1 Package Options

As of now, just one:

`noxtracmds` This will suppress the definition of a few auxiliary commands. See 1.1.

1.1 Auxiliary Commands

Those are based on some ideas from Redaelli et al. (CircuiTiKz). Main differences: a variable number of parameters (see below) and it always also adds an empty node <coord>

`\showcoordstrue` `\showcoordstrue`
`\shoocoordsfalse` `\showcoordsfalse`

These will affect as the `\coord` will behave, with `\showcoordstrue` a red pin will be added to the newly defined coordinate.

`\coord` `\coord(⟨coord⟩)`
`\pincoord` `\pincoord(⟨coord⟩)`
`\pincoord(⟨coord⟩ , ⟨color⟩)`
`\pincoord(⟨coord⟩, ⟨color⟩ , ⟨angle⟩)`
`\pincoord(⟨coord⟩, ⟨color⟩, ⟨angle⟩ , ⟨distance⟩)`

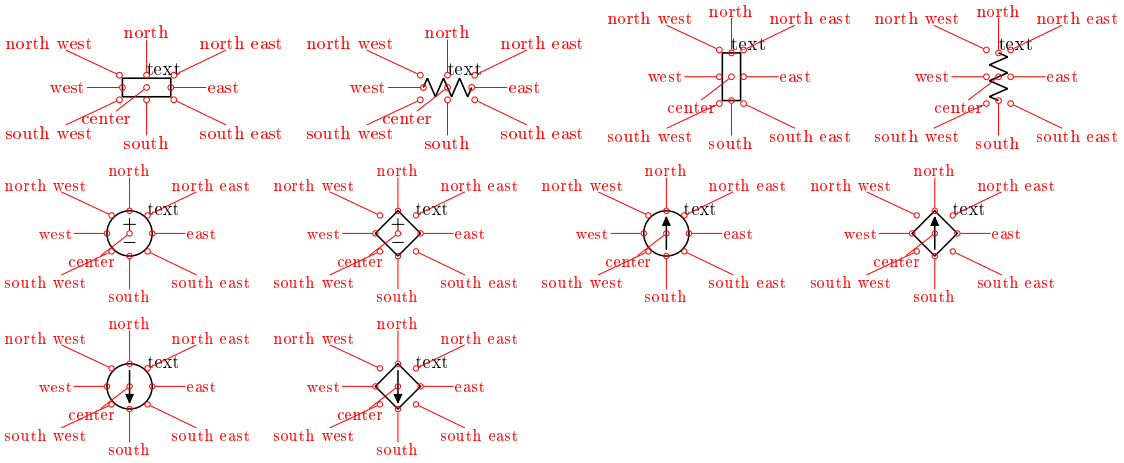
The `\coord` always expects a single parameter ⟨coord⟩. A coordinate and node with the same name will be created. If `\showcoordstrue` is en force, it will also add a pin.
The `\pincoord` expects from one to 4 parameters, as listed. If omitted, the default value for distance is 4 (unit: pt), the default value for the angle is -45 (degrees), the default value for color is blue. In fact, the `\coord(name)` is just a short cut for `\pincoord (name,red,45)`, if `\showcoordstrue`.

2 Auxiliary Shapes and Basic Keys

Those shapes are not intended for end users.

2.1 Auxiliary shapes

A set of auxiliary shapes are defined, but not meant to be used otherwise, though their anchors might be relevant:



Note: The main point being that, regardless of the sub-shape orientation, the intuitive geographical coordinates applies.

2.2 General Keys

The following set of keys allow for shape fine tunning:

outer sep	Text outer separation, initial value: 1.5pt
inner sep	Text inner separation, initial value: 1pt
thickness	Components thickness (relative to the drawing thickness), initial value: 2
tip len	tip len (current source). initial value: 4pt
tip type	possible values: *triangle* and *bezier*. initial value: *triangle*
minussign len	Minus sign len (voltage source). initial value: `\pgf@circ@Rlen/14`
plussign len	Plus sign len (voltage source). initial value: `1.1\pgf@circ@Rlen/14`
source radius	The base radius. initial value: `0.3\pgf@circ@Rlen`
round sources	Sources will be round ones
control sources	Sources will be control/diamond ones
generic, european	Impedances will be generic rectangles
zigzag, american	Impedances will be draw as zigzags

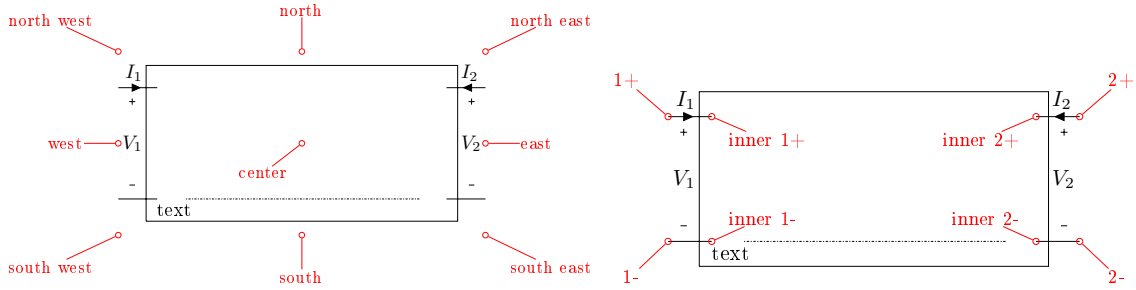
Note: Those keys can be used with all the following components: ⟨Quad⟩, ⟨Quad Z⟩, ⟨Quad Y⟩, ⟨Quad G⟩, ⟨Quad H⟩, ⟨ToQuad⟩, ⟨ToQuad Z⟩, ⟨ToQuad Y⟩, ⟨ToQuad G⟩, ⟨ToQuad H⟩, ⟨Black Box⟩, ⟨Thevenin⟩, ⟨Norton⟩, ⟨ToBlack Box⟩, ⟨ToThevenin⟩ and ⟨ToNorton⟩.

3 Z, Y, G, H Quadripoles

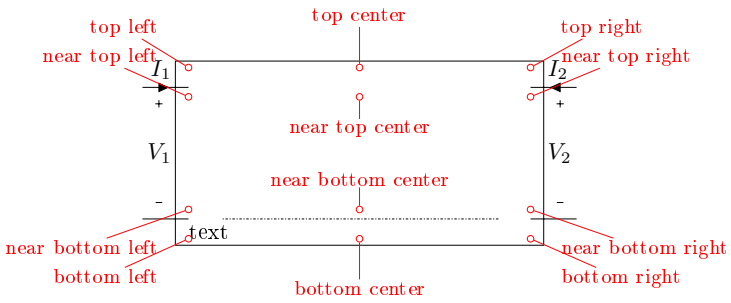
A set of configurable Quadripoles is defined, whereas quadripoles parameters (for instance Z_{11}, Z_{12}, Z_{21} and Z_{22}) are <key-value> parameters.

3.1 The Base Quadripole

The base shape just draws a base box and sets some connection anchors: 1+, 1−, *inner* 1+, *inner* 1−, 2+, 2−, *inner* 2+ and *inner* 2−, besides the geographic ones:



And also a set of (meant for) *text* anchors:



3.2 Customization keys

Additionally, one has:

base width	The 'box' width
half base width	Ditto, half width. Initial value:`2\pgf@circ@Rlen`.
base height	The distance between *1+* and *1-*. The 'box' full height is equal to 2*(*half base height* + *height ext* + *height ext+*).
half base height	Ditto, half height. Initial value:`\pgf@circ@Rlen/7`
height ext	Initial value:`2\pgf@circ@Rlen/7`
height ext+	Initial value:0
inner ext	distance between the 'box' and `inner1+/1-/2+/2-`. initial value: `\pgf@circ@Rlen/7`
outer ext	distance between the 'box' and `1+/1-/2+/2-`. initial value: `5\pgf@circ@Rlen/14`
inner marks	If set, the inner anchors will be marked.
outer marks	If set, the outer anchors will be marked.
invert	The shape will be inverted, more or less like 'x scale=-1'.
alt, opt	Case a Voltage source is zero, a series impedance will be draw vertically.
outer x fit to	*outer x fit*=`{⟨CoordA⟩}{⟨CoordB⟩}`. The width will be set so that ⟨1+⟩ and ⟨2+⟩ (or ⟨1-⟩ and ⟨2-⟩, depending on the used anchor) will fit ⟨CoordA⟩ and ⟨CoordB⟩

inner x fit to

inner x fit={⟨CoordA⟩}{⟨CoordB⟩}. The width will be set so that ⟨inner 1+⟩ and ⟨inner 2+⟩ (or ⟨inner 1-⟩ and ⟨inner 2-⟩, depending on the used anchor) will fit ⟨CoordA⟩ and ⟨CoordB⟩

y fit to

y fit={⟨CoordA⟩}{⟨CoordB⟩}. In the case of a quadripole, the distance between, lets say *1+* and *1-* will be made equal to the distance between *CoordA* and *CoordB*. In the case of a Thevenin/Norton, *1+* and *1-* will fit *CoordA* and *CoordB* respectively.

Note: *outer x fit* and *inner x fit* might result in a shape rotation. *y fit* in case of a quadripole will never result in a rotation, while in case of a thevenin/norton it might.

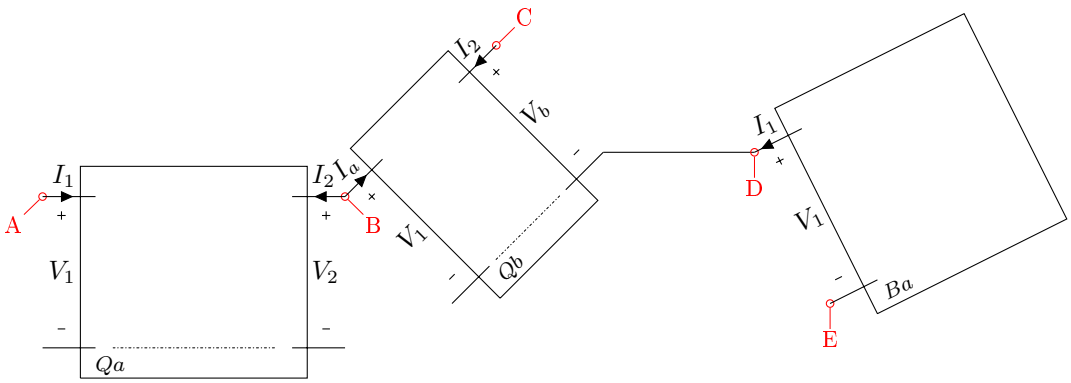
Note: Those keys can be used with all the following components: ⟨Quad⟩, ⟨Quad Z⟩, ⟨Quad Y⟩, ⟨Quad G⟩, ⟨Quad H⟩, ⟨ToQuad⟩, ⟨ToQuad Z⟩, ⟨ToQuad Y⟩, ⟨ToQuad G⟩, ⟨ToQuad H⟩, ⟨Black Box⟩, ⟨Thevenin⟩, ⟨Norton⟩, ⟨ToBlack Box⟩, ⟨ToThevenin⟩ and ⟨ToNorton⟩.

A small example of the 'fit to' keys:

LaTeX Code:

```
\begin{tikzpicture}
\draw (0,0) coordinate(A) \showcoord(A)<225:0.2> ++(4,0) coordinate(B) \showcoord(B)<-45:0.2> ++(2,2) coordinate(C
    ) \showcoord(C)<45:0.2> ;

\draw (A) node[Quad,anchor=1+,outer x fit to={A}{B}](Qa){\footnotesize$Qa$};
\draw (B) node[Quad,anchor=1+,outer x fit to={B}{C},I1=$I_a$,V2=$V_b$](Qb){\footnotesize$Qb$};

\draw (Qb.2-) -- ++(2,0) coordinate(D) \showcoord(D)<-90:0.2> ++(1,-2) coordinate(E) \showcoord(E)<-90:0.2>;

\draw (D) node[Black Box,anchor=1+,y fit to={D}{E}](Ba){\footnotesize$Ba$};

\draw (Qa.1-) ++(0,-1);
\end{tikzpicture}
```

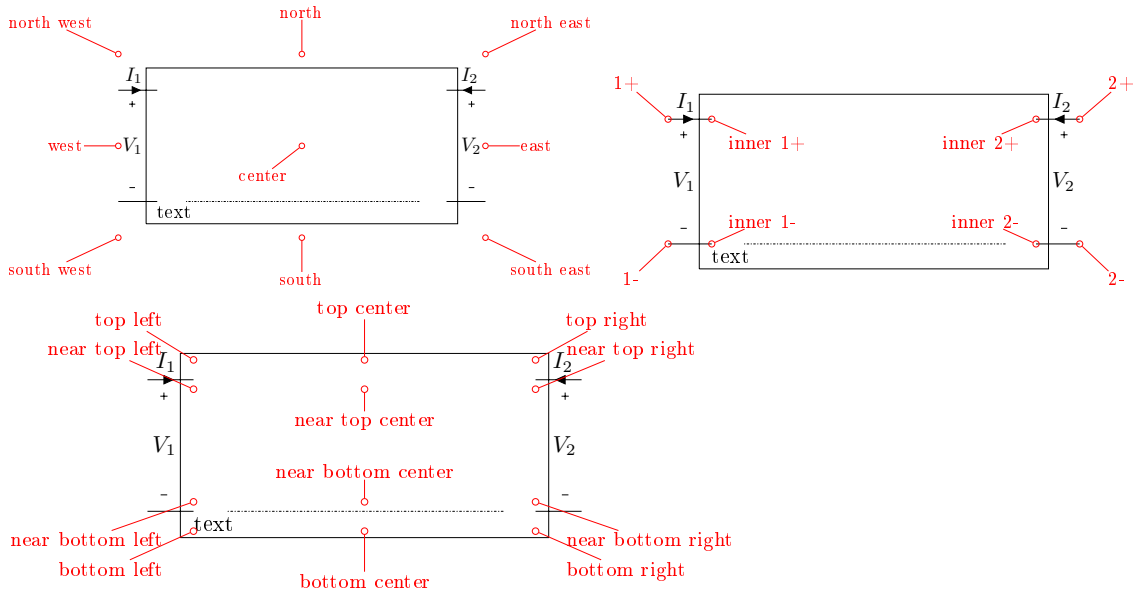
LaTeX Result:



3.3 Quad

This is just the base shape, to be used in cases whereas one just want to emphasises part of a circuit (using, for instance, the *inner x fit to* key, or just mark a two port black box.

Note: There is also a *ToQuad* to be used in a *to[ ]* path, in which case the key *outer x fit to* style will be triggered with the starting and ending points of the *to[ ]* path.



3.3.1 Quad Keys

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
I2	Initial value:`$I_2$`
V1	Initial value:`$V_1$`
V2	Initial value:`$V_2$`

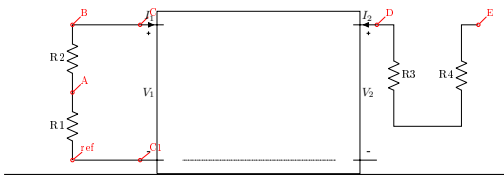
3.3.2 Example of fit to uses

Squeezing a Quadripole between two parts of a circuit (nodes C and D):

LaTeX Code:

```
\resizebox{0.4\textwidth}{!}{
\begin{tikzpicture}
  \draw (0,0) coordinate(ref) \showcoord(ref)<45:0.2> to[R=R1] ++(0,2) coordinate(A) \showcoord(A)<45:0.2>  to[R=R
      2] ++(0,2) coordinate(B) \showcoord(B)<45:0.2>
  -- ++(2,0) coordinate(C) \showcoord(C)<45:0.2> (C |- ref) coordinate(C1) \showcoord(C1)<45:0.2> -- (ref);
  \draw (C) ++(7,0)  coordinate(D) \showcoord(D)<45:0.2> -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4]
      ++(0,3) -- ++(0.5,0)  coordinate(E) \showcoord(E)<45:0.2>;
  \draw (C) node[Quad,anchor=1+,y fit to={C}{C1},outer x fit to={C}{D}]{};
\end{tikzpicture}}
```

LaTeX Result:

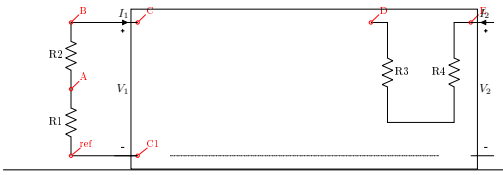


Fitting some circuit inside the Quadripole (nodes C and E):

LaTeX Code:

```
\resizebox{0.4\textwidth}{!}{
\begin{tikzpicture}
  \draw (0,0) coordinate(ref) \showcoord(ref)<45:0.2> to[R=R1] ++(0,2) coordinate(A) \showcoord(A)<45:0.2>  to[R=R
      2] ++(0,2) coordinate(B) \showcoord(B)<45:0.2>
  -- ++(2,0) coordinate(C) \showcoord(C)<45:0.2> (C |- ref) coordinate(C1) \showcoord(C1)<45:0.2> -- (ref);
  \draw (C) ++(7,0)  coordinate(D) \showcoord(D)<45:0.2> -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4]
      ++(0,3) -- ++(0.5,0)  coordinate(E) \showcoord(E)<45:0.2>;
  \draw (C) node[Quad,anchor=inner 1+,y fit to={C}{C1},inner x fit to={C}{E}]{};
\end{tikzpicture}}
```

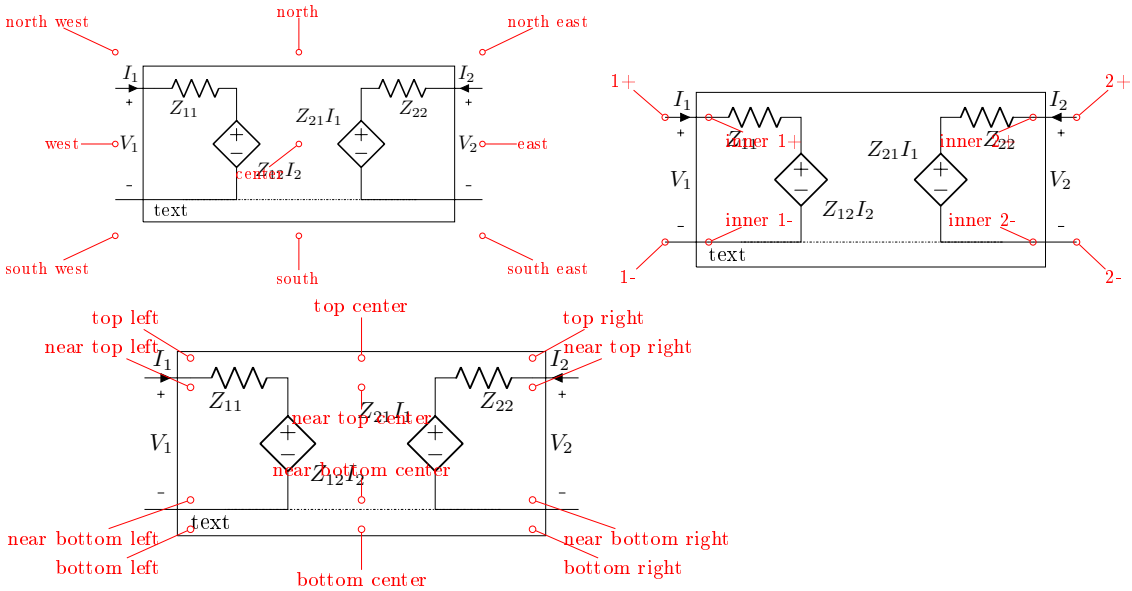
LaTeX Result:



3.4 Quad Z

This shape, besides the base anchors (see 3) it has 4 internal nodes: *<node>-Z11*, *<node>-Z12*, *<node>-Z21* and *<node>-Z22* and each of those sub-nodes has geographic anchors as defined at 2.1.

> ***Note:*** There is also a *ToQuad Z* to be used in a *to[ ]* path, in which case the key *outer x fit to* style will be triggered with the starting and ending points of the *to[ ]* path.



3.4.1 Quad Z keys

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
I2	Initial value:`$I_2$`
V1	Initial value:`$V_1$`

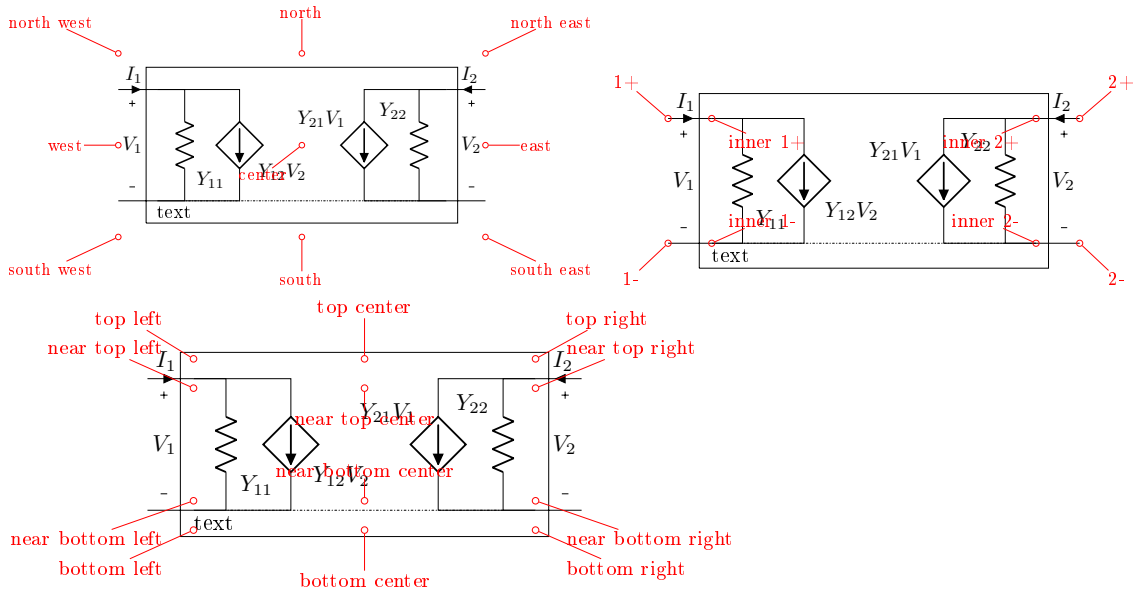
V2	Initial value:`$V_2$`
Z11	Initial value:`$Z_{11}$`
Z12	Initial value:`$Z_{12}$`
Z21	Initial value:`$Z_{21}$`
Z22	Initial value:`$Z_{22}$`
Z11 label pos	changes the label position. Defaults to: `{south west}{top left}`
Z12 label pos	changes the label position. Defaults to: `{south east}{top left}`
Z21 label pos	changes the label position. Defaults to: `{north west}{bottom right}`
Z22 label pos	changes the label position. Defaults to: `{south east}{top right}`

Note: The label pos keys expects two anchor names (... label pos=`{`⟨anchor A⟩`}{`⟨anchor B⟩`}`). The first anchors refers the sub-shape node and the second anchor is the text one.

3.5 Quad Y

This shape, besides the base anchors (see 3) it has 4 internal nodes: *<node>-Y11*, *<node>-Y12*, *<node>-Y21* and *<node>-Y22* and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a *ToQuad Y* to be used in a *to[ ]* path, in which case the key *outer x fit to* style will be triggered with the starting and ending points of the *to[ ]* path.



3.5.1 Quad Y keys

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
I2	Initial value:`$I_2$`
V1	Initial value:`$V_1$`
V2	Initial value:`$V_2$`
Y11	Initial value:`$Y_{11}$`
Y12	Initial value:`$Y_{12}$`
Y21	Initial value:`$Y_{21}$`

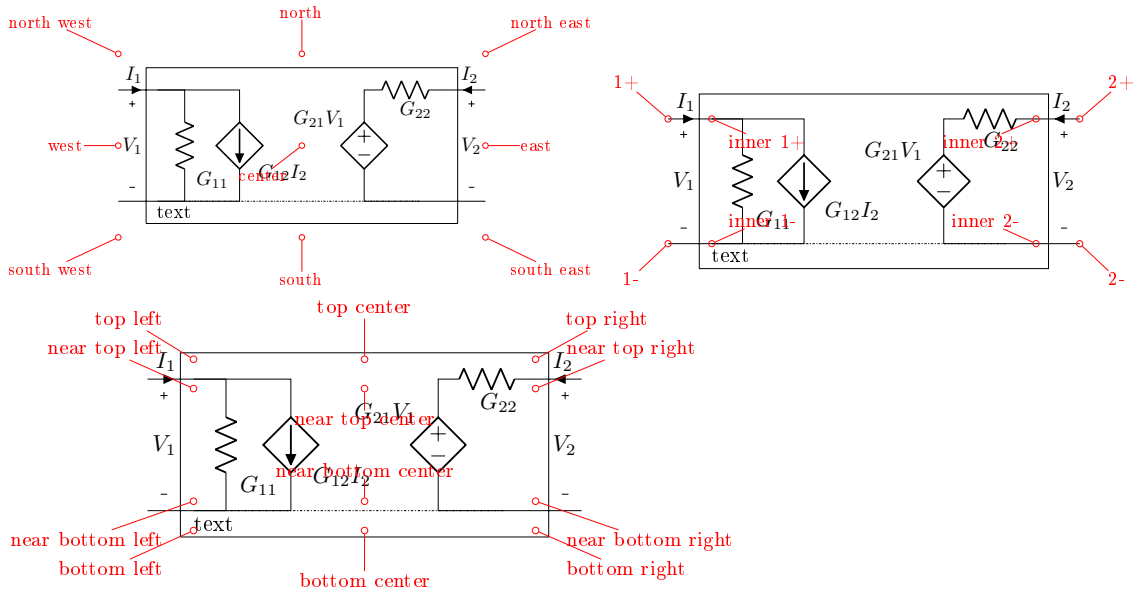
Y22	Initial value:`$Y_{22}$`
Y11 label pos	changes the label position. Defaults to: `{south west}{top left}`
Y12 label pos	changes the label position. Defaults to: `{south east}{top left}`
Y21 label pos	changes the label position. Defaults to: `{north west}{bottom right}`
Y22 label pos	changes the label position. Defaults to: `{north west}{bottom right}`

Note: The label pos keys expects two anchor names (... label pos=`{⟨anchor A⟩}{⟨anchor B⟩}`). The first anchors refers the sub-shape node and the second anchor is the text one.

3.6 Quad G

This shape, besides the base anchors (see 3) it has 4 internal nodes: *<node>-G11*, *<node>-G12*, *<node>-G21* and *<node>-G22* and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a *ToQuad G* to be used in a *to[ ]* path, in which case the key *outer x fit to* style will be triggered with the starting and ending points of the *to[ ]* path.



3.6.1 Quad G keys

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
I2	Initial value:`$I_2$`
V1	Initial value:`$V_1$`
V2	Initial value:`$V_2$`
G11	Initial value:`$G_{11}$`
G12	Initial value:`$G_{12}$`
G21	Initial value:`$G_{21}$`
G22	Initial value:`$G_{22}$`
G11 label pos	changes the label position. Defaults to: `{south west}{top left}`
G12 label pos	changes the label position. Defaults to: `{south east}{top left}`
G21 label pos	changes the label position. Defaults to: `{north west}{bottom right}`

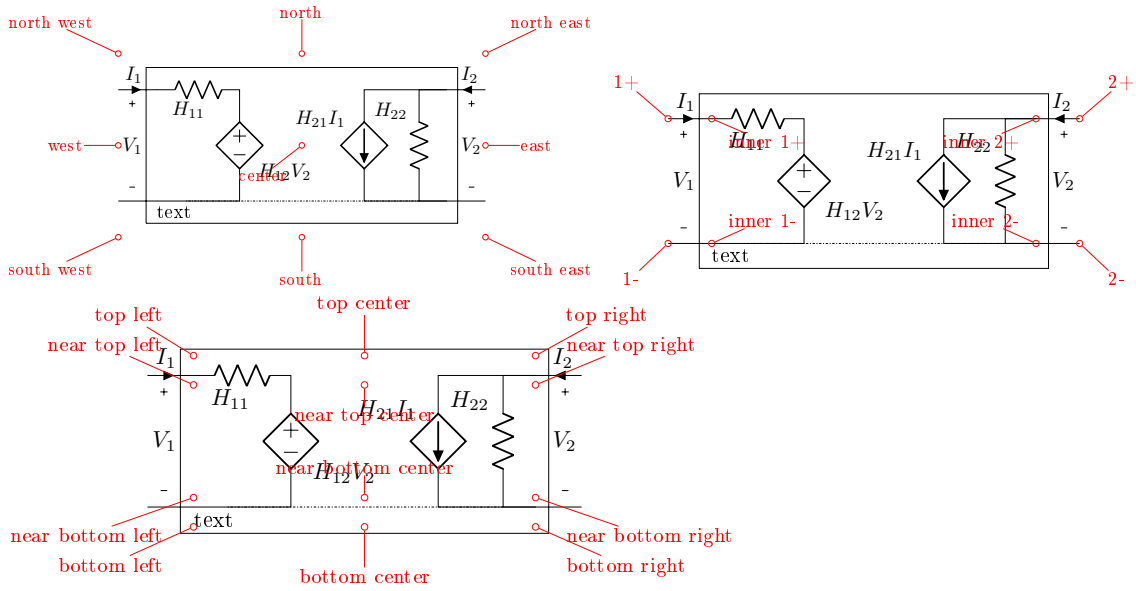
G22 label pos changes the label position. Defaults to: `{south east}{top right}`

Note: The label pos keys expects two anchor names (... label pos={⟨anchor A⟩}{⟨anchor B⟩}). The first anchors refers the sub-shape node and the second anchor is the text one.

3.7 Quad H

This shape, besides the base anchors (see 3) it has 4 internal nodes: *<node>-H11*, *<node>-H12*, *<node>-H21* and *<node>-H22* and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a *ToQuad H* to be used in a *to[ ]* path, in which case the key *outer x fit to* style will be triggered with the starting and ending points of the *to[ ]* path.



3.7.1 Quad H keys

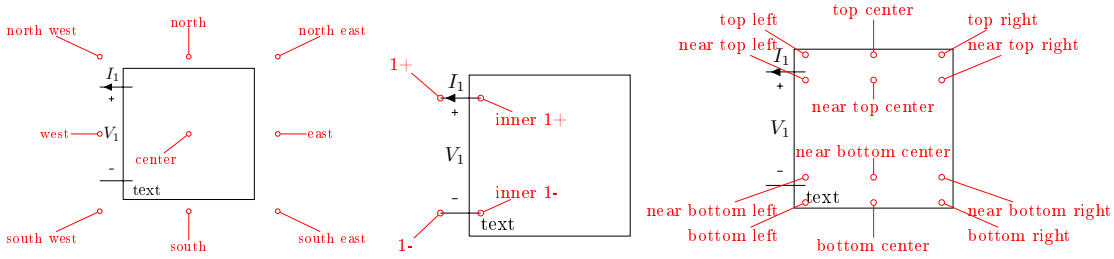
name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
I2	Initial value:`$I_2$`
V1	Initial value:`$V_1$`
V2	Initial value:`$V_2$`
H11	Initial value:`$H_{11}$`
H12	Initial value:`$H_{12}$`
H21	Initial value:`$H_{21}$`
H22	Initial value:`$H_{22}$`
H11 label pos	changes the label position. Defaults to: `{south west}{top left}`
H12 label pos	changes the label position. Defaults to: `{south east}{top left}`
H21 label pos	changes the label position. Defaults to: `{north west}{bottom right}`
H22 label pos	changes the label position. Defaults to: `{north west}{bottom right}`

Note: The label pos keys expects two anchor names (... label pos={⟨anchor A⟩}{⟨anchor B⟩}). The first anchors refers the sub-shape node and the second anchor is the text one.

4 Thevenin, Norton single port dipoles

4.1 The Base Black Box

The base shape just draws a base box and sets some connection anchors: 1+, 1−, *inner* 1+, *inner* 1−, besides the geographic and text ones:



4.2 Customization keys

Additionally, one has:

base width — The 'box' width

half base width — Ditto, half width. Initial value:`2\pgf@circ@Rlen`.

base height — The distance between *1+* and *1-*. The 'box' full height is equal to 2*(*half base height* + *height ext* + *height ext+*).

half base height — Ditto, half height. Initial value:`\pgf@circ@Rlen/7`

height ext — Initial value:`2\pgf@circ@Rlen/7`

height ext+ — Initial value:0

inner ext — distance between the 'box' and `inner1+/1-/2+/2-`. initial value: `\pgf@circ@Rlen/7`

outer ext — distance between the 'box' and `1+/1-/2+/2-`. initial value: `5\pgf@circ@Rlen/14`

inner marks — If set, the inner anchors will be marked.

outer marks — If set, the outer anchors will be marked.

invert — The shape will be inverted, more or less like 'x scale=-1'.

alt, opt — Case a Voltage source is zero, a series impedance will be draw vertically.

outer x fit=`{⟨CoordA⟩}{⟨CoordB⟩}`. The width will be set so that ⟨1+⟩ and ⟨2+⟩ (or ⟨1-⟩ and ⟨2-⟩, depending on the used anchor) will fit ⟨CoordA⟩ and ⟨CoordB⟩

outer x fit to

inner x fit=`{⟨CoordA⟩}{⟨CoordB⟩}`. The width will be set so that ⟨inner 1+⟩ and ⟨inner 2+⟩ (or ⟨inner 1-⟩ and ⟨inner 2-⟩, depending on the used anchor) will fit ⟨CoordA⟩ and ⟨CoordB⟩

inner x fit to

y fit=`{⟨CoordA⟩}{⟨CoordB⟩}`. In the case of a quadripole, the distance between, lets say *1+* and *1-* will be made equal to the distance between *CoordA* and *CoordB*. In the case of a Thevenin/Norton, *1+* and *1-* will fit *CoordA* and *CoordB* respectively.

y fit to

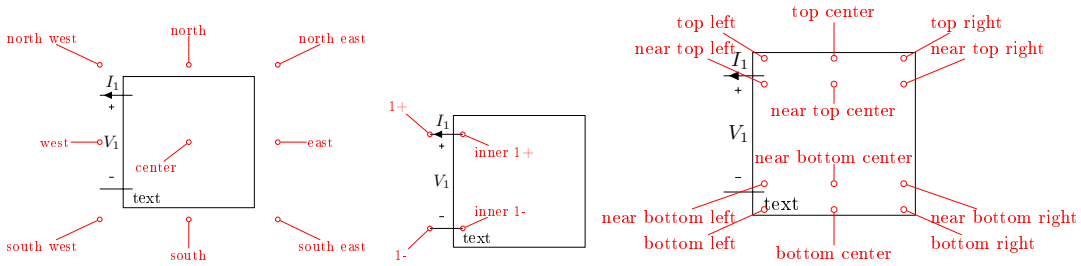
Note: *outer x fit* and *inner x fit* might result in a shape rotation. *y fit* in case of a quadripole will never result in a rotation, while in case of a thevenin/norton it might.

Note: Those keys can be used with all the following components: ⟨Quad⟩, ⟨Quad Z⟩, ⟨Quad Y⟩, ⟨Quad G⟩, ⟨Quad H⟩, ⟨ToQuad⟩, ⟨ToQuad Z⟩, ⟨ToQuad Y⟩, ⟨ToQuad G⟩, ⟨ToQuad H⟩, ⟨Black Box⟩, ⟨Thevenin⟩, ⟨Norton⟩, ⟨ToBlack Box⟩, ⟨ToThevenin⟩ and ⟨ToNorton⟩.

4.3 Black Box, BB

This is just the base shape, to be used in cases whereas one just want to emphasises part of a circuit (using, for instance, the *inner x fit to* key, or just mark a single port black box.

> ***Note:*** There is also a *ToBlack Box* to be used in a *to[ ]* path, in which case the key *y fit to* style will be triggered with the starting and ending points of the *to[ ]* path.



4.3.1 Black Box keys

name	⟨node-name⟩, when using a to[] path.
I1	Initial value:I_1
V1	Initial value:V_1

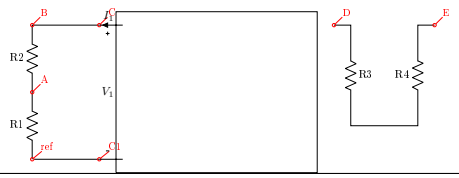
4.4 Example of fit to uses

Squeezing a Black Box between two parts of a circuit (nodes C and D):

LaTeX Code:

```
\resizebox{0.4\textwidth}{!}{
\begin{tikzpicture}
  \draw (0,0) coordinate(ref) \showcoord(ref)<45:0.2> to[R=R1] ++(0,2) coordinate(A) \showcoord(A)<45:0.2>  to[R=R
      2] ++(0,2) coordinate(B) \showcoord(B)<45:0.2>
  -- ++(2,0) coordinate(C) \showcoord(C)<45:0.2> (C |- ref) coordinate(C1) \showcoord(C1)<45:0.2> -- (ref);
  \draw (C) ++(7,0)  coordinate(D) \showcoord(D)<45:0.2> -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4]
      ++(0,3) -- ++(0.5,0)  coordinate(E) \showcoord(E)<45:0.2>;
  \draw (C) node[Black Box,anchor=1+,y fit to={C}{C1},outer x fit to={C}{D}]{};
\end{tikzpicture}
}
```

LaTeX Result:

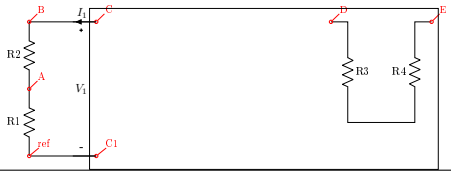


Fitting some circuit inside the Black Box (nodes C and E):

LaTeX Code:

```
\resizebox{0.4\textwidth}{!}{
\begin{tikzpicture}
  \draw (0,0) coordinate(ref) \showcoord(ref)<45:0.2> to[R=R1] ++(0,2) coordinate(A) \showcoord(A)<45:0.2>  to[R=R
      2] ++(0,2) coordinate(B) \showcoord(B)<45:0.2>
  -- ++(2,0) coordinate(C) \showcoord(C)<45:0.2> (C |- ref) coordinate(C1) \showcoord(C1)<45:0.2> -- (ref);
  \draw (C) ++(7,0)  coordinate(D) \showcoord(D)<45:0.2> -- ++(0.5,0) to[R=R3] ++(0,-3) -- ++(2,0) to[R=R4]
      ++(0,3) -- ++(0.5,0)  coordinate(E) \showcoord(E)<45:0.2>;
  \draw (C) node[Black Box,anchor=inner 1+,y fit to={C}{C1},inner x fit to={C}{E}]{};
\end{tikzpicture}
}
```

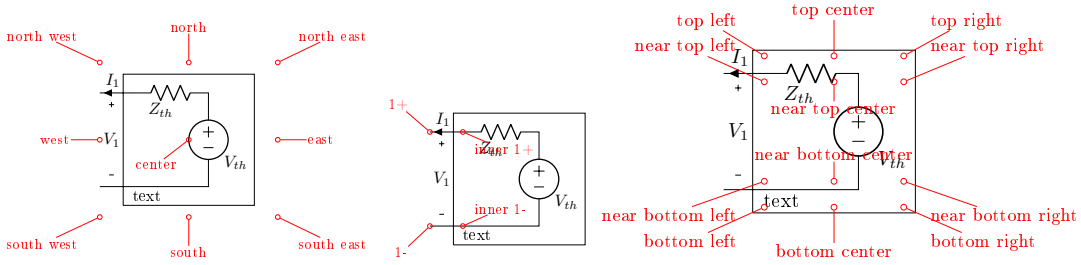
LaTeX Result:



4.5 Thevenin

This is the classical Thevenin circuit. Besides the base anchors (see 4.1) it has 2 internal nodes: *<node>-Zth* and *<node>-Vth* and each of those sub-nodes has geographic anchors as defined at 2.1.

> ***Note:*** There is also a *ToThevenin* to be used in a *to[ ]* path, in which case the key *y fit to* style will be triggered with the starting and ending points of the *to[ ]* path.



4.5.1 Thevenin keys

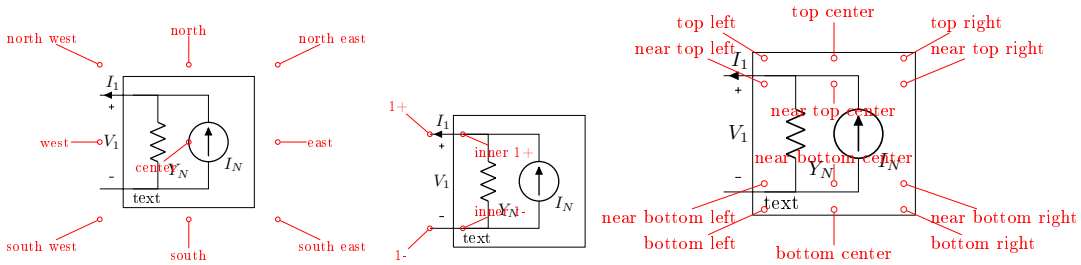
name	⟨node-name⟩, when using a to[] path.
I1	Initial value:`$I_1$`
V1	Initial value:`$V_1$`
Zth	Initial value:`$Z_{th}$`
Vth	Initial value:`$V_{th}$`
Zth label pos	changes the label position. Defaults to: `{south west}{top left}`
Vth label pos	changes the label position. Defaults to: `{south east}{top left}`

> ***Note:*** The label pos keys expects two anchor names (... label pos={⟨anchor A⟩}{⟨anchor B⟩}). The first anchors refers the sub-shape node and the second anchor is the text one.

4.6 Norton

This is the classical Norton circuit. Besides the base anchors (see 4.1) it has 2 internal nodes: `<node>-Yn` and `<node>-In` and each of those sub-nodes has geographic anchors as defined at 2.1.

Note: There is also a `ToNorton` to be used in a `to[ ]` path, in which case the key `y fit to` style will be triggered with the starting and ending points of the `to[ ]` path.



4.6.1 Norton keys

`name`	⟨node-name⟩, when using a to[] path.
`I1`	Initial value:`$I_1$`
`V1`	Initial value:`$V_1$`
`Yn`	Initial value:`$Y_{N}$`
`In`	Initial value:`$I_{N}$`
`Yn label pos`	changes the label position. Defaults to: `{south west}{top left}`
`In label pos`	changes the label position. Defaults to: `{south east}{top left}`

Note: The label pos keys expects two anchor names (... label pos=`{`⟨`anchor A`⟩`}{`⟨`anchor B`⟩`}). The first anchors refers the sub-shape node and the second anchor is the text one.