The tikzdotncross Package
Marking Coordinates and Crossing Paths
Version 1.1
Alceu Frigeri*
January 2024

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 in general is how to interpret a crossing line. There are many
conventions, notably, 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 are now marking a connection with dots and non
touching lines with a jump, which is a bit overkill, but at least there is no margin for error.
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 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).
\showcoordstrue
\showcoordsfalse
These will affect \coord, \dotcoord and \odotcoord will behave, with \showcoordstrue 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.
\coord\pincoord
\coord((coord))
\pincoord((coord))
\pincoord((coord), (color))
\pincoord((coord), (color), (angle))
\pincoord((coord), (color), (angle), (distance))
The \coord 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 \coord. It expects
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.

*https://github.com/alceu-frigeri/tikzdotncross
These are the same as \ncoord and friends, just adding a dot (a filled in, small circle) at the coordinate.

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

3 Crossing Paths

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

Then this will “calculate” the intersections between the line (defined by the coordinates (\coordA) and (\coordB)) and the path named \pathname. At each intersection a coordinate named (\crossname-i) and a node (\crossname-i) will be defined (i goes from 1 up to the number of crossings detected.) A macro named \crossnameT 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.

Note: The default \crossname is “cross”. It may contain only characters, as any valid \TeX 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 (\crossname-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.
4 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 `\showcoordstrue` (showing all coordinates defined with `\ncoord`).

\texttt{\LaTeX} Code:

```latex
\begin{tikzpicture}
% This is the reference, named path
\draw [name path=base circ]
(0,0) \dotcoord(A) to[V,invert,l=$v_i(t)$] ++(0,2) -- ++(0,1) \ncoord(Y)
to[C] ++(1,0) \pincoord(A1) ++(1,0) \ncoord(B)
++(1,0) node[\textcolor{red}{\nnp},anchor=E] (T1){}
(A) -- (\textcolor{red}{A} -l B) to[R=R_{b_2}] ++(0,2) \ncoord(Bb) (B) ++(0,1) \ncoord(Ob) to[R=R_{b_1}] ++(0,2) \ncoord(C)
\node[\textcolor{red}{\nnp}](T1.C){}
to[R,l=$R_c$] (T1.C |- A) -- (A)
(T1.E) to[R=R_{e}] (T1.E |- C) -- (C |- A) -- ++(-2,0) \ncoord(X) to[V,v=$V_{cc}$] (X |- A) node[\textcolor{red}{\ncirc}]{},

\node[\textcolor{red}{\nnp},anchor=B](T1){}
(A) -- (A -| B) to[R=R_{b_2}] ++(0,2) \ncoord(Bb) (B) ++(0,1) \ncoord(Ob) to[R=R_{b_1}] ++(0,2) \ncoord(C)
\node[\textcolor{red}{\nnp}](T1.C){}
to[R,l=$R_c$] (T1.C |- A) -- (A)
(T1.E) to[R=R_{e}] (T1.E |- C) -- (C |- A) -- ++(-2,0) \ncoord(X) to[V,v=$V_{cc}$] (X |- A) node[\textcolor{red}{\ncirc}]{},

\node[\textcolor{red}{\nnp},anchor=B](T1){}
(A) -- (A -| B) to[R=R_{b_2}] ++(0,2) \ncoord(Bb) (B) ++(0,1) \ncoord(Ob) to[R=R_{b_1}] ++(0,2) \ncoord(C)
\node[\textcolor{red}{\nnp}](T1.C){}
to[R,l=$R_c$] (T1.C |- A) -- (A)
(T1.E) to[R=R_{e}] (T1.E |- C) -- (C |- A) -- ++(-2,0) \ncoord(X) to[V,v=$V_{cc}$] (X |- A) node[\textcolor{red}{\ncirc}]{},

\node[\textcolor{red}{\nnp},anchor=B](T1){}
(A) -- (A -| B) to[R=R_{b_2}] ++(0,2) \ncoord(Bb) (B) ++(0,1) \ncoord(Ob) to[R=R_{b_1}] ++(0,2) \ncoord(C)
\node[\textcolor{red}{\nnp}](T1.C){}
to[R,l=$R_c$] (T1.C |- A) -- (A)
(T1.E) to[R=R_{e}] (T1.E |- C) -- (C |- A) -- ++(-2,0) \ncoord(X) to[V,v=$V_{cc}$] (X |- A) node[\textcolor{red}{\ncirc}]{},

\end{tikzpicture}
```

\texttt{\LaTeX} Result:

![Diagram](image1.png)

And the same with `\showcoordsfalsetrue`.

\texttt{\LaTeX} Result:

![Diagram](image2.png)
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:

\begin{tikzpicture}
\pathcross{A1}{A2}{base circ}[4pt] \draw (Y) +(0,2) node(){N.cross A:\crossT};
\pathcross[sec]{D2}{D1}{base circ}[6pt] \draw (Y) +(0,1.6) node(){N.cross B:\secT};
\end{tikzpicture}