurement and control system, which acts as a simple controller. For instance, it may be a programmable logic controller (PLC) or a distribute control system (DCS). In such cases, a special pic exists. It is defined as a pic with arguments called `controller`:

```
\pic at (0,0) {controller=PLC};
```

and yields a square anchored in its centre, with `thin` line thickness and containing the specification of the controller:



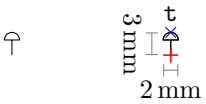
Arguments to be used with a `controller` pic follow exactly the same logic of the ones used within an `instrument`: simple text specifying the controller functions.

Actuator

Signal paths have already been discussed and can be drawn with the `signal` style, but there is another utility for control instrumentation. Connections between signals and units can be made through a sensor, to collect informations, or through an actuator, to act on the units. Usually sensors are indicated simply connecting a signal to the point of measurement, a unit or a stream, but for actuators an additional symbol is used. It is defined as a simple pic called `actuator`:

```
\pic at (0,0) {actuator};
```

and yields a half circle supported by a vertical stem, at the end of which there is the anchor point:

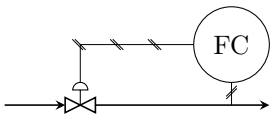


The `actuator` is a strange thing to represent in a scheme and obtaining a good graphical result is not that easy. In order to get a better drawing, its line thickness is the same of instruments, but its dimensions are defined to be relative to the units. This aspect will be better clarified in the future.

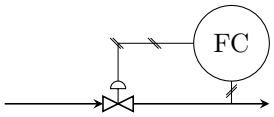
This is a good time to give a practical example of how coordinate nodes can be used. Introducing the `valve`, it was said that its `anchor` node can come in hand when representing actuators. This simple code:

```
\pic (V) at (1,0) {valve=main};
\pic (A) at (V-anchor) {actuator};
\pic (FC) at (3,0.8) {instrument=FC};
\draw[main stream] (0,0) -- (V-left);
\draw[main stream] (V-right) -- (3.5,0);
\draw[short signal] (3,0) -- (FC-bottom);
\draw[signal] (FC-left) -| (A-top);
```

produces the representation of a flow-rate control loop:



It should be noticed that coordinate nodes are really useful, but appealing to them implies that the diagram my need to be planned in a non-linear way. In particular, it is convenient to fix positions of all of the units first and then connecting their nodes using streams and signals. This procedure requires a little of practice, but, doubtlessly, its advantages worth the effort. A simple example: if one decides to move the valve from the coordinates `(1,0)` to the coordinates `(1.5,0)`, the only thing to do is to change the coordinates of the point where the `valve` is placed in the first line of the example code and all of the other points will automatically move themselves accordingly:

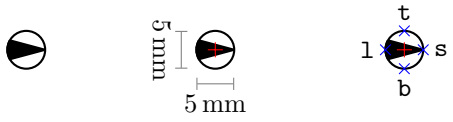


5.3 Process Inlets and Outlets

When representing the scheme of a process, it is always necessary to indicate where prime matters enter and where products leave. One can simply use a stream with a free end, but UNICHIM defines two particular symbols to these special purposes. An inlet is defined as a simple pic called `inlet`:

```
\pic at (0,0) {inlet};
```

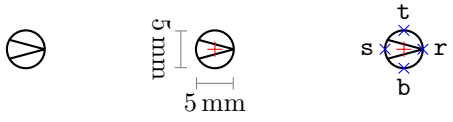
and yields a circle, in which centre there is the anchor, with a filled arrow sketch:



while an outlet is defined as a simple pic called `outlet`:

```
\pic at (0,0) {output};
```

and yields a circle, in which centre there is the anchor, with an empty arrow sketch:



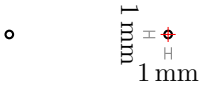
For both `inlet` and `outlet` pics, streams should be connected to specific points. More precisely, inlet streams enter the scheme coming from the tip of the arrow sketch, while outlet streams leave the scheme going into the base of the arrow sketch. These special anchor points are marked as `stream` nodes, which positions are indicated in the above drawings by means of the abbreviated name `s`. Other nodes are defined to make the labelling of inlets and outlets easier.

5.4 Nozzles

It is sometimes useful to highlight nozzles on units to indicate where a fluid can go in and where it will come out, if there is a one way path to be followed. Two simple pics are defined to accomplish this kind of need. An input nozzle is defined as a simple pic called `input`:

```
\pic at (0,0) {input};
```

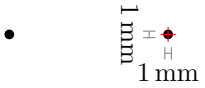
and yields an empty circle anchored in its centre:



while an output nozzle is defined as a simple pic called `output`:

```
\pic at (0,0) {output};
```

and yields a filled circle anchored in its centtr:



Both `input` and `output` pics are drawn with a `thick` line, the same thickness used for units.

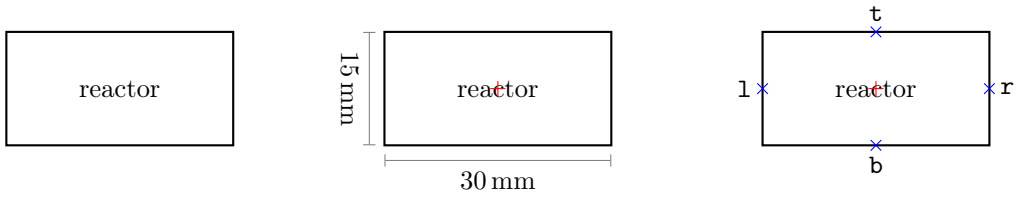
Except for the `anchor` node, placed on the anchor points of the units, both `input` and `output` have no extra coordinate nodes.

5.5 Blocks

Units introduced so far are useful to represent the PFD of a chemical process, but sometimes it is enough (or required) to represent a process using a much more simpler BFD, a diagram in which entire sections of the process are gathered into a self-explicative block. In order to represent BFDs, only the `main stream` style should be used for lines. Blocks can be obtained by means of a special pic with arguments called `block`:

```
\pic at (0,0) {block=reactor};
```

which yields a rectangle anchored in its centre, with `thick` line thickness and containing the specification of the block:

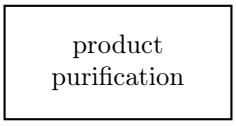


The argument passed to the `block` pic is its specification, so it should be a text string containing the name of the block. Text is, by default, written in `\footnotesize` and placed in the centre of the block.

It should be noticed that the definition of the `block` pic permits to split the text passed as argument over multiple lines; use the `\\` command to brake lines. For example, the code:

```
\pic at (0,0) {block=product\\purification};
```

yields:



and the text is always aligned to the centre of the block.

Nodes defined for the block deserve an explanation. A BDF is a very free and general representation of a process, thus there are no specific positions defined to connect streams. The nodes defined by chemplants are the ones on the "remarkable boundaries" of the block, but a stream may be connected to any point of the block. If one wants to connect a stream in a different place with respect to the given nodes, coordinates have to be calculated by hand (and it not so difficult with a rectangle).

6 Transforming Units

As told before, all of the units described so far are shown in their default orientation. Anyway, the method of defining them as pics permits to apply all of the pic actions available in Ti*k*Z. Among them, the most useful ones are for sure tranformations.

Useful Transformations

A transformation refers to the manipulation of the coordinates that describe the drawing to obtain some specific effects, of example a rotation or a scaling. These two are the main transformations which can be applied to units, they will be briefly summarised in the following using as a model a `heat exchanger biphase`.

- To make the confrontation easier, the standard `heat exchanger biphase` is:

- A pic can be rotated using the `rotate` transformation:

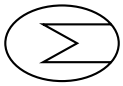
```
\pic[rotate=90] at (0,0) {heat exchanger biphase};
```

  which yields:

- A pic can be horizontally scaled using the `xscale` transformation:

```
\pic[xscale=1.5] at (0,0) {heat exchanger biphase};
```

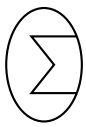
  which yields:



- A pic can be vertically scaled using the `yscale` transformation:

```
\pic[yscale=1.5] at (0,0) {heat exchanger biphase};
```

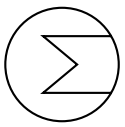
  which yields:



- A pic can be scaled using the `scale` transformation:

  ```
  \pic[scale=1.5] at (0,0) {heat exchanger biphase};
  ```

  which yields:

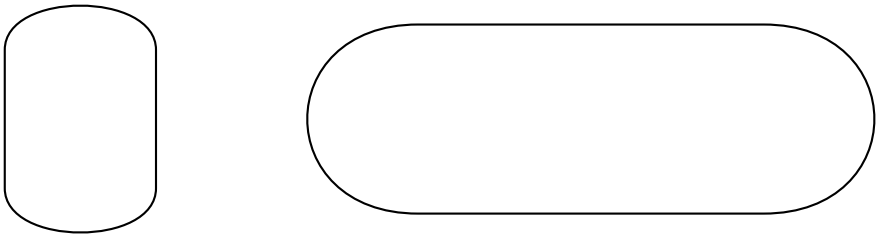


Some Tricks

Now some tricks based on transformations. Even though scaling a unit in a single direction can appear useless, this transformation can be used in a clever way. One example is "transforming" a tank in a wide horizontal basin with a code like:

```
\pic at (0,0) {tank};
\pic[xscale=2.5, yscale=1.25, rotate=90] at (6.75,0) {tank};
```

which yields:



(Note that, for some obscure reasons, the pic is firstly rotated and then scaled).

Another useful trick based on the single direction scaling can be used to flip a unit. If a condenser with the utility stream going from left to right is needed, the `condenser` can be flipped horizontally scaling its coordinates by a -1 factor, thus inverting them. The code:

```
\pic at (0,0) {condenser};
\pic[xscale=-1] at (3.4,0) {condenser};
```

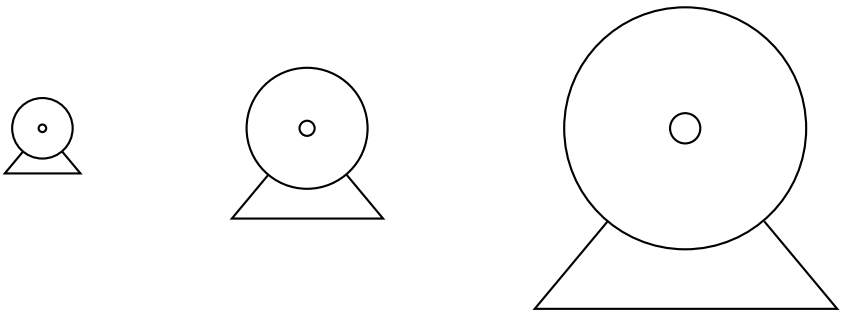
yields:

This trick works with `yscale` and `scale` as well

Last, but not least, it should be noticed that sequential scaling factors are cumulative. The code:

```
\pic at (0,0) {centrifugal pump};
\pic[scale=2] at (3.5,0) {centrifugal pump};
\pic[scale=2, scale=2] at (8.5,0) {centrifugal pump};
```

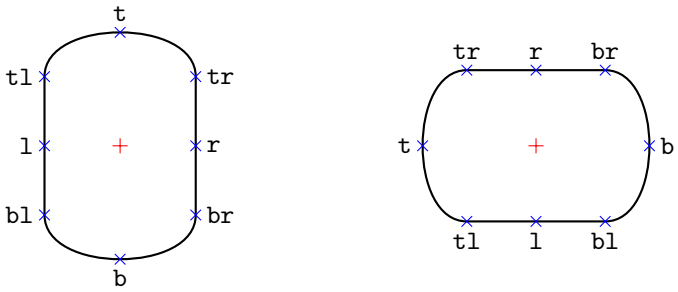
yields:



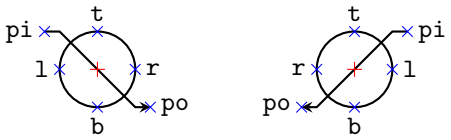
in fact the pump in the middle has twice the dimensions of the one on the left, while the pump on the right has twice the dimensions of the one in the middle, which means four times the dimensions of the one on the left. This characteristic of the scaling factors is useful for a feature of chemplants which will be introduced soon. Note also that the thickness of the lines is not scaled, which is the main advantage to scale the geometric description of a figure rather than its vectorial representation.

It is important to remark that all of the above mentioned transformations affect not just the pics drawings, but also the positions of the nodes defined into them. Anyway, names never change, so careful evaluations have to be done when it is required to transform a unit and to use its nodes at the same time.

Here are some examples of the most "invasive" transformations. Rotating a unit rotates also its nodes:



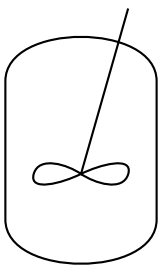
while flipping a unit, for example horizontally with `xscale=-1`, flips also its nodes:



Finally, some special considerations concerning associative pics used to give flexible representation of reactors. It has been specified that the `tank reactor` is analogue to the `tank` and that the scale factor to "convert" the first into the second is 1.25. This is useful, for example, if one has to represent a stirrer within a tank. The code:

```
\pic (T) at (0,0) {tank};
\pic[scale=1.25] at (T-anchor) {stirrer};
```

yields:

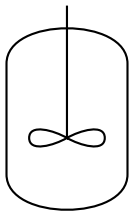


This is just an example, but this trick can be used to associate any two pics one wants to bond.

The `stirrer` can be subject to another transformation not mentioned yet. By default, the pic is slanted to avoid that the interception point between the shaft of the stirrer and the dome of the tank reactor falls on the `top` node of the tank. If one does not like the slanted shape of the `stirrer` and prefers to have a straight stirrer, the `xslant` transformation can be used. The code:

```
\pic (R) at (0,0) {tank reactor};
\pic[xslant=-0.285, xshift=-2.04] at (R-anchor) {stirrer};
```

yields:

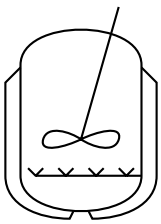


As it can be seen, the transformation is not that straightforward, in fact also a little of shift in the x direction is needed.

Finally, the mirroring trick based on the `xscale=-1` transformation can be used to mirror both the `sprayer` and the `bubbler` to connect them to the right side of the `tank reactor` rather than to its left. Taking as an example a jacketed sparger reactor, if one wants the gas to enter from the right side, the code:

```
\pic (R) at (0,0) {tank reactor};
\pic at (R-anchor) {jacket};
\pic at (R-anchor) {stirrer};
\pic[xscale=-1] at (R-bottom right) {bubbler};
```

will do the job:



7 Customisation

Some of the graphical aspects used by the chemplants package can be customised at will of the user. Of course, symbols of the units should not be changed, but there could be some precise necessities that makes the standard chemplants parameters unsuitable to accomplish requirements. A case could be the need to draw the scheme of a big process: standard dimensione of units may make the whole diagram too large to fit it into a single sheet. Another case is when one is not satisfied by the default unit or streams line thicknesses and would like to change them.

To solve issues like these, chemplants provides a rudimental mechanism to set a number of graphical features of streams and units. In the following, the customisable parameters are described.

Styles for streams can be tuned setting arrow tips and line thickness; the former is common to all of the streams and the latter is specific to each stream style.

- Arrow tip used by `main stream`, `secondary stream` and `utility stream` styles can be set using the command:

  `\setchpstreamtip{`⟨*arrow tip*⟩`}`

  which requires as argument the name of a Ti*k*Z arrow tip; default value is `stealth`.

- Line thickness used by the `main stream` style can be set using the command:

  `\setchpmainstreamthickness{`⟨*line thickness*⟩`}`

  which requires as argument the name of a Ti*k*Z line thickness; default value is `semithick`.

- Line thickness used by the `secondary stream` style can be set using the command:

  `\setchpsecondarystreamthickness{`⟨*line thickness*⟩`}`

  which requires as argument the name of a Ti*k*Z line thickness; default value is `thin`.

- Line thickness used by the `utility stream` style can be set using the command:

  `\setchputilitystreamthickness{`⟨*line thickness*⟩`}`

  which requires as argument the name of a Ti*k*Z line thickness; default value is `very thin`.

Styles for units can be tuned setting line thickness and unit base dimension. The latter is extremely important because permits to scale all of the units by a given scale factor, thus enabling the overall scaling by means of a single command. Units can furthermore be scaled individually witusing the `scale` transformation thanks to the cumulative behaviour of scaling factors.

- Line thickness used by all of the units defined as pics can be set using the command:

  `\setchpunitthickness{`⟨*line thickness*⟩`}`

  which requires as argument the name of a Ti*k*Z line thickness; default value is `thick`.

- Scale factor to be applied to all of the units defined as pics can be set using the command:

  `\setchpunitscale{`⟨*scale factor*⟩`}`

  which requires as argument a numerical value to be used as scale factor and that will multiply every dimension of all of the units; default value is `1`.

Styles for control instrumentation can be tuned setting signal style line thickness, instruments pics line thickness, font dimension and base dimension. This last feature works exactly like the one of units, but it is a different parameter in order to allow independent scaling of units and instruments.

- Line thickness used by the `signal` style can be set using the command:

  `\setchpsignalthickness{`⟨*line thickness*⟩`}`

  which requires as argument the name of a Ti*k*Z line thickness; default value is `very thin`.

- Line thickness used by all of the instrumentation defined as pics can be set using the command:

  `\setchpinstrumentthickness{`⟨*line thickness*⟩`}`

  which requires as argument the name of a Ti*k*Z line thickness; default value is `thin`.

- Scale factor to be applied to all of the instrumentation defined as pics can be set using the command:

  `\setchpinstrumentscale{`⟨*scale factor*⟩`}`

  which requires as argument a numerical value to be used as scale factor and that will multiply every dimension of all of the instrumentation; default value is `1`.

- Font size to be used within all of the instrumentation defined as pics can be set using the command:

  `\setchpinstrumentfontsize{`⟨*font size*⟩`}`

  which requires as argument a font size attribute understood by LaTeX; default value is `\footnotesize`.

Since parameters of both units and instrumentation have been introduced, it is possible to better specify how these features influence the `actuator` pic, a sort of hybrid between a unit and an instrument. As told before, `actuator` is sensible to instruments line thickness, but not to their dimensions, in fact it is defined in relative terms to the unit dimensions. This means that `\setchpunitscale` will change the scale of the `actuator`, while `\setchpinstrumentthickness` will change its line thickness.

Styles for hidden streams and components can be tuned setting line patterns for both styles.

- Line pattern used by the `hidden stream` style can be set using the command:

  `\setchphiddenstreamstyle{`⟨*line pattern*⟩`}`

  which requires as argument the name of a Ti*k*Z line pattern; default value is `dashed`.

- Line pattern used by the `hidden component` style can be set using the command:

  `\setchphiddencomponentstyle{`⟨*line pattern*⟩`}`

  which requires as argument the name of a Ti*k*Z line pattern; default value is `densely dotted`.

Styles for the `\measure` command can be tuned setting line color, tips and thickness, plus font size of the measure value.

- Line color used by the `\measure` command can be set using the command:

`\setchpmeasurecolor{`⟨*line color*⟩`}`

which requires as argument whichever expression can be used to indicate a color in LaTeX; default value is `gray`.

- Line thickness used by the `\measure` command can be set using the command:

  `\setchpmeasurethickness{`⟨*line thickness*⟩`}`

  which requires as argument the name of a Ti*k*Z line thickness; default value is `thin`.

- Line tip used by the `\measure` command can be set using the command:

  `\setchpmeasuretip{`⟨*line tip*⟩`}`

  which requires as argument the name of a Ti*k*Z arrow tip; default value is `|`.

- Font size to be used within the `\measure` command can be set using the command:

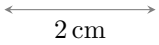
  `\setchpmeasurefontsize{`⟨*font size*⟩`}`

  which requires as argument a font size attribute understood by LaTeX; default value is `\footnotesize`.

For what concerns the `\measure` line tip, it should be noticed that there is no possibility to set different tips for the two extremes of the line. Using the name of an arrow tip will yield the same tip pointing out on both sides. For example, after:

```
\setchpmeasuretip{stealth}
```

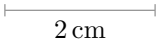
the `\measure` will appear again as:



Although rudimental, the customisation mechanism of chemplants offers a very useful opportunity. After giving a new:

```
\setchpmeasuretip{|}
```

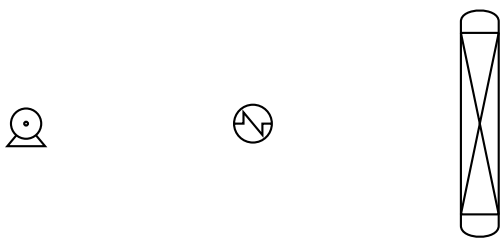
the `\measure` will appear again as:



and this means that all of the setting commands listed above act like declarations and have local validity, so they can be changed everywhere into the document (even multiple times inside the same `tikzpicture` environment). This is particularly useful to solve the problem introduced above as example, the possibly too large base dimension of units in the case of large diagrams. One can simply scale down all of the units before the scheme and restore the default unit scale after that. An example. After giving:

```
\setchpunitscale{0.5}
```

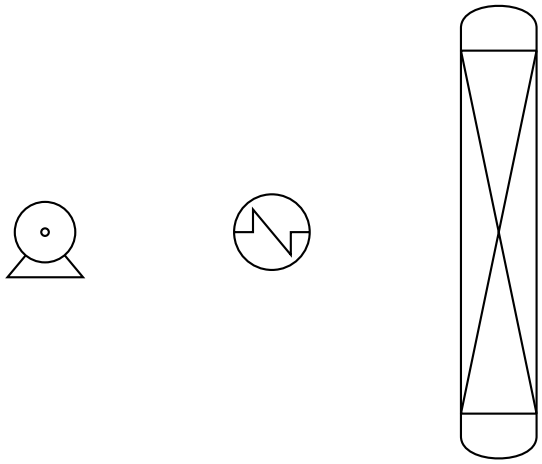
all of the units will be half their default sizes:



and after a new:

`\setchpunitscale{1}`

all of the units will be restored to their default dimensions:



Styles for blocks can be tuned setting line thickness, font dimension and base dimension. Also this last feature, like the scale factor for instruments, regards blocks only and it is independent of the units scale factor.

- Line thickness used by the `block` pic can be set using the command:

  `\setchpblockthickness{`⟨*line thickness*⟩`}`

  which requires as argument the name of a Ti*k*Z line thickness; default value is `thick`.

- Scale factor to be applied to the `block` pic can be set using the command:

  `\setchpblockscale{`⟨*scale factor*⟩`}`

  which requires as argument a numerical value to be used as scale factor and that will multiply every dimension of the block; default value is `1`.

- Font size to be used within the `block` pic can be set using the command:

  `\setchpinstrumentfontsize{`⟨*font size*⟩`}`

  which requires as argument a font size attribute understood by LaTeX; default value is `\footnotesize`.

A last remark: the listed commands permit to easily tune a number of graphical parameters in order to satisfy different necessities. Demanding users who have a good knowledge of the Ti*k*Z package can anyway customise even more the styles used by chemplants looking for the code of the package and modifying the "internal styles" (the ones not meant to be used directly by the "common" user and prefixed by `chp`) as they like, but at their own risk.

8 Examples

Some "complete" examples will be presented in order to give a better view of how the chemplants package works. Both codes and resulting diagrams will be shown.

Some Remarks on the Drawing Procedure

If some users want to try and replicate (copy and paste) the examples, they are free to do that. In this case they will find useful to add some additional macros to their preambles, some commands I defined during time and that I use extensively:

```
\renewcommand{\vec}[1]{\boldsymbol{#1}} % vector notation
\newcommand{\flow}[1]{\dot{#1}}         % flow-rate notation
```

and which I defined only for easiness of change. Moreover, one of the examples will use the basic system to parse chemistry notation of the chemformula package (that, by the way, is an excellent suite for documents dealing with chemistry). Finally, two macros defined by the Italian option passed to the babel package:

```
\providecommand*{\ap}[1]{%              % upright superscripts
    \textormath{%
        \textsuperscript{#1}%
    }%
    {%
        ^{\mathrm{#1}}%
    }%
}
\providecommand*{\ped}[1]{%             % upright subscripts
    \textormath{%
        $_{\mbox{\fontsize\sf@size\z@\selectfont#1}}$%
    }%
    {%
        _\mathrm{#1}%
    }%
}
```

Before moving on to the real examples it is useful to recall the two main approaches one can follow to draw the scheme of a chemical process. As a general rule, it is almost always better to plan the scheme before starting its real drawing, where "to plan" means to decide where to put units, how to connect them through streams, where signal lines should pass on the scheme, how much space is required between units and so on. This permits to avoid to reach the end of the scheme and to realize only too late that it is too large to fit on the page, when spacing units a little less would have solved the problem.

Once the plan has been done, it is possible to move to the real construction of the drawing, but here it is necessary to choose the approach:

- building the scheme thinking linearly (following the streams) and calculating by hand the coordinates of the points in which units will be placed and in which streams will be connected;
- building the scheme thinking in a more global way, placing units first and then using their coordinates nodes to connect them through streams and signals.

As told before, the second way requires a little more practice, but the resulting scheme is much more easy to obtain and flexible to modify (even though the code can be slightly more complex to read). Anyway, users are free to choose the approach they like the most, being ready to accept all of its implications.

In the following examples, both approaches will be shown, but not for the same code: some of the examples will be structured calculating coordinates by hand, while

the remaining will use nodes. (Readers should use care in deciding to follow strictly the drawing methods which will be shown in the following codes, in fact examples are mainly the schemes I had to represent. It is likely that better ways to draw a scheme can be found.)

Hand Calculation of Coordinates

Listing 1: Scheme of a flash process.

```
\begin{tikzpicture}
    \draw[main stream] (-0.5,0) -- (1.5,0);
    \node[above right] at (-0.5,0) {$\flow{n}\ap{F}, \vec{z}$};
    \pic at (1.5,0) {centrifugal pump};
    \draw[main stream] (1.5,0.4) -- (3,0.4);
    \pic at (3.5,0.4) {heat exchanger};
    \draw[utility stream] (3.5,1.4) -- (3.5,0.9);
    \node[above left] at (3.5,0.9) {$\flow{Q}$};
    \draw[utility stream] (3.5,-0.1) -- (3.5,-0.6);
    \draw[main stream] (4,0.4) -- (4.8,0.4);
    \pic at (5,0.4) {lamination valve};
    \draw[main stream] (5.2,0.4) -- (6,0.4);
    \node[above left] at (6,0.4) {$P$};
    \pic at (6.8,0.4) {gas-liquid separator};
    \draw[main stream] (6.8,1.9) |- (9,2.5);
    \node[above left] at (9,2.5) {$\flow{n}\ap{V}, \vec{y}$};
    \draw[main stream] (6.8,-1.1) |- (9,-1.7);
    \node[below left] at (9,-1.7) {$\flow{n}\ap{L}, \vec{x}$};
\end{tikzpicture}
```

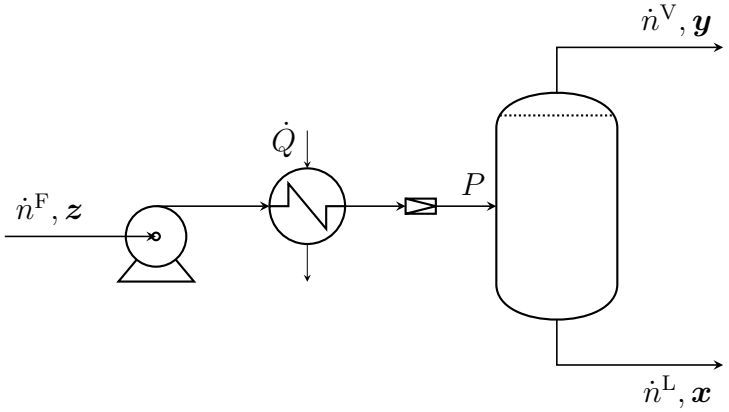


Figure 1: Scheme of a flash process produced by the listing 1 code.

Listing 2: Scheme of a countercurrent multiple flash process.

```
\begin{tikzpicture}
    \draw[main stream] (0.5,0) -- (2,0);
    \node[above right] at (0.5,0)
        {$\flow{n}\ap{F}, \vec{z}\ap{F}$};
    \pic at (3,0) {tank};
    \draw[main stream] (3,1.5) |- (5,2);
    \node[above] at (3,2)
```

```
    {$\flow{n}\ap{V_\mathit{j + 1}},
    \vec{y}\ap{V_\mathit{j + 1}}$};
\pic at (6,3) {tank};
\draw[main stream] (6,4.5) |- (8.333,5) |- (9,2);
\node[above] at (6,5)
    {$\flow{n}\ap{V_\mathit{j}},
    \vec{y}\ap{V_\mathit{j}}$};
\pic at (10,3) {tank};
\draw[main stream] (10,4.5) |- (12.3,5);
\pic[rotate=180] at (12.5,5) {valve triple=main};
\draw[main stream] (12.7,5) -- (14,5);
\node[above left] at (14,5)
{$\flow{n}\ap{D}, \vec{y}\ap{D}$};
\draw[main stream] (12.5,4.8) -- (12.5,4);
\pic[rotate=-90] at (12.5,3.5) {heat exchanger};
\draw[utility stream] (11.5,3.5) -- (12,3.5);
\draw[utility stream] (13,3.5) -- (13.5,3.5);
\node[above] at (13.5,3.5) {$\flow{Q}\ped{C}$};
\draw[main stream] (12.5,3) |- (11,2);
\draw[main stream] (10,1.5) |- (7.688,1) |- (7,4);
\node[below] at (10,1)
{$\flow{n}\ap{L_1}, \vec{x}\ap{L_1}$};
\draw[main stream] (6,1.5) |- (4,1);
\node[below] at (6,1)
    {$\flow{n}\ap{L_\mathit{j}},
    \vec{x}\ap{L_\mathit{j}}$};
\draw[main stream] (3,-1.5) |- (5,-2);
\node[below] at (3,-2)
    {$\flow{n}\ap{L_\mathit{k - 1}},
    \vec{x}\ap{L_\mathit{k - 1}}$};
\pic at (6,-3) {tank};
\draw[main stream] (6,-4.5) |- (8.333,-5) |- (9,-2);
\node[below] at (6,-5)
    {$\flow{n}\ap{L_\mathit{k}},
    \vec{x}\ap{L_\mathit{k}}$};
\pic at (10,-3) {tank};
\draw[main stream] (10,-4.5) |- (12.3,-5);
\pic at (12.5,-5) {valve triple=main};
\draw[main stream] (12.7,-5) -- (14,-5);
\node[below left] at (14,-5)
    {$\flow{n}\ap{R}, \vec{x}\ap{R}$};
\draw[main stream] (12.5,-4.8) -- (12.5,-4);
\pic[rotate=-90] at (12.5,-3.5) {heat exchanger};
\draw[utility stream] (13.5,-3.5) -- (13,-3.5);
\draw[utility stream] (12,-3.5) -- (11.5,-3.5);
\node[below] at (13.5,-3.5) {$\flow{Q}\ped{B}$};
\draw[main stream] (12.5,-3) |- (11,-2);
\node[above] at (6,-1)
    {$\flow{n}\ap{V_\mathit{k}},
    \vec{y}\ap{V_\mathit{k}}$};
\draw[main stream] (10,-1.5) |- (7.688,-1) |- (7,-4);
\node[above] at (10,-1)
    {$\flow{n}\ap{V_{\mathit{N\ped{T}}}}},
```

```
        \vec{y}\ap{V_{\mathit{N\ped{T}}}}$};
    \draw[main stream] (6,-1.5) |- (4,-1);
\end{tikzpicture}
```

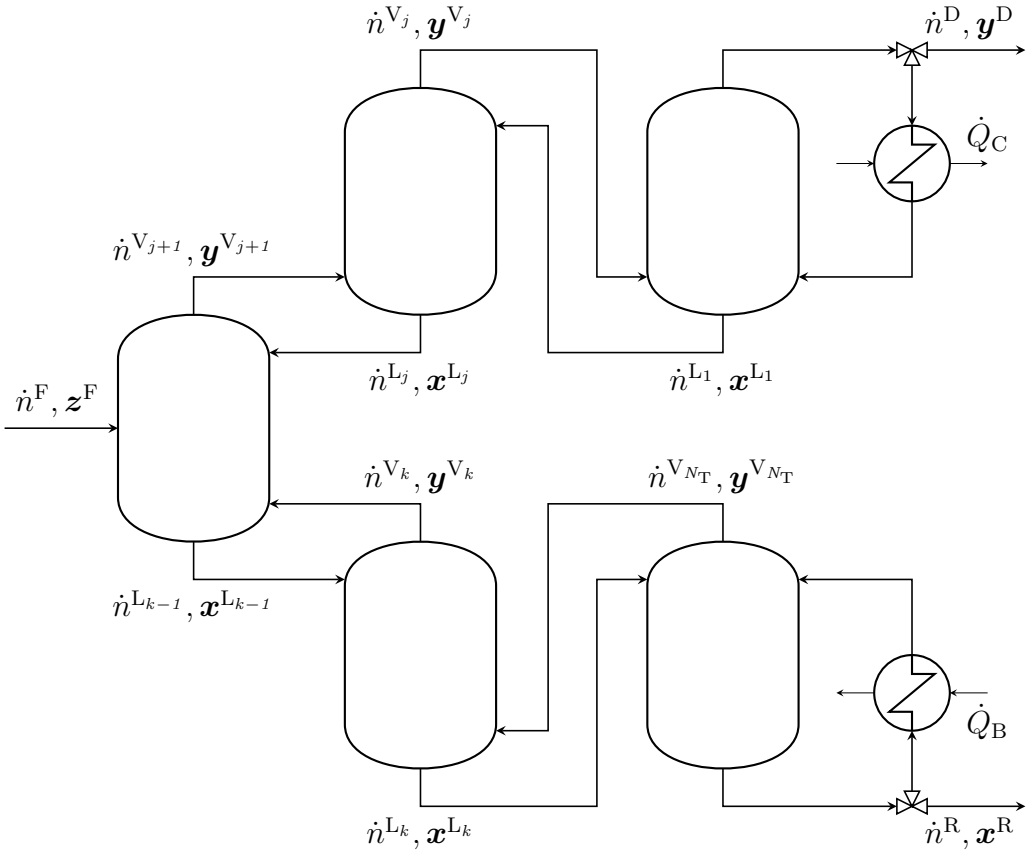


Figure 2: Scheme of a countercurrent multiple flash process produced by the listing 2 code.

Listing 3: Scheme of a continuous distillation process.

```
\begin{tikzpicture}
    \draw [main stream] (-2,0) -- (-0.5,0);
    \node[above right] at (-2,0)
        {$\flow{n}\ap{F}, \vec{z}\ap{F}$};
    \pic at (0,0) {column=trayed};
    \pic at (1.5,3.5) {condenser};
    \draw [main stream] (0,3) |- (1,3.5);
    \node [above right] at (0,3.5) {$\flow{n}\ap{V}$};
    \draw [main stream] (1.5,4) |- (3,4.5);
    \node[above left] at (3,4.5)
        {$\flow{n}\ap{D}, \vec{y}\ap{D}$};
    \draw [secondary stream] (1.5,3) |- (0.5,2.6);
    \node [below left] at (1.5,2.6) {$\flow{n}\ap{L}$};
    \pic at (1.5,-3.5) {boiler};
    \draw [main stream] (0,-3) |- (1,-3.5);
    \node [below right] at (0,-3.5) {$\flow{n}\ap{L'}$};
    \draw [main stream] (1.5,-4) |- (3,-4.5);
    \node[below left] at (3,-4.5)
        {$\flow{n}\ap{R}, \vec{x}\ap{R}$};
    \draw [secondary stream] (1.5,-3) |- (0.5,-2.6);
```

```
    \node [above left] at (1.5,-2.6) {$\flow{n}\ap{V'}$};
\end{tikzpicture}
```

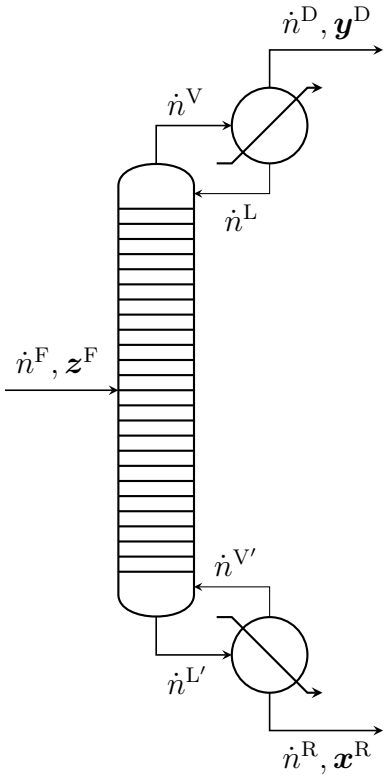


Figure 3: Scheme of a continuous distillation process produced by the listing 3 code.

The following is the very first usage I did of chemplants (when it was just a foggy idea), and, probably, still the most complex one. In this scheme, I had the need to represent sensors on valves, so I used the `actuator` pic to this aim.

Listing 4: Complete PFD of a vanadium redox flow battery pilot plant.

```
\begin{tikzpicture}[font=\footnotesize]
% Positive Electrolyte
    % tank outlet
    \pic at (3,3.5) {tank};
    \node[align=center] at (3,4) {Positive\\ Electrolyte};
    \node at (3,3) {\ch{VO2+}/\ch{VO^2+}};
    \draw[main stream] (3,2) -- (3,1.7);
    \pic[rotate=90] at (3,1.5) {valve triple=main};
    \draw[main stream] (3,1.3) |- (2,0.6);
    \pic at (2,0.6) {centrifugal pump};
    \draw[main stream] (2,1) -| (1,1.3);
    \pic[rotate=270] at (1,1.5) {valve triple=main};
    % pump recycle
    \draw[secondary stream] (1.2,1.5) -- (1.8,1.5);
    \pic at (2,1.5) {valve=secondary};
    \node[above] at (2,1.5) {$V\ped{P}^+$};
    \draw[secondary stream] (2.2,1.5) -- (2.8,1.5);
    % battery stream
    \draw[main stream] (1,1.7) -- (1,3);
    \pic at (1,3.5) {instrument=FM};
```

```
    \draw[main stream] (1,4) -- (1,5.3);
    \draw[signal] (0.5,3.5) -| (0.25,2.5);
    \pic at (0.25,2) {controller=PID};
    \draw[signal] (0.25,1.5) |- (1.6,0.6);
    \pic[rotate=90] at (1,5.5) {valve=main};
    \node[right] at (1,5.5) {$V_1^+$};
    \draw[main stream] (1, 5.7) |- (2.8,6.9);
    \pic[rotate=90] at (3,6.9) {valve triple=main};
    \draw[main stream] (3,7.1) |- (6.2,7.5);
    \pic at (4,8.5) {instrument=TM};
    \draw[short signal] (4,8) -- (4,7.5);
    % battery bypass
    \draw[secondary stream] (3,6.7) -- (3,6.4);
    \pic[rotate=270] at  (3,6.2) {valve=secondary};
    \pic[rotate=270] at  (3,6.2) {actuator};
    \node[left] at (3,6.2) {$V\ped{B}^+$};
    \draw[secondary stream] (3,6) -- (3,5.7);
    % battery outlet
    \draw[main stream] (8,7) |- (5.2,5.5);
    \pic at (6,4.5) {instrument=TM};
    \draw [short signal] (6,5) -- (6,5.5);
    \pic at (5,5.5) {valve=main};
    \pic at (5,5.5) {actuator};
    \node[below] at (5,5.5) {$V_2^+$};
    \draw[main stream] (4.8,5.5) -- (3.2,5.5);
    \pic at (5,6.5) {instrument=PM};
    \draw[short signal] (5,7) -- (5,7.5);
    \draw[signal] (5.5,6.5) -- (7,6.5);
    % tank inlet
    \pic[rotate=270] at (3,5.5) {valve triple=main};
    \draw[main stream] (3,5.3) -- (3,5);
% Negative Electrolyte
    % tank outlet
    \pic at (12,3.5) {tank};
    \node[align=center] at (12,4) {Negative\\ Electrolyte};
    \node at (12,3) {\ch{V^3+}/\ch{V^2+}};
    \draw[main stream] (12,2) -- (12,1.7);
    \pic[rotate=270] at (12,1.5) {valve triple=main};
    \draw[main stream] (12,1.3) |- (13,0.6);
    \pic at (13,0.6) {centrifugal pump};
    \draw[main stream] (13,1) -| (14,1.3);
    \pic[rotate=90] at (14,1.5) {valve triple=main};
    % pump recycle
    \draw[secondary stream] (13.8,1.5) -- (13.2,1.5);
    \pic at (13,1.5) {valve=secondary};
    \node[above] at (13,1.5) {$V\ped{P}^-$};
    \draw[secondary stream] (12.8,1.5) -- (12.2,1.5);
    % battery stream
    \draw[main stream] (14,1.7) -- (14,3);
    \pic at (14,3.5) {instrument=FM};
    \draw[main stream] (14,4) -- (14,5.3);
    \draw[signal] (14.5,3.5) -| (14.75,2.5);
    \pic at (14.75,2) {controller=PID};
```

```
    \draw[signal] (14.75,1.5) |- (13.4,0.6);
    \pic[rotate=90] at (14,5.5) {valve=main};
    \node[left] at (14,5.5) {$V_1^-$};
    \draw[main stream] (14, 5.7) |- (12.2,6.9);
    \pic[rotate=270] at (12,6.9) {valve triple=main};
    \draw[main stream] (12,7.1) |- (8.8,7.5);
    \pic at (11,8.5) {instrument=TM};
    \draw[short signal] (11,8) -- (11,7.5);
    % battery bypass
    \draw[secondary stream] (12,6.7) -- (12,6.4);
    \pic[rotate=90] at  (12,6.2) {valve=secondary};
    \pic[rotate=90] at  (12,6.2) {actuator};
    \node[right] at (12,6.2) {$V\ped{B}^-$};
    \draw[secondary stream] (12,6) -- (12,5.7);
    % battery outlet
    \draw[main stream] (7,7) |- (8.5,5) |- (9.8,5.5);
    \pic at (9,4.5) {instrument=TM};
    \draw[short signal] (9,5) -- (9,5.5);
    \pic at (10,5.5) {valve=main};
    \pic at (10,5.5) {actuator};
    \node[below] at (10,5.5) {$V_2^-$};
    \draw[main stream] (10.2,5.5) -- (11.8,5.5);
    \pic at (10,6.5) {instrument=PM};
    \draw[short signal] (10,7) -- (10,7.5);
    \draw[signal] (9.5,6.5) -- (8,6.5);
    % tank inlet
    \pic[rotate=90] at (12,5.5) {valve triple=main};
    \draw[main stream] (12,5.3) -- (12,5);
% Battery
    % block structure
    \draw[thick] (6,7) rectangle (9, 9.6);
    \node[above] at (7.5,9.6) {Battery};
    % positive main stream
    \pic at (6.2,7.5) {input};
    \draw[main stream, hidden stream]
        (6.2,7.55) |- (7.8,9.2) |- (7.95,8.5);
    \pic at (8,8.5) {output};
    \draw[main stream] (8,8.5) -- (8.3,8.5);
    \pic[rotate=90] at (8.5,8.5) {valve triple=main};
    \pic[rotate=270] at (8.5,8.5) {actuator};
    \draw[main stream] (8.5,8.3) |- (8,8);
    \pic at (8,8) {input};
    \draw[main stream, hidden stream] (8,7.95) -- (8,7.7);
    \pic[rotate=90, hidden component] at (8,7.5)
        {valve triple=main};
    \draw[main stream, hidden stream] (8,7.3)  -- (8,7);
    % negative main stream
    \pic at (8.8,7.5) {input};
    \draw[main stream, hidden stream]
        (8.8,7.55) |- (7.2,9.4) |- (7.05,8.5);
    \pic at (7,8.5) {output};
    \draw[main stream] (7,8.5) -- (6.7,8.5);
    \pic[rotate=270] at (6.5,8.5) {valve triple=main};
```

```
    \pic[rotate=90] at (6.5,8.5) {actuator};
    \draw[main stream] (6.5,8.3) |- (7,8);
    \pic at (7,8) {input};
    \draw[main stream, hidden stream] (7,7.95) -- (7,7.7);
    \pic[rotate=270, hidden component] at (7,7.5)
        {valve triple=main};
    \draw[main stream, hidden stream] (7,7.3)  -- (7,7);
    % positive to negative remixing
    \draw[secondary stream, color=red] (8.5,8.7) |- (8,9);
    \pic at (8,9) {input};
    \draw[secondary stream, hidden stream, color=red]
        (7.95,9) -| (7.4,7.5) -- (7.2,7.5);
    % negative to positive remixing
    \draw[secondary stream, color=red] (6.5,8.7) |- (7,9);
    \pic at (7,9) {input};
    \draw[secondary stream, hidden stream, color=red]
        (7,8.95) |- (7.6,8.75) |- (7.8,7.5);
% Tanks Connection
    \draw (4,2.585) -- (7.3,2.585);
    \pic at (7.5,2.585) {valve=secondary};
    \node[below] at (7.5,2.585) {$V\ped{C}$};
    \draw (7.7,2.585) -- (11,2.585);
\end{tikzpicture}
```

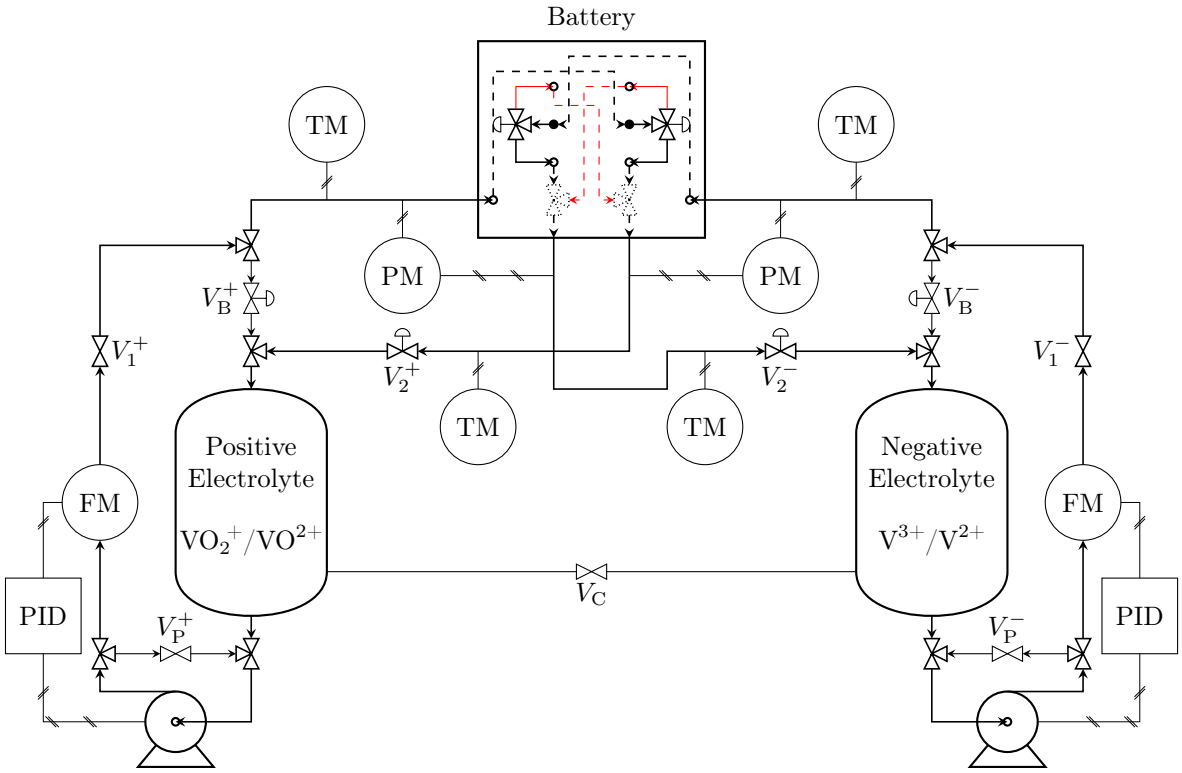


Figure 4: Complete PFD of a vanadium redox flow battery pilot plant produced by the listing 4 code.

Usage of Nodes

Starting from the next example, the second way outlined in the introduction to the examples will be used to draw schemes: units placement first and nodes usage to connect streams.

Listing 5: Scheme of an absorption process with solvent regeneration through steam stripping.

```
\begin{tikzpicture}[font=\footnotesize]
% Units Placement and Labelling
    \pic (absorption tower) at (2,4) {column=packed};
    \pic (stripping tower) at (10,4) {column=packed};
    \pic (mid heat exchanger) at (6,4) {heat exchanger};
    \pic (cooler) at (4,6.6) {heat exchanger};
    \pic (heater) at (7,6.6) {heat exchanger};
    \pic (pump) at (8.5,0.6) {centrifugal pump};
    \pic (lamination valve) at (8.5,6.6) {lamination valve};
    \node[above, rotate=-90, align=center] at
        (absorption tower-right)
        {absorption tower};
    \node[above, rotate=-90, align=center] at
        (stripping tower-right)
        {stripping tower};
    \node[left=5] at (mid heat exchanger-bottom)
        {heat integration};
    \node[below] at (lamination valve-anchor)
        {expansion};
    \node[below right] at (pump-right)
        {compression};
% Main Streams Connections and Labelling
    \draw[main stream] (0,1.4) --
        (absorption tower-bottom left);
    \node[below left] at (absorption tower-bottom left)
        {dirty gas};
    \draw[main stream] (0,6.6) --
        (absorption tower-top left);
    \node[below left] at (absorption tower-top left)
        {solvent make-up};
    \draw[main stream] (absorption tower-top) -- ++(0,1);
    \node[above left] at (absorption tower-top)
        {clean gas};
    \draw[main stream] (absorption tower-bottom) --
        ++(0,-0.5) -| (mid heat exchanger-bottom);
    \node[below right] at (absorption tower-bottom)
        {dirty solvent};
    \draw[main stream] (mid heat exchanger-top) |-
        (heater-internal tubes left);
    \draw[main stream] (heater-internal tubes right) --
        (lamination valve-inlet);
    \draw[main stream] (lamination valve-outlet) --
        (stripping tower-top left);
    \draw[main stream] (stripping tower-bottom) |-
        (pump-anchor);
    \node[below right] at (stripping tower-bottom)
```

```
        {clean solvent};
    \draw[main stream] (pump-left) |-
        (mid heat exchanger-internal tubes right);
    \draw[main stream]
        (mid heat exchanger-internal tubes left) --
        ++(-0.5,0) |- (cooler-internal tubes right);
    \draw[main stream] (cooler-internal tubes left) --
        (absorption tower-top right);
    \draw[main stream] (12,1.4) --
        (stripping tower-bottom right);
    \node[above right] at (stripping tower-bottom right)
        {clean steam};
    \draw[main stream] (stripping tower-top) |- ++(2,0.5);
    \node[above right] at (stripping tower-top)
        {dirty steam};
% Utility Streams Connections and Labelling
    \draw[utility stream] (4,7.6) -- (cooler-top);
    \node[above right] at (cooler-top)
        {cooling water};
    \draw[utility stream] (cooler-bottom) -- (4,5.6);
    \draw[utility stream] (7,7.6) -- (heater-top);
    \node[above right] at (heater-top)
        {heating steam};
    \draw[utility stream] (heater-bottom) -- (7,5.6);
\end{tikzpicture}
```

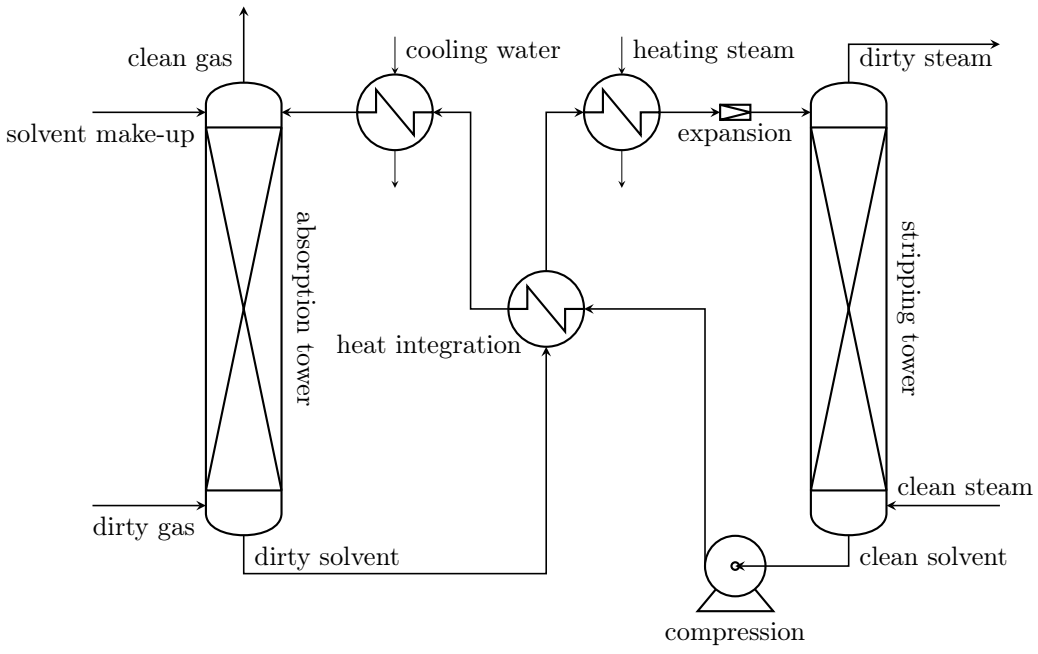


Figure 5: Scheme of an absorption process with solvent regeneration through steam stripping produced by the listing 5 code.

9 What Happens Next

As conceived, chemplants is just a toolbox to draw in an easier way chemical process schematics, but a lot of tools are still missing in the box.

At the time of the first writing of this documentation, only symbols and styles I had the need to use were defined. In the future, a lot of new units will be defined, but this is a work that will require time and study, so it will be done step by step. I have a list of the "most commonly used" units to define, but users are free (and invited) to send suggestion about new units which they would like to see in the chemplants palette.

Besides the mentioned modifications, I am always opened to suggestions which can improve the functionality and usability of chemplants. Users who have something to suggest, find errors and bugs or are simply happy to use this package can contact me writing to `elia24913@me.com` and possibly placing "chemplants" into the object field. I will be very grateful to these users.

References

[1] Alfonso Cacciatore and Mariano Calatozzolo. *Manuale di disegno di impianti chimici*. Torino: Edisco, 2018. ISBN: 978 88 441 2085 6.

[2] Jacques Crémer. *A Very Minimal Introduction to TikZ*. 2011. URL: `http://cremeronline.com/LaTeX/minimaltikz.pdf`.

[3] Claudio Fiandrino. *Introduzione all'uso di TikZ in ingegneria*. 2014. URL: `http://www.guitex.org/home/images/doc/GuideGuIT/introingtikz.pdf`.

[4] Lorenzo Pantieri. *LaTeXpedia*. 2017. URL: `http://www.lorenzopantieri.net/LaTeX_files/LaTeXpedia.pdf`.

[5] Lorenzo Pantieri and Tommaso Gordini. *L'arte di disegnare con LaTeX*. 2014.

[6] Till Tantau. *TikZ and PGF Manual. Version 3.1.1*. 2019. URL: `http://ctan.mirror.garr.it/mirrors/CTAN/graphics/pgf/base/doc/pgfmanual.pdf`.

[7] UNICHIM. *Impianti chimici. Simboli e sigle per schemi e disegni*. Manuale N. 6. Milano, 1994.