

The tikzdotncross Package

Marking Coordinates and Crossing Paths

Version 1.3

Alceu Frigeri*

October 2025

Abstract

This package offers a few alternative ways for declaring and marking coordinates and drawing a line with “jumps over” an already given path, which is a quite common issue when drawing, for instance, Electronics Circuits, e.g. *CircuiTikZ*.

1 Introduction

One recurring problem when drawing circuits in general is how to interpret a crossing line. There are many conventions, notably, for the old school (like the author of this) a jump denotes “non touching lines” while a simple cross is a connection, more recently (like the past 25 years), the winning convention has been that a dot marks a connection, whilst a simple cross denotes “non touching lines”. Many, for the sake of staying in the safe side of the wall, mark a connection with dots and non touching lines with a jump, which is a bit overkill, but at least there is no margin for interpretation errors.

And that’s it, this package defines some commands to mark/pin a connection, declaring a coordinate and node at the same spot, for later reference, and a command to draw a line jumping over crossing lines of a pre-existent path.

2 Package Options

`pinsize` pin (circle) size (default: 1.2), in pt.
`pinang` pin angle (default: 45).
`pincolor` pin color (default: blue).
`pinlength` pin length (default: 4), in pt.
`coordcolor` coordinate color (default: red), used if `\showcoordtrue`.

Those can also be set, middle code, via:

```
\setpindefaults \setpindefaults {<options as above>}
```

new: 2024/10/22

3 Declaring and Marking Coordinates/Nodes

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 `n<coord>`.

```
\showcoordtrue \showcoordtrue  
\showcoordsfalse \showcoordsfalse
```

These will affect how `\ncoord`, `\dotcoord` and `\odotcoord` will behave, with `\showcoordtrue` a red pin will also be added to the newly defined coordinate/node. The initial state is `\showcoordsfalse`. It can be turned on/off as needed.

*<https://github.com/alceu-frigeri/tikzdotncross>

`\showcoords` `\showcoords {<val>}`

new: 2025/10/29

Alternative form to set newly defined coordinates visibility. If `<val>` is either `on` or `true` this will be equivalent to `\showcoordstrue`, otherwise if `<val>` is either `off` or `false` this will be equivalent to `\showcoordsfalse`.

`\ncoord` `\ncoord(<coord>)`

`\pincoord` `\pincoord(<coord>)`
`\pincoord(<coord>, <color>)`
`\pincoord(<coord>, <color>, <angle>)`
`\pincoord(<coord>, <color>, <angle>, <length>)`

The `\ncoord` always expects a single parameter `<coord>`. A coordinate named `<coord>` and node named `n<coord>` (a “n” is added as a prefix) will be created for later use/reference. If `\showcoordstrue` is en force, it will also add a pin.

The `\pincoord` always draws a pin, besides declaring a coordinate and node as `\ncoord`. It expects one to 4 parameters, as listed. If omitted, the default length is 4 (unit: pt), the default angle is -45 (degrees), the default color is blue. Likewise, if `\showcoordstrue`, `\ncoord(name)` is just a short cut for `\pincoord (name,red,45)`.

Note: Those defaults can be changed via package options, see 2, or `\setpindefaults`.

`\dotcoord` `\dotcoord(<coord>)`

`\dotpincoord` `\dotpincoord(<coord>)`
`\dotpincoord(<coord>, <color>)`
`\dotpincoord(<coord>, <color>, <angle>)`
`\dotpincoord(<coord>, <color>, <angle>, <length>)`

These are the same as `\ncoord` and friends, just adding a dot (a filled in, small circle) at the coordinate.

`\odotcoord` `\odotcoord(<coord>)`

`\odotpincoord` `\odotpincoord(<coord>)`
`\odotpincoord(<coord>, <color>)`
`\odotpincoord(<coord>, <color>, <angle>)`
`\odotpincoord(<coord>, <color>, <angle>, <length>)`

These are the same as `\ncoord` and friends, just adding an open dot (a small circle filled with white) at the coordinate.

4 Crossing Paths

`\pathcross` `\pathcross* [<cross-name>] {<coordA>} {<coordB>} {<path-name>} [<width>]`

This will draw a line from `<coordA>` to `<coordB>` “jumping over” any pre-existent (soft) path named `<path-name>`. First of, the reference path `<path-name>` has to be defined using the `name path=<path-name>` key (`name path=<path-name>`).

Then this will “calculate” the intersections between the line (defined by the coordinates `(<coordA>)` and `(<coordB>)`) and the path named `<path-name>`. At each intersection a coordinate named `(<cross-name>-i)` and a node `(n<cross-name>-i)` will be defined (i goes from 1 up to the number of crossings detected.) A macro named `<cross-name>T` will have the number of crossings found.

At each intersection a semi-circle will be drawn, and finally a line will be draw connecting `<coordA>` to `<coordB>` over all intermediate nodes.

The star version flips the semi-circles orientation.

Note: The default `<cross-name>` is “cross”. It may contain only characters, as any valid T_EX macro name. The default `<width>` of the semi-circle is 7pt.

Note: This is based on the `tikz` library `intersections`, inheriting it’s limitations. The main one: It only detects crossings over “soft paths”, this means, if the line defined by `<coordA>` and `<coordB>` crosses over a node, it will, in most cases, miss it (depends on how the node is draw and interacts with the soft path system).

Note: When using the crossing coordinates, like `(<cross-name>-i)`, be aware that in some ill-defined cases, `intersections` might detect a crossing either at the starting and/or ending points. `\pathcross` accounts for that, but you will be left with some extra reference coordinates, either the first one, last one or both.

5 Some Examples

Note: In the examples below, the circuit doesn't make much/any sense, it is just a way to show the commands possibilities.

A first example with `\showcoords{true}` (showing all coordinates defined with `\ncoord`).

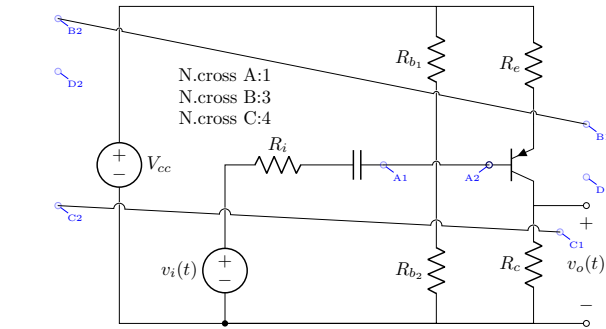
LaTeX Code:

```

1 \resizebox{0.5\textwidth}{!}{
2 \begin{tikzpicture}
3     %% This is the reference, named path
4     %%
5     \draw[name path=base circ]
6     (0,0) \dotncoord(A) to[V,invert,l=$v_i(t)$] ++(0,2) -- ++(0,1) \ncoord(Y)
7     to[R=$R_i$] ++(2,0)
8     to[C] ++(1,0) \pincoord(A1) ++(1,0) \ncoord(B)
9     ++(1,0) node[pnp,anchor=B] (T1){}
10    (A) -- (A|-B) to[R=$R_{b_2}$] ++(0,2) \ncoord(Bb) (B) ++(0,1) \ncoord(Cb) to[R=$R_{b_1}$] ++(0,2) \ncoord(C)
11    (T1.C) to[R,l=$R_c$] (T1.C|-A) -- (A)
12    (T1.E) to[R,l=$R_e$] (T1.E|-C) -- (C|-A) -- ++(-2,0) \ncoord(X) to[V,l=$V_{cc}$] (X|-A) -- (A)
13    (T1.C) -- ++(1,0) node[ocirc]{} \ncoord(k) to[open,v=$v_o(t)$] (k|-A) node[ocirc]{} -- (A)
14    (Bb) -- (Cb)
15    ;
16    %% These are just a few, marked, coords (they could be part of the previous path)
17    %%
18    \path (T1.E) ++(1,0) \pincoord(B1) ++(-10,2) \pincoord(B2)
19    (B1) ++(0,-1) \pincoord(D1) (B2) ++(0,-1) \pincoord(D2)
20    (T1.C) ++(0.5,-0.5) \pincoord(C1) (T1.C) ++(-9,0) \pincoord(C2)
21    (T1.B) \odotpincoord(A2,blue,225)
22    ;
23    %% And that's all, a few crossing lines
24    %%
25    \pathcross[A1]{A2}{base circ}[4pt] \draw (Y) +(0,1.7) node{}{N.cross A:\crossT};
26    \pathcross*[B1]{B2}{base circ}[3pt] \draw (Y) +(0,1.3) node{}{N.cross B:\crossT};
27    \pathcross*[sec]{C1}{C2}{base circ}[6pt] \draw (Y) +(0,0.9) node{}{N.cross C:\secT};
28
29 \end{tikzpicture}
30 }

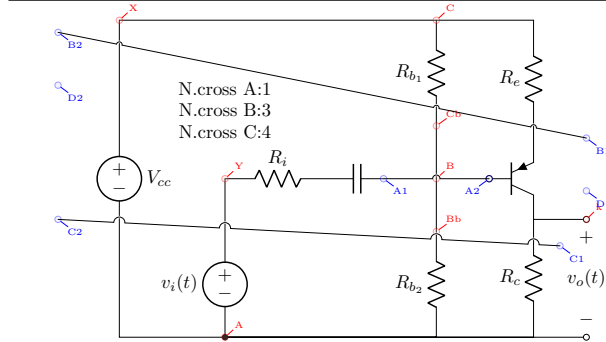
```

LaTeX Result:



And the same with `\showcoords{false}`

LaTeX Result:



As said, the main limitation (derived from how *intersections* works) is that crossings between the line and nodes might not be detected at all. For example, if someone tries to connect the nodes *D1* and *D2*, it will, unfortunately, fail detecting the node (pnp transistor) entirely:

LaTeX Code:

```

1 \resizebox{0.5\textwidth}{!}{
2 \begin{tikzpicture}
3     %% This is the reference, named path
4     %%
5     \draw[name path=base circ]
6     (0,0) \dotcoord(A) to[V,invert,l=$v_i(t)$] ++(0,2) -- ++(0,1) \ncoord(Y)
7     to[R=$R_i$] ++(2,0)
8     to[C] ++(1,0) \pincoord(A1) ++(1,0) \ncoord(B)
9     ++(1,0) node[pnp,anchor=B] (T1){}
10    (A) -- (A|-B) to[R=$R_{b2}$] ++(0,2) \ncoord(Bb) (B) ++(0,1) \ncoord(Cb) to[R=$R_{b1}$] ++(0,2) \ncoord(C)
11    (T1.C) to[R,l=$R_c$] (T1.C|-A) -- (A)
12    (T1.E) to[R,l=$R_e$] (T1.E|-C) -- (C|-A) -- ++(-2,0) \ncoord(X) to[V,l=$V_{cc}$] (X|-A) -- (A)
13    (T1.C) -- ++(1,0) node[ocirc]{} \ncoord(k) to[open,v=$v_o(t)$] (k|-A) node[ocirc]{} -- (A)
14    (Bb) -- (Cb)
15    ;
16    %% These are just a few, marked, coords (they could be part of the previous path)
17    %%
18    \path (T1.E) ++(1,0) \pincoord(B1) ++(-10,2) \pincoord(B2)
19    (B1) ++(0,-1) \pincoord(D1) (B2) ++(0,-1) \pincoord(D2)
20    (T1.C) ++(0.5,-0.5) \pincoord(C1) (T1.C) ++(-9,0) \pincoord(C2)
21    (T1.B) \pincoord(A2,blue,225)
22    ;
23    %% And that's all, a few crossing lines
24    %%
25    \pathcross[A1]{A2}{base circ}[4pt] \draw (Y) +(0,2) node{}{N.cross A:\crossT};
26    \pathcross[sec]{D2}{D1}{base circ}[6pt] \draw (Y) +(0,1.6) node{}{N.cross D:\secT};
27 \end{tikzpicture}
28 }

```

LaTeX Result:

