The **pfdicons** Package: Documentation

Aaron Drews
adrews@ucsd.edu

v1.1 from 2026/02/11

Contents

1 Introduction

This package provides Ti*k*Z shapes and a few supporting functions to draw icons and streams for process flow diagrams (PFDs) in chemical engineering. Unit operations provided by this package are expected to cover–in Turton's estimate–about 90% of fluid processing systems. Additional icons have been provided for solids processing, instrumentation, safety, and stream labeling and descriptions.

This package was developed to support students and faculty in undergraduate chemical engineering courses where excessive detail and customization are not necessary to convey meaning, as opposed to professionals who may require more customization than this package currently provides. Examples from several standard textbooks in undergraduate chemical engineering have been reproduced in the Examples section to illustrate the applicability of this package. A visual summary of all process units is provided at the end of this document as a quick reference.

2 Usage

To use this package simply provide

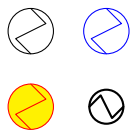
```
\usepackage{tikz}
\usepackage{pfdicons}
```

in your preamble. The `tikz` package must be loaded first. The `pfdicons` package automatically loads the `ifthen` package as well as the following Ti*k*Z libraries: `positioning`, `shapes`, `arrows.meta`, and `spath3`. Most shapes provided by the package are drawn within a `tikzpicture` environment using the basic syntax

```
\begin{tikzpicture}
  \node[<shape>] {};
\end{tikzpicture}
```

where `<shape>` is the desired process unit. Like most Ti*k*Z shapes the icons provided by `pfdicons` can be scaled, rotated, and colored in the usual ways:

```
1 \begin{tikzpicture}
2   \node[basic hx] at (0,0) {};
3   \node[basic hx, draw=blue] at (1,0) {};
4   \node[basic hx, draw=red, fill=yellow] at (0,-1) {};
5   \node[basic hx, scale=0.75, rotate=90, anchor=center] at (1,-1) {};
6 \end{tikzpicture}
```



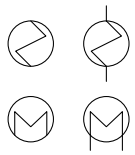
Two additional PGF keys are available for some process units: `unit int` which modifies the interior contents of the process unit, and `unit ext` which modifies the exterior of the process unit.[1] These keys can be used as

```
\node[<shape>, unit int=<intopt>, unit ext=<extopt>] {};
```

[1] A few other keys are available for specific units and are introduced in the documentation for those units.

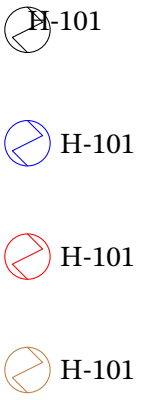
where <intopt> and <extopt> are interior and exterior options specific to the process unit. For most process units both keys can be used to combine effects. For example, the basic hx shape drawn above has keys U tube and stems which can be combined as follows:

```
\begin{tikzpicture}
  \node[basic hx] at (0,0) {};
  \node[basic hx, unit ext=util] at (1,0) {};
  \node[basic hx, unit fill=U tube] at (0,-1) {};
  \node[basic hx, unit fill=U tube, unit ext=util] at (1,-1) {};
\end{tikzpicture}
```



Since the flowsheet icons are intended to be geometric shapes *the curly braces should always be empty*; that is, text cannot be directly included as part of the node. Annotations and labels of shapes from pfdicons can be included as labels or separate nodes:

```
\begin{tikzpicture}
% No:
\node[basic hx] at (0,0) {H-101};

% Yes, as a separate node:
\node[basic hx, blue] at (0,-1.5) (g) {};
\node[right] at (g.east) {H-101};

% Yes, using standard label syntax:
\node[basic hx, red, label={right:H-101}] at (0,-3) {};

% Yes, with \usetikzlibrary{quotes} syntax:
\node[basic hx, brown, "H-101" right] at (0,-4.5) {};
\end{tikzpicture}
```



2.1 Anchors, anchor aliases, and polar coordinate specification

All process units have the standard cardinal and off-cardinal anchors which can be accessed using the standard Ti*k*Z anchor notation such as f.north or mycolumn.south east. Some units have additional anchors on half-cardinal points (*e.g.*, north north east or east south east) or special process unit features (*e.g.*, east jacket or actuator).

All cardinal anchors of custom shapes provided by the pfdicons package are **aliased with abbreviated versions**, *e.g.*, n for north, ene for east north east, *etc.* to allow for shortened code. Full anchor names can always be used to provide syntax consistency with core Ti*k*Z.

Most shapes support polar coordinate specifications such as mycolumn.75 (as of v1.1). The border on which anchors specified in this manner will appear *mostly* coincides with the anchors described above but some icons may have variations.

The following shapes are used to denote various anchors in this documentation:

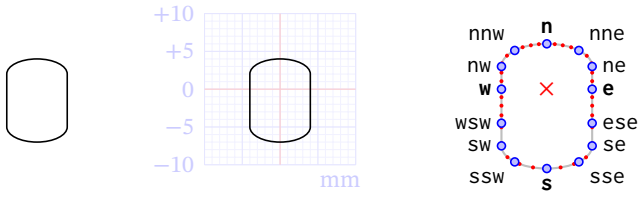
- cardinal and off-cardinal anchors such as north, south east, w, nw, *etc.*
- special anchors for specific icon locations, *e.g.* north motor, east jacket, *etc.*
- anchor border for anchors placed using polar notation such as shape.75, shape.-135, *etc.*
- × center anchor of node

3 Process units

3.1 Reactors

3.1.1 Tank reactor

```
\node[tank reactor] {};
```



The `tank reactor` shape provides a cylindrical tank reactor, also referred to as a stirred-tank reactor, continuous stirred-tank reactor (CSTRs), or autoclave. The `center` anchor is offset slightly to accommodate fill and interior options as described below. Several key modifiers are available:

Table 1: Key-value pairs for the **`tank reactor`** shape.

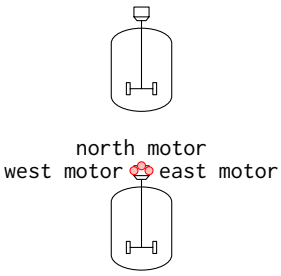
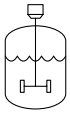
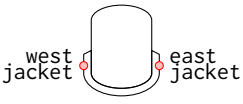
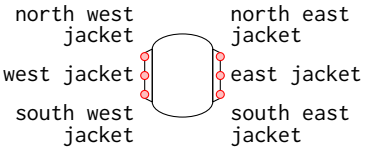
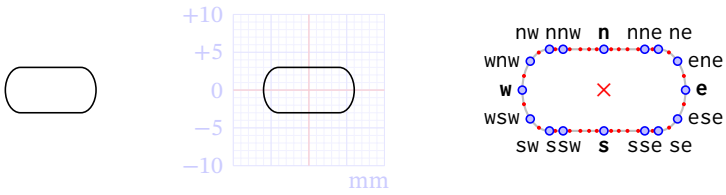
key=value	Description	Example
`unit int=stirred`	Adds a rounded stirrer with motor and mixing blade. Three additional anchors are specific to this option: `west motor`, `north motor`, and `east motor`.	
`unit int=liquid`	Adds wavy lines from `west` to `east` to indicate a liquid surface.	
`unit int=stirred liquid`	A combination of `unit int=stirred` and `unit int=liquid`.	
`unit ext=lower jacket`	Adds a jacket around the lower portion of the reactor. Two additional anchors are specific to this option: `west jacket` and `east jacket`.	

Table 1: Key-value pairs for the **tank reactor** shape *(cont'd)*

unit jacket	ext=side	Adds a jacket around the side of the reactor. Several additional anchors available: west jacket, north west jacket, south west jacket, east jacket, north east jacket, and south east jacket.	

3.1.2 Tube reactor

```
1 \node[tube reactor] {};
```



The tube reactor shape provides a horizontal cylinder with curved sides to be used for such reactors as plug flow reactors, packed bed reactors, fixed bed reactors, tubular reactors, and so forth. Several key-value pairs can be provided to indicate common fills and utility features:

Table 2: Key-value pairs for the **tube reactor** shape.

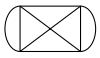
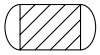
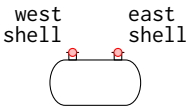
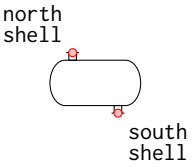
key=value	Description	Example
unit int=packed	Adds a cross representing a packed, dumped, or random fill within the reactor.	
unit int=fixed	Adds angled lines representing a fixed bed.	
unit int=tubular	Adds horizontal lines representing tubes for (*e.g.*) shell-and-tube reactors, double-pipe reactors, and so forth.	
unit ext=cis shell	Adds two shell-side stems, both on the north side with anchors west shell and east shell. Stems can be moved to the south side by rotating (rotate=180) or scaling (yscale=-1) the node.	

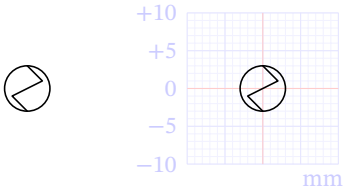
Table 2: Key-value pairs for the **tube reactor** shape *(cont'd)*

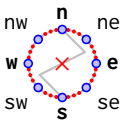
`unit ext=trans shell`	Adds two shell-side stems, one on the north side and one on the south side, with anchors `north shell` and `south shell`. Stem locations can be swapped by scaling the node (`yscale=-1` or `xscale=-1`).	

3.2 Heat exchangers

3.2.1 Basic heat exchanger

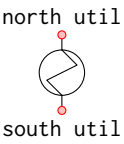
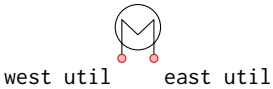
```
\node[basic hx] {};
```





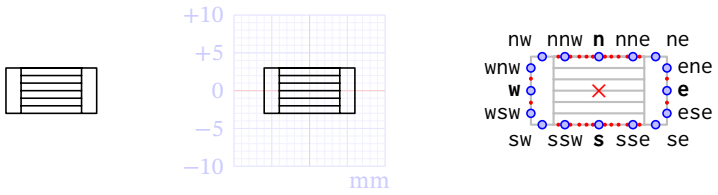
The `basic hx` shape provides a circle with a jagged interior line and represents a generic heat exchange unit. Only the four cardinal and intercardinal anchors are used; the half-cardinals `nnw`, `nne`, `ssw`, and `sse` anchors are not used. Protrusions from the inner "tube" line can be added to provide additional indication of tube-side fluids, typically to indicate utility fluids (hence the `util` value). Additional anchors can be used to place labels at ends of the protrusions.

Table 3: Key-value pairs for the **basic hx** shape.

key=value	Description	Example
`unit ext=util`	Adds protrusions from the inner tube, typically to indicate utility fluids. Additional anchors `south util` and `north util` are available at the terminals.	
`unit int=U tube`	Modifies interior to a generic U-tube orientation with both tube points on the same shell side. The `south west` and `south east` anchors are located at these tube points.	
`unit int=U tube, unit ext=util`	Modifies interior to U-tube orientation and adds protrusions, typically to indicate utility fluids. Additional anchors `west util` and `east util` are available at the terminals.	

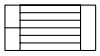
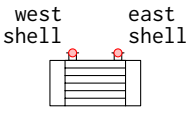
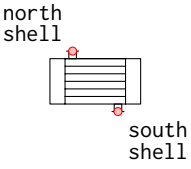
3.2.2 Shell and tube heat exchanger

```
\node[shell and tube hx] {};
```



The `shell and tube hx` shape provides a rectangle with an inner set of horizontal lines representing a shell and tube heat exchanger. Keys can modify the appearance to provide shell-side stems or modify the tube ends to represent two-pass or four-pass arrangements.

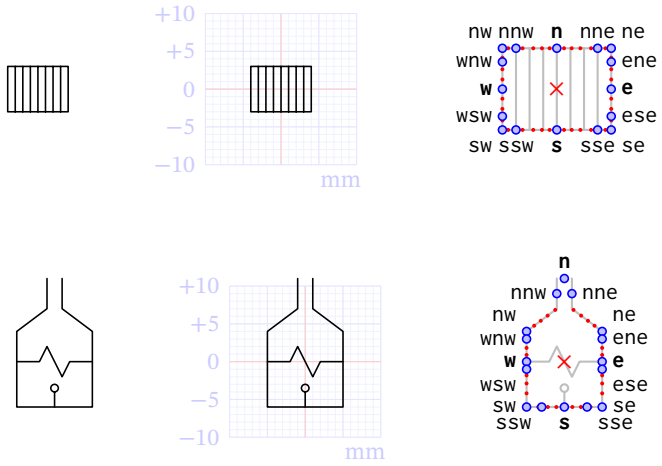
Table 4: Key-value pairs for the **`shell and tube hx`** shape.

key=value	Description	Example
`unit int=two pass`	Adds a horizontal line on the west shell side to indicate a two-pass heat exchanger.	
`unit int=four pass`	Adds two horizontal line on the west shell side and one on the east to indicate a four-pass heat exchanger.	
`unit ext=cis shell`	Adds two shell-side stems, both on the north side with anchors `west shell` and `east shell`. Stems can be moved to the south side by rotating (`rotate=180`) or scaling (`yscale=-1`) the node.	
`unit ext=trans shell`	Adds two shell-side stems, one on the north side and one on the south side, with anchors `north shell` and `south shell`. Stem locations can be swapped by scaling the node (`yscale=-1` or `xscale=-1`).	

3.2.3 Plate heat exchanger

```
\node[plate hx] {};
```

The `plate hx` shape provides a rectangle with an inner set of vertical lines representing a plate-and-frame heat exchanger. This unit has no `key=value` pairs for additional modification.

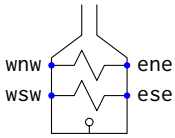
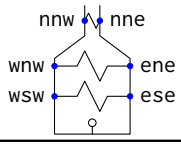


3.2.4 Fired heater

```
\node[fired hx] {};
```

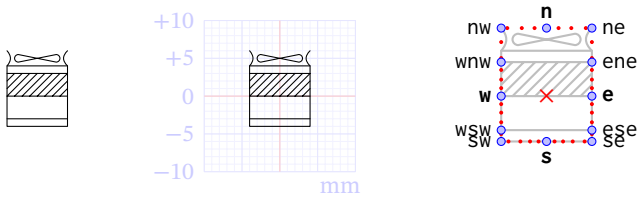
The `fired hx` shape provides a square unit with tapered chimney representing a fired heat exchanger or fired heater. A horizontal through-line represents the process tube; a small circular icon at the bottom indicates the combustion region of the heat exchanger. The default number of process tubes is one (`unit int=single` or omitted) but additional process tubes can be added using the `unit int` key. Inlets and outlets will shift to correspond to half-cardinal anchors as shown below.

Table 5: Key-value pairs for the **`fired hx`** shape.

key=value	Description	Example
`unit int=double`	Provides two tubes roughly corresponding to the radiative and convective zones.	
`unit int=triple`	Provides three tubes roughly corresponding to a radiative zone and two convective zones.	

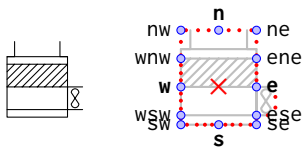
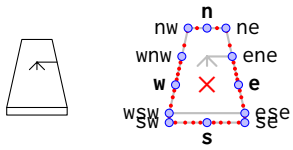
3.2.5 Cooling tower

```
\node[cooling tower] {};
```



The default cooling tower unit is of the induced draft design with a fan on top, packing in the middle, and a liquid reservoir at bottom. Options for forced and natural draft are available via the `unit ext` key, which also modifies anchors and anchor border, with `unit ext = induced` the same as not providing a key value.

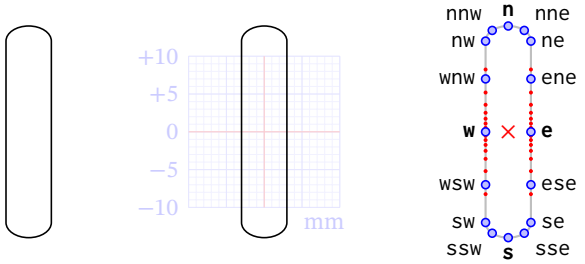
Table 6: Key-value pairs for the **`cooling tower`** shape.

key=value	Description	Example
`unit ext=forced`	Includes side fan and modifies vent stack for visual distinction.	
`unit ext=natural`	Natural draft tower with sprayer.	

3.3 Separation units

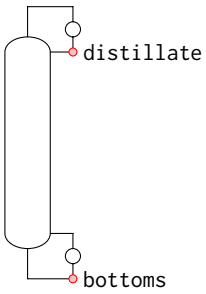
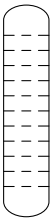
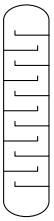
3.3.1 Column

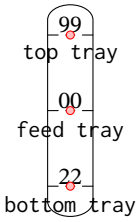
```
\node[column] {};
```



The `column` shape provides a vertical, elongated cylinder as a generic separation column. The default fill is empty but several `key=value` pairs are provided to represent the most common separation units. Several additional anchors and keys are available based on these settings.

Table 7: Key-value pairs for the **`column`** shape.

key=value	Description	Example
`unit ext=simple hx`	Provides stylized representation of condenser and reboiler along with anchors `distillate` and `bottoms` for stream connections. These anchors always exist but coincide with `north east` and `south east` when the `simple hx` key is omitted.	
`unit int=tray` `unit int=dashed tray`	Dashed, horizontal lines. Dashes will always have three segments and two openings.	
`unit int=weir tray`	Solid, horizontal lines with small risers to represent tray weirs.	

`unit int=numbered tray`
 	
associated keys:
`top tray=99`
`feed tray=00`
`bottom tray=22`
	Dashed, horizontal lines at top, feed, and bottom trays. Associated keys `top tray`, `feed tray` , and `bottom tray` can be used to indicate tray numbers **if column is not transformed**. If column is transformed then associated keys can be omitted and anchors `feed tray`, `top tray`, and `bottom tray` used for placement of additional text nodes.	

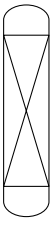
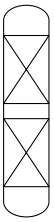
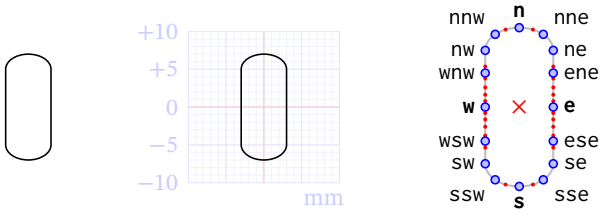
`unit int=packed`	Large cross symbol indicating random or dumped fill (packing).	

Table 7: Key-value pairs for the **column** shape *(cont'd)*

`unit int=double packed`	Two sections of random or dumped fill (packing) separated by a small, central gap.	

3.3.2 Vessel

```
\node[vessel] {};
```



The `vessel` shape provides a vertical cylinder as a generic vessel. Such a vessel can be, for example, down-scaled and rotated 90° can represent a product tank or a phase separator after a condenser. The default fill is empty but several `key=value` pairs are provided to represent two common vapor-liquid units as well as liquid accumulation tanks. Several additional anchors are available based on these settings.

Table 8: Key-value pairs for the **vessel** shape.

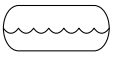
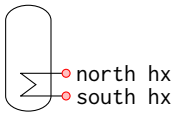
key=value	Description	Example
`unit int=liquid`	Adds wavy lines across the middle of the vessel to indicate a liquid level.	
`unit int = liquid rotated`	Adds wavy lines down the center of the vessel to indicate a liquid level. This key is intended to be used with a rotated node such as `\node[vessel, rotate=90, unit int= liquid rotated] {};`	
`unit int=phase sep`	Adds a knock-down element near the `west` anchor and a dashed, horizontal line near the top of the vessel to indicate a demister pad.	

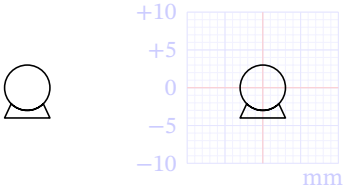
Table 8: Key-value pairs for the **vessel** shape *(cont'd)*

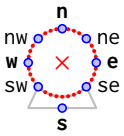
`unit ext=simple hx`	Adds a simplified heat transfer element to the bottom portion of the vessel. Two additional anchors, `north hx` and `south hx`, are located at the exterior points of the heat exchanger tubes.	

3.4 Fluid transport

3.4.1 Centrifugal pump

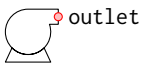
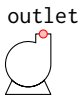
```
\node[centrifugal pump] {};
```





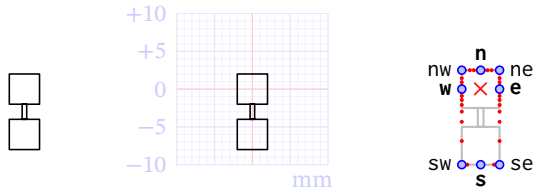
The `centrifugal pump` shape provides a circular element with a triangular base to represent a centrifugal pump. An additional half-circle can be added using the `unit int` key to indicate the pump inlet and a rectangular outlet direction can be added using the `unit ext` key.

Table 9: Key-value pairs for the **`centrifugal pump`** shape.

key=value	Description	Example
`unit int=inlet west`	Adds a half-circle around the `center` anchor to accept an inlet stream to enter from the west.	
`unit int=inlet east`	Adds a half-circle around the `center` anchor to accept an inlet stream to enter from the east.	
`unit int=inlet north`	Adds a half-circle around the `center` anchor to accept an inlet stream to enter from the north.	
`unit int=inlet south`	Adds a half-circle around the `center` anchor to accept an inlet stream to enter from the south. Note that there is no means of removing the triangular base; such an inlet stream should be drawn as going through the base.	
`unit ext=outlet east`	Adds a rectangular protrusion facing east to represent an outlet. Also adds the `outlet` anchor in the center of the outlet box.	
`unit ext=outlet west`	Adds a rectangular protrusion facing west to represent an outlet. Also adds the `outlet` anchor in the center of the outlet box.	
`unit ext=outlet north east`	Adds a rectangular protrusion facing north on the east side of the pump to represent an outlet. Also adds the `outlet` anchor in the center of the outlet box.	
`unit ext=outlet north west`	Adds a rectangular protrusion facing north on the west side of the pump to represent an outlet. Also adds the `outlet` anchor in the center of the outlet box.	

3.4.2 Reciprocating pump

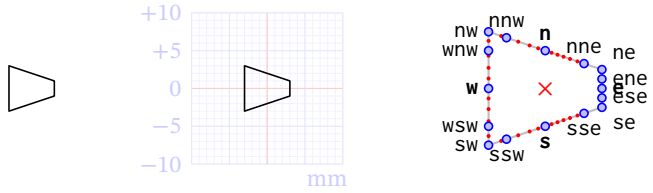
```
\node[reciprocating pump] {};
```



The `reciprocating pump` shape provides two square elements–a pump head and motor–connected by a simple shaft. Most anchors are placed relative to the pump head. This unit has no `key=value` pairs for additional modification.

3.4.3 Compressor

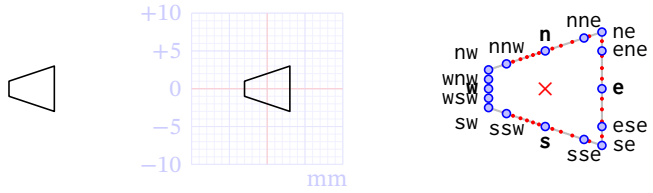
```
\node[compressor] {};
```



The `compressor` shape provides a quadrilateral to represent a gas-phase compressor. This unit has no `key=value` pairs for additional modification. If mirrored along the x-axis (*e.g.*, by using something like `xscale=-1` or `rotate=90`) then the `compressor` shape is identical to the `turbine` shape but with different anchor points.

3.4.4 Turbine

```
\node[turbine] {};
```



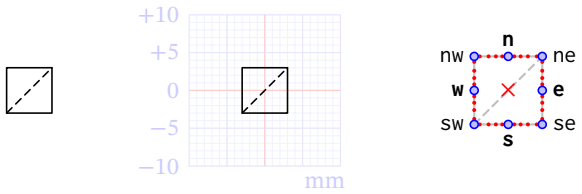
The `turbine` shape provides a quadrilateral to represent a gas-phase turbine. This unit has no `key=value` pairs for additional modification. If mirrored along the x-axis (*e.g.*, by using something like `xscale=-1` or `rotate=90`) then the `turbine` shape is identical to the `compressor` shape but with different anchor points.

4 Solids processing

4.1 Solids separation

4.1.1 Filter

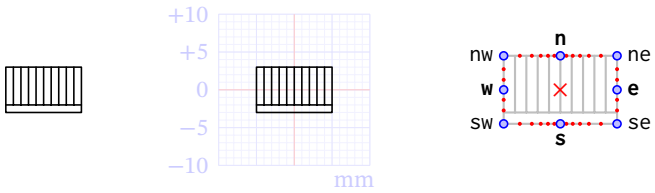
```
\node[filter] {};
```



The `filter` shape provides a square outline with dashed divider to represent a generic filter operation for solids. This unit has no `key=value` pairs for additional modification.

4.1.2 Press filter

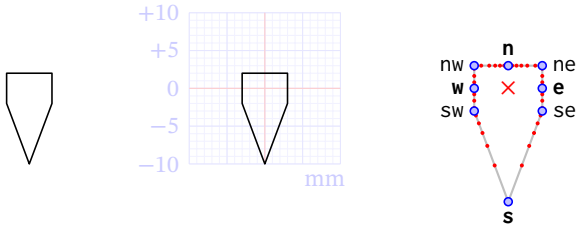
```
\node[press filter] {};
```



The `press filter` shape provides a rectangular outline with vertical lines to represent a series of parallel filter sheets pressed together with a frame as a filter operation for solids, usually for liquid/solid. This unit has no `key=value` pairs for additional modification.

4.1.3 Cyclone

```
\node[cyclone] {};
```

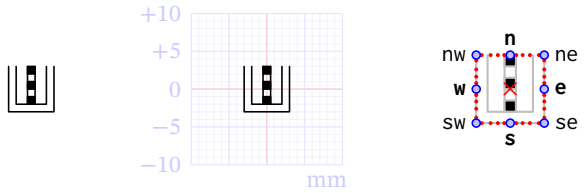


The `cyclone` shape provides a rectangular top with tapered bottom as a gas/solid separation unit. This unit has no `key=value` pairs for additional modification.

4.1.4 Centrifuge

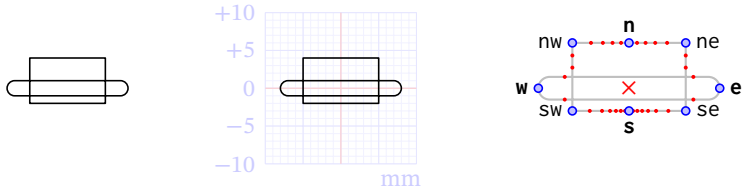
```
\node[centrifuge] {};
```

The `centrifuge` shape provides a square outline with offset inner basket, usually for liquid/solid separations. The inner detail represents a mechanical shaft to spin the inner basket. This unit has no `key=value` pairs for additional modification.



4.1.5 Belt dryer

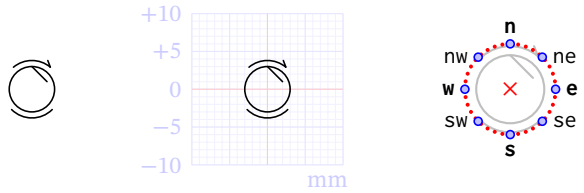
```
\node[belt dryer] {};
```



The `belt dryer` shape provides a long rectangle with oval belt for solids drying. This unit has no `key=value` pairs for additional modification.

4.1.6 Rotary dryer

```
\node[rotary dryer] {};
```



The `rotary dryer` shape provides a circle with interior scraper, exterior heater at bottom, and rotation indicator at top. Anchor positioning for this node is slightly outside the drawing itself to provide uniformity of appearance, for example allowing streams entering the top or bottom to not overlap the exterior arcs. This unit has no `key=value` pairs for additional modification.

4.2 Solids generation

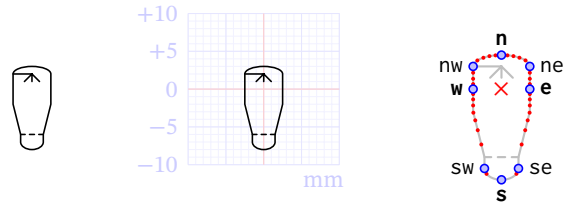
4.2.1 Granulator or agglomerator

```
\node[granulator] {};
```

or

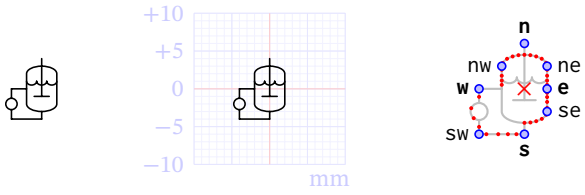
```
\node[agglomerator] {};
```

The `granulator` shape provides a spray drum with gas filter at bottom. This device is also referred to as an agglomerator and is aliased as such for the purpose of generating nodes in typical picture environments. The sprayer inlet can be connected at the `nw` (`north west`) anchor if desired. The underlying PGF shape is only known as a granulator. This shape has no `key=value` pairs for additional modification.



4.2.2 Crystallizer

```
\node[crystallizer] {};
```

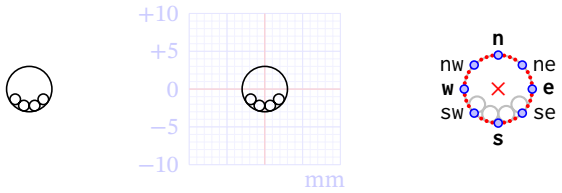


The `crystallizer` shape provides a small liquid reactor with integrated stirrer and recycle loop with integrated heater for solids production. Anchors `sw` and `s` are positioned for typical feed and product line connections. This unit has no `key=value` pairs for additional modification.

4.3 Solids size reduction

4.3.1 Ball mill

```
\node[ball mill] {};
```

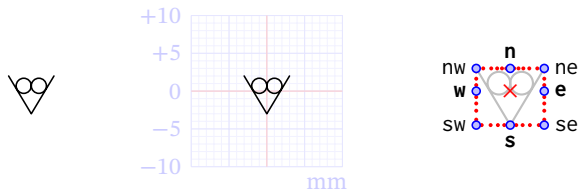


The `ball mill` shape provides a circle with smaller interior circles (balls) for solids pulverization. At large scale factors the interior circles may lose their alignment with the mill exterior or other circles. This unit has no `key=value` pairs for additional modification.

4.3.2 Roll crusher

```
\node[roll crusher] {};
```

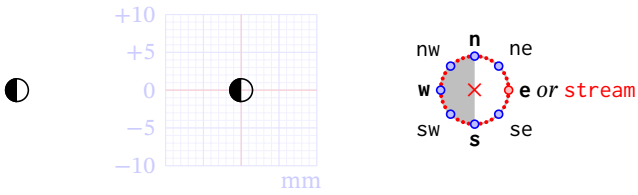
The `roll crusher` shape provides a conical feeder with smaller interior circles (rollers) for solids pulverization. At large scale factors the interior circles may lose their alignment with the feeder exterior. This unit has no `key=value` pairs for additional modification.



5 Streams

5.1 Feed terminal

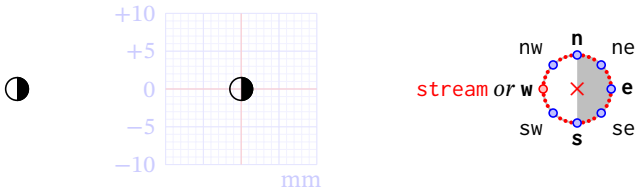
```
\node[feed] {};
```



The `feed` shape provides a half-filled circle representing a process feed point. The special anchor `stream` corresponds to the `east` anchor and is provided as an optional method to define stream starting points. This unit has no `key=value` pairs for additional modification.

5.2 Product terminal

```
\node[product] {};
```

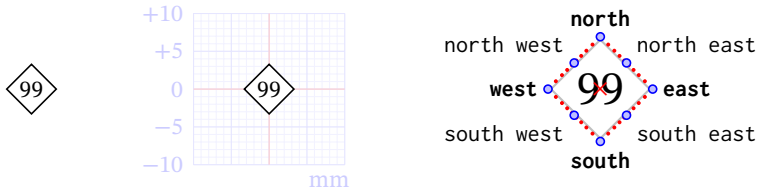


The `product` shape provides a half-filled circle representing a process product point. The special anchor `stream` corresponds to the `west` anchor and is provided as an optional method to define stream ending points. This unit has no `key=value` pairs for additional modification.

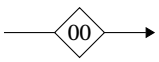
5.3 Stream numbers

```
\node[sid] {99};
```

The `sid` (read “ess eye dee” for “stream identification”) shape provides a modified `diamond` shape from the `shapes` library for the purpose of labeling streams. The anchors **are not aliased** with their abbreviations because *(a)* this is not a custom shape of the `pfdicons` package and *(b)* the `sid` shape is typically used within a line (stream) without accessing its anchors. The fill of `sid` is white which allows it to be conveniently placed within a stream-drawing command such as




```
\begin{tikzpicture}
  \draw[->] (0,0) -- node[sid] {00} (2,0);
\end{tikzpicture}
```

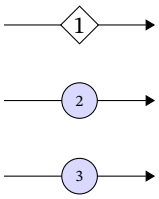


The `sid` shape has no `key=value` pairs to modify its appearance but you can modify the appearance by redefining its `tikzstyle`, which by default is

```
\tikzstyle{sid} = [diamond, draw, solid, fill=white, text badly centered, inner sep=1pt, font=
    \footnotesize]
```

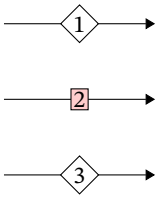
Modifications to this definition can occur either in the preamble (to affect all drawings) or within the document (to affect individual drawings or lines). For example, if you'd like to use circular identifiers with a blue fill and more space around bigger numbers then you could do something like this:

```
\begin{tikzpicture}
  \draw[->] (0,0) -- node[sid] {1} (2,0);

  \tikzstyle{sid} = [circle, draw, solid, fill=blue!15, text badly
      centered, inner sep=3pt, font=\tiny]

  \draw[->] (0,-1) -- node[sid] {2} (2,-1);

  \draw[->] (0,-2) -- node[sid] {3} (2,-2);
\end{tikzpicture}
```



You can also modify individual parameters while retaining others by passing additional modifiers within the `sid` node usage, like this:

```
\begin{tikzpicture}
  \draw[->] (0,0) -- node[sid] {1} (2,0);

  \draw[->] (0,-1) -- node[sid, rectangle, fill=red!20] {2} (2,-1);

  \draw[->] (0,-2) -- node[sid] {3} (2,-2);
\end{tikzpicture}
```



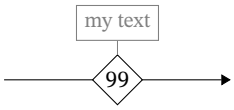
5.4 Stream data pins

Several Ti*k*Z pins–`pfdpin1`, `pfdpin2`, and `pfdpin3`–have been defined to support display of stream properties such as temperature, pressure, mass flow rate, *etc.* Choose the `pfdpinN` such that `N` is the number of information regions desired in the pin. These pins are modifications of the `rectangle split` pin and follow similar syntax, namely

```
node[pfdpin1={location:my text}] {}
```

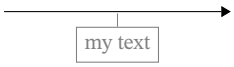
where `location` can be `above`, `below`, `left`, or `right`. In the labeling of streams this node will typically be included as part of stream drawing statement along with `sid`, for example

```
\begin{tikzpicture}
  \draw[->] (0,0) -- node[sid, pfdpin1={above:my text}] {99} (3,0);
\end{tikzpicture}
```



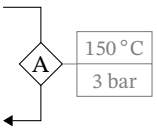
It is not required that the `sid` node specification be provided. However the pin connector will connect to the exterior bounding box of the resulting invisible node, so options such as `inner sep` and `minimum height` or `minimum width` must be set to bring the pin to the desired connection point in that event. For example,

```
\begin{tikzpicture}
  \draw[->] (0,0) -- node[inner sep=0pt, pfdpin1={below:my text}] {}
      (3,0);
\end{tikzpicture}
```



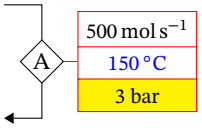
Multiple entries–up to a maximum of 3– can be obtained by using `pfdpin2` or `pfdpin3`, with corresponding text entries separated using `/`:

```
% \usepackage{siunitx} to enable \qty function
\begin{tikzpicture}
  \draw[->] (0,0) -- (0.5,0) |- node[sid, pos=0.25, pfdpin2={right:\qty
      {150}{\celsius}/3 bar}] {A} (0,-1.5);
\end{tikzpicture}
```



Functions have been provided to modify several aspects of the stream data pin without undertaking redefinition of the underlying `rectangle split` pin. For example,

```
\begin{tikzpicture}
  \setpfdpindrawcolor{red}
  \setpfdpintextcolor{black}
  \setpfdpinsecondtextcolor{blue}
  \setpfdpinthirdboxfillcolor{yellow}
  % appropriate pfdpin3 definition
\end{tikzpicture}
```



The full list of available modification functions for `pfdpinN` is provided in the table below.

Table 10: Modifications for the **`pfdpinN`** stream labels.

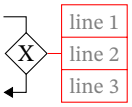
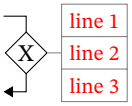
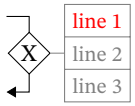
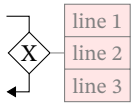
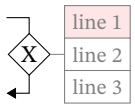
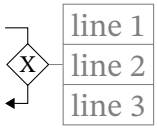
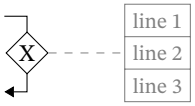
Function	Description	Example
`\setpfdpindrawcolor{<color>}` `\resetpfdpindrawcolor`	Set box and pin stroke color. Default is `gray`.	`\setpfdpindrawcolor{red}` 
`\setpfdpintextcolor{<color>}` `\resetpfdpintextcolor`	Set all pin text colors. Default is `gray`.	`\setpfdpintextcolor{red}` 

Table 10: Modifications for the **pfdpinN** stream labels. *(cont'd)*

`\setpfdpinfirsttextcolor{<color>}` `\resetpfdpinfirsttextcolor`	Set pin text color of first (top) box. Default is `gray`. Change `first` to `second` or `third` to target middle or bottom boxes.	`\setpfdpinfirsttextcolor{red}` 
`\setpfdpinboxfillcolor{<color>}` `\resetpfdpinboxfillcolor`	Set all box fill colors. Default is `none`.	`\setpfdpinboxfillcolor{red!10}` 
`\setpfdpinfirstboxfillcolor{<color>}` `\resetpfdpinfirstboxfillcolor`	Set first (top) box fill colors. Default is `none`. Change `first` to `second` or `third` to target middle or bottom boxes.	`\setpfdpinfirstboxfillcolor{red!10}` 
`\setpfdpintextsize{<size>}` `\resetpfdpintextsize`	Set all pin text size. Default is **`\scriptsize`**.	`\setpfdpintextsize{\large}` 
`\setpfdpindistance{<dist>}` `\resetpfdpindistance`	Set length of pin connector. Default is `5pt`.	`\setpfdpindistance{2cm}` 
`\setpfdpinconnector{<style>}` `\resetpfdpinconnector`	Set connector line style. Default is `solid`. Try increasing pin distance if effect is not observable.	`\setpfdpinconnector{dashed}` 
`\resetpfdpinproperties`	Reset all `pfdpin` properties to the defaults noted above.	

5.5 Stream arrow tips

The arrow tip is set using the standard TikZ syntax

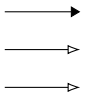
```
\tikzset{>=<tip style>}
```

When loaded the `pfdicons` package sets `<tip style>` to `Triangle` as

```
\tikzset{>=Triangle}
```

You can change this setting in the preamble or within the document itself to whatever arrow tip you prefer (see §16.5 in the PGF manual for a summary of arrow tips). For example,

```
\begin{tikzpicture}
  \draw[->] (0,0) -- (1,0);

  \tikzset{>=Latex[open]}
  \draw[->] (0,-0.5) -- (1,-0.5);
  \draw[->] (0,-1) -- (1,-1);
\end{tikzpicture}
```



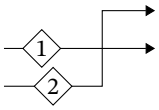
You can also make modifications to individual arrow tips simply by using a different tip for that particular arrow:

```
\begin{tikzpicture}
  \draw[->] (0,0) -- (1,0);
  \draw[-{Stealth[red]}] (0,-0.5) -- (1,-0.5);
  \draw[->] (0,-1) -- (1,-1);
\end{tikzpicture}
```

5.6 Stream crossings

When two streams cross it's desirable to indicate their status as distinct streams to avoid confusion with junctions or mixing points. Consider the following example:

```
\begin{tikzpicture}
  \draw[->] (0,0.5) -- node[sid, pos=0.25]{1} (2,0.5);
  \draw[->] (0,0) -- node[sid]{2} (1.5,0) -- (1.5,1) -- (2,1);
\end{tikzpicture}
```

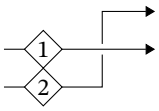


It may not be immediately obvious that the intersection between streams 1 and 2 is not a four-way pipe junction with two inlets and two outlets. To make such distinction a *bridge* can be used, most commonly in the form of a break in one stream, an arc of one stream, or a combination of a break and an arc. The `pfdicons` package provides a `bridge` style which uses the `spath3` package to detect crossings and create bridges. The workflow to use the `bridge` style is as follows:

1. Identify that a crossing has occurred (*e.g.*, by completing a drawing as done above). Determine which stream is to be drawn continuously (the `over` stream) and which stream is to be broken (the `under` stream).
2. Re-define the original streams using the **`path`** operation, removing all decorations.
3. Use the `bridge` key with `tikzset` to set the `over` and `under` paths as `\tikzset{bridge={over}{under}}`.
4. Re-draw the streams with a **`draw`** operation, replacing any previously removed decorations.

To demonstrate this process on the previous example we select stream 1 as the `over` stream (Turton recommends horizontal streams be continuous and vertical streams be broken). The remaining steps are completed as follows:

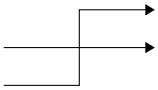
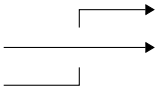
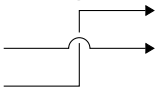
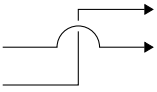
```
\begin{tikzpicture}
  % \draw[->] (0,0.5) -- node[sid, pos=0.25]{1} (2,0.5);
  % \draw[->] (0,0) -- node[sid]{2} (1.5,0) -- (1.5,1) -- (2,1);
  \path[spath/save=over] (0,0.5) -- (2,0.5);
  \path[spath/save=under] (0,0) -- (1.3,0) -- (1.3,1) -- (2,1);
  \tikzset{bridge={over}{under}}
  \draw[->, spath/use=over] node[sid, pos=0.15]{1};
  \draw[->, spath/use=under] node[sid, pos=0.1]{2};
\end{tikzpicture}
```



Lines 4 and 5 define the original streams using the **path** operation. Line 6 sets the `bridge` key to identify the `over` and `under` stream (the names here could be anything as long as the first input is the "over" path and the second input is the "under" path). Lines 7 and 8 re-draw the streams with an appropriate `draw` operation, adding the stream labels as well. Notice that the positioning of the stream labels must be modified slightly from the original syntax; getting the labels *precisely* where they were before the bridging operation can be challenging (but is usually not necessary).

The bridge style can be modified by using `\tikzset` to set the value of two keys, `bridge gap` and `bridge radius`, as summarized below.

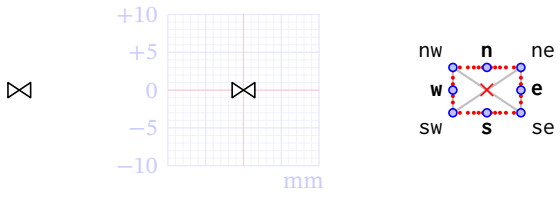
Table 11: Key-value pairs for use with stream crossings

key	Description	Example
`bridge gap`	Adjusts the amount of space around a stream break. The default value is `4pt`.	`\tikzset{bridge gap=0pt}`  `\tikzset{bridge gap=15pt}` 
`bridge radius`	Adjusts the radius of an arc bridge. The default value is `0pt` which produces a straight line.	`\tikzset{bridge radius=8pt}`  `\tikzset{bridge radius=20pt}` 

6 Control elements

6.1 Valve

```
\node[valve] {};
```

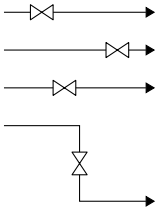


The `valve` shape provides two opposing triangles as a generic valve shape. Unlike most other shapes the fill color of a `valve` shape is white which allows it to be conveniently placed within a line-drawing operation such as

```
\begin{tikzpicture}
  \draw[->] (0,0) -- node[valve] {} (2,0);
\end{tikzpicture}
```

Placement of the valve on the stream can be modified using the standard placement keys (*e.g.*, `middle` or `near start`) or by the `pos=number` key. The valve can also be rotated to an arbitrary angle using the `rotate=degree` key. For example:

```
\begin{tikzpicture}
  \draw[->] (0,0) -- node[valve, near start] {} (2,0);
  \draw[->] (0,-0.5) -- node[valve, near end] {} (2,-0.5);
  \draw[->] (0,-1) -- node[valve, pos=0.4] {} (2,-1);
  \draw[->] (0,-1.5) -- (1,-1.5) -- node[valve, rotate=90] {} (1,-2.5)
      -- ++(1,0);
\end{tikzpicture}
```



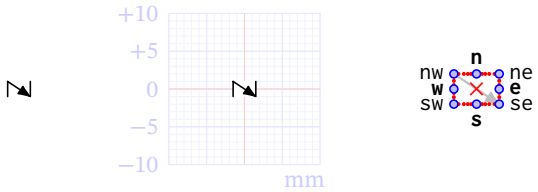
A generic actuator can be added to the `valve` shape by using the `unit ext` key as summarized below.

Table 12: Key-value pairs for the **`valve`** shape.

key=value	Description	Example
`unit ext=actuator`	Adds a hemisphere-capped protrusion as an indicator of a generic actuation device. The `actuator` anchor is placed at the top of the hemisphere for electrical connections. Polar coordinate border adjusts to include actuator; no other anchors are shifted.	`actuator` 

6.2 Non-return valve

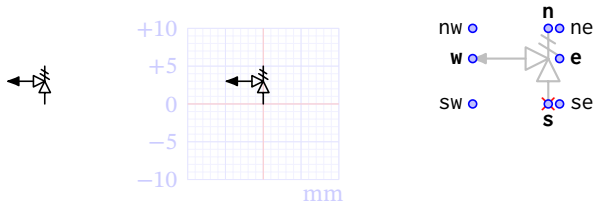
```
\node[nrv] {};
```



The `nrv` shape provides a slanted arrow between vertical lines as a no-return (or non-return) valve. The arrow tip is set to `Latex[round]` to distinguish the valve operation from other streams. Generally the direction of the arrow should indicate the allowable direction of flow; valve direction can be flipped with `xscale=-1`. This valve is intended to be placed on streams just as `valve` is, and thus the interior fill is white by default. This unit has no `key=value` pairs for additional modification.

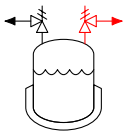
6.3 Pressure relief valve

```
\node[prv] {};
```



The `prv` shape provides a pressure relief valve. The shape's center is offset to provide simple alignment in most use cases where the valve is located above another node, for example

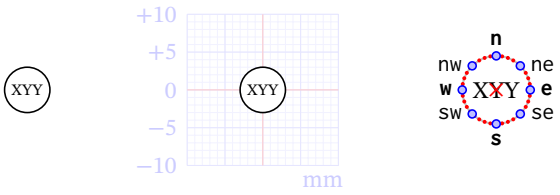
```
1 \begin{tikzpicture}
2   \node [tank reactor, unit int=liquid, unit ext=lower jacket] (r) {};
3   \node [prv] at (r.nnw) {};
4   \node [prv, red, xscale=-1] at (r.nne) {};
5 \end{tikzpicture}
```



The arrow tip is set to `Latex[round]` to distinguish the vent line from other streams. This unit has no `key=value` pairs for additional modification and does not support polar coordinate notation.

6.4 Controller

```
\node[ctrl, ctrl sense=X, ctrl type=YY] {};
```



The `ctrl` shape provides a flexible controller shape for providing detailed controller information. The various `key=value` pairs shown below allow customization of measured variable, controller type, controller alarm, controller display, and controller numerical identification. Shape conventions follow those of Towler and Sinnott, 2013, Ch. 5.3.

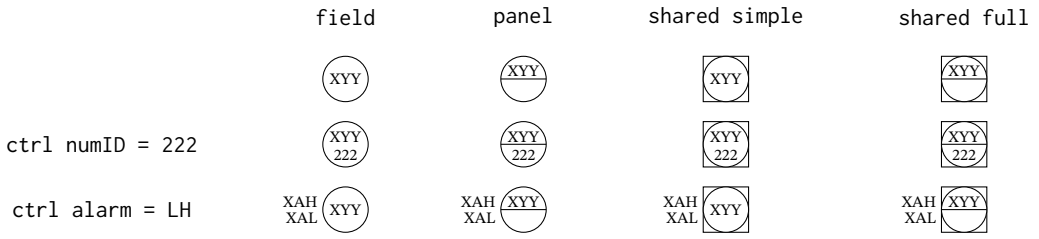
Table 13: Key-value pairs for the **`ctrl`** shape.

key	Description
`ctrl sense`	Sets the process variable to be measured such as `T` = temperature, `P` = pressure, `F` = flow, *etc.* This key must be combined with a controller type (below) to appear on the controller. This signal variable is copied to the `alarm` setting if provided.

Table 13: Key-value pairs for the **ctrl** shape *(cont'd)*

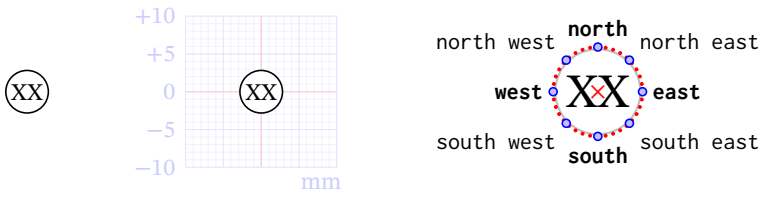
`ctrl type`	Sets the type of controller. Typical types are `I` = indicating, `C` = controlling, `IC` = indicating controlling, `RC` = recording controlling, `T` = transmitting, `V` = final control element.
`ctrl display`	This key sets the shape outline–round or square–and the location of text. Allowable values are `field` = field controller (default, round), `panel` (round with horizontal line), `shared simple` (round and square together), and `shared full` (round, square, and horizontal line all together).
`ctrl alarm`	Alarm indicator. Allowable values are `L` = low alarm only, `H` = high alarm only, `LH` or `HL` = both high and low alarms.
`ctrl alarm side`	Sets the side on which alarm indicator will be drawn. Allowable values are `left` = left side, `right` = right side.
`ctrl numID`	Numerical value to serve as identification, up to 3 digits.

Below are several examples. Combinations of `ctrl sense` and `ctrl type` should not exceed four characters to avoid sizing issues. In these examples, `ctrl sense=X` and `ctrl type=YY` for illustration purposes.



6.5 Electrical instruments

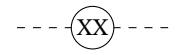
```
\node[elec] {XX};
```



The `elec` shape provides a modified `circle` shape from the `shapes` library for the purpose of indicating (in a general way) electrical instrumentation such as analyzers, transmitters, and controllers (see the

`ctrl` node for more advanced controller design). The anchors **are not aliased** with their abbreviations because this is not a custom shape of the `pfdicons` package. The fill of `elec` is white which allows it to be conveniently placed within a stream-drawing command such as

```
\begin{tikzpicture}
  \draw[dashed] (0,0) -- node[elec] {XX} (2,0);
\end{tikzpicture}
```

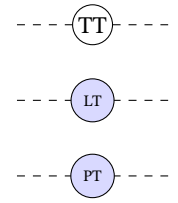


The `elec` shape has no `key=value` pairs to modify its appearance but you can modify the appearance by redefining its `tikzstyle`, which by default is

```
\tikzstyle{elec} = [circle, draw, solid, fill=white, text badly centered, inner sep=1pt, font=
    \footnotesize]
```

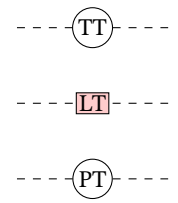
Modifications to this definition can occur either in the preamble (to affect all drawings) or within the document (to subsequent drawings or lines). For example, if you'd like to use circular identifiers with a blue fill and more space around bigger numbers then you could do something like this:

```
\begin{tikzpicture}
  \draw[dashed] (0,0) -- node[elec] {TT} (2,0);

  \tikzstyle{elec} = [circle, draw, solid, fill=blue!15, text badly
      centered, inner sep=3pt, font=\tiny]

  \draw[dashed] (0,-1) -- node[elec] {LT} (2,-1);

  \draw[dashed] (0,-2) -- node[elec] {PT} (2,-2);
\end{tikzpicture}
```



You can also modify individual parameters while retaining others by passing additional modifiers within the `sid` node usage, like this:

```
\begin{tikzpicture}
  \draw[dashed] (0,0) -- node[elec] {TT} (2,0);

  \draw[dashed] (0,-1) -- node[elec, rectangle, fill=red!20] {LT}
      (2,-1);

  \draw[dashed] (0,-2) -- node[elec] {PT} (2,-2);
\end{tikzpicture}
```



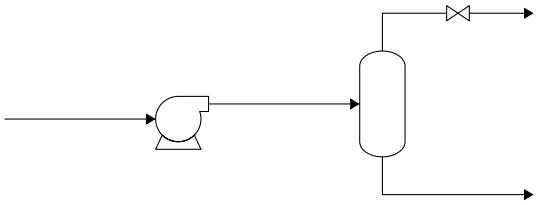
7 Examples

The following examples use the `pfdicons` package to replicate process flow diagrams and other diagrams found in several standard undergraduate textbooks. The examples are generally grouped according to the (approximate) course in which they occur: material and energy balances, thermodynamics, reaction engineering, separations, process controls, and capstone design.

Caution: Sometimes copy/pasting example code will introduce artifacts (mostly whitespaces) into the pasted code which are not apparent in this document. If this occurs you can (1) check and remove any erroneous whitespaces in your pasted code, (2) type out the examples here by hand, or (3) copy the code directly from the `pfdicons-doc.tex` file on the CTAN page for this package.

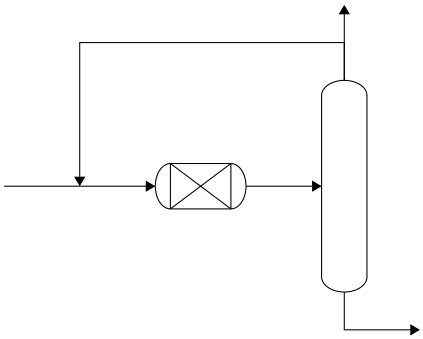
7.1 Material and energy balances

EXAMPLE 1: A basic separation process consisting of a pump, flash unit, and valve. Source: Himmelblau, D.; Riggs, J. *Basic Principles and Calculations in Chemical Engineering*, 8th ed.; Prentice Hall, 2012.



```
\begin{tikzpicture}
  \node[centrifugal pump, unit ext=outlet east] (c) {};
  \node[vessel, right=2cm of c.outlet] (s) {};
  \draw[<-] (c.west) -- ++(-2,0);
  \draw[->] (c.outlet) -- (s.west);
  \draw[->] (s.north) -- ++(0,0.5) -- node[valve] {} ++(2,0);
  \draw[->] (s.south) |- ++(2,-0.5);
\end{tikzpicture}
```

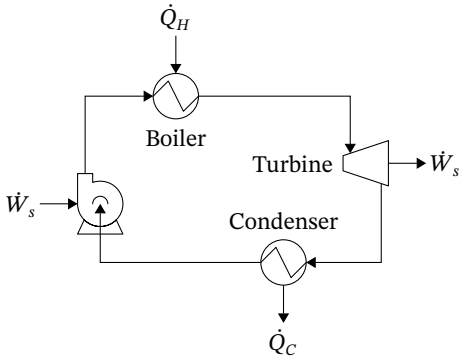
EXAMPLE 2: A simplified ethylene dichloride synthesis process illustrating the reactor-separator-recycle-purge (RSRP) arrangement. Source: Himmelblau, D.; Riggs, J. *Basic Principles and Calculations in Chemical Engineering*, 8th ed.; Prentice Hall, 2012.



```
\begin{tikzpicture}
  \node[tube reactor, unit int=packed] (r) {};
  \node[column, right=of r] (s) {};
  \coordinate [left=of r] (m);

  \draw[<-] (r.west) -- ++(-2,0);
  \draw[->] (r.east) -- (s.west);
  \draw[->] (s.south) |- ++(1,-0.5);
  \draw[->] (s.north) -- ++(0,1);
  \draw[->] (s.north) -- ++(0,0.5) -| (m);
\end{tikzpicture}
```

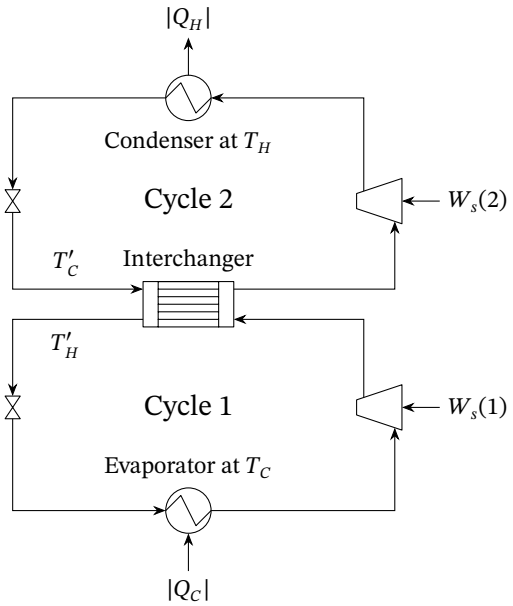
7.2 Thermodynamics

EXAMPLE 3: A simple steam power plant. Source: Figure 8.1 on pg. 270 in Smith, J.M.; Van Ness, H.C.; Abbot, M.M. *Introduction to Chemical Engineering Thermodynamics*, 6th ed.; McGraw-Hill, 2001.




```
\begin{tikzpicture}[font=\footnotesize]
  % Units
  \node [basic hx, rotate=90] (boiler) {};
  \node [turbine, below right=1cm and 2cm of boiler] (turbine) {};
  \node [basic hx, rotate=90, below left=of turbine] (cond) {};
  \node [centrifugal pump, below left=of boiler, unit int=inlet south, unit ext=outlet
      north west] (pump) {};

  % Streams
  \draw[->] (boiler.south) -| (turbine.nnw);
  \draw[->] (turbine.sse) |- (cond.south);
  \draw[->] (cond.north) -| (pump.center);
  \draw[->] (pump.outlet) |- (boiler.north);

  % Labels
  \node[below] at (boiler.west) {Boiler};
  \node[above] at (cond.east) {Condenser};
  \node[left] at (turbine.west) {Turbine};
  \draw[<-] (boiler.east) -- node[pos=1.5] {$\dot{Q}_H$} ++(0,5mm);
  \draw[->] (cond.west) -- node[pos=1.5] {$\dot{Q}_C$} ++(0,-5mm);
  \draw[->] (turbine.east) -- node[pos=1.5] {$\dot{W}_s$} ++(5mm,0);
  \draw[<-] (pump.west) -- node[pos=1.5] {$\dot{W}_s$} ++(-5mm,0);
\end{tikzpicture}
```

EXAMPLE 4: A two-stage cascade refrigeration system. Here a `matrix` structure is used for node placement but one could equally use positioning keys (*i.e.*, `above`, `below`, and so forth) to achieve similar placement. Source: Figure 9.3 on pg. 301 in Smith, J.M.; Van Ness, H.C.; Abbot, M.M. *Introduction to Chemical Engineering Thermodynamics*, 6th ed.; McGraw-Hill, 2001.

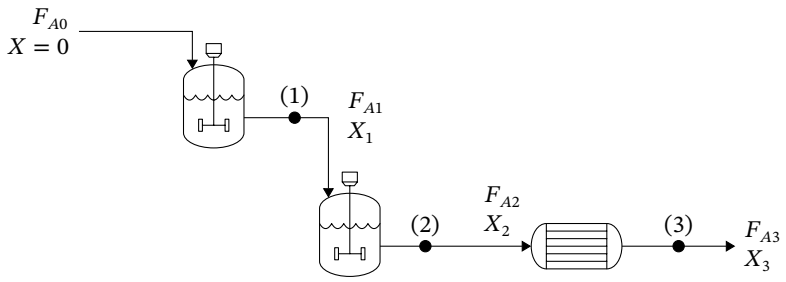



```latex
\begin{tikzpicture}[font=\footnotesize]
  % \usetikzlibrary{matrix}
  % Units
  \matrix [column sep=1.5cm, row sep=0.75 cm] {
    % Row 1
    & \node[basic hx, rotate=90] (cond) {};
    & \\
    % Row 2
    \node[valve, rotate=90] (v2) {}; &
    \node{\normalsize Cycle 2}; &
    \node[turbine] (c2) {}; \\
    % Row 3
    & \node[shell and tube hx] (intx) {};
    & \\
    % Row 4
    \node[valve, rotate=90] (v1) {}; &
    \node{\normalsize Cycle 1}; &
    \node[turbine] (c1) {}; \\
    % Row 5
    & \node[basic hx, rotate=90] (evap) {}; & \\
  };

  % Streams
  \tikzset{>=Stealth}
  % Upper loop
  \draw[->] (cond.n) -| (v2.e);
  \draw[->] (v2.w) |- (intx.wnw);
  \draw[->] (intx.ene) -| (c2.sse);
  \draw[->] (c2.nnw) |- (cond.s);
  % Lower loop
  \draw[->] (intx.wsw) -| (v1.e);
  \draw[->] (v1.w) |- (evap.n);
  \draw[->] (evap.s) -| (c1.sse);
  \draw[->] (c1.nnw) |- (intx.ese);

  % Labels
  \node[above] at (intx.north) {Interchanger};
```

```
  \node[below] at (cond.w) {Condenser at $T_H$};
  \node[above] at (evap.e) {Evaporator at $T_C$};
  \node[above left=1pt and 7mm of intx.wnw] {$T^{\prime}_C$};
  \node[below left=1pt and 7mm of intx.wsw] {$T^{\prime}_H$};

  \draw[<-] (c2.e) -- node[pos=2] {$W_s(2)$} ++(5mm,0);
  \draw[<-] (c1.e) -- node[pos=2] {$W_s(1)$} ++(5mm,0);
  \draw[->] (cond.e) -- node[pos=1.5] {$\lvert Q_H \rvert$} ++(0,5mm);
  \draw[<-] (evap.w) -- node[pos=1.5] {$\lvert Q_C \rvert$} ++(0,-5mm);
\end{tikzpicture}
```

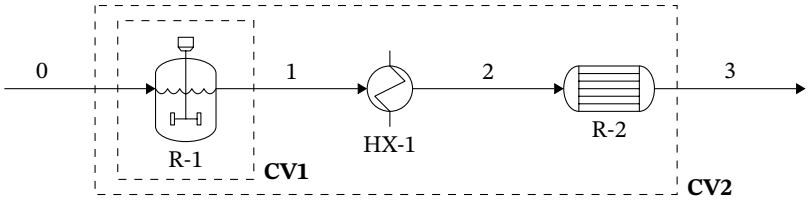
7.3 Reaction engineering

EXAMPLE 5: A series of two CSTRs followed by a PFR as part of a discussion regarding Levenspiel plots. Source: Figure 2-10 on pg. 61 in Fogler, H.S. *Elements of Chemical Reaction Engineering*, 4th ed.; Pearson, 2006.




```
\begin{tikzpicture}[font=\footnotesize]
  % Custom point marker
  \tikzstyle{dot} = [circle, draw, fill=black, inner sep=1.5pt]

  % Units
  \node[tank reactor, unit int=stirred liquid] (R1) {};
  \node[tank reactor, unit int=stirred liquid, below right=of R1] (R2) {};
  \node[tube reactor, unit int=tubular, right=2cm of R2.ese] (R3) {};

  % Streams
  \draw[<-] (R1.nnw) |- coordinate[at end] (p0) ++(-1.5,0.5);
  \draw[->] (R1.ese) -| node[dot, pos=0.3] (p1) {} (R2.nnw);
  \draw[->] (R2.ese) -- node[dot, pos=0.3] (p2) {} (R3.w);
  \draw[->] (R3.e) -- node[dot] (p3) {} coordinate[at end] (p4) ++(1.5,0);

  % Labels
  \node[left, align=right] at (p0) {$F_{A0}$\\$X=0$};
  \node[above] at (p1) {(1)};
  \node[right=5mm of p1, align=left] {$F_{A1}$\\$X_1$};
  \node[above] at (p2) {(2)};
  \node[above left, align=left] at (R3.w) {$F_{A2}$\\$X_2$};
  \node[above] at (p3) {(3)};
  \node[right, align=left] at (p4) {$F_{A3}$\\$X_3$};
\end{tikzpicture}
```

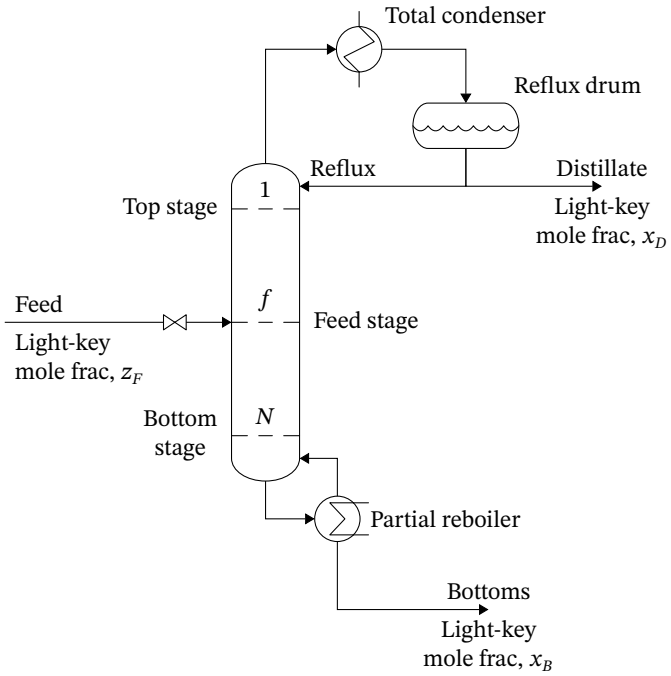
EXAMPLE 6: Two adiabatic reactors separated by a heat exchanger as part of a discussion regarding interstage heating and cooling. Source: Figure 11.2 on pg. 300 in Drews, A. *An Introduction to Chemical Reaction Engineering with MATLAB*; ceng113.eng.ucsd.edu (2019).




```
\begin{tikzpicture}[font=\footnotesize]
  % Units
  \node[tank reactor, unit int=stirred liquid] (R1) {};
  \node[basic hx, unit ext=util, right=2cm of R1] (HX1) {};
  \node[tube reactor, unit int=tubular, right=2cm of HX1] (R2) {};

  % Streams
  \draw[<-] (R1.w) -- node[above, near end] {0} ++(-2,0);
  \draw[->] (R1.e) -- node[above] {1} (HX1.w);
  \draw[->] (HX1.e) -- node[above] {2} (R2.w);
  \draw[->] (R2.e) -- node[above] {3} ++(2,0);

  % Control volumes
  \coordinate[above left=7mm and 5mm of R1.nw] (a);
  \coordinate[below right=7mm and 5mm of R1.se] (b);
  \coordinate[above left=9mm and 8mm of R1.nw] (c);
  \coordinate[below right=11mm and 5mm of R2.se] (d);
  \draw[dashed] (a) rectangle (b);
  \draw[dashed] (c) rectangle (d);

  % Labels
  \node[below] at (R1.s) {R-1};
  \node[below] at (HX1.south util) {HX-1};
  \node[below] at (R2.s) {R-2};
  \node[anchor=base west] at (b) {\textbf{CV1}};
  \node[anchor=base west] at (d) {\textbf{CV2}};
\end{tikzpicture}
```

7.4 Separations

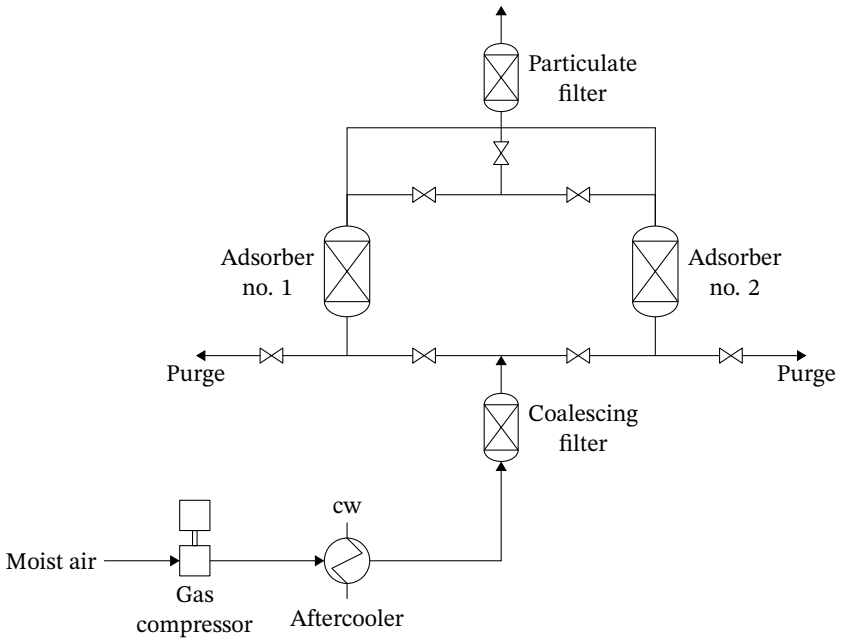
EXAMPLE 7: A general distillation system using a total condenser and partial reboiler. Some annotations have been excluded for clarity (clarity of this document, not the figure: if so desired it would only be a matter of additional `node` elements to replace the excluded annotations). Source: Figure 7.2 on pg. 261 in Seader, J.D.; Henley, E.J.; Roper, D.K. *Separation Process Principles*, 3rd ed.; John Wiley and Sons, 2011.




```
\begin{tikzpicture}[font=\footnotesize]
  %\usetikzlibrary{decorations.pathmorphing}
  % Units
  \node[column, unit int=numbered tray, scale=1.5] (c) {};
  \node[vessel, rotate=90, unit int=liquid rotated, above right=8mm and 15mm of c, anchor=
      north] (drum) {};
  \node[basic hx, unit ext=util, above left=8mm and 5mm of drum.north] (cond) {};
  \node[basic hx, unit int=U tube, unit ext=util, rotate=90, below right=5mm and 5mm of c,
      anchor=east] (reboil) {};

  % Helpful points
  \coordinate[left=3cm of c.west] (F);
  \coordinate[right=4cm of c.north east] (D);
  \coordinate[below right=2cm and 2.5cm of c.south east] (B);

  % Streams
  \draw[->] (c.north) |- (cond.west);
  \draw[->] (cond.east) -| (drum.east);
  \draw[->] (drum.west) |- (c.north east);
  \draw[->] (drum.west) |- (D);
  \draw[->] (c.south) |- (reboil.north);
  \draw[->] (reboil.east) |- (c.south east);
  \draw[->] (reboil.west) |- (B);
  \draw[->] (F) -- node[valve, near end] {} (c.west);

  % Distillate labels
  \node[above=0.5pt of c.top tray] {1};
  \node[left=5mm of c.top tray, anchor=east] {Top stage};
  \node[anchor=south west] at (c.ne) {Reflux};
  \node[above] at (D) {Distillate};
  \node[above right] at (cond.ne) {Total condenser};
  \node[above right] at (drum.south east) {Reflux drum};

  % Feed labels
```

```
  \node[above=0.5pt of c.feed tray] {$f$};
  \node[anchor=south west] at (F) {Feed};
  \node[right=5mm of c.feed tray, anchor=west] {Feed stage};
  \node[align=left, anchor=north west] at (F) {Light-key\\mole frac, $z_F$};

  % Bottoms labels
  \node[above=0.5pt of c.bottom tray] {$N$};
  \node[above] at (B) {Bottoms};
  \node[right] at (reboil.south) {Partial reboiler};
  \node[left=5mm of c.bottom tray, anchor=east, align=center] {Bottom\\stage};
  \node[below, align=center] at (D) {Light-key\\mole frac, $x_D$};
  \node[below, align=center] at (B) {Light-key\\mole frac, $x_B$};
\end{tikzpicture}
```

EXAMPLE 8: A schematic of pressure-swing adsorption for the dehydration of air. In this diagram the original authors used approximately the same symbol for a filter as the adsorber, a similarity retained here. A matrix was again used to help with alignment and several `tikzstyle` definitions were used to create "shortcut" shapes for the `filter`, `ads`, and `pump` units. Source: Figure 15.2 on pg. 570 in Seader, J.D.; Henley, E.J.; Roper, D.K. *Separation Process Principles*, 3rd ed.; John Wiley and Sons, 2011.

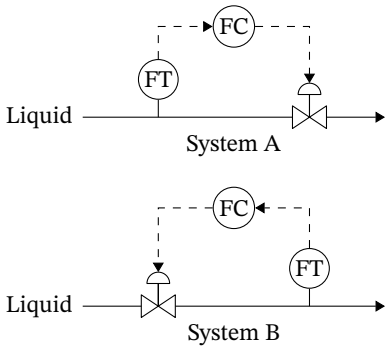



```
\begin{tikzpicture}[font=\footnotesize]
  % \usetikzlibrary{matrix}

  % Definitions
  \tikzstyle{filter} = [tube reactor, unit int=packed, rotate=90, scale=0.75];
  \tikzstyle{ads} = [tube reactor, unit int=packed, rotate=90];
  \tikzstyle{pump} = [reciprocating pump, yscale=-1];

  % Nodes
  \matrix [column sep=1.5cm, row sep=0.5 cm] {
    % 4 columns: & & & \\
    & & \node[filter] (f1) {}; & \\
```

```
    & & & \\
    & & & \\
    & \node[ads] (ads1) {}; & & \node[ads] (ads2) {};\\
    & & & \\
    & & \node[filter] (f2) {}; & \\ % f2
    \node[pump] (p) {}; & \node[basic hx, unit ext=util] (hx) {};& & \\ % pump+cooler
  };

  % Helper points
  \coordinate[above=5mm of f2.e] (v2); % for valves near f2
  \coordinate[below=11mm of f1.w] (v1); % for valves near f1

  % Streams
  \draw[<-] (p.w) -- node[pos=1.7] {Moist air} ++(-1,0);
  \draw[->] (p.e) -- (hx.w);
  \draw[->] (hx.e) -| (f2.w);
  \draw[->] (f2.e) -- (v2);
  \draw (v2) -| node[valve, near start] {} (ads1.w);
  \draw[->] (ads1.w |- v2) -- node[valve] {} node[anchor=north, at end] {Purge} ++(-2,0);
  \draw (v2) -| node[valve, near start] {} (ads2.w);
  \draw[->] (ads2.w |- v2) -- node[valve] {} node[anchor=north, at end] {Purge} ++(2,0);
  \draw (ads1.e) |- node[valve, near end] {} (v1) -| node[valve, near start] {} (ads2.e);
  \draw (v1) -- node[valve, rotate=90] {} (f1.w);
  \draw (ads1.e) -- ++(0,1.3) -| (ads2.e);
  \draw[->] (f1.e) -- ++(0,5mm);

  % Labels
  \node[right, align=center] at (f1.s) {Particulate\\filter};
  \node[right, align=center] at (ads2.s) {Adsorber\\no. 2};
  \node[left, align=center] at (ads1.n) {Adsorber\\no. 1};
  \node[right, align=center] at (f2.s) {Coalescing\\filter};
  \node[below, align=center] at (p.n) {Gas\\compressor};
  \node[below] at (hx.south util) {Aftercooler};
  \node[above] at (hx.north util) {cw};

\end{tikzpicture}
```

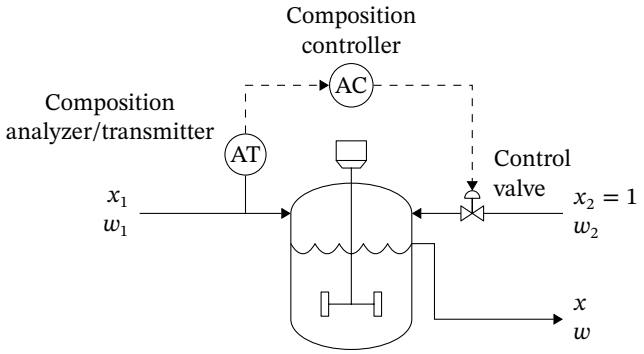
7.5 Process controls

EXAMPLE 9: A comparison of feedforward and feedback control systems. This example illustrates the use of absolute positioning (only for variety, not because it's necessary). Source: Figure E1.9 on pg. 13 in Seborg, D.E.; Edgar, T.F.; Mellichamp, D.A.; Doyle III, F.J. *Process Dynamics and Control*, 4th ed.; John Wiley and Sons, 2016.



```
\begin{tikzpicture}[font=\footnotesize]
  % Feedforward system
  \draw[->] (0,0) -- ++(4,0);
  \node[elec] at (1, 0.5) (ft1) {FT};
  \node[elec] at (2, 1.2) (fc1) {FC};
  \node[valve, unit ext=actuator, scale=1.5] at (3,0) (v1) {};
  \node[left] at (0,0) {Liquid};
  \node[below] at (2,-0.1) {System A};
  \draw (1,0) -- (ft1.south);
  \draw[->, dashed] (ft1.north) |- (fc1.west);
  \draw[->, dashed] (fc1.east) -| (v1.actuator);

  % Feedback system
  \draw[->] (0,-2.5) -- ++(4,0);
  \node[valve, unit ext=actuator, scale=1.5] at (1,-2.5) (v2) {};
  \node[elec] at (2, -1.2) (fc2) {FC};
  \node[elec] at (3, -2) (ft2) {FT};

  \node[left] at (0,-2.5) {Liquid};
  \node[below] at (2,-2.6) {System B};
  \draw (3,-2.5) -- (ft2.south);
  \draw[->, dashed] (ft2.north) |- (fc2.east);
  \draw[->, dashed] (fc2.west) -| (v2.actuator);
\end{tikzpicture}
```

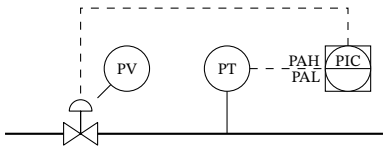
EXAMPLE 10: A blending system controlled by measuring the composition of the stream 1 and adjusting a control valve on stream two. Source: Figure 1.5 on pg. 5 in Seborg, D.E.; Edgar, T.F.; Mellichamp, D.A.; Doyle III, F.J. *Process Dynamics and Control*, 4th ed.; John Wiley and Sons, 2016.




```
\begin{tikzpicture}[font=\footnotesize]
  % Units
  \node[tank reactor, unit int=stirred liquid, scale=2] (r) {};
  \node[elec, above=of r] (ac) {AC};

  % Helpful points
  \coordinate[left=2cm of r.nw] (f1);
  \coordinate[right=2cm of r.ne] (f2);
  \coordinate[right=2cm of r.se] (p);

  % Streams and stream labels
  \draw[->] (f1) -- coordinate[pos=0.7] (s) (r.nw);
  \draw[->] (f2) -- node[valve, unit ext=actuator, pos=0.6] (v) {} (r.ne);
  \draw[->] (r.e) -- ++(3mm,0) |- (p);
  \node[left, align=right] at (f1) {$x_1$\\$w_1$};
```

```
  \node[right, align=left] at (f2) {$x_2=1$\\$w_2$};
  \node[right, align=left] at (p) {$x$\\$w$};

  % Control elements
  \node[elec, above=5mm of s] (at) {AT};
  \draw (s) -- (at.south);
  \draw[->, dashed] (at.north) |- (ac.west);
  \draw[->, dashed] (ac.east) -| (v.actuator);

  % Labels
  \node[align=center, anchor=south east] at (at.west) {Composition\\analyzer/transmitter};
  \node[align=center, anchor=south west] at (v.ne) {Control\\valve};
  \node[align=center, above] at (ac.north) {Composition\\controller};
\end{tikzpicture}
```

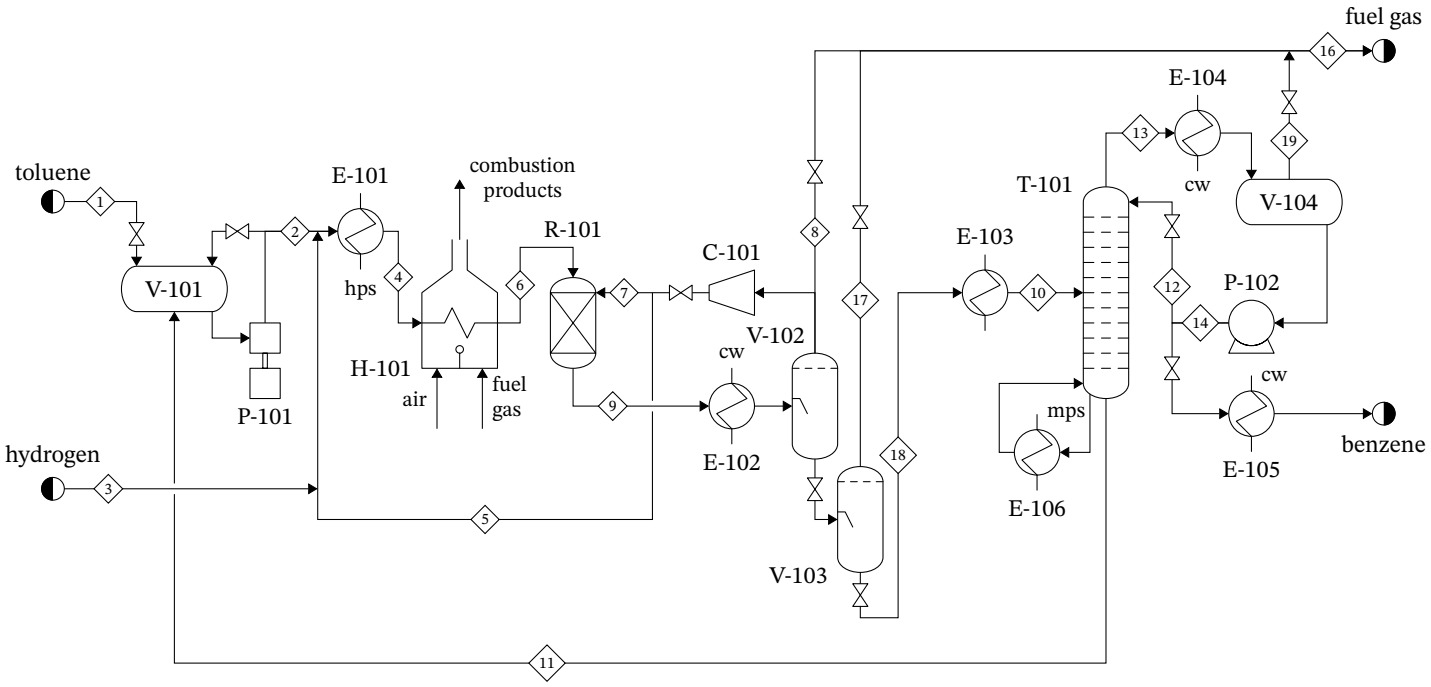
EXAMPLE 11: A typical control loop with transmitter, controller, alarms, and control element. Source: Figure 5.8 on pg. 221 in Sinnott, T.; Towler, G. *Chemical Engineering Design*, 6th ed.; Elsevier, 2020.




```
\begin{tikzpicture}
  % Process stream
  \draw[thick] (0,0) -- (5cm, 0cm);

  % Controller elements
  \node [valve, unit ext=actuator, scale=1.5] (v) at (1cm, 0cm) {};
  \node [ctrl, ctrl sense=P, ctrl type=V, above right=2mm and 4mm of v.actuator] (pv) {};
  \node [ctrl, ctrl sense=P, ctrl type=T, above right=5mm and 15mm of v] (pt) {};
  \node [ctrl,
    ctrl display=shared full,
    ctrl sense=P,
    ctrl type=IC,
    ctrl alarm=HL,
    ctrl alarm side=left,
    right=of pt
  ] (pic) {};

  % Signal lines
  \draw (pt.s) -- (pt.s |- v.w);
  \draw[dashed] (pt.e) -- (pic.w);
  \draw[dashed] (pic.n) -- ++(0mm,5mm) -| (v.actuator);
  \draw (pv.sw) -- ++(215:2mm);
\end{tikzpicture}
```

7.6 Capstone processes

EXAMPLE 12: Hydrodealkylation of toluene to produce benzene. Source: Figure 1.3 in Turton, R.; Bailie, R.C.; Whiting, W.B.; Shaeiwitz, J.A. *Analysis, Synthesis, and Design of Chemical Processes*, 3rd ed.; Pearson, 2009.



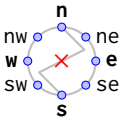
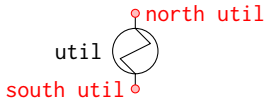
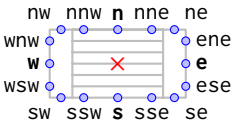
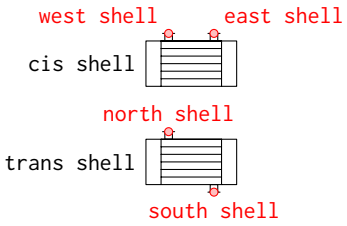
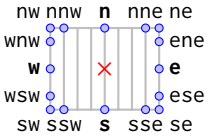
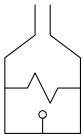
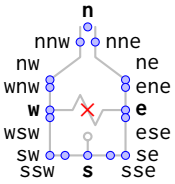
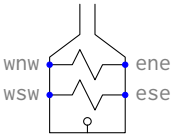
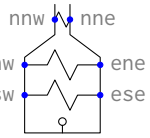
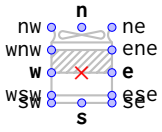

```
\begin{tikzpicture}[font=\footnotesize]
  % Reduce stream label sizes
  \tikzstyle{sid} = [diamond, draw, solid, fill=white, text badly centered, inner sep=1pt, font
      =\tiny]

  % Units
  \node[vessel, rotate=90] (V101) {};
  \node[reciprocating pump, below right=0.75cm and 0.5cm of V101] (P101) {};
  \node[basic hx, unit ext=util, above right=0.25cm and 2.75cm of V101] (E101) {};
  \node[fired hx, below right=1cm and 0.6cm of E101, anchor=west] (H101) {};
  \node[tube reactor, unit int=packed, right=of H101, anchor=center, rotate=90] (R101) {};
  \node[turbine, right=1.5cm of R101.se] (C101) {};
  \node[basic hx, unit ext=util, below=of C101] (E102) {};
  \node[vessel, unit int=phase sep, right=0.5cm of E102] (V102) {};
  \node[vessel, unit int=phase sep, below=of V102.se, anchor=west] (V103) {};
  \node[basic hx, unit ext=util, right=2.75cm of C101] (E103) {};
  \node[column, unit int=tray, right=1cm of E103] (T101) {};
  \node[basic hx, unit ext=util, below left=0.5cm and 0.7cm of T101.s] (E106) {};
  \node[basic hx, unit ext=util, above right=0.5cm and 1cm of T101.n] (E104) {};
  \node[vessel, rotate=90, below right=1cm and 0.5cm of E104] (V104) {};
  \node[centrifugal pump, below=1cm of V104.nw] (P102) {};
  \node[basic hx, unit ext=util, below=0.5cm of P102] (E105) {};
  \node[product, right=1.3cm of E105] (bz) {};
  \node[product, above=4.5cm of bz] (fuel) {};
  \node[feed, above left=0.75cm and 1cm of V101.ne] (tol) {};
  \node[feed, below=3.5cm of tol] (h2) {};

```

```
  % Streams
  \draw[->] (tol.e) -| node[sid, near start]{1} node[valve, near end, rotate=90]{} (V101.ne);
  \draw[->] (V101.sw) |- (P101.w);
  \draw[->] (P101.n) |- (E101.w);
  \draw[<-] (V101.se) |- node[valve, pos=0.6]{} node[sid, pos=0.83]{2} (E101.w) node[pos=0.92](
      n1){} ;
  \draw[->] (E101.e) -- ++(0.2,0) |- node[sid, near start]{4} (H101.w);
  \draw[->] (H101.e) -| node[sid,near end]{6} ++(0.3,1) -| (R101.e);
  \draw[->] (E102.e) -- (V102.w);
  \draw[->] (V102.s) |- node[valve, rotate=90, near start]{} (V103.w);
  \draw[->] (V103.s) |- node[valve, rotate=90, near start]{} ++(0.5,-0.6) |- node[sid, near
      start]{18} (E103.w);
  \draw[->] (E103.e) -- node[sid, pos=0.4]{10} (T101.w);
  \draw[->] (T101.ssw) |- (E106.e);
  \draw[->] (E106.w) -- ++(-0.2,0) |- (T101.sw);
  \draw[->] (T101.n) |- node[sid,near end]{13} (E104.w);
  \draw[->] (E104.e) -| (V104.ne);
  \draw[->] (V104.sw) |- (P102.e);
  \draw[->] (P102.w) -- ++(-0.75,0) |- node[sid,pos=0.15]{12} node[valve,rotate=90,pos=0.4]{} (
      T101.ne);
  \draw[->] (P102.w) -- node[sid]{14} ++(-0.75,0) |- node[valve,rotate=90,near start]{} (E105.w
      );
  \draw[->] (E105.e) -- (bz.w);
  \draw[->] (V103.n) |- node[sid, pos=0.2]{17} node[valve,rotate=90, pos=0.3]{} (fuel.w);
  \draw[->] (V102.n) |- node[sid, pos=0.2]{8} node[valve,rotate=90, pos=0.3]{} node[sid, pos
      =0.96]{16} (fuel.w);
  \draw[->] (V104.e) -- node[sid,pos=0.3]{19} node[valve,rotate=90,pos=0.6]{} (V104 |- fuel);
  \draw[->] (V102.n) |- (C101.e);
  \draw[->] (C101.w) -- node[valve, near start]{} node[sid,near end]{7} (R101.se) node[pos
      =0.5](n2){};

  % Cross stream 9 over 5
  \path[spath/save=over1] (R101.w) |- (E102.w);
  \path[spath/save=under1] (n2 |- C101) |- (n1 |- V103) -| (n1 |- E101);
  \tikzset{bridge={over1}{under1}}
  \draw[->, spath/use=over1] node[sid, pos=0.5]{9};
  \draw[->, spath/use=under1] node[sid, pos=0.5]{5};

  % Cross stream 3 over 11
  \path[spath/save=over2] (h2.e) -- (n1 |- h2);
  \path[spath/save=under2] (T101.s) -- ++(0,-3.5) -| (V101.w);
  \tikzset{bridge={over2}{under2}}
  \draw[->, spath/use=over2] node[sid, pos=0.2]{3};
  \draw[->, spath/use=under2] node[sid, pos=0.4]{11};

  % Unit labels
  \node at (V101.center) {V-101};
  \node[below] at (P101.s) {P-101};
  \node[above] at (E101.north util) {E-101};
  \node[left] at (H101.sw) {H-101};
  \node[above=4mm of R101.e] {R-101};
  \node[above=1mm of C101.n] {C-101};
  \node[below] at (E102.south util) {E-102};
  \node[left, anchor=south east] at (V102.n) {V-102};
  \node[below left, anchor=north east] at (V103.sw) {V-103};
  \node[above] at (E103.north util) {E-103};
  \node[above left] at (T101.nw) {T-101};
  \node[above] at (E104.north util) {E-104};
  \node at (V104.center) {V-104};
```

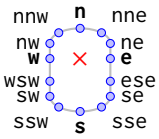
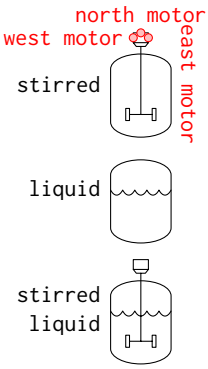
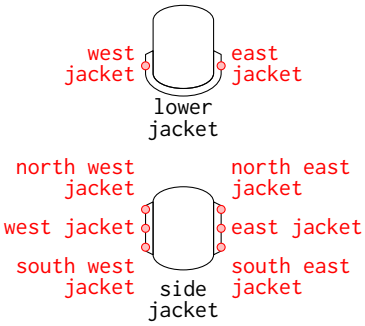
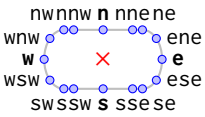
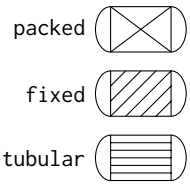
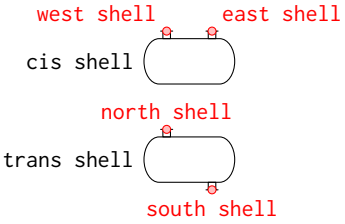
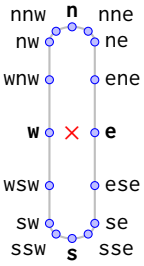
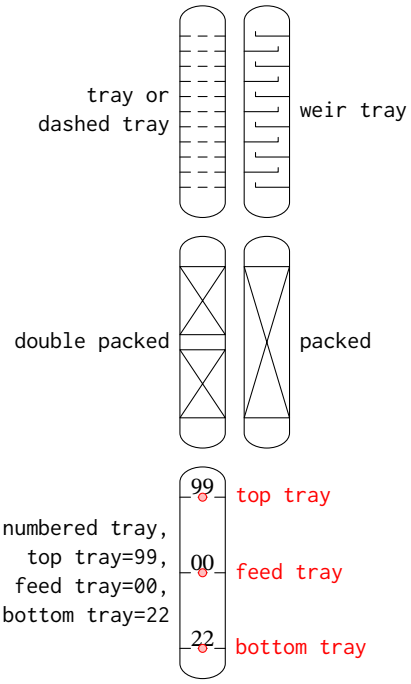
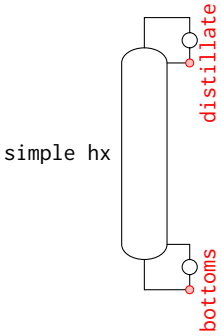
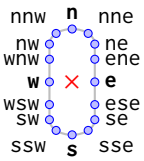
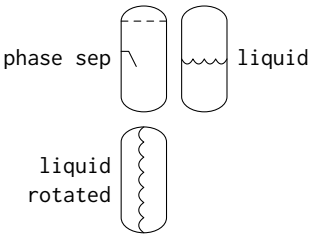
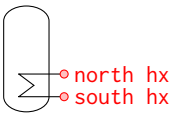
```
  \node[above] at (P102.n) {P-102};
  \node[below] at (E105.south util) {E-105};
  \node[below] at (E106.south util) {E-106};

  % Stream labels
  \node[above] at (tol.n) {toluene};
  \node[above] at (h2.n) {hydrogen};
  \node[below] at (bz.s) {benzene};
  \node[above] at (fuel.n) {fuel gas};

  % Utility labels
  \tikzset{font=\scriptsize}
  \node[right] at (E105.north util) {cw};
  \node[below] at (E104.south util) {cw};
  \node[above] at (E102.north util) {cw};
  \node[right] at (E106.north util) {mps};
  \node[below] at (E101.south util) {hps};

  \draw[->] (H101.n) -- node[at end, right, align=center] {combustion\\products} ++(0,0.8);
  \draw[<-] (H101.sse) -- node[right, align=left] {fuel\\gas} ++(0,-0.8);
  \draw[<-] (H101.ssw) -- node[left] {air} ++(0,-0.8);
\end{tikzpicture}
```

8 Quick reference

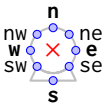
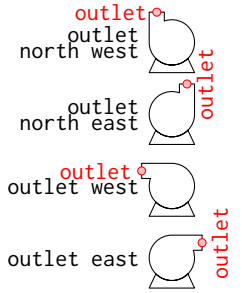
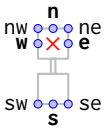
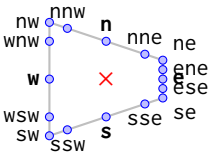
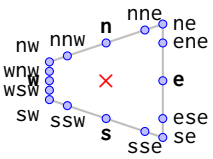
Heat exchangers

shape name	default	anchors	unit int	unit ext
basic hx				
			U tube 	
shell and tube hx			two pass  four pass 	
plate hx				
fired hx			double  triple 	
cooling tower				forced  natural 

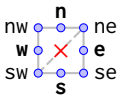
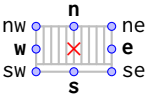
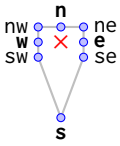
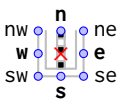
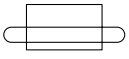
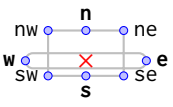
Vessels

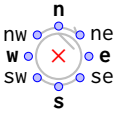
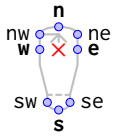
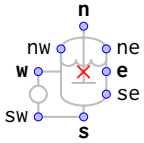
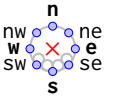
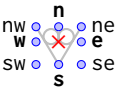
shape name	default	anchors	unit int	unit ext
tank reactor				
tube reactor				
column				
vessel				

Fluid transport

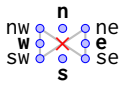
shape name	default	anchors	unit int	unit ext
centrifugal pump				
reciprocating pump				
compressor				
turbine				

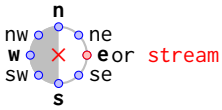
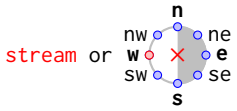
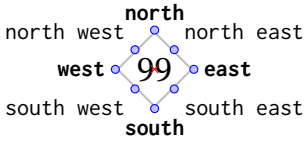
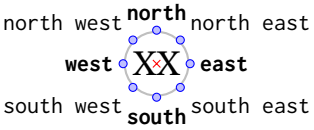
Solids processing

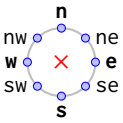
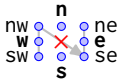
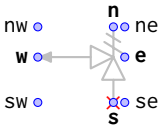
shape name	default	anchors
filter		
press filter		
cyclone		
centrifuge		
belt dryer		

shape name	default	anchors
rotary dryer		
granulator or agglomerator		
crystallizer		
ball mill		
roll crusher		

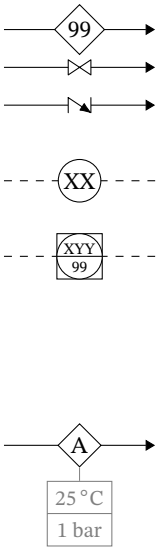
Stream annotations

shape name	default	anchors	unit int	unit ext
valve	⋈			actuator 

shape name	default	anchors
feed	◐	
product	◑	
sid		
elec		

shape name	default	anchors
ctrl		
nrv		
prv		

```
\begin{tikzpicture}
  \draw[->] (0,0) -- node[sid] {99} (2,0);
  \draw[->] (0,-0.5) -- node[valve] {} (2,-0.5);
  \draw[->] (0,-1) -- node[nrv] {} (2,-1);


  \draw[dashed] (0,-2) -- node[elec] {XX} (2,-2);


  \draw[dashed] (0,-3.5) --
    node[ctrl, solid,
     ctrl sense=X,
     ctrl type=YY,
     ctrl display=shared full,
     ctrl numID=99] {} (2,-3.5);

  % \usepackage{siunitx} for \qty{}{}
  \draw[->] (0,-5.5) -- node[sid,
    pfdpin2={below:\qty{25}{\celsius}/1 bar},
    ] {A} (2,-5.5);

\end{tikzpicture}
```




```
\begin{tikzpicture}
  \tikzset{bridge radius=8pt} % default 0pt=straight line
  \path[->, spath/save=over] (0,0) -- (2,0);
  \path[->, spath/save=under] (1,-1) -- (1,1);
  \tikzset{bridge={over}{under}}
  \draw[->, spath/use=over] node[sid, pos=0.1]{1};
  \draw[->, spath/use=under] node[sid, pos=0.2]{2};
\end{tikzpicture}
```