

CircuiTikZ
version git:2e95431 (2019/04/14)

Massimo A. Redaelli (m.redaelli@gmail.com)
Stefan Lindner (stefan.lindner@fau.de)
Stefan Erhardt (stefan.erhardt@fau.de)
Romano Giannetti (romano.giannetti@gmail.com)

April 14, 2019

Contents

1	Introduction	2
1.1	About	2
1.2	Loading the package	2
1.3	Installing a new version of the package.	2
1.4	Requirements	2
1.5	Incompatible packages	3
1.6	License	3
1.7	Feedback	3
1.8	Incompabilities between version	3
1.9	Package options	4
2	Tutorials	6
2.1	Getting started with CircuiTikZ: a current shunt	6
2.2	A more complex tutorial: circuits Romano’s style.	8
3	The components	12
3.1	Path-style components	13
3.1.1	Anchors	13
3.1.2	Customization	13
3.1.2.1	Components size	14
3.1.2.2	Thickness of the lines	14
3.1.2.3	Shape of the components	15
3.1.3	Descriptions	15
3.2	Node-style components	15
3.2.1	Mirroring and flipping	16
3.2.2	Anchors	16
3.2.3	Descriptions	16
3.3	Grounds and supply voltages	17
3.3.1	Grounds anchors	18
3.4	Instruments	18
3.5	Resistive bipoles	19
3.5.1	Generic sensors anchors	21
3.6	Diodes and such	21
3.7	Tripole-like diodes	23
3.7.1	Triacs anchors	23
3.8	Basic dynamical bipoles	24
3.9	Stationary sources	25
3.10	Sinusoidal sources	26
3.11	Controlled sources	26
3.12	Noise sources	27

3.13	Special sources	28
3.14	DC sources	29
3.15	Mechanical Analogy	29
3.16	Other bipoles	29
3.17	Block diagram components	30
3.17.1	Blocks anchors	33
3.17.2	Blocks customization	34
3.17.2.1	Multi ports	34
3.17.2.2	Labels and custom twoport boxes	34
3.17.2.3	Box option	34
3.17.2.4	Dash optional parts	34
3.18	Transistors	35
3.18.1	Transistors anchors	38
3.18.2	Transistor paths	40
3.19	Electronic Tubes	40
3.20	RF components	41
3.20.1	Microstrip customization	42
3.21	Electro-Mechanical Devices	43
3.22	Double bipoles	44
3.22.1	Double dipoles anchors	45
3.23	Amplifiers	46
3.23.1	Amplifiers anchors	47
3.23.2	Amplifiers customization	49
3.24	Support shapes and bipoles	49
3.24.1	Terminal shapes	50
3.24.2	Crossings	50
3.24.3	Arrows size	51
3.25	Switches and buttons	51
3.25.1	Traditional switches	51
3.25.2	Cute switches	52
3.25.3	Switches anchors	53
3.26	Logic gates	53
3.26.1	American Logic gates	54
3.26.2	European Logic gates	54
3.26.3	Special components	55
3.26.4	Logic port customization	55
3.26.5	Logic port anchors	56
3.27	Chips	57
3.27.1	DIP and QFP chips customization	57
3.27.2	Chips anchors	58
3.27.3	Chips rotation	59
3.27.4	Chip special usage	59

4	Labels and similar annotations	60
4.1	Labels and Annotations	60
4.2	Currents and voltages	62
4.3	Currents	66
4.4	Flows	68
4.5	Voltages	68
4.5.1	European style	68
4.5.2	American style	70
4.5.3	Voltage position	70
4.5.4	Global properties of voltages and currents	71
4.6	Nodes	71
4.7	Special components	73
4.8	Integration with <code>siunitx</code>	75
4.9	Mirroring and Inverting	75
4.10	Putting them together	76
4.11	Line joins between Path Components	76
5	Colors	77
5.1	Shape colors	77
5.2	Fill colors	78
6	FAQ	79
7	Defining new components	80
7.1	Suggested setup	80
7.2	Path-style component	81
7.3	Node-style component	84
7.3.1	Finishing your work	84
8	Examples	84
9	Changelog	89
	Index of the components	95

1 Introduction

1.1 About

CircuiTikZ was initiated by Massimo Redaelli in 2007, who was working as a research assistant at the Polytechnic University of Milan, Italy, and needed a tool for creating exercises and exams. After he left University in 2010 the development of CircuiTikZ slowed down, since L^AT_EX is mainly established in the academic world. In 2015 Stefan Lindner and Stefan Erhardt, both working as research assistants at the University of Erlangen-Nürnberg, Germany, joined the team and now maintain the project together with the initial author. In 2018 Romano Giannetti, full professor of Electronics at Comillas Pontifical University of Madrid, joined the team.

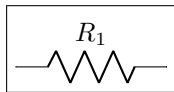
The use of CircuiTikZ is, of course, not limited to academic teaching. The package gets widely used by engineers for typesetting electronic circuits for articles and publications all over the world.

1.2 Loading the package

L ^A T _E X	ConT _E Xt ¹
<code>\usepackage{circuitikz}</code>	<code>\usemodule[circuitikz]</code>

TikZ will be automatically loaded.

CircuiTikZ commands are just TikZ commands, so a minimum usage example would be:



```
\tikz \draw (0,0) to[R=$R_1$] (2,0);
```

1.3 Installing a new version of the package.

The stable version of the package should come with your L^AT_EX distribution. Downloading the files from CTAN and installing them locally is, unfortunately, a distribution-dependent task and sometime not so trivial. If you search for `local texmf tree` and the name of your distribution on <https://tex.stackexchange.com/> you will find a lot of hints.

Anyway, the easiest way of using whichever version of CircuiTikZ is to point to the github page <https://circuitikz.github.io/circuitikz/> of the project, and download the version you want. You will download a simple (bigish) file, called `circuitikz.sty`.

Now you can just put this file in your local `texmf` tree, if you have one, or simply adding it into the same directory where your main file resides, and then use

```
\usepackage[...options...]{circuitikzgit}
```

instead of `circuitikz`. This is also advantageous for “future resilience”; the authors try hard not to break backward compatibility with new versions, but sometime things happen.

1.4 Requirements

- `tikz`, version ≥ 3 ;
- `xstring`, not older than 2009/03/13;
- `siunitx`, if using `siunitx` option.

¹ConT_EXt support was added mostly thanks to Mojca Miklavac and Aditya Mahajan.

1.5 Incompatible packages

TikZ's own `circuit` library, which is based on `CircuitikZ`, (re?)defines several styles used by this library. In order to have them work together you can use the `compatibility` package option, which basically prefixes the names of all `CircuitikZ` to[] styles with an asterisk.

So, if loaded with said option, one must write `(0,0) to[*R] (2,0)` and, for transistors on a path, `(0,0) to[*Tmos] (2,0)`, and so on (but `(0,0) node[nmos] {}`). See example at page 89.

1.6 License

Copyright © 2007–2019 Massimo Redaelli. This package is author-maintained. Permission is granted to copy, distribute and/or modify this software under the terms of the L^AT_EX Project Public License, version 1.3.1, or the GNU Public License. This software is provided ‘as is’, without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

1.7 Feedback

The easiest way to contact the authors is via the official Github repository: <https://github.com/circuitikz/circuitikz/issues>

1.8 Incompatibilities between version

Here, we will provide a list of incompatibilities between different version of `circuitikz`. We will try to hold this list short, but sometimes it is easier to break with old syntax than including a lot of switches and compatibility layers. You can check the used version at your local installation using the macro `\pgfcircversion{}`.

- After v0.8.4: the parameters `tripoles/american` or `port/aaa, ...bbb, ...ccc` and `...ddd` are no longer used and are silently ignored; the same stands for `nor`, `xor`, and `xnor` ports.
- After v0.8.4: voltage and current directions/sign (plus and minus signs in case of `american` `voltages` and arrows in case of `european` `voltages` have been rationalized with a couple of new options (see details in section 4.2. The default case is still the same as v0.8.4.
- Since v0.8.2: voltage and current label directions(`v<=` / `i<=`) do NOT change the orientation of the drawn source shape anymore. Use the “invert” option to rotate the shape of the source. Furthermore, from this version on, the current label(`i=`) at current sources can be used independent of the regular label(`l=`).
- Since v0.7?: The label behaviour at mirrored bipoles has changes, this fixes the voltage drawing, but perhaps you have to adjust your label positions.
- Since v0.5.1: The parts `pfet`, `pigfete`, `pigfetebulk` and `pigfetd` are now mirrored by default. Please adjust your `yscale`-option to correct this.
- Since v0.5: New voltage counting direction, here exists an option to use the old behaviour

For older projects, you can use an older version locally using the `git-version` and picking the correct commit from the repository (branch `gh-pages`).

1.9 Package options

Circuit people are very opinionated about their symbols. In order to meet the individual gusto you can set a bunch of package options. The standard options are what the authors like, for example you get this:



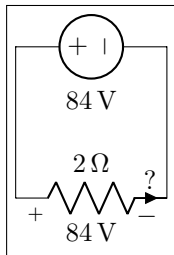
```

1 \begin{circuitikz}
2   \draw (0,0) to[R=2<\ohm>, i=?, v=84<\volt>] (2,0) --
3     (2,2) to[V<=84<\volt>] (0,2)
4     -- (0,0);
5 \end{circuitikz}

```

Feel free to load the package with your own cultural options:

LaTeX	ConTeXt
<code>\usepackage[american]{circuitikz}</code>	<code>\usemodule[circuitikz][american]</code>



```

1 \begin{circuitikz}
2   \draw (0,0) to[R=2<\ohm>, i=?, v=84<\volt>] (2,0) --
3     (2,2) to[V<=84<\volt>] (0,2)
4     -- (0,0);
5 \end{circuitikz}

```

Here is the list of all the options:

- **europenvoltages**: uses arrows to define voltages, and uses european-style voltage sources;
- **straightvoltages**: uses arrows to define voltages, and uses straight voltage arrows;
- **americanvoltages**: uses $-$ and $+$ to define voltages, and uses american-style voltage sources;
- **europencurrents**: uses european-style current sources;
- **americancurrents**: uses american-style current sources;
- **europenresistors**: uses rectangular empty shape for resistors, as per european standards;
- **americanresistors**: uses zig-zag shape for resistors, as per american standards;
- **europeninductors**: uses rectangular filled shape for inductors, as per european standards;
- **americaninductors**: uses "4-bumps" shape for inductors, as per american standards;
- **cuteinductors**: uses my personal favorite, "pig-tailed" shape for inductors;
- **americanports**: uses triangular logic ports, as per american standards;
- **europenports**: uses rectangular logic ports, as per european standards;
- **americangfsgearrester**: uses round gas filled surge arresters, as per american standards;

- `europengfsurgearrester`: uses rectangular gas filled surge arresters, as per european standards;
- `european`: equivalent to `europeancurrents`, `europeanvoltages`, `europeanresistors`, `europeaninductors`, `europeanports`, `europengfsurgearrester`;
- `american`: equivalent to `americancurrents`, `americanvoltages`, `americanresistors`, `americaninductors`, `americanports`, `americangfsurgearrester`;
- `siunitx`: integrates with `SIunitx` package. If labels, currents or voltages are of the form `#1<#2>` then what is shown is actually `\SI{#1}{#2}`;
- `nosunitx`: labels are not interpreted as above;
- `fulldiode`: the various diodes are drawn *and* filled by default, i.e. when using styles such as `diode`, `D`, `sD`, ... Other diode styles can always be forced with e.g. `Do`, `D-`, ...
- `strokediode`: the various diodes are drawn *and* stroke by default, i.e. when using styles such as `diode`, `D`, `sD`, ... Other diode styles can always be forced with e.g. `Do`, `D*`, ...
- `emptydiode`: the various diodes are drawn *but not* filled by default, i.e. when using styles such as `D`, `sD`, ... Other diode styles can always be forced with e.g. `Do`, `D-`, ...
- `arrowmos`: pmos and nmos have arrows analogous to those of pnp and npn transistors;
- `noarrowmos`: pmos and nmos do not have arrows analogous to those of pnp and npn transistors;
- `fetbodydiode`: draw the body diode of a FET;
- `nofetbodydiode`: do not draw the body diode of a FET;
- `fetsolderdot`: draw solderdot at bulk-source junction of some transistors;
- `nofetsolderdot`: do not draw solderdot at bulk-source junction of some transistors;
- `emptypmoscircle`: the circle at the gate of a pmos transistor gets not filled;
- `lazymos`: draws lazy nmos and pmos transistors. Chip designers with huge circuits prefer this notation;
- `straightlabels`: labels on bipoles are always printed straight up, i.e. with horizontal baseline;
- `rotatelabels`: labels on bipoles are always printed aligned along the bipole;
- `smartlabels`: labels on bipoles are rotated along the bipoles, unless the rotation is very close to multiples of 90°;
- `compatibility`: makes it possible to load `CircuitikZ` and `TikZ` circuit library together.
- Voltage directions: until v0.8.3, there was an error in the coherence between american and european voltages styles (see section 4.2 for the batteries. This has been fixed, but to guarantee backward compatibility and nasty surprises, the fix is available with new options:
 - `oldvoltage`direction: Use old way of voltage direction having a difference between european and american direction, with wrong default labelling for batteries;
 - `nooldvoltage`direction: The standard from 0.5 onward, utilize the (German?) standard of voltage arrows in the direction of electric fields (without fixing batteries);

- **RPvoltages** (meaning Rising Potential voltages): the arrow is in direction of rising potential, like in **oldvoltagedirections**, but batteries and current sources are fixed to follow the passive/active standard;
- **EFvoltages** (meaning Electric Field voltages): the arrow is in direction of the electric field, like in **nooldvoltagedirections**, but batteries are fixed;

If none of these option are given, the package will default to **nooldvoltagedirections**, but will give a warning. The behavior is also selectable circuit by circuit with the **voltage dir** style.

- **betterproportions**²: nicer proportions of transistors in comparison to resistors;

The old options in the singular (like **american voltage**) are still available for compatibility, but are discouraged.

Loading the package with no options is equivalent to the following options: `[nofetsolderdot, europeancurrents, europeanvoltages, americanports, americanresistors, cuteinductors, europeangfsurgearrester, nosiunitx, noarrowmos, smartlabels, nocompatibility]`.

In ConT_EXt the options are similarly specified: `current= european|american, voltage= european|american, resistor= american|european, inductor= cute|american|european, logic= american|european, siunitx= true|false, arrowmos= false|true`.

2 Tutorials

To draw a circuit, you have to load the **circuitikz** package; this can be done with

```
1 \usepackage[siunitx, RPvoltages]{circuitikz}
```

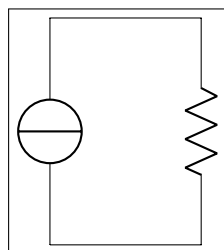
somewhere in your document preamble. It will load automatically the needed packages if not already done before.

2.1 Getting started with CircuiTikZ: a current shunt

Let's say we want to prepare a circuit to teach how a current shunt works; the idea is just draw a current generator, a couple of resistors in parallel, and the indication of currents and voltages for the discussion.

A circuit in CircuiTikZ is drawn into a **circuitikz** environment (which is really an alias for a **tikzpicture** one). In this first example we will use absolute coordinates. The electrical components can be divided in two big categories: the one that are bipoles and are placed along a path (also known as **to-style** component, for their usage) and components that are nodes and can have any number of poles, or connections.

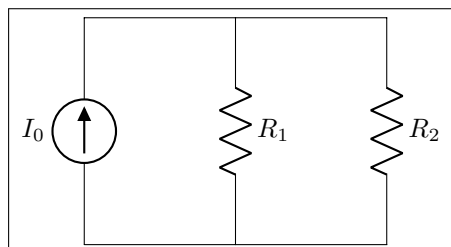
Let's start with the first type of components and build a basic mesh:



```
1 \begin{circuitikz}[]
2   \draw (0,0) to[isource] (0,3) -- (2,3)
3     to[R] (2,0) -- (0,0);
4 \end{circuitikz}
```

²May change in the future!

The symbol for the current source can surprise somebody; this is actually the european-style symbol, and the symbols chosen reflects the default options of loaded in the package (see section 1.9). Let's change the style for now (the author of the tutorial, Romano, is European but he has used since ever American style circuit, so...); and while at it, let add the other branch and some label.

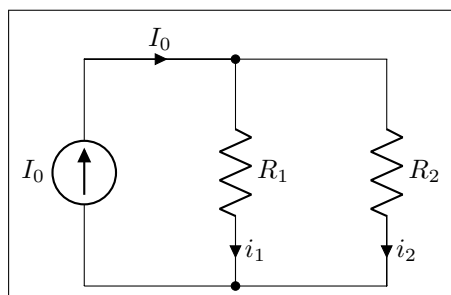


```

1 \begin{circuitikz}[american]
2   \draw (0,0) to[isource, l=$I_0$] (0,3) --
      (2,3)
3   to[R=$R_1$] (2,0) -- (0,0);
4   \draw (2,3) -- (4,3) to[R=$R_2$]
5       (4,0) -- (2,0);
6 \end{circuitikz}

```

You can use a single path or multiple path when drawing your circuit, it's just a question of style (but be aware that closing path could be non-trivial, see section 4.11), and you can use standard TikZ lines (--, |- or similar) for the wires. Nonetheless, sometime using the CircuiTikZ specific **short** component for the wires can be useful, because then we can add labels and nodes at it, like for example in the following circuit.

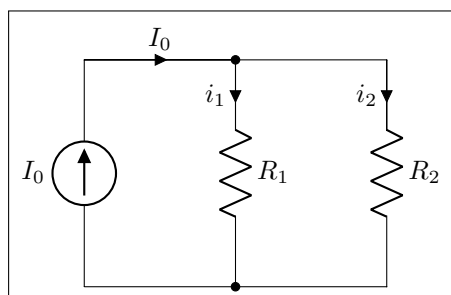


```

1 \begin{circuitikz}[american]
2   \draw (0,0) to[isource, l=$I_0$] (0,3)
3   to[short, -*, i=$I_0$] (2,3)
4   to[R=$R_1$, i=$i_1$] (2,0) -- (0,0);
5   \draw (2,3) -- (4,3)
6   to[R=$R_2$, i=$i_2$]
7   (4,0) to[short, -*] (2,0);
8 \end{circuitikz}

```

One of the problems with this circuit is that we would like to have the current in a different position, such as for example on the upper side of the resistors, so that the Kirchoff's Current Law at the node is better shown to students. No problem; as you can see in section 4.2 you can use the position specifier $\langle \rangle^{\sim}$ after the key **i**:

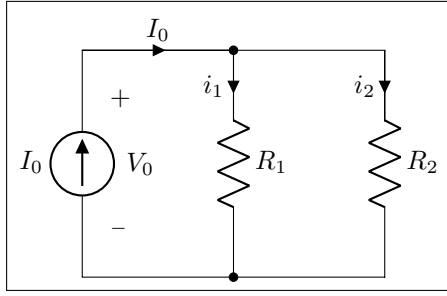


```

1 \begin{circuitikz}[american]
2   \draw (0,0) to[isource, l=$I_0$] (0,3)
3   to[short, -*, i=$I_0$] (2,3)
4   to[R=$R_1$, i>_=$i_1$] (2,0) -- (0,0);
5   \draw (2,3) -- (4,3)
6   to[R=$R_2$, i>_=$i_2$]
7   (4,0) to[short, -*] (2,0);
8 \end{circuitikz}

```

Finally, we would like to add voltages indication for carrying out the current formulas; as the default position of the voltage signs seems a bit cramped to me, I am adding the **voltage shift** parameter to make a bit more space for it...

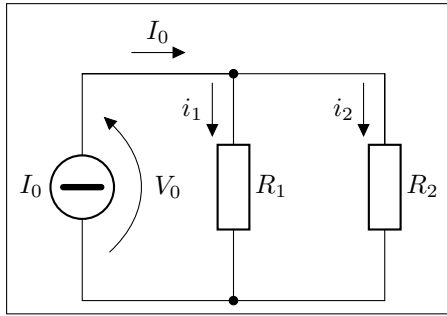


```

1 \begin{circuitikz}[american, voltage shift
    =0.5]
2   \draw (0,0) to[isource, l=$I_0$, v=$V_0$]
    (0,3)
3   to[short, -*, i=$I_0$] (2,3)
4   to[R=$R_1$, i>_=$i_1$] (2,0) -- (0,0);
5   \draw (2,3) -- (4,3)
6   to[R=$R_2$, i>_=$i_2$]
7   (4,0) to[short, -*] (2,0);
8 \end{circuitikz}

```

Et voilà! Remember that this is still \LaTeX , which means that you have done a description of your circuit, which is, in a lot of way, independent of the visualization of it. If you ever have to adapt the circuit to, say, a journal that force European style and flows instead of currents, you just change a couple of things and you have what seems a completely different diagram:

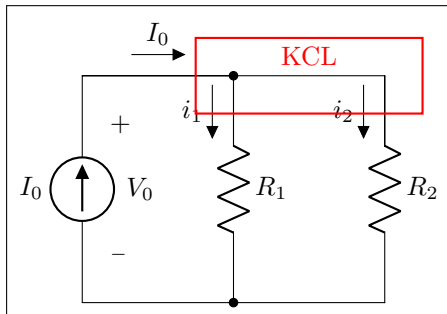


```

1 \begin{circuitikz}[european, voltage shift
    =0.5]
2   \draw (0,0) to[isourceC, l=$I_0$, v=$V_0$]
    (0,3)
3   to[short, -*, f=$I_0$] (2,3)
4   to[R=$R_1$, f>_=$i_1$] (2,0) -- (0,0);
5   \draw (2,3) -- (4,3)
6   to[R=$R_2$, f>_=$i_2$]
7   (4,0) to[short, -*] (2,0);
8 \end{circuitikz}

```

And finally, this is still TikZ, so that you can freely mix other graphics element to the circuit.



```

1 \begin{circuitikz}[american, voltage shift
    =0.5]
2   \draw (0,0) to[isource, l=$I_0$, v=$V_0$]
    (0,3)
3   to[short, -*, f=$I_0$] (2,3)
4   to[R=$R_1$, f>_=$i_1$] (2,0) -- (0,0);
5   \draw (2,3) -- (4,3)
6   to[R=$R_2$, f>_=$i_2$]
7   (4,0) to[short, -*] (2,0);
8   \draw[red, thick] (1.5,2.5) rectangle
    (4.5,3.5)
9   node[pos=0.5, above]{KCL};
10 \end{circuitikz}

```

2.2 A more complex tutorial: circuits Romano's style.

The idea is to draw a two-stage amplifier for a lesson, or exercise, on the different qualities of BJT and MOSFET transistors. Notice that this is a more “personal” tutorial, showing a way to draw circuits that is, in the author’s opinion, highly reusable and easy to do. The idea is using relative coordinates and named nodes as much as possible, so that changes in the circuit are easily done by changing keys numbers of position, and crucially, each block is reusable in other diagrams.

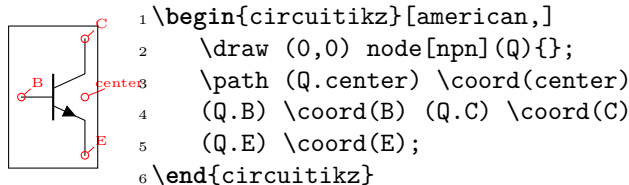
First of all, let’s define a handy function to show the position of nodes:

```

1 \def\coord(#1){coordinate(#1)}
2 \def\coord(#1){node[circle, red, draw, inner sep=1pt, pin={[red, overlay, inner
   sep=0.5pt, font=\tiny, pin distance=0.1cm, pin edge={red, overlay
   ,}]45:#1}}] (#1){}}

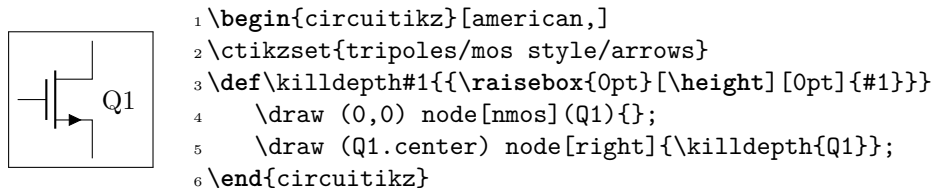
```

The idea is that you can use `\coord()` instead of `coordinate()` in paths, and that will draw sort of *markers* showing them. For example:



After the circuit is drawn, simply commenting out the second definition of `\coord` will hide all the markers.

So let's start with the first stage transistor; given that my preferred way of drawing a MOSFET is with arrows, I'll start issuing the command `\ctikzset{tripoles/mos style/arrows}`:



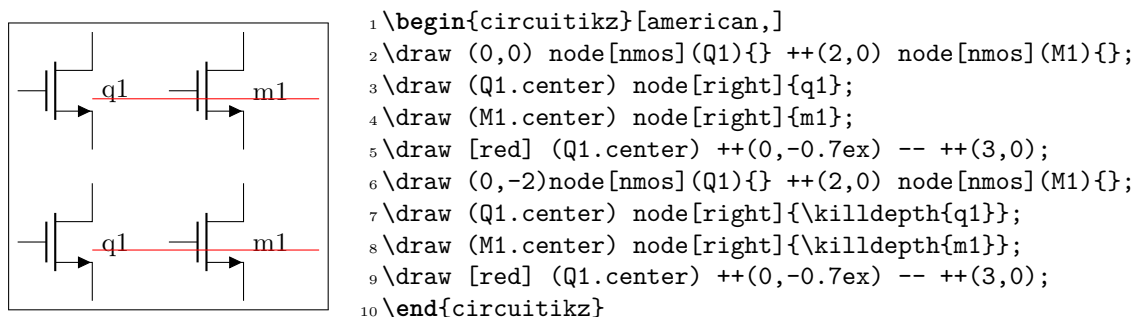
Another thing I like to modify with respect to the standard is the position of the arrows in transistors, which are normally midway the symbol. Issuing the following settings will move the arrows to the end or start of the corresponding pin.

```

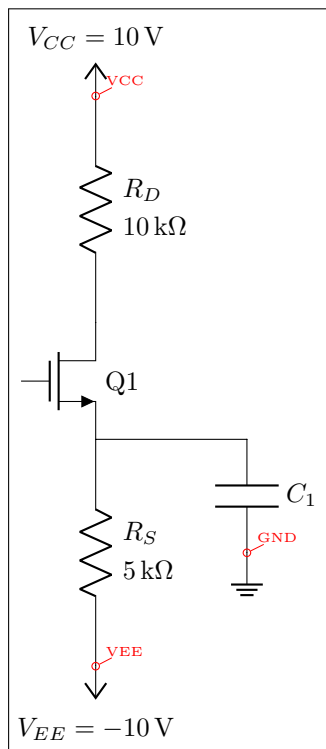
1 \ctikzset{tripoles/mos style/arrows,
2 tripoles/npn/arrow pos=0.8,
3 tripoles/pnp/arrow pos=0.8,
4 tripoles/nmos/arrow pos=0.8,
5 tripoles/pmos/arrow pos=0.6, }

```

The tricky thing about `\killdepth{}` macro is a finicky details; I do not like the standard position of labels on transistors (which is near the collector/drain) so I plot the label at the right of the **center** anchor. Without the `\killdepth` macro, the labels of different transistor will be adjusted so that the center of the box is at the **center** anchor, and as an effect, labels with descenders (like Q) will have a different baseline than labels without. You can see this here (it's really subtle):



We will start connecting the first transistor with the power supply with a couple of resistors. Notice that I am naming the nodes **GND**, **VCC** and **VEE**, so that I can use the coordinates to have all the supply rails at the same vertical position (more on this later).



```

1 \begin{circuitikz}[american,]
2   \draw (0,0) node[nmos,] (Q1){};
3   \draw (Q1.center) node[right]{\killdepth{Q1}};
4   \draw (Q1.S) to[R, l2^=$R_S$ and \SI{5}{k\ohm}]
5     ++(0,-3)
6     node[vee] (VEE){$V_{EE}=\SI{-10}{V}$};
7   \draw (Q1.D) to[R, l2_=$R_D$ and \SI{10}{k\ohm}]
8     ++(0,3)
9     node[vcc] (VCC){$V_{CC}=\SI{10}{V}$};
10  \draw (Q1.S) to[short] ++(2,0) to[C=$C_1$]
11    ++(0,-1.5) node[ground] (GND){};
12  \path (GND) \coord(GND) (VCC) \coord(VCC)
13    (VEE) \coord(VEE);
14 \end{circuitikz}

```

After that, let's add the input part. I will use a named node here, to refer to it to add the input source. Notice how the ground node is positioned: the coordinate (**in** | - **GND**) is the point with the horizontal coordinate of (**in**) and the horizontal one of (**GND**), lining it up with the ground of the capacitor C_1 .



```

1 \begin{circuitikz}[american, scale=0.7]
2   \draw (0,0) node[nmos,](Q1){};
3   \draw (Q1.center) node[right]
4     {\killdepth{Q1}};
5   \draw (Q1.S) to[R, l2^=$R_S$ and \SI
6     {5}{k\ohm}] ++(0,-3)
7     node[vee] (VEE){\SI{-10}{V}
8       };
9   \draw (Q1.D) to[R, l2^=$R_D$ and \SI
10     {10}{k\ohm}] ++(0,3)
11     node[vcc] (VCC){\SI{10}{V}
12       };
13   \draw (Q1.S) to[short] ++(2,0) to[C
14     =$C_1$] ++(0,-1.5) node[ground] (
15     GND){};
16   \draw (Q1.G) to[short] ++(-1,0)
17     \coord (in) to[R, l2^=$R_G$ and \
18     SI{1}{M\ohm}]
19     (in |- GND) node[ground]{};
20   \draw (in) to[C, l_=$C_2$,*-o]
21     ++(-1.5,0) node[left] (vi1){\SI{v_i=
22     v_{i1}}{};
23   \end{circuitikz}

```

Notice that the only absolute coordinate here is the first one, (0,0); so the elements are connected with relative movements and can be moved by just changing one number (for example, changing the `to[C=C_1] ++(0,-1.5)` will move *all* the grounds down).

This is the final circuit, with the nodes still marked:

```

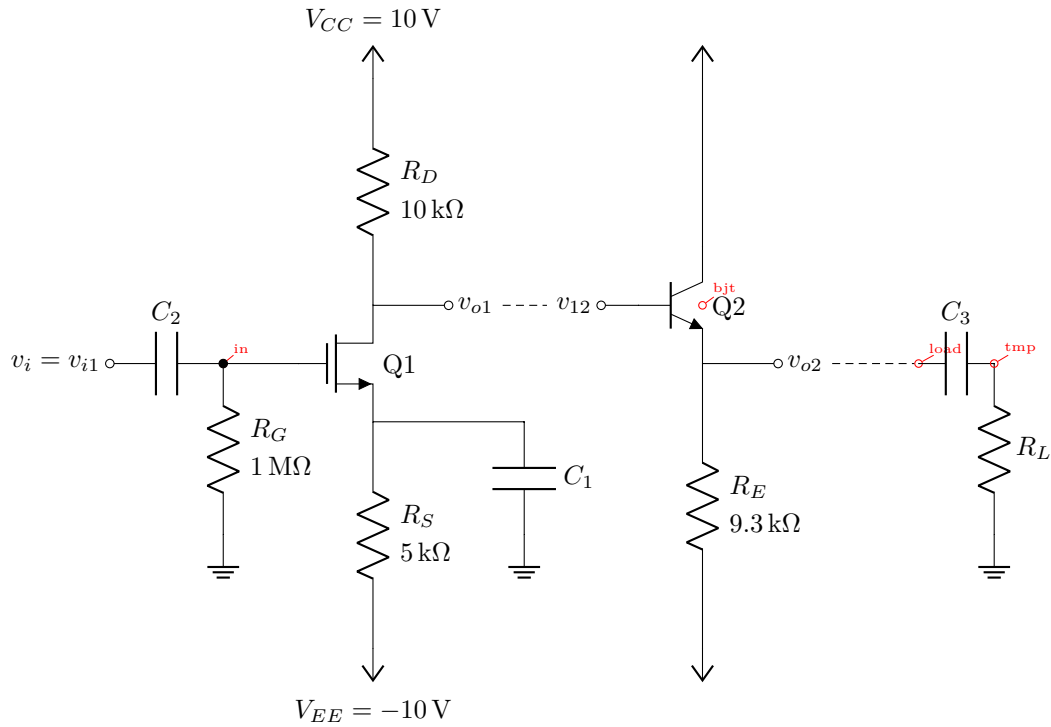
1 \tikzset{blockdef/.style={%
2   {Straight Barb[harpoon, reversed, right, length=0.2cm]}-{Straight Barb[
3     harpoon, reversed, left, length=0.2cm]},
4   blue, %densely dotted,
5 }}
6 \def\killdepth#1{{\raisebox{0pt}{\height}[0pt]{#1}}}
7 \def\coord(#1){coordinate(#1)}
8 \def\coord(#1){node[circle, red, draw, inner sep=1pt,pin={red, overlay, inner
9   sep=0.5pt, font=\tiny, pin distance=0.1cm, pin edge={red, overlay
10   },]45:#1}}(#1){}}
11 \begin{circuitikz}[american, ]
12   \draw (0,0) node[nmos,](Q1){};
13   \draw (Q1.center) node[right]{\killdepth{Q1}};
14   \draw (Q1.S) to[R, l2^=$R_S$ and \SI{5}{k\ohm}] ++(0,-3) node[vee] (VEE){\SI{-10}{V}}; %define VEE level
15   \draw (Q1.S) to[short] ++(2,0) to[C=$C_1$] ++(0,-1.5) node[ground] (GND){};
16   \draw (Q1.G) to[short] ++(-1,0) \coord (in) to[R, l2^=$R_G$ and \SI{1}{M\ohm}] (in |- GND) node[ground]{};
17   \draw (in) to[C, l_=$C_2$,*-o] ++(-1.5,0) node[left] (vi1){\SI{v_i=v_{i1}}{};
18   \draw (Q1.D) to[R, l2^=$R_D$ and \SI{10}{k\ohm}] ++(0,3) node[vcc] (VCC){\SI{10}{V}};
19   \draw (Q1.D) to[short, -o] ++(1,0) node[right] (vo1){\SI{v_o}};
20   %
21   \path (vo1) -- ++(3,0) \coord(bjt);
22   %

```

```

20 \draw (bjt) node[npn, ](Q2){};
21 \draw (Q2.center) node[right]{\killdepth{Q2}};
22 \draw (Q2.B) to[short, -o] ++(-0.5,0) node[left](vi2){$v_{12}$};
23 \draw (Q2.E) to[R, l2^=$R_E$ and \SI{9.3}{k\ohm}] (Q2.E |- VEE) node[vee]{};
24 \draw (Q2.E) to[short, -o] ++(1,0) node[right](vo2){$v_{o2}$};
25 \draw (Q2.C) to[short] (Q2.C |- VCC) node[vcc]{};
26 %
27 \path (vo2) ++(1.5,0) \coord(load);
28 \draw (load) to[C=$C_3$] ++(1,0) \coord(tmp) to[R=$R_L$] (tmp |- GND) node[
    ground]{};
29 \draw [densely dashed] (vo2) -- (load);
30 %
31 \draw [densely dashed] (vo1) -- (vi2);
32 %
33 \draw [blockdef] (vi1|-VEE) ++(0,-2) \coord(tmp)
34 -- node[midway, fill=white]{bloque 1} (vo1|- tmp);
35 \draw [blockdef] (vi2|-VEE) ++(0,-2) \coord(tmp)
36 -- node[midway, fill=white]{bloque 2} (vo2|- tmp);
37
38 \end{circuitikz}

```



 bloque 1

 bloque 2

3 The components

Components in CircuiTikZ come in two forms: a path-style form, to be used in to path specifications, and node-style, which will be instantiated by a `node` specification.

3.1 Path-style components

The path-style components are used as in the following way:

```

1 \begin{circuitikz}
2 \draw (0,0) to[#1=#2, #options] (2,0);
3 \end{circuitikz}

```

where **#1** is the name of the component, **#2** is an (optional) label, and **options** are optional labels, annotations, style specifier that will be explained in the rest of the manual.

Transistors and some other node-style components can also be placed using the syntax for bipoles. See section 3.18.2.

Most path-style components can be used as a node-style components; to access them, you add a **shape** to the main name of component (for example, **diodeshape**). Such a “node name” is specified in the description of each component.

3.1.1 Anchors

Normally, path-style component do not need anchors, although they have them just in case you need them. You have the basic “geographical” anchors (bipoles are defined horizontally and then rotate as needed):



In the case of bipoles, also shortened geographical anchors exists. In the description, it will be shown when a bipole has additional anchors. To use the anchors, just give a name to the bipole element.



```

1 \begin{circuitikz}
2 \draw (0,0) to[potentiometer, name=P, mirror] ++(0,2);
3 \draw (P.wiper) to[L] ++(2,0);
4 \end{circuitikz}

```

Alternatively, that you can use the shape form, and then use the **left** and **right** anchors to do your connections.



```

1 \begin{circuitikz}
2 \draw (0,0) node[potentiometershape, rotate=-90] (P){};
3 \draw (P.wiper) to[L] ++(2,0);
4 \end{circuitikz}

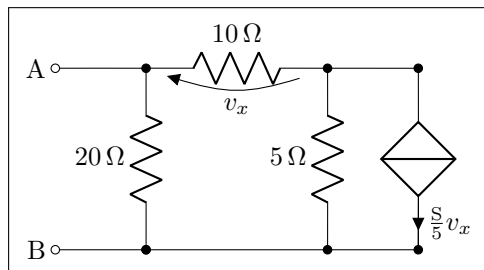
```

3.1.2 Customization

Pretty much all CircuiTikZ relies heavily on **pgfkeys** for value handling and configuration. Indeed, at the beginning of **circuitikz.sty** and in the file **pgfcirc.define.tex** a series of key definitions can be found that modify all the graphical characteristics of the package.

All can be varied using the **\ctikzset** command, anywhere in the code.

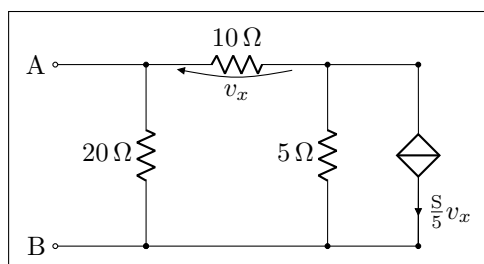
3.1.2.1 Components size Perhaps the most important parameter is `bipoles/length` (default 1.4 cm), which can be interpreted as the length of a resistor (including reasonable connections): all other lengths are relative to this value. For instance:



```

1 \ctikzset{bipoles/length=1.4cm}
2 \begin{circuitikz}[scale=1.2]\draw
3   (0,0) node[anchor=east] {B}
4     to[short, o-] (1,0)
5     to[R=20<\ohm>, *-] (1,2)
6     to[R=10<\ohm>, v=$v_x$] (3,2) -- (4,2)
7     to[cI=$\frac{\si{\siemens}}{5} v_x$, *-] (4,0) -- (3,0)
8     to[R=5<\ohm>, *-] (3,2)
9   (3,0) -- (1,0)
10  (1,2) to[short, -o] (0,2) node[anchor=east]{A}
11 \end{circuitikz}

```



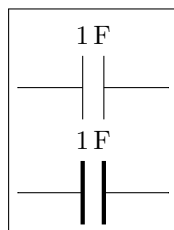
```

1 \ctikzset{bipoles/length=.8cm}
2 \begin{circuitikz}[scale=1.2]\draw
3   (0,0) node[anchor=east] {B}
4     to[short, o-] (1,0)
5     to[R=20<\ohm>, *-] (1,2)
6     to[R=10<\ohm>, v=$v_x$] (3,2) -- (4,2)
7     to[cI=$\frac{\si{\siemens}}{5} v_x$, *-] (4,0) -- (3,0)
8     to[R=5<\ohm>, *-] (3,2)
9   (3,0) -- (1,0)
10  (1,2) to[short, -o] (0,2) node[anchor=east]{A}
11 \end{circuitikz}

```

3.1.2.2 Thickness of the lines (globally)

You can change the thickness of the components lines with the parameter `bipoles/thickness` (default 2). The number is relative to the thickness of the normal lines leading to the component.



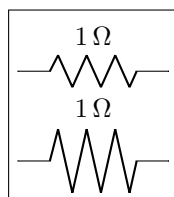
```

1 \ctikzset{bipoles/thickness=1}
2 \tikz \draw (0,0) to[C=1<\farad>] (2,0); \par
3 \ctikzset{bipoles/thickness=4}
4 \tikz \draw (0,0) to[C=1<\farad>] (2,0);

```

3.1.2.3 Shape of the components (on a per-component-class basis)

The shape of the components are adjustable with a lot of parameters; in this manual we will comment the main ones, but you can look into the source files specified above to find more.



```

1 \tikz \draw (0,0) to[R=1<\ohm>] (2,0); \par
2 \ctikzset{bipoles/resistor/height=.6}
3 \tikz \draw (0,0) to[R=1<\ohm>] (2,0);

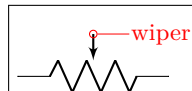
```

3.1.3 Descriptions

The typical entry in the component list will be like this:

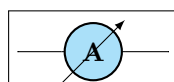


resistor: resistor, american style, type: path-style ,
nodename: resistorshape. Aliases: R, american
resistor.



pR: potentiometer, american style, type: path-style,
nodename: potentiometershape. Aliases: pR,
american potentiometer.

where you have all the needed information about the bipole, with also no-standard anchors. If the component can be filled it will be specified in the description. In addition, as an example, the component shown will be filled with the option `fill=cyan!30!white`:



ammeter: Ammeter, type: path-style, fillable ,
nodename: ammetershape.

3.2 Node-style components

Node-style components (monopoles, multipoles) can be drawn at a specified point with this syntax, where #1 is the name of the component:

```

1 \begin{circuitikz}
2 \draw (0,0) node[#1,#2] (#3) {#4};
3 \end{circuitikz}

```

Explanation of the parameters:

- #1: component name³ (mandatory)
- #2: list of comma separated options (optional)
- #3: name of an anchor (optional)
- #4: text written to the text anchor of the component (optional)

Most path-style components can be used as a node-style components; to access them, you add a **shape** to the main name of component (for example, **diodeshape**). Such a “node name” is specified in the description of each component.

Notice: Nodes must have curly brackets at the end, even when empty. An optional anchor (#3) can be defined within round brackets to be addressed again later on. And please don't forget the semicolon to terminate the `\draw` command.

Also notice: If using the `\tikzexternalize` feature, as of Tikz 2.1 all pictures must end with `\end{tikzpicture}`. Thus you *cannot* use the `circuitikz` environment.

Which is ok: just use the environment `tikzpicture`: everything will work there just fine.

3.2.1 Mirroring and flipping

Mirroring and flipping of node components is obtained by using the TikZ keys `yscale` and `yscale`. Notice that this parameters affect also text labels, so they need to be un-scaled by hand.

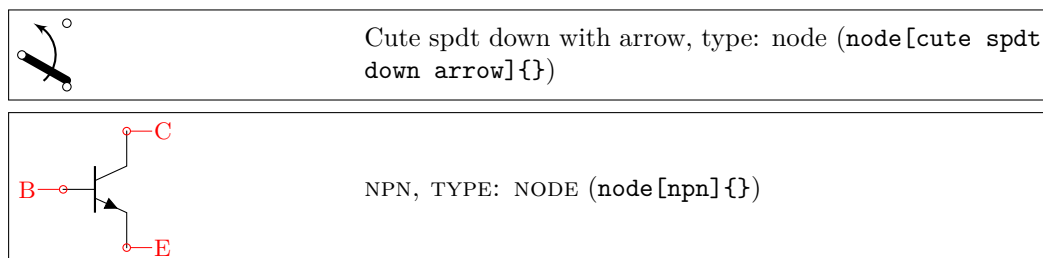


3.2.2 Anchors

Node components anchors are variable across the various kind of components, so they will be described better after each category is presented in the manual.

3.2.3 Descriptions

The typical entry in the component list will be like this:



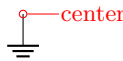










³For using bipoles as nodes, the name of the node is **#1shape**.

All the shapes defined by CircuiTikZ. These are all **pgf** nodes, so they are usable in both **pgf** and **TikZ**. If the component can be filled it will be specified in the description. In addition, as an example, the component shown will be filled with the option `fill=cyan!30!white`:



3.3 Grounds and supply voltages

For the grounds, the **center** anchor is put on the connecting point of the symbol, so that you can use them directly in a **path** specification.

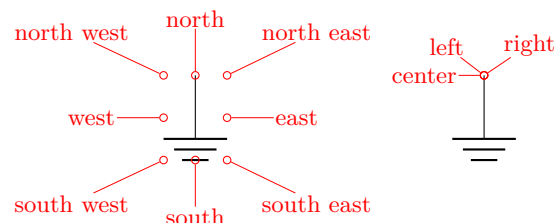
	Ground, type: node (<code>node[ground]{}</code>)
	Reference ground, type: node (<code>node[rground]{}</code>)
	Signal ground, type: node, fillable (<code>node[sground]{}</code>)
	Thicker ground, type: node (<code>node[tground]{}</code>)
	Noiseless ground, type: node (<code>node[nground]{}</code>)
	Protective ground, type: node (<code>node[pground]{}</code>)
	Chassis ground ⁴ , type: node (<code>node[cground]{}</code>)
	European style ground, type: node (<code>node[eground]{}</code>)
	European style ground, version 2 ⁵ , type: node (<code>node[eground2]{}</code>)
	VCC/VDD, type: node (<code>node[vcc]{}</code>)
	VEE/VSS, type: node (<code>node[vee]{}</code>)

⁴These last three were contributed by Luigi «Liverpool»

⁵These last two were contributed by @fotesan)

3.3.1 Grounds anchors

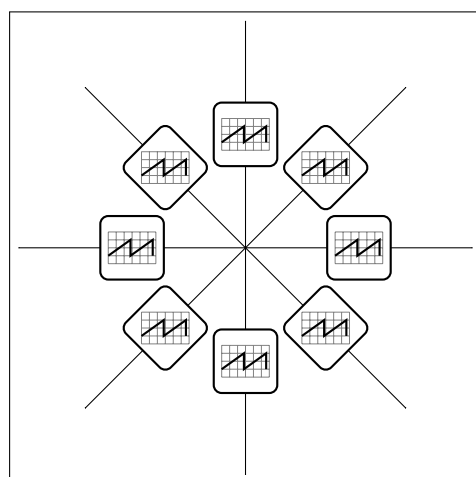
Anchors for grounds are a bit strange, given that they have the `center` spot at the same location than `north` and all the ground will develop “going down”:



3.4 Instruments

	<code>ammeter</code> : Ammeter, type: path-style, fillable , nodename: ammetershape.
	<code>voltmeter</code> : Voltmeter, type: path-style, fillable , nodename: voltmetershape.
	<code>ohmmeter</code> : Ohmmeter, type: path-style, fillable , nodename: ohmmetershape.
	<code>oscope</code> : Oscilloscope ⁶ , type: path-style, fillable, nodename: oscopeshape.

The `oscope` element will not rotate the “graph” shown with the component:



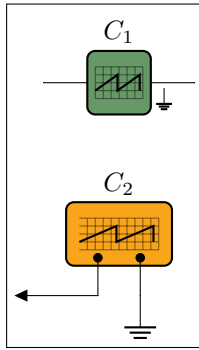
```

1 \begin{circuitikz}
2   \foreach \a in {0,45,...,350} {
3     \draw (0,0) to[oscope] (\a:3);
4   }
5 \end{circuitikz}

```

Moreover, the node-style usage of the `oscope` is also interesting, using the additional `in 1` and `in 2` anchors; notice that in this case you can use the text content of the node to put labels above it. Moreover, you can change the size of the oscilloscope by changing `bipoles/oscope/width` and `bipoles/oscope/height` keys (which both default at 0.6).

⁶Suggested by @nobl on GitHub



```

1 \begin{circuitikz}
2   \draw (0,1)
3     to[oscope=$C_1$, fill=green!20!gray, name=O1] ++(2,0);
4   \path (O1.right)
5     node[ground, scale=0.5, below right=4pt]{};
6   \ctikzset{bipoles/oscope/width=1.0}
7   \draw (1,-1)
8     node[oscopeshape, fill=yellow!20!orange] (O2){$C_2$};
9   \draw (O2.in 2) to[short, *-] ++(0,-0.5) node[ground]{};
10  \draw (O2.in 1) to[short, *-] ++(0,-0.5)
11    -- ++(-1,0) node[currarrow, xscale=-1]{};
12 \end{circuitikz}

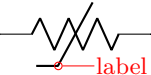
```

3.5 Resistive bipoles


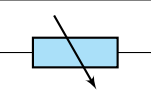
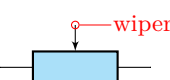
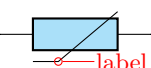
	short: Short circuit, type: path-style , nodename: shortshape.
	open: Open circuit, type: path-style , nodename: openshape.
	generic: Generic (symmetric) bipole, type: path-style, fillable , nodename: genericshape.
	tgeneric: Tunable generic bipole, type: path-style, fillable , nodename: tgenericshape.
	ageneric: Generic asymmetric bipole, type: path-style, fillable , nodename: agenericshape.
	fullgeneric: Generic asymmetric bipole (full), type: path-style , nodename: fullgenericshape.
	tfullgeneric: Tunable generic bipole (full), type: path-style , nodename: tfullgenericshape.
	memristor: Memristor, type: path-style, fillable , nodename: memristorshape. Aliases: Mr.

If `americanresistors` option is active (or the style `[american resistors]` is used; this is the default for the package), the resistors are displayed as follows:

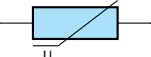



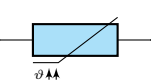
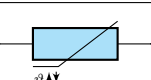


	R: Resistor, type: path-style, nodename: resistorshape. Aliases: american resistor.
	vR: Variable resistor, type: path-style, nodename: vresistorshape. Aliases: variable american resistor.
	pR: Potentiometer, type: path-style, nodename: potentiometershape. Aliases: american potentiometer.

	sR : Resisitive sensor, type: path-style, nodename: resistivesensshape. Aliases: american resistive sensor.
---	--

If instead `europeanresistors` option is active (or the style `[european resistors]` is used), the resistors, variable resistors and potentiometers are displayed as follows:

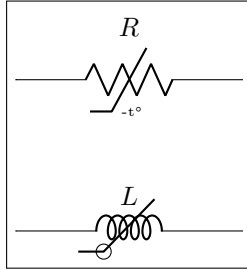
	R : Resistor, type: path-style, fillable, nodename: genericshape. Aliases: european resistor.
	vR : Variable resistor, type: path-style, fillable, nodename: tgenericshape. Aliases: variable european resistor.
	pR : Potentiometer, type: path-style, fillable, nodename: genericpotentiometershape. Aliases: european potentiometer.
	sR : Resistive sensor, type: path-style, fillable, nodename: thermistorshape. Aliases: european resistive sensor.

Other miscellaneous resistor-like devices:

	varistor : Varistor, type: path-style, fillable , nodename: varistorshape.
	phR : Photoresistor, type: path-style, fillable, nodename: photoresistorshape. Aliases: photoresistor.
	thermocouple : Thermocouple, type: path-style , nodename: thermocoupleshape.
	thR : Thermistor, type: path-style, fillable, nodename: thermistorshape. Aliases: thermistor.
	thRp : PTC thermistor, type: path-style, fillable, nodename: thermistorptcshape. Aliases: thermistor ptc.
	thRn : NTC thermistor, type: path-style, fillable, nodename: thermistorntcshape. Aliases: thermistor ntc.
	fuse : Fuse, type: path-style, fillable , nodename: fuseshape.
	afuse : Asymmetric fuse, type: path-style, fillable , nodename: afuseshape. Aliases: asymmetric fuse.

3.5.1 Generic sensors anchors

Generic sensors have an extra label to help positioning the type of dependence, if needed:



```

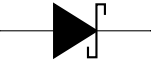
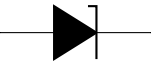
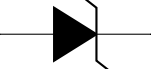
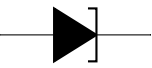

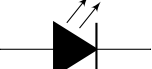


1 \begin{circuitikz}
2   \draw (0,2) to[sR, l=$R$, name=mySR] ++(3,0);
3   \node [font=\tiny, right] at(mySR.label) {-t\si{\degree}};
4   \draw (0,0) to[sL, l=$L$, name=mySL] ++(3,0);
5   \node [draw, circle, inner sep=2pt] at(mySL.label) {};
6 \end{circuitikz}

```

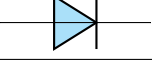
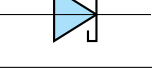
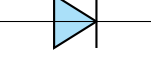
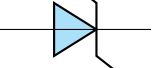
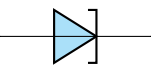
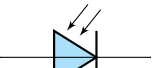
The anchor is positioned just on the corner of the segmented line crossing the component.

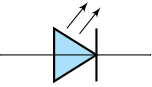
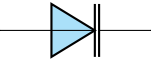
3.6 Diodes and such

	empty diode: Empty diode, type: <code>path-style</code> , fillable, nodename: <code>emptydiodeshape</code> . Aliases: <code>Do</code> .
	empty Schottky diode: Empty Schottky diode, type: <code>path-style</code> , fillable, nodename: <code>emptysdiodeshape</code> . Aliases: <code>sDo</code> .
	empty Zener diode: Empty Zener diode, type: <code>path-style</code> , fillable, nodename: <code>emptyzdiodeshape</code> . Aliases: <code>zDo</code> .
	empty ZZener diode: Empty ZZener diode, type: <code>path-style</code> , fillable, nodename: <code>emptyzzdiodeshape</code> . Aliases: <code>zzDo</code> .
	empty tunnel diode: Empty tunnel diode, type: <code>path-style</code> , fillable, nodename: <code>emptytdiodeshape</code> . Aliases: <code>tDo</code> .
	empty photodiode: Empty photodiode, type: <code>path-style</code> , fillable, nodename: <code>emptypdiodeshape</code> . Aliases: <code>pDo</code> .
	empty led: Empty led, type: <code>path-style</code> , fillable, nodename: <code>emptylediodeshape</code> . Aliases: <code>leDo</code> .
	empty varcap: Empty varcap, type: <code>path-style</code> , fillable, nodename: <code>emptyvarcapshape</code> . Aliases: <code>VCo</code> .
	empty bidirectionaldiode: Empty bidirectionaldiode, type: <code>path-style</code> , fillable, nodename: <code>emptybidirectionaldiodeshape</code> . Aliases: <code>biDo</code> .
	full diode: Full diode, type: <code>path-style</code> , nodename: <code>fulldiodeshape</code> . Aliases: <code>D*</code> .

	full Schottky diode: Full Schottky diode, type: path-style, nodename: fullsdiodeshape. Aliases: sD*.
	full Zener diode: Full Zener diode, type: path-style, nodename: fullzdiodeshape. Aliases: zD*.
	full ZZener diode: Full ZZener diode, type: path-style, nodename: fullzzdiodeshape. Aliases: zzD*.
	full tunnel diode: Full tunnel diode, type: path-style, nodename: fulltdiodeshape. Aliases: tD*.
	full photodiode: Full photodiode, type: path-style, nodename: fullpdiodeshape. Aliases: pD*.
	full led: Full led, type: path-style, nodename: fulllediodeshape. Aliases: leD*.
	full varcap: Full varcap, type: path-style, nodename: fullvarcapshape. Aliases: VC*.
	full bidirectionaldiode: Full bidirectionaldiode, type: path-style, nodename: fullbidirectionaldiodeshape. Aliases: biD*.

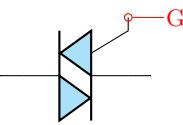
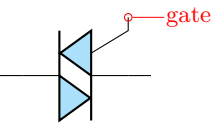
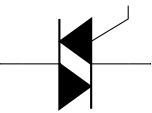
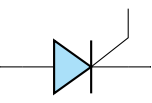
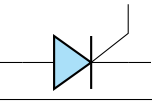
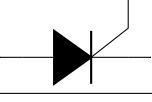
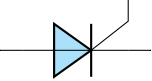
These shapes has no exact node-style counterpart, because the stroke line is build upon the empty variants:

	stroke diode: Stroke diode, type: path-style, fillable, nodename: emptydiodeshape. Aliases: D-.
	stroke Schottky diode: Stroke Schottky diode, type: path-style, fillable, nodename: emptysdiodeshape. Aliases: sD-.
	stroke Zener diode: Stroke Zener diode, type: path-style, fillable, nodename: emptyzdiodeshape. Aliases: zD-.
	stroke ZZener diode: Stroke ZZener diode, type: path-style, fillable, nodename: emptyzzdiodeshape. Aliases: zzD-.
	stroke tunnel diode: Stroke tunnel diode, type: path-style, fillable, nodename: emptytdiodeshape. Aliases: tD-.
	stroke photodiode: Stroke photodiode, type: path-style, fillable, nodename: emptypdiodeshape. Aliases: pD-.

	stroke led: Stroke led, type: path-style, fillable, nodename: emptylediodeshape. Aliases: leD-.
	stroke varcap: Stroke varcap, type: path-style, fillable, nodename: emptyvarcapshape. Aliases: VC-.

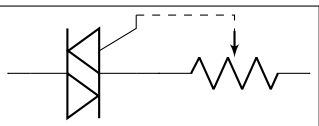
3.7 Tripole-like diodes

The following tripoles are entered with the usual command of the form

	triac: Standard triac (shape depends on package option), type: path-style, fillable, nodename: emptytriacshape. Aliases: Tr.
	empty triac: Empty triac, type: path-style, fillable, nodename: emptytriacshape. Aliases: Tro.
	full triac: Full triac, type: path-style, nodename: fulltriacshape. Aliases: Tr*.
	thyristor: Standard thyristor (shape depends on package option), type: path-style, fillable, nodename: emptythyristorshape. Aliases: Ty.
	empty thyristor: Empty thyristor, type: path-style, fillable, nodename: emptythyristorshape. Aliases: Tyo.
	full thyristor: Full thyristor, type: path-style, nodename: fullthyristorshape. Aliases: Ty*.
	stroke thyristor: Stroke thyristor, type: path-style, fillable, nodename: emptythyristorshape. Aliases: Ty-.

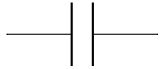
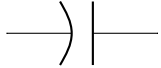
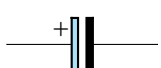
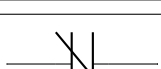
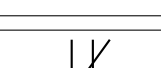
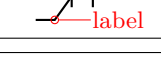
3.7.1 Triacs anchors

When inserting a thyristor, a triac or a potentiometer, one needs to refer to the third node—gate (gate or G) for the former two; wiper (wiper or W) for the latter one. This is done by giving a name to the bipole:


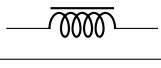

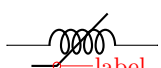
	<pre> 1 \begin{circuitikz} \draw 2 (0,0) to[Tr, n=TRI] (2,0) 3 to[pR, n=POT] (4,0); 4 \draw[dashed] (TRI.G) - (POT.wiper) 5 ;\end{circuitikz} </pre>
---	---

The package options `fulldiode`, `strokediode`, and `emptydiode` (and the styles `[full diodes]`, `[stroke diodes]`, and `[empty diodes]`) define which shape will be used by abbreviated commands such that `D`, `sD`, `zD`, `zzD`, `tD`, `pD`, `leD`, `VC`, `Ty`, `Tr` (no stroke symbol available!).


3.8 Basic dynamical bipoles



	capacitor : Capacitor, type: <code>path-style</code> , nodename: <code>capacitorshape</code> . Aliases: <code>C</code> .
	polar capacitor : Polar capacitor, type: <code>path-style</code> , nodename: <code>polarcapacitorshape</code> . Aliases: <code>pC</code> .
	ecapacitor : Electrolytic capacitor, type: <code>path-style</code> , fillable, nodename: <code>ecapacitorshape</code> . Aliases: <code>eC</code> , <code>elko</code> .
	variable capacitor : Variable capacitor, type: <code>path-style</code> , nodename: <code>vcapacitorshape</code> . Aliases: <code>vC</code> .
	capacitive sensor : Capacitive sensor, type: <code>path-style</code> , nodename: <code>capacitivesensshape</code> . Aliases: <code>sC</code> .
	piezoelectric : Piezoelectric Element, type: <code>path-style</code> , fillable, nodename: <code>piezoelectricshape</code> . Aliases: <code>PZ</code> .

If (default behaviour) `cuteinductors` option is active (or the style `[cute inductors]` is used), the inductors are displayed as follows:



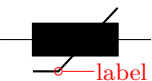
	L : Inductor, type: <code>path-style</code> , nodename: <code>cuteinductorshape</code> . Aliases: <code>cute inductor</code> .
	cute choke : Choke, type: <code>path-style</code> , nodename: <code>cutechokesshape</code> .
	vL : Variable inductor, type: <code>path-style</code> , nodename: <code>vcuteinductorshape</code> . Aliases: <code>variable cute inductor</code> .
	sL : Inductive sensor, type: <code>path-style</code> , nodename: <code>scuteinductorshape</code> . Aliases: <code>cute inductive sensor</code> .

If `americaninductors` option is active (or the style `[american inductors]` is used), the inductors are displayed as follows:

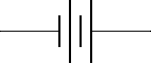
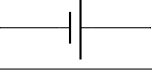

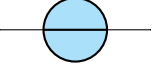



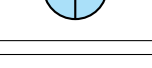
	L : Inductor, type: <code>path-style</code> , nodename: <code>americaninductorshape</code> . Aliases: <code>american inductor</code> .
---	---


	vL: Variable inductor, type: <code>path-style</code> , nodename: <code>vamericaninductorshape</code> . Aliases: <code>variable american inductor</code> .
	sL: Inductive sensor, type: <code>path-style</code> , nodename: <code>samericaninductorshape</code> . Aliases: <code>american inductive sensor</code> .

Finally, if `europeaninductors` option is active (or the style `[european inductors]` is used), the inductors are displayed as follows:

	L: Inductor, type: <code>path-style</code> , nodename: <code>fullgenericshape</code> . Aliases: <code>european inductor</code> .
	vL: Variable inductor, type: <code>path-style</code> , nodename: <code>tfullgenericshape</code> . Aliases: <code>variable european inductor</code> .
	sL: Inductive sensor, type: <code>path-style</code> , nodename: <code>sfullgenericshape</code> . Aliases: <code>european inductive sensor</code> .

3.9 Stationary sources

	battery: Battery, type: <code>path-style</code> , nodename: <code>batteryshape</code> .
	battery1: Single battery cell, type: <code>path-style</code> , nodename: <code>battery1shape</code> .
	battery2: Single battery cell, type: <code>path-style</code> , nodename: <code>battery2shape</code> .
	european voltage source: Voltage source (european style), type: <code>path-style</code> , fillable, nodename: <code>vsourceshape</code> .
	cute european voltage source: Voltage source (cute european style), type: <code>path-style</code> , fillable, nodename: <code>vsourceshape</code> . Aliases: <code>vsourceshapeC</code> , <code>ceV</code> .
	american voltage source: Voltage source (american style), type: <code>path-style</code> , fillable, nodename: <code>vsourceshape</code> .
	european current source: Current source (european style), type: <code>path-style</code> , fillable, nodename: <code>isourceshape</code> .
	cute european current source: Current source (cute european style), type: <code>path-style</code> , fillable, nodename: <code>isourceshape</code> . Aliases: <code>isourceshapeC</code> , <code>ceI</code> .



	american current source: Current source (american style), type: path-style, fillable, nodename: isourceAMshape.
---	--

If (default behaviour) `européancurrents` option is active (or the style `[european currents]` is used), the shorthands `current source`, `isource`, and `I` are equivalent to `european current source`. Otherwise, if `americancurrents` option is active (or the style `[american currents]` is used) they are equivalent to `american current source`.






Similarly, if (default behaviour) `europeanvoltages` option is active (or the style `[european voltages]` is used), the shorthands `voltage source`, `vsource`, and `V` are equivalent to `european voltage source`. Otherwise, if `americanvoltages` option is active (or the style `[american voltages]` is used) they are equivalent to `american voltage source`.

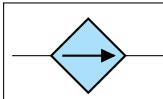
3.10 Sinusoidal sources

Here because I was asked for them. But how do you distinguish one from the other?!

	sinusoidal voltage source: Sinusoidal voltage source, type: path-style, fillable, nodename: vsourcesinshape. Aliases: vsourcesin, sV.
	sinusoidal current source: Sinusoidal current source, type: path-style, fillable, nodename: isourcesinshape. Aliases: isourcesin, sI.

3.11 Controlled sources

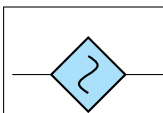
	european controlled voltage source: Controlled voltage source (european style), type: path-style, fillable, nodename: cvsourceshape.
	cute european controlled voltage source: Voltage source (cute european style), type: path-style, fillable, nodename: cvsourcCshape. Aliases: cvsourcC, cceV.
	american controlled voltage source: Controlled voltage source (american style), type: path-style, fillable, nodename: cvsourcAMshape.
	european controlled current source: Controlled current source (european style), type: path-style, fillable, nodename: cisourceshape.
	cute european controlled current source: Current source (cute european style), type: path-style, fillable, nodename: cisourcCshape. Aliases: cisourcC, cceI.



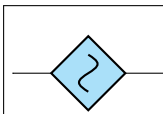
american controlled current source: Controlled current source (american style), type: path-style, fillable, nodename: cisourceAMshape.

If (default behaviour) `européancurrents` option is active (or the style `[european currents]` is used), the shorthands `controlled current source`, `cisource`, and `cI` are equivalent to `european controlled current source`. Otherwise, if `americancurrents` option is active (or the style `[american currents]` is used) they are equivalent to `american controlled current source`.

Similarly, if (default behaviour) `europeanvoltages` option is active (or the style `[european voltages]` is used), the shorthands `controlled voltage source`, `cvsource`, and `cV` are equivalent to `european controlled voltage source`. Otherwise, if `americanvoltages` option is active (or the style `[american voltages]` is used) they are equivalent to `american controlled voltage source`.



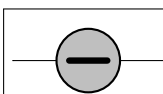
controlled sinusoidal voltage source: Controlled sinusoidal voltage source, type: path-style, fillable, nodename: cvsourcesinshape. Aliases: `controlled vsourcesin`, `cvsourcesin`, `csV`.



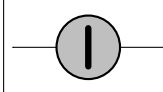
controlled sinusoidal current source: Controlled sinusoidal current source, type: path-style, fillable, nodename: cisourcesinshape. Aliases: `controlled isourcesin`, `cisourcesin`, `csI`.

3.12 Noise sources

In this case, the “direction” of the source has no sense. Noise sources are filled in gray by default, but if you choose the dashed style, they become fillable.

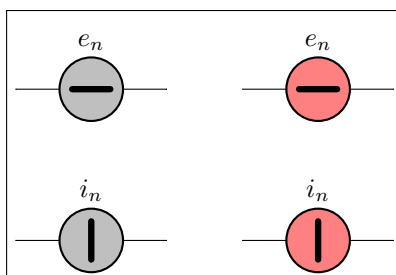


noise voltage source: Sinusoidal voltage source, type: path-style, nodename: vsourcenshape. Aliases: `vsourcen`, `nV`.



noise current source: Sinusoidal current source, type: path-style, nodename: isourcenshape. Aliases: `isourcen`, `nI`.

You can change the fill color with the key `circuitikz/bipoles/noise sources/fillcolor`:

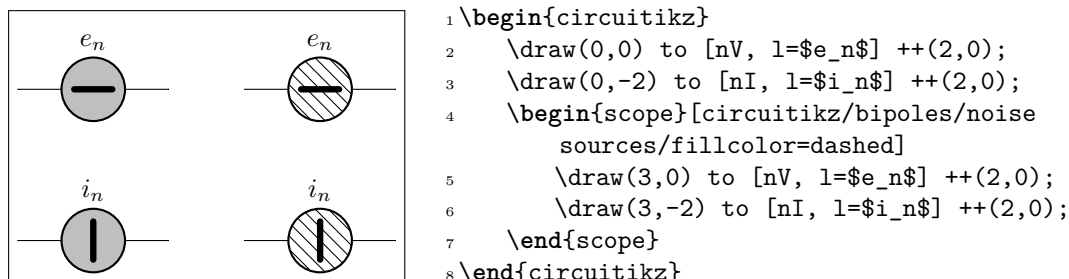


```

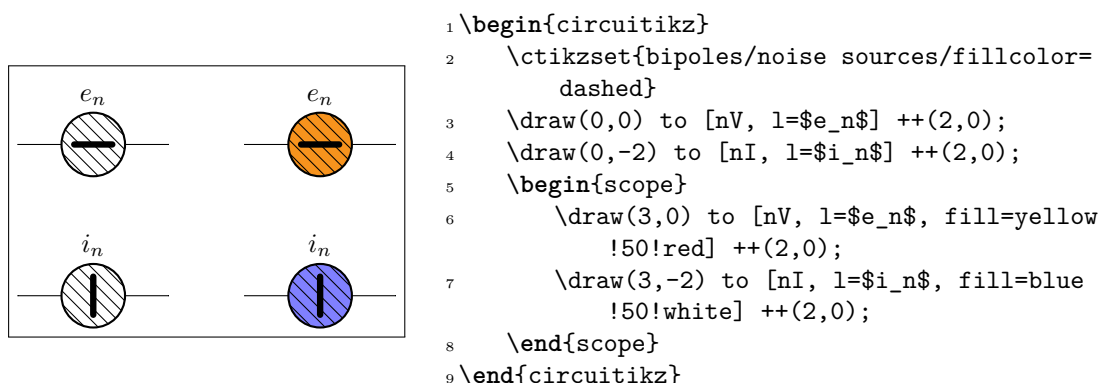
1 \begin{circuitikz}
2   \draw(0,0) to [nV, l=$e_n$] ++(2,0);
3   \draw(0,-2) to [nI, l=$i_n$] ++(2,0);
4   \begin{scope}[circuitikz/bipoles/noise
5     sources/fillcolor=red!50]
6     \draw(3,0) to [nV, l=$e_n$] ++(2,0);
7     \draw(3,-2) to [nI, l=$i_n$] ++(2,0);
8   \end{scope}
9 \end{circuitikz}

```

If you prefer a patterned noise generator (similar to the one you draw by hand) you can use the fake color `dashed`:



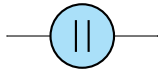
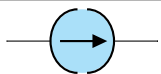
Notice that if you choose the dashed style, the noise sources are fillable:



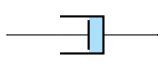

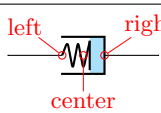
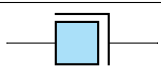
3.13 Special sources

	square voltage source: Square voltage source, type: path-style, fillable, nodename: vsourcesquashape. Aliases: vsourcesquare, sqV.
	vsourcetri: Triangle voltage source, type: path-style, fillable, nodename: vsourcetrishape. Aliases: tV.
	esource: Empty voltage source, type: path-style, fillable, nodename: esourceshape.
	pvsources: Photovoltaic-voltage source, type: path-style, fillable, nodename: pvsourceshape.
	ioosources: Double Zero style current source, type: path-style, fillable, nodename: oosourceshape.
	voosources: Double Zero style voltage source, type: path-style, fillable, nodename: oosourceshape.

3.14 DC sources

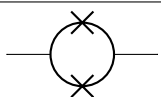
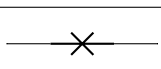
	dcvsource: DC voltage source, type: path-style, fillable , nodename: dcvsourceshape.
	dcisource: DC current source, type: path-style, fillable , nodename: dcisourceshape.

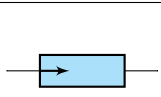
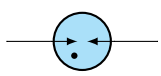
3.15 Mechanical Analogy

	dampers: Mechanical Damping, type: path-style, fillable , nodename: dampershape.
	spring: Mechanical Stiffness, type: path-style , nodename: springshape.
	viscoe: Mechanical viscoelastic element ⁷ , type: path-style, fillable , nodename: viscoeshape.
	mass: Mechanical Mass, type: path-style, fillable , nodename: massshape.

3.16 Other bipoles

Here you'll find bipoles that are not easily grouped in the categories above.

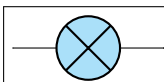
	squid: Squid, type: path-style , nodename: squidshape.
	barrier: Barrier, type: path-style , nodename: barriershape.

	european gas filled surge arrester: European gas filled surge arrester, type: path-style, fillable , nodename: european gas filled surge arrestershape.
	american gas filled surge arrester: American gas filled surge arrester, type: path-style, fillable , nodename: american gas filled surge arrestershape.

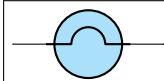
⁷Suggested by @Alex in <https://tex.stackexchange.com/q/484268/38080>

If (default behaviour) `europiangfsurgearrester` option is active (or the style `[european gas filled surge arrester]` is used), the shorthands `gas filled surge arrester` and `gf surge arrester` are equivalent to the european version of the component.

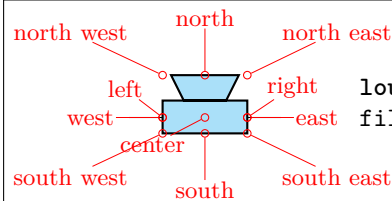
If otherwise `americangfsurgearrester` option is active (or the style `[american gas filled surge arrester]` is used), the shorthands `gas filled surge arrester` and `gf surge arrester` are equivalent to the american version of the component.



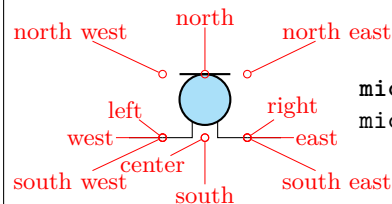
lamp: Lamp, type: path-style, fillable ,
nodename: lampshape.



bulb: Bulb, type: path-style, fillable , nodename:
bulbshape.

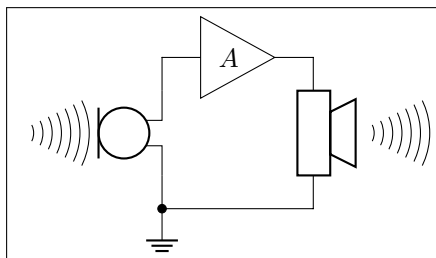


loudspeaker: loudspeaker, type: path-style,
fillable , nodename: loudspeakershape.



mic: mic, type: path-style, fillable , nodename:
micshape.

You can use microphones and loudspeakers with `waves` (see section 3.24) too:



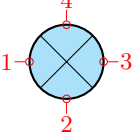
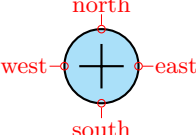


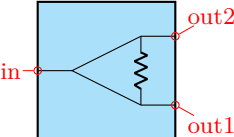
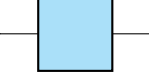
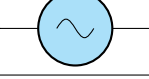
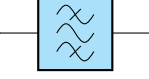



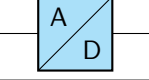

```


1  \begin{circuitikz}
2      \draw (0,0) to[mic, name=M] ++(0,2)
3      to[amp, t=$A$] ++(2,0)
4      to[loudspeaker, name=L] ++(0,-2)
5      to[short, -*] (0,0) node[ground]{};
6      \node [waves, scale=0.7, left=5pt]
7          at(M.north) {};
8      \node [waves, scale=0.7, right]
9          at(L.north) {};
10 \end{circuitikz}



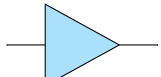
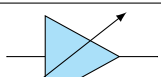
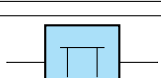
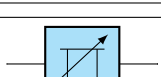

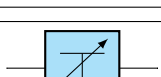

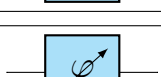
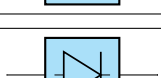
```

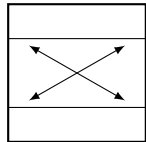
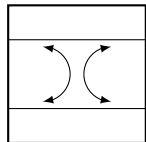
3.17 Block diagram components

Contributed by Stefan Erhardt.

	MIXER, TYPE: NODE, FILLABLE (node[mixer]{})
	ADDER, TYPE: NODE, FILLABLE (node[adder]{})
	OSCILLATOR, TYPE: NODE, FILLABLE (node[oscillator]{})
	CIRCULATOR, TYPE: NODE, FILLABLE (node[circulator]{})
	WILKINSON DIVIDER, TYPE: NODE, FILLABLE (node[wilkinson]{})
	twoport : generic two port ⁸ , type: path-style, fillable , nodename: twoportshape.
	vco : vco, type: path-style, fillable , nodename: vcoshape.
	bandpass : bandpass, type: path-style, fillable , nodename: bandpassshape.
	bandstop : bandstop, type: path-style, fillable , nodename: bandstopshape.
	highpass : highpass, type: path-style, fillable , nodename: highpassshape.
	lowpass : lowpass, type: path-style, fillable , nodename: lowpassshape.
	adc : A/D converter, type: path-style, fillable , nodename: adcshape.
	dac : D/A converter, type: path-style, fillable , nodename: dacshape.

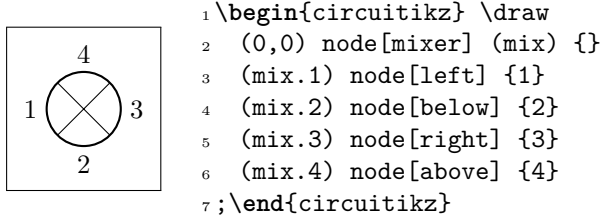
⁸To specify text to be put in the component: `twoport[t=text]`: 

	dsp: DSP, type: path-style, fillable , nodename: dspshape.
	fft: FFT, type: path-style, fillable , nodename: fftshape.
	amp: amplifier, type: path-style, fillable , nodename: ampshape.
	vamp: VGA, type: path-style, fillable , nodename: vampshape.
	piattenuator: π attenuator, type: path-style, fillable , nodename: piattenuatorshape.
	vpiattenuator: var. π attenuator, type: path-style, fillable , nodename: vpiattenuatorshape.
	tattenuator: T attenuator, type: path-style, fillable , nodename: tattenuatorshape.
	vtattenuator: var. T attenuator, type: path-style, fillable , nodename: vtattenuatorshape.
	phaseshifter: phase shifter, type: path-style, fillable , nodename: phaseshiftershape.
	vphaseshifter: var. phase shifter, type: path-style, fillable , nodename: vphaseshiftershape.
	detector: detector, type: path-style, fillable , nodename: detectorshape.

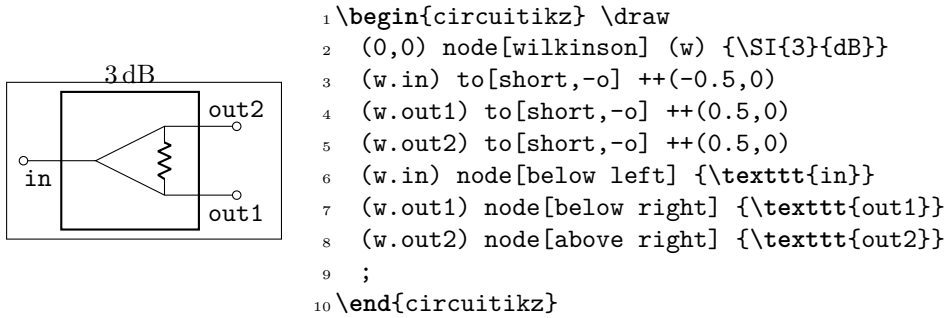
	Coupler, type: node (node[coupler]{})
	Coupler, 2, type: node (node[coupler2]{})

3.17.1 Blocks anchors

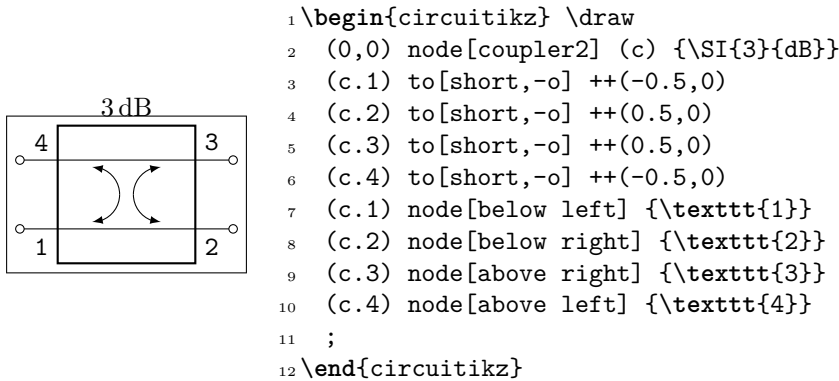
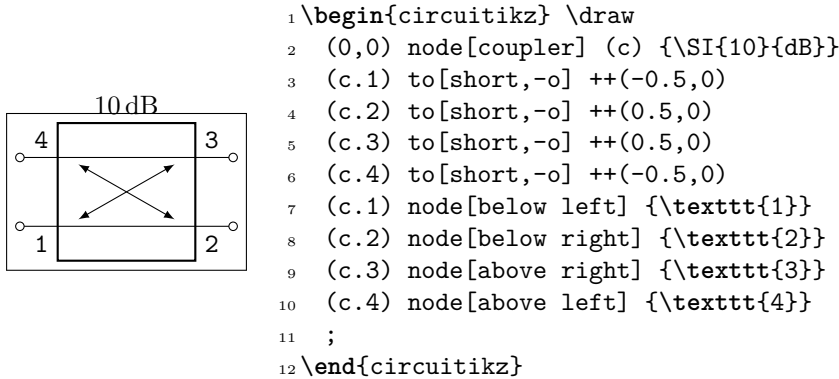
The ports of the mixer and adder can be addressed with numbers or `west/south/east/north`:



The Wilkinson divider has:

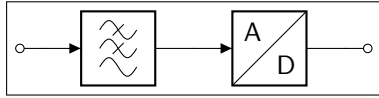


The couplers have:



3.17.2 Blocks customization

With the option > you can draw an arrow to the input of the block diagram symbols.

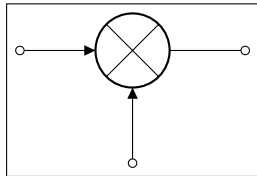


```

1 \begin{circuitikz} \draw
2   (0,0) to[short,o-] ++(0.3,0)
3   to[lowpass,>] ++(2,0)
4   to[adc,>] ++(2,0)
5   to[short,-o] ++(0.3,0);
6 \end{circuitikz}

```

3.17.2.1 Multi ports Since inputs and outputs can vary, input arrows can be placed as nodes. Note that you have to rotate the arrow on your own:

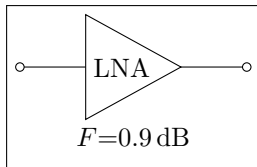


```

1 \begin{circuitikz} \draw
2   (0,0) node[mixer] (m) {}
3   (m.1) to[short,-o] ++(-1,0)
4   (m.2) to[short,-o] ++(0,-1)
5   (m.3) to[short,-o] ++(1,0)
6   (m.1) node[inputarrow] {}
7   (m.2) node[inputarrow,rotate=90] {};
8 \end{circuitikz}

```

3.17.2.2 Labels and custom twoport boxes Some twoports have the option to place a normal label (l=) and a inner label (t=).

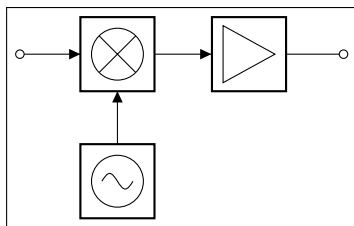


```

1 \begin{circuitikz}
2   \ctikzset{bipoles/amp/width=0.9}
3   \draw (0,0) to[amp,t=LNA,l=$F{=}0.9\$,dB,o-o] ++(3,0);
4 \end{circuitikz}

```

3.17.2.3 Box option Some devices have the possibility to add a box around them. The inner symbol scales down to fit inside the box.

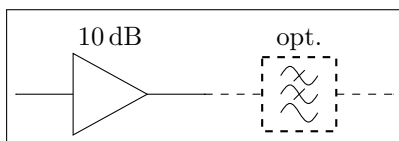


```

1 \begin{circuitikz} \draw
2   (0,0) node[mixer,box,anchor=east] (m) {}
3   to[amp,box,>,-o] ++(2.5,0)
4   (m.west) node[inputarrow] {} to[short,-o]
5     ++(-0.8,0)
6   (m.south) node[inputarrow,rotate=90] {} --
7     ++(0,-0.7) node[oscillator,box,anchor=north] {};
8 \end{circuitikz}

```

3.17.2.4 Dash optional parts To show that a device is optional, you can dash it. The inner symbol will be kept with solid lines.

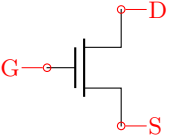
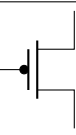
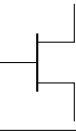
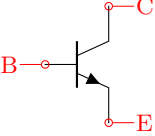
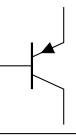
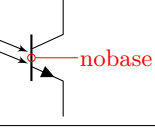
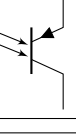
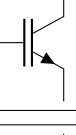
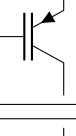



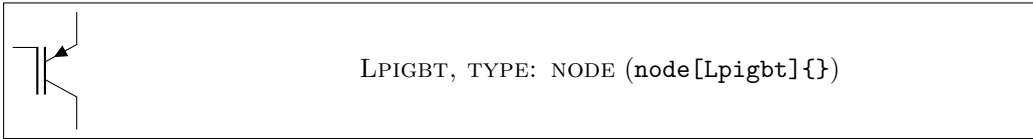
```

1 \begin{circuitikz}
2   \draw (0,0) to[amp,l=\SI{10}{dB}] ++(2.5,0);
3   \draw[dashed] (2.5,0) to[lowpass,l=opt.]
4     ++(2.5,0);
5 \end{circuitikz}

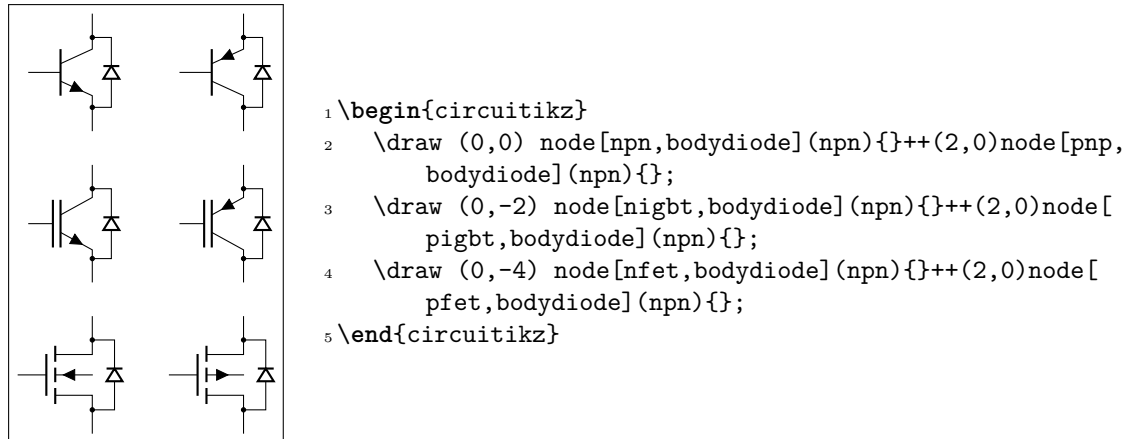
```

3.18 Transistors

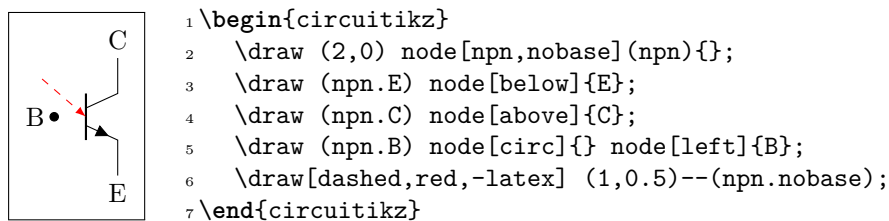
	NMOS, TYPE: NODE (node[nmos]{})
	PMOS, TYPE: NODE (node[pmos]{})
	HEMT, TYPE: NODE (node[hemt]{})
	NPN, TYPE: NODE (node[npn]{})
	PNP, TYPE: NODE (node[pnp]{})
	NPN, TYPE: NODE (node[npn,photo]{})
	PNP, TYPE: NODE (node[pnp,photo]{})
	NIGBT, TYPE: NODE (node[nigbt]{})
	PIGBT, TYPE: NODE (node[pigbt]{})
	LNIGBT, TYPE: NODE (node[lnigbt]{})



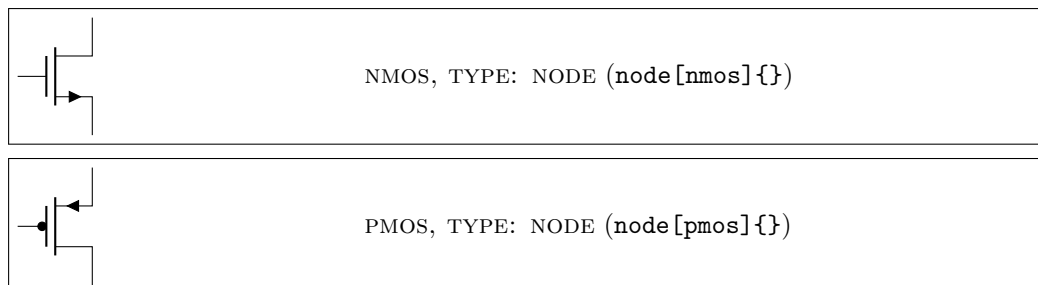
For all transistors a bodydiode (or freewheeling diode) can automatically be drawn. Just use the global option `bodydiode`, or for single transistors, the `tikz`-option `bodydiode`:



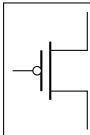
The Base/Gate connection of all transistors can be disabled by using the options `nogate` or `nobase`, respectively. The Base/Gate anchors are floating, but there is an additional anchor "nogate"/"nobase", which can be used to point to the unconnected base:



If the option `arrowmos` is used (or after the command `\ctikzset{tripoles/mos style/arrows}` is given), this is the output:

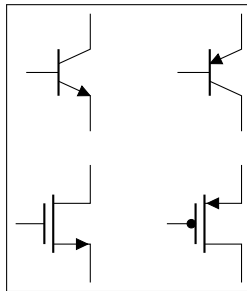


To draw the PMOS circle non-solid, use the option `emptycircle` or the command `\ctikzset{tripoles/pmos style/emptycircle}`.



PMOS, TYPE: NODE (node[pmos,emptycircle]{})

If you prefer different position of the arrows in transistors and FETs, you can adjust them like this (it works for the other BJT-based transistors, too):

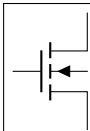


```

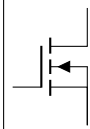
1 \begin{circuitikz}
2   \ctikzset{tripoles/mos style/arrows,
3     tripoles/npn/arrow pos=0.8,
4     tripoles/pnp/arrow pos=0.8,
5     tripoles/nmos/arrow pos=0.8,
6     tripoles/pmos/arrow pos=0.6, }
7   \draw (0,0) node[npn, ](nnp){};
8   \draw (2,0) node[pnp, ](npn){};
9   \draw (0,-2) node[nmos, ](nnp){};
10  \draw (2,-2) node[pmos, ](npn){};
11 \end{circuitikz}

```

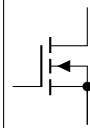
NFETs and PFETs have been incorporated based on code provided by Clemens Helfmeier and Theodor Borsche. Use the package options `fetsolderdot`/`nofetsolderdot` to enable/disable solderdot at some fet-transistors. Additionally, the `solderdot` option can be enabled/disabled for single transistors with the option `"solderdot"` and `"nosolderdot"`, respectively.



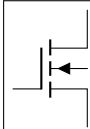
NFET, TYPE: NODE (node[nfet]{})



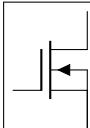
NIGFETE, TYPE: NODE (node[nigfete]{})



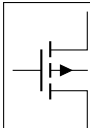
NIGFETE, TYPE: NODE (node[nigfete,solderdot]{})



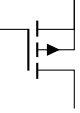
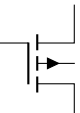
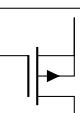
NIGFETEBULK, TYPE: NODE (node[nigfetebulk]{})



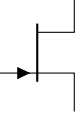
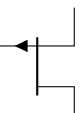
NIGFETD, TYPE: NODE (node[nigfetd]{})



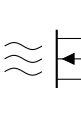
PFET, TYPE: NODE (node[pfet]{})

	PIGFETE, TYPE: NODE (node[pigfete]{})
	PIGFETEBULK, TYPE: NODE (node[pigfetebulk]{})
	PIGFETD, TYPE: NODE (node[pigfetd]{})

NJFET and PJFET have been incorporated based on code provided by Danilo Piazzalunga:

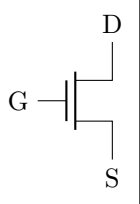
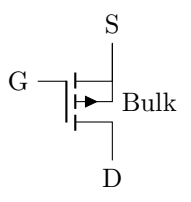
	NJFET, TYPE: NODE (node[njfet]{})
	PJFET, TYPE: NODE (node[pjfet]{})

ISFET

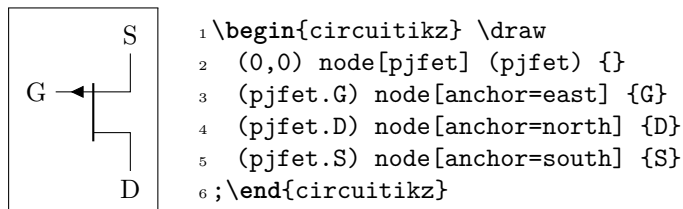
	ISFET, TYPE: NODE (node[isfet]{})
---	-----------------------------------

3.18.1 Transistors anchors

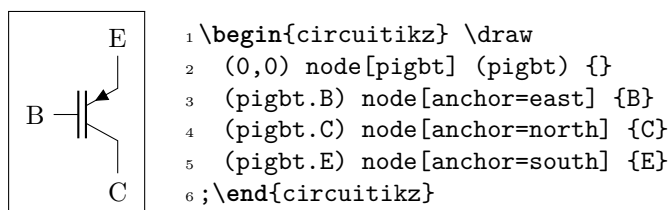
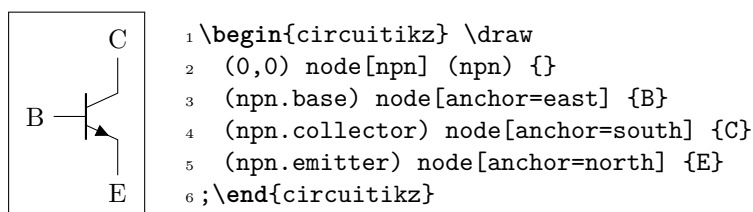
For NMOS, PMOS, NFET, NIGFETE, NIGFETD, PFET, PIGFETE, and PIGFETD transistors one has base, gate, source and drain anchors (which can be abbreviated with B, G, S and D):

	<pre> 1 \begin{circuitikz} \draw 2 (0,0) node[nmos] (mos) {} 3 (mos.gate) node[anchor=east] {G} 4 (mos.drain) node[anchor=south] {D} 5 (mos.source) node[anchor=north] {S} 6 ;\end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz} \draw 2 (0,0) node[pigfete] (pigfete) {} 3 (pigfete.G) node[anchor=east] {G} 4 (pigfete.D) node[anchor=north] {D} 5 (pigfete.S) node[anchor=south] {S} 6 (pigfete.bulk) node[anchor=west] {Bulk} 7 ;\end{circuitikz} </pre>

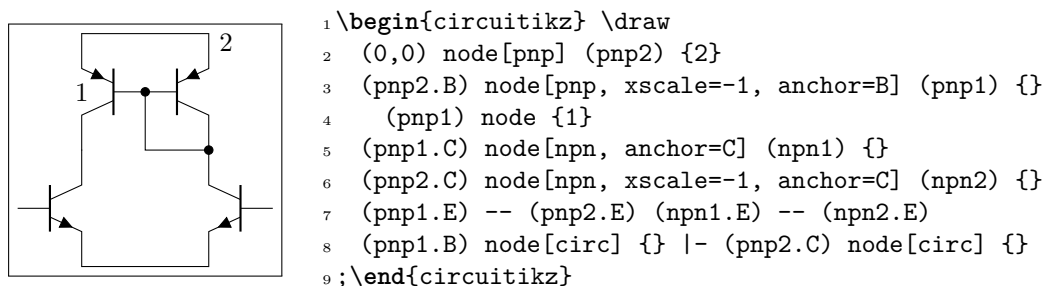
Similarly NJFET and PJFET have **gate**, **source** and **drain** anchors (which can be abbreviated with **G**, **S** and **D**):



For NPN, PNP, NIGBT, and PIGBT transistors the anchors are **base**, **emitter** and **collector** anchors (which can be abbreviated with **B**, **E** and **C**):

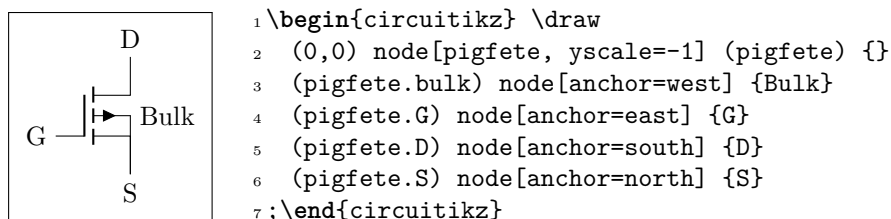


Here is one composite example (please notice that the `xscale=-1` style would also reflect the label of the transistors, so here a new node is added and its text is used, instead of that of `pnnp1`):



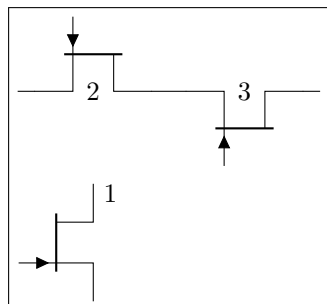
Notice that the text labels of transistors are somewhat buggy. It is better to see explicit anchors to set transistor's names.

Similarly, transistors like other components can be reflected vertically:



3.18.2 Transistor paths

For syntactical convenience transistors can be placed using the normal path notation used for bipoles. The transistor type can be specified by simply adding a “T” (for transistor) in front of the node name of the transistor. It will be placed with the base/gate orthogonal to the direction of the path:

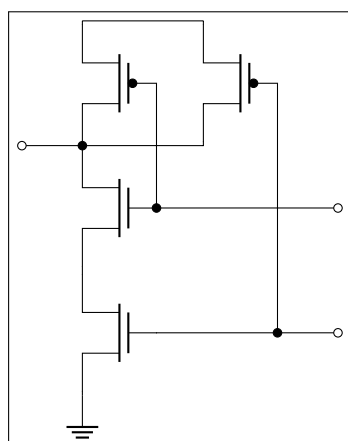


```

1 \begin{circuitikz} \draw
2   (0,0) node[njfet] {1}
3   (-1,2) to[Tnjfet=2] (1,2)
4     to[Tnjfet=3, mirror] (3,2);
5 \end{circuitikz}

```

Access to the gate and/or base nodes can be gained by naming the transistors with the **n** or **name** path style:



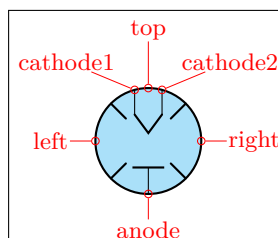
```

1 \begin{circuitikz} \draw[yscale=1.1, xscale=.8]
2   (2,4.5) -- (0,4.5) to[Tpmos, n=p1] (0,3)
3     to[Tnmos, n=n1] (0,1.5)
4     to[Tnmos, n=n2] (0,0) node[ground] {}
5   (2,4.5) to[Tpmos,n=p2] (2,3) to[short, -*] (0,3)
6   (p1.G) -- (n1.G) to[short, *-o] ($(n1.G)+(3,0)$)
7   (n2.G) ++(2,0) node[circ] {} -| (p2.G)
8   (n2.G) to[short, -o] ($(n2.G)+(3,0)$)
9   (0,3) to[short, -o] (-1,3)
10 \end{circuitikz}

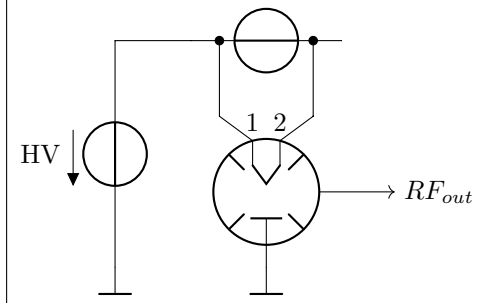
```

The **name** property is available also for bipoles, although this is useful mostly for triac, potentiometer and thyristor (see 3.7).

3.19 Electronic Tubes



Magnetron, type: node, fillable (`node[magnetron]{}`)



```

1 \begin{circuitikz}
2 \draw (0,-2)node[rground](gnd){} to[
   voltage source,v<={HV}]++(0,3)
   --++(1,0)to[V,n=DC]++(2,0);
3 \draw (2,-1) node[magnetron,scale=1](
   magn){};
4 \draw (DC.left)++(-0.2,0)to [short,*-]
   ++(0,-1) to [short] (magn.cathode1);
5 \draw (DC.right)++(0.2,0)to [short,*-]
   ++(0,-1) to [short] (magn.cathode2);
6 \draw (magn.anode) to [short] (magn.
   anode|-gnd) node[rground]{};
7 \draw (magn.cathode1)node[above]{$1$};
8 \draw (magn.cathode2)node[above]{$2$};
9 \draw[->](magn.east) --++(1,0)node[
   right]{$RF_{out}$};
10 \end{circuitikz}

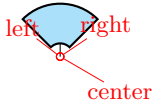
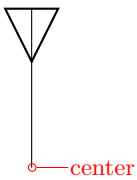
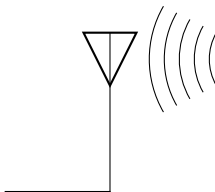
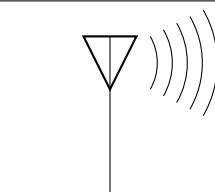
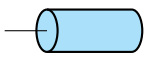


```

3.20 RF components

For the RF components, similarly to the grounds and supply rails, the `center` anchor is put on the connecting point of the symbol, so that you can use them directly in a `path` specification.

Notes that in the transmission and receiving antennas, the “waves” are outside the geographical anchors.

	Bare Antenna, type: node, fillable (node[bareantenna]{A})
	Bare TX Antenna, type: node, fillable (node[bareTXantenna]{Tx})
	Bare RX Antenna, type: node, fillable (node[bareRXantenna]{Rx})
	mstline : Microstrip transmission line ⁹ , type: path-style, fillable , nodename: mstlineshape.
	Microstrip stub, type: node, fillable (node[mslstub]{text})
	Microstrip port, type: node, fillable (node[msport]{T})

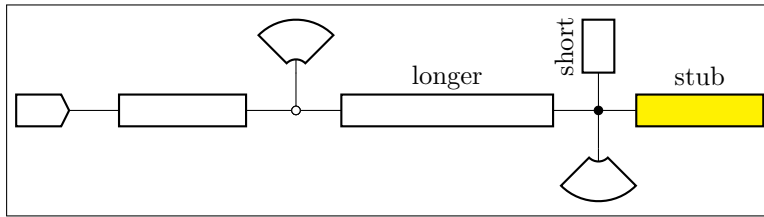
	Microstrip radial stub, type: node, fillable (<code>node[msrstub]{}</code>)
	Antenna, type: node (<code>node[antenna]{}</code>)
	Receiving antenna, type: node (<code>node[rxantenna]{}</code>)
	Transmitting antenna, type: node (<code>node[txantenna]{}</code>)
	Transmission line stub, type: node, fillable (<code>node[tlinestub]{}</code>)
	TL: Transmission line, type: path-style, fillable, nodename: <code>tlineshape</code> . Aliases: <code>transmission line</code> , <code>tline</code> .
	match, type: node (<code>node[match]{}</code>)

3.20.1 Microstrip customization

The microstrip linear components (`mstline`, `msslub`, `msport`) height depend from the parameters `bipoles/mstline/height` (for the three or them, default 0.3). The width are specified in `bipoles/mstline/width` for the first two and by `monopoles/msport/width` for the port (defaults: 1.2, 0.5).

For the length parameter of the transmission line there is a shortcut in the form of the direct parameter `mstlinelen`.

⁹This four components were suggested by @tcpluess on GitHub



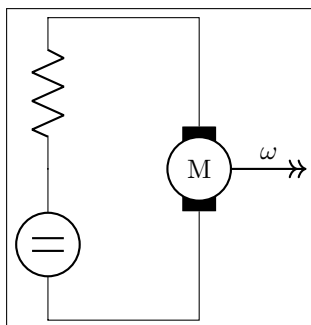
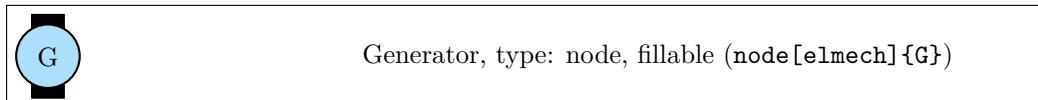
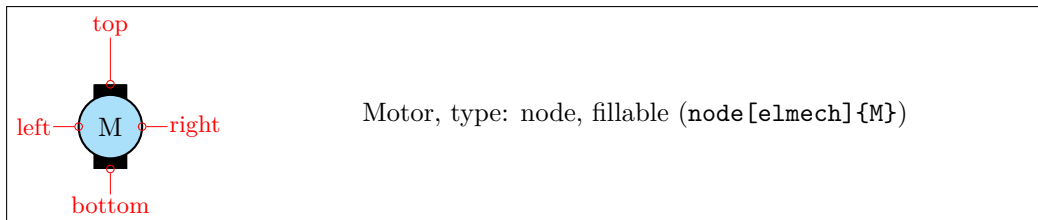
```

1 \begin{circuitikz}
2   \draw (0,0) node[msport, right, xscale=-1]{}
3     to[mstline, -o] ++(3,0) coordinate(there)
4     to[mstline, mstlinelen=2, l=longer, o-*] ++(4,0)
5       coordinate(here) -- ++(0.5,0) node[mslstub, fill=yellow]{stub}
6       (here) -- ++(0,0.5) node[mslstub, rotate=90, mstlinelen=0.5]{short};
7   \draw (there) to[short, o-] ++(0, 0.5) node[msrstub]{};
8   \draw (here) -- ++(0, -0.5) node[msrstub, yscale=-1]{};
9 \end{circuitikz}

```

3.21 Electro-Mechanical Devices

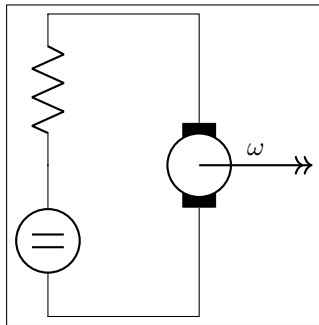
The internal part of the motor and generator are, by default, filled white (to avoid compatibility problems with older versions of the package).



```

1 \begin{circuitikz}
2 \draw (2,0) node[elmech](motor){M};
3 \draw (motor.north) |-(0,2) to [R] ++(0,-2) to[
4   dcvsource]++(0,-2) -| (motor.bottom);
5 \draw[thick,->>] (motor.right)--++(1,0)node[midway,
6   above]{\omega};
7 \end{circuitikz}

```

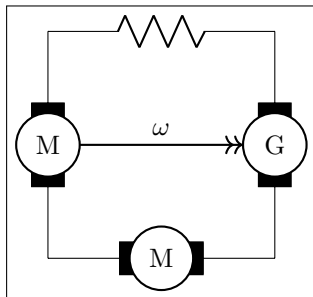


```

1 \begin{circuitikz}
2 \draw (2,0) node[elmech] (motor){};
3 \draw (motor.north) |- (0,2) to [R] ++(0,-2) to [
    dcvsource] ++(0,-2) -| (motor.bottom);
4 \draw[thick,->>] (motor.center) -- ++(1.5,0) node[midway,
    above] {\omega};
5 \end{circuitikz}

```

The symbols can also be used along a path, using the transistor-path-syntax (T in front of the shape name, see section 3.18.2). Don't forget to use parameter n to name the node and get access to the anchors:



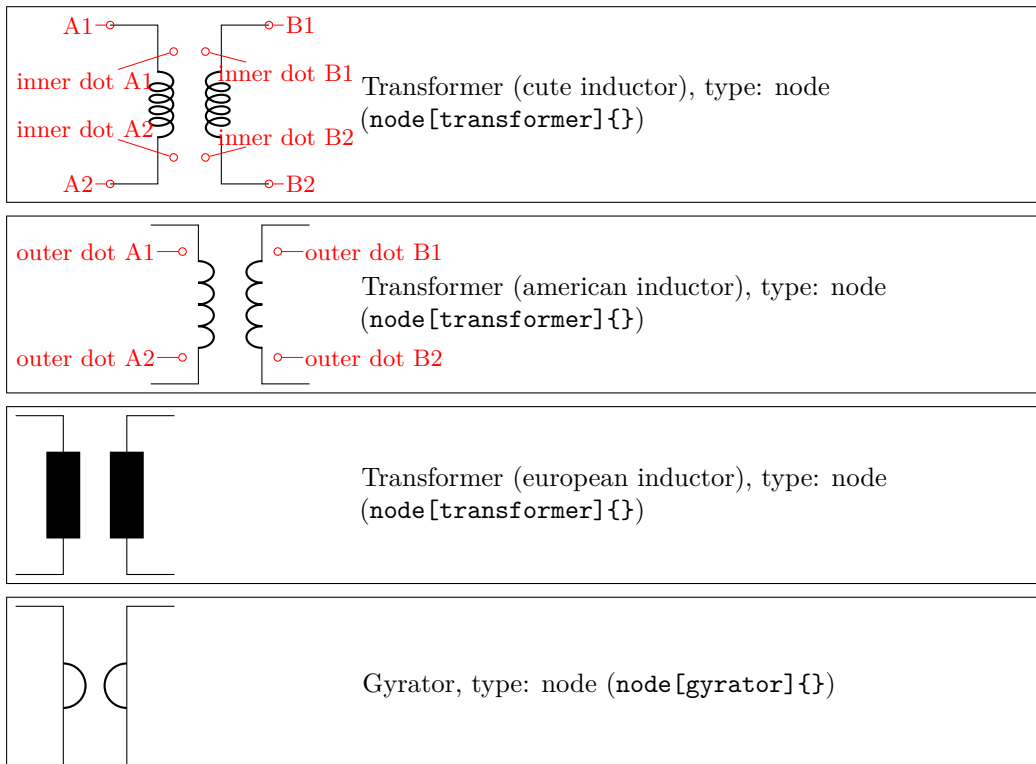
```

1 \begin{circuitikz}
2 \draw (0,0) to [Telmech=M,n=motor] ++(0,-3) to [
    Telmech=M] ++(3,0) to [Telmech=G,n=generator]
    ++(0,3) to [R] (0,0);
3 \draw[thick,->>] (motor.left) -- (generator.left) node[
    midway,above] {\omega};
4 \end{circuitikz}

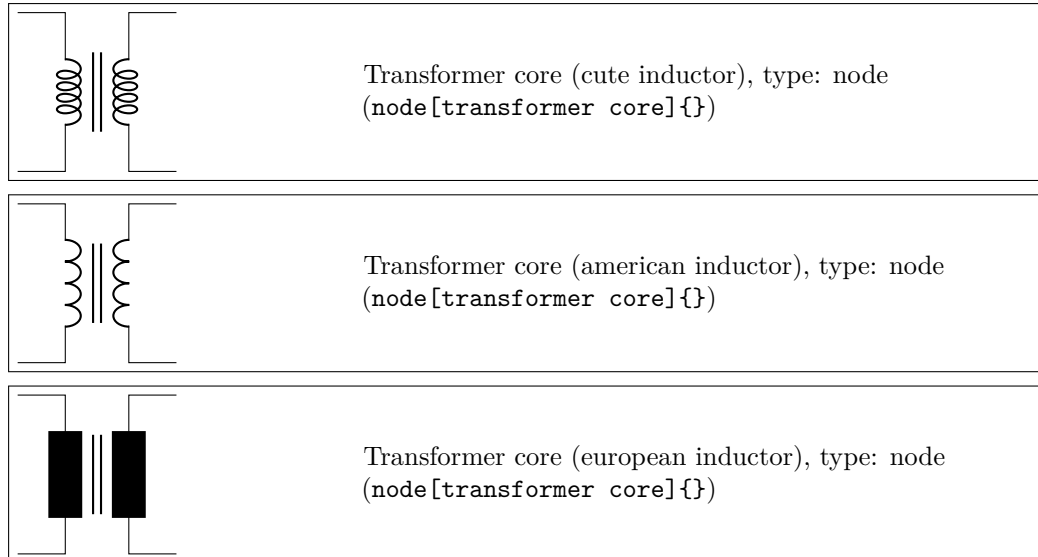
```

3.22 Double bipoles

Transformers automatically use the inductor shape currently selected. These are the three possibilities:



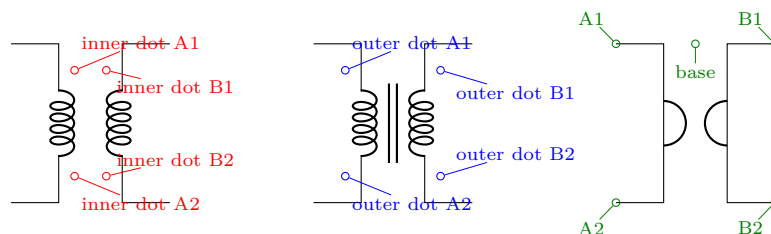
Transformers with core are also available:



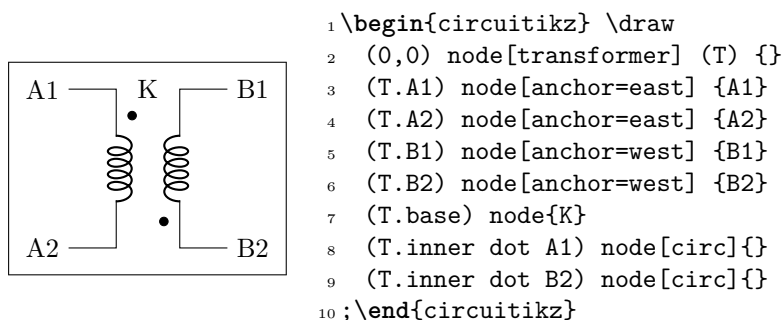
3.22.1 Double dipoles anchors

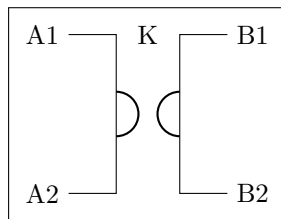
All the double bipoles/quadrupoles have the four anchors, two for each port. The first port, to the left, is port A, having the anchors A1 (up) and A2 (down); same for port B.

They also expose the `base` anchor, for labelling, and anchors for setting dots or signs to specify polarity. The set of anchors, to which the standard “geographical” `north`, `north east`, etc. is here:



Also the standard “geographical” `north`, `north east`, etc. are defined. A couple of examples follow:



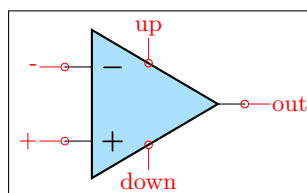


```

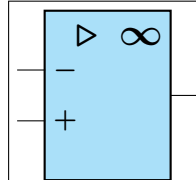
1 \begin{circuitikz} \draw
2 (0,0) node[gyrator] (G) {}
3 (G.A1) node[anchor=east] {A1}
4 (G.A2) node[anchor=east] {A2}
5 (G.B1) node[anchor=west] {B1}
6 (G.B2) node[anchor=west] {B2}
7 (G.base) node{K}
8 ;\end{circuitikz}

```

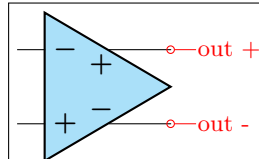
3.23 Amplifiers



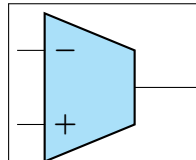
Operational amplifier, type: node, fillable (node[op amp]{})



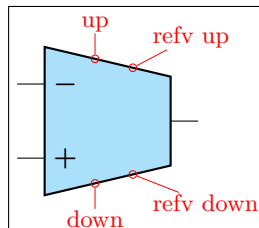
Operational amplifier compliant to DIN/EN 60617 standard, type: node, fillable (node[en amp]{})



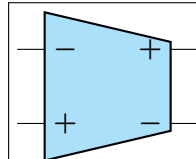
Fully differential operational amplifier¹⁰, type: node, fillable (node[fd op amp]{})



transconductance amplifier, type: node, fillable (node[gm amp]{})

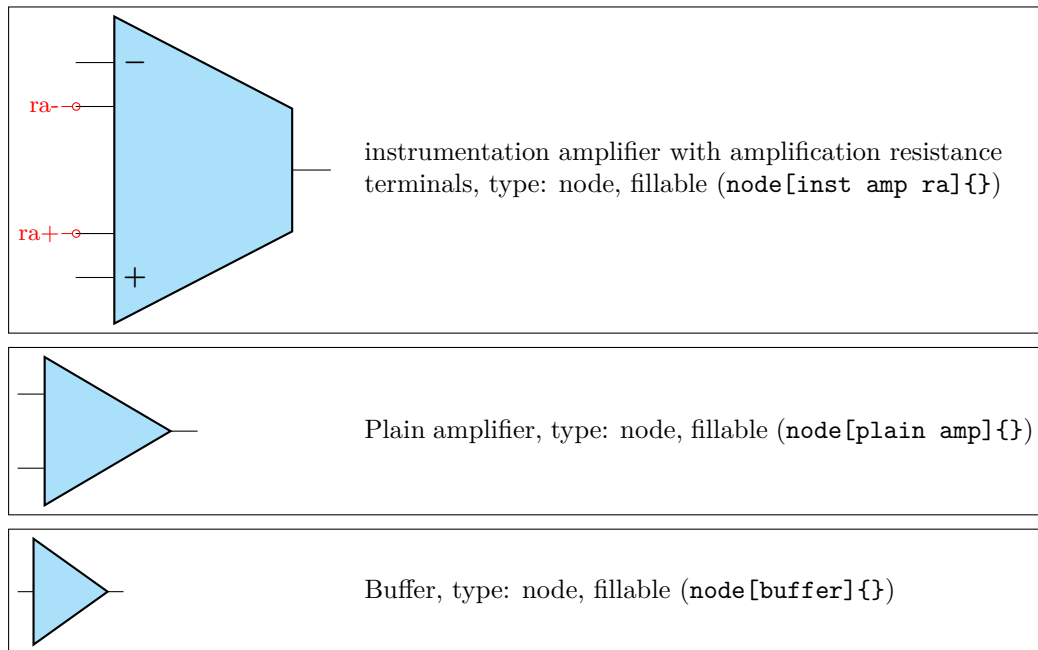


plain instrumentation amplifier, type: node, fillable (node[inst amp]{})



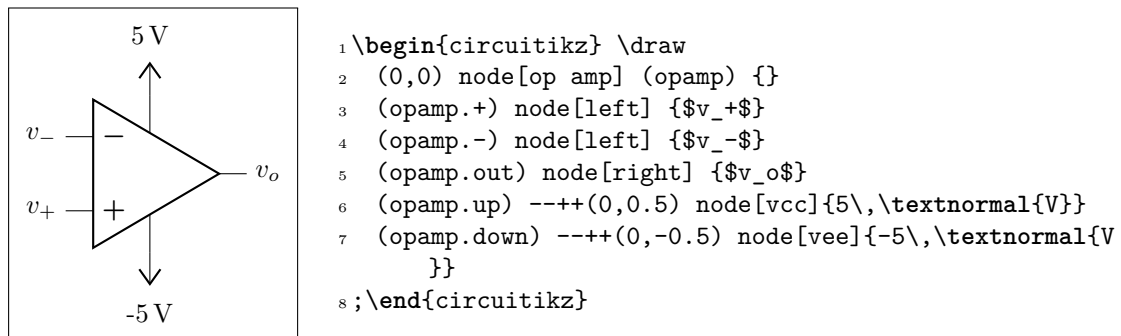
Fully differential instrumentation amplifier, type: node, fillable (node[fd inst amp]{})

¹⁰Contributed by Kristofer M. Monisit.

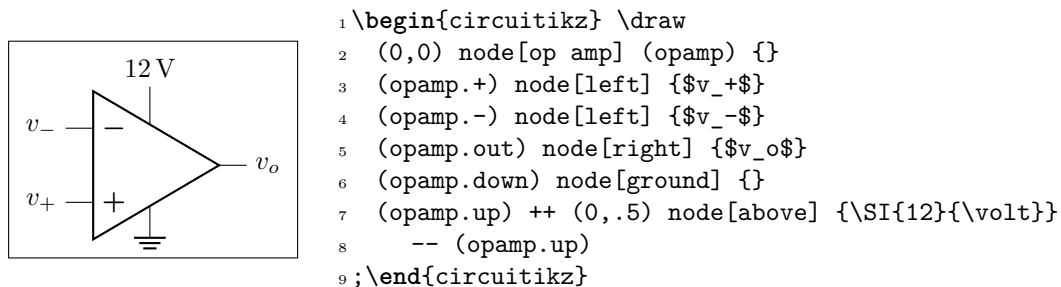


3.23.1 Amplifiers anchors

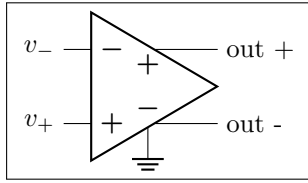
The op amp defines the inverting input (-), the non-inverting input (+) and the output (out) anchors:



There are also two more anchors defined, up and down, for the power supplies:



The fully differential op amp defines two outputs:

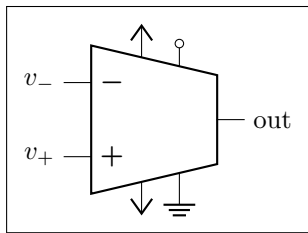


```

1 \begin{circuitikz} \draw
2   (0,0) node[fd op amp] (opamp) {}
3   (opamp.+) node[left] {$v_+$}
4   (opamp.-) node[left] {$v_-$}
5   (opamp.out +) node[right] {out +}
6   (opamp.out -) node[right] {out -}
7   (opamp.down) node[ground] {}
8 ;\end{circuitikz}

```

The instrumentation amplifier inst amp defines also references (normally you use the "down", unless you are flipping the component):

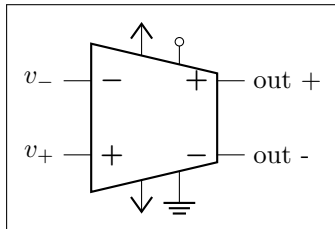


```

1 \begin{circuitikz} \draw
2   (0,0) node[inst amp] (opamp) {}
3   (opamp.+) node[left] {$v_+$}
4   (opamp.-) node[left] {$v_-$}
5   (opamp.out) node[right] {out}
6   (opamp.up) node[vcc]{}
7   (opamp.down) node[vee] {}
8   (opamp.refv down) node[ground]{}
9   (opamp.refv up) to[short, -o] ++(0,0.3)
10 ;\end{circuitikz}

```

The fully differential instrumentation amplifier inst amp defines two outputs:

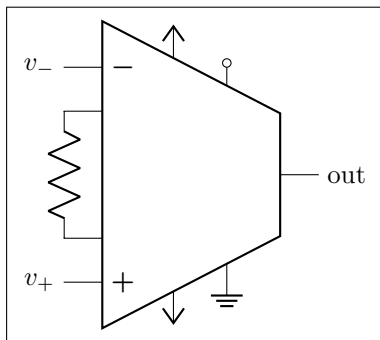


```

1 \begin{circuitikz} \draw
2   (0,0) node[fd inst amp] (opamp) {}
3   (opamp.+) node[left] {$v_+$}
4   (opamp.-) node[left] {$v_-$}
5   (opamp.out +) node[right] {out +}
6   (opamp.out -) node[right] {out -}
7   (opamp.up) node[vcc]{}
8   (opamp.down) node[vee] {}
9   (opamp.refv down) node[ground]{}
10  (opamp.refv up) to[short, -o] ++(0,0.3)
11 ;\end{circuitikz}

```

The instrumentation amplifier with resistance terminals (inst amp ra) defines also terminals to add an amplification resistor:



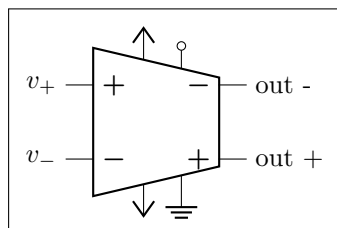
```

1 \begin{circuitikz} \draw
2   (0,0) node[inst amp ra] (opamp) {}
3   (opamp.+) node[left] {$v_+$}
4   (opamp.-) node[left] {$v_-$}
5   (opamp.out) node[right] {out}
6   (opamp.up) node[vcc]{}
7   (opamp.down) node[vee] {}
8   (opamp.refv down) node[ground]{}
9   (opamp.refv up) to[short, -o] ++(0,0.3)
10  (opamp.ra-) to[R] (opamp.ra+)
11 ;\end{circuitikz}

```

3.23.2 Amplifiers customization

All these amplifier have the possibility to flip input and output (if needed) polarity. You can change polarity of the input with the `noinv input down` (default) or `noinv input up` key; and the output with `noinv output up` (default) or `noinv output down` key:



```

1 \begin{circuitikz} \draw
2   (0,0) node[fd inst amp,
3     noinv input up,
4     noinv output down] (opamp) {}
5   (opamp.+) node[left] {$v_+$}
6   (opamp.-) node[left] {$v_-$}
7   (opamp.out +) node[right] {out +}
8   (opamp.out -) node[right] {out -}
9   (opamp.up) node[vcc]{}
10  (opamp.down) node[vee] {}
11  (opamp.refv down) node[ground]{}
12  (opamp.refv up) to[short, -o] ++(0,0.3)
13 ;\end{circuitikz}

```

When you use the `noinv input/output ...` keys the anchors (+, -, out +, out -) will change with the effective position of the terminals. You have also the anchors `in up`, `in down`, `out up`, `out down` that will not change with the positive or negative sign.

3.24 Support shapes and bipoles

Path style:



crossing: Jumper style non-contact crossing, **type:** `path-style`, **nodename:** `crossingshape`. Aliases: `xing`.

Node style:



Arrows (current and voltage), **type:** `node` (`node[currarrow]{}`)



Arrow to draw at its tip, useful for block diagrams., **type:** `node` (`node[inputarrow]{}`)



Connected terminal, **type:** `node` (`node[circ]{}`)



Unconnected terminal, **type:** `node` (`node[ocirc]{}`)



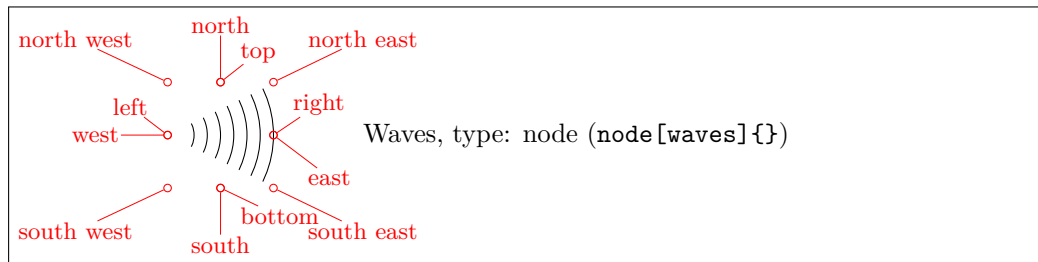
Diamond-style terminal, **type:** `node` (`node[diamondpole]{}`)



Jumper-style crossing node, **type:** `node` (`node[jump crossing]{}`)



Plain style crossing node, **type:** `node` (`node[plain crossing]{}`)



3.24.1 Terminal shapes

Since version 0.8.4, `circ`, `ocirc`, and `diamondpole` have all the standard geographical anchors, so you can do things like these:

```

1 \begin{circuitikz}[american,]
2   \draw (0,-1) node[draw] (R){R};
3   \draw (R.east) node[ocirc, right]{};
4 \end{circuitikz}

```

3.24.2 Crossings

All circuit-drawing standards agree that to show a crossing without electric contact, a simple crossing of the wires suffices; the electrical contact must be explicitly marked with a filled dot.

```

1 \begin{circuitikz}[]
2 \draw(1,-1) to[short] (1,1)
3   (0,0) to[short] (2,0);
4 \draw(4,-1) to[short] (4,1)
5   (3,0) to[short] (5,0)
6   (4,0) node[circ]{};
7 \end{circuitikz}

```

However, sometime it is advisable to mark the non-contact situation more explicitly. To this end, you can use a path-style component called `crossing`:

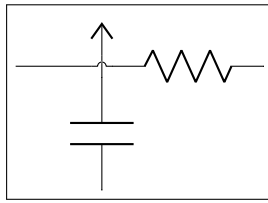
```

1 \begin{circuitikz}[]
2 \draw(1,-1) to[short] (1,1) (0,0) to[crossing]
3   (2,0);
4 \draw(4,-1) to[short] (4,1) (3,0) to[short]
5   (5,0)
6   (4,0) node[circ]{};
7 \end{circuitikz}

```

That should suffice most of the time; the only problem is that the crossing jumper will be put in the center of the subpath where the `to[crossing]` is issued, so sometime a bit of trials and errors are needed to position it.

For a more powerful (and elegant) way you can use the crossing nodes:



```

1 \begin{circuitikz}[]
2   \node at (1,1)[jump crossing](X){};
3   \draw (X.west) -- ++(-1,0);
4   \draw (X.east) to[R] ++(2,0);
5   \draw (X.north) node[vcc]{};
6   \draw (X.south) to[C] ++(0,-1.5);
7 \end{circuitikz}

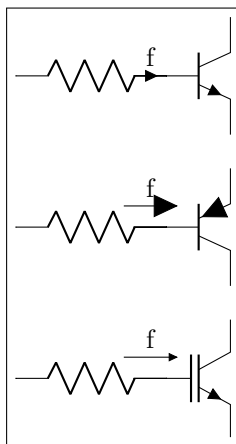
```

Notice that the **plain crossing** and the **jump crossing** have a small gap in the straight wire, to enhance the effect of crossing (as a kind of shadow).

The size of the crossing elements can be changed with the key **bipoles/crossing/size** (default 0.2).

3.24.3 Arrows size

You can use the parameter **current arrow scale** to change the size of the arrows in various components and indicators; the normal value is 16, higher numbers give smaller arrows and so on. You need to use **circuitikz/current arrow scale** if you use it into a node.



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, i=f] ++(2,0) node[npn, anchor=B]{};
3   \draw (0,-2) to[R, f=f, current arrow scale=8] ++(2,0)
4     node[pnp, anchor=B, circuitikz/current arrow scale
5       =8]{};
6   \draw (0,-4) to[R, f=f, current arrow scale=24] ++(2,0)
7     node[nigt, anchor=B]{};
8 \end{circuitikz}

```

3.25 Switches and buttons

Switches and button come in to-style (the simple ones and the pushbuttons), and as nodes.

3.25.1 Traditional switches

These are all of the to-style type:

	switch: Switch, type: path-style , nodename: cspstshape . Aliases: spst .
	closing switch: Closing switch, type: path-style , nodename: cspstshape . Aliases: cspst .
	opening switch: Opening switch, type: path-style , nodename: ospstshape . Aliases: ospst .
	normal open switch: Normally open switch, type: path-style , nodename: nosshape . Aliases: nos .

	normal closed switch: Normally closed switch, type: path-style, nodename: ncsshape. Aliases: ncs.
	push button: Normally open push button, type: path-style, nodename: pushbuttonshape. Aliases: normally open push button, nopb.
	normally closed push button: Normally closed push button, type: path-style, nodename: ncpushbuttonshape. Aliases: ncpb.
	toggle switch: Toggle switch, type: path-style, nodename: toggleswitchshape.

While this is a node-style component

	spdt, type: node (node[spdt]{})
--	---------------------------------

3.25.2 Cute switches




These switches have been introduced after version 0.8.4, and they come in also in to-style and in node-style, but they are size-matched so that they can be used together in a seamless way.

The path element (to-style) are:

	cute closed switch: Cute closed switch, type: path-style, nodename: cuteclosedswitchshape. Aliases: ccsw.
	cute open switch: Cute open switch, type: path-style, nodename: cuteopenswitchshape. Aliases: cosw.
	cute closing switch: Cute closing switch, type: path-style, nodename: cuteclosingswitchshape. Aliases: ccgsw.
	cute opening switch: Cute opening switch, type: path-style, nodename: cuteopeningswitchshape. Aliases: cogsw.

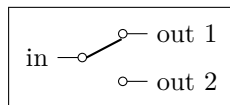
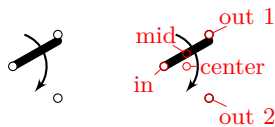
while the node-style components are the single-pole, double-throw (spdt) ones:

	Cute spdt up, type: node (node[cute spdt up]{})
	Cute spdt mid, type: node (node[cute spdt mid]{})
	Cute spdt down, type: node (node[cute spdt down]{})

	Cute spdt up with arrow, type: node (node[cute spdt up arrow]{})
	Cute spdt mid with arrow, type: node (node[cute spdt mid arrow]{})
	Cute spdt down with arrow, type: node (node[cute spdt down arrow]{})

3.25.3 Switches anchors

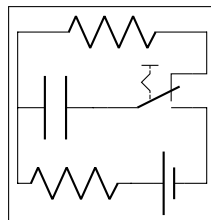
The nodes-style switches have the following anchors:



```

1 \begin{circuitikz} \draw
2 (0,0) node[spdt] (Sw) {}
3 (Sw.in) node[left] {in}
4 (Sw.out 1) node[right] {out 1}
5 (Sw.out 2) node[right] {out 2}
6 ;\end{circuitikz}

```

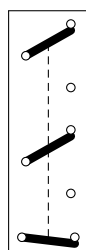


```

1 \begin{circuitikz} \draw
2 (0,0) to[C] (1,0) to[toggle switch , n=Sw] (2.5,0)
3 -- (2.5,-1) to[battery1] (1.5,-1) to[R] (0,-1) -| (0,0)
4 (Sw.out 2) -| (2.5, 1) to[R] (0,1) -- (0,0)
5 ;\end{circuitikz}

```

The `mid` anchor in the cute switches (both path- and node-style) can be used to combine switches to get more complex configurations:



```

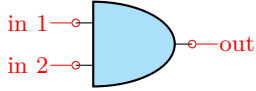
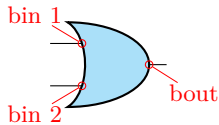
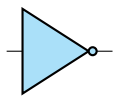
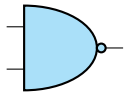
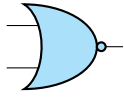
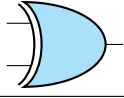
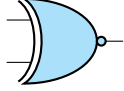
1 \begin{circuitikz}
2 \draw (0,1.4) node[cute spdt up](S1){};
3 \draw (0,0) node[cute spdt up](S2){};
4 \draw (0,-1) node[cuteclosedswitchshape, yscale=-1](S3){};
5 \draw [densely dashed] (S1.mid)--(S2.mid)--(S3.mid);
6 \end{circuitikz}

```


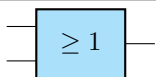

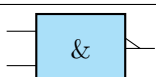
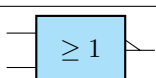
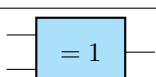
3.26 Logic gates

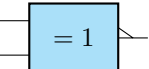
Logic gates, with two or more input, are supported. Albeit in principle these components are multipoles, they are considered tripoles here, for historical reasons (when they just had two inputs).

3.26.1 American Logic gates

	American AND port, type: node, fillable (node[american and port]{})
	American OR port, type: node, fillable (node[american or port]{})
	American NOT port, type: node, fillable (node[american not port]{})
	American NAND port, type: node, fillable (node[american nand port]{})
	American NOR port, type: node, fillable (node[american nor port]{})
	American XOR port, type: node, fillable (node[american xor port]{})
	American XNOR port, type: node, fillable (node[american xnor port]{})

3.26.2 European Logic gates

	European AND port, type: node, fillable (node[european and port]{})
	European OR port, type: node, fillable (node[european or port]{})
	European NOT port, type: node, fillable (node[european not port]{})
	European NAND port, type: node, fillable (node[european nand port]{})
	European NOR port, type: node, fillable (node[european nor port]{})
	European XOR port, type: node, fillable (node[european xor port]{})

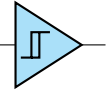
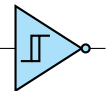
	European XNOR port, type: node, fillable (<code>node[european xnor port]{}</code>)
---	---

If (default behaviour) `americanports` option is active (or the style `[american ports]` is used), the shorthands `and port`, `or port`, `not port`, `nand port`, `nor port`, `xor port`, and `xnor port` are equivalent to the american version of the respective logic port.

If otherwise `europ ports` option is active (or the style `[european ports]` is used), the shorthands `and port`, `or port`, `not port`, `nand port`, `nor port`, `xor port`, and `xnor port` are equivalent to the european version of the respective logic port.

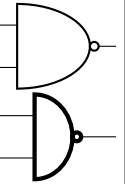
3.26.3 Special components

There is no “european” version of these symbols.

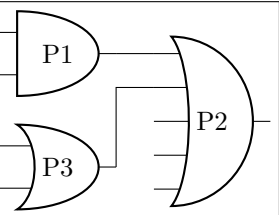
	Non-Inverting SCHMITTTRIGGER, type: node, fillable (<code>node[schmitt]{}</code>)
	Inverting SCHMITTTRIGGER, type: node, fillable (<code>node[invschmitt]{}</code>)

3.26.4 Logic port customization

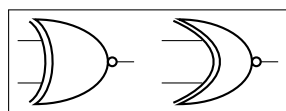
As for most components, you can change the width and height of the ports; the thickness is given by the parameter `tripoles/thickness` (default 2):

	<pre> 1\tikz \draw (0,0) node[nand port] {}; \par 2\ctikzset{tripoles/american nand port/input height=.2} 3\ctikzset{tripoles/american nand port/port width=.4} 4\ctikzset{tripoles/thickness=4} 5\tikz \draw (0,0) node[nand port] {};</pre>
---	---

This is especially useful if you have ports with more than two inputs, which are instantiated with the parameter `number inputs` :

	<pre> 1\begin{circuitikz} 2\draw (0,3) node[american and port] (A) {P1}; 3\begin{scope} 4\ctikzset{tripoles/american or port/height=1.6} 5\draw (A.out) -- ++(0.5,0) 6node[american or port, 7number inputs=5, 8anchor=in 1] (B) {P2}; 9\end{scope} 10\draw (0,1.5) node[american or port] (C) {P3}; 11\draw (C.out) - (B.in 2); 12\end{circuitikz}</pre>
---	--

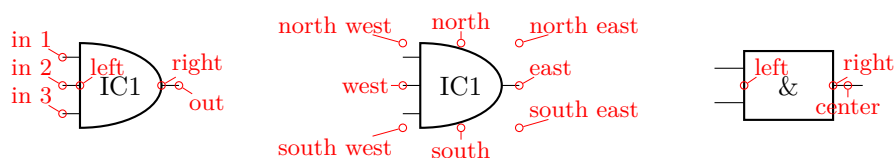
You can tweak the appearance of american “or” family (`or`, `nor`, `xor` and `xnor`) ports, too, with the parameters `inner` (how much the base circle go “into” the shape, default 0.3) and `angle` (the angle at which the base starts, default 70).



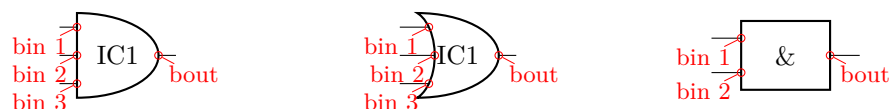
```
1 \tikz \draw (0,0) node[xnor port] {};
2 \ctikzset{tripoles/american xnor port/inner=.7}
3 \ctikzset{tripoles/american xnor port/angle=40}
4 \tikz \draw (0,0) node[xnor port] {};
```

3.26.5 Logic port anchors

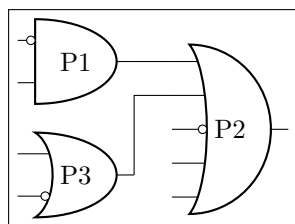
These are the anchors for logic ports:



You have also “border pin anchors”:

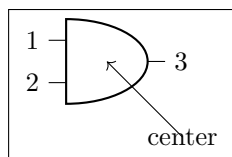


These anchors are especially useful if you want to negate inputs:

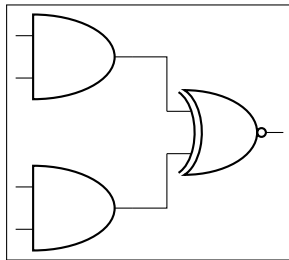


```
1 \begin{circuitikz}
2 \draw (0,3) node[american and port] (A) {P1};
3 \node at (A.bin 1) [ocirc, left]{} ;
4 \begin{scope}
5 \ctikzset{tripoles/american or port/height=1.6}
6 \draw (A.out) -- ++(0.5,0) node[american or port,
7 number inputs=5, anchor=in 1] (B) {P2};
8 \node at (B.bin 3) [ocirc, left]{} ;
9 \end{scope}
10 \draw (0,1.5) node[american or port] (C) {P3};
11 \node at (C.bin 2) [ocirc, left]{} ;
12 \draw (C.out) |- (B.in 2);
13 \end{circuitikz}
```

As you can see, the `center` anchor is (for historic reasons) not in the center at all. You can fix this with the command `\ctikzset{logic ports origin=center}`:



```
1 \begin{circuitikz}
2 \ctikzset{logic ports origin=center}
3 \draw (0,0) node[and port] (myand) {}
4 (myand.in 1) node[anchor=east] {1}
5 (myand.in 2) node[anchor=east] {2}
6 (myand.out) node[anchor=west] {3};
7 \draw[<-] (myand.center) -- ++(1,-1)
8 node{center};
9 \end{circuitikz}
```

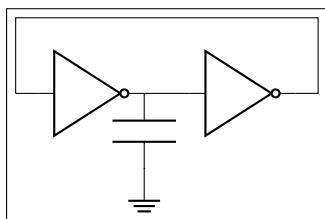


```

1 \begin{circuitikz} \draw
2   (0,2) node[and port] (myand1) {}
3   (0,0) node[and port] (myand2) {}
4   (2,1) node[xnor port] (myxnor) {}
5   (myand1.out) -| (myxnor.in 1)
6   (myand2.out) -| (myxnor.in 2)
7 ;\end{circuitikz}

```

In the case of NOT, there are only `in` and `out` (although for compatibility reasons `in 1` is still defined and equal to `in`):



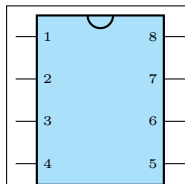
```

1 \begin{circuitikz} \draw
2   (1,0) node[not port] (not1) {}
3   (3,0) node[not port] (not2) {}
4   (0,0) -- (not1.in)
5   (not2.in) -- (not1.out)
6   ++(0,-1) node[ground] {} to[C] (not1.out)
7   (not2.out) -| (4,1) -| (0,0)
8 ;\end{circuitikz}

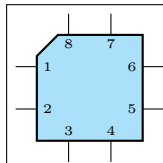
```

3.27 Chips

CircuitikZ supports two types of variable-pin chips: DIP (Dual-in-Line Package) and QFP (Quad-Flat Package).



Dual-in-Line Package chip, type: node, fillable
(`node[dipchip] {}`)



Quad-Flat Package chip, type: node, fillable
(`node[qfpchip] {}`)

3.27.1 DIP and QFP chips customization

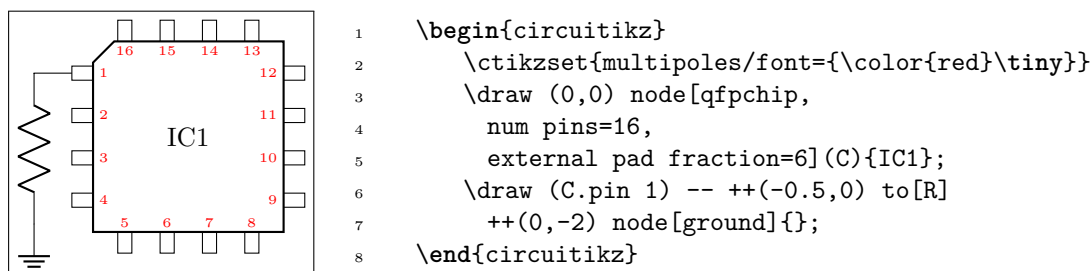
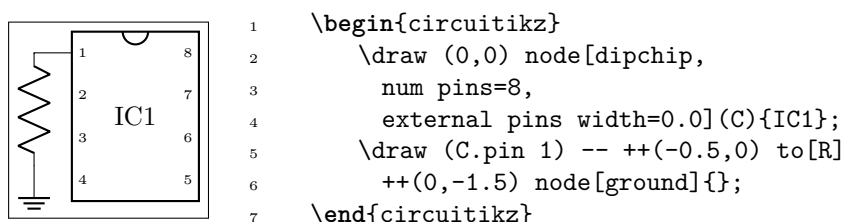
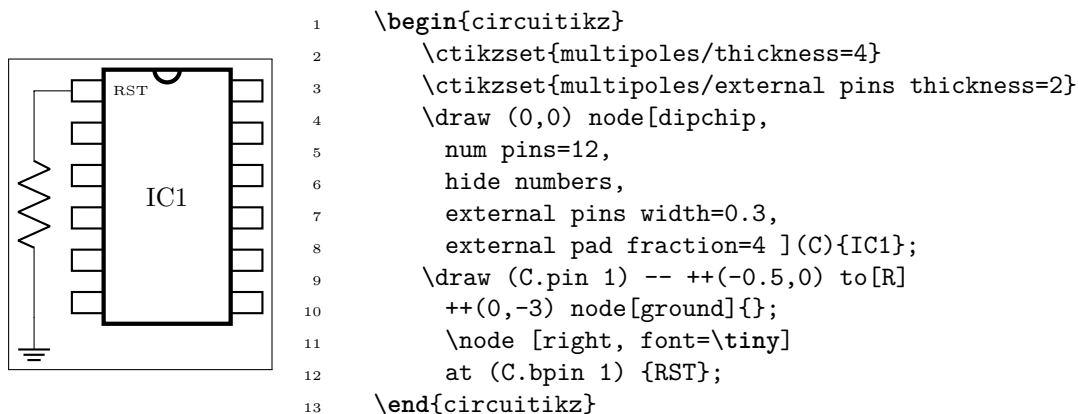
You can customize the DIP chip with the keys `multipoles/dipchip/width` (default 1.2) and `multipoles/dipchip/pin spacing` (default 0.4) that are expressed in fraction of basic lengths (see section 3.1.2). The height of the chip will be equal to half the numbers of pins multiplied by the spacing, plus one spacing for the borders. For the QFP chips, you can only chose the pin spacing with `multipoles/qfpchip/pin spacing` key.

The pins of the chip can be “hidden” (that is, just a spot in the border, optionally marked with a number) or “stick out” with a thin lead by setting `multipoles/external pins width` greater than 0 (default value is 0.2, so you’ll have leads as shown above). Moreover, you can transform the thin lead into a pad by setting the key `multipoles/external pad fraction` to something different from 0 (default is 0); the value expresses the fraction of the pin spacing space that the pad will use on both sides of the pin.

The number of pins is settable with the key `num pins`. **Please notice** that the number of pins **must** be *even* for `dipchips` and *multiple of 4* for `qfpchips`, otherwise havoc will ensue.

You can, if you want, avoid printing the numbers of the pin with `hide numbers` (default `show numbers`) if you prefer positioning them yourself (see the next section for the anchors you can use). The font used for the pins is adjustable with the key `multipoles/font` (default `\tiny`) For special use you can suppress the orientation mark with the key `no topmark` (default `topmark`).

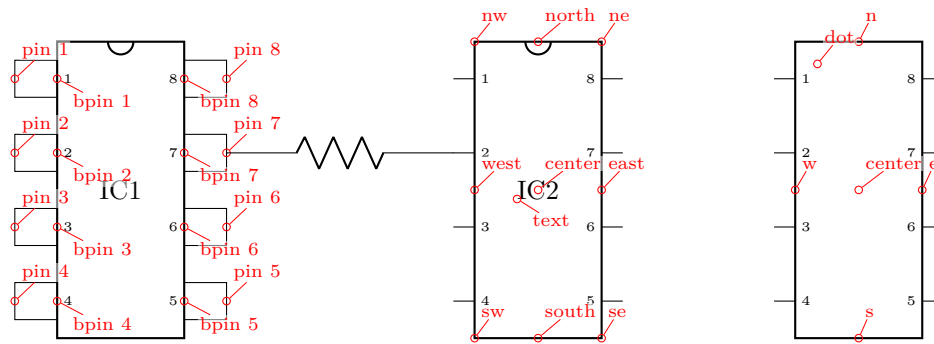
The line thickness of the main shape is controlled by `multipoles/thickness` (default 2) and the one of the external pins/pads with `multipoles/external pins thickness` (default 1).



3.27.2 Chips anchors

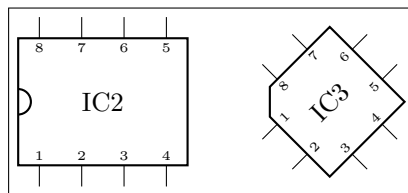
Chips have anchors on pins and global anchors for the main shape. The pin anchors to be used to connect wires to the chip are called `pin 1`, `pin 2`, ..., with just one space between `pin` and the number. Border pin anchors (`bpin 1...`) are always on the box border, and can be used to add numbers or whatever markings are needed. Obviously, in case of `multipoles/external pins width` equal to zero, border and normal pin anchors will coincide.

Additionally, you have geometrical anchors on the chip “box”, see the following figure. The nodes are available with the full name (like `north`) and with the short abbreviations `n`, `nw`, `w`, The `dot` anchor is useful to add a personalized marker if you use the `no topmark` key.



3.27.3 Chips rotation

You can rotate chips, and normally the pin numbers are kept straight (option `straight numbers`, which is the default), but you can rotate them if you like with `rotated numbers`. Notice that the main label has to be (counter-)rotated manually in this case.



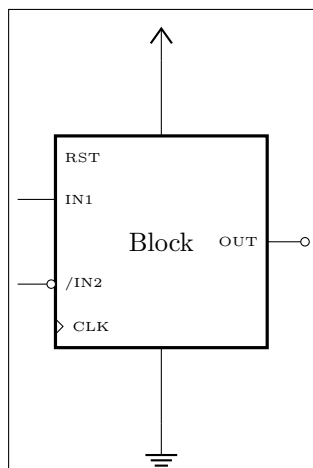
```

1 \begin{circuitikz}
2   \draw (0,0) node[dipchip,
3     rotate=90]{%
4       \rotatebox{-90}{IC2}};
5   \draw (3,0) node[qfpchip,
6     rotated numbers,
7     rotate=45]{IC3};
8 \end{circuitikz}

```

3.27.4 Chip special usage

You can use the chips to have special, personalized blocks. Look at the following example, which is easily put into a macro.

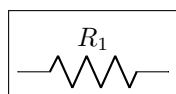


```

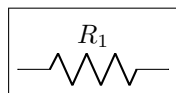
1 \begin{circuitikz}
2   \ctikzset{multipoles/thickness=3}
3   \ctikzset{multipoles/dipchip/width=2}
4   \draw (0,0) node[dipchip,
5     num pins=10, hide numbers, no topmark,
6     external pins width=0](C){Block};
7   \node [right, font=\tiny] at (C.bpin 1) {RST};
8   \node [right, font=\tiny] at (C.bpin 2) {IN1};
9   \node [right, font=\tiny] at (C.bpin 4) {/IN2};
10  \node [left, font=\tiny] at (C.bpin 8) {OUT};
11  \draw (C.bpin 2) -- ++(-0.5,0) coordinate(extpin);
12  \node [ocirc, anchor=0](notin2) at (C.bpin 4) {};
13  \draw (notin2.180) -- (C.bpin 4 -| extpin);
14  \draw (C.bpin 8) to[short,-o] ++(0.5,0);
15  \draw (C.bpin 5) ++(0,0.1) -- ++(0.1,-0.1)
16    node[right, font=\tiny]{CLK} -- ++(-0.1,-0.1);
17  \draw (C.n) -- ++(0,1) node[vcc]{};
18  \draw (C.s) -- ++(0,-1) node[ground]{};
19 \end{circuitikz}

```

4 Labels and similar annotations



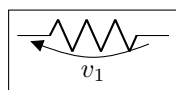
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, l=$R_1$] (2,0);
3 \end{circuitikz}
```



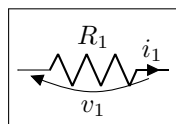
```
1 \begin{circuitikz}
2   \draw (0,0) to[R=$R_1$] (2,0);
3 \end{circuitikz}
```



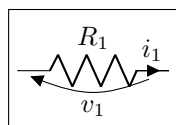
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i=$i_1$] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, v=$v_1$] (2,0);
3 \end{circuitikz}
```

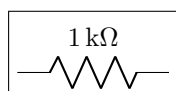


```
1 \begin{circuitikz}
2   \draw (0,0) to[R=$R_1$, i=$i_1$, v=$v_1$] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R=$R_1$, i=$i_1$, v=$v_1$] (2,0);
3 \end{circuitikz}
```

Long names/styles for the bipoles can be used:

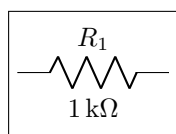


```
1 \begin{circuitikz}\draw
2   (0,0) to[resistor=1<\kilo\ohm>] (2,0)
3 ;\end{circuitikz}
```

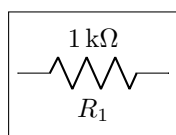
4.1 Labels and Annotations

Since Version 0.7, beside the original label (l) option, there is a new option to place a second label, called annotation (a) at each bipole. Up to now this is a beta-test and there can be problems. For example, up to now this option is not compatible with the concurrent use of voltage labels.

The position of (a) and (l) labels can be adjusted with `_` and `^`, respectively.

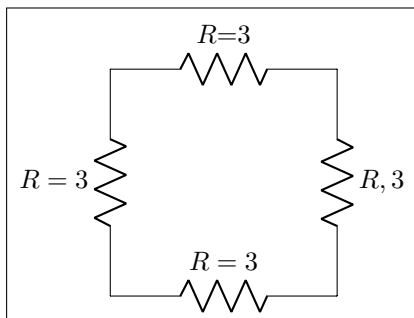


```
1 \begin{circuitikz}
2   \draw (0,0) to[R, l=$R_1$,a=1<\kilo\ohm>] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, l_=$R_1$,a^=1<\kilo\ohm>] (2,0);
3 \end{circuitikz}
```

Caveat: notice that the way in which `circuitikz` processes the options, there will be problems if the label (or annotation, or voltage, or current) contains one of the characters `=` (equal) or `,` (comma), giving unexpected errors and wrong output. These two characters must be protected to the option parser using an `\mbox` command, or redefining the characters with a `TeX \def`:

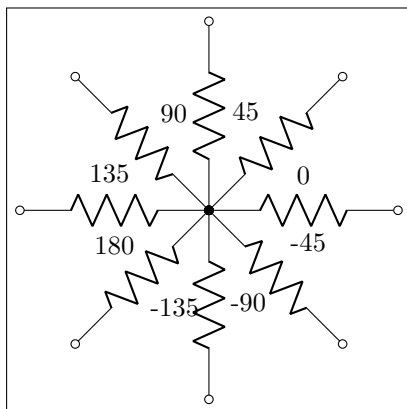


```

1 \def\eq{=}
2 \begin{circuitikz}
3   % the following will fail:
4   % \draw (0,0) to[R, l={\$R=3\$}] (3,0);
5   \draw (0,0) to[R, l=\mbox{\$R=3\$}] (3,0);
6   \draw (0,0) to[R, l=\$R\eq3\$] (0,3);
7   \draw (3,3) to[R, l=\mbox{\$R,3\$}] (3,0);
8   % this works, but it has wrong spacing
9   \draw (0,3) to[R, l=\$R{= }3\$] (3,3);
10 \end{circuitikz}

```

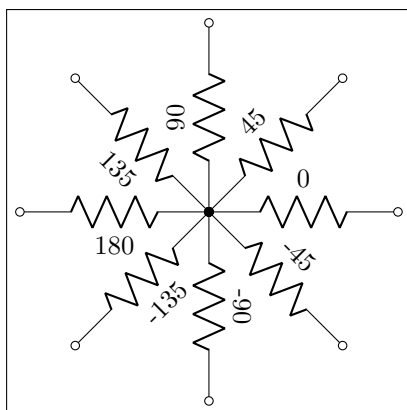
The default orientation of labels is controlled by the options `smartlabels`, `rotatelabels` and `straightlabels` (or the corresponding `label/align` keys). Here are examples to see the differences:



```

1 \begin{circuitikz}
2 \ctikzset{label/align = straight}
3 \def\DIR{0,45,90,135,180,-90,-45,-135}
4 \foreach \i in \DIR {
5   \draw (0,0) to[R=\i, *-o] (\i:2.5);
6 }
7 \end{circuitikz}

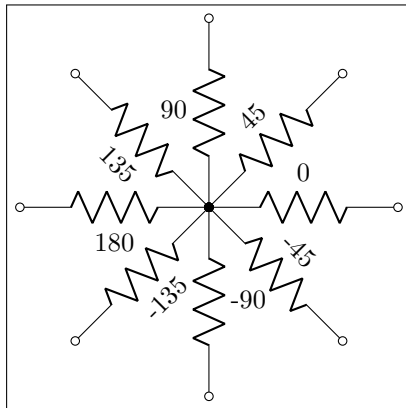
```



```

1 \begin{circuitikz}
2 \ctikzset{label/align = rotate}
3 \def\DIR{0,45,90,135,180,-90,-45,-135}
4 \foreach \i in \DIR {
5   \draw (0,0) to[R=\i, *-o] (\i:2.5);
6 }
7 \end{circuitikz}

```

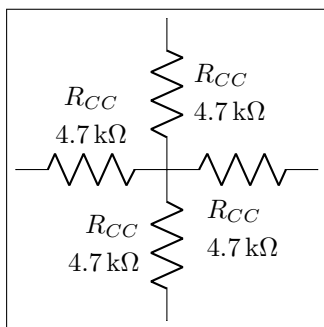



```

1 \begin{circuitikz}
2 \ctikzset{label/align = smart}
3 \def\DIR{0,45,90,135,180,-90,-45,-135}
4 \foreach \i in \DIR {
5   \draw (0,0) to[R=\i, *-o] (\i:2.5);
6 }
7 \end{circuitikz}

```

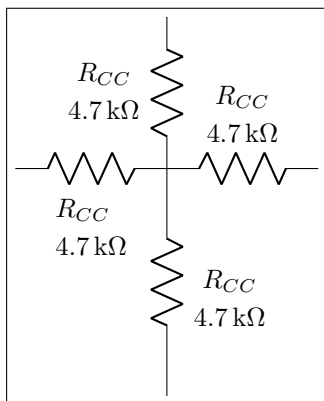
You also can use stacked (two lines) labels. The example should be self-explanatory: the two lines are specified as `l2=`*line1* and *line2*. You can use the keys `l2 halign` to control horizontal position (left, center, right) and `l2 valign` to control the vertical one (bottom, ccenter, top).



```

1 \begin{circuitikz}[ american, ]
2   %
3   % default is l2 halign=l, l2 valign=c
4   %
5   \draw (0,0) to[R, l2_=$R_{CC}$ and \SI{4.7}{k\ohm},
6               , l2 valign=t] (2,0);
7   \draw (0,0) to[R, l2_=$R_{CC}$ and \SI{4.7}{k\ohm},
8               , ] (0,2);
9   \draw (0,0) to[R, l2_=$R_{CC}$ and \SI{4.7}{k\ohm}, l2 halign=c, l2 valign=b] (-2,0);
10  \draw (0,0) to[R, l2_=$R_{CC}$ and \SI{4.7}{k\ohm}, l2 halign=r, l2 valign=c] (0, -2);
11 \end{circuitikz}

```



```

1 \begin{circuitikz}[ american, ]
2   \draw (0,0) to[R, l2^=$R_{CC}$ and \SI{4.7}{k\ohm}, l2 halign=c, l2 valign=b] (2,0);
3   \draw (0,0) to[R, l2^=$R_{CC}$ and \SI{4.7}{k\ohm}, l2 halign=c, ] (0,2);
4   \draw (0,0) to[R, l2^=$R_{CC}$ and \SI{4.7}{k\ohm}, , l2 valign=t] (-2,0);
5   \draw (0,0) to[R, l2^=$R_{CC}$ and \SI{4.7}{k\ohm}, l2 halign=c, l2 valign=t] (0, -3);
6 \end{circuitikz}

```

4.2 Currents and voltages

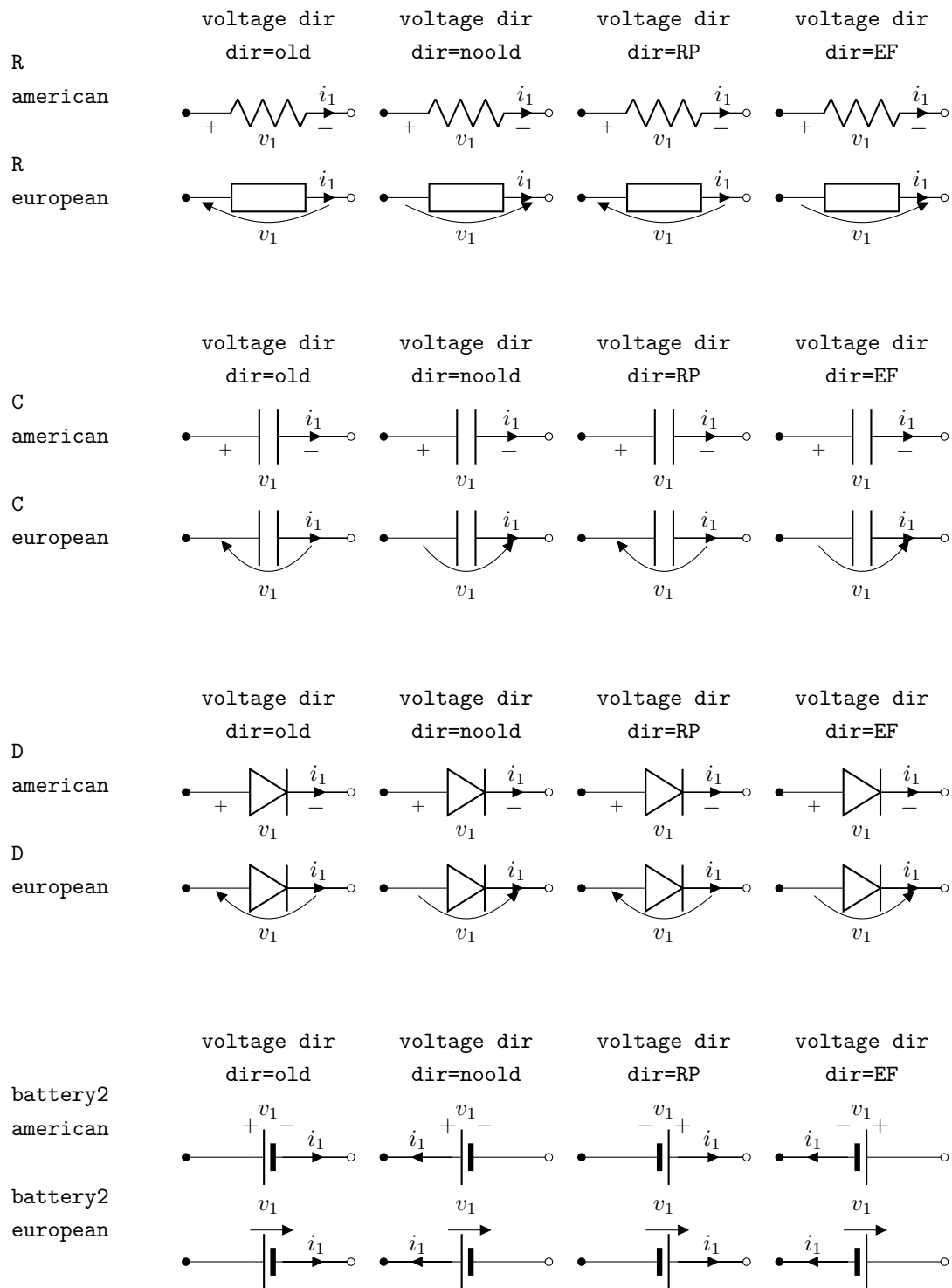
The default direction/sign for currents and voltages in the components is, unfortunately, not standard, and can change across country and sometime across different authors. This unfortunate situation created a bit of confusion in `circuitikz` across the versions, with several incompatible changes starting from version 0.5. From version 0.8.4 onward, the maintainers agreed a new policy for the directions of bipoles's voltages and currents, depending on 4 different possible options:

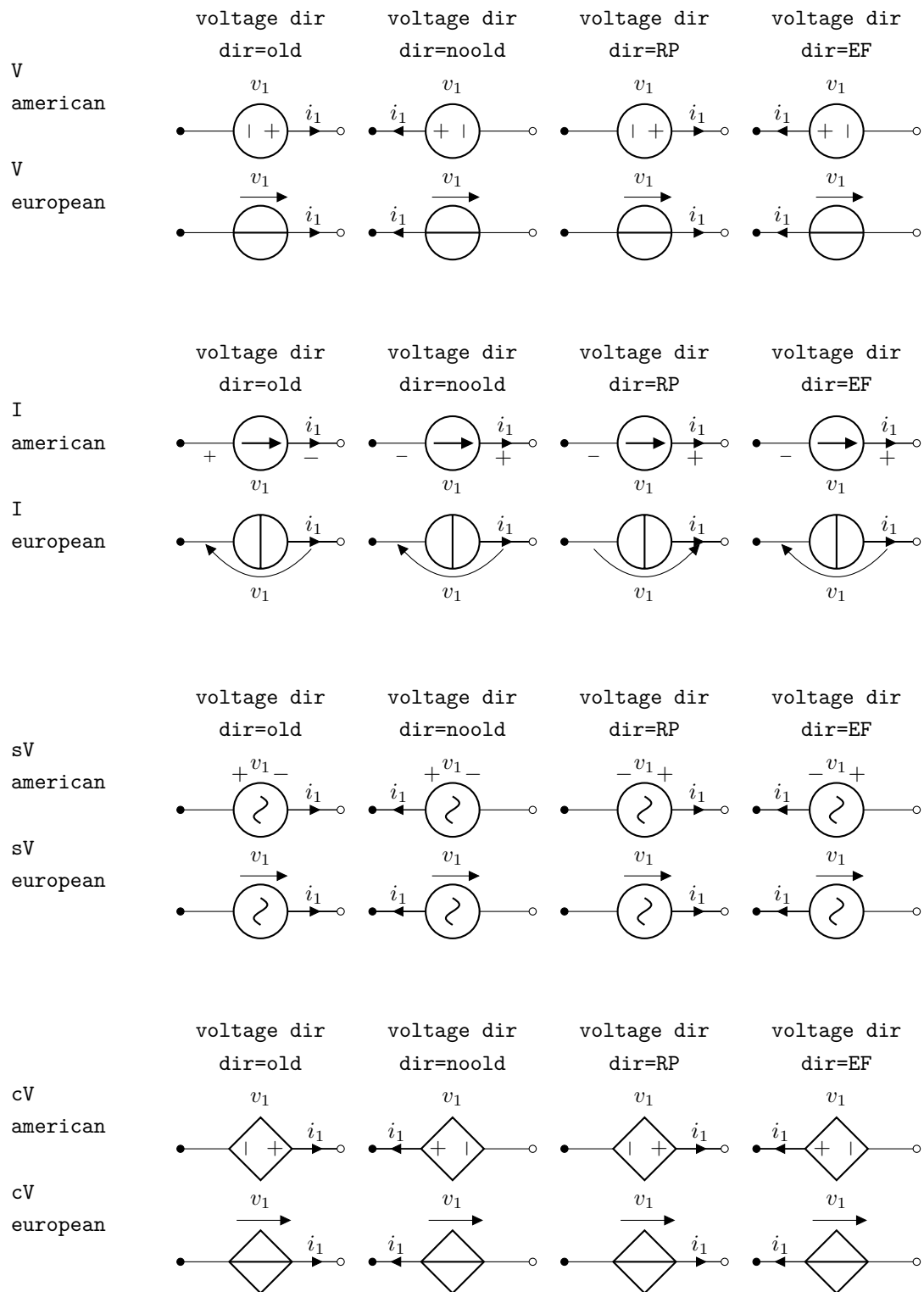
- `oldvoltagedirection`, or the key style `voltage dir=old`: Use old way of voltage direction having a difference between european and american direction, with wrong default labelling for batteries (it was the default before version 0.5);
- `nooldvoltagedirection`, or the key style `voltage dir=noold`: The standard from version 0.5 onward, utilize the (German?) standard of voltage arrows in the direction of electric fields (without fixing batteries);
- `RPvoltages` (meaning Rising Potential voltages), or the key style `voltage dir=RP`: the arrow is in direction of rising potential, like in `oldvoltagedirections`, but batteries and current sources are fixed so that they follow the passive/active standard: the default direction of v and i are chosen so that, when both values are positive:
 - in passive component, the element is *dissipating power*;
 - in active components (generators), the element is *generating power*.
- `EFvoltages` (meaning Electric Field voltages), or the key style `voltage dir=EF`: the arrow is in direction of the electric field, like in `nooldvoltagedirections`, but batteries are fixed;

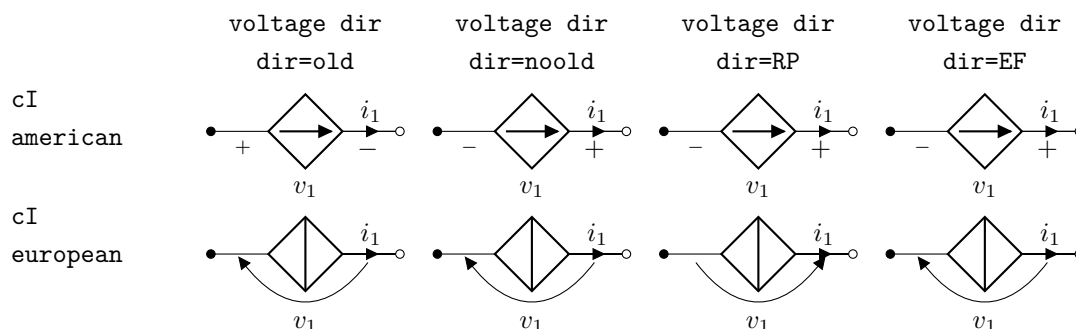
The standard direction of currents, flows and voltages are changed by these options; notice that the default drops in case of passive and active elements is normally different. Take care that in the case of `noold` and `EFvoltages` also the currents can switch directions. It is much easier to understand the several behaviors by looking at the following examples, that have been generated by the code:

```

1 \foreach\element in {R, C, D, battery2, V, I, sV, cV, cI}{%
2   \noindent\ttfamily
3   \begin{tabular}{p{2cm}}
4     \element \\\ american \\\[15pt]
5     \element \\\ european \\\
6   \end{tabular}
7   \foreach\mode in {old, noold, RP, EF} {
8     \begin{tabular}{@{}l@{}}
9       \multicolumn{1}{c}{voltage dir} \\\
10      \multicolumn{1}{c}{dir=\mode} \\\[4pt]
11      \begin{tikzpicture}[
12        american, voltage dir=\mode,
13      ]
14        \draw (0,0) to[\element, *-o, v=$v_1$, i=$i_1$, ] (2.5,0);
15      \end{tikzpicture}\\
16      \begin{tikzpicture}[
17        european, voltage dir=\mode,
18      ]
19        \draw (0,0) to[\element, *-o, v=$v_1$, i=$i_1$, ] (2.5,0);
20      \end{tikzpicture}
21    \end{tabular}
22    \medskip
23  }
24 \par
25 }
```







Obviously, you normally use just one between current and flows, but anyway you can change direction of the voltages, currents and flows using the complete keys $i_>$, $i^<$, $i>_>$, $i^>$, as shown in the following examples.

This manual has been typeset with the option `RPvoltages`.

4.3 Currents

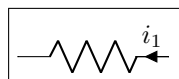
Inline (along the wire) currents are selected with $i_>$, $i^<$, $i>_>$, $i^>$, and various simplification; the default position and direction is obtained with the key `i=...`



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i^>=$i_1$] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i_>=$i_1$] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i^<=$i_1$] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i_<=$i_1$] (2,0);
3 \end{circuitikz}
```



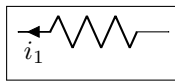
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i^>=$i_1$] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i_>=$i_1$] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i^<=$i_1$] (2,0);
3 \end{circuitikz}
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, i<_=$i_1$] (2,0);
3 \end{circuitikz}

```

Also



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, i<_=$i_1$] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, i>_=$i_1$] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, i^_=$i_1$] (2,0);
3 \end{circuitikz}

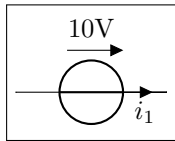
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, i_=$i_1$] (2,0);
3 \end{circuitikz}

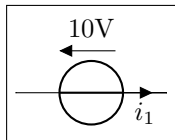
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[V=10V, i_=$i_1$] (2,0);
3 \end{circuitikz}

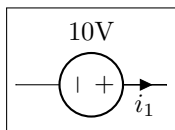
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[V<=10V, i_=$i_1$] (2,0);
3 \end{circuitikz}

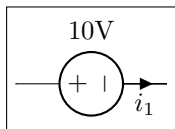
```



```

1 \begin{circuitikz}[american]
2   \draw (0,0) to[V=10V, i_=$i_1$] (2,0);
3 \end{circuitikz}

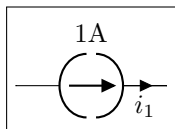
```



```

1 \begin{circuitikz}[american]
2   \draw (0,0) to[V=10V,invert, i_=$i_1$] (2,0);
3 \end{circuitikz}

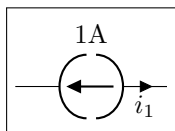
```



```

1 \begin{circuitikz}[american]
2   \draw (0,0) to[dcsource=1A, i_=$i_1$] (2,0);
3 \end{circuitikz}

```



```

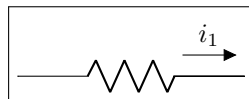
1 \begin{circuitikz}[american]
2   \draw (0,0) to[dcsource=1A,invert, i_=$i_1$] (2,0);
3 \end{circuitikz}

```

4.4 Flows

As an alternative for the current arrows, you can also use the following flows. They can also be used to indicate thermal or power flows. The syntax is pretty the same as for currents.

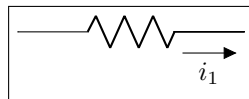
This is a new beta feature since version 0.8.3, therefore, please provide bugreports or hints to optimize this feature regarding placement and appearance! This means, that the appearance may change in the future!



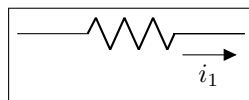
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, f=$i_1$] (3,0);
3 \end{circuitikz}
```



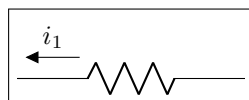
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, f<=$i_1$] (3,0);
3 \end{circuitikz}
```



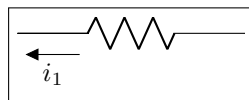
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, f_=$i_1$] (3,0);
3 \end{circuitikz}
```



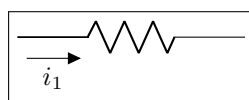
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, f_>=$i_1$] (3,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, f^<=$i_1$] (3,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, f<_=$i_1$] (3,0);
3 \end{circuitikz}
```



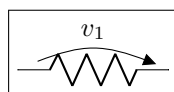
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, f>_=$i_1$] (3,0);
3 \end{circuitikz}
```

4.5 Voltages

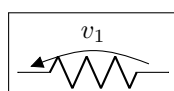
See introduction note at Currents (chapter 4.2, page 62)!

4.5.1 European style

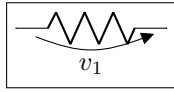
The default, with arrows. Use option `europeanvoltage` or style `[european voltages]`.



```
1 \begin{circuitikz}[european voltages]
2   \draw (0,0) to[R, v^>=$v_1$] (2,0);
3 \end{circuitikz}
```



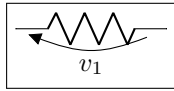
```
1 \begin{circuitikz}[european voltages]
2   \draw (0,0) to[R, v^<=$v_1$] (2,0);
3 \end{circuitikz}
```



```

1 \begin{circuitikz}[european voltages]
2   \draw (0,0) to[R, v_>=$v_1$] (2,0);
3 \end{circuitikz}

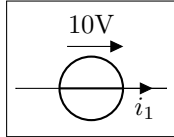
```



```

1 \begin{circuitikz}[european voltages]
2   \draw (0,0) to[R, v_<=$v_1$] (2,0);
3 \end{circuitikz}

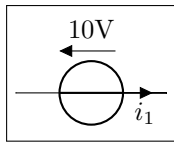
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[V=10V, i_=$i_1$] (2,0);
3 \end{circuitikz}

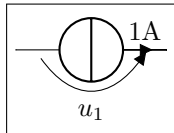
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[V<=10V, i_=$i_1$] (2,0);
3 \end{circuitikz}

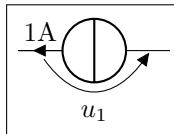
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[I=1A, v_=$u_1$] (2,0);
3 \end{circuitikz}

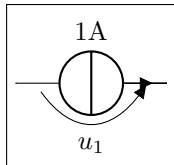
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[I<=1A, v_=$u_1$] (2,0);
3 \end{circuitikz}

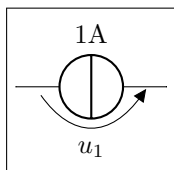
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[I=$\sim$,l=1A, v_=$u_1$] (2,0);
3 \end{circuitikz}

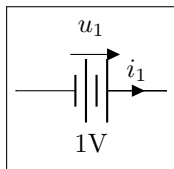
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[I,l=1A, v_=$u_1$] (2,0);
3 \end{circuitikz}

```



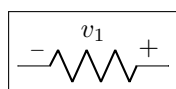
```

1 \begin{circuitikz}
2   \draw (0,0) to[battery,l=1V, v=$u_1$, i=$i_1$] (2,0);
3 \end{circuitikz}

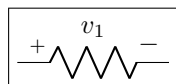
```


4.5.2 American style

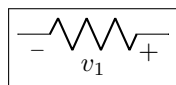
Use option `americanvoltage` or set `[american voltages]`.



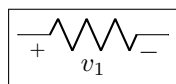
```
1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[R, v^>=$v_1$] (2,0);
3 \end{circuitikz}
```



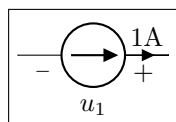
```
1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[R, v^<=$v_1$] (2,0);
3 \end{circuitikz}
```



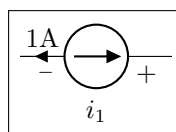
```
1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[R, v_>=$v_1$] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[R, v_<=$v_1$] (2,0);
3 \end{circuitikz}
```



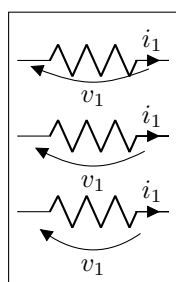
```
1 \begin{circuitikz}[american]
2   \draw (0,0) to[I=1A, v_=$u_1$] (2,0);
3 \end{circuitikz}
```



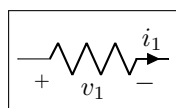
```
1 \begin{circuitikz}[american]
2   \draw (0,0) to[I<=1A, v_=$i_1$] (2,0);
3 \end{circuitikz}
```

4.5.3 Voltage position

It is possible to move away the arrows and the plus or minus signs with the key `voltages shift` (default value is 0, which gives the standard position):

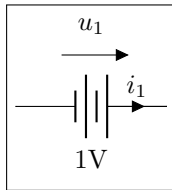


```
1 \begin{circuitikz}[]
2   \draw (0,0) to[R, v=$v_1$, i=$i_1$] (2,0);
3   \draw (0,-1) to[R, v=$v_1$, i=$i_1$,
4     voltage shift=0.5] (2,-1);
5   \draw (0,-2) to[R, v=$v_1$, i=$i_1$,
6     voltage shift=1.0, ] (2,-2);
7 \end{circuitikz}
```



```
1 \begin{circuitikz}[american voltages, voltage shift=0.5]
2   \draw (0,0) to[R, v=$v_1$, i=$i_1$] (2,0);
3 \end{circuitikz}
```

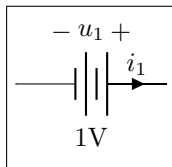
Notes that `american voltage` will not affect batteries.



```

1 \begin{circuitikz}[voltage shift=0.5]
2 \draw (0,0) to[battery,l_=1V, v=$u_1$, i=$i_1$] (2,0);
3 \end{circuitikz}

```

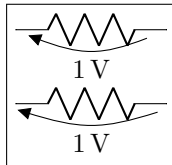


```

1 \begin{circuitikz}[american voltages, voltage shift=0.5]
2 \draw (0,0) to[battery,l_=1V, v=$u_1$, i=$i_1$] (2,0);
3 \end{circuitikz}

```

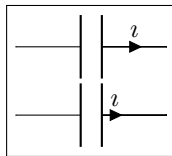
4.5.4 Global properties of voltages and currents



```

1 \tikz \draw (0,0) to[R, v=1<\volt>] (2,0); \par
2 \ctikzset{voltage/distance from node=.1}
3 \tikz \draw (0,0) to[R, v=1<\volt>] (2,0);

```

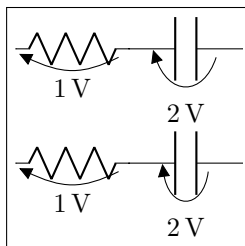


```

1 \tikz \draw (0,0) to[C, i=$\imath$] (2,0); \par
2 \ctikzset{current/distance = .2}
3 \tikz \draw (0,0) to[C, i=$\imath$] (2,0);

```

However, you can override the properties `voltage/distance from node`¹¹, `voltage/bump b`¹² and `voltage/european label distance`¹³ on a per-component basis, in order to fine-tune the voltages:



```

1 \tikz \draw (0,0) to[R, v=1<\volt>] (1.5,0)
2 to[C, v=2<\volt>] (3,0); \par
3 \ctikzset{bipoles/capacitor/voltage/%
4 distance from node/.initial=.7}
5 \tikz \draw (0,0) to[R, v=1<\volt>] (1.5,0)
6 to[C, v=2<\volt>] (3,0); \par

```

4.6 Nodes



```

1 \begin{circuitikz}
2 \draw (0,0) to[R, o-o] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2 \draw (0,0) to[R, -o] (2,0);
3 \end{circuitikz}

```

¹¹That is, how distant from the initial and final points of the path the arrow starts and ends.

¹²Controlling how high the bump of the arrow is — how curved it is.

¹³Controlling how distant from the bipole the voltage label will be.



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, o-] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, *-] (2,0);
3 \end{circuitikz}

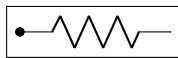
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, -*] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, *-] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, d-d] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, -d] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, d-] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, o-*] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, *-o] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, o-d] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, d-o] (2,0);
3 \end{circuitikz}

```



```

1 \begin{circuitikz}
2   \draw (0,0) to[R, *-d] (2,0);
3 \end{circuitikz}

```



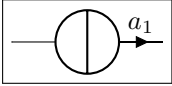
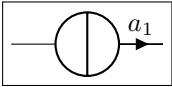
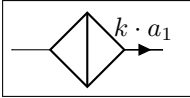
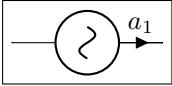
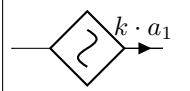
```

1 \begin{circuitikz}
2   \draw (0,0) to[R, d-*] (2,0);
3 \end{circuitikz}

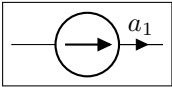
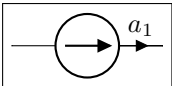
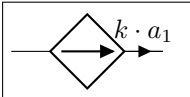
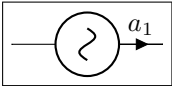
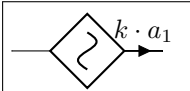
```

4.7 Special components

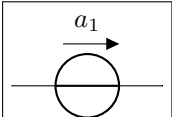
For some components label, current and voltage behave as one would expect:

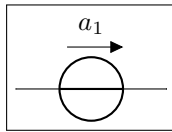
	<pre> 1 \begin{circuitikz} 2 \draw (0,0) to[I=\$a_1\$] (2,0); 3 \end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz} 2 \draw (0,0) to[I, i=\$a_1\$] (2,0); 3 \end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz} 2 \draw (0,0) to[cI=\$k \cdot a_1\$] (2,0); 3 \end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz} 2 \draw (0,0) to[sI=\$a_1\$] (2,0); 3 \end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz} 2 \draw (0,0) to[csI=\$k \cdot a_1\$] (2,0); 3 \end{circuitikz} </pre>

The following results from using the option `americancurrents` or using the style `[americancurrents]`.

	<pre> 1 \begin{circuitikz}[american currents] 2 \draw (0,0) to[I=\$a_1\$] (2,0); 3 \end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz}[american currents] 2 \draw (0,0) to[I, i=\$a_1\$] (2,0); 3 \end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz}[american currents] 2 \draw (0,0) to[cI=\$k \cdot a_1\$] (2,0); 3 \end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz}[american currents] 2 \draw (0,0) to[sI=\$a_1\$] (2,0); 3 \end{circuitikz} </pre>
	<pre> 1 \begin{circuitikz}[american currents] 2 \draw (0,0) to[csI=\$k \cdot a_1\$] (2,0); 3 \end{circuitikz} </pre>

The same holds for voltage sources:

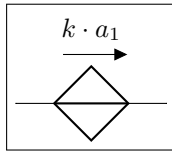
	<pre> 1 \begin{circuitikz} 2 \draw (0,0) to[V=\$a_1\$] (2,0); 3 \end{circuitikz} </pre>
---	---



```

1 \begin{circuitikz}
2   \draw (0,0) to[V, v=$a_1$] (2,0);
3 \end{circuitikz}

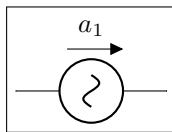
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[cV=$k \cdot a_1$] (2,0);
3 \end{circuitikz}

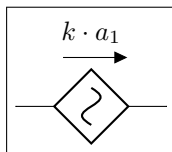
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[sV=$a_1$] (2,0);
3 \end{circuitikz}

```

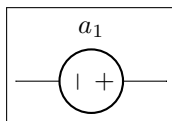


```

1 \begin{circuitikz}
2   \draw (0,0) to[csV=$k \cdot a_1$] (2,0);
3 \end{circuitikz}

```

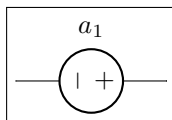
The following results from using the option `americanvoltage` or the style `[american voltages]`.



```

1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[V=$a_1$] (2,0);
3 \end{circuitikz}

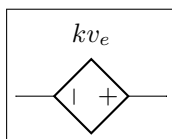
```



```

1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[V, v=$a_1$] (2,0);
3 \end{circuitikz}

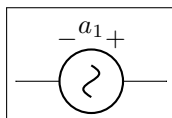
```



```

1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[cV=$k v_e$] (2,0);
3 \end{circuitikz}

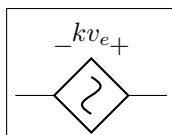
```



```

1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[sV=$a_1$] (2,0);
3 \end{circuitikz}

```



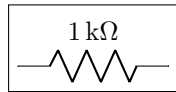
```

1 \begin{circuitikz}[american voltages]
2   \draw (0,0) to[csV=$k v_e$] (2,0);
3 \end{circuitikz}

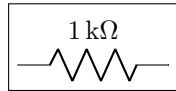
```

4.8 Integration with siunitx

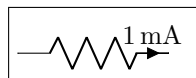
If the option `siunitx` is active (and *not* in ConT_EXt), then the following are equivalent:



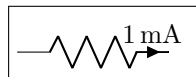
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, l=1<\kilo\ohm>] (2,0);
3 \end{circuitikz}
```



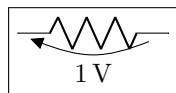
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, l=\SI{1}{\kilo\ohm}] (2,0);
3 \end{circuitikz}
```



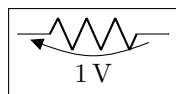
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i=1<\milli\ampere>] (2,0);
3 \end{circuitikz}
```



```
1 \begin{circuitikz}
2   \draw (0,0) to[R, i=\SI{1}{\milli\ampere}] (2,0);
3 \end{circuitikz}
```



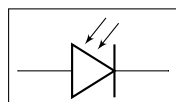
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, v=1<\volt>] (2,0);
3 \end{circuitikz}
```



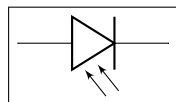
```
1 \begin{circuitikz}
2   \draw (0,0) to[R, v=\SI{1}{\volt}] (2,0);
3 \end{circuitikz}
```

4.9 Mirroring and Inverting

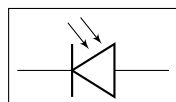
Bipole paths can also mirrored and inverted (or reverted) to change the drawing direction.



```
1 \begin{circuitikz}
2   \draw (0,0) to[pD] (2,0);
3 \end{circuitikz}
```

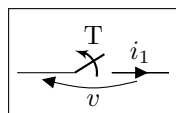


```
1 \begin{circuitikz}
2   \draw (0,0) to[pD, mirror] (2,0);
3 \end{circuitikz}
```

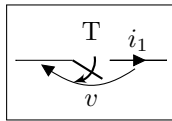


```
1 \begin{circuitikz}
2   \draw (0,0) to[pD, invert] (2,0);
3 \end{circuitikz}
```

Placing labels, currents and voltages works also, please note, that mirroring and inverting does not influence the positioning of labels and voltages. Labels are by default above/right of the bipole and voltages below/left, respectively.



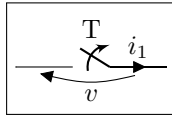
```
1 \begin{circuitikz}
2   \draw (0,0) to[ospst=T, i=\$i_1$, v=\$v$] (2,0);
3 \end{circuitikz}
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[ospst=T, mirror, i=$i_1$, v=$v$] (2,0);
3 \end{circuitikz}

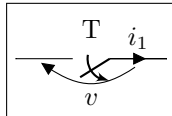
```



```

1 \begin{circuitikz}
2   \draw (0,0) to[ospst=T, invert, i=$i_1$, v=$v$] (2,0);
3 \end{circuitikz}

```

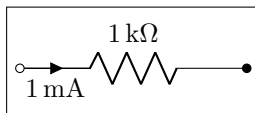


```

1 \begin{circuitikz}
2   \draw (0,0) to[ospst=T,mirror,invert, i=$i_1$, v=$v$] (2,0);
3 \end{circuitikz}

```

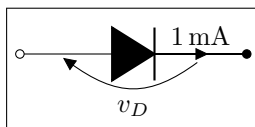
4.10 Putting them together



```

1 \begin{circuitikz}
2   \draw (0,0) to[R=1<\kilo\ohm>,
3     i>_1<\milli\ampere>, o-*] (3,0);
4 \end{circuitikz}

```



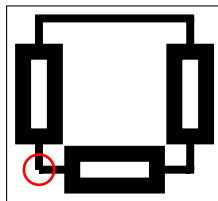
```

1 \begin{circuitikz}
2   \draw (0,0) to[D*, v=$v_D$,
3     i=1<\milli\ampere>, o-*] (3,0);
4 \end{circuitikz}

```

4.11 Line joins between Path Components

Line joins should be calculated correctly, if the were on the same path and if the path is not closed. For example, the following path is not closed correctly (*-cycle* does not work here!):

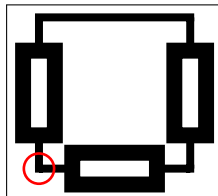


```

1 \begin{tikzpicture}[line width=3pt,european]
2   \draw (0,0) to[R]++(2,0)to[R]++(0,2)
3     --++(-2,0)to[R]++(0,-2);
4   \draw[red,line width=1pt] circle(2mm);
5 \end{tikzpicture}

```

To correct the line ending, there are support shapes to fill the missing rectangle. They can be used like the support shapes(*,o,d) using a dot (.) on one or both ends of a component(have a look at the last resistor in this example:



```

1 \begin{tikzpicture}[line width=3pt,european]
2   \draw (0,0) to[R]++(2,0)to[R]++(0,2)
3     --++(-2,0)to[R,.]++(0,-2);
4   \draw[red,line width=1pt] circle(2mm);
5 \end{tikzpicture}

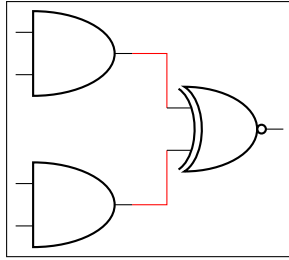
```

5 Colors

5.1 Shape colors

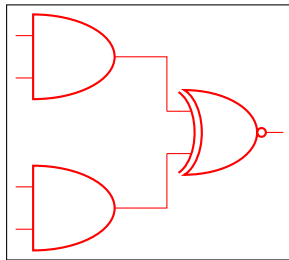
The color of the components is stored in the key `\circuitikzbasekey/color`. CircuiTikZ tries to follow the color set in TikZ, although sometimes it fails. If you change color in the picture, please do not use just the color name as a style, like `[red]`, but rather assign the style `[color=red]`.

Compare for instance



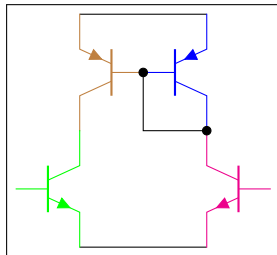
```
1 \begin{circuitikz} \draw[red]
2 (0,2) node[and port] (myand1) {}
3 (0,0) node[and port] (myand2) {}
4 (2,1) node[xnor port] (myxnor) {}
5 (myand1.out) -| (myxnor.in 1)
6 (myand2.out) -| (myxnor.in 2)
7 ;\end{circuitikz}
```

and



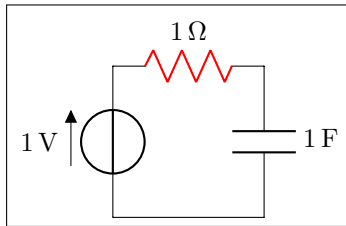
```
1 \begin{circuitikz} \draw[color=red]
2 (0,2) node[and port] (myand1) {}
3 (0,0) node[and port] (myand2) {}
4 (2,1) node[xnor port] (myxnor) {}
5 (myand1.out) -| (myxnor.in 1)
6 (myand2.out) -| (myxnor.in 2)
7 ;\end{circuitikz}
```

One can of course change the color *in medias res*:



```
1 \begin{circuitikz} \draw
2 (0,0) node[PNP, color=blue] (pnp2) {}
3 (pnp2.B) node[PNP, xscale=-1, anchor=B, color=brown] (pnp1) {}
4 (pnp1.C) node[NPN, anchor=C, color=green] (npn1) {}
5 (pnp2.C) node[NPN, xscale=-1, anchor=C, color=magenta] (npn2) {}
6 (pnp1.E) -- (pnp2.E) (npn1.E) -- (npn2.E)
7 (pnp1.B) node[circ] {} |- (pnp2.C) node[circ] {}
8 ;\end{circuitikz}
```

The all-in-one stream of bipoles poses some challenges, as only the actual body of the bipole, and not the connecting lines, will be rendered in the specified color. Also, please notice the curly braces around the `to`:

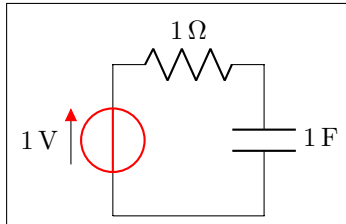


```

1 \begin{circuitikz} \draw
2   (0,0) to[V=1<\volt>] (0,2)
3     { to[R=1<\ohm>, color=red] (2,2) }
4       to[C=1<\farad>] (2,0) -- (0,0)
5 ;\end{circuitikz}

```

Which, for some bipoles, can be frustrating:

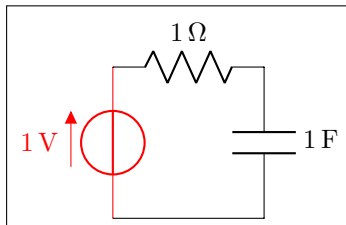


```

1 \begin{circuitikz} \draw
2   (0,0){to[V=1<\volt>, color=red] (0,2) }
3     to[R=1<\ohm>] (2,2)
4     to[C=1<\farad>] (2,0) -- (0,0)
5 ;\end{circuitikz}

```

The only way out is to specify different paths:



```

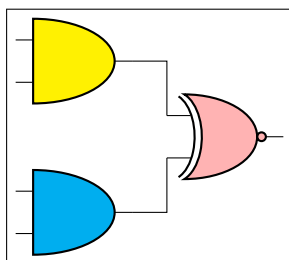
1 \begin{circuitikz} \draw[color=red]
2   (0,0) to[V=1<\volt>, color=red] (0,2);
3   \draw (0,2) to[R=1<\ohm>] (2,2)
4     to[C=1<\farad>] (2,0) -- (0,0)
5 ;\end{circuitikz}

```

And yes: this is a bug and *not* a feature...

5.2 Fill colors

Since version 0.8.4, you can also fill most shapes with a color (the manual specifies which ones are fillable or not). The syntax is quite intuitive:

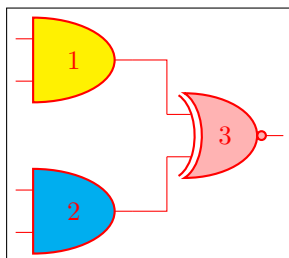


```

1 \begin{circuitikz} \draw
2   (0,2) node[and port, fill=yellow] (myand1) {}
3   (0,0) node[and port, fill=cyan] (myand2) {}
4   (2,1) node[xnor port, fill=red!30!white] (myxnor) {}
5   (myand1.out) -| (myxnor.in 1)
6   (myand2.out) -| (myxnor.in 2)
7 ;\end{circuitikz}

```

You can combine shape colors with fill colors, too, but you should use the `draw color` option style for this:

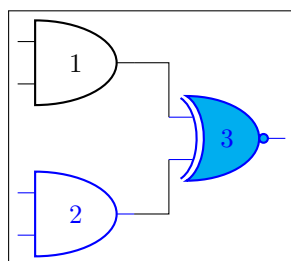


```

1 \begin{circuitikz} \draw[color=red]
2   (0,2) node[and port, fill=yellow] (myand1) {1}
3   (0,0) node[and port, fill=cyan] (myand2) {2}
4   (2,1) node[xnor port, fill=red!30!white] (myxnor) {3}
5   (myand1.out) -| (myxnor.in 1)
6   (myand2.out) -| (myxnor.in 2)
7 ;\end{circuitikz}

```

This is because, as you can see from the following example in port 2, you can't specify both a fill and a color in the node (yes, it's a bug too, but it's quite complex to solve given the current circuitTikZ architecture). a workaround is shown in port 3:

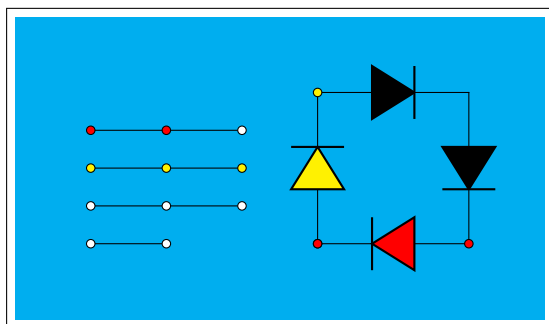


```

1 \begin{circuitikz} \draw
2   (0,2) node[and port, color=black] (myand1) {1}
3   (0,0) node[and port, color=blue, fill=cyan] (myand2)
      {2}
4   (2,1) {[color=blue] node[xnor port, fill=cyan] (myxnor
      ) {3}}
5   (myand1.out) -| (myxnor.in 1)
6   (myand2.out) -| (myxnor.in 2)
7 ;\end{circuitikz}

```

Notice also that the connection points are always filled, although the color *tries* to follow the color of the filling of the component:



```

1 \begin{circuitikz}
2   \fill[cyan] (0,3.0) rectangle (7,7);
3   \draw [fill=yellow, ] (4,4) to [D,o-o] ++(0,2) to[D*, fill=yellow] ++(2,0)
4     to[D*] ++(0,-2) to[D, fill=red, o-o] ++(-2,0);
5   \draw (1,4) node[ocirc]{} -- ++(1,0) node[ocirc]{};
6   \draw (1,4.5) to[short, o-o] ++(1,0) to[short, -o] ++(1,0);
7   \draw[fill=yellow] (1,5) to[short, o-o] ++(1,0) to[short, -o] ++(1,0);
8   \draw (1,5.5) to[short, fill=red, o-o] ++(1,0) to[short, -o] ++(1,0);
9 \end{circuitikz}

```

6 FAQ

Q: When using `\tikzexternalize` I get the following error:

! Emergency stop.

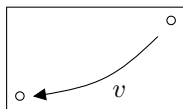
A: The TikZ manual states:

Furthermore, the library assumes that all \LaTeX pictures are ended with `\end{tikzpicture}`.

Just substitute every occurrence of the environment `circuitikz` with `tikzpicture`. They are actually pretty much the same.

Q: How do I draw the voltage between two nodes?

A: Between any two nodes there is an open circuit!



```

1 \begin{circuitikz} \draw
2   node[ocirc] (A) at (0,0) {}
3   node[ocirc] (B) at (2,1) {}
4   (A) to[open, v=$v$] (B)
5 ;\end{circuitikz}

```

Q: I cannot write `to[R = $R_1=12V$]` nor `to[ospst = open, 3s]`: I get errors.

A: It is a limitation of the parser.

Use `\def{\eq}{=}` to `to[R = $R_1\eq 12V$]` and `to[ospst = open{,} 3s]` instead; see caveat in section 4.1.

7 Defining new components

Per me si va ne la città dolente,
 per me si va ne l'eterno dolore,
 per me si va tra la perduta gente.
 ...
 Lasciate ogne speranza, voi ch'intrate.¹⁴

Big fat warning: this material is reserved to \TeX -hackers; do not delve into this if you have no familiarity with (at least) a bit of core \TeX programming and to the basic \TikZ layer. You have been warned.

7.1 Suggested setup

The suggested way to start working on a new component is to use the utilities of the `CircuitikZ` manual for checking and testing your device. Basically, find (or download) the source code of the last version of `CircuitikZ` and find the file `ctikzmanutils.sty`; copy it in your directory and prepare a file like this:

```

1 \documentclass[a4paper, titlepage]{article}
2 \usepackage{a4wide} %smaller borders
3 \usepackage[utf8]{inputenc}
4 \usepackage[T1]{fontenc}
5 \parindent=0pt
6 \parskip=4pt plus 6pt minus 2pt
7 \usepackage[siunitx, RPvoltages]{circuitikz}
8 \usepackage{ctikzmanutils}
9 \makeatletter
10 %% Test things here
11 % defines
12
13 % components
14
15 % paths
16 \makeatother
17
18 \begin{document}
19

```

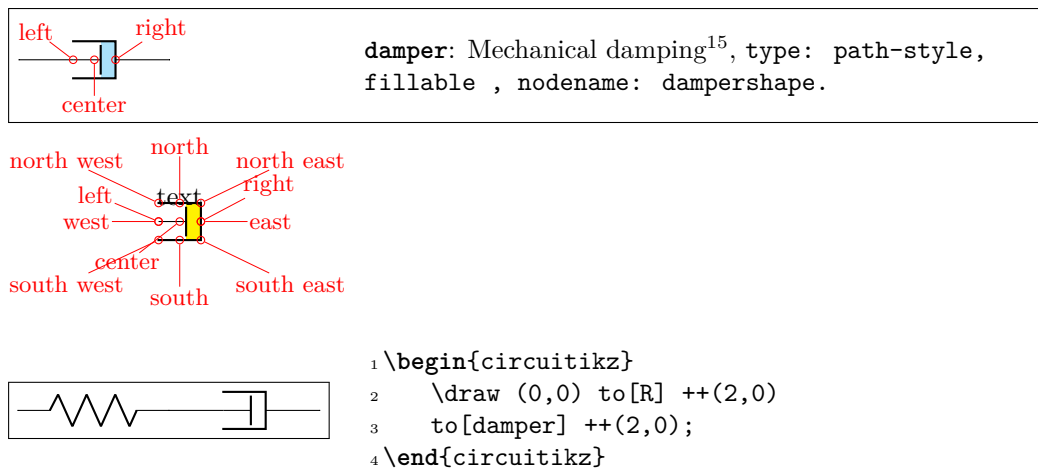
¹⁴<https://classicsincontext.wordpress.com/2010/02/28/canto-iii-per-me-si-va-ne-la-citta-dolente/>

```

20 \circuitdescbip*{damper}{Mechanical damping\footnotemark}{}(left/135/0.2,
    right/45/0.2, center/-90/0.3)
21
22 \geolrcoord{dampershape, fill=yellow}
23
24 \begin{LTXexample}[varwidth]
25 \begin{circuitikz}
26   \draw (0,0) to[R] ++(2,0)
27     to[damper] ++(2,0);
28 \end{circuitikz}
29 \end{LTXexample}
30 \end{document}

```

This will compile in something like this (in this case, we are using a couple of existing components to check everything is ok):



The command `circuitdescbip*` is used to show the component description (you can check the definition and the usage looking at `ctikzmanutils.sty` file, and the `\geolrcoord` is used to show the main anchors (geographical plus `left` and `right`) of the component.

From now on, you can add the new commands for the component between the `\makeatletter` and `\makeatother` commands and, modifying the example, check the results.

7.2 Path-style component

Let's define for example a path style component, like the one suggested by the user `@alex` on tex.stackexchange.com. The component will be a mix of the `damper` and the `spring` components already present.

The first step is to check if we can use the definition already existing for similar elements (for coherence of size) or if we need to define new ones; for this you have to check the file `pgfcirc.defines.tex`: we find

```

1 \ctikzset{bipoles/spring/height/.initial=.5}
2 \ctikzset{bipoles/spring/width/.initial=.5}
3 \ctikzset{bipoles/damper/height/.initial=.35}
4 \ctikzset{bipoles/damper/length/.initial=.3}
5 \ctikzset{bipoles/damper/width/.initial=.4}

```

We will use them; at this stage you can decide to add other parameters if you need them. (Notice, however, than although flexibility is good, these parameters should be described in the manual, otherwise they're as good as a fixed number in the code).

To define the new component we will look into `pgfcircbipoles.tex` and we will copy, for example, the definition of the damper into our code, just changing the name:

```

1 %% mechanical resistor - damper
2 \pgfcircdeclarebipole
3 {} % extra anchors
4 {\ctikzvalof{bipoles/damper/height}} % depth (under the path line)
5 {viscoe} % name
6 {\ctikzvalof{bipoles/damper/height}} % height (above the path line)
7 {\ctikzvalof{bipoles/damper/width}} % width
8 { % draw the bipole
9   \pgfpathrectanglecorners{\pgfpoint{\ctikzvalof{bipoles/damper/length}}\
    pgf@circ@res@right}{\pgf@circ@res@down}}{\pgfpoint{\pgf@circ@res@right}
    }\pgf@circ@res@up}}
10  \pgf@circ@maybefill
11
12  % line into the damper
13  \pgfpathmoveto{\pgfpoint{\pgf@circ@res@left}{\pgf@circ@res@zero}}
14  \pgfpathlineto{\pgfpoint{\ctikzvalof{bipoles/damper/length}}\
    pgf@circ@res@right}
15    {\pgf@circ@res@zero}}
16  \pgfusepath{stroke}
17
18  % damper box
19  \pgfsetlinewidth{\pgfkeysvalueof{/tikz/circuitikz/bipoles/thickness}}\
    pgfstartlinewidth}
20  \pgfpathmoveto{\pgfpoint{\pgf@circ@res@left}{\pgf@circ@res@down}}
21  \pgfpathlineto{\pgfpoint{\pgf@circ@res@right}{\pgf@circ@res@down}}
22  \pgfpathlineto{\pgfpoint{\pgf@circ@res@right}{\pgf@circ@res@up}}
23  \pgfpathlineto{\pgfpoint{\pgf@circ@res@left}{\pgf@circ@res@up}}
24
25  \pgfsetrectcap
26  \pgfsetmiterjoin
27  \pgfusepath{stroke}
28
29  % damper vertical element
30  \pgfpathmoveto{\pgfpoint{\ctikzvalof{bipoles/damper/length}}\
    pgf@circ@res@right}
31    {.8\pgf@circ@res@down}}
32  \pgfpathlineto{\pgfpoint{\ctikzvalof{bipoles/damper/length}}\
    pgf@circ@res@right}
33    {.8\pgf@circ@res@up}}
34  \pgfsetbuttcap
35  \pgfusepath{stroke}
36 }

```

This command will define a shape that is named `viscoeshape`, with all the correct geographical anchors based on the depth, height and width defined in the parameters of `\pgfcircdeclarebipole`. This is not sufficient for using the element in a `to[]` path command; you need to “activate” it with (this commands are normally in `pgfcircpath.tex`):

```

1 \def\pgf@circ@viscoe@path#1{\pgf@circ@bipole@path{viscoe}{#1}}

```

```

2 \compattikzset{viscoe/.style = {\circuitikzbasekey,
3   /tikz/to path=\pgf@circ@dviscoe@path, l=#1}}

```

And now you can show it with:

```

1 \circuitdescbip*{viscoe}{Mechanical viscoelastic element\footnotemark}{(left
   /135/0.2, right/45/0.2, center/-90/0.3)}
2
3 \geolrcoord{viscoeshape, fill=yellow}
4
5 \begin{LTExample}[varwidth]
6 \begin{circuitikz}
7   \draw (0,0) to[spring] ++(2,0)
8     to[viscoe] ++(2,0);
9 \end{circuitikz}
10 \end{LTExample}

```

Obviously, at first you just have a component that is the same as the one you copied with another name. It is now just a matter of modifying it so that it has the desired shape; in the example above you can already see the new symbol after the changes.

When doing the drawing, the `\pgfcircdeclarebipole` will setup the lengths `\pgf@circ@res@right` and `\pgf@circ@res@up` as the x - y coordinates of the upper right corner, and `\pgf@circ@res@left` and `\pgf@circ@res@down` as the x - y coordinates of the lower left corner of your shape. The center coordinate is usually at $(0pt, 0pt)$.

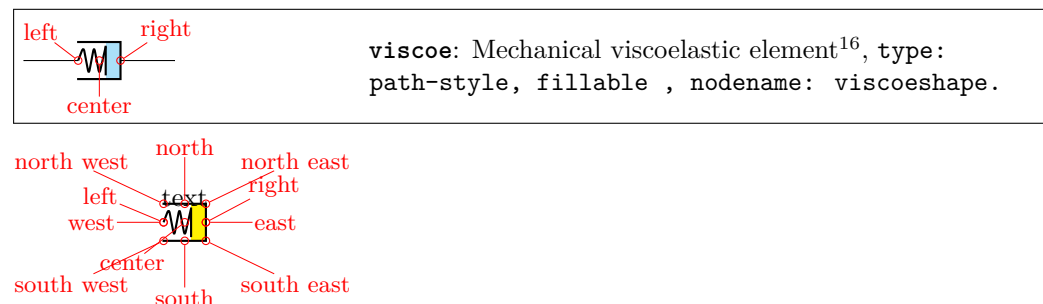
Looking at the implementation of the `spring` element, a possible implementation is changing the lines between lines 12 and 16 with:

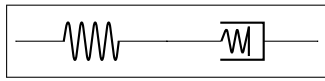
```

1   % spring into the damper
2   \pgfscope
3     \pgfpathmoveto{\pgfpoint{\pgf@circ@res@left}{\pgf@circ@res@zero}}
4     \pgfsetlinewidth{\pgfkeysvalueof{/tikz/circuitikz/bipoles/thickness}}\
      pgfstartlinewidth}
5     \pgfsetcornersarced{\pgfpoint{.25\pgf@circ@res@up}{.25\pgf@circ@res@up}}
6     \pgfpathlineto{\pgfpoint{.75\pgf@circ@res@left}{.75\pgf@circ@res@up}}
7     \pgfpathlineto{\pgfpoint{.5\pgf@circ@res@left}{-.75\pgf@circ@res@up}}
8     \pgfpathlineto{\pgfpoint{.25\pgf@circ@res@left}{.75\pgf@circ@res@up}}
9     \pgfpathlineto{\pgfpoint{0pt}{-.75\pgf@circ@res@up}}
10    \pgfpathlineto{\pgfpoint{\ctikzvalof{bipoles/damper/length}}\
      pgf@circ@res@right}{.75\pgf@circ@res@up}}
11    \pgfusepath{stroke}
12  \endpgfscope

```

which leads to:





```

1 \begin{circuitikz}
2   \draw (0,0) to[spring] ++(2,0)
3     to[viscoe] ++(2,0);
4 \end{circuitikz}

```

As a final note, notice that the `viscoe` element is already added to the standard package.

7.3 Node-style component

Adding a node-style component is much more straightforward. Just define it by following examples in, for example, `pgfcircstripoles.tex` or the other files; be careful that you should define all the geographical anchors of the shape if you want that the TikZ positioning options (like `left`, `above`, etc.) behave correctly with your component.

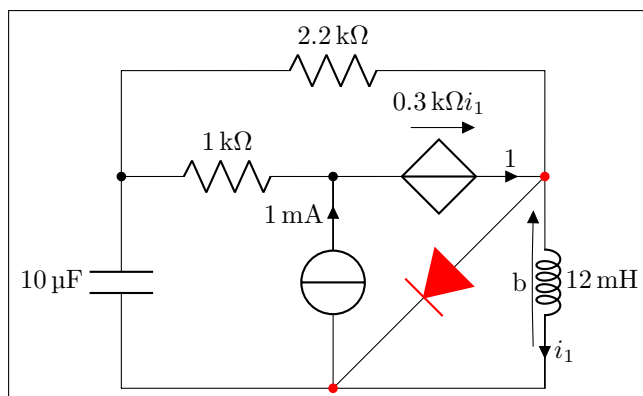
7.3.1 Finishing your work

Once you have a satisfactory element, you should

- Clean up your code;
- write a piece of documentation explaining its use with example;
- Propose the element for inclusion in the GitHub page of the project (you will have to license this as explained in that page, of course).

The best way of contributing is forking the project, adding your component in the correct files, modify the manual and creating a pull request for the developers to merge. Anyway, if this is a problem, just open an issue and someone (when they have time...) will answer.

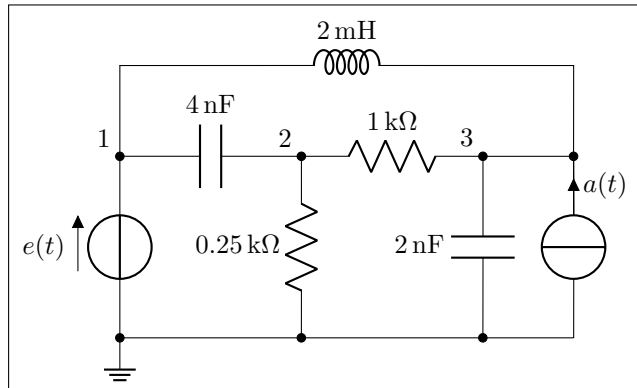
8 Examples



```

1 \begin{circuitikz}[scale=1.4]\draw
2   (0,0) to[C, l=10<\micro\farad>] (0,2) -- (0,3)
3     to[R, l=2.2<\kilo\ohm>] (4,3) -- (4,2)
4     to[L, l=12<\milli\henry>, i=$i_1$,v=b] (4,0) -- (0,0)
5   (4,2) { to[D*, **-, color=red] (2,0) }
6   (0,2) to[R, l=1<\kilo\ohm>, *-] (2,2)
7     to[cV, l=1,v=$\SI{.3}{\kilo\ohm} i_1$] (4,2)
8   (2,0) to[I, i=1<\milli\ampere>, -*] (2,2)
9;\end{circuitikz}

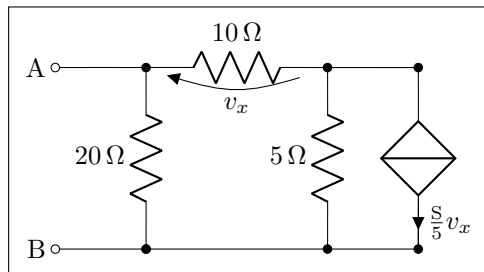
```



```

1 \begin{circuitikz}[scale=1.2]\draw
2   (0,0) node[ground] {}
3     to[V=$e(t)$, *-] (0,2) to[C=4<\nano\farad>] (2,2)
4     to[R, l_=.25<\kilo\ohm>, *-] (2,0)
5   (2,2) to[R=1<\kilo\ohm>] (4,2)
6     to[C, l_2=2<\nano\farad>, *-] (4,0)
7   (5,0) to[I, i_=$a(t)$, *-] (5,2) -- (4,2)
8   (0,0) -- (5,0)
9   (0,2) -- (0,3) to[L, l=2<\milli\henry>] (5,3) -- (5,2)
10
11  {[anchor=south east] (0,2) node {1} (2,2) node {2} (4,2) node {3}}
12;\end{circuitikz}

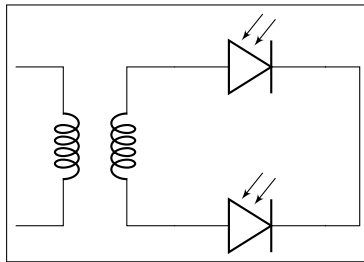
```



```

1 \begin{circuitikz}[scale=1.2]\draw
2   (0,0) node[anchor=east] {B}
3     to[short, o-*] (1,0)
4     to[R=20<\ohm>, *-] (1,2)
5     to[R=10<\ohm>, v=$v_x$] (3,2) -- (4,2)
6     to[cI=$\frac{\siemens}{5} v_x$, *-] (4,0) -- (3,0)
7     to[R=5<\ohm>, *-] (3,2)
8   (3,0) -- (1,0)
9   (1,2) to[short, -o] (0,2) node[anchor=east]{A}
10;\end{circuitikz}

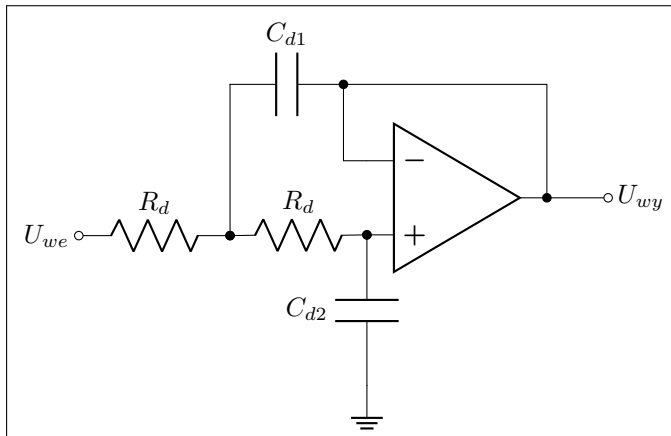
```

```

1 \begin{circuitikz}[scale=1]\draw
2   (0,0) node[transformer] (T) {}
3   (T.B2) to[pD] ($(T.B2)+(2,0)$) -| (3.5, -1)
4   (T.B1) to[pD] ($(T.B1)+(2,0)$) -| (3.5, -1)
5 ;\end{circuitikz}

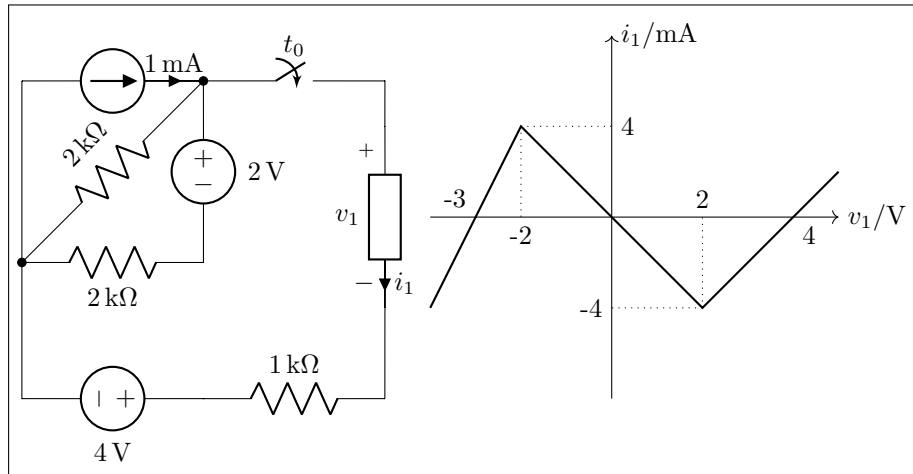
```



```

1 \begin{circuitikz}[scale=1]\draw
2   (5,.5) node [op amp] (opamp) {}
3   (0,0) node [left] {$U_{we}$} to [R, l=$R_d$, o-] (2,0)
4   to [R, l=$R_d$, *-] (opamp.+)
5   to [C, l=$C_{d2}$, *-] ($(opamp.+) + (0,-2)$) node [ground] {}
6   (opamp.out) |- (3.5,2) to [C, l=$C_{d1}$, *-] (2,2) to [short] (2,0)
7   (opamp.-) -| (3.5,2)
8   (opamp.out) to [short, *-o] (7,.5) node [right] {$U_{wy}$}
9 ;\end{circuitikz}

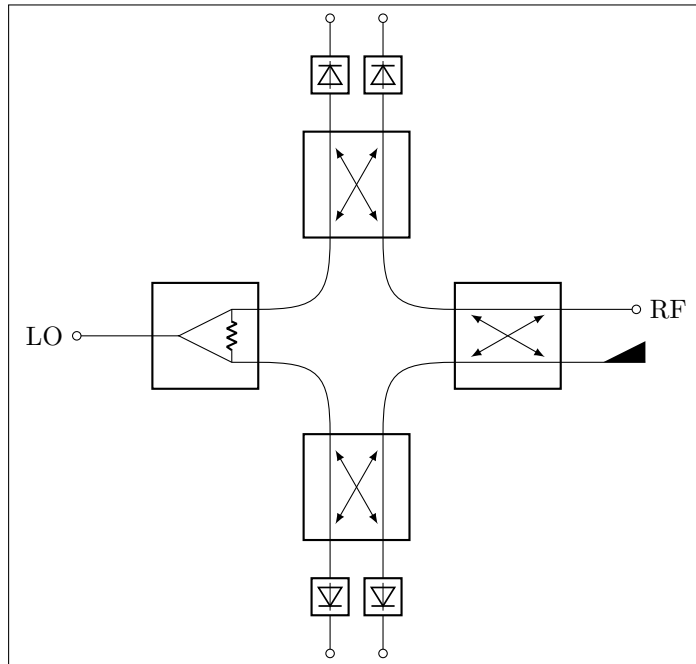
```



```

1 \begin{circuitikz}[scale=1.2, american]\draw
2   (0,2) to[I=1<\milli\ampere>] (2,2)
3     to[R, l=2<\kilo\ohm>, *-] (0,0)
4     to[R, l=2<\kilo\ohm>] (2,0)
5     to[V, v=2<\volt>] (2,2)
6     to[cspst, l=$t_0$] (4,2) -- (4,1.5)
7     to [generic, i=$i_1$, v=$v_1$] (4,-.5) -- (4,-1.5)
8   (0,2) -- (0,-1.5) to[V, v=4<\volt>] (2,-1.5)
9     to [R, l=1<\kilo\ohm>] (4,-1.5);
10
11 \begin{scope}[xshift=6.5cm, yshift=.5cm]
12   \draw [->] (-2,0) -- (2.5,0) node[anchor=west] {$v_1/\text{volt}$};
13   \draw [->] (0,-2) -- (0,2) node[anchor=west] {$i_1/\text{SI}\{\}\text{milli}\text{ampere}\}$} ;
14   \draw (-1,0) node[anchor=north] {-2} (1,0) node[anchor=south] {2}
15         (0,1) node[anchor=west] {4} (0,-1) node[anchor=east] {-4}
16         (2,0) node[anchor=north west] {4}
17         (-1.5,0) node[anchor=south east] {-3};
18   \draw [thick] (-2,-1) -- (-1,1) -- (1,-1) -- (2,0) -- (2.5,.5);
19   \draw [dotted] (-1,1) -- (-1,0) (1,-1) -- (1,0)
20             (-1,1) -- (0,1) (1,-1) -- (0,-1);
21 \end{scope}
22 \end{circuitikz}

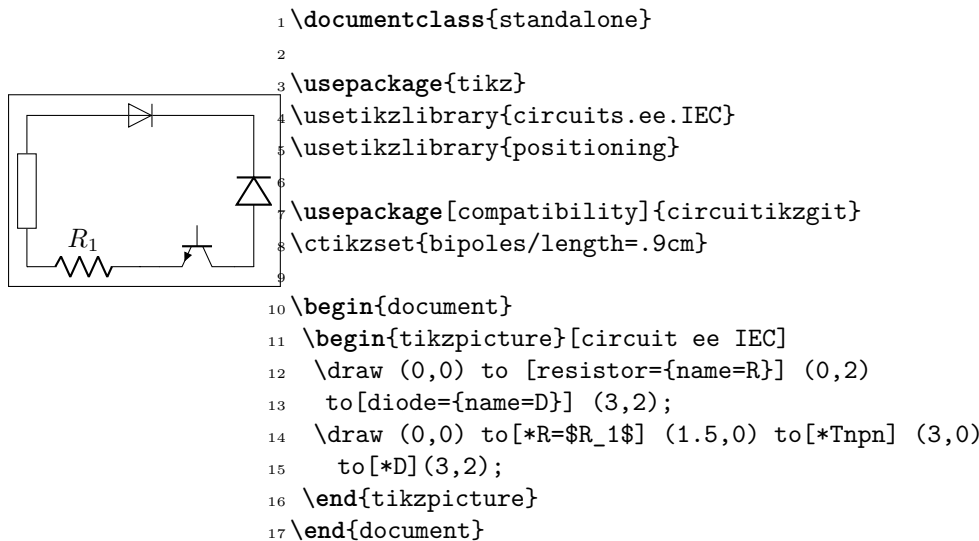
```



```

1  \begin{circuitikz}[scale=1]
2      \ctikzset{bipoles/detector/width=.35}
3      \ctikzset{quadpoles/coupler/width=1}
4      \ctikzset{quadpoles/coupler/height=1}
5      \ctikzset{tripoles/wilkinson/width=1}
6      \ctikzset{tripoles/wilkinson/height=1}
7      %\draw[help lines,red,thin,dotted] (0,-5) grid (5,5);
8      \draw
9      (-2,0) node[wilkinson](w1){}
10     (2,0) node[coupler] (c1) {}
11     (0,2) node[coupler,rotate=90] (c2) {}
12     (0,-2) node[coupler,rotate=90] (c3) {}
13     (w1.out1) .. controls ++(0.8,0) and ++(0,0.8) .. (c3.3)
14     (w1.out2) .. controls ++(0.8,0) and ++(0,-0.8) .. (c2.4)
15     (c1.1) .. controls ++(-0.8,0) and ++(0,0.8) .. (c3.2)
16     (c1.4) .. controls ++(-0.8,0) and ++(0,-0.8) .. (c2.1)
17     (w1.in) to[short,-o] ++(-1,0)
18     (w1.in) node[left=30] {LO}
19     (c1.2) node[match,yscale=1] {}
20     (c1.3) to[short,-o] ++(1,0)
21     (c1.3) node[right=30] {RF}
22     (c2.3) to[detector,-o] ++(0,1.5)
23     (c2.2) to[detector,-o] ++(0,1.5)
24     (c3.1) to[detector,-o] ++(0,-1.5)
25     (c3.4) to[detector,-o] ++(0,-1.5)
26     ;
27     \end{circuitikz}

```



9 Changelog

The major changes among the different circuitikz versions are listed here. See <https://github.com/circuitikz/circuitikz/commits> for a full list of changes.

- Version git (unreleased)
 - Fixed placement of straightlabels within 4th quadrant
 - Fixed straightvoltages at Diodes, varcap and some other components
 - Adjusted ground symbols to better match ISO standard
 - Fixed a bug about straightlabels (thanks to @fotesan)
 - Added Romano as contributor
 - Added a CONTRIBUTING file
 - Added new sources (cute european versions, noise sources)
 - Added new types of amplifiers, and option to flip inputs and outputs
 - Added bidirectional diodes (diac) thanks to Andre Lucas Chinazzo
 - Added L,R,C sensors (with european, american and cute variants)
 - Added stacked labels (thanks to the original work by Claudio Fiandrino)
 - Make the position of voltage symbols adjustable
 - Make the position of arrows in FETs and BJTs adjustable
 - Added the bulb symbol
 - Added options for solving the voltage direction problems.
 - Added chips (DIP, QFP) with a generic number of pins.
 - Added special anchors for transformers (and fixed the wrong center anchor)
 - Changed the logical port implementation to multiple inputs (thanks to John Kormylo)
 - Changed labels spacing so that they are independent on scale factor
 - Fixed the position of text labels in amplifiers
 - Added new switches.

- Added border anchors for the logic gates
- Added antennas with no wires, loudspeakers and microphone
- Make the components fillable
- Added the oscilloscope component
- Added viscoelastic element
- Added a manual section on how to define new components
- Version 0.8.3 (2017-05-28)
 - Removed unwanted lines at to-paths if the starting point is a node without a explicit anchor.
 - Fixed scaling option, now all parts are scaled by bipoles/length
 - Surge arrester appears no more if a to path is used without []-options
 - Fixed current placement now possible with paths at an angle of around 280°
 - Fixed voltage placement now possible with paths at an angle of around 280°
 - Fixed label and annotation placement (at some angles position not changable)
 - Adjustable default distance for straight-voltages: ‘bipoles/voltage/straight label distance’
 - Added Symbol for bandstop filter
 - New annotation type to show flows using f=... like currents, can be used for thermal, power or current flows
- Version 0.8.2 (2017-05-01)
 - Fixes pgfkeys error using alternatively specified mixed colors(see pgfplots manual section “4.7.5 Colors”)
 - Added new switches “ncs” and “nos”
 - Reworked arrows at spst-switches
 - Fixed direction of controlled american voltage source
 - “v<=” and “i<=” do not rotate the sources anymore(see them as “counting direction indication”, this can be different then the shape orientation); Use the option “invert” to change the direction of the source/appearance of the shape.
 - current label “i=” can now be used independent of the regular label “l=” at current sources
 - rewrite of current arrow placement. Current arrows can now also be rotated on zero-length paths
 - New DIN/EN compliant operational amplifier symbol “en amp”
- Version 0.8.1 (2017-03-25)
 - Fixed unwanted line through components if target coordinate is a name of a node
 - Fixed position of labels with subscript letters.
 - Absolute distance calculation in terms of ex at rotated labels
 - Fixed label for transistor paths (no label drawn)
- Version 0.8 (2017-03-08)
 - Allow use of voltage label at a [short]

- Correct line joins between path components (to[...])
- New Pole-shape `.-.` to fill perpendicular joins
- Fixed direction of controlled american current source
- Fixed incorrect scaling of magnetron
- Fixed: Number of american inductor coils not adjustable
- Fixed Battery Symbols and added new battery2 symbol
- Added non-inverting Schmitttrigger
- Version 0.7 (2016-09-08)
 - Added second annotation label, showing, e.g., the value of an component
 - Added new symbol: magnetron
 - Fixed name conflict of diamond shape with `tikz.shapes` package
 - Fixed varcap symbol at small scalings
 - New packet-option `straightvoltages`, to draw straight(no curved) voltage arrows
 - New option `invert` to revert the node direction at paths
 - Fixed american voltage label at special sources and battery
 - Fixed/rotated battery symbol(longer lines by default positive voltage)
 - New symbol Schmitttrigger
- Version 0.6 (2016-06-06)
 - Added Mechanical Symbols (damper,mass,spring)
 - Added new connection style diamond, use `(d-d)`
 - Added new sources voosource and ioosource (double zero-style)
 - All diode can now drawn in a stroked way, just use globel option `strokediode` or `stroke` instead of full/empty, or D-. Use this option for compliance with DIN standard EN-60617
 - Improved Shape of Diodes:tunnel diode, Zener diode, schottky diode (bit longer lines at cathode)
 - Reworked igbt: New anchors G,gate and new L-shaped form Lnight, Lpight
 - Improved shape of all fet-transistors and mirrored p-chan fets as default, as pnp, pmos, pfet are already. This means a backward-incompatibility, but smaller code, because p-channels mosfet are by default in the correct direction(source at top). Just remove the `'yscale=-1'` from your p-chan fets at old pictures.
- Version 0.5 (2016-04-24)
 - new option boxed and dashed for hf-symbols
 - new option solderdot to enable/disable solderdot at source port of some fets
 - new parts: photovoltaic source, piezo crystal, electrolytic capacitor, electromechanical device(motor, generator)
 - corrected voltage and current direction(option to use old behaviour)
 - option to show body diode at fet transistors
- Version 0.4
 - minor improvements to documentation

- comply with TDS
- merge high frequency symbols by Stefan Erhardt
- added switch (not opening nor closing)
- added solder dot in some transistors
- improved ConTeXt compatibility
- Version 0.3.1
 - different management of color...
 - fixed typo in documentation
 - fixed an error in the angle computation in voltage and current routines
 - fixed problem with label size when scaling a tikz picture
 - added gas filled surge arrester
 - added compatibility option to work with Tikz's own circuit library
 - fixed infinite in arctan computation
- Version 0.3.0
 - fixed gate node for a few transistors
 - added mixer
 - added fully differential op amp (by Kristofer M. Monisit)
 - now general settings for the drawing of voltage can be overridden for specific components
 - made arrows more homogeneous (either the current one, or latex' bt pgf)
 - added the single battery cell
 - added fuse and asymmetric fuse
 - added toggle switch
 - added varistor, photoresistor, thermocouple, push button
 - added thermistor, thermistor ptc, thermistor ptc
 - fixed misalignment of voltage label in vertical bipoles with names
 - added isfet
 - added noiseless, protective, chassis, signal and reference grounds (Luigi «Liverpool»)
- Version 0.2.4
 - added square voltage source (contributed by Alistair Kwan)
 - added buffer and plain amplifier (contributed by Danilo Piazzalunga)
 - added squid and barrier (contributed by Cor Molenaar)
 - added antenna and transmission line symbols contributed by Leonardo Azzinnari
 - added the changeover switch spdt (suggestion of Fabio Maria Antoniali)
 - rename of context.tex and context.pdf (thanks to Karl Berry)
 - updated the email address
 - in documentation, fixed wrong (non-standard) labelling of the axis in an example (thanks to prof. Claudio Beccaria)
 - fixed scaling inconsistencies in quadrupoles
 - fixed division by zero error on certain vertical paths

- introduced options `straightlabels`, `rotatelabels`, `smartlabels`
- Version 0.2.3
 - fixed compatibility problem with label option from `tikz`
 - Fixed resizing problem for shape `ground`
 - Variable capacitor
 - polarized capacitor
 - ConTeXt support (read the manual!)
 - `nfet`, `nigfete`, `nigfetd`, `pfet`, `pigfete`, `pigfetd` (contribution of Clemens Helfmeier and Theodor Borsche)
 - `njfet`, `pjfet` (contribution of Danilo Piazzalunga)
 - `pigbt`, `nigbt`
 - *backward incompatibility* potentiometer is now the standard resistor-with-arrow-in-the-middle; the old potentiometer is now known as variable resistor (or `vR`), similarly to variable inductor and variable capacitor
 - triac, thyristor, memristor
 - new property “name” for bipoles
 - fixed voltage problem for batteries in american voltage mode
 - european logic gates
 - *backward incompatibility* new american standard inductor. Old american inductor now called “cute inductor”
 - *backward incompatibility* transformer now linked with the chosen type of inductor, and version with core, too. Similarly for variable inductor
 - *backward incompatibility* styles for selecting shape variants now end are in the plural to avoid conflict with paths
 - new placing option for some tripoles (mostly transistors)
 - mirror path style
- Version 0.2.2 - 20090520
 - Added the shape for lamps.
 - Added options `europeanresistor`, `europeaninductor`, `americanresistor` and `americaninductor`, with corresponding styles.
 - FIXED: error in transistor arrow positioning and direction under negative `xscale` and `yscale`.
- Version 0.2.1 - 20090503
 - Op-amps added
 - added options `arrowmos` and `noarrowmos`, to add arrows to pmos and nmos
- Version 0.2 - 20090417 First public release on CTAN
 - *Backward incompatibility*: labels ending with `:angle` are not parsed for positioning anymore.
 - Full use of TikZ keyval features.
 - White background is not filled anymore: now the network can be drawn on a background picture as well.

- Several new components added (logical ports, transistors, double bipoles, ...).
 - Color support.
 - Integration with `{siunitx}`.
 - Voltage, american style.
 - Better code, perhaps. General cleanup at the very least.
- Version 0.1 - 2007-10-29 First public release

Index of the components

adc, 33
adder, 33
afuse, 22
ageneric, 21
american and port, 56
american controlled current source, 28
american controlled voltage source, 28
american current source, 27
american gas filled surge arrester, 31
american inductive sensor, *see* sL
american inductor, *see* L
american nand port, 56
american nor port, 56
american not port, 56
american or port, 56
american potentiometer, *see* pR, *see* pR
american resistive sensor, *see* sR
american resistor, *see* resistor, *see* R
american voltage source, 27
american xnor port, 56
american xor port, 56
ammeter, 17, 20
amp, 34
antenna, 44
asymmetric fuse, *see* afuse

bandpass, 33
bandstop, 33
bareantenna, 43
bareRXantenna, 43
bareTXantenna, 43
barrier, 31
battery, 27
battery1, 27
battery2, 27
biD*, *see* full bidirectionaldiode
biDo, *see* empty bidirectionaldiode
buffer, 49
bulb, 32

C, *see* capacitor
capacitive sensor, 26
capacitor, 26
cceI, *see* cute european controlled current source
cceV, *see* cute european controlled voltage source
ccgsw, *see* cute closing switch
ccsw, *see* cute closed switch
ceI, *see* cute european current source
ceV, *see* cute european voltage source
cground, 19

circ, 51
circulator, 33
cisourceC, *see* cute european controlled current source
cisesourcesin, *see* controlled sinusoidal current source
closing switch, 53
cogsw, *see* cute opening switch
controlled isourcesin, *see* controlled sinusoidal current source
controlled sinusoidal current source, 29
controlled sinusoidal voltage source, 29
controlled vsourcesin, *see* controlled sinusoidal voltage source
cosw, *see* cute open switch
coupler, 34
coupler2, 34
crossing, 51
csI, *see* controlled sinusoidal current source
cspst, *see* closing switch
csV, *see* controlled sinusoidal voltage source
curarrow, 51
cute choke, 26
cute closed switch, 54
cute closing switch, 54
cute european controlled current source, 28
cute european controlled voltage source, 28
cute european current source, 27
cute european voltage source, 27
cute inductive sensor, *see* sL
cute inductor, *see* L
cute open switch, 54
cute opening switch, 54
cute spdt down, 54
cute spdt down arrow, 18, 55
cute spdt mid, 54
cute spdt mid arrow, 55
cute spdt up, 54
cute spdt up arrow, 55
cvsourceC, *see* cute european controlled voltage source
cvsourcesin, *see* controlled sinusoidal voltage source

D*, *see* full diode
D-, *see* stroke diode
dac, 33
damper, 31, 83
dcisource, 31
dcvsource, 31
detector, 34
diamondpole, 51

dipchip, 59
 Do, *see* empty diode
 dsp, 33

eC, *see* ecapacitor
 ecapacitor, 26
 eground, 19
 eground2, 19
 elko, *see* ecapacitor
 elmech, 45
 empty bidirectionaldiode, 23
 empty diode, 23
 empty led, 23
 empty photodiode, 23
 empty Schottky diode, 23
 empty thyristor, 25
 empty triac, 25
 empty tunnel diode, 23
 empty varcap, 23
 empty Zener diode, 23
 empty ZZener diode, 23
 en amp, 48
 esource, 30
 european and port, 56
 european controlled current source, 28
 european controlled voltage source, 28
 european current source, 27
 european gas filled surge arrester, 31
 european inductive sensor, *see* sL
 european inductor, *see* L
 european nand port, 56
 european nor port, 56
 european not port, 56
 european or port, 56
 european potentiometer, *see* pR
 european resistive sensor, *see* sR
 european resistor, *see* R
 european voltage source, 27
 european xnor port, 57
 european xor port, 56

fd inst amp, 48
 fd op amp, 48
 fft, 34
 full bidirectionaldiode, 24
 full diode, 23
 full led, 24
 full photodiode, 24
 full Schottky diode, 23
 full thyristor, 25
 full triac, 25
 full tunnel diode, 24
 full varcap, 24
 full Zener diode, 24

full ZZener diode, 24
 fullgeneric, 21
 fuse, 22

generic, 21
 gm amp, 48
 ground, 19
 gyrator, 46

hemt, 37
 highpass, 33

inputarrow, 51
 inst amp, 48
 inst amp ra, 49
 invschmitt, 57
 ioosource, 30
 isfet, 40
 isourceC, *see* cute european current source
 isourceN, *see* noise current source
 isourcesin, *see* sinusoidal current source

jump crossing, 51

L, 26, 27
 lamp, 32
 leD*, *see* full led
 leD-, *see* stroke led
 leDo, *see* empty led
 Lnigbt, 37
 loudspeaker, 32
 lowpass, 33
 Lpigt, 38

magnetron, 42
 mass, 31
 match, 44
 memristor, 21
 mic, 32
 mixer, 33
 Mr, *see* memristor
 mslstub, 43
 msport, 43
 msrstub, 44
 mstline, 43

ncpb, *see* normally closed push button
 ncs, *see* normal closed switch
 nfet, 39
 nground, 19
 nI, *see* noise current source
 nigbt, 37
 nigfetd, 39
 nigfete, 39
 nigfete,solderdot, 39

nigfetebulk, 39
njfet, 40
nmos, 37, 38
noise current source, 29
noise voltage source, 29
nopb, *see* push button
normal closed switch, 53
normal open switch, 53
normally closed push button, 54
normally open push button, *see* push button
nos, *see* normal open switch
npn, 18, 37
npn,photo, 37
nV, *see* noise voltage source

ocirc, 51
ohmmeter, 20
op amp, 48
open, 21
opening switch, 53
oscillator, 33
oscope, 20
ospst, *see* opening switch

pC, *see* polar capacitor
pD*, *see* full photodiode
pD-, *see* stroke photodiode
pDo, *see* empty photodiode
pfet, 39
pground, 19
phaseshifter, 34
photoresistor, *see* phR
phR, 22
piattenuator, 34
piezoelectric, 26
pigbt, 37
pigfetd, 40
pigfete, 40
pigfetebulk, 40
pjfet, 40
plain amp, 19, 49
plain crossing, 51
pmos, 37, 38
pmos,emptycircle, 39
pnp, 37
pnp,photo, 37
polar capacitor, 26
pR, 17, *see* pR, 21, 22
push button, 54
pvsources, 30
PZ, *see* piezoelectric

qfpchip, 59

R, *see* resistor, 21, 22

resistor, 17
rground, 19
rxantenna, 44

sC, *see* capacitive sensor
schmitt, 57
sD*, *see* full Schottky diode
sD-, *see* stroke Schottky diode
sDo, *see* empty Schottky diode
sground, 19
short, 21
sI, *see* sinusoidal current source
sinusoidal current source, 28
sinusoidal voltage source, 28
sL, 26, 27
spdt, 54
spring, 31
spst, *see* switch
square voltage source, 30
squid, 31
sqV, *see* square voltage source
sR, 21, 22
stroke diode, 24
stroke led, 24
stroke photodiode, 24
stroke Schottky diode, 24
stroke thyristor, 25
stroke tunnel diode, 24
stroke varcap, 25
stroke Zener diode, 24
stroke ZZener diode, 24
sV, *see* sinusoidal voltage source
switch, 53

tattenuator, 34
tD*, *see* full tunnel diode
tD-, *see* stroke tunnel diode
tDo, *see* empty tunnel diode
tfullgeneric, 21
tgeneric, 21
tground, 19
thermistor, *see* thR
thermistor ntc, *see* thRn
thermistor ptc, *see* thRp
thermocouple, 22
thR, 22
thRn, 22
thRp, 22
thyristor, 25
TL, 44
tline, *see* TL
tlinestub, 44
toggle switch, 54
Tr, *see* triac

Tr*, *see* full triac
 transformer, 46
 transformer core, 47
 transmission line, *see* TL
 triac, 25
 Tro, *see* empty triac
 tV, *see* vsourcetri
 twoport, 33
 txantenna, 44
 Ty, *see* thyristor
 Ty*, *see* full thyristor
 Ty-, *see* stroke thyristor
 Tyo, *see* empty thyristor

 vamp, 34
 variable american inductor, *see* vL
 variable american resistor, *see* vR
 variable capacitor, 26
 variable cute inductor, *see* vL
 variable european inductor, *see* vL
 variable european resistor, *see* vR
 varistor, 22
 vC, *see* variable capacitor
 VC*, *see* full varcap
 VC-, *see* stroke varcap
 vcc, 19
 VCo, *see* empty varcap

 vco, 33
 vee, 19
 viscoe, 31, 85
 vL, 26, 27
 voltmeter, 20
 voosource, 30
 vphaseshifter, 34
 vpiattenuator, 34
 vR, 21, 22
 vsourceC, *see* cute european voltage source
 vsourceN, *see* noise voltage source
 vsourcesin, *see* sinusoidal voltage source
 vsourcesquare, *see* square voltage source
 vsourcetri, 30
 vtattenuator, 34

 waves, 52
 wilkinson, 33

 xing, *see* crossing

 zD*, *see* full Zener diode
 zD-, *see* stroke Zener diode
 zDo, *see* empty Zener diode
 zzD*, *see* full ZZener diode
 zzD-, *see* stroke ZZener diode
 zzDo, *see* empty ZZener diode