CircuiTikz
Cheatsheet, page 1 of 2

1 Circuitikz

```
\usepackage{circuitikz}
\begin{circuitikz}[american]
  \draw (<point>)
  node[<e_type>?,<e_opt>*]?
  [
    to[ <wire_opt>*,
        <e_type>?,
        <e_opt>*    ]
    node[<e_type>,<e_opt>*]?
    (<point>)?
  ]*
```

Options

europeanvoltages
straightvoltages
americanvoltages
europeancurrents
americancurrents
europeanresistors
americanresistors
europeaninductors
americaninductors
cuteinductors

europeanports
americanports
european
american
siunitx
fulldiode
strokediode
emptydiod
arrowmos
noarrowmos
fetbodydiode

nofetbodydiode
fetsolderdot
nofetsolderdot
emptymoscircle
lazymos
straightlabels
rotatelabels
smartlabels
betterproportions

2 Annotations

Path Adornments

Bipole wire-end shorthands:

```
\draw (0,0) to[short] (2,0);      % wire only
\draw (0,0) to[open] (2,0);       % open wire
\draw (0,0) to[R,o-o] (2,0);      % open nodes
\draw (0,0) to[R,*-*] (2,0);      % closed
\draw (0,0) to[R, *-o] (2,0);     % mixed ends
```

Labels

An optional less-than (<) sign changes direction of a flow symbol (i, v, or f); to put a label above or below, use a carat (^) or underscore (_) respectively. Use `smartlabels`, `rotatelabels`, *or* `straightlabels`, *for control over automatic label rotations, and* `halign` *or* `valign` *for label alignment.*

- i= % current arrow
- v=<volts> % volt-diff
- f= % current arrow
- l=<name> % label
- l2=<lab1> and <lab2>

Flow & text placement & font can be controlled:

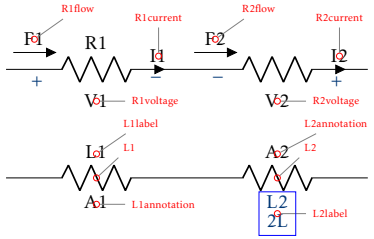
```
\ctikzset{current/distance=.2}    % move arrow
\ctikzset{voltage/label distance=.5}% mv label
\ctikzset{current/font=\tiny}     % font
```

Units

Normal	SIunits
<\ohm>	\SI {5}{k\ohm}
<\milli \farad>	\SI {-10}{V}
<\siemens>	\SI {8}{S}

Anchors

Each element and annotation defines its own set of anchors which can be referenced and moved:



3 Graphical Rendering

Classes

Most, but not all, elements have a "class". This allows for uniform referencing of components by class:

```
\ctikzset{inductors/scale=.7}     % scale all Ls
```

Classes are as follows (legacy are starred):

amplifiers
batteries
blocks
bipole*
capacitors
chips
csources
default
diodes
displays

electromech'ls
flipflops
grounds
inductors
instruments
logic ports
mechanicals
misc.
multipole*

muxdemuxes
power suppl's
resistors
RF
sources
switches
transistors
tripole*
tubes

Mirroring, Inversion, Rotation

Can `mirror` *or* `invert` *circuit elements drawn into paths, and* `rotate` *any node elements:*

```
\draw (0,0) to[pD, mirror] (2,0);% mirroring
\draw (0,0) to[pD, invert] (2,0); % inverting
\node[not port, rotate=-90] {};   % rotating
```

Use `ctikzflip-` *macro for "unflipping" text:*

```
\node[op amp, xscale=-1] {\ctikzflipx{OA2} };
```

Scaling

```
\ctikzset{scale=.7}               % whole doc
\ctikzset{resistors/scale=.7}     % by class
\ctikzset{resistors/height=.6}    % in 1 dir.
\node{}[C,scale=.7]               % by elem.
\node{}[xscale=.7]                % in 1 dir.
```

Line Thickness

```
\ctikzset{bipoles/thickness=4}    % by class
```

Colors

4 Advanced

Reusable Blocks

Define new commands that place one or more elements and wires, using only relative (local) coords. Commands can be parameterized and blocks and sub-block elements can/should be named:

```
\newcommand* {\myblock}[1]{ % define cmd.
node[sr-ff](#1-FF){} % use param. #1
node[and prt](#1-AND1) at ($(#1-FF) + (0,-2)$)
...} % end of definition

% now use the newly-defined command:
\draw (0,0) \myblock[B1]; % place block
\draw (0,4) \myblock[B2]; % and another

% can reference sub-block components:
\draw (B1-FF.pin 1) -- (B2-AND.in 2);
```

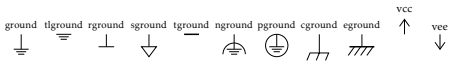
New Components

Package Version

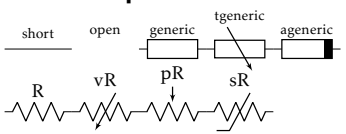
Upgrading on overleaf

5 Elements

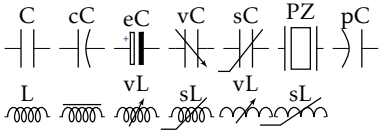
Grounds & Supplies



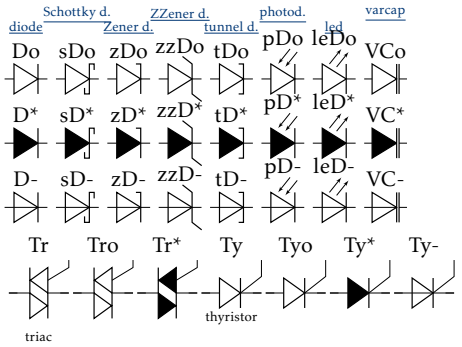
Resistive Bipoles



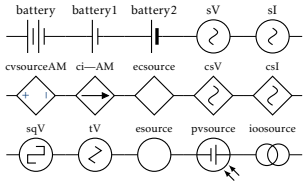
Capacitors & Inductors



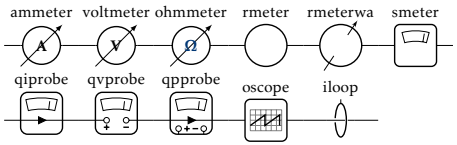
Diodes



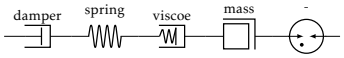
Sources & Generators



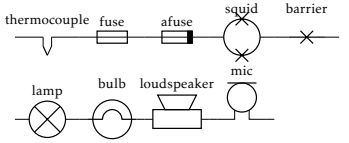
Instruments



Mechanical

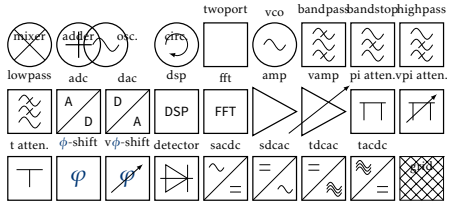


Miscellaneous Bipoles

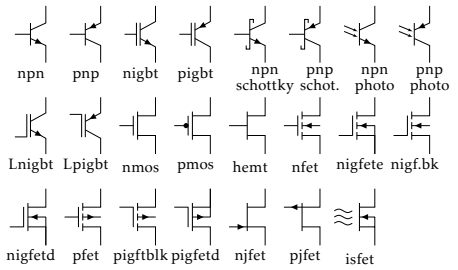


Crossings

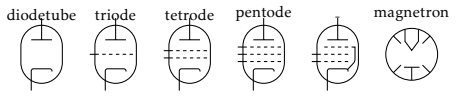
Block Diagrams



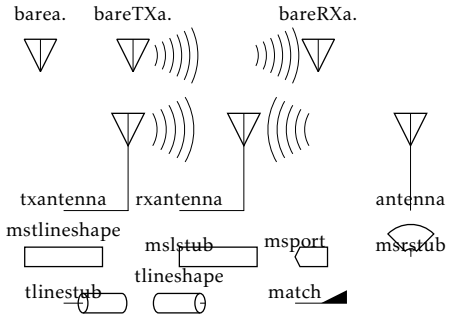
Transistors



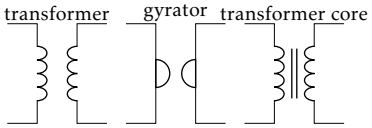
Electronic Tubes



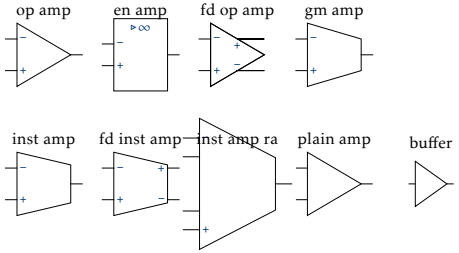
RF Components



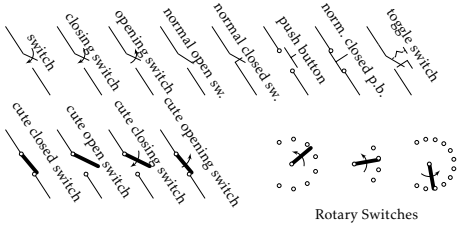
Transformers



Amplifiers

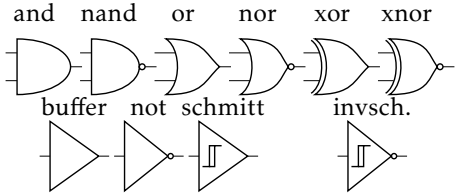


Switches



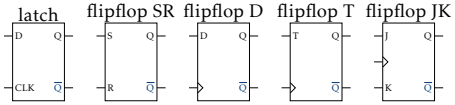
Logic Gates

All are nodes and end in "port" except schmitts. Buffers and schmitts can be used in the path style:



Flip-Flops

Customize further with `flipflop def` *which takes several options for pin names (* `t<n>` *), and for the clock signal (* `c<n>` *. Can create not signal with* `\ctikztextnot{}` *macro.*



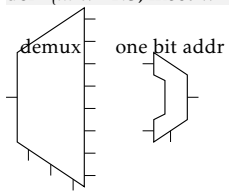
Multiplexers

Customize the number of pins left (`NL`*), bottom (*`NB`*) and right(*`NR`*); and relative size left (*`Lh`*) and right (*`Rh`*):*

```
demux/.style={muxdemux muxdemux
def={Lh=4, Rh=8, NL=1, NB=3, NR=8}}
```

Can create insets with `w`, `inset w`, `inset Lh`:

```
one bit adder/.style={muxdemux muxdemux
def={... w=1.5, inset w=.5, inset Lh=2, inset Rh=1.5}}
```

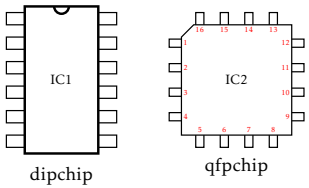


Chips

Customize chips with options ∈:

width	extn. pad fraction
pin spacing	num pins
extn. pins width	hide numbers
	thickness

```
node[dipchip,num pins=12,hide numbers,
  external pins width=0.3, external pad fraction=4]
node[qfpchip,num pins=16, external pad fraction=6]
```



Seven-Segment Displays

Options ∈:

width	sep	color on
thickness	box sep	color off
segment		

```
\ctikzset{sevenseg/color on=black}
```

Examples:

```
\foreach \i in {0,...,15} \path (\i,0)
  node[seven segment val=\i dot on box off]{};
\foreach \i in {0,...,15} \path(\i/2,-1)
  node[seven segment val=\i dot off box off,
  fill=gray!30!white]{};
```