

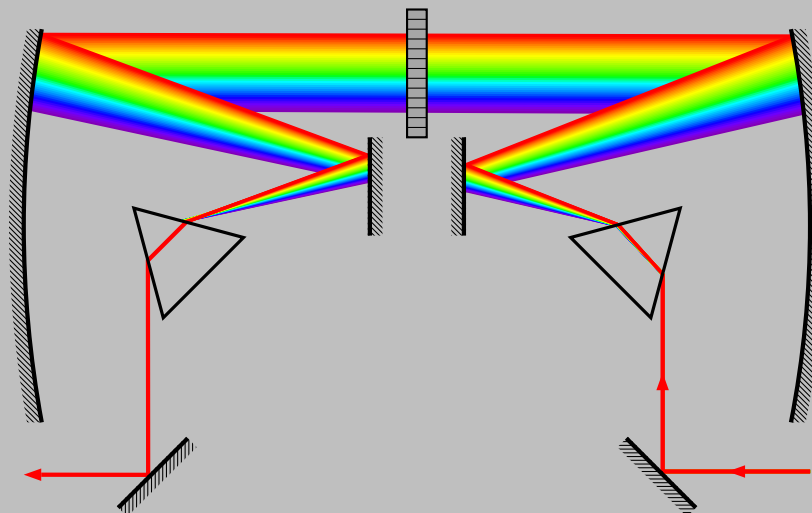
PSTricks

pst-optexp

Drawing optical experimental setups

v5.1

2014/11/19



Package author:
Christoph Bersch

Contents

1. Introduction	6
1.1. About the package	6
1.2. Requirements	6
1.3. Distribution and installation	6
1.4. License	7
1.5. Backward compatibility	7
1.6. Notation	7
1.7. Acknowledgements	8
2. Basic ideas	9
2.1. The components	9
2.2. Labels	10
2.3. Free-ray connections	10
2.4. Fiber connections	12
2.5. Step-by-step examples	13
2.5.1. Galilean telescope	13
2.5.2. Fiber setup	15
2.5.3. Fiber setup (alternative)	16
2.5.4. Michelson-Interferometer	17
3. General component parameters	19
3.1. Labels	19
3.2. Positioning	21
3.3. Rotating and shifting	22
3.4. Using psstyles	24
3.5. Component appearance	25
4. Free-ray components	27
4.1. Lens	27
4.2. Optical plate	29
4.3. Retardation plate	29
4.4. Pinhole	30
4.5. Box	30

4.6. Arrow component	31
4.7. Bar component	32
4.8. Optical source	33
4.9. Crystal	34
4.10. Optical diode	35
4.11. Dove prism	36
4.12. Glan–Thompson prism	36
4.13. Polarization	37
4.14. Mirror	38
4.15. Parabolic mirror	41
4.16. Off-axis parabolic mirror	42
4.17. Beamsplitter	43
4.18. Optical grating	44
4.19. Transmission grating	45
4.20. Acousto-optic modulator	46
4.21. Prism	50
4.22. Right-angle prism	51
4.23. Penta prism	52
5. Fiber components	53
5.1. Optical fiber	53
5.2. Optical amplifier	54
5.3. Mach–Zehnder modulator	55
5.4. Polarization controller	55
5.5. Isolator	55
5.6. Optical switch	56
5.7. Fiber delay-line	56
5.8. Polarizer	57
5.9. Optical circulator	57
5.10. Fiber coupler	58
5.10.1. Input and output nodes	61
5.10.2. Reference nodes	61
5.11. Fiber box	61
6. Electrical components	64
6.1. Coupler	64
6.2. Synthesizer	65
6.3. Mixer	66

7. Hybrid components	68
7.1. Optical filter	68
7.2. Fiber collimator	69
7.3. Detector	70
8. Special nodes	72
8.1. Component identifiers	73
8.2. Reference nodes	74
8.3. Center node	75
8.4. Label node	75
8.5. External nodes	75
8.6. Interface nodes	77
8.7. Rotation reference node	78
8.8. Beam nodes	79
8.9. Beam vector	81
8.10. Node overview	83
9. Connecting components	84
9.1. Accessing components	84
9.2. Drawing beams	85
9.2.1. Raytracing	86
9.2.2. Refractive index	88
9.2.3. Initial conditions	91
9.2.4. Internal beam path	93
9.2.5. Connecting with nodes	96
9.2.6. Automatic connections	98
9.2.7. Beam appearance	98
9.3. Drawing wide beams	100
9.3.1. Beam appearance	101
9.4. Error handling	102
9.5. Custom beams	103
9.6. Drawing fibers	107
9.6.1. Fiber angles	107
9.6.2. Fiber appearance	111
9.7. Drawing wires	112
9.8. Automatic fiber and wire connections	114
9.8.1. Appearance of automatic connections	116
9.8.2. Known limitations	118
9.9. Layers	118

10. Custom components	120
10.1. Customized versions of existing components	122
10.2. Defining new objects	123
10.2.1. The component drawing	124
10.2.2. Support for raytracing	125
10.2.3. Support for external nodes	126
10.2.4. Examples of custom components	127
11. Examples	128
12. Additional information	154
12.1. Internal component structure	154
12.2. Overview of special nodes	156
12.2.1. External and rotation reference nodes	156
12.2.2. Interface nodes	158
12.3. Backward compatibility	160
12.3.1. Version 4.3a	160
12.3.2. Version 4.3	160
12.3.3. Version 4.2	160
12.3.4. Version 4.0	161
12.3.5. Version 3.3	161
12.3.6. Version 3.0	162
Documentation index	165
A. Revision history	187

1. Introduction

1.1. About the package

The package `pst-optexp` is a collection of optical components that facilitate easy sketching of optical experimental setups. A lot of different free-ray and fiber components are provided, which alignment, positioning and labelling can be achieved in very simple and flexible ways. The components can be connected with fibers or beams, realistic raytraced beam paths are also possible.

1.2. Requirements

`pst-optexp` requires \LaTeX and recent versions of `pstricks` (≥ 2.53), `pst-plot` (≥ 1.61), `pst-node`, `multido`, `pstricks-add`, `pst-eucl`, `pst-intersect` (≥ 0.4) and `environ`.

All PSTricks package rely heavily on the Postscript language so that the typical workflow involves `latex`, `dvips`, and `ps2pdf`. Of course there are several alternative ways to compile your documents.¹

1.3. Distribution and installation

This package is available on CTAN² and is included in \TeX Live and \MiKTeX .

The `pst-optexp` package consists of the two main files `pst-optexp.ins` and `pst-optexp.dtx`. By running `tex pst-optexp.ins` the following derived files are generated:

- `pst-optexp.pro`: the Postscript prolog file

¹<http://tug.org/PSTricks/main.cgi?file=pdf/pdfoutput>

²<http://mirror.ctan.org/help/Catalogue/entries/pst-optexp.html>

- `pst-optexp.sty`: the \LaTeX style file

Save the files in a directory which is part of your local \TeX tree.

Do not forget to run `texhash` to update this tree. For $\text{MiK}\text{\TeX}$ users, do not forget to update the file name database (FNDB).

For more detailed information see the documentation of your personal \LaTeX distribution on installing packages to your local \TeX system.

1.4. License

Permission is granted to copy, distribute and/or modify this software under the terms of the \LaTeX Project Public License, version 1.3c.³ This package is author-maintained.

1.5. Backward compatibility

Version 3.0 introduced a lot of advanced features which made it too difficult to maintain full backward compatibility with version 2.1. Especially the `\drawbeam` macro was broken by design, so that it was best to drop the old flaw as early as possible.

You find a more detailed list of changes and information for migration of old documents in Sec. 12.3.

1.6. Notation

Often the parameter types are self-explanatory, but in some cases a clear distinction is needed, e.g. between $\langle num \rangle$ and $\langle psnum \rangle$. Tab. 1.1 explains some commonly used types.

³<http://www.latex-project.org/lppl.txt>

<i>Name</i>	description
<i>num</i>	float number
<i>psnum</i>	Postscript code which evaluates to a number
<i>int</i>	integer number
<i>dimen</i>	dimension
<i>psstyle</i>	custom graphics parameter configuration defined with <code>\newpsstyle</code>
<i>refpoint</i>	Like the reference point of <code>\rput</code> , can be any combination of c (center), t (top), b (bottom), l (left), and r (right).

Table 1.1.: Parameter types used in the parameter reference.

1.7. Acknowledgements

I thank all the people of the PSTricks mailinglist for the continuous help, especially Herbert Voß. Thanks also to various package authors from which I learned and adopted code for this package: David Carlisle, Florent Chervet, Enrico Gregorio, Rolf Niepraschk and Heiko Oberdiek. Christine Römer convinced me with her article in the german DTK⁴ to provide a german translation of the documentation. The documentation style is a mixture of the `pst-doc` class (Herbert Voß) and the `ltxdockit` package for the `biblatex` documentation (Philipp Lehman).

⁴Pakete in Deutsch dokumentieren. in Die TeXnische Komödie. Heft 2/2011, S. 28-35.

2. Basic ideas

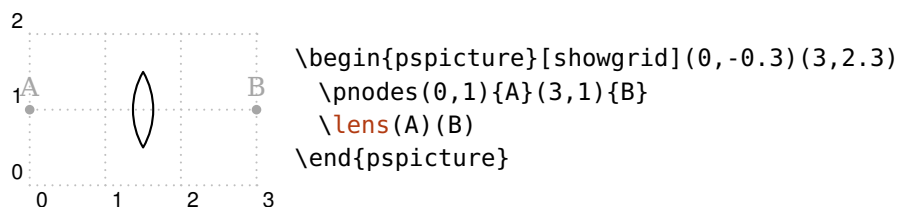
This chapter shows the basic ideas and concepts of this package. Starting with simple examples, basic functionality such as alignment, positioning (2.1) and labelling (2.2) of components is presented. Following, the connection of components with beam rays (2.3) or fiber (2.4) is demonstrated.

In Sec. 2.5 several propositions for preparing large experimental sketches are illustrated step by step. For a complete reference of all macros and their parameters, see Sec. 3–10.

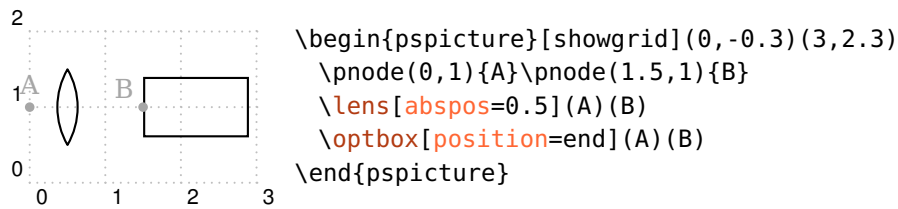
A drawing consists of components, which can be positioned, shifted and rotated based on their reference nodes, and may have an optional label. Then, the components are connected with beam rays or fibers.

2.1. The components

A component is positioned according to the nodes which are used for its definition—the reference nodes. In the following examples, the lens is placed centered between nodes $\langle A \rangle$ and $\langle B \rangle$:



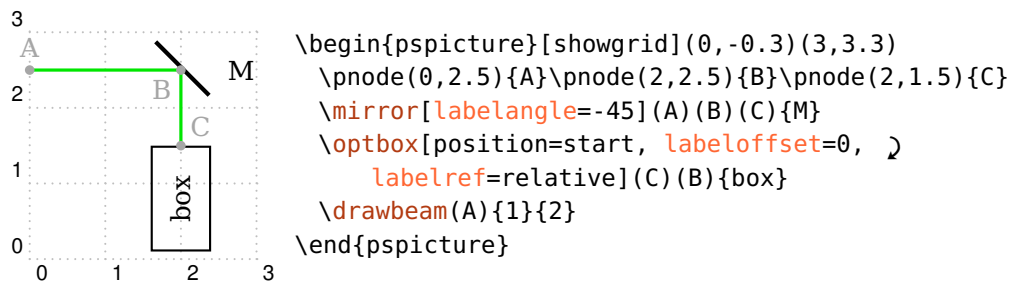
This position can be changed with a number of parameters, such as `abspos` (see Sec. 3.2 and Sec. 3.3). In the next example, the lens is positioned 0.5 units apart from node $\langle A \rangle$, the box is placed at the end of the reference line \overline{AB} .



The components can be divided in two categories: free-ray and fiber components. These differ only in their connection possibilities: Free-ray components can have fiber or free-ray connections, whereas fiber components support fiber connections only.

2.2. Labels

Every component can have an optional label, which is positioned relative to the component.

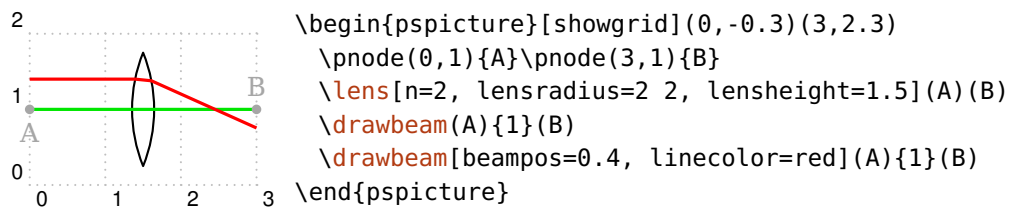


2.3. Free-ray connections

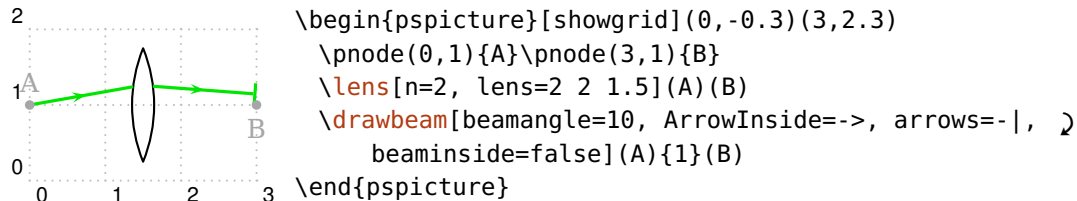
Free-ray components can be connected with beam rays, which path can be calculated by raytracing. Besides the realistic tracing possibilities, there are many ways to manipulate the beam paths because you are dealing with a package for *sketching* experimental setups.

In the simplest case a single line is drawn, which optical path is determined by the refractive index n and Snell's law.

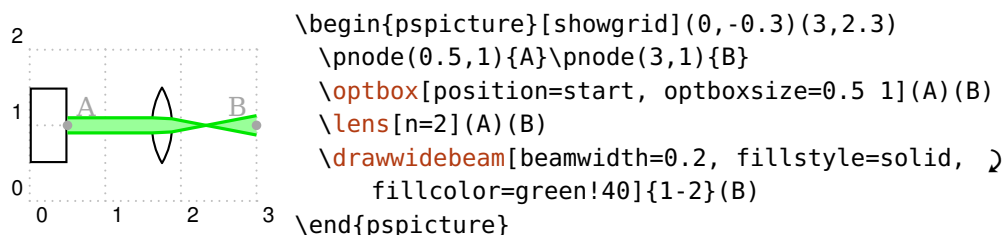
The example shows two beams which hit the lens with different initial conditions. The red beam starts at a distance of 0.2 from the optical axis (`beampos`) and is deflected by the lens accordingly.



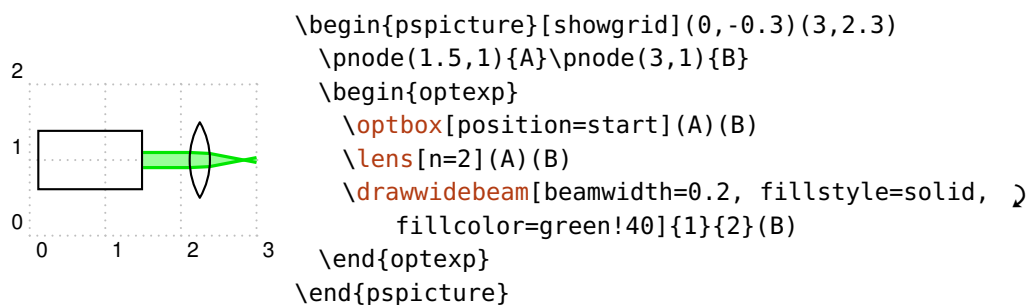
The beams can be decorated with arbitrary dashes and arrows.



You can also draw wide beams. The marginal rays can be changed with `linestyle` etc. and the filling with `fillstyle` etc.

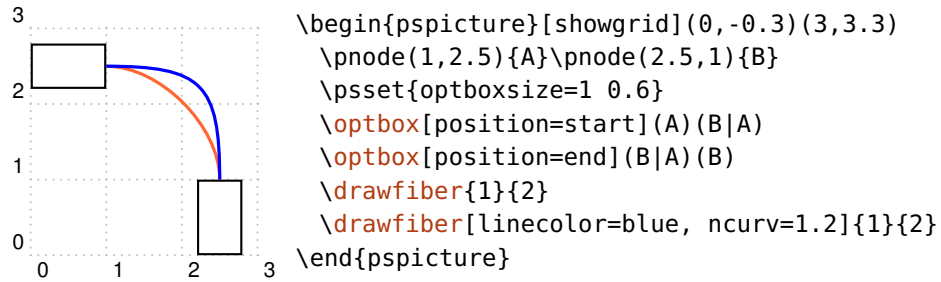


The beams can also lay behind the components. For this you must enclose all respective components and rays in a `optexp` environment (compare the previous example with the following one).

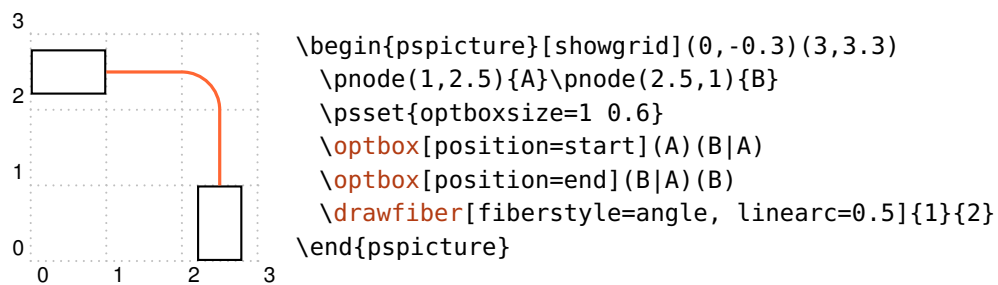


2.4. Fiber connections

Fibers (`\drawfiber`) are another possibility of connecting components. Here, you have several way of automatically adapting the fiber connection to the component alignment.



Usually, the fibers are drawn as `\ncurve` connections, other node connections are possible as well (`fiberstyle`).

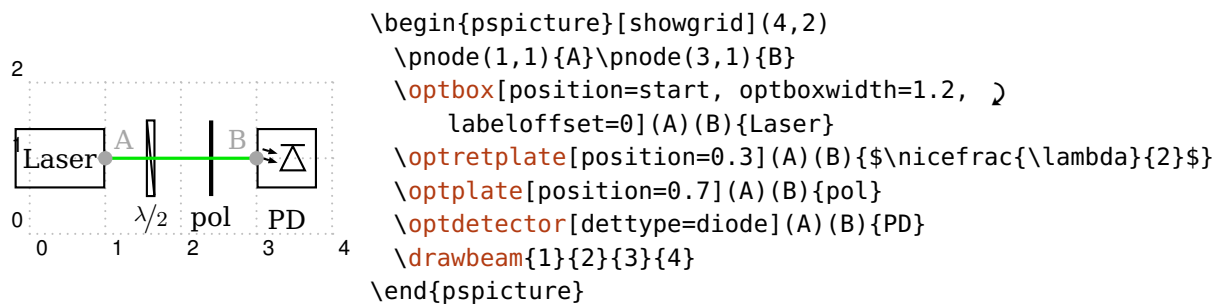


2.5. Step-by-step examples

In this first example, all components are distributed on a single line and connected with a beam.

First we define the start node $\langle A \rangle$ and the stop node $\langle B \rangle$. The first `\optbox` is positioned at the start, the wave plate and the polarizer are positioned between the nodes. The parameter `position` expects a value between 0 and 1 which is relative to the length of the reference line, or start/end. The detector is always positioned at the end of its reference line.

Finally, all components are connected with a simple beam ray.

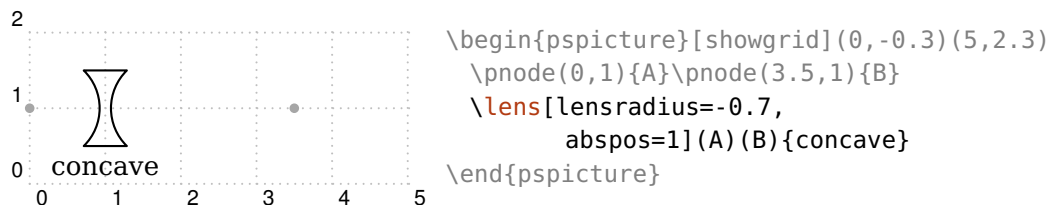


2.5.1. Galilean telescope

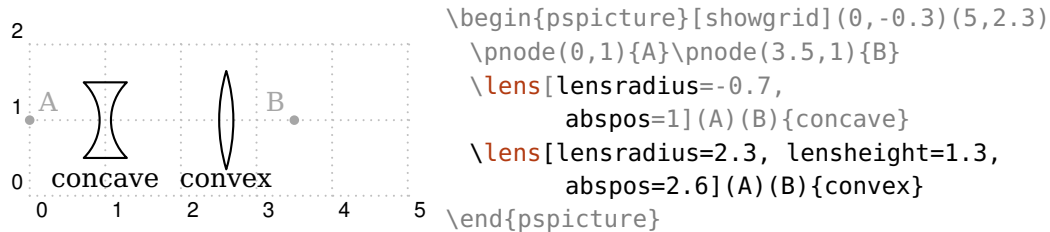
This example describes a simple setup with a concave and a convex lens and the raytracing through them.

First, we define the reference nodes $\langle A \rangle$ and $\langle B \rangle$ which are used for alignment of the components. For better overview, these nodes are highlighted and a grid is drawn.

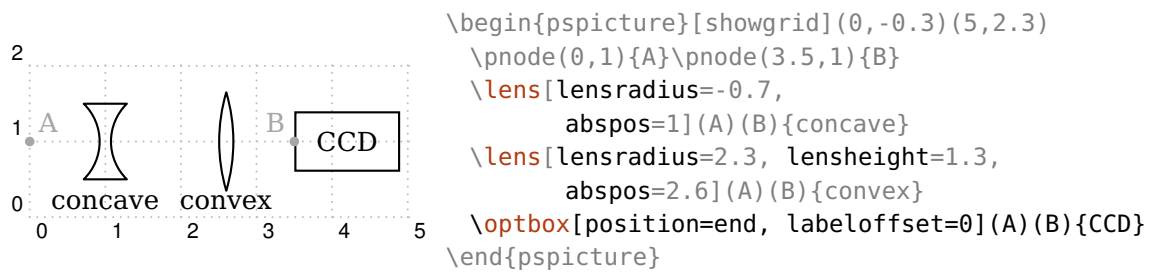
The first lens should have concave interfaces. With `lensradius` the interface radii are set to negative values, which correspond to concave curvatures. With `abspos` the lens is positioned one unit from $\langle A \rangle$ node.



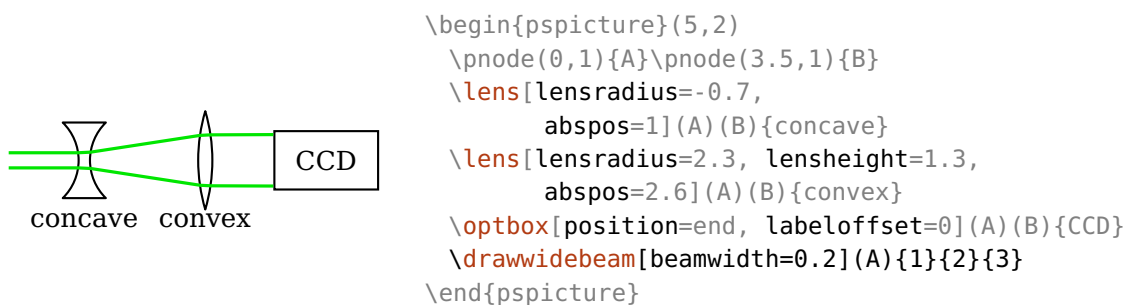
The second lens has a convex curvature (`lensradius` is positive) and is positioned 2.6 units apart from $\langle A \rangle$. Parameter `lensheight` sets the total height of the lens.



The beam rays are supposed to end at the CCD camera which is drawn as a `\optbox`. With `position=end` the box is positioned at the end of the reference line from $\langle A \rangle$ to $\langle B \rangle$. The label is put at the center of the component with `labeloffset`.

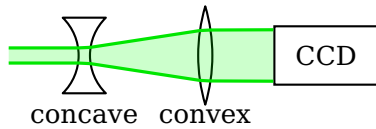


Finally, a wide beam with an initial beam width of 0.2 without divergence is sent through the components. The beam path is calculated by raytracing. The lens parameters in this example were set such, that for a collimated input beam also the output beam is collimated.



The appearance of the beam can be changed via the `Beam` style, and may have e.g. a semitransparent fill pattern.

Ex. 2.1



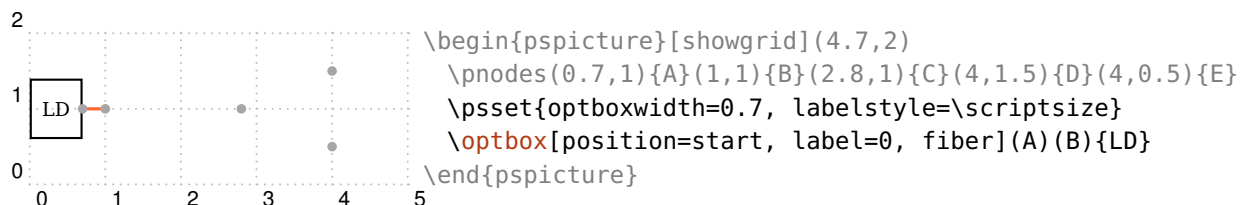
```
\begin{pspicture}(5,2)
  \pnode(0,1){A}\pnode(3.5,1){B}
  \lens[lensradius=-0.7,
    abspos=1](A)(B){concave}
  \lens[lensradius=2.3, lensheight=1.3,
    abspos=2.6](A)(B){convex}
  \optbox[position=end, labeloffset=0](A)(B){CCD}
  \addtopsstyle{Beam}{fillstyle=solid,
    fillcolor=green, opacity=0.2}
  \drawwidebeam[beamwidth=0.2](A){1}{2}{3}
\end{pspicture}
```

2.5.2. Fiber setup

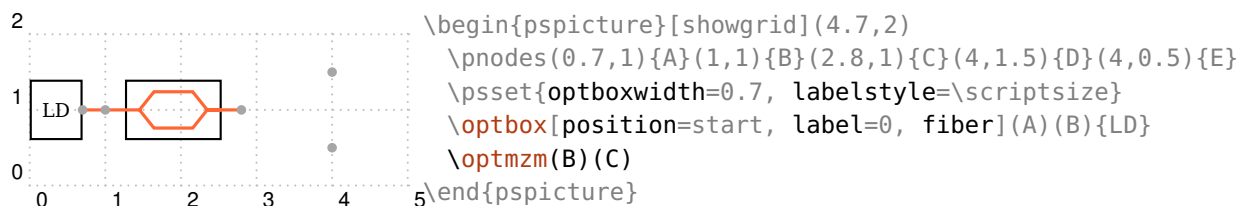
This example shows how to setup a fiber drawing with automatic fiber connections.

Again, the reference nodes are shown for better orientation. In contrast to the previous example, here, we need more reference nodes because the components are connected directly with them.

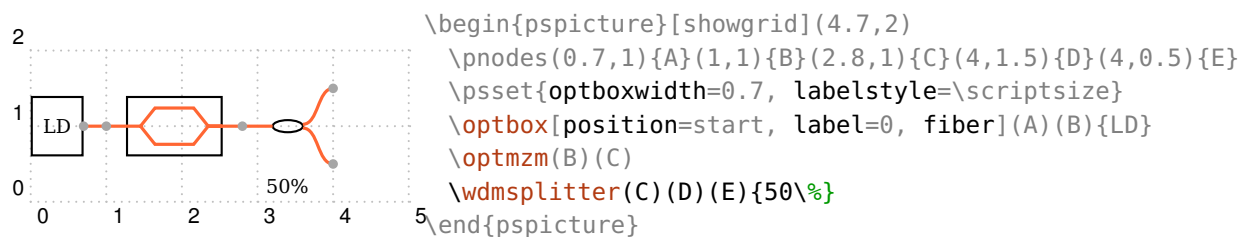
The laser is set at the start. The label offset is set with the `label` parameter, which would allow setting all label parameters at once. A `\optbox` is originally a free-ray component, so that for it the fiber connection must be enabled with `fiber`. The width of all `\optbox` is changed to 0.7.



The Mach-Zehner modulator is drawn without label and connected automatically with fibers to its reference nodes.



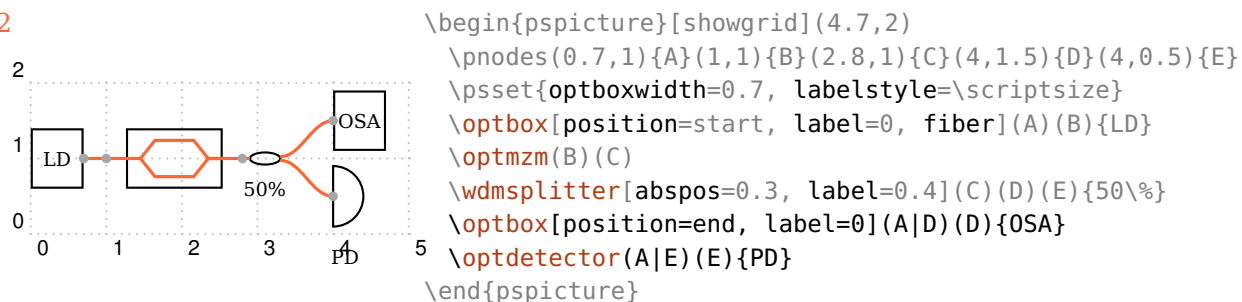
Then, we draw a fiber coupler with one input and two output fibers. The coupler is positioned between the input node and the center of the two output nodes.



At the ends of the coupler we put an optical spectrum analyzer (OSA) and a power meter (PD). The `\optdetector` is set by default at the end of the reference line, it has `position` preset to end. These two components are drawn without fibers, because their end nodes coincide with the two outputs of the coupler.

The notation $(A|E)$ uses the x -coordinate of $\langle A \rangle$ and the y -coordinate of $\langle E \rangle$.

Ex. 2.2



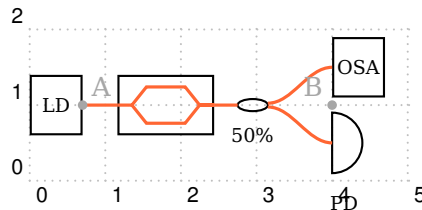
2.5.3. Fiber setup (alternative)

This example has the same result as the previous one (Ex. 2.2), but is constructed differently.

This time we do not use the automatic fiber connections, but draw all components first and connect them afterwards with fibers (`\drawfiber`). This requires less reference nodes, the fiber connections are suppressed globally with `fiber=none`.

The choice of the procedure depends on personal preferences, for loop setups, like for fiber lasers, it looks better if the fibers are drawn after the components with `\drawfiber`.

Ex. 2.3



```

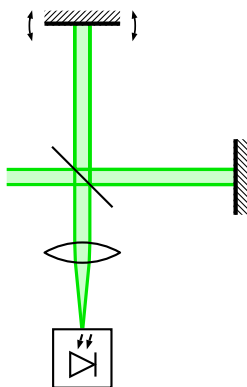
\begin{pspicture}[showgrid](4.9,2)
  \nnode(0.7,1){A}\nnode(4,1){B}
  \psset{optboxwidth=0.7, labelstyle=\scriptsize, 2
        fiber=none}
  \optbox[position=start, labeloffset=0](A)(B){LD}
  \optmzm[abspos=1.1](A)(B)
  \wdmsplitter[abspos=2.25, label=0.4](A)(B)(B){50\%}
  \optbox[position=end, label=0, 2
        compshift=0.5](A)(B){OSA}
  \optdetector[compshift=-0.5](A)(B){PD}
  \drawfiber{1-4}
  \drawfiber{3}{5}
\end{pspicture}

```

2.5.4. Michelson-Interferometer

In this example we use lenses, reflective elements (mirrors) and components which can be either transmittive or reflective (beamsplitter).

Ex. 2.4



```

1 \begin{pspicture}(3.2,5)
2   \pnodes(0,3){A}(1,3){BS}(3,3){M1}(1,5){M2}(1,1){PD}
3   \psset{mirrortype=extended, mirrordepth=0.2}
4   \begin{optexp}
5     \beamsplitter[bsstyle=plate, compname=BS](A)(BS)(PD)
6     \mirror[compname=M1](BS)(M1)(BS)
7     \mirror[compname=M2, variable](BS)(M2)(BS)
8     \lens[compname=L](BS)(PD)
9     \optdetector[compname=Det, dettype=diode](BS)(PD)
10    \addtopsstyle{Beam}{beamwidth=0.2, fillstyle=solid, 2
        fillcolor=green!20!white}
11    \drawwidebeam(A){BS}{M1}{BS}{M2}{BS}{L}{Det}
12  \end{optexp}
13 \end{pspicture}

```

We start with definition of the reference nodes (Line 2) and setting some basic parameters (Line 3).

Following, all components are positioned (Line 5–9). The reflective ones need three reference nodes, the input node, the node where the reflective interface is situated, and the output node. Then, the appearance of the beam is set (Line 10), and the beam is drawn (Line 11).

In this example, all components have a name (`compname`), which can be used instead of the numerical IDs—automatically assigned, unique numbers—for the raytracing. This can simplify following the beam path.

All components and beams are enclosed in a `optexp` environment. Inside of this, all beams are drawn behind the components.

3. General component parameters

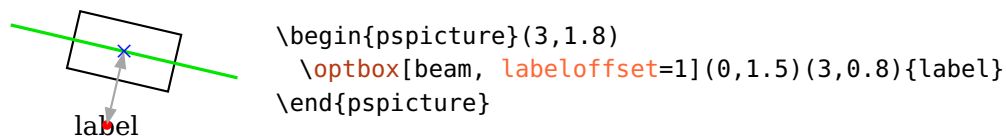
This chapter introduces generic parameters and concepts such as labels (3.1), positioning (3.2), rotating and shifting of components (3.3) and changing the appearance of components (3.4 and 3.5).

3.1. Labels

All components may have a label which position and alignment can be adjusted at will. The label is optional for every component. If the various parameters are not sufficient, you can access the label node—marked as red dot in some examples—directly (see also Sec. 8.4).

`labeloffset`= $\langle num \rangle$ default: 0.8

The offset of the label reference node (red dot) from the component center (blue cross).

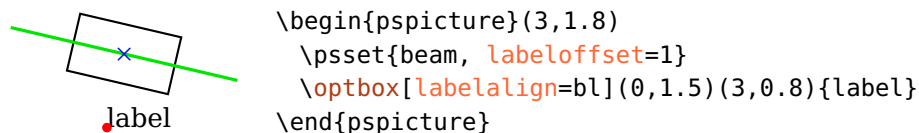


`labelstyle`= $\langle macros \rangle$

The text style that is used to typeset the label.

`labelalign`= $\langle refpoint \rangle$ default: c

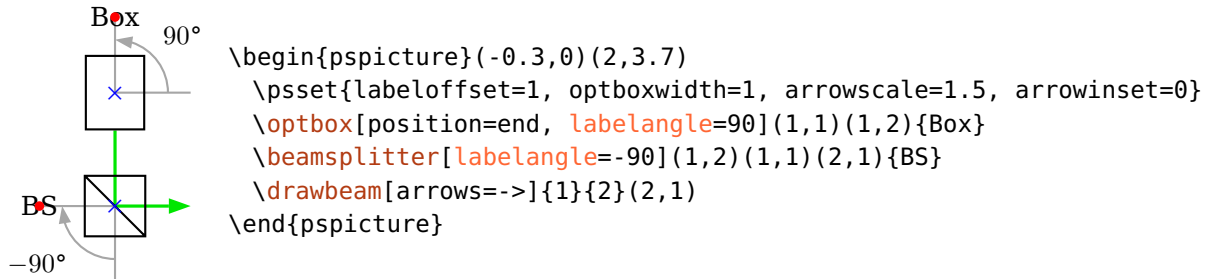
Defines the alignment of the label relative to the label node.



`labelangle=<num>`

default: 0

The rotation angle of the label node around the component center. The origin depends both on the component and the chosen reference system (see `labelref`).

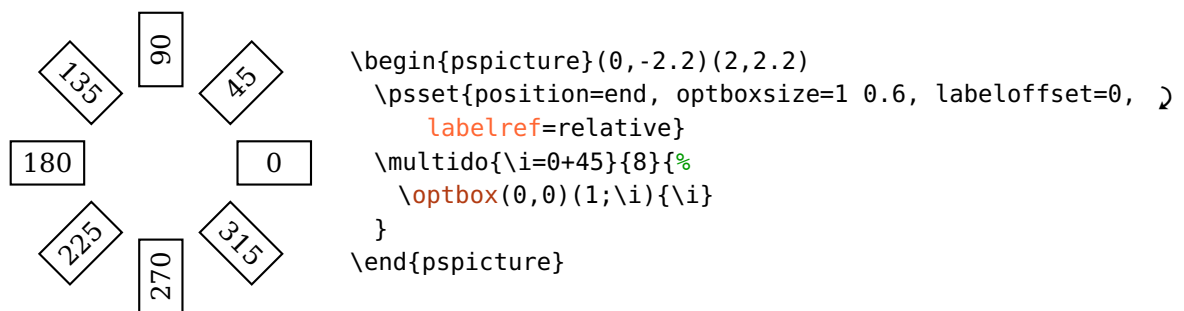


`labelref=relative, relgrav, global, absolute`

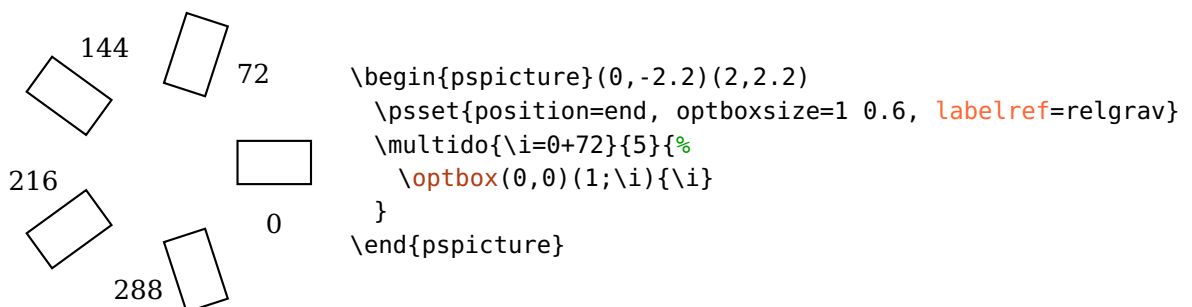
default: relgrav

Set the reference coordinate system for the `labelangle` and the orientation of the label text.

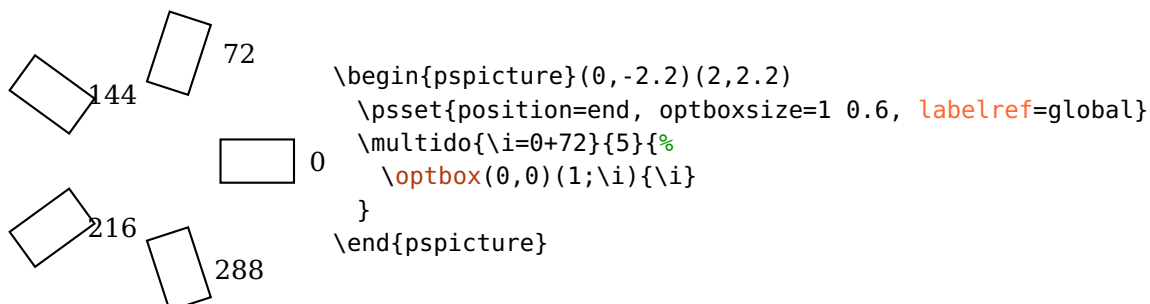
relative The label is parallel to the component reference line and the actual angle is always projected to the range between -90° and $+90^\circ$.



relgrav The label itself is always horizontal, only the label node is rotated together with the component.

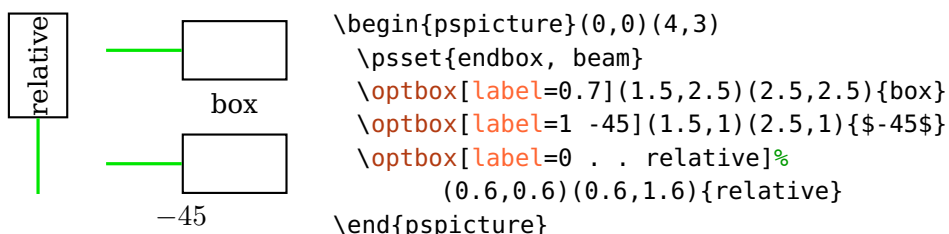


global, absolute The label is independent of the component alignment.



label= $\langle\text{offset}\rangle[\langle\text{angle}\rangle[\langle\text{refpoint}\rangle[\langle\text{labelref}\rangle]]]$

Allows compact definition of several label parameters. It takes up to four space-separated arguments (**labeloffset**, **labelangle**, **labelalign**, and **labelref**). Unchanged intermediate arguments can be skipped with a dot.



innerlabel=true

This is an alias for **label**=0 . c relative

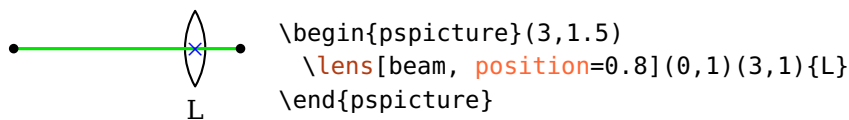
3.2. Positioning

All components except for the free-ray tripoles can be positioned between their two reference nodes **\oenableRefA** and **\oenableRefB**.

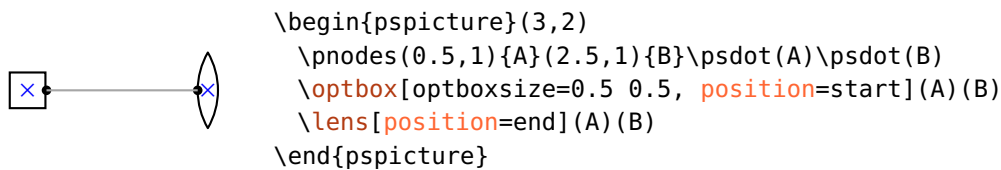
position= $\langle\text{num}\rangle$, start, end

Controls the relative position of an object center (blue cross) between its two reference points (black dots), if the value is a number. In this case it is equivalent to the **npos** parameter of **\ncput**, i.e. a number in the range $[0, 1]$ (see **pst-node** documentation¹).

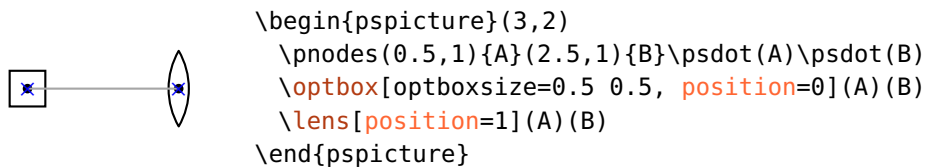
¹<http://mirror.ctan.org/help/Catalogue/entries/pst-node.html>



The values `start` and `end` place an object at the start or end of the reference line, respectively. The respective object interface is placed on the respective reference node (black dot) instead of the object center (blue cross) which would be the case for `position=0` or `position=1`. Achieving this with numerical values would be more difficult and inflexible, because the object size and shape must be taken into account.

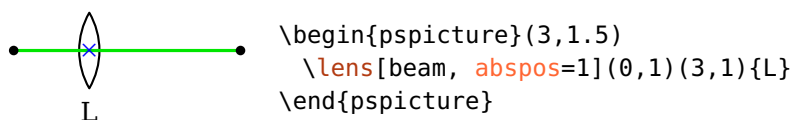


Here is for comparison the positioning with `position=0/1`.



`abspos=<num>, start, end`

Controls the absolute position of an object on the reference line. The values `start` and `end` are handled identically to `position`.



`endbox=true, false`

This parameter should not be used any more, it is equivalent to `position=end`.

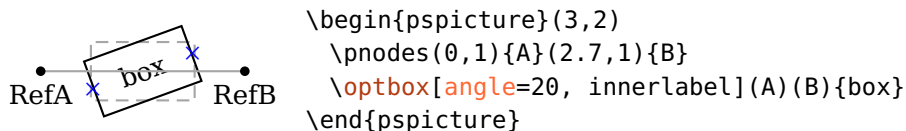
3.3. Rotating and shifting

All components can be shifted and rotated around different reference nodes.

Please note, that together with the components also their interface nodes (Sec. 8.6, marked as blue crosses in the following examples) are transformed, which may impact the connections. The transformed reference nodes are accessible as new nodes (Sec. 8.2).

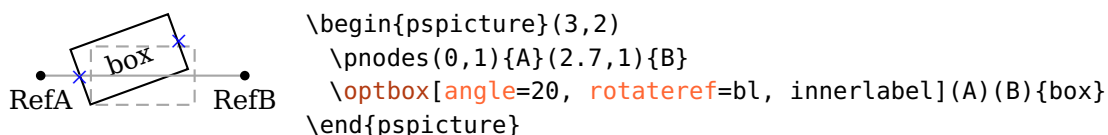
angle= $\langle psnum \rangle$ default: 0

Rotate a component by an angle $\langle num \rangle$ (in degree). The original position is drawn gray dashed.



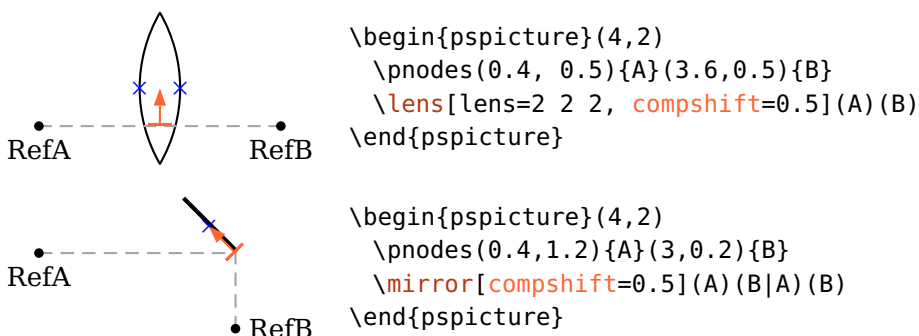
rotateref= $\langle refpoint \rangle$ default: c

Set the reference point for the rotation of the component. Please see Sec. 8.7 for a detailed explanation and Sec. 12.2.1 for a list of the possible reference nodes of all components. The original position is drawn gray dashed.



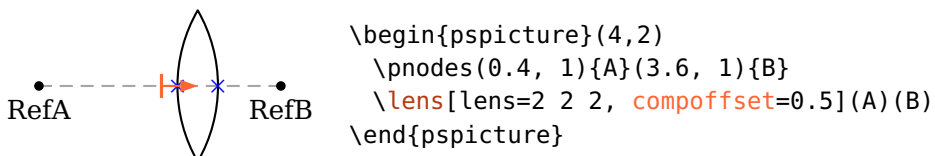
compshift= $\langle num \rangle$ default: 0

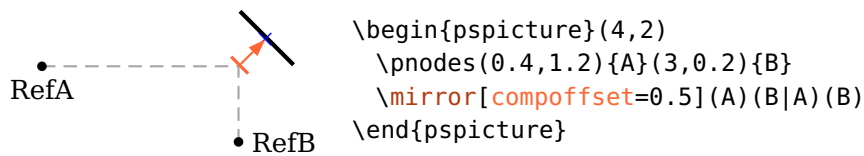
Shift a component perpendicular to its reference line (see Sec. 8.2). For reflective components it is shifted along the reflective interface.



compoffset= $\langle num \rangle$ default: 0

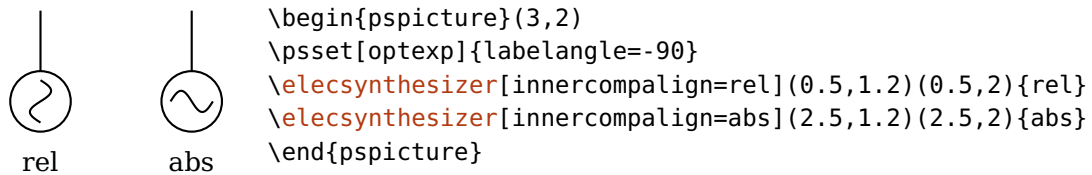
Shift a component perpendicular to its reference line (see Sec. 8.2). For reflective components it is shifted along the reflective interface. The displacement is vertical to a displacement with **compshift**. For an example see Ex. 11.25.





`innercompalign`=rel, relative, abs, absolute

If this parameter is set to `absolute`, the “inner” component part is not rotated with the component. At the moment, this option is used by `\optfilter`, `\elecsynthesizer`, and `\optdipole`.

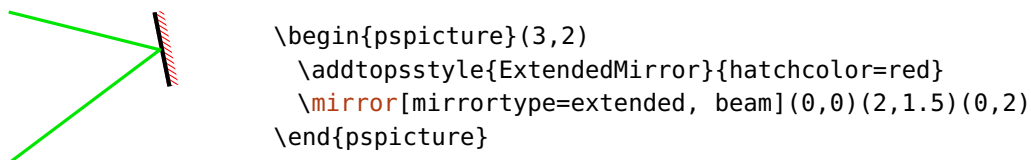


3.4. Using pststyles

Pststyles are custom graphics parameter configurations for PSTricks objects. The `pst-optexp` package makes extensive use of styles to facilitate flexible design of single, logical parts of components. This is in general not possible with the optional argument.

Existing styles can be extended with `\addtopsstyle` and overwritten with `\newpsstyle`.

As example we consider an extended mirror and want to change only the “extended” part which is controllable with the `ExtendedMirror` style:

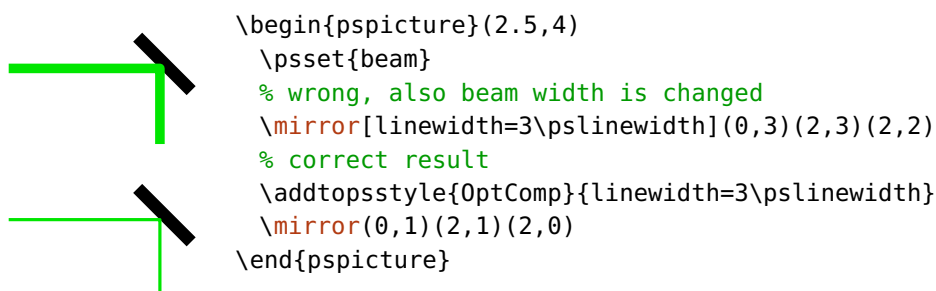


The changeable parts of each component are listed in the respective reference.

3.5. Component appearance

OptComp $\langle psstyle \rangle$

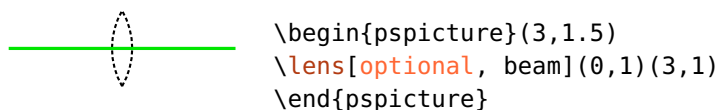
Affects the basic appearance of all optical components. Using standard graphics parameters like e.g. `linestyle` would change also the connections that are drawn together with the component (`beam`, `fiber` etc.). Use the key `newOptComp` and `addtoOptComp` to change the style for single components or for use with `\newpsobject` via the optional argument.



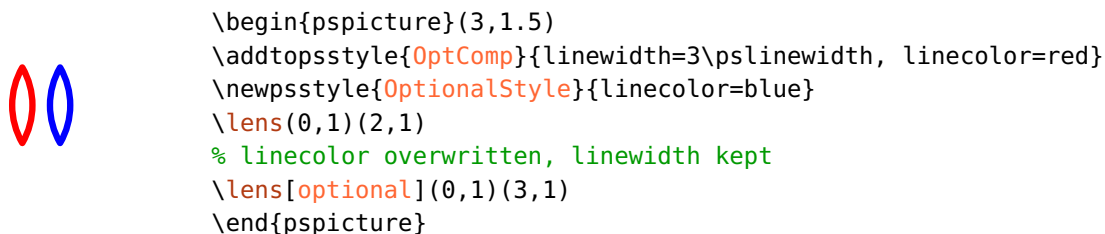
OptionalStyle $\langle psstyle \rangle$

default: `linestyle=dashed,dash=1.5pt 1pt`

This style is applied to components marked as `optional` in addition to `OptComp`.



`OptionalStyle` is applied after `OptComp` so that it can overwrite the general settings.



VariableStyle $\langle psstyle \rangle$

default: `linewidth=0.8\pslinewidth, arrowinset=0, arrowscale=0.8, arrows=<->`

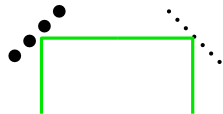
The style of the arrows of variable components (`\mirror` and `\optgrating`).

`addtoOptComp=<list>`

The `OptComp` style is extended by locally adding the parameters contained in `<list>`. The `<list>` must be surrounded by curly braces.

`newOptComp=<list>`

Similar to `addtoOptComp`, but an existing `OptComp` style is overwritten with the new parameter set.



```

\begin{pspicture}(3,1.5)
\psset{beam}
\newpsstyle{OptComp}{linewidth=3\pslinewidth}
\mirror[addtoOptComp={linestyle=dotted}](0.5,0)(0.5,1)(1.5,1)
% overwrites the linewidth settings
\mirror[newOptComp={linestyle=dotted}](1.5,1)(2.5,1)(2.5,0)
\end{pspicture}

```

`optional=true, false`

default: false

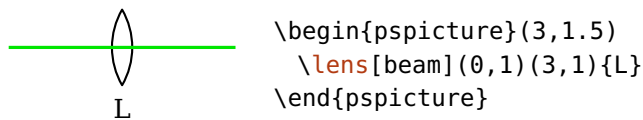
Mark an object as optional by applying the `OptionalStyle` style.

4. Free-ray components

This chapter describes all free-ray components and their options. Free-ray components differ from fiber components in that they provide transmittive or reflective interfaces which can be used to raytrace beams. They are divided in two types: some components like lenses need only two nodes for positioning (4.1–4.13), whereas e.g. mirrors require three nodes to determine the orientation (4.14–4.23).

4.1. Lens

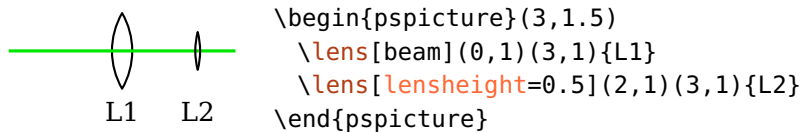
`\lens[⟨options⟩](⟨in⟩)(⟨out⟩){⟨label⟩}`



A lens is defined by its height and the radii of the input and output interfaces.

`lensheight=⟨num⟩` default: 1

Set the height of the lens.

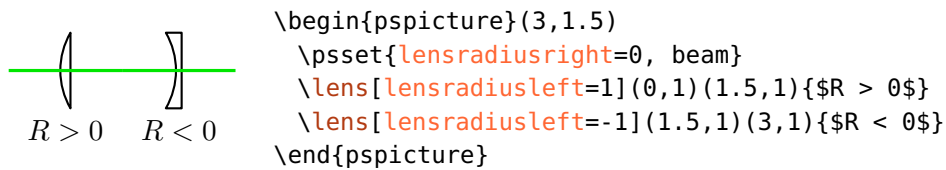


`lensradiusleft=⟨num⟩` default: 1

Set the left radius of the lens. A positive $\langle num \rangle$ is for convex, a negative one for concave curvatures. Use zero for a plain surface.

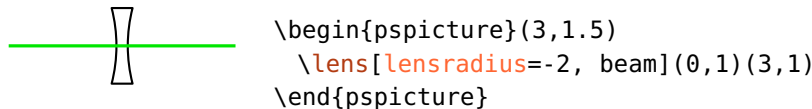
`lensradiusright=⟨num⟩` default: 1

Same as `lensradiusleft` but for the right surface.



lensradius= $\langle left \rangle$ [$\langle right \rangle$]

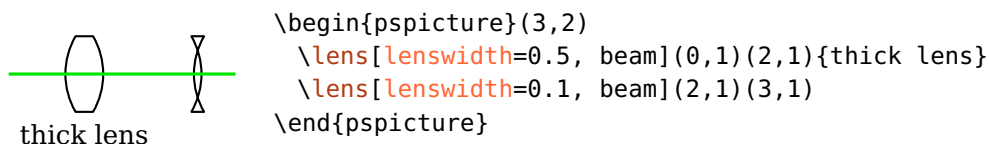
Set the left and right lens curvatures. If only a single value is given, it is used for both curvatures.



lenswidth= $\langle num \rangle$

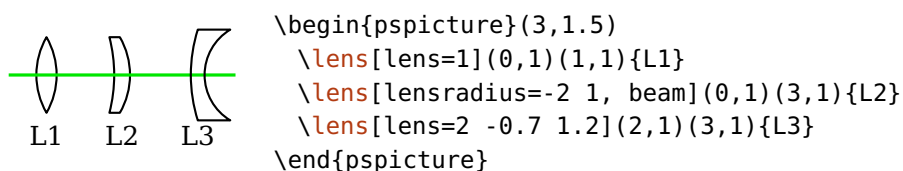
default: 0

Usually only the height and the two radii are used to construct the lens, the width is calculated automatically. For **lenswidth** greater than zero, this width is used instead. This is only useful if you want to draw thick lenses and it can give ugly results if the value is chosen too small.



lens= $\langle radiusleft \rangle$ [$\langle radiusright \rangle$][$\langle height \rangle$][$\langle width \rangle$]]

A convenience option to specify all lens parameters with a single option. Intermediate values can be skipped with a dot, values at the end can be omitted (e.g. `lens=. 1 1`).



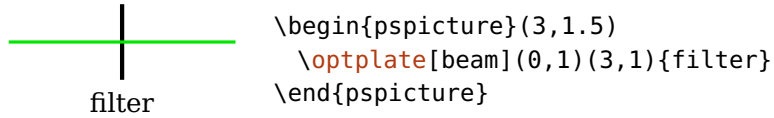
thicklens=true, false

default: false

This option was necessary until version 2.1 in order to draw a thick lens of width **lenswidth** instead of using the calculated lens width. This option is superfluous since version 3.0 because setting **lenswidth** to a positive value automatically draws a thick lens.

4.2. Optical plate

`\optplate`[*<options>*](*<in>*)(*<out>*){*<label>*}



`plateheight`=*<num>* default: 1

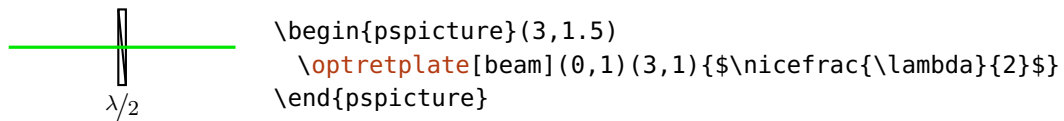
The height of the plate.

`platelinewidth`=*<num>* or *<dimen>* default: 2\pslinewidth

The linewidth of the plate. This could be defined also with the generic linewidth option. But this parameter allows setting globally the linewidth of all plates.

4.3. Retardation plate

`\optretplate`[*<options>*](*<in>*)(*<out>*){*<label>*}



`plateheight`=*<num>* default: 1

The height of the plate.

`platewidth`=*<num>* default: 0.1

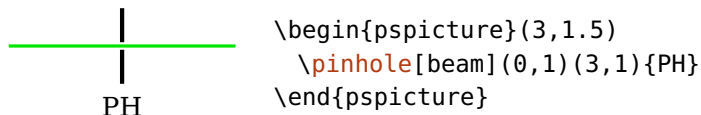
The width of the plate.

`platesize`=*<width>* *<height>*

The width and height of the plate, is equivalent to calling both `platewidth` and `plateheight`.

4.4. Pinhole

`\pinhole`[*<options>*](*<in>*)(*<out>*){*<label>*}



`outerheight`=*<num>* default: 1

The total height of the pinhole.

`innerheight`=*<num>* default: 0.1

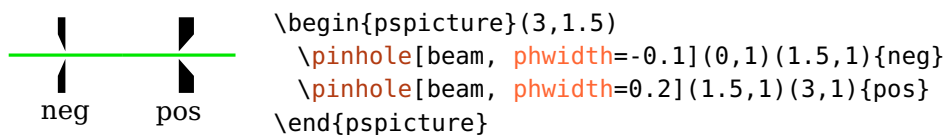
The height of the hole.

`phlinewidth`=*<num>* or *<dimen>* default: 2\pslinewidth

The linewidth of the pinhole. This could be defined also with the generic `linewidth` option. But this parameter allows setting globally the linewidth of all pinholes.

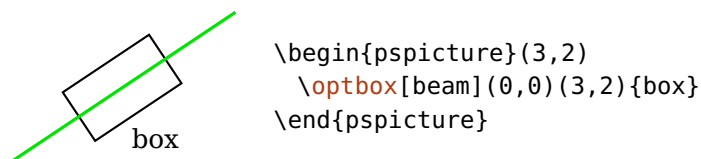
`phwidth`=*<num>* default: 0

The pinhole is drawn in a more plastic style if `phwidth` is not zero, in this case the `phlinewidth` is ignored. For negative values the shape is mirrored.



4.5. Box

`\optbox`[*<options>*](*<in>*)(*<out>*){*<label>*}



`optboxwidth`=*<num>* default: 1.4

The width of the box.

`optboxheight`= $\langle num \rangle$ default: 0.8

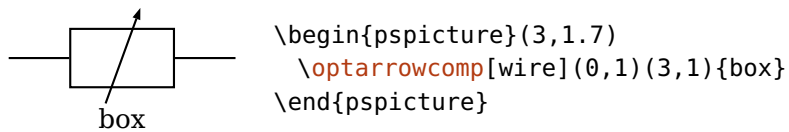
The height of the box.

`optboxsize`= $\langle width \rangle \langle height \rangle$

The width and height of the box, is equivalent to calling both `optboxwidth` and `optboxheight`.

4.6. Arrow component

`\optarrowcomp`[$\langle options \rangle$]($\langle in \rangle$)($\langle out \rangle$){ $\langle label \rangle$ }



A general purpose component, similar to `\fiberdelayline`, but much more flexible.

`arrowcompwidth`= $\langle num \rangle$ default: 1.4

The width of the box.

`arrowcompheight`= $\langle num \rangle$ default: 0.8

The height of the Komponente.

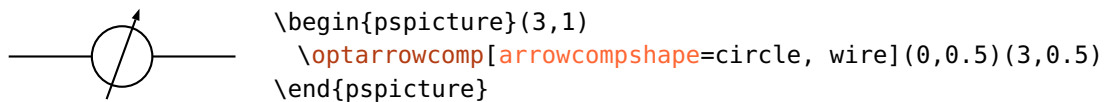
`arrowcompsize`= $\langle size \rangle$ or $\langle width \rangle \langle height \rangle$

The width and height of the component, is equivalent to calling both `arrowcompwidth` and `arrowcompheight`.

`arrowcompangle`= $\langle num \rangle$ default: 70

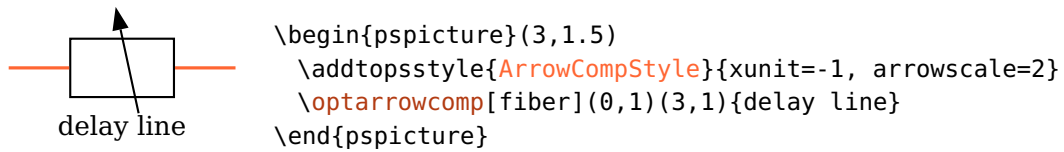
The angle of the arrow with respect to the x -axis.

`arrowcompshape`=rectangle, circle default: rectangle



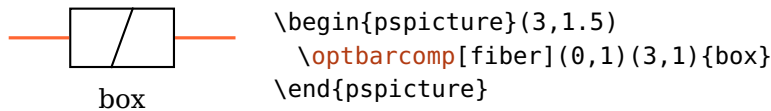
`ArrowCompStyle` $\langle psstyle \rangle$ default: `arrowinset=0, arrows=->`

The style of the arrow. This can be especially useful to adapt the length and orientation of the arrow.



4.7. Bar component

`\optbarcomp`[*<options>*](*<in>*)(*<out>*){*<label>*}



A general purpose component, similar to `\optfiberpolarizer`, but much more flexible.

`barcompwidth`=*<num>* default: 1.4

The width of the box.

`barcompheight`=*<num>* default: 0.8

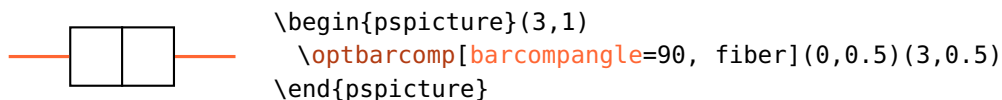
The height of the box.

`barcomp`*size*=*<size>* or *<width>* *<height>*

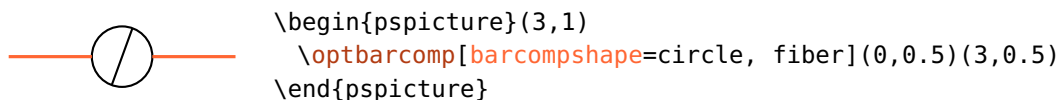
The width and height of the box, is equivalent to calling both `barcompwidth` and `barcompheight`.

`barcompangle`=*<num>* default: 70

The angle of the bar with respect to the *x*-axis.

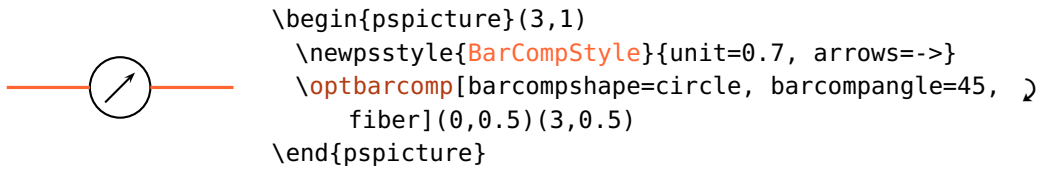


`barcompshape`=rectangle, circle default: rectangle



`BarCompStyle` *<psstyle>*

The style of the bar.



4.8. Optical source

`\optsource[⟨options⟩](⟨in⟩)(⟨out⟩){⟨label⟩}`



Laser

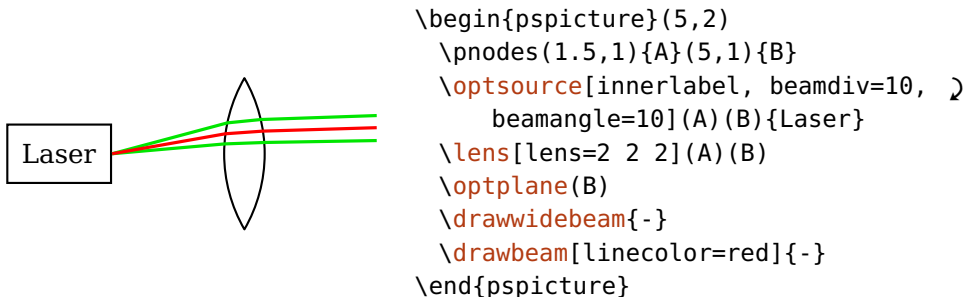
```

\begin{pspicture}(3,2)
\optsource(1.5,1)(3,1){Laser}
\end{pspicture}

```

An optical source predefines some properties of any outgoing beam, if it is the first component in a beam path. The definable beam properties are `beamangle`, `beamwidth`, `beamdiv` and `beamalign`. The properties defined by the optical source cannot be overwritten by the beam properties. If a property wasn't explicitly defined for a source, it is taken from the beam properties.

The default settings for the component position are `position=start`.



`sourcewidth=⟨num⟩` default: 1.4

The width of the box.

`sourceheight=⟨num⟩` default: 0.8

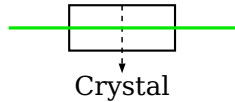
The height of the box.

`sourcesize=⟨width⟩⟨height⟩`

The width and height of the box, is equivalent to calling both `sourcewidth` and `sourceheight`.

4.9. Crystal

`\crystal[⟨options⟩](⟨in⟩)(⟨out⟩){⟨label⟩}`



```
\begin{pspicture}(3,1.3)
\crystal[beam](0,1)(3,1){Crystal}
\end{pspicture}
```

`crystalwidth=⟨num⟩`

default: 1.4

The width of the crystal.

`crystalheight=⟨num⟩`

default: 0.6

The height of the crystal.

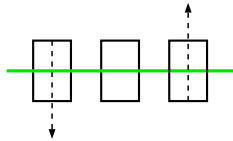
`crystalsize=⟨width⟩⟨height⟩`

The width and height of the crystal, is equivalent to calling both `crystalwidth` and `crystalheight`.

`caxislength=⟨num⟩`

default: 0.3

The additional length of the *c*-axis, is dropped when set to 0 and is mirrored if the values is negative. The total length is `caxislength` plus `crystalheight`.



```
\begin{pspicture}(0,0.4)(3,1.5)
\psset{crystalsize=0.5 0.8, label=-45 . l}
\crystal[position=0.2, caxislength=0.5](0,1)(3,1){pos}
\crystal[beam, caxislength=0](0,1)(3,1){0}
\crystal[position=0.8, caxislength=-0.5](0,1)(3,1){neg}
\end{pspicture}
```

`caxisinv=true, false`

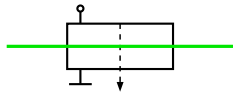
default: false

Invert the direction of the *c*-axis, this is equivalent to changing the sign of `caxislength`.

`voltage=true, false`

default: false

Draw a voltage connection and a ground symbol.

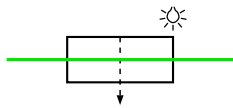


```
\begin{pspicture}(0,0.4)(3,1.5)
\crystal[voltage, beam](0,1)(3,1)
\end{pspicture}
```

`lamp=true, false`

default: false

Draw a lamp near the crystal.



```
\begin{pspicture}(0,0.4)(3,1.7)
\crystal[lamp, beam](0,1)(3,1)
\end{pspicture}
```

`lampscale=<num>`

Scaling of the lamp. This parameter is deprecated since version 3.0, use style `Crystallamp` instead.

`CrystalCaxis` `<psstyle>` default: linewidth=0.7\pslinewidth, linestyle=dashed, dash=2pt 2pt, arrowinset=0, arrows=->

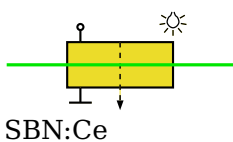
The style of the *c*-axis including the arrow type.



```
\begin{pspicture}(0,0.5)(3,1.6)
\newpsstyle{CrystalCaxis}{linecolor=red, arrows=->}
\crystal[beam](0,1)(3,1)
\end{pspicture}
```

`Crystallamp` `<psstyle>` default: linewidth=0.6\pslinewidth

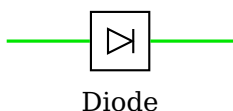
The style of the crystal background lamp.



```
\begin{pspicture}(3,1.7)
\crystal[fillstyle=solid, fillcolor=yellow!90!black,
label=1.2 -45, voltage,
lamp, beam](0,1)(3,1){SBN:Ce}
\end{pspicture}
```

4.10. Optical diode

`\optdiode` [`<options>`] (`<in>`) (`<out>`) {`<label>`}



```
\begin{pspicture}(3,2)
\optdiode[beam](0,1)(3,1){Diode}
\end{pspicture}
```

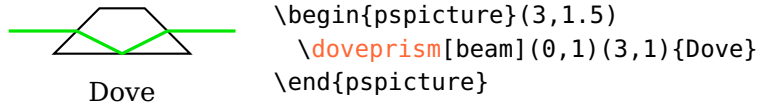
The optical diode has a default setting of `allowbeaminside=false`.

`optdiodesize=<num>` default: 0.8

The side length of the optical diode.

4.11. Dove prism

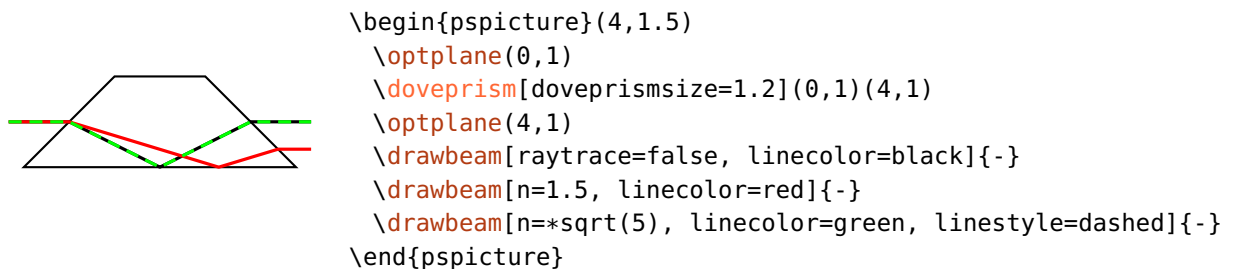
`\doveprism`[*options*](*in*)(*out*){*label*}



`doveprismsize`=*num* or *width* *height*

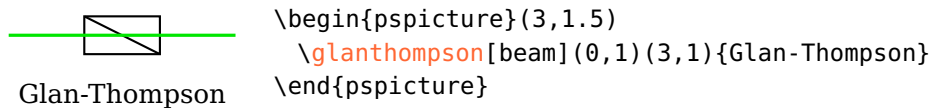
default: 0.6

A single number defines the height of the prism, the total width is set to three times the height. Two number define the width and height of the dove prism. The angles at the input and output faces are always 45°.



4.12. Glan-Thompson prism

`\glanthompson`[*options*](*in*)(*out*){*label*}



`glanthompsonwidth`=*num*

default: 1.0

The width of the prism.

`glanthompsonheight`=*num*

default: 0.5

The height of the prism.

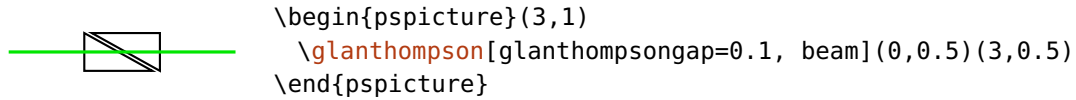
`glanthompsonsiz`=*width* *height*

The width and height of the prism, is equivalent to calling both `glanthompsonwidth` and `glanthompsonheight`.

`glanthompsongap=<num>`

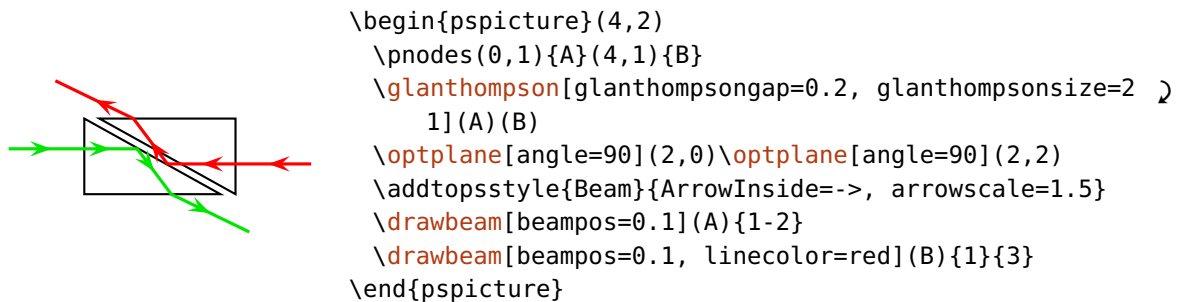
default: 0

The separation of the two prisms in x -direction. The total width does not depend on this value, but the angle of the reflective plane is affected.



The implementation of the Glan–Thompson prism for `glanthompsongap=0` is equivalent to a beamsplitter which is positioned with two nodes only and which is not quadratic.

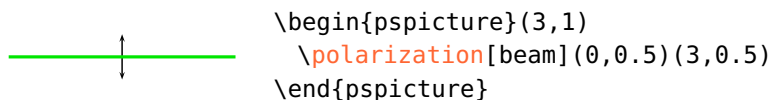
Please note, that the Prism cannot be used properly from both sides for `glanthompsongap>0` because the current implementation allows only a single ambiguous interface (reflective or transmittive). In the following example you see, that the red beam is reflected at the wrong interface.



Ex. 11.21 shows a further setup with a Glan–Thompson prism.

4.13. Polarization

`\polarization[<options>](<in>)(<out>){<label>}`



`polsize=<num>`

default: 0.6




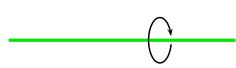

The size of the polarization symbol, the circle for perp and polymisc is half of this.

`pollinewidth`= $\langle num \rangle$ or $\langle dimen \rangle$ default: `0.7\pslinewidth`

The linewidth of the polarisation sign. This could be defined also with the generic `linewidth` option. But this parameter allows setting globally the linewidth of all polarisation signs. This parameter is deprecated since version 3.0, use style `Polarization` instead.

`poltype`=parallel, perp, misc, lcirc, rcirc default: parallel

This parameter chooses the polarization type.

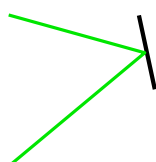
	<code>\begin{pspicture}(0,-0.3)(3,4.3)</code>
	<code>\psset[optexp]{beam}</code>
	<code>\begin{optexp}</code>
	<code>\polarization[poltype=parallel, abspos=0.5](0,4)(3,4)</code>
	<code>\polarization[poltype=perp, abspos=1](0,3)(3,3)</code>
	<code>\polarization[poltype=misc, abspos=1.5](0,2)(3,2)</code>
	<code>\polarization[poltype=lcirc, abspos=2](0,1)(3,1)</code>
	<code>\polarization[poltype=rcirc, abspos=2.5](0,0)(3,0)</code>
	<code>\end{optexp}</code>
	<code>\end{pspicture}</code>

`Polarization` $\langle psstyle \rangle$ default: `arrowscale=0.8, linewidth=0.7\pslinewidth, dotsize=3\pslinewidth`

Affects the style of the polarisation symbols.

4.14. Mirror

`\mirror`[$\langle options \rangle$]($\langle in \rangle$)($\langle center \rangle$)($\langle out \rangle$){ $\langle label \rangle$ }

	<code>\begin{pspicture}(3,2)</code>
	<code>\mirror[beam](0,0)(1.8,1.5)(0,2){M}</code>
	<code>\end{pspicture}</code>

`mirrorwidth`= $\langle num \rangle$ default: 1

The width of the mirror.

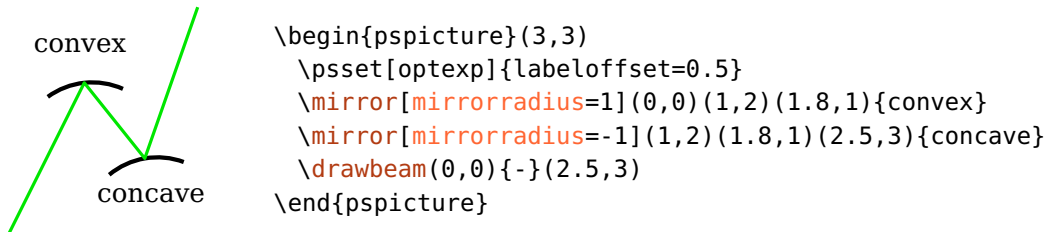
`mirrorlinewidth`= $\langle num \rangle$ or $\langle dimen \rangle$ default: `2\pslinewidth`

The linewidth of the mirror. This could be defined also with the generic `linewidth` option. But this parameter allows setting globally the linewidth of all mirrors.

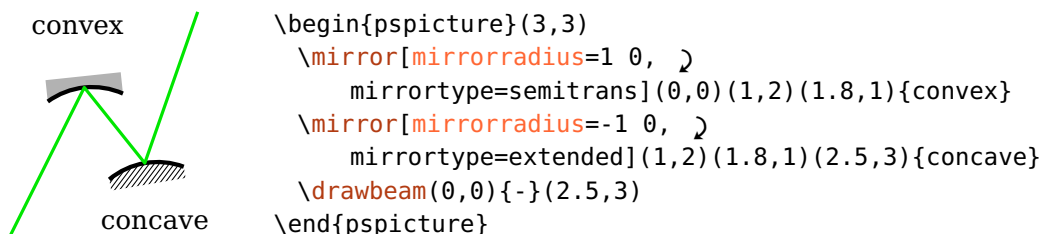
`mirrorradius`= $\langle radius \rangle$ [0]

default: 0

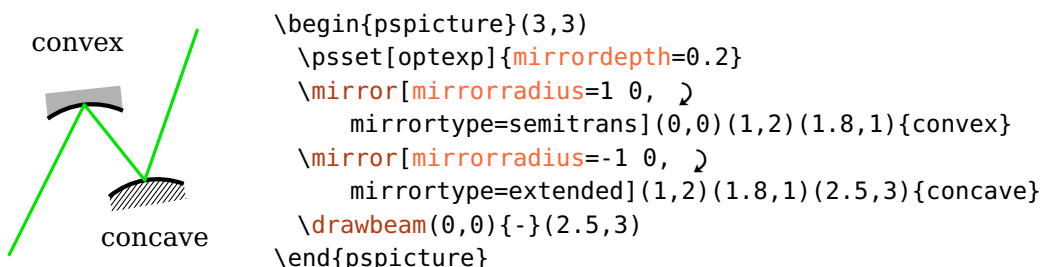
This parameter defines the curvature of the mirror. A value of 0 is for a plain mirror, a negative radius for a convex mirror and a positive radius gives you a concave mirror.



If no second value is specified, the curvature of the second interface of an extended or semitrans mirror is the same as that of the main interface (see previous example). If zero is given as second value (other values are not supported), the second interface is plain.



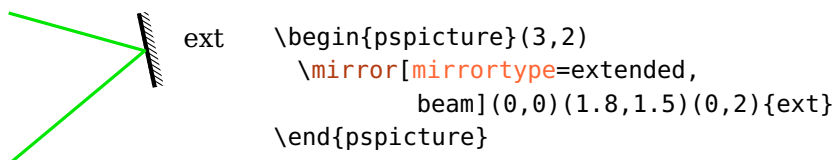
The value of `mirrordepth` always refers to the minimum depth of the mirror.



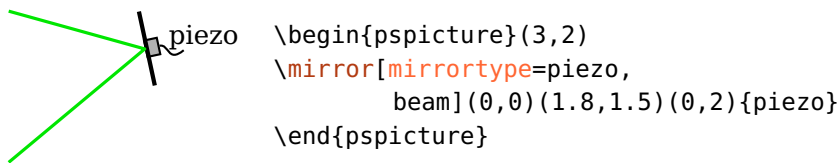
`mirrortype`=plain, piezo, extended, semitrans

default: plain

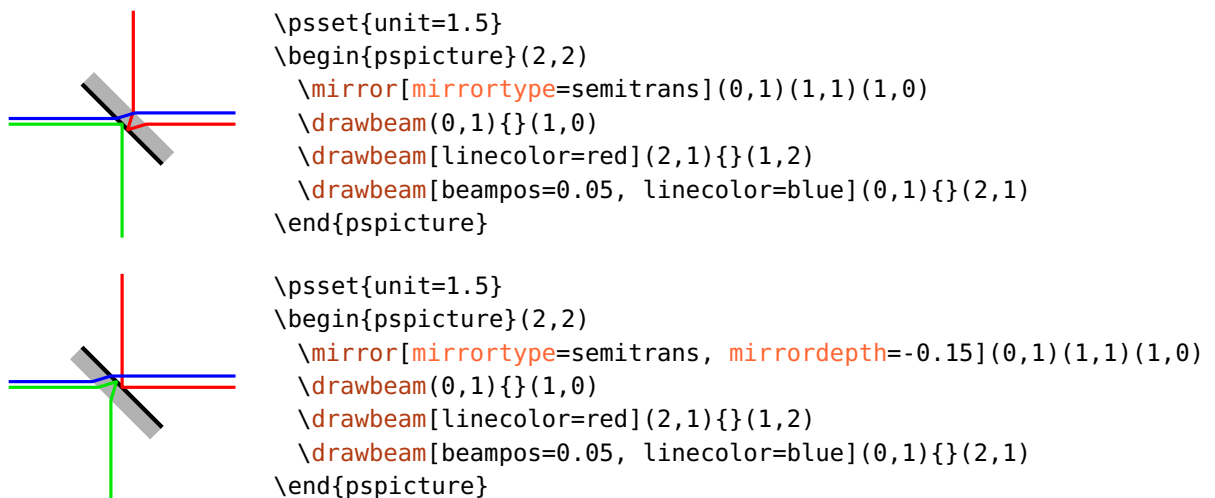
The `mirrortype` selects between different types of mirrors. The style is controlled with `PiezoMirror`, `ExtendedMirror`, and `SemitransMirror`.



Note, that the default piece of wire is omitted when using `extnode` with a piezo mirror.



A semi-transparent mirror (`semitrans`) has two interfaces, in contrast to the other mirror types. Their distance and position depends on the value and sign of `mirrordepth`.

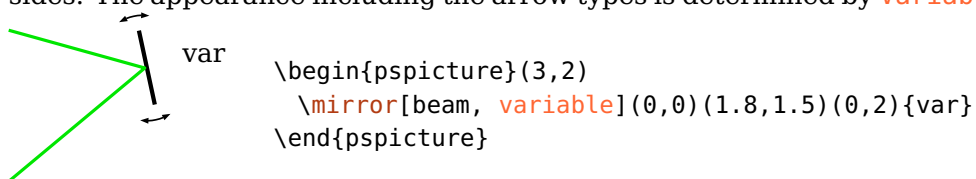


A negative `mirrordepth` is equivalent to a positive depth combined with `angle=180`.

`variable=true, false`

default: false

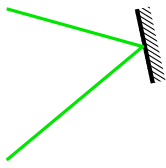
Draw an adjustable mirror which has two additional curved arrows on both sides. The appearance including the arrow types is determined by `VariableStyle`.



`mirrordepth=<num>`

default: 0.1

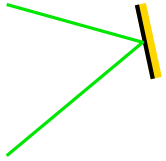
The total depth of an extended mirror.



```
\begin{pspicture}(3,2)
  \mirror[mirrortype=extended, mirrordepth=0.2,
    beam](0,0)(1.8,1.5)(0,2)
\end{pspicture}
```

ExtendedMirror *<psstyle>* default: linestyle=none, hatchwidth=0.5\pslinewidth, hatchsep=1.4\pslinewidth, fillstyle=hlines

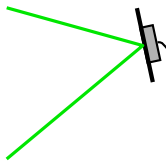
The style for the extended mirror.



```
\begin{pspicture}(3,2)
  \newpsstyle{ExtendedMirror}{fillstyle=solid,
    fillcolor=Gold,
    linestyle=none}
  \mirror[mirrortype=extended,
    beam](0,0)(1.8,1.5)(0,2){ext}
\end{pspicture}
```

PiezoMirror *<psstyle>* default: fillstyle=solid, fillcolor=black!30

This style defines the appearance of the piezo mirror. This can also be used to change the size of piezo part, as the example shows.



```
\begin{pspicture}(3,2)
  \addtopsstyle{PiezoMirror}{xunit=2}
  \mirror[mirrortype=piezo,
    beam](0,0)(1.8,1.5)(0,2){piezo}
\end{pspicture}
```

SemitransMirror *<psstyle>* default: linestyle=none, fillstyle=solid, fillcolor=black!30

This style defines the appearance of the semitrans mirror.

4.15. Parabolic mirror

\parabolicmirror [*<options>*] (*<in>*) (*<out>*) {*<label>*}



PM

```
\begin{pspicture}(3,2)
  \parabolicmirror(0,1)(2,1){PM}
\end{pspicture}
```

parmirrorwidth=*<num>*

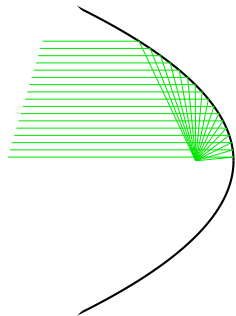
default: 1

The width of the parabolic mirror.

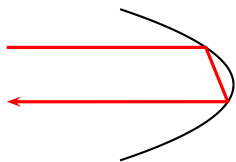
`parmirrorheight`= $\langle num \rangle$

default: 1

The height of the parabolic mirror.



```
\begin{pspicture}(0,-2)(3,2)
\parabolicmirror[parmirrorheight=4, parmirrorwidth=2](0,0)(3,0)
\addtopsstyle{Beam}{beamalign=abs, linewidth=0.2\pslinewidth, 2
beaminsidelast, stopinsidecount=1}
\multido{\r=-1.55+0.1}{33}{%
\drawbeam[beampos=\r](0,0){1}}
\end{pspicture}
```

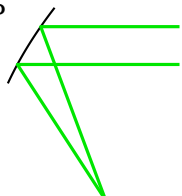


```
\begin{pspicture}(0,-1)(3,1)
\optplane(0,0)
\parabolicmirror[parmirrorheight=2, 2
parmirrorwidth=1.5](0,0)(3,0)
\addtopsstyle{Beam}{arrows=->, beamalign=abs}
\drawbeam[beampos=0.5, linecolor=red]{1}{2}{1}
\end{pspicture}
```

4.16. Off-axis parabolic mirror

`\oapmirror`[$\langle options \rangle$]($\langle in \rangle$)($\langle center \rangle$)($\langle focus \rangle$){ $\langle label \rangle$ }

OAP



```
\begin{pspicture}(3,2.5)
\oapmirror(3,2)(1,2)(2,0){OAP}
\drawwidebeam[beamwidth=0.5](3,2){1}(2,0)
\end{pspicture}
```

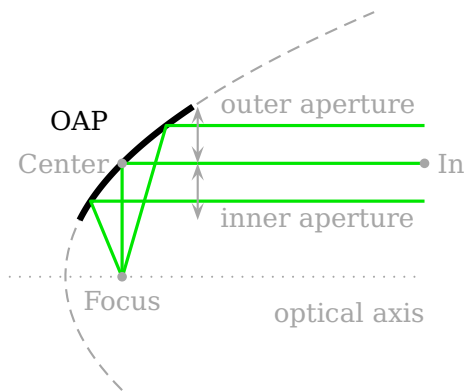
`oapmirroraperture`= $\langle num \rangle$ or $\langle inner \rangle$ $\langle outer \rangle$

default: 1

Mirror aperture with respect to the input direction. If a single number is given, it specifies the whole aperture, measured perpendicularly and symmetric to the connection from $\langle in \rangle$ to $\langle center \rangle$. Two numbers give the inner and the outer apertures.

The mirror construction using the given nodes is done as follows: the line from node `<in>` to `<center>` is parallel to the optical axis, `<focus>` specifies the focal point of the parabola. The mirror aperture is given by the parameter `oapmirroraperture`, measured perpendicular to the incoming beam.

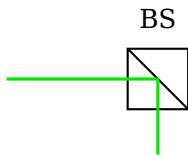
The different parameters and the whole original parabola are shown in the following illustration:



```
\begin{pspicture}(6,5)
\pnodes(5.5,3){In}(1.5,3){Center}(1.5,1.5){Focus}
\oapmirror[oapmirroraperture=1.5, 2
linewidth=3\pslinewidth, 2
plotpoints=200](In)(Center)(Focus){OAP}
\multido{\r=-0.5+0.5}{3}{%
\drawbeam[beampos=\r](In){}(Focus)}
\end{pspicture}
```

4.17. Beamsplitter

`\beamsplitter`[`<options>`](`<in>`)(`<center>`)(`<out>`){`<label>`}



```
\begin{pspicture}(3,2)
\beamsplitter[beam](0,1)(2,1)(2,0){BS}
\end{pspicture}
```

`bssize`=`<num>`

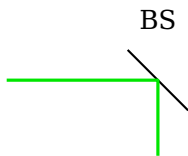
default: 0.8

The beamsplitter size.

`bsstyle`=cube, plate

default: cube

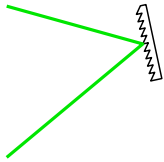
Selects between two types of beamsplitters: the beamsplitter cube (cube) and the semitransparent mirror (plate). This semitransparent mirror is equivalent to `\mirror` with `mirrorrttype`=semitrans and `mirrordepth`=0.



```
\begin{pspicture}(3,2)
\beamsplitter[bsstyle=plate, beam](0,1)(2,1)(2,0){BS}
\end{pspicture}
```

4.18. Optical grating

`\optgrating`[*<options>*](*<in>*)(*<center>*)(*<out>*){*<label>*}



```
\begin{pspicture}(3,2)
  \optgrating[beam](0,2)(1.8,1.5)(0,0){G}
\end{pspicture}
```

Before package version 3.0 this component was called `\optgrid`. Since then this command name is deprecated and will be removed in future versions.

`gratingwidth`=*<num>* default: 1

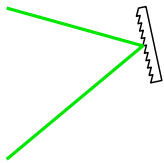
The width of the grating.

`gratingheight`=*<num>* default: 0.15

The total height of the grating.

`gratingdepth`=*<num>* default: 0.075

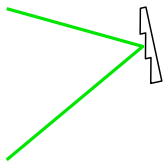
The modulation depth of the grating structure. The total height is adapted to this value if it is smaller than the modulation depth. Otherwise the total height is not affected.



```
\begin{pspicture}(3,2)
  \optgrating[gratingdepth=0.05,
    beam](0,2)(1.8,1.5)(0,0){G}
\end{pspicture}
```

`gratingcount`=*<int>* default: 10

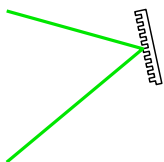
The number of grating grooves.



```
\begin{pspicture}(3,2)
  \optgrating[gratingcount=3,
    beam](0,2)(1.8,1.5)(0,0){G}
\end{pspicture}
```

`gratingtype`=blazed, binary default: blazed

Select between a binary and a blazed grating.

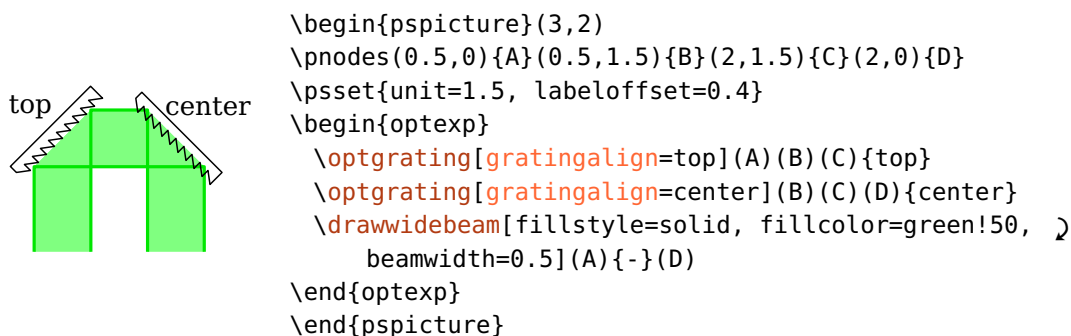


```
\begin{pspicture}(3,2)
  \optgrating[gratingtype=binary,
    beam](0,2)(1.8,1.5)(0,0){G}
\end{pspicture}
```

gratingalign=t, top, c, center

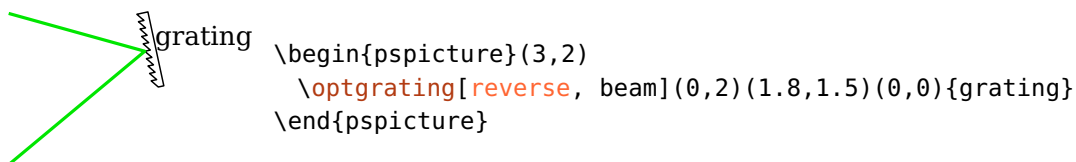
default: top

Selects if the reflection plane resides on top of the grating or in its center. If you choose center, you should use the **optexp** environment or a semitransparent beam filling.

**reverse**=true, false

default: false

Reverse the slope of the grooves of the blazed grating.

**gratinglinewidth**=⟨num⟩ or ⟨dimen⟩

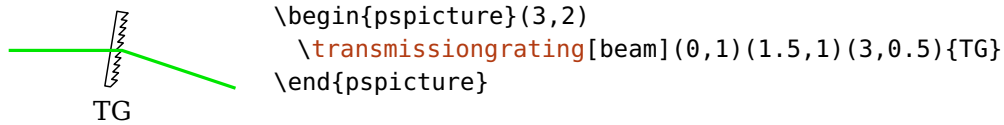
default: 0.7\pslinewidth

The linewidth of the grating. This could be defined also with the generic linewidth option. But this parameter allows setting globally the linewidth of all gratings.

optgridwidth These parameters were renamed to the respective grating* parameters and
optgridheight are deprecated since version 3.0.
optgriddepth
optgridcount
optgridtype
optgridlinewidth

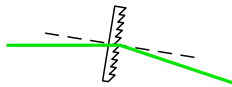
4.19. Transmission grating

\transmissiongrating[⟨options⟩](⟨in⟩)(⟨center⟩)(⟨out⟩){⟨label⟩}

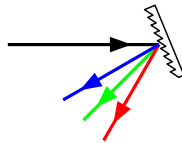


The transmission grating supports all options of `\optgrating`. The beam path is calculated such, that the beam is reflected at an interface which is perpendicular to the drawn grating. This virtual interface is shown in the following example:

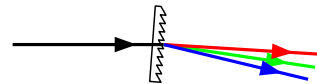
virtual reflection
interface



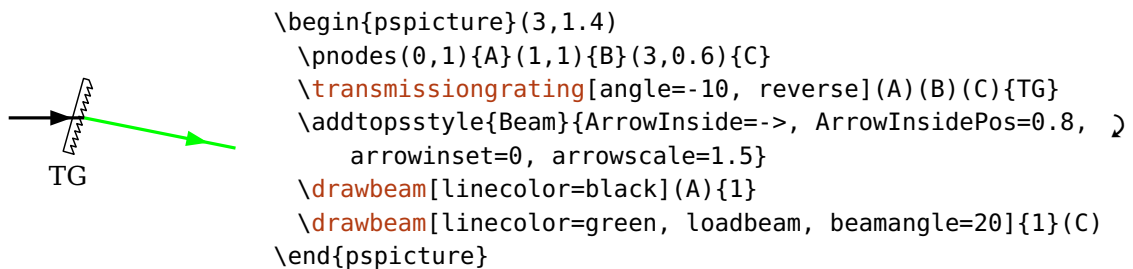
reflection grating
`\optgrating`



transmission grating
`\transmissiongrating`

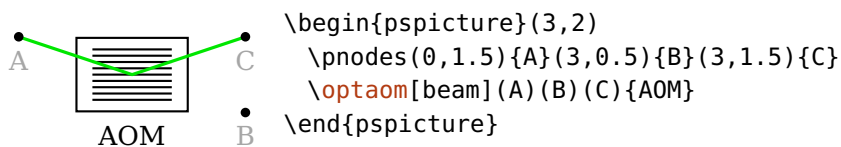


If this is not what you want, you may e.g. rotate the component further with `angle`, and then adjust also the `beamangle`.



4.20. Acousto-optic modulator

`\optaom`[`<options>`](`<in>`)(`<trans>`)(`<diff>`){`<label>`}



This macro draw an acousto-optic modulator, which either be transmissive or reflective, like a `\beamsplitter`. Please note, that the nodes do not correspond

to those of a usual tripole like an `\optgrating`. The second node is in the direction of the transmitted beam, the third node gives the direction of the first diffraction order.

This first implementation in version 4.8 is still experimental, because I didn't have enough test cases to check all possible cases. Therefore, future versions may have changes, which are not backwards compatible.

`aomheight`= $\langle num \rangle$ default: 1

The height of the modulator.

`aomwidth`= $\langle num \rangle$ default: 1.5

The width of the modulator.

`aomsize`= $\langle width \rangle \langle height \rangle$

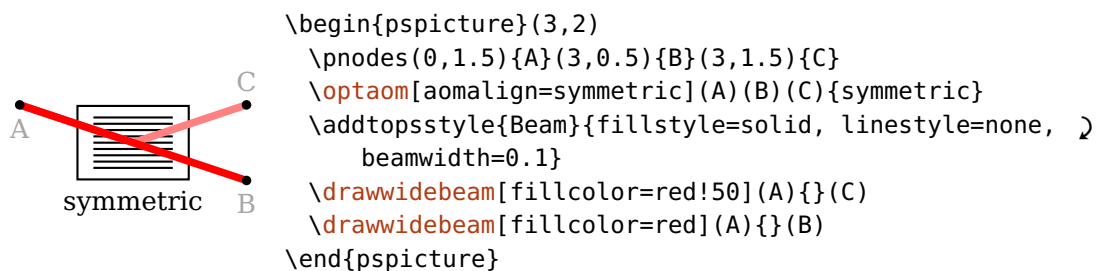
The width and height of the modulator, is equivalent to calling both `aomwidth` and `aomheight`.

`aomgratingcount`= $\langle int \rangle$ default: 9

Changes the number of grating lines. For other changes of the grating you must use the `aomcomp` parameter to redefine the whole grating.

`aomalign`=symmetric, straight default: symmetric

The alignment of an `\optaom` with respect to the three reference nodes can be set with this option. The symmetric alignment is shown in the following example:



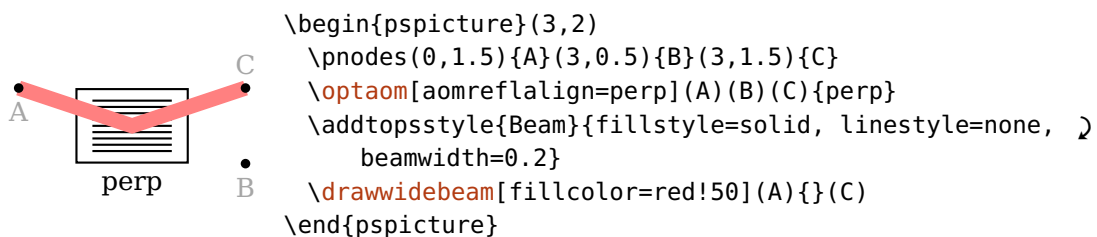
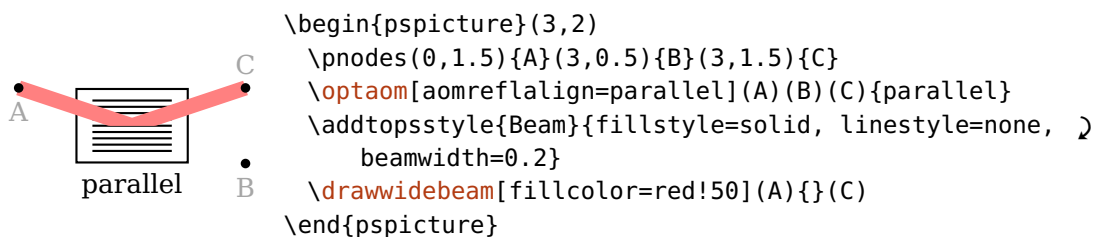
For straight alignment the sketch looks like



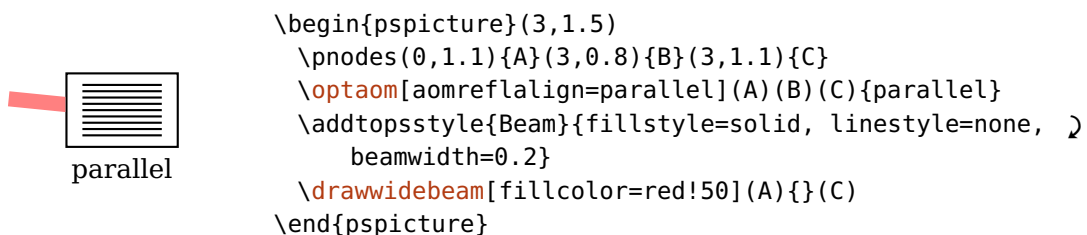
`aomreflalign=perp, parallel`

default: parallel

The reflective interface can be oriented either parallel or perpendicular to the component. This option was necessary in first place for technical reasons, the direction of the reflected ray is not affected by this option.



The following example show a use case, which doesn't work properly with `aomreflalign=parallel`, because the incident angle is so shallow, that the intersection points with the interface lay outside of the component.



`aomcomp=default, <macro>`

default: default

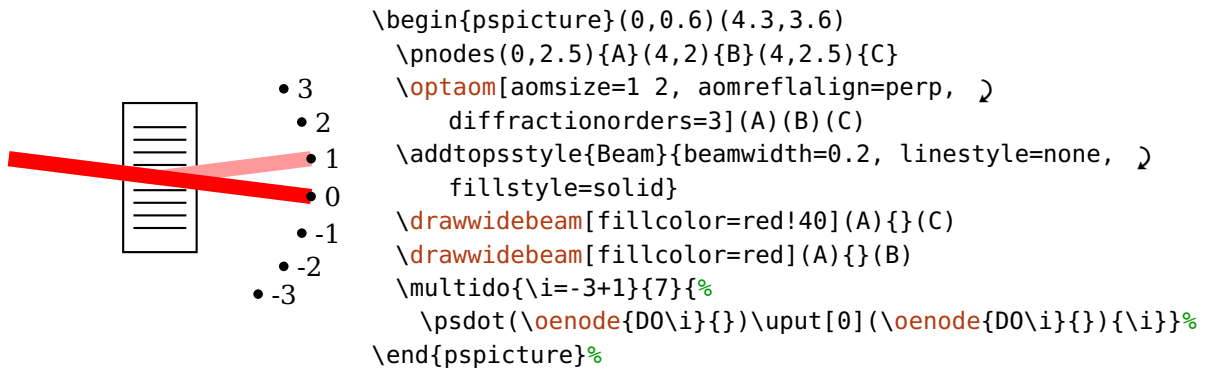
Changes the grating lines to an arbitrary drawing.

`diffractionorders=<int>`

default: 2

The number or diffraction orders (positive and negative) which can be selected with `beamdiffractionorder`, i.e. $-\langle num \rangle, -\langle num \rangle + 1, \dots, \langle num \rangle$.

At the same time the nodes `\oencode{D0-1}{<comp>}`, `\oencode{D00}{<comp>}`, `\oencode{D01}{<comp>}` etc. are defined.



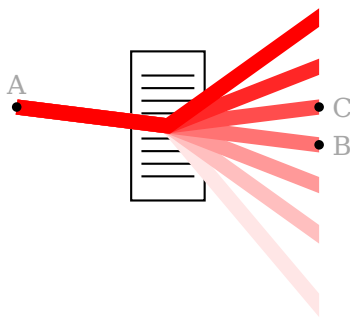
`beamdiffractionorder=<int>`

default: {}

The diffraction order, which is used for a reflected beam going through the aom. The option is ignored if its value is empty {}.

By using this option you automatically select the behaviour of the interface, which then doesn't depend on the relative position of AOM and the following component. For `beamdiffractionorder` $\neq 0$ a beam gets reflected, for the value 0 it gets transmitted. The parameter `beammode` is ignored.

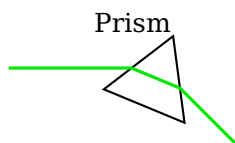
You can select only those orders, where have been defined for the respective component with `diffractionorders`. If an undefined order is selected, the closest one is chosen instead and a Postscript warning is issued (see `pswarning`).



```
\begin{pspicture}(0,-0.2)(4.4,3.8)
  \pnodes(0,2.5){A}(4,2){B}(4,2.5){C}
  \optaom[aomsize=1 2, aomreflalign=perp, diffractionorders=3](A)(B)(C)
  \optplane(B)
  \addtopsstyle{Beam}{beamwidth=0.2, linestyle=none, fillstyle=solid}
  \multido{\i=-3+1,\r=10+15}{7}{%
    \drawwidebeam[fillcolor=red!\r, beamdiffractionorder=\i](A){-}}
\end{pspicture}%
```

4.21. Prism

`\optprism`[*<options>*](*<in>*)(*<center>*)(*<out>*){*<label>*}



```
\begin{pspicture}(3,2)
  \optprism[beam](0,1)(2,1)(3,0){Prism}
\end{pspicture}
```

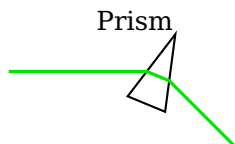
The prism is always placed symmetric between *<in>* and *<out>* nodes. For asymmetric beam traces see Sec. 9.

`prismsize`=*<num>* default: 1

The height of the prism.

`prismangle`=*<num>* default: 60

The upper angle of the prism.

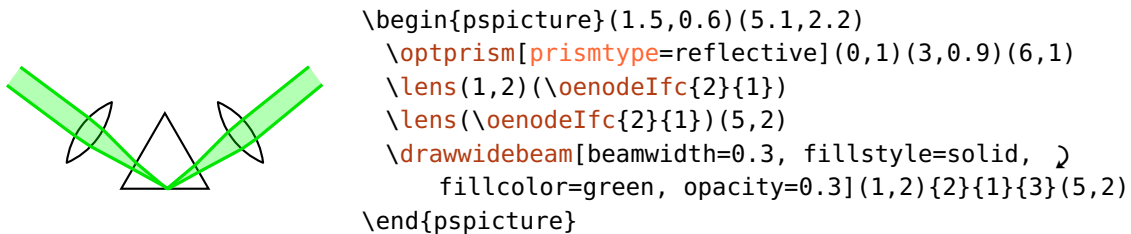


```
\begin{pspicture}(3,2)
  \optprism[prismangle=30, beam](0,1)(2,1)(3,0){Prism}
\end{pspicture}
```

prismtype=transmissive, reflective

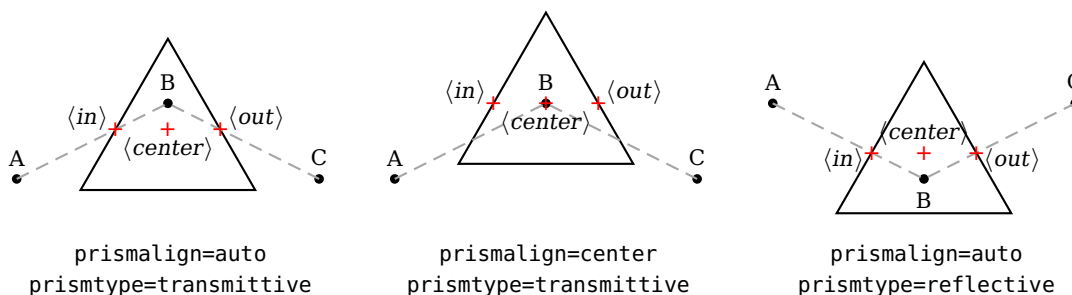
default: transmissive

If this parameter is set to reflective, the prism base is used as additional, reflective interface.

**prismalign**=auto, center

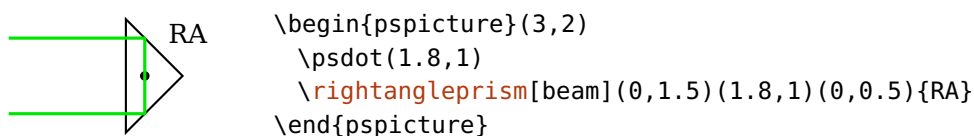
default: auto

Sets the vertical alignment of the prism with respect to the “reflection” node $\langle center \rangle$. For auto, the prism is shifted such, that the interface nodes lay on the connection between the respective reference node and the “reflection” node. If the value is center, the “reflection” node coincides with the component center. See the following examples for further explanation.

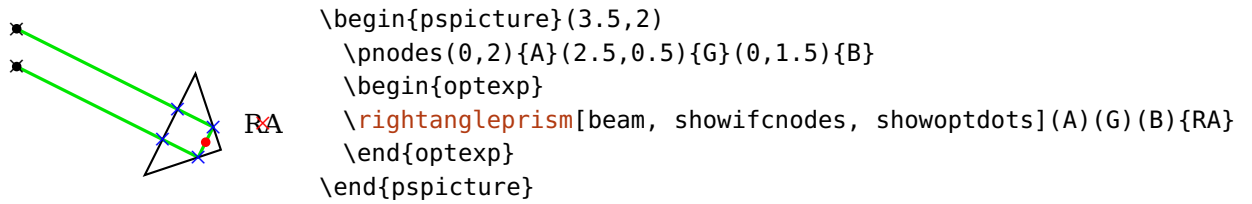


4.22. Right-angle prism

\rightangleprism[$\langle options \rangle$]($\langle in \rangle$)($\langle center \rangle$)($\langle out \rangle$){ $\langle label \rangle$ }



The right-angle prisms is align such that the incoming and reflected beam are parallel and the $\langle center \rangle$ node is vertically centered in the prism.



`raprismsize=<num>`

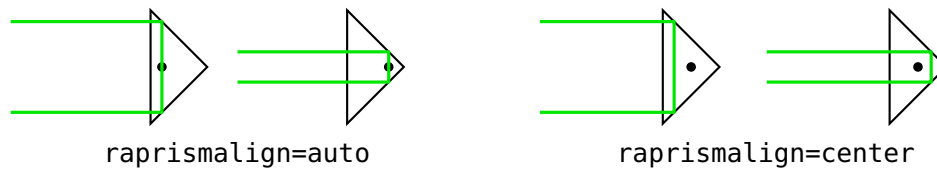
default: 1.5

The length of the input plane.

`raprismalign=auto, center`

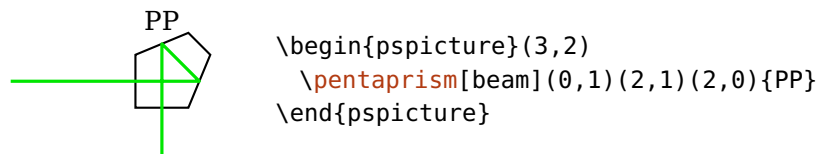
default: auto

Switch between automatic and fixed alignment of the component vertical to the entrance interface:



4.23. Penta prism

`\pentaprism[<options>](<in>)(<center>)(<out>){<label>}`



`pentaprismsize=<num>`

default: 0.7

The length of the input and output plane.

5. Fiber components

This chapter describes all fiber components and their options. They differ from free-ray components in that they are connected by default with fibers to their reference nodes, and that they cannot be used for raytracing. Instead of automatic fiber connections the components can also be connected manually with fibers, see Sec. 9.6.

Most components require only two reference nodes and are handled like the free-ray dipoles. Some special components like couplers (Sec. 5.10) and circulators (Sec. 5.9) are treated differently.

`usefiberstyle=true, false`

For some components (e.g. `\optfilter` or `\optmzm`) it can be nice to highlight some internals. For example, if this option is enabled the passing parts of the optical filter are drawn with the **Fiber** style. In the documentation this parameter is enabled to show the parts which are affected. Takes precedence over `usewirestyle`.


`usewirestyle=true, false`

Can be used to draw internal connections als **Wire**. Sets `usefiberstyle` to false, therefore it can be useful to define e.g. an electrical filter:

```
\newpsobject{elecfilter}{optfilter}{usewirestyle}
```

5.1. Optical fiber

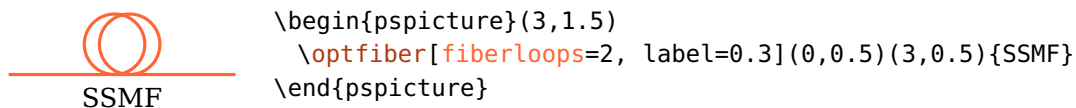
`\optfiber[⟨options⟩](⟨in⟩)(⟨out⟩){⟨label⟩}`

	<pre>\begin{pspicture}(3,1.5) \optfiber[label=0.3](0,0.5)(3,0.5){SSMF} \end{pspicture}</pre>
---	--

fiberloops= $\langle int \rangle$

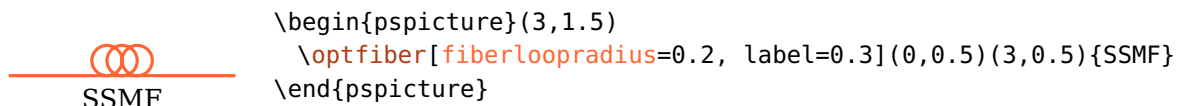
default: 3

Number of the fiber loops.

**fiberloopradius**= $\langle num \rangle$

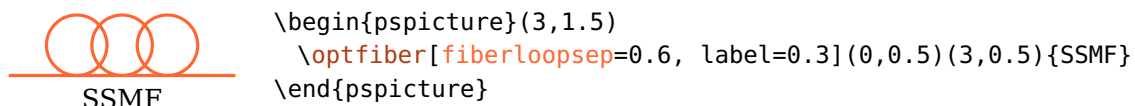
default: 0.4

Radius of the fiber loops.

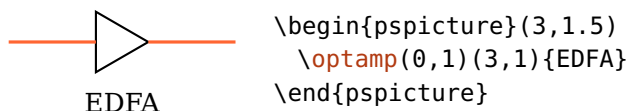
**fiberloopsep**= $\langle num \rangle$

default: 0.2

Separation between two successive fiber loops.



5.2. Optical amplifier

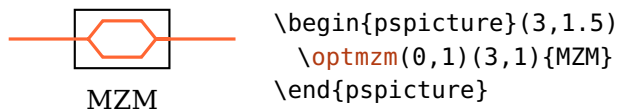
\optamp[$\langle options \rangle$]($\langle in \rangle$)($\langle out \rangle$){ $\langle label \rangle$ }**optampsize**= $\langle num \rangle$ or $\langle width \rangle \langle height \rangle$

default: 0.8

A single number gives the side length of the amplifier, two numbers the width and height. Note, that `optampsize=1` and `optampsize=1 1` do not give the same result. You may also change the relation of width and height with `xunit` and `yunit`.

5.3. Mach-Zehnder modulator

`\optmzm` [*options*] (*in*) (*out*) {*label*}

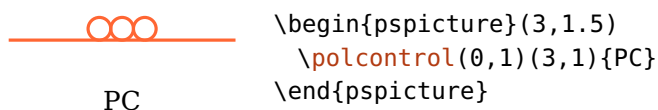


`optmzmsize`=*num* or *width* *height* default: 0.8

A single number gives the modulator height, the width is 1.6 times the height.
Two numbers define width and height directly.

5.4. Polarization controller

`\polcontrol` [*options*] (*in*) (*out*) {*label*}

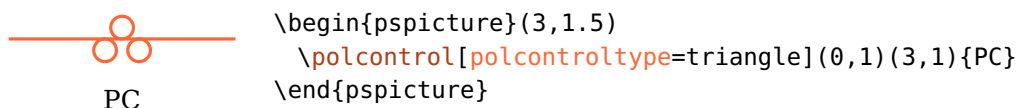


`polcontrolsize`=*num* default: 0.15

The radius of the polarization controller circles.

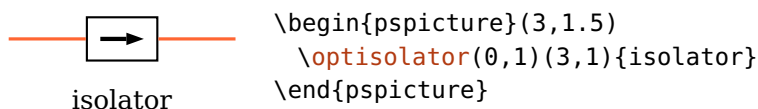
`polcontroltype`=linear, triangle default: linear

The type of polarization controller.



5.5. Isolator

`\optisolator` [*options*] (*in*) (*out*) {*label*}

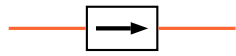


isolatorsizes= $\langle num \rangle$ or $\langle width \rangle \langle height \rangle$ default: 0.6

A single number gives the isolator height, the width is 1.6 times the height. Two numbers define width and height directly.

IsolatorArrow $\langle psstyle \rangle$ default: linewidth=2\pslinewidth, arrowinset=0

The style of the isolator arrow. This can be especially useful to adapt the length of the arrow.



```

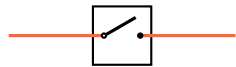
\begin{pspicture}(3,1.5)
\addtopsstyle{IsolatorArrow}{xunit=1.2}
\optisolator(0,1)(3,1){isolator}
\end{pspicture}

```

isolator

5.6. Optical switch

\optswitch[$\langle options \rangle$]($\langle in \rangle$)($\langle out \rangle$){ $\langle label \rangle$ }



```

\begin{pspicture}(3,1.5)
\optswitch(0,1)(3,1){opened switch}
\end{pspicture}

```

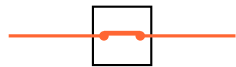
opened switch

switchsize= $\langle num \rangle$ or $\langle width \rangle \langle height \rangle$ default: 0.8

A single number defines the side length of the switch, otherwise the height and width can be specified separately.

switchstyle=opened, closed default: opened

Indicate the switch as opened or closed.



```

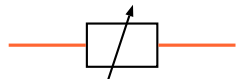
\begin{pspicture}(3,1.5)
\optswitch[switchstyle=closed](0,1)(3,1){closed switch}
\end{pspicture}

```

closed switch

5.7. Fiber delay-line

\fiberdelayline[$\langle options \rangle$]($\langle in \rangle$)($\langle out \rangle$){ $\langle label \rangle$ }



```

\begin{pspicture}(3,1.5)
\fiberdelayline(0,1)(3,1){delay line}
\end{pspicture}

```

delay line

fdlsize= $\langle num \rangle$ or $\langle width \rangle \langle height \rangle$ default: 0.6


A single number gives the component height, the width is 1.6 times the height.
Two numbers define width and height directly.

FdlArrow $\langle psstyle \rangle$ default: arrowinset=0, arrows=->

See **ArrowCompStyle**.

5.8. Polarizer

\optfiberpolarizer [$\langle options \rangle$] ($\langle in \rangle$) ($\langle out \rangle$) { $\langle label \rangle$ }



```

\begin{pspicture}(3,1.5)
  \optfiberpolarizer(0,1)(3,1){polarizer}
\end{pspicture}

```

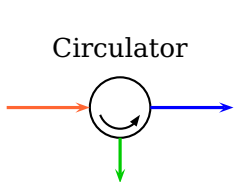
polarizer

fiberpolsize= $\langle num \rangle$ or $\langle width \rangle \langle height \rangle$ default: 0.6

A single number gives the component height, the width is 1.6 times the height.
Two numbers define width and height directly.

5.9. Optical circulator

\optcirculator ($\langle left \rangle$) ($\langle right \rangle$) ($\langle bottom \rangle$) { $\langle label \rangle$ }



```

\begin{pspicture}(3,2)
  \addtopsstyle{Fiber}{arrows=->}
  \addtopsstyle{FiberOut1}{linecolor=blue}
  \addtopsstyle{FiberOut2}{linecolor=green!80!black}
  \optcirculator(0,1)(3,1)(1.5,0){Circulator}
\end{pspicture}

```

Circulator

optcircsize= $\langle num \rangle$ default: 0.8

The diameter of the circulator.

optcircangleA= $\langle num \rangle$ default: -160

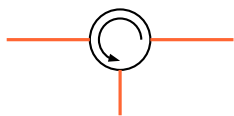
The starting angle of the internal arrow.

optcircangleB= $\langle num \rangle$ default: -20

The ending angle of the internal arrow.

`optcircangle=<num> <num>`

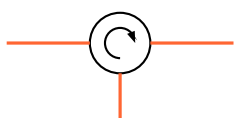
Short notation to specifying both angles.



```
\begin{pspicture}(3,1.5)
  \optcirculator[optcircangle=0 -90](0,1)(3,1)(1.5,0)
\end{pspicture}
```

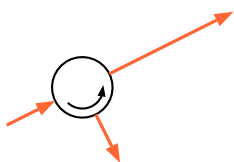
`OptCircArrow <psstyle>` default: unit=0.7, arrows=->, arrowinset=0

The style of the internal arrow. It specifies the direction of the arrow and the size of the arc.



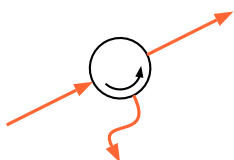
```
\begin{pspicture}(3,1.5)
  \newpsstyle{OptCircArrow}{unit=0.5, arrows=<-, arrowinset=0}
  \optcirculator[optcircangle=0 -90](0,1)(3,1)(1.5,0)
\end{pspicture}
```

The circulator is positioned by default such that the input and output fibers are orthogonal to each other, like shown in the example:



```
\begin{pspicture}(3,2)
  \addtopsstyle{Fiber}{arrowscale=1.3, arrows=->, arrowinset=0}
  \optcirculator(0,0.5)(3,2)(1.5,0)
\end{pspicture}
```

The positioning parameters (Sec. 3.2) refer to the reference nodes *<left>* and *<right>* as usual and are not related to the automatic position. This implies, that the default position is not equivalent to `position=0.5`.

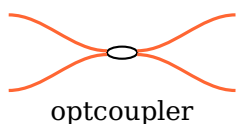


```
\begin{pspicture}(3,2)
  \addtopsstyle{Fiber}{arrowscale=1.3, arrows=->, arrowinset=0,
    ncurv=1.5}
  \optcirculator[position=0.5](0,0.5)(3,2)(1.5,0)
\end{pspicture}
```

5.10. Fiber coupler

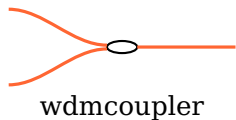
The package provides three fiber couplers which all share the same shapes and parameters.

`\optcoupler (<tl>)(<bl>)(<tr>)(
){<label>}`



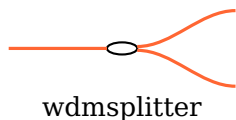
```
\begin{pspicture}(0,-0.4)(3,1)
  \optcoupler(0,1)(0,0)(3,1)(3,0){optcoupler}
\end{pspicture}
```

`\wdmcoupler`($\langle tl \rangle$)($\langle bl \rangle$)($\langle r \rangle$){ $\langle label \rangle$ }



```
\begin{pspicture}(0,-0.4)(3,1)
  \wdmcoupler(0,1)(0,0)(3,0.5){wdmcoupler}
\end{pspicture}
```

`\wdmsplitter`($\langle l \rangle$)($\langle tr \rangle$)($\langle br \rangle$){ $\langle label \rangle$ }



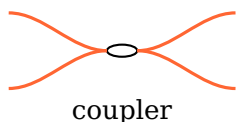
```
\begin{pspicture}(0,-0.4)(3,1)
  \wdmsplitter(0,0.5)(3,1)(3,0){wdmsplitter}
\end{pspicture}
```

`couplersize`= $\langle num \rangle$ or $\langle width \rangle$ $\langle height \rangle$ default: 0.2

For a single number the width is twice this value, the height 0.8 times this value. Two numbers define width and height directly.

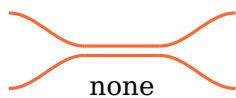
`couplersep`= $\langle num \rangle$ default: 0.05

The vertical distance between two fiber ports.

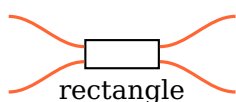


```
\begin{pspicture}(0,-0.5)(3,1)
  \optcoupler[couplersep=0](0,1)(0,0)(3,1)(3,0){coupler}
\end{pspicture}
```

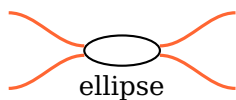
If this value is set to {}, the separation is calculated automatically according to the `couplertype` and `couplersize`.



```
\begin{pspicture}(3,4.5)
  \psset{couplersep={}, couplersize=0.5, labeloffset=0.5}
  \optcoupler[couplertype=none]%
    (0,4.5)(0,3.5)(3,4.5)(3,3.5){none}
```



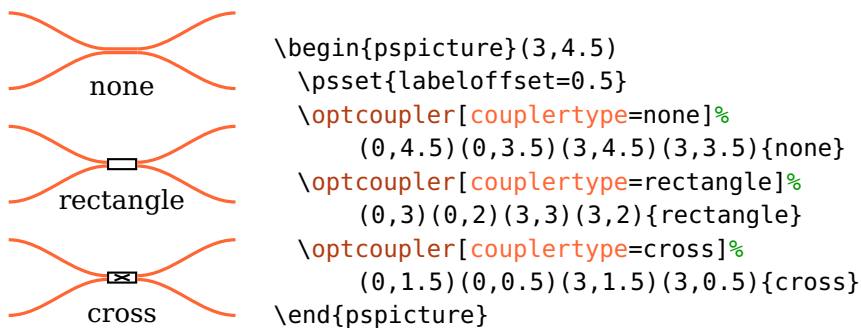
```
\optcoupler[couplertype=rectangle]%
  (0,3)(0,2)(3,3)(3,2){rectangle}
```



```
\optcoupler[couplertype=ellipse]%
  (0,1.5)(0,0.5)(3,1.5)(3,0.5){ellipse}
\end{pspicture}
```

`couplertype`=none, ellipse, rectangle, cross default: ellipse

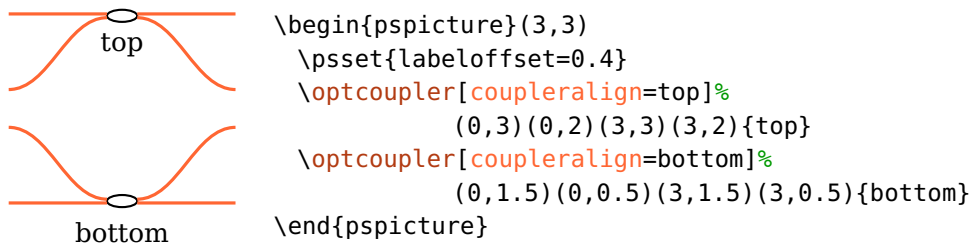
Select between different coupler types.



coupleralign=t, top, b, bottom, c, center

default: center

The alignment of the coupler with respect to the reference nodes. Note, that the exact position of the coupler center and accordingly that of the label depends on this setting (see 8.3).



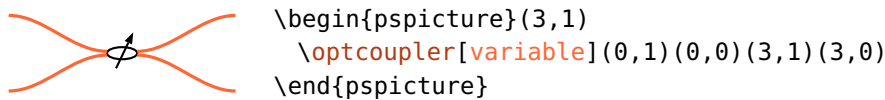
align=top, bottom, center

default: center

This parameter was renamed in version 3.0 to **coupleralign** and is deprecated.

variable=true, false

default: true

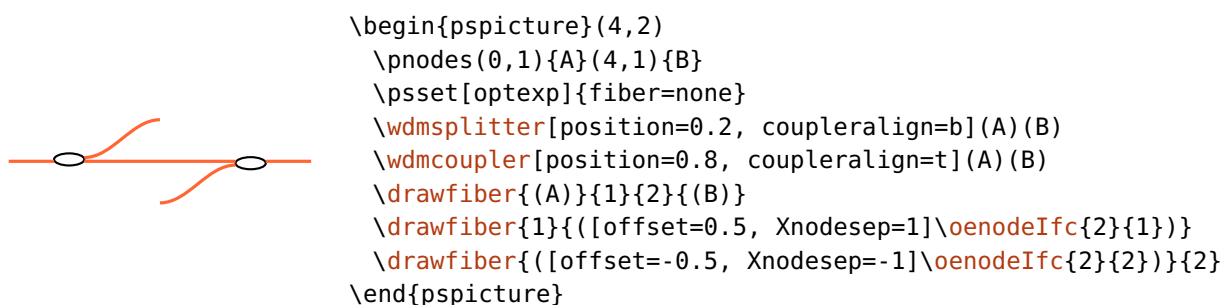


VariableCoupler *<psstyle>*

default: arrowinset=0, arrows=->

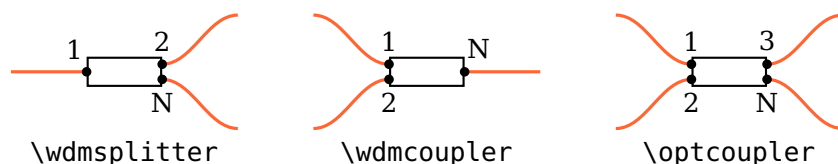
The style of the arrow of the variable coupler.

Every coupler has a variant which needs only two nodes, one input and one output node. For **\optcoupler**, both nodes are used twice, **\wdmsplitter** uses node *<out>* twice, and **\wdmcoupler** uses node *<in>* twice. This can be very useful if you do not use the automatic connections but connect the components manually.



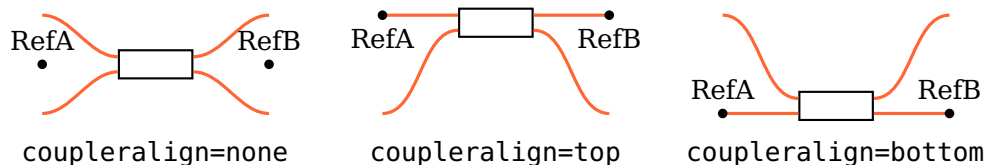
5.10.1. Input and output nodes

The definition of input and output nodes from Sec. 8.6 cannot be applied to couplers. Here, the nodes are simply numbered from 1 (left top) to N (right bottom).



5.10.2. Reference nodes

The definition of reference nodes from Sec. 8.2 cannot be applied to couplers and depends on `coupleralign`.



5.11. Fiber box

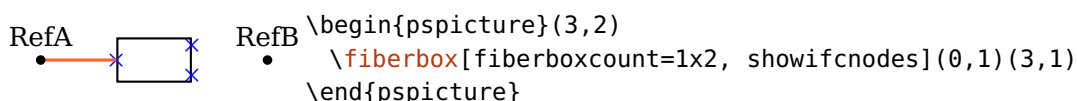
```

\fiberbox((in))((out)){\label}
\fiberbox((tl))((bl))((tr))((br)){\label}

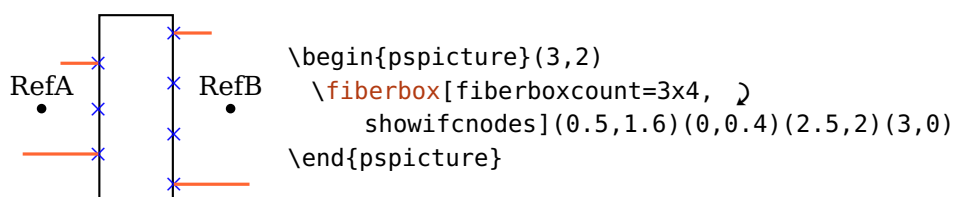
```

A `\fiberbox` can be positioned either with two, or four nodes which affects the positioning of the component and the interface nodes, and the automatic fiber connections.

If only two nodes are used these are the reference nodes and the separation between the components input and output nodes must be specified manually (see `fiberboxsepin` and `fiberboxsepout`). Only the side of the component is connected automatically which has a single node only (see `fiberboxcount` and the following example).



When four nodes are used the interface nodes and their separation are determined automatically. The component is positioned between the center of the two input and the center of the two output nodes. The first input interface node is aligned with the `<tl>` node, the last input node is aligned with `<bl>`, all other input nodes are equally distributed between these. The output nodes are handled accordingly. The four specified nodes are connected automatically to the component, each with the interface node it is aligned to.

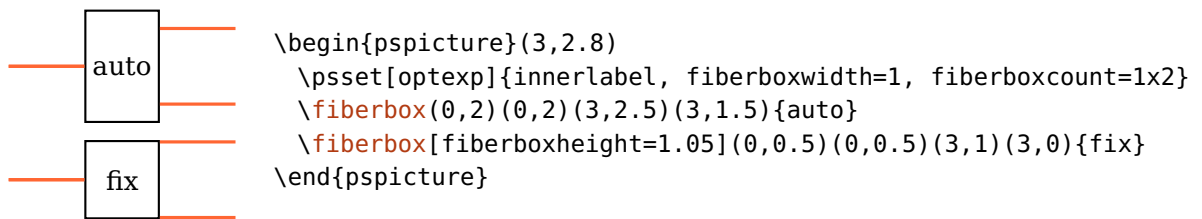


`fiberboxwidth=<num>` default: 1

The width of the fiber box.

`fiberboxheight=<num>` default: 0

The height of the fiber box. If the height is lower than the space required by the input or output nodes, it is calculated automatically. The calculated height is $\max(\text{sepin} \times (N - 1), \text{sepout} \times (M - 1))$.



`fiberboxsize`= $\langle width \rangle \langle height \rangle$

The width and height of the fiber box, is equivalent to calling both `fiberboxwidth` and `fiberboxheight`.

`fiberboxsepin`= $\langle num \rangle$ default: 0.2

The separation between two input nodes. This value is used only if the fiber box is positioned with two nodes. Otherwise the separation is calculated automatically.

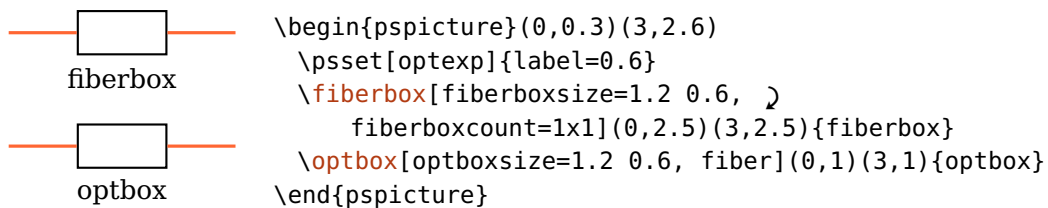
`fiberboxsepout`= $\langle num \rangle$ default: 0.2

Equivalent to `fiberboxsepout`, but for the output nodes.

`fiberboxcount`= $\langle N \rangle \times \langle M \rangle$ default: 2x2

Number of input ($\langle N \rangle$) and output nodes ($\langle M \rangle$).

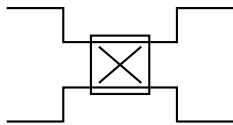
For certain parameter configurations a `\fiberbox` is equivalent to an `\optbox`.



6. Electrical components

6.1. Coupler

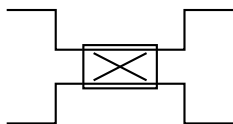
`\eleccoupler`($\langle tl \rangle$)($\langle bl \rangle$)($\langle tr \rangle$)($\langle br \rangle$){ $\langle label \rangle$ }



```
\begin{pspicture}(3,1.5)
\eleccoupler(0,1.5)(0,0)(3,1.5)(3,0)
\end{pspicture}
```

`eleccouplersize`= $\langle size \rangle$ or $\langle width \rangle$ $\langle height \rangle$ default: 0.8

The width and height of the couplers, if only one value is given the component is squared.



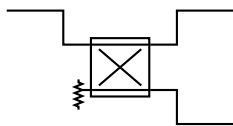
```
\begin{pspicture}(3,1.5)
\eleccoupler[eleccouplersize=1 0.6](0,1.5)(0,0)(3,1.5)(3,0)
\end{pspicture}
```

`eleccouplersep`= $\langle num \rangle$ default: {}

The vertical distance between the two input and output ports. If the value is empty {}, then the separation is set to 75% of the height.

`eleccouplertype`=standard, directional default: standard

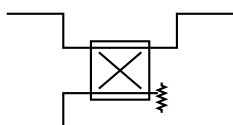
Select between different coupler types.



```
\begin{pspicture}(3,1.5)
\eleccoupler[eleccouplertype=directional](0,1.5)(0,0)(3,1.5)(3,0)
\end{pspicture}
```

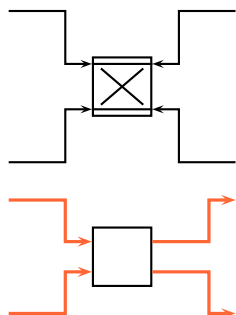
`eleccouplerinput`=left, right default: left

Selects the input side for the directional coupler.



```
\begin{pspicture}(3,1.5)
\eleccoupler[eleccouplertype=directional,
\eleccouplerinput=right](0,1.5)(0,0)(3,1.5)(3,0)
\end{pspicture}
```

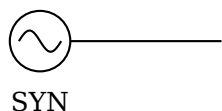

The automatic connections of the electrical coupler are all directed towards the component, in contrast to the optical coupler. This gives a symmetric layout of the automatic electrical coupler connections when using the default of `wirestyle=angle`.



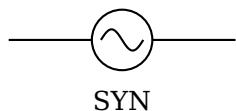
```
\begin{pspicture}(3,4)
  \addtopsstyle{Wire}{arrows=->}
  \addtopsstyle{Fiber}{arrows=->, fiberstyle=angle}
  \eleccoupler(0,4)(0,2)(3,4)(3,2)
  \optcoupler[couplersize=0.8 0.8, couplersep={}, 2
    couplertype=rectangle](0,1.5)(0,0)(3,1.5)(3,0)
\end{pspicture}
```

6.2. Synthesizer

`\elecsynthesizer` ($\langle in \rangle$) ($\langle out \rangle$) { $\langle label \rangle$ }



```
\begin{pspicture}(3,1.5)
  \elecsynthesizer(1,1)(3,1){SYN}
\end{pspicture}
```



```
\begin{pspicture}(3,1.5)
  \elecsynthesizer[position=0.5](0,1)(3,1){SYN}
\end{pspicture}
```

`synthsize`= $\langle size \rangle$ or $\langle width \rangle$ $\langle height \rangle$

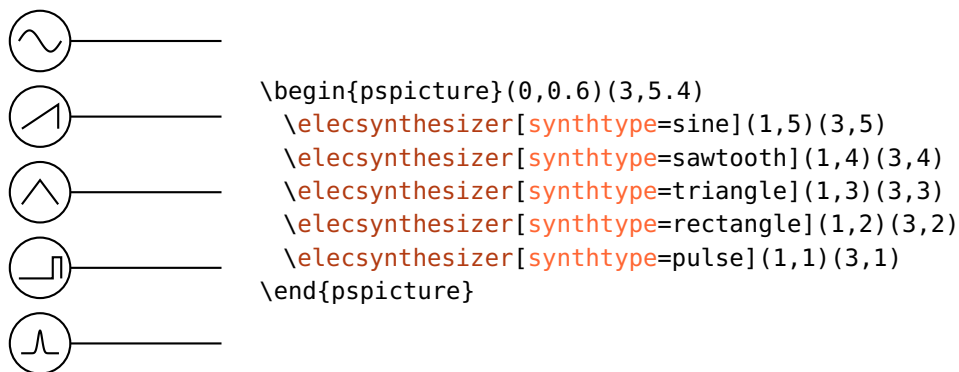
default: 0.8

The size of the component, if this is of circular shape and two values were given, the smaller one is used as diameter.

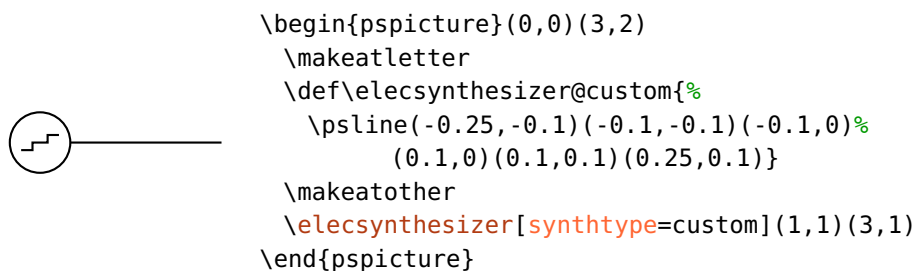
`synthtype`=sine, pulse, sawtooth, rectangle, triangle, custom

default: sine

Select between different synthesizer types.



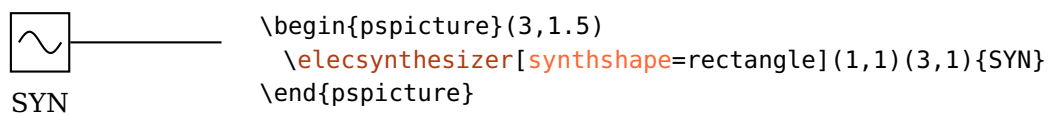
You can also use a freely customizable waveform.



synthshape=circle, rectangle

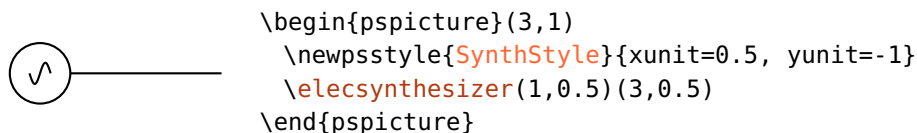
default: circle

Select between circular and rectangular outer shape.



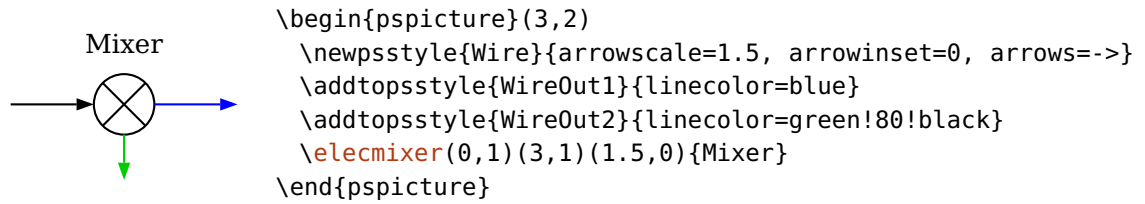
SynthStyle $\langle psstyle \rangle$

Change the style of the internal synthesizer symbol.



6.3. Mixer

\elec mixer ($\langle left \rangle$) ($\langle right \rangle$) ($\langle bottom \rangle$) { $\langle label \rangle$ }



`elecmmixer`size= $\langle num \rangle$

default: 0.8

The diameter of the mixer.

For further details about the reference nodes and the positioning, please refer to `\optcirculator`.


7. Hybrid components

This chapter describes the components which can be used both for free-ray and fiber optics but which differ from these two categories.

The `\optfilter` is connected by default with automatic fiber connections, but can likewise be used with free-ray beams, in contrast to the fiber components (Sec. 5). The `\fibercollimator` has one interface only, the other connection is a fiber and the `\optdetector` has one interface, the other connection is a wire.

7.1. Optical filter

`\optfilter`[*<options>*](*<in>*)(*<out>*){*<label>*}


	<pre>\begin{pspicture}(3,1.5) \optfilter(0,1)(3,1){bandpass} \end{pspicture}</pre>
bandpass	


`filtersize`=*<num>* default: 0.8


The size of the filter.

`filtertype`=bandpass, bandstop, lowpass, highpass default: bandpass

Select between different filter types.

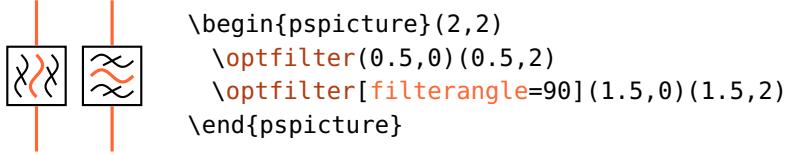
	<pre>\begin{pspicture}(3,1.5) \optfilter[filtertype=bandstop](0,1)(3,1){bandstop} \end{pspicture}</pre>
bandstop	

	<pre>\begin{pspicture}(3,1.5) \optfilter[filtertype=lowpass](0,1)(3,1){lowpass} \end{pspicture}</pre>
lowpass	

	<pre>\begin{pspicture}(3,1.5) \optfilter[filtertype=highpass](0,1)(3,1){highpass} \end{pspicture}</pre>
highpass	

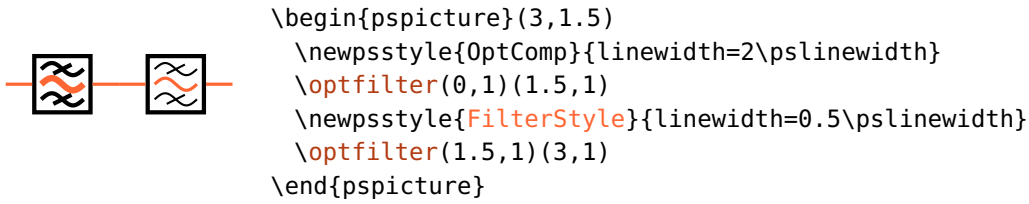
`filterangle=<num>` default: 0

Rotates the “inner” part of the filter relativ to its frame. Alternatively `innercompalign` can be used.

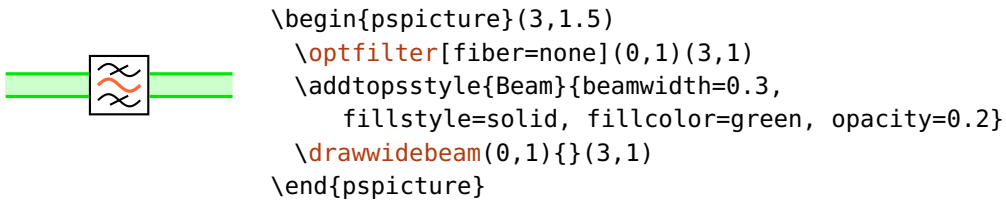


`FilterStyle <psstyle>`

Change the style of the internal filter lines.

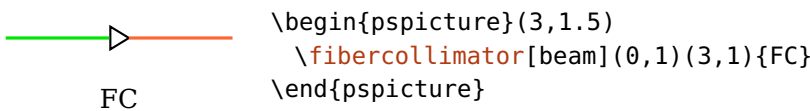


Usage as free-ray component, `allowbeaminside` is set to false by default:

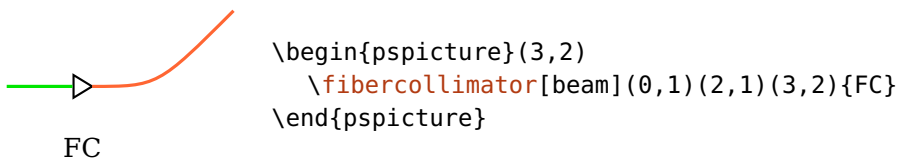


7.2. Fiber collimator

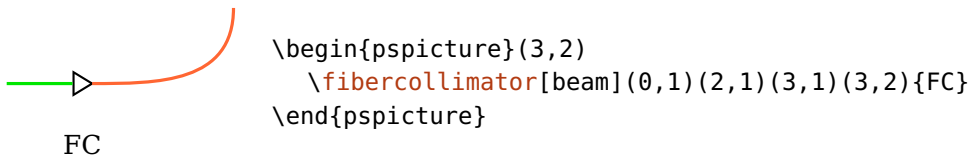
`\fibercollimator (<in>)(<A>)()(<out>){<label>}`



The fiber collimator can be used with two, three or four points. With two points, the collimator is placed like any other dipole component between `<in>` and `<out>` node. For three nodes, the fiber is drawn as `\psbezier` curve for which the central node `<A>` is used twice. Positioning parameters (see Sec. 3.2) can be used to shift the object between `<in>` and `<A>` nodes.



For four nodes, the fiber is drawn as `\psbezier` curve with the specified nodes. Positioning parameters (siehe Sec. 3.2) can be used to shift the object between the first two nodes (`\in`) and (`A`).

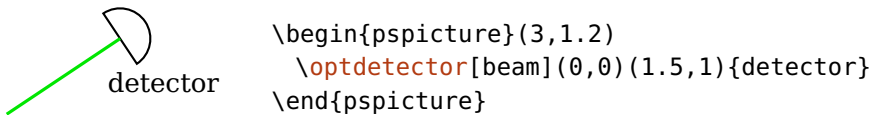


`\fibercolsize=<num>` or `<width> <height>` default: 0.3

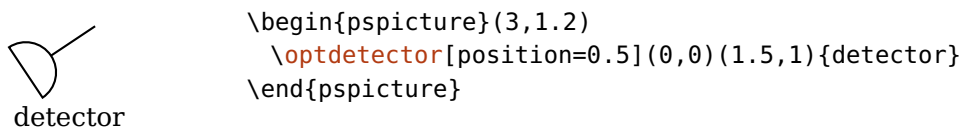
A single number gives the side length of the collimator, two numbers the width and height. Note, that `\fibercolsize=1` and `\fibercolsize=1 1` do not give the same result. You may also change the relation of width and height with `xunit` and `yunit`.

7.3. Detector

`\optdetector[<options>](<in>)(<out>){<label>}`



The detector is placed by default at the end of the reference line (`\position=end`). If this setting is overwritten, the output connection is treated like that of an electrical component.

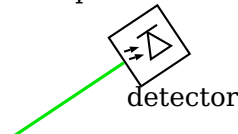


detsize= $\langle num \rangle$ or $\langle width \rangle \langle height \rangle$ default: 0.8

If a single number is given it is the side length (diode) or the diameter (round) of the detector. Two numbers define the width and height of the detector. Note, that detsize=1 and detsize=1 1 are not equivalent for dettype=round.


dettype=round, diode default: round

This parameter selects between different types of detectors.

 `\begin{pspicture}(3,1.5)`
`\optdetector[beam, dettype=diode](0,0)(1.5,1){detector}`
`\end{pspicture}`

DetectorStyle $\langle psstyle \rangle$

The style of the diode for **dettype**=diode.

 `\begin{pspicture}(3,1.5)`
`\newpsstyle{OptComp}{linewidth=3\pslinewidth}`
`\newpsstyle{DetectorStyle}{linewidth=0.333\pslinewidth}`
`\optdetector[dettype=diode](0,1)(2,1)`
`\end{pspicture}`

8. Special nodes

Every `pst-optexp` object of an experimental setup provides several special nodes which are related to its geometry and positioning. They can be accessed and used for related positioning and drawing.

You should always use the dedicated macros to access these node names.

`\oenode{<node>}{<comp>}`

This is the basic command to access any node associated with a certain component. The first argument `<node>` is the identifier of the requested node. The second argument `<comp>` is the name of the target component (according to Sec. 8.1). If left empty, it uses the last component defined.

For most special nodes an appropriate macro is provided, which you are strongly advised to use, because the naming conventions may change. Using the macros makes sure, that you always get the correct node names. Therefore, the available identifiers are not listed explicitly.

`namingscheme=old, new`

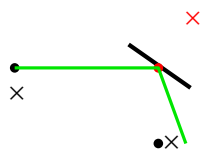
default: new

This option is for backward compatibility only. In version 2.1 special component nodes had to be accessed by their explicit name. You should use this option only if you accessed internal nodes directly in old documents. Since version 3.0 explicit macros are provided to access all special component nodes, so that the actual naming scheme does not matter.

`showoptdots=true, false`

default: false

Draw some special component nodes for debugging: The black points are the normal and the black crosses are the transformed reference nodes (Sec. 8.2), the red point is the center node (Sec. 8.3), and the red cross is the label node (Sec. 8.4).



```
\begin{pspicture}(3,2)
\mirror[showoptdots, angle=10, beam](0,1)(1.9,1)(1.9,0)
\end{pspicture}
```


8.1. Component identifiers

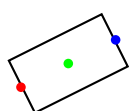
All components of a setup drawing are numbered automatically in increasing order according to their definition in the code, starting with 1. The components as well as their special nodes can always be accessed by this number (ID).

ID specifications relative to the current ID can be used with the syntax $\langle num \rangle$: $\langle 1 \rangle$ is the previous ID, $\langle 2 \rangle$ is the current ID minus two etc. The previous ID can also be accessed by only $\langle \rangle$, and the current ID is $\langle 0 \rangle$, which is equivalent to an empty parameter.

comname= $\langle string \rangle$

Assigns a name identifier to a component, which can then be referenced both by this name and by its ID. The parameter can be assigned only locally, i.e. in the optional argument of a `pst-optexp` component, and should be unique within one `pspicture` environment.

Examples for different methods to access the same component.



```
\begin{pspicture}(2,2)
  \optbox[comname=MyBox](0,1)(2,2)
  \psdot[linecolor=red](\oenameIn{MyBox})% use the comname
  \psdot[linecolor=blue](\oenameOut{1}) % use the ID
  \psdot[linecolor=green](\oenameCenter{}) % the last component
\end{pspicture}
```

Using relative IDs can be very handy to define a component relative to the previous one. It is not possible to use e.g. `\oenameOut{}` as reference node, because at the time these coordinates are evaluated, they reference to the new, not yet defined component. A call like `\optbox(\oenameOut{})(B)` is, therefore, not possible and gives an error. With relative IDs one can achieve that with `\optbox(\oenameOut{<})(B)`:



```
\begin{pspicture}(4,2)
  \psset{optboxsize=0.4 0.8, abspos=0.5}
  \optbox(0,1)(4,1)
  \multido{}{3}{\optbox(\oenameOut{<})(4,1)}
\end{pspicture}
```

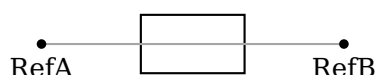
8.2. Reference nodes

`\oenodeRefA{<comp>}`

`\oenodeRefB{<comp>}`

The input and output reference nodes.

These are the original nodes which were used for the component positioning, `\oenodeRefA` is the first and `\oenodeRefB` the last node. This definition is not valid couplers (5.10.2) and circulators (5.9).



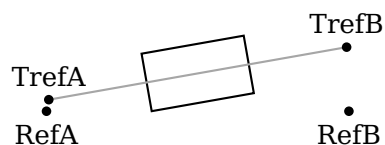
```
\begin{pspicture}(5,1)
  \pnodes(0.5,0.5){A}(4.5,0.5){B}
  \optbox(A)(B)
  \psline[style=Refline](\oenodeRefA){}(\oenodeRefB){}
  \psdot(\oenodeRefA){}\uput[-90](\oenodeRefA){RefA}
  \psdot(\oenodeRefB){}\uput[-90](\oenodeRefB){RefB}
\end{pspicture}
```

`\oenodeTrefA{<comp>}`

`\oenodeTrefB{<comp>}`

The transformed input and output reference nodes.

These are the input and output reference nodes which are transformed together with the component according to the `compshift` and `angle` parameters.

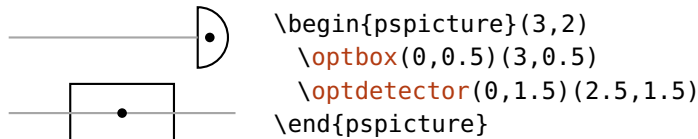


```
\begin{pspicture}(5,1.8)
  \pnodes(0.5,0.5){A}(4.5,0.5){B}
  \optbox[compshift=0.5, angle=10](A)(B)
  \psdot(\oenodeRefA){}\uput[-90](\oenodeRefA){RefA}
  \psdot(\oenodeRefB){}\uput[-90](\oenodeRefB){RefB}
  \psdot(\oenodeTrefA){}\uput[90](\oenodeTrefA){TrefA}
  \psdot(\oenodeTrefB){}\uput[90](\oenodeTrefB){TrefB}
\end{pspicture}
```

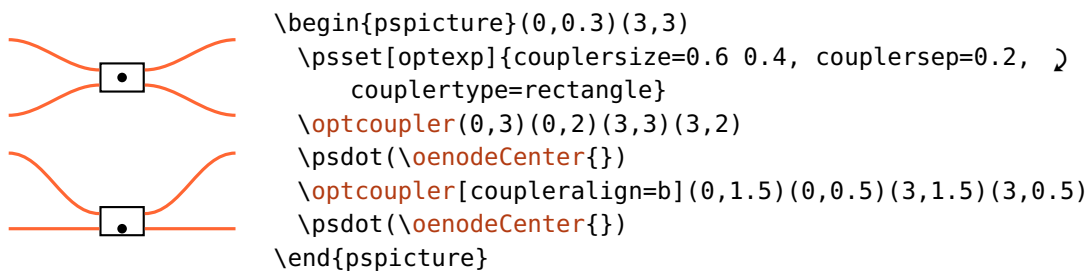
8.3. Center node

`\oenodeCenter{<comp>}`

This node lays, except for a few components like e.g. `\optcirculator` and `\optcoupler` (see below), in the center of the component.



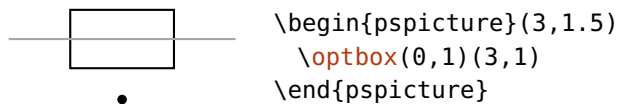
For `\optcoupler` the position of the component center depends on the `coupleralign` settings.



8.4. Label node

`\oenodeLabel{<comp>}`

The component label is placed at this node. It is also available if no label was specified. The label node is identical with the center node for `labeloffset=0`.



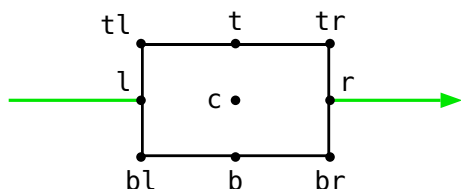
8.5. External nodes

`\oenodeExt{<comp>}`

An external node can be placed at different positions along the component boundary. It gets defined only for external usage and does not affect the component in any way.

extnode= \langle refpoint \rangle

Set the position of the external node. Like the reference point of `\rput`, can be any combination of c (center), t (top), b (bottom), l (left), and r (right). Not all components support any possible combination. The allowed positions of each component are listed in Sec. 12.2.1.

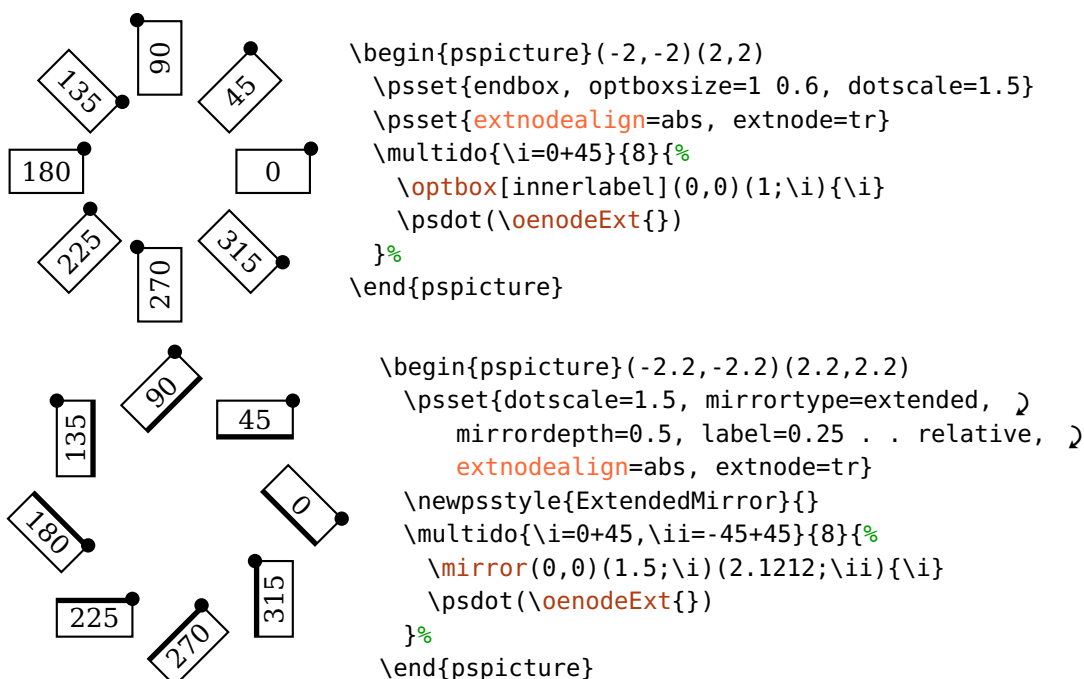


extnodealign=rel, relative, abs, absolute

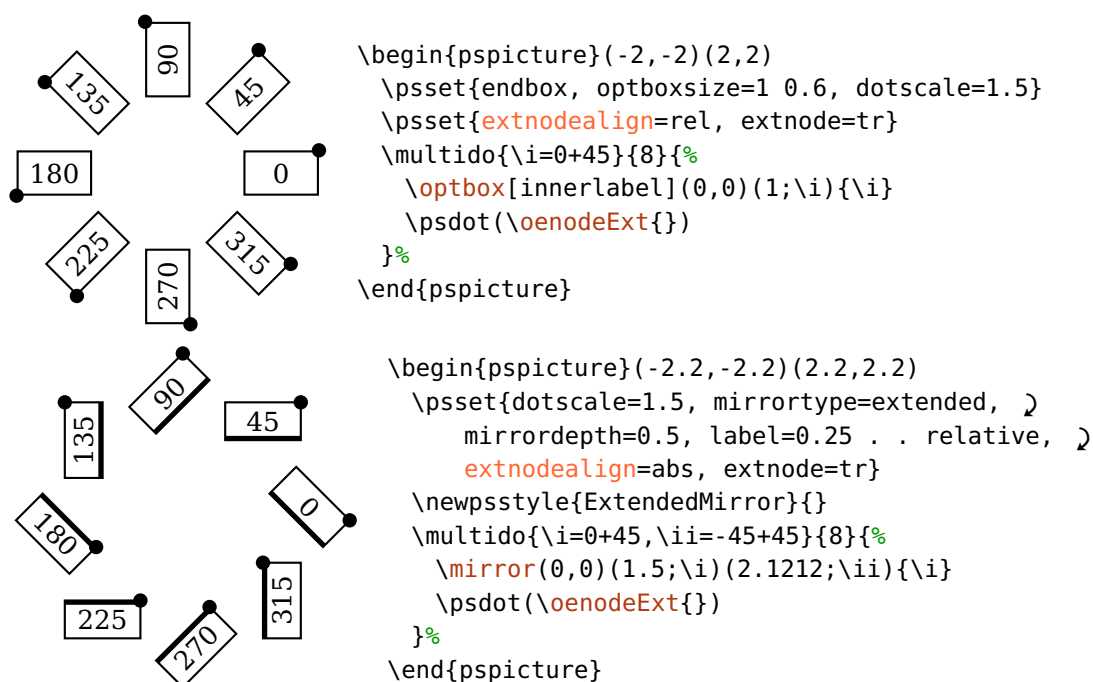
default: abs

The definition of “top” (t) and the other **extnode** refpoint parameters can be absolute or relative to the component.

In the following example the external node is always placed “top right” independent of the actual order of the reference nodes (**extnodealign**=abs). The behaviour is identical to the label rotation for **labelref**=relative.

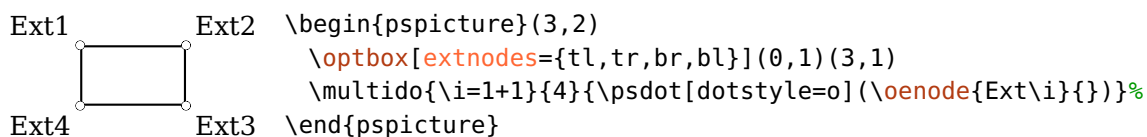


In the next example the position is relative to the order of input and output reference node (**extnodealign**=rel).



`extnodes=<list>`

Define multiple external nodes, with their *<refpoints>* given in the *<list>* and separated by commas. The first node can be accessed as usual with `\oenodeExt` (is equivalent to `\oenode{Ext}` and `\oenode{Ext1}`). All additional nodes must be accessed with `\oenode` and the node label `Ext<num>`: `\oenode{Ext1}` for the first one, `\oenode{Ext2}` for the second, etc.



8.6. Interface nodes

`\oenodeIfc{<num>}{<comp>}`

The interface node *<num>*, where the number is a positive integer or the character N. The last node is always defined as N but can also be accessed via its number.

`\oenodeIn{<comp>}`

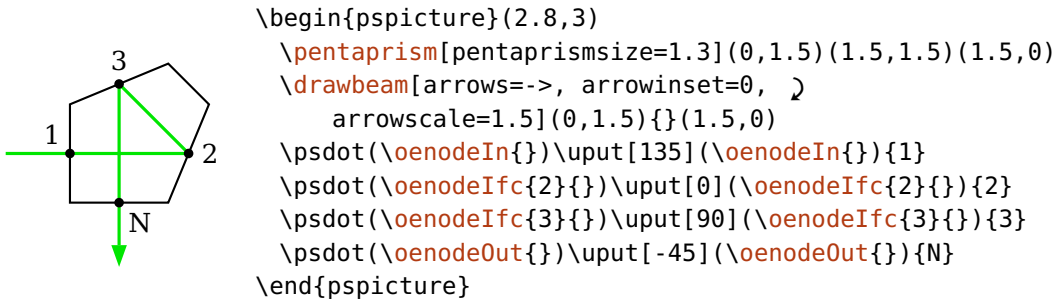
The interface node $\langle 1 \rangle$ is the input node and should always be accessed via `\oenodeIn`.

`\oenodeOut{<comp>}`

The interface node $\langle N \rangle$ is the output node and should always be accessed via `\oenodeOut`.

“Input” and “output” can define only relative orientations. The input node is by definition the node to which a beam coming from the reference node `\oenodeRefA` is connected to. Correspondingly is the definition of the output node. This definition breaks down only for fiber couplers (see Sec. 5.10.1) and beamsplitter (4.17).

`\oenodeIn` is equivalent to `\oenodeIfc{1}` and `\oenodeOut` is equivalent to `\oenodeIfc{N}`. You should use the `\oenodeIfc` macro only to access the nodes which are not the input and output nodes as for those explicit macros are provided.



See Sec. 12.2.2 for a complete list of all components with their interface nodes.

8.7. Rotation reference node

`\oenodeRotref{<comp>}`

The node around which a component is rotated by `angle`. The position is defined with the `rotateref` parameter and can take the same values as `extnode`. See Sec. 12.2.1 for possible rotation reference nodes of all components.



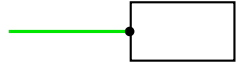
```
\begin{pspicture}(0,0.1)(12,1.8)
  \optbox[angle=20](0.5,1)(4.5,1)
  \psdot(\oencodeRotref{})\uput[-90](\oencodeRotref{}){Rotref}
  \optbox[angle=20, rotateref=bl](7.5,1)(11.5,1)
  \psdot(\oencodeRotref{})\uput[-90](\oencodeRotref{}){Rotref}
\end{pspicture}
```

8.8. Beam nodes

The end points of a recent `\drawbeam` or `\drawwidebeam` command can be reused if `savebeampoints` is set.

`\oencodeBeam{<num>}`

Access the end point of a preceeding `\drawbeam` command. If `<num>` is left empty it defaults to 1 which usually refers to the last beam ray. However, generally it depends on the settings of `savebeampoints` and the choice of `<num>` which beam you refer to.

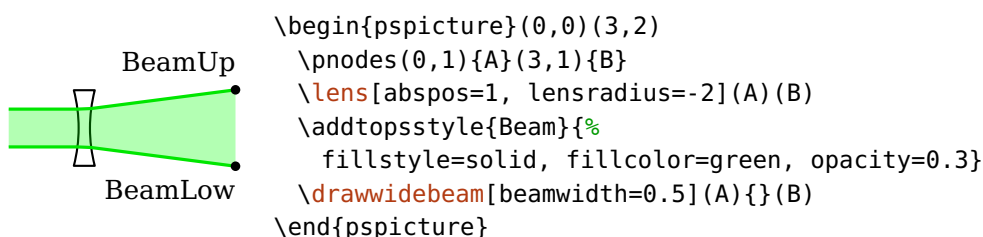


```
\begin{pspicture}(0,0)(3,2)
  \pnodes(0,1){A}(1.6,1){B}
  \optbox[position=end](A)(B)
  \drawbeam[beaminsidelast=false](A){1}
  \psdot(\oencodeBeam{})
\end{pspicture}
```

`\oencodeBeamUp{<num>}`

`\oencodeBeamLow{<num>}`

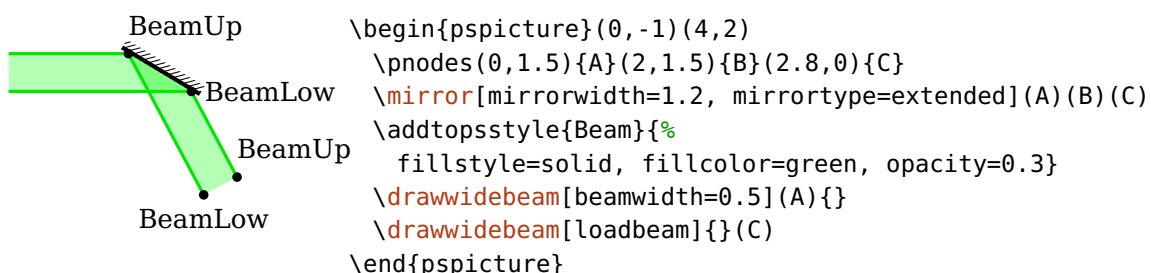
Access the upper or lower end point of a preceeding `\drawwidebeam` command. If `<num>` is left empty it defaults to 1, which usually refers to the last beam ray. However, generally it depends on the settings of `savebeampoints` and the choice of `<num>` which beam you refer to.



Note, that “upper” and “lower” is defined with respect to the propagation direction of the beam. A marginal beam which started as “upper” can change to “lower” if it crosses the other marginal beam. This is shown in the following example, where the marginal rays exchange their roles:



After a reflection the roles can also be exchanged.



Note, that the orientation of “upper” and “lower” changed with version 3.3, for more details see Sec. 12.3.5.

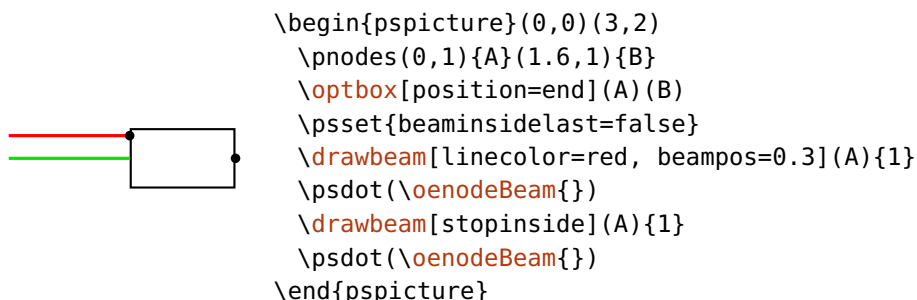
`\oeBeamCenter{<num>}`

Get the mean of upper and lower beam node. The coordinates are available only at Postscript level, in contrast to the `\oenodeBeam*` macros. For an example see Ex. 8.1.

The beam nodes can be accessed only within the `pspicture` environment they were defined in. If you want to use one of these nodes from outside, you must explicitly redefine them beforehand:

```
\pnode(\oenodeBeam{}){MyBeamNode}
```


The end points are also affected by `stopinside`. See Sec. 9.5 for further details.



8.9. Beam vector

In addition to the beam nodes (see Sec. 8.8) also the normalized vectors of the last saved beams (see `savebeam`) can be accessed at Postscript level.

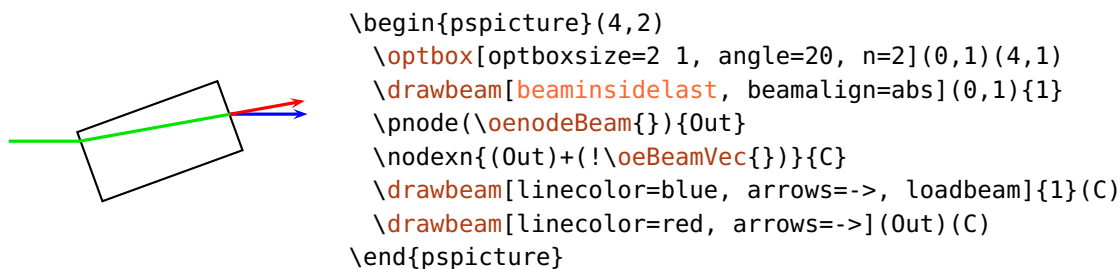
These are again separated in single, upper and lower beams. $\langle num \rangle$ has the same meaning as for `\oenodeBeam`.

`\oeBeamVec` $\{\langle num \rangle\}$

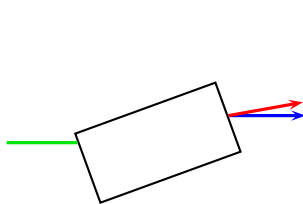
`\oeBeamVecUp` $\{\langle num \rangle\}$

`\oeBeamVecLow` $\{\langle num \rangle\}$

The saved values are those of the normalized direction vector of the last beam segment and are drawn in red in the next example. In contrast to this, the vector saved with `savebeam` is that behind the last interface and would be used for a following beam. This is shown in blue in the following example.

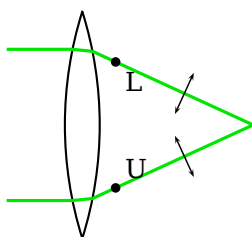


This behaviour is independent of whether the last beam was only calculated or also drawn. The next example uses `stopinside` instead of `beaminsidelast` compared to the previous example; the beam vectors are not affected by this.



```
\begin{pspicture}(4,2)
\optbox[optboxsize=2 1, angle=20, n=2](0,1)(4,1)
\drawbeam[stopinside, beamalign=abs](0,1){1}
\pnode(\oenodeBeam){Out}
\nodexn{(Out)+(!\oeBeamVec{)}}{C}
\drawbeam[linecolor=blue, arrows=->, loadbeam]{1}(C)
\drawbeam[linecolor=red, arrows=->](0,1)(C)
\end{pspicture}
```

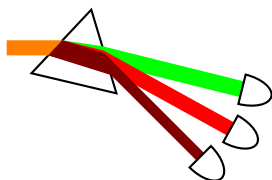
A possible application is marking of the polarization of a beam. In the following example we define the nodes $\langle U \rangle$ and $\langle L \rangle$ which are shifted with `\nodexn` from the end point of the upper and lower beam along the respective beam trajectory.



```
\begin{pspicture}(3,3)
\pnodes(0,1.5){A}(3,1.5){B}
\lens[abspos=1, lens=5 5 3, n=2.0](A)(B)
\optplane[angle=90](B)
\drawwidebeam[beamwidth=2](A){1-2}
\nodexn{(\oenodeBeamUp{))-2(!\oeBeamVecUp{)}}{U}
\nodexn{(\oenodeBeamLow{))-2(!\oeBeamVecLow{)}}{L}
\polarization(U)(\oenodeBeamUp{))
\polarization(L)(\oenodeBeamLow{))
\end{pspicture}
```

`\oeBeamVecMedian`{ $\langle num \rangle$ }The mean of upper and lower beam vector.

Ex. 8.1

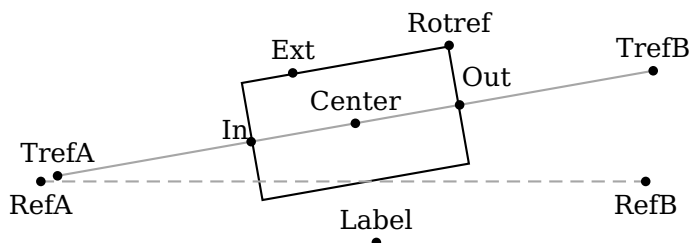


```
\begin{pspicture}(3.5,2.6)
\pnodes(0,2){A}(1,2){B}(3,1){C}
\optprism(A)(B)(C)
\newsstyle{Beam}{linestyle=none, fillstyle=solid,
beamwidth=0.2, beamnodealign=vector}
\drawwidebeam[fillcolor=orange](A){}
\psset[optexp]{beampathskip=1}
\drawwidebeam[fillcolor=green, n=1.2, savebeam=2](A){}(C)
\drawwidebeam[fillcolor=red, n=1.4, savebeam=3](A){}(C)
\drawwidebeam[fillcolor=red!50!black, n=1.57,
savebeam=4](A){}(C)
\multido{\i=2+1}{3}{%
\backlayer{%
\pnode(!\oeBeamCenter{\i}){Det\i}
\nodexn{(Det\i)-(!\oeBeamVecMedian{\i})}{Det\i'}}
\optdetector[detsize=0.4 0.4](Det\i')(Det\i)}
\end{pspicture}
```

Further example are Ex. 11.21 and Ex. 11.27.

8.10. Node overview

This is an overview of all available component nodes.



```
\begin{pspicture}(0.3,0.1)(4.8,1.6)
  \pnodes(0.5,0.5){A}(4.5,0.5){B}
  \optbox[compshift=0.5, angle=10, rotateref=tr, extnode={-0.5,1}](A)(B)
\end{pspicture}
```

9. Connecting components


The `pst-optexp` package provides several methods for automatic and manual connections of components with beams, fibers, or wires.

- Sec. 9.1 describes how to access objects and nodes for use with the connection macros.
- Beam drawing in general and the behavior of single beams is explained in Sec. 9.2.
- Sec. 9.3 extends this description of beam drawing to wide beams and their special properties.
- Sec. 9.4 shows how errors in beam connections (missed interfaces etc.) are handled.
- Sec. 9.5 explains how customized, piecewise defined beam paths, like e.g. in a telescope, can be easily constructed.
- Manual fiber connections are described in Sec. 9.6, manual wire connections, which show the same behavior, are treated in Sec. 9.7.
- Automatic connections are explained in Sec. 9.8.
- Sec. 9.9 introduces the concept of front and back layer for the drawings.

9.1. Accessing components

All macros for connections can take either an ID, a `comprname` identifier (see Sec. 8.1) or PSTricks nodes as arguments, which are denoted with $\langle obj_1 \rangle$, $\langle obj_2 \rangle$, To distinguish between nodes and components, nodes must be either enclosed in parenthesis within the brackets, or enclosed only in parenthesis: `\drawbeam{(node)}{comp}`, or `\drawbeam(node){comp}`.

1. All components can always be accessed via their ID.

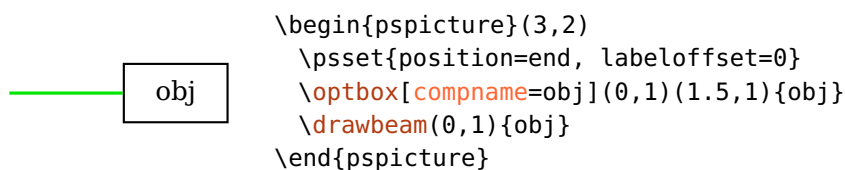


```

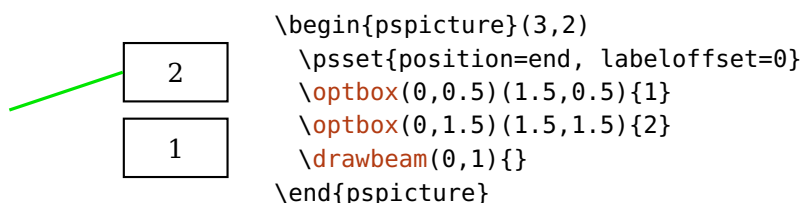
\begin{pspicture}(3,1.5)
  \pnodes(0,1){A}(1.5,1){B}
  \optbox[position=end, labeloffset=0](A)(B){1}
  \drawbeam(A){1}
\end{pspicture}

```

2. A component which has been assigned a name with `compname` can also be accessed via this instead of its ID.

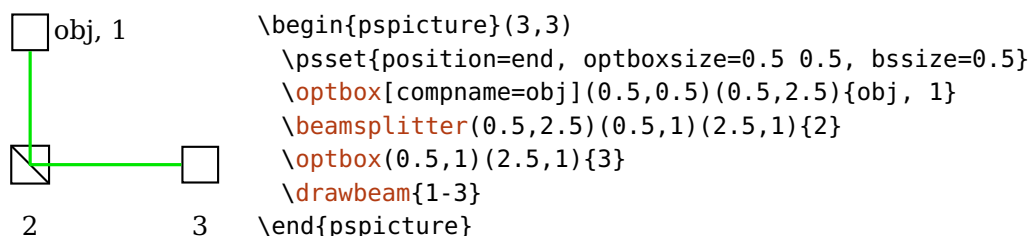


3. If the argument is left empty, use the last component defined.



4. You can also use number ranges, valid range specifications are

- x-y** From x to y, if x is greater than y, the numbers are decremented. If the whole range is not defined, an error is raised.
- x-** From x to the last component.
- y** From the first component to y. If y is outside the valid range, a warning is raised.
- Connect all components.



9.2. Drawing beams

The package provides several ways to connect components and nodes with beams, which can be either single rays or wide beams. Most parameters have the same effect on both single and wide beams so most examples will be shown for single beams only. For options specific to wide beams please see Sec. 9.3.

```
\drawbeam[⟨options⟩]{⟨obj1⟩}{⟨obj2⟩}...
```

```
\drawwidebeam[⟨options⟩]{⟨obj1⟩}{⟨obj2⟩}...
```

These are the two macros for beam connections; they can take a variable number of arguments (minimum of two) which can be either an ID, an ID range, a `compname`, or a PSTricks nodes (see Sec. 9.1). To distinguish between nodes and components, nodes must be either enclosed in parenthesis within the brackets, or enclosed only in parenthesis: `\drawbeam{(node)}{comp}`, or `\drawbeam(node){comp}`.

9.2.1. Raytracing

The beam connection macros support two modes of tracing the beam across the optical components: one is to use refractive indices and Snell's law to determine the beam path. The other mode is to connect the components only, regardless of their optical properties.

You must always keep in mind, that this is a package for *sketching* experimental setups. Therefore, we do not provide a comprehensive raytracing framework, because deviations from the actual physical path are often desired for sketches: beam angles and divergences should be in many cases more extreme than in the real setup in order to highlight certain aspects. You should also consider the refractive index (see Sec. 9.2.2) only as a tool to optimize the visual effects of the beam path, and not stick with its physically correct values.

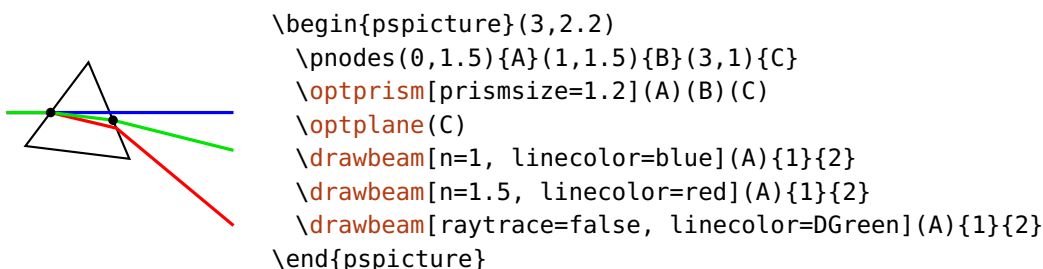
```
raytrace=true, false
```

default: true

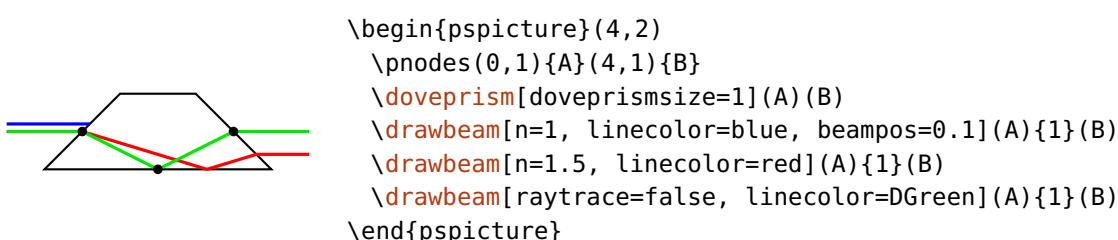
Chooses the mode of tracing the beam path, `true` uses the refractive index ("raytracing"), `false` selects component connection ("connect" mode).

In most cases, the "connect" mode is equivalent to "raytracing" with `n=1`, except for `\optprism` and `\doveprism`, where the interface nodes (Sec. 8.6) are simply connected. See the following two example for the differences between the tracing modes using these two components. The interface nodes are highlighted for clarity.

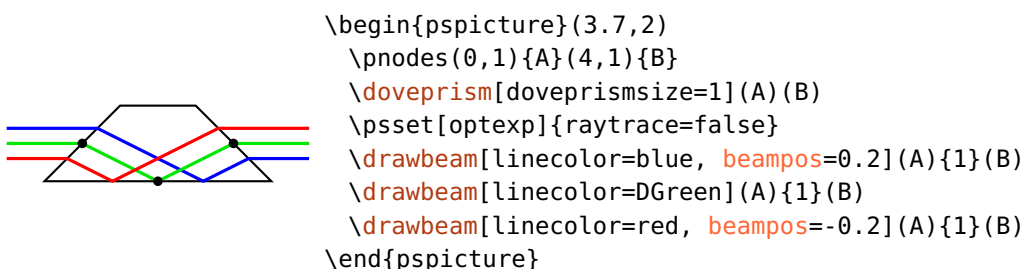
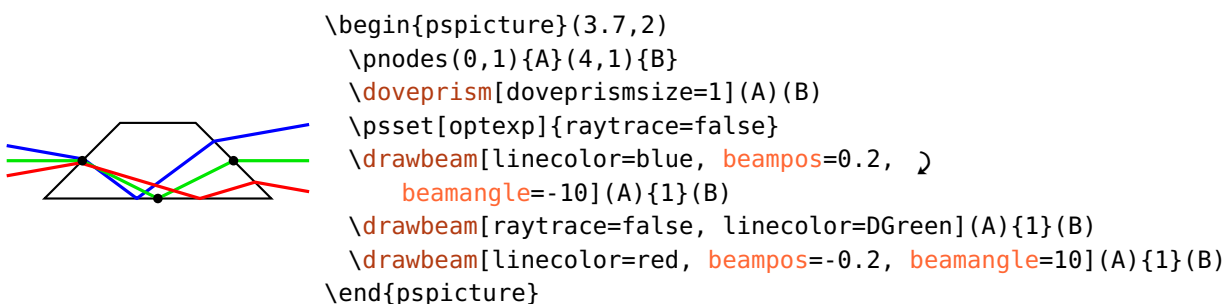
The raytraced beam with `n=1` (blue) passes the component without changing its direction, the refraction for `n=1.5` is calculated (red), and the "connect" beam (green) just passes through the two interface nodes.



For the `\doveprism` the same applies as for the `\optprism`, but the raytraced beam with $n=1$ (blue) misses the second interface because it is not refracted and stops therefore at the first interface (see Sec. 9.4). Again, the green “connect” ray passes through all interface nodes.



The “connect” mode also supports changing the initial conditions (9.2.3).

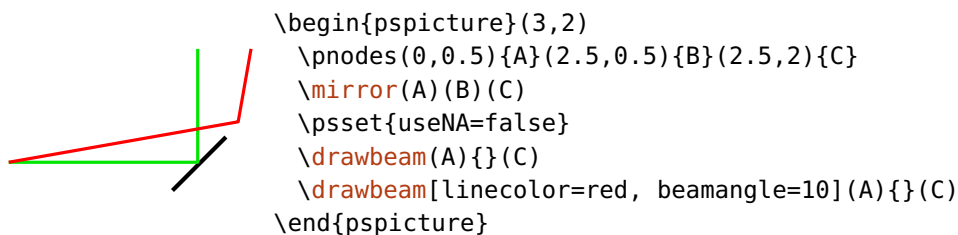


`useNA=true, false`

default: true

The beam path is interrupted if an interface is missed. In some cases, e.g. to estimate the correct refractive index for a ray trace, it can be useful if the

numerical aperture of the components is not considered. The red beam in the following example is drawn, although it hits the mirror outside the drawn mirror surface.



9.2.2. Refractive index

The raytracing depends on the refractive index, which may be set for each component separately. The refractive index is always relative to the background which has a constant value for the whole sketch.

n=*<code>* default: 1.5

Sets the relative refractive index respect to the background. This parameter has two scopes, you can either set the refractive index of each component or of each beam ray:

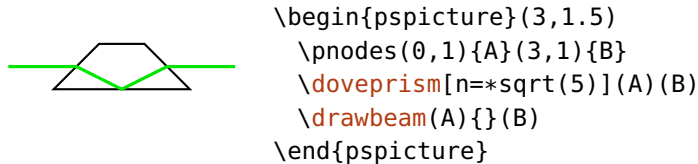
1. Using **n** as global parameter or component argument sets a fixed refractive index for each affected component. Without further actions, all beam paths experience this refractive index.
2. As parameter for `\drawbeam` or `\drawwidebeam` you can overwrite or even change the refractive index of the components for specific beam paths. This can be very useful e.g. to simulate chromatic dispersion. To set this globally for all beams you must add the respective parameter definition to the **Beam** style because global definitions of **n** affect only the components and not the beams.

The *<code>* can in general be any Postscript code which evaluates to a number, e.g. 1.5, or `5 sqrt`. If the *<code>* starts with a `*`, it will be interpreted as an algebraic expression¹, e.g. `*sqrt(5)` is equivalent to `5 sqrt`. Inside the *<code>* you can access the predefined refractive index of the component via `n` which allows you to change the index relative to its original value. Use e.g. `n=n 1.01 mul` to change `n` by one percent.

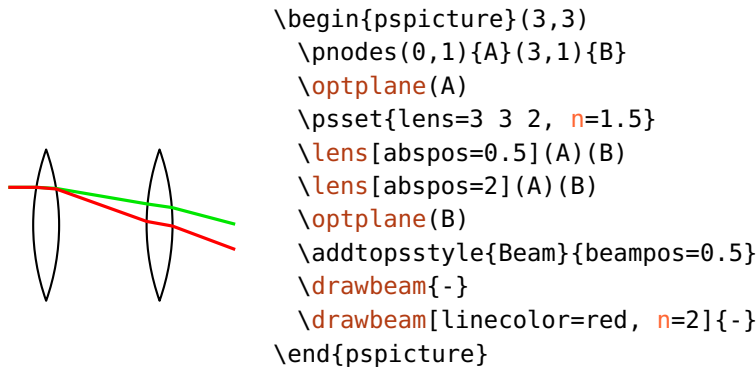
¹<http://mirror.ctan.org/graphics/pstricks/base/doc/pst-news08.pdf>

In the following you find several examples about the different aspects of using `n`.

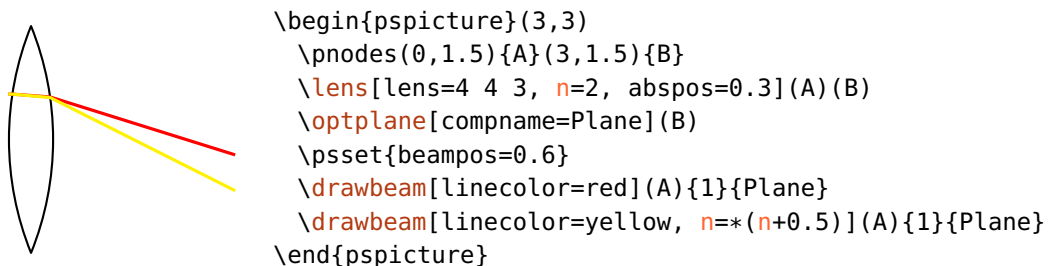
1. Set a fixed refractive index for the component. Interestingly, for the default shape of the `\doveprism` a refractive index of $\sqrt{5}$ gives the ideal output direction which is the same as the input direction.



2. In this example one beam uses the default refractive index of the components (green), the other one overwrites it with a value of 2 (red).

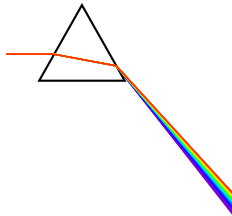


3. Now, the red beam uses the default refractive index of the component, the yellow beam adds 0.5 to the default index.



4. Dynamically changing the refractive index is very useful to emulate chromatic dispersion. Here, we sketch the dispersion of a prism by calculating the refractive index with Sellmaier's equation² for different wavelengths.

²http://en.wikipedia.org/wiki/Sellmeier_equation



```

\begin{pspicture}(3,3)
  \pnodes(-1,0){A}(1,2.2){B}(3,0){C}
  \optplane(0,2.15)
  \optprism[prismalign=center, prismangle=59](A)(B)(C)
  \optplane(C)
  \definecolor[ps]{bl}{rgb}{%
    tx@addDict begin tx@OptexpDict begin
      Wavelength wavelengthToRGB
      Red Green Blue
    end end }%
  \addtopsstyle{Beam}{linecolor=bl, }
    linewidth=0.4\pslinewidth, beamalign=abs}
  \multido{\iLambda=400+4}{60}{%
    \pstVerb{/Wavelength \iLambda\space def }%
    \drawbeam[n=Wavelength Sellmaier]{-}%
  }%
\end{pspicture}

```

5. Using the numerical aperture of a pinhole as spectral filter.



```

\begin{pspicture}(0.3,0)(3.3,5.2)
  \pnodes(0.3,1){A}(2,1){B}(3,2){C}(2.5,5){D}
  \optplane(A)
  \optgrating[reverse](A)(B)(C)
  \pinhole[phwidth=-0.1, innerheight=0.03, abspos=0.5](D)(B)
  \optplate[position=end, plateheight=1.5](B)(D)
  \definecolor[ps]{bl}{rgb}{%
    tx@addDict begin Red Green Blue end}%
  \drawbeam[linecolor=red]{1-2}
  \addtopsstyle{Beam}{linecolor=bl, }
    linewidth=0.4\pslinewidth}
  \multido{\i=0+1}{60}{%
    \pstVerb{%
      \i\space 650 400 sub 59 div mul 400 add
      tx@addDict begin wavelengthToRGB end }%
    \drawbeam[beamangle=\i\space 0.1 mul 3 sub]{2-}%
  }%
\end{pspicture}

```

`refractiveindex`=`<code>`

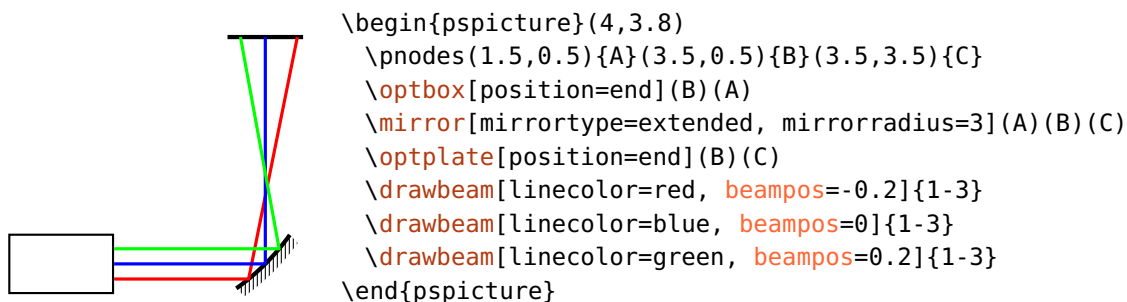
This parameter is deprecated since version 3.0, use `n` instead.

9.2.3. Initial conditions

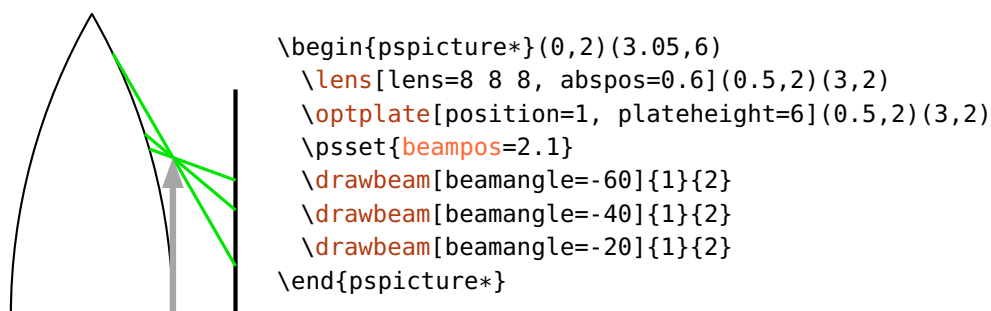
beampos=[$\langle x \rangle$] $\langle y \rangle$

default: 0

This is the start position ($\langle x \rangle$, $\langle y \rangle$) of the beam at the first interface. Both values are of $\langle psnum \rangle$ type. If a single number is given, the x -coordinate is set to 0.



The start position is relative to the respective interface node, as it is illustrated in the following example. The beam start position is fixed, but depending on e.g. the **beamangle**, the actual intersection with the curved interface is at a different position.

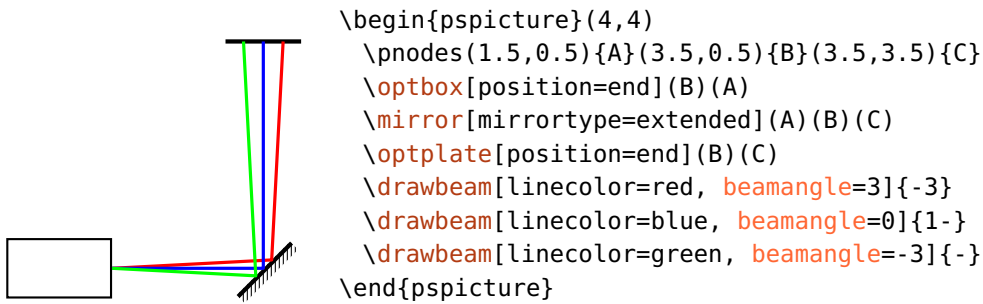


Which interface the beam starts at is determined by the options **beaminsidefirst** and **startinside**.

beamangle= $\langle psnum \rangle$

default: 0

This is the start angle of the beam. Usually, it is relative to the connection between the first two components, use **beamalign** to change this.

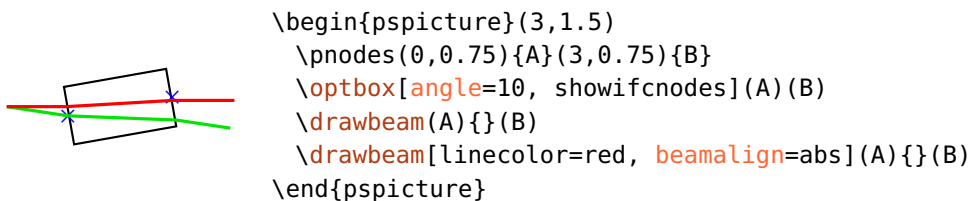


beamalign=rel, relative, abs, absolute, firstcomp

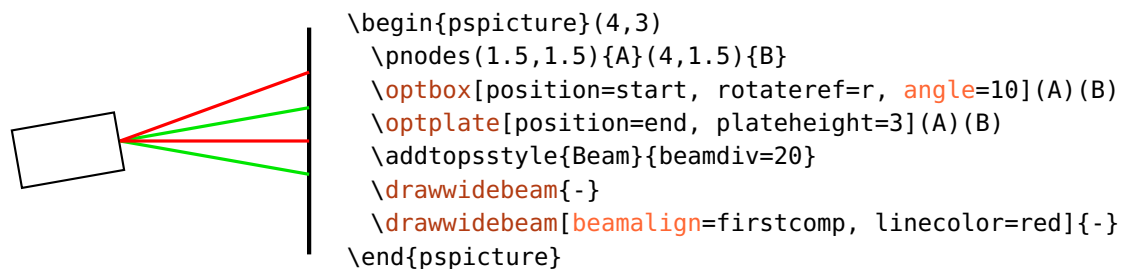
default: relative

Select whether the **beamangle** is relative to the connection between the first two components, or if it is an absolute angle.

A typical situation where an absolute angle is helpful is with rotated or shifted components. By rotating or shifting a component also its interface nodes are rotated (see Sec. 3.3) which are then used to determine the input angle for relative alignment. Accordingly, in the following example, the green beam with the relative alignment is not horizontal.



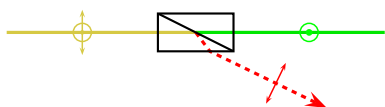
Both components in the following example are align on the horizontal line \overline{AB} . The green beam with **beamalign**=rel is accordingly aligned horizontally. The red beam with **beamalign**=firstcomp is aligned according to the rotation of the first component:



beampathskip= $\langle num \rangle$

default: 0

Skip $\langle num \rangle$ segments at the beginning of the beam path. This option determines only at which segment the beam is drawn, the ray tracing always starts at the first interface. To change this, see option **startinsidecount**.



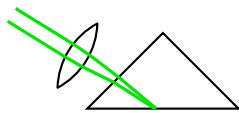
```

\begin{pspicture}(5,2)
\pnodes(0,1){A}(5,1){B}(2,0){C}
\begin{optexp}
\glanthompson(A)(B)
\optplane(B)\optplane[angle=90](C)
\polarization[abspos=1, poltype=misc, linecolor=yellow!80!black](A)(B)
\drawbeam[beaminsidelast, linecolor=yellow!80!black](A){1}
\drawbeam[beampathskip=2](A){1-2}
\drawbeam[beampathskip=2, savebeam, linestyle=dashed, dash=2pt 2pt, 
linecolor=red,arrowscale=1.5, arrows=->](A){1}{3}
\end{optexp}
\nodexn{(\oenameBeam{})-1.5(!\oeBeamVec{})}{D}
\polarization[poltype=parallel, linecolor=red](D)(\oenameBeam{})
\polarization[abspos=4, poltype=perp, linecolor=green, dotscale=1.5](A)(B)
\end{pspicture}

```

beampathcount= $\langle num \rangle$ default: -1

Number of beam segments, which are drawn. Segments which are skipped at the beam start with **beampathskip** do not count.



```

\begin{pspicture}(3,2)
\pnodes(0,2){A}(2,0.8){B}(4,2){C}
\optprism[comname=P, prismtype=reflective, 
prismangle=90](A)(B)(C)
\lens[abspos=1.33](\oenameIfc{2}{P})(A)%
\drawwidebeam[beamwidth=0.2, beaminsidelast, 
beampathcount=4](A){2}{1}
\end{pspicture}

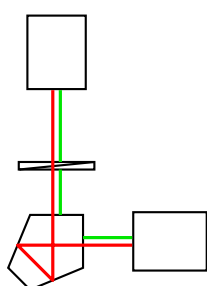
```

9.2.4. Internal beam path

A beam path is composed of rays between components and rays inside the components. Depending on some options these internal beams can be drawn or not. Generally, the first, the last, and all other components are treated separately.

beaminside=true, false default: true

Draw the internal beams in all intermediate components, the first and last components are not affected.

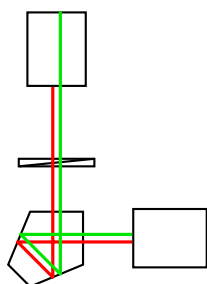


```
\begin{pspicture}(0.3,0.3)(3,4)
\psset{optboxwidth=1}
\optbox[position=end](1,1)(1,3)
\optretplate(1,1)(1,3)
\pentaprism(1,3)(1,1)(2,1)
\optbox[position=end](1,1)(2,1)
\drawbeam[beampos=-0.05, linecolor=red]{-}
\drawbeam[beampos=0.05, beaminside=false]{-}
\end{pspicture}
```

`beaminsidefirst=true, false`

default: false

Draw the internal beams of the first component.

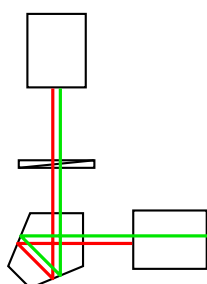


```
\begin{pspicture}(0.3,0.3)(3,4)
\psset{optboxwidth=1}
\optbox[position=end](1,1)(1,3)
\optretplate(1,1)(1,3)
\pentaprism(1,3)(1,1)(2,1)
\optbox[position=end](1,1)(2,1)
\drawbeam[beampos=-0.05, linecolor=red]{-}
\drawbeam[beampos=0.05, beaminsidefirst]{-}
\end{pspicture}
```

`beaminsidelast=true, false`

default: false

Draw the internal beams of the last component.

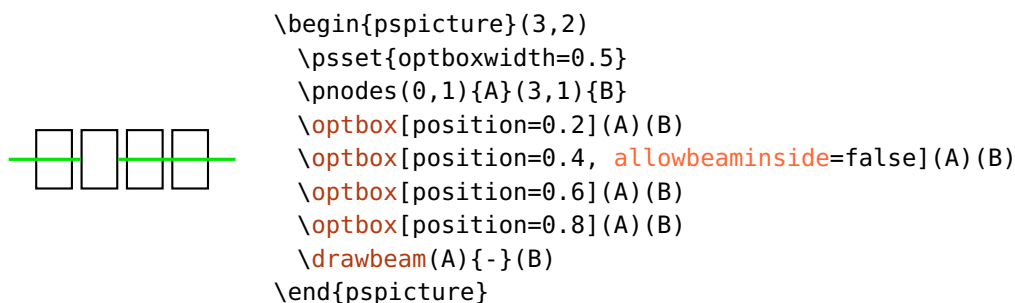


```
\begin{pspicture}(0.3,0.3)(3,4)
\psset{optboxwidth=1}
\optbox[position=end](1,1)(1,3)
\optretplate(1,1)(1,3)
\pentaprism(1,3)(1,1)(2,1)
\optbox[position=end](1,1)(2,1)
\drawbeam[beampos=-0.05, linecolor=red]{-}
\drawbeam[beampos=0.05, beaminsidelast]{-}
\end{pspicture}
```

`allowbeaminside=true, false`

default: true

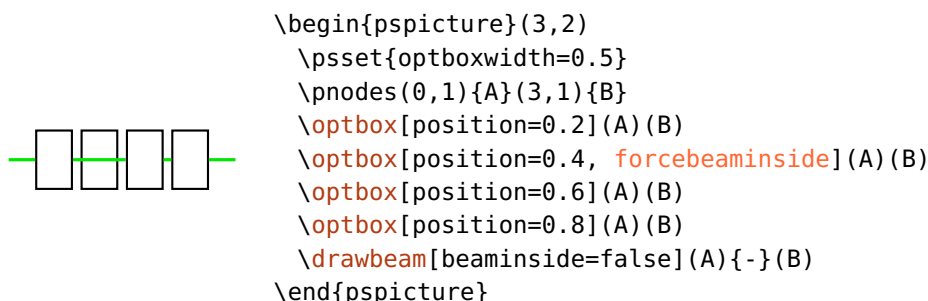
Define whether a beam is drawn inside a component. The option affects components only and overwrites all settings of `beaminside`, `beaminsidefirst`, and `beaminsidelast` if set to false. With this you can forbid any internal beams for components where it does not make any sense (e.g. `\optfilter` and `\optdiode`).



`forcebeaminside=true, false`

default: false

Forces to draw a beam inside a component. The option affects components only and overwrites all settings of `allowbeaminside`, `beaminside`, `beaminsidefirst`, and `beaminsidelast` if set to true. With this you can force an internal beams for single components within a beam path.



`startinsidecount=<num>`

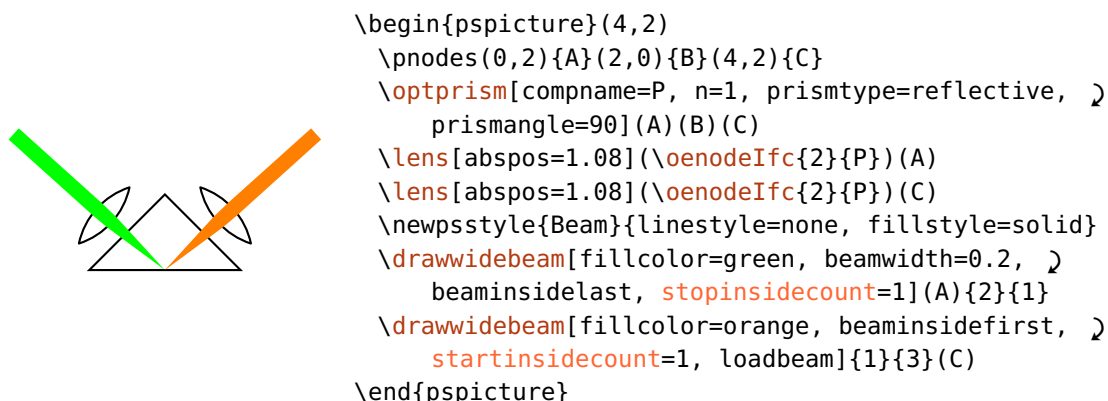
default: -1

Number of segments, which are used inside the first component. This option is in effect only with `beaminsidefirst` or `startinside` set to true.

`stopinsidecount=<num>`

default: -1

Number of segments, which are used inside the last component. This option is in effect only with `beaminsidelast` or `stopinside` set to true.



beammode=refl, trans, reflective, transmissive, auto

default: auto

With this option one can force the beam path through e.g. a `\beamsplitter` to reflective or transmissive independent of the chosen nodes and their position relative to the component.

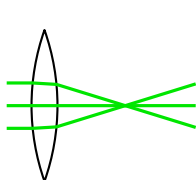
9.2.5. Connecting with nodes

As mentioned earlier in this chapter, beams can connect components also with nodes. For raytracing you need a plane, which can be created automatically or manually in different ways.

beamnodealign=vec, conn, vector, connection

default: connection

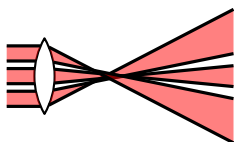
This parameter controls the alignment of the automatically created node interfaces. By default (connection) the interface is aligned perpendicular to the direct connection between node and the previous interface node.



```

\begin{pspicture}(3,2)
  \pnodes(0,1){A}(2.5,1){B}
  \lens[lens=3 3 2, n=2.5, abspos=0.5](A)(B)
  \multido{\r=-0.3+0.3}{3}{%
    \drawbeam[beampos=\r](A){}(B)}
\end{pspicture}

```

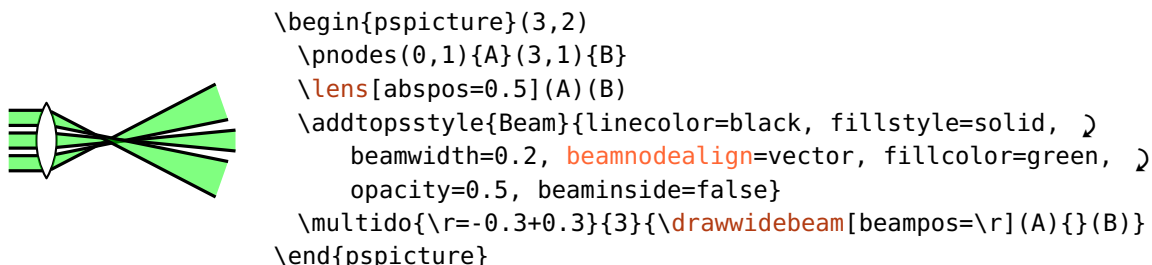


```

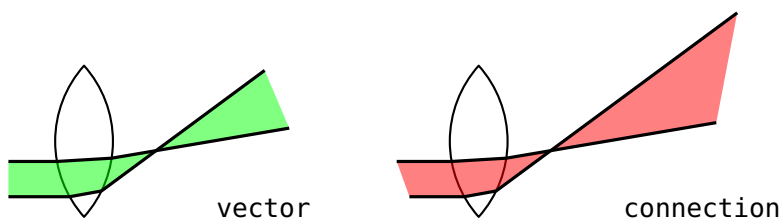
\begin{pspicture}(3,2)
  \pnodes(0,1){A}(3,1){B}
  \lens[abspos=0.5](A)(B)
  \addtopsstyle{Beam}{linecolor=black, fillstyle=solid, 2
    beamwidth=0.2, fillcolor=red, opacity=0.5, beaminside=false}
  \multido{\r=-0.3+0.3}{3}{\drawwidebeam[beampos=\r](A){}(B)}
\end{pspicture}

```


For a value vector, the plane is aligned vertical to the incoming beam vector. For wide beams the mean beam vector is used (see `\oeBeamVecMedian`). Examples where this is very convenient are Ex. 8.1 and Ex. 11.27.



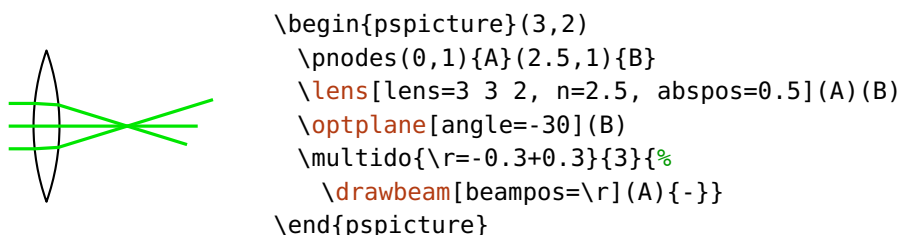
The automatic alignment is very handy in many cases, but does not allow to rotate the node planes to a fixed angle and can give again problems with shifted and rotated components (see Sec. 3.3). In the following example the green beam is with `beamnodealign` set to vector, for the red beam it is set to connection:



For more flexibility, the package provides an “invisible” plane which can be used for this purpose.

`\optplane(<center>)`

The plane is defined by a single node only, the rotation around this node is defined with `angle`. The angle is with respect to the default alignment, which is vertical.



9.2.6. Automatic connections

Beam can also be drawn at definition time of a component, which can be handy, but allows only very elemental beams.

beam=true, false default: false

Can be used as option for every component, is equivalent to the statement

```
\drawbeam[raytrace=false](\oenodeRefA{}){}(\oenodeRefB{})
```

which draws a beam from one reference node via the component to the other reference node.

9.2.7. Beam appearance

Beam $\langle psstyle \rangle$ default: linecolor=green!90!black, linejoin=1

The appearance of beams is controlled with this style. The optional argument of `\draw*beam` can also be used, these options can overwrite settings from the **Beam** style.

addtoBeam= $\langle list \rangle$

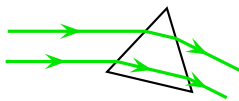
The **Beam** style is extended by locally adding the parameters contained in $\langle list \rangle$. The $\langle list \rangle$ must be surrounded by curly braces.

newBeam= $\langle list \rangle$

Similar to **addtoBeam**, but an existing **Beam** style is overwritten with the new parameter set.

ArrowInsideMinLength= $\langle psnum \rangle$ default: 0.2

`\draw*beam` uses also the option **ArrowInside** from `pstricks-add`. In this case the arrows would be drawn for every beam segment, which is often undesired for the internal, short beam paths. **ArrowInsideMinLength** sets a minimum segment length, below which no internal arrows are drawn. In the following example the value is selected such that only the lower internal beam has an arrow (see also Ex. 11.16).

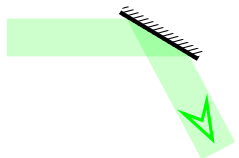


```
\begin{pspicture}(3,2)
\optprism(0,1)(2,1)(3,0.5)
\drawwidebeam[beamwidth=0.4, ArrowInside=->,
\ArrowInsideMinLength=0.5, arrowscale=1.5,
raytrace=false](0,1){}(3,0.5)
\end{pspicture}
```

ArrowInsideMaxLength= $\langle psnum \rangle$ default: -1

Like **ArrowInsideMinLength**, but sets the maximum length of a segment for which the arrows are drawn. If the value is negative, no upper limit is defined.

ArrowInside can be used also without a linestyle.



```
\begin{pspicture}(0,0)(3.5,2)
\pnodes(0,1.5){A}(2,1.5){B}(2.8,0){C}
\mirror[mirrorwidth=1.2, mirrortype=extended](A)(B)(C)
\newsstyle{Beam}{linecolor=green, linestyle=none,
fillstyle=solid, fillcolor=green, opacity=0.2}
\drawwidebeam[beamwidth=0.5](A){}(C)
\drawbeam[ArrowInside=->, arrowscale=2.5, ArrowFill=false,
linewidth=1.5\pslinewidth, ArrowInsidePos=0.9,
ArrowInsideMaxLength=1.8](A){}(C)
\end{pspicture}
```

fade $\langle linestyle \rangle$

Fade from the current linecolor to another color, defined by **fadeto**. The line style currently works only for line, not for curves. The complete path is divided in **fadepoints**−1 segments.



```
\begin{pspicture}(3,1)
\psline[linewidth=0.2, linecolor=red, linestyle=fade,
fadepoints=50](0,0.5)(3,0.5)
\end{pspicture}
```

fadeto=white, black, transparency default: white

The color at the fade end, depends on **fadefuncname** and **fadefunc**.



```
\begin{pspicture}(3,1)
\psset{linewidth=0.2, linecolor=red, linestyle=fade,
fadepoints=50}
\psline[fadeto=white](0,0.9)(3,0.9)
\psline[fadeto=black](0,0.5)(3,0.5)
\psline[fadeto=transparency](0,0.1)(3,0.1)
\end{pspicture}
```

`fadepoints`= $\langle num \rangle$ default: 200


Number of points used for lines with style fade.

`fadefuncname`=gauss, linear, squared, exp, custom default: linear

Some predefined fade functions, which determine the color gradient. If set to custom, the Postscript code in `fadefunc` is used.

`fadefunc`= $\langle PS\ code \rangle$

Define a custom gradient for line style `fade`. The $\langle PS\ code \rangle$ must evaluate a number on the stack in the range $[0, 1]$ (0: path start, 1: path end) to $[0, 1]$ (0: `fadeto` color, 1: current line color). The function values are clipped to $[0, 1]$.



```

\begin{pspicture}(3,1)
  \psset{linewidth=0.2, linecolor=red, linestyle=fade,
    fadepoints=50}
  \psline[fadefuncname=gauss](0,1)(3,1)
  \psline[fadefuncname=exp](0,0.7)(3,0.7)
  \psline[fadefuncname=custom, fadefunc={360 mul cos 1 add 0.5
    mul}](0,0.4)(3,0.4)
  \psline[fadefuncname=custom, fadefunc={neg 1 add 2
    mul}](0,0.1)(3,0.1)
\end{pspicture}

```

For a nice example see Ex. 11.30.

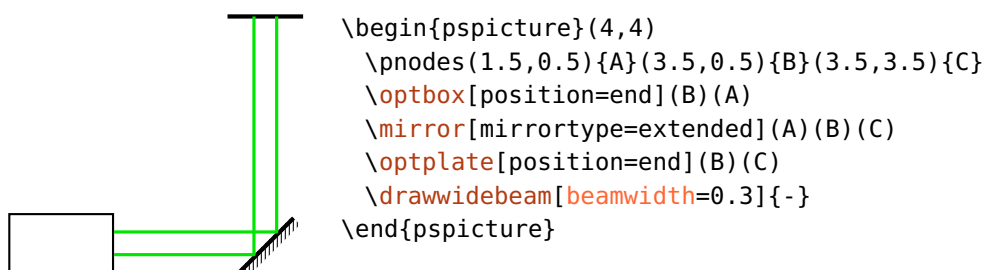
9.3. Drawing wide beams

All information about single beams (see Sec. 9.2) apply to wide beams as well. Here, you find all additional information which apply to wide beams only.

`\drawwidebeam`[$\langle options \rangle$]{ $\langle obj_1 \rangle$ }{ $\langle obj_2 \rangle$ }...

`beamwidth`= $\langle psnum \rangle$ default: 0

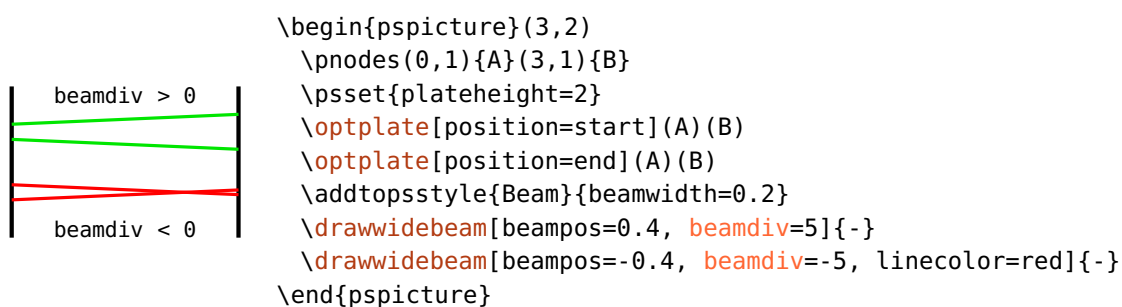
The initial beam width, measured at the initial position (defined with `beampos`) perpendicular to the beam direction.



`beamdiv`= $\langle psnum \rangle$

default: 0

The total beam divergence at the starting position, given in degree. Positive values give an expanding beam, negative values a contracting beam.

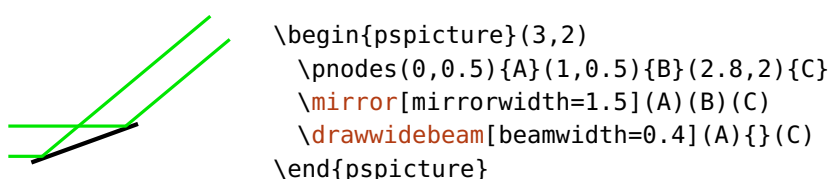


9.3.1. Beam appearance

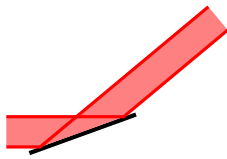
The appearance of wide beams can be controlled like that of single beams with the optional argument or the `Beam` style (see Sec. 9.2.7).

Wide beams have two marginal rays, which are drawn according to the line settings, and may be filled.

1. Only marginal rays.

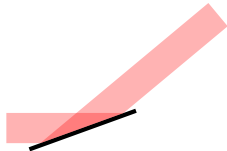


2. Marginal rays, and with fill style.



```
\begin{pspicture}(3,2)
  \pnodes(0,0.5){A}(1,0.5){B}(2.8,2){C}
  \mirror[mirrorwidth=1.5](A)(B)(C)
  \addtopsstyle{Beam}{fillstyle=solid,
    fillcolor=red!50!white, linecolor=red}
  \drawwidebeam[beamwidth=0.4](A){}(C)
\end{pspicture}
```

3. Only with fill style.

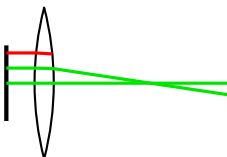


```
\begin{pspicture}(3,2)
  \pnodes(0,0.5){A}(1,0.5){B}(2.8,2){C}
  \mirror[mirrorwidth=1.5](A)(B)(C)
  \addtopsstyle{Beam}{fillstyle=solid, fillcolor=red,
    opacity=0.3, linestyle=none}
  \drawwidebeam[beamwidth=0.4](A){}(C)
\end{pspicture}
```

9.4. Error handling

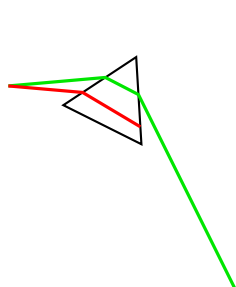
All beam are calculated and drawn directly in Postscript, so that a comprehensive error handling at compilation time is not possible. If a situation arises which is not supported by the drawing algorithm, the beam path is truncated at this point. This happens in the following cases:

1. A beam misses an interface. The red beam in the following example misses the last plane and is truncated at the last valid interface.



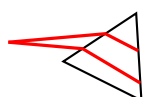
```
\begin{pspicture}(3,2.7)
  \pnodes(0,1){A}(3,1){B}
  \optplate[position=start](A)(B)
  \lens[lens=4 4 2, abspos=0.5, n=2.5](A)(B)
  \optplate[position=end, plateheight=0.5](A)(B)
  \drawbeam{-}
  \drawbeam[beampos=0.2]{-}
  \drawbeam[beampos=0.4, linecolor=red]{-}
\end{pspicture}
```

2. Total internal reflection occurs at an interface which is not set up for this. The red beam in the following example would be reflected, but this is not supported and the beam is truncated at the interface where the total internal reflection occurs.



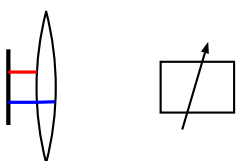
```
\begin{pspicture}(3,2.5)
  \pnodes(0,2){A}(3,0){B}
  \optplane(A)
  \optprism[n=1.8](A)([Xnodesep=1.5]A)(B)
  \optplane(B)
  \drawbeam[beamangle=5]{-}
  \drawbeam[beamangle=-5, linecolor=red]{-}
\end{pspicture}
```

These conditions apply to wide beams (see Sec. 9.3) as well, the beam path is interrupted as soon as one of the marginal rays has an error.



```
\begin{pspicture}(3,2.5)
  \pnodes(0,2){A}(3,0){B}
  \optplane(A)
  \optprism[n=1.8](A)([Xnodesep=1.5]A)(B)
  \optplane(B)
  \drawwidebeam[beamdiv=10, linecolor=red]{-}
\end{pspicture}
```

3. A component is to be connected, which does not support free-ray connections. In this case, the wrong component and all following ones are ignored.



```
\begin{pspicture}(3,2.7)
  \pnodes(0,1){A}(2,1){B}
  \optplate[position=start](A)(B)
  \lens[lens=4 4 2, abspos=0.5, n=2.5](A)(B)
  \fiberdelayline[fdlsize=1 0.7, position=end, 2
    fiber=none](A)(B)
  \drawbeam[beampos=0.2, linecolor=red]{-}
  \drawbeam[beampos=-0.2, linecolor=blue, beaminsidelast]{-}
\end{pspicture}
```

`pswarning=true, false`

default: false

To simplify debugging, you can have a message printed out if any of these situations arises. The message will be printed to standard output, when a Postscript interpreter reads the file, e.g. when you view the file, or you convert it with `ps2pdf` or similar.

9.5. Custom beams

In principal, all kind of beam paths can be realized with an appropriate choice of refractive index and component parameters (e.g. lens curvature). This can

become very cumbersome, so that the package provides the means to simplify this. The end points of one beam path can be used as starting points for the next beam, which allows piecewise definition of the beam divergence, angle, and direction.

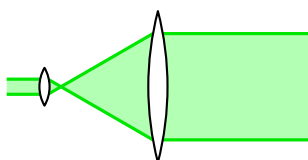
`savebeampoints=true, false, $\langle int \rangle$` default: true

Saves the end points of a `\draw*beam` macro, previous points are overwritten. `\drawbeam` and `\drawwidebeam` are treated separately. If a number $\langle int \rangle > 0$ is given, you can save as many end points as you like, see e.g. Ex. 11.32. But in most cases enabling (1) and disabling (0) this option is sufficient.

`loadbeampoints=true, false, $\langle int \rangle$` default: false

Use the end points of a previous beam as starting points for the next beam. If no number $\langle int \rangle$ is specified, 1 is used.

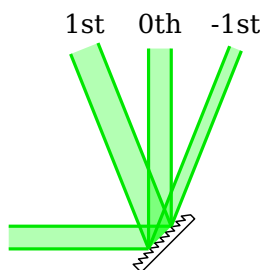
As example we consider a telescope which has no divergence at the input and output. Usually we would need to tweak the refractive index and the lens parameters to get an output beam without divergence. Instead, we draw the three beam parts separately using the end points of the previous beams as new starting points and specifying the divergence for each beam part explicitly.



```
\begin{pspicture}(4,2)
  \pnodes(0,1){A}(4,1){B}
  \begin{optexp}
    \lens[lens=0.5 0.5 0.5, abspos=0.5](A)(B)
    \lens[lens=4 4 2, abspos=2](A)(B)
    \addtopsstyle{Beam}{%
      fillstyle=solid, fillcolor=green, opacity=0.3}
    \psset{loadbeampoints}
    \drawwidebeam[beamwidth=0.2, stopinside](A){1}
    \drawwidebeam[beamdiv=-60]{1}{2}
    \drawwidebeam{2}(B)
  \end{optexp}
\end{pspicture}
```

The next example shows diffraction from a grating in the zeroth and first diffraction order. First we draw the beam to the grating, its end points are used for the beams into both diffraction orders.

Ex. 9.1



```
\begin{pspicture}(3.5,3.5)
  \pnodes(0,0.5){A}(2,0.5){B}(2,3){C}
  \nodexn{(C)-(1,0)}{C'}
  \nodexn{(C)(1,0)}{C''}
  \optgrating(A)(B)(C)
  \addtopsstyle{Beam}{%
    fillstyle=solid, fillcolor=green, opacity=0.3}
  \drawwidebeam[beamwidth=0.3](A){1}
  \psset{savebeampoints=false, loadbeampoints}
  \drawwidebeam{1}(C)
  \drawwidebeam{1}(C')
  \drawwidebeam{1}(C'')
\end{pspicture}
```

`savebeam=true, false, $\langle int \rangle$`

default: true

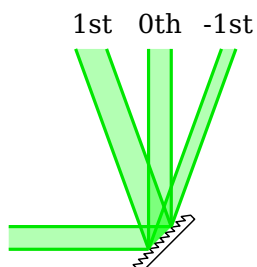
Saves the end conditions, i.e. end points and output vectors, of each `\draw*beam` macro, previous values are overwritten. `\drawbeam` and `\drawwidebeam` are treated separately. If a number $\langle int \rangle > 0$ is given, you can save as many end points as you like, but in most use cases enabling and disabling this option is sufficient.

`loadbeam=true, false, $\langle int \rangle$`

default: false

Use the end condition (points and vectors) of a previous beam as starting conditions for the next beam. The parameter `beamangle` can be used to change the direction relative to the loaded one (see following example).

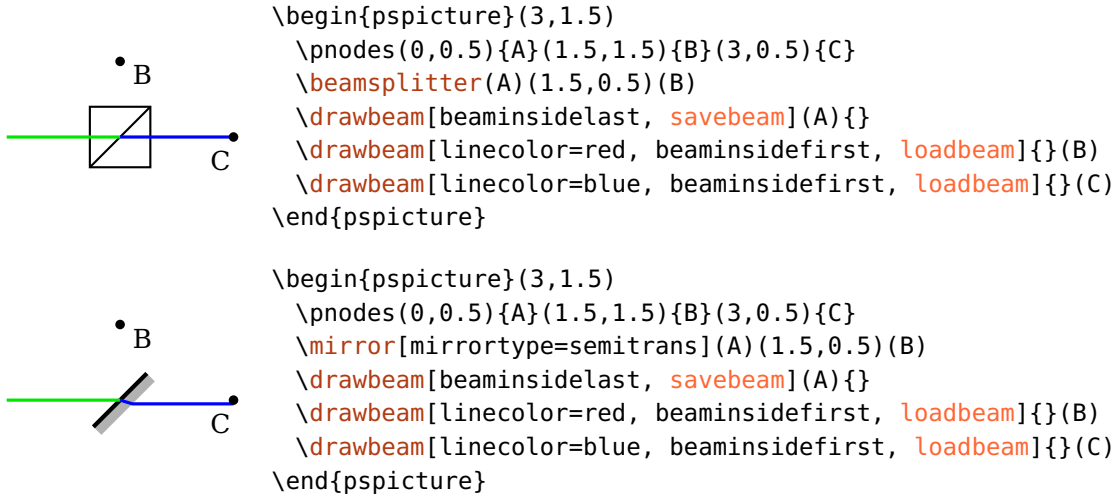
Ex. 9.2



```
\begin{pspicture}(3.5,3.5)
  \pnodes(0,0.5){A}(2,0.5){B}(2,3){C}
  \optgrating(A)(B)(C)
  \addtopsstyle{Beam}{%
    fillstyle=solid, fillcolor=green, opacity=0.3}
  \drawwidebeam[savebeam, beamwidth=0.3](A){1}
  \psset{savebeam=false, loadbeam}
  \drawwidebeam[beamangle=-20]{1}(C)
  \drawwidebeam{1}(C)
  \drawwidebeam[beamangle=20]{1}(C)
\end{pspicture}
```

Please note, that `loadbeam` and `savebeam` do not always work, if the beam path ends inside a semitransparent mirror or a beamsplitter. In these cases only the transmitted ray vector is saved and continuing the beam via the reflective path is not possible.

Accordingly, in the following examples the red beam is not drawn, which should continue the green beam from the component to node $\langle B \rangle$.

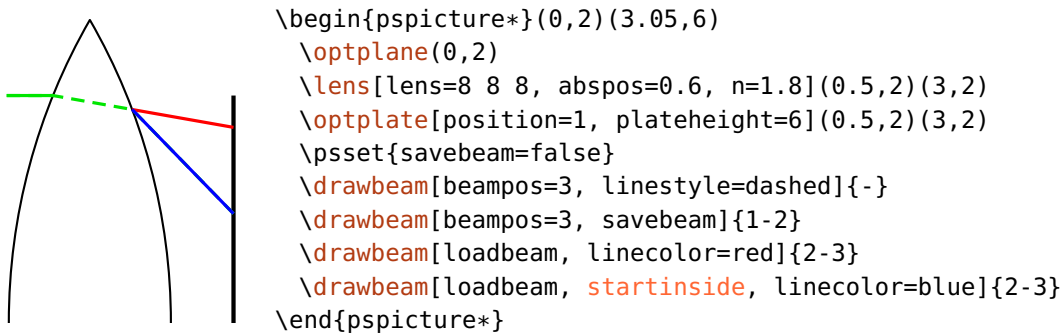


`startinside=true, false`

default: false

The parameter `beaminsidefirst` controls if the internal beams are drawn inside the first component. However, for custom beams it can also be necessary to trace the internal beam of the first component without drawing it, e.g. if the previous beam ended at the first interface.

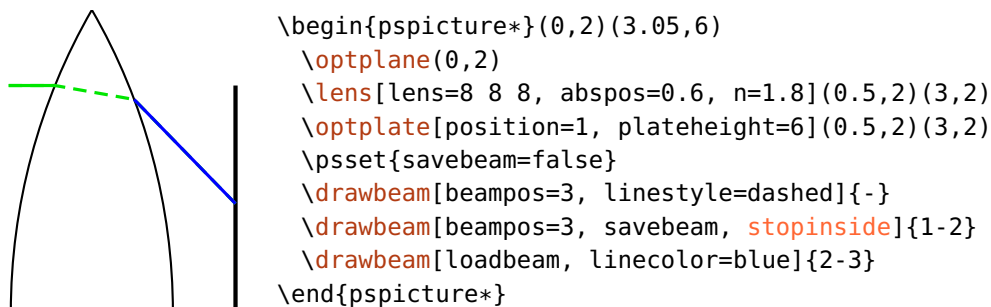
In the following example the red beam is the wrong continuation of the incoming green beam because it assumes that the loaded beam conditions refer to the output interface. But the previous beam stopped at the input interface, so one needs to set `startinside` in order to continue the beam properly (see blue beam). The dashed line helps to understand the correct beam path through the lens.



`stopinside=true, false`

default: false

This option is equivalent to `startinside` but refers to the last component.



9.6. Drawing fibers

All available components can be connected with fiber connections, the fiber components in Sec. 5 are connected automatically.

`\drawfiber`[$\langle options \rangle$]{ $\langle obj_1 \rangle$ }{ $\langle obj_2 \rangle$ }...

This macro draws fiber connections and takes a variable number of arguments (minimum of two) which can be either an ID, an ID range, a `compname`, or a PSTricks node (see Sec. 9.1). To distinguish between nodes and components, nodes must be either enclosed in parenthesis within the brackets, or enclosed only in parenthesis: `\drawfiber{(node)}{comp}`, or `\drawfiber(node){comp}`. If you specify more than two arguments, it draws the connections $1 \rightarrow 2$, $2 \rightarrow 3$, and so on.

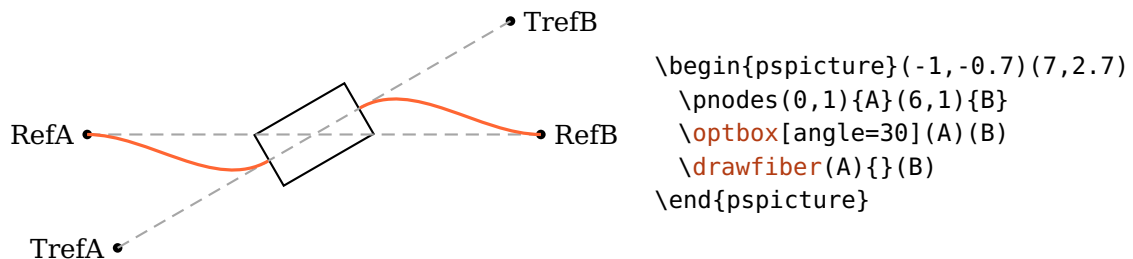
The automatic fiber connections are drawn internally with `\drawfiber` commands from each reference node to the component.

9.6.1. Fiber angles

The fibers are usually drawn as `\ncurve` connections, the angles at both curve ends are automatically determined by the component orientation and depend on whether you connect a component with a component, or a component with a node.

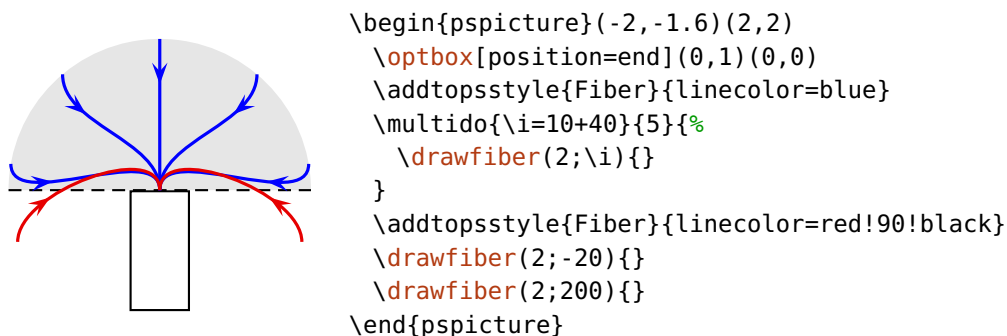
Component with node

A connection of a component with a node is what is used for automatic connections (see Sec. 9.8). Consider the following example for clarification of the curve alignment:



The angle at the component itself agrees with the transformed reference line. The angle at the node corresponds to the angle of the original reference line of the component which it gets connected to. The fiber is always directed to the component node.

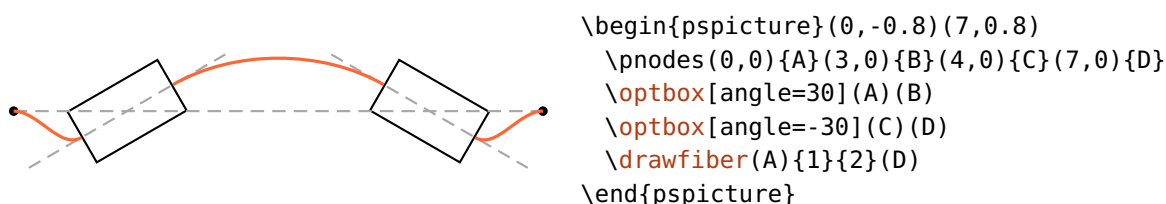
Considering the following example, this means that the start angles of the red fibers are rotated by 180° with respect to the blue fibers. The angles at the component do not change.



Component with component

This variant is very helpful to achieve smoother connections between components which are not aligned on the same reference line but are arranged maybe in a loop or similar, see Ex. 9.3 and Ex. 9.4.

In this case, the curve angles are determined by the orientation of the component which the respective curve end is connected to. The fibers are always directed away from the component center.

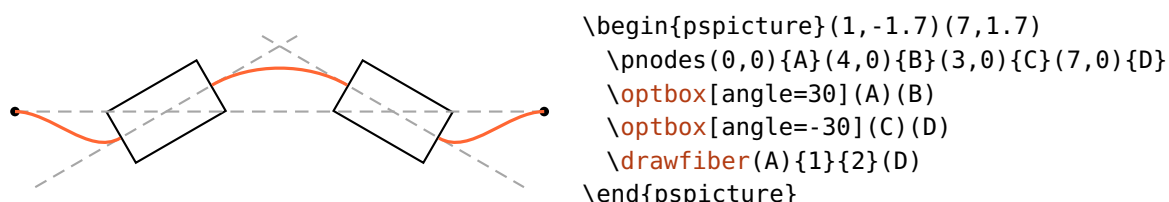


These automatically calculated start and stop angles can be changed in a variety of ways, the available parameters are listed in the following.

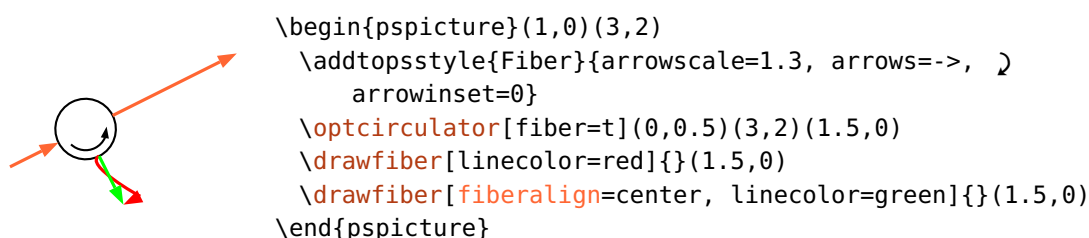
fiberalign=rel, relative, center, abs, absolute default: relative

Set the reference for the fiber angles.

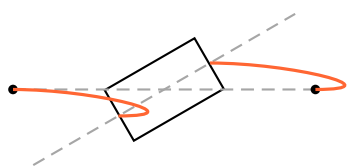
relative The fiber angles are relative to the component. When connecting a node to a component, the angle at the node is that of the component reference line. The angle at the component agrees with that of its transformed reference line.



center The fiber angles both at the node and the component are relative to the connection between the involved interface node and the center node of the component. In most cases this is equivalent to relative, but necessary e.g. for `\optcirculator` and `\elecmmixer`. In the example, the green fiber shows the correct alignment, while the red one is with relative alignment.



absolute The fiber angles are absolute, no automatic values are added.



```
\begin{pspicture}(1,-1)(4.5,1)
  \pnodes(0,0){A}(4,0){B}
  \optbox[angle=30](A)(B)
  \drawfiber[fiberalign=abs,ncurv=1.5](A){}(B)
\end{pspicture}
```

fiberangleA= $\langle num \rangle$

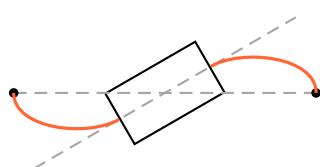
default: 0

An additive value for the starting angle, the total angle depends on **fiberalign**. If this parameter is set to absolute, then **fiberangleA** is the total angle, otherwise its value is added to the calculated angle.

fiberangleB= $\langle num \rangle$

default: 0

Like **fiberangleA**, but for the end angle.



```
\begin{pspicture}(0,-1)(4,1)
  \pnodes(0,0){A}(4,0){B}
  \optbox[angle=30](A)(B)
  \drawfiber[fiberangleA=-90](A){}
  \drawfiber[fiberangleB=-90]{}(B)
\end{pspicture}
```

startnode=auto, N, 1, 2, ...

default: auto

The connections with **\drawfiber** are drawn between the two nodes which are nearest to each other. This fits most of the time, but not always. With this option you can choose the start node explicitly.

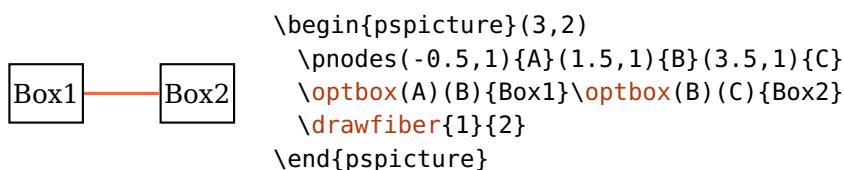
stopnode=auto, N, 1, 2, ...

default: auto

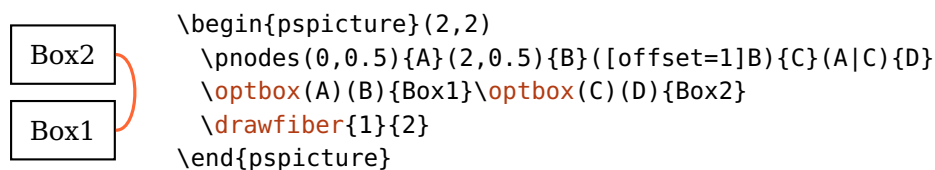
Like **startnode**, but for the stop node.

The following examples show some use cases for **startnode** and **stopnode**.

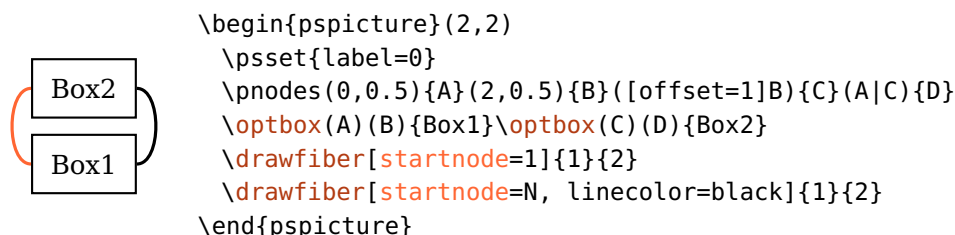
1. This example shows a situation, where choosing the nearest nodes is correct.



2. In this case, the nearest connection is not well defined because two connections have the same length.

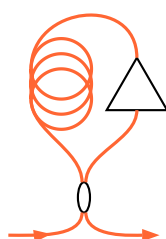


3. To select the desired connection it is often enough to set either `startnode` or `stopnode`. With this you fix one node, the other is the one nearest to it. In rare cases it might be necessary to fix both nodes.



4. Another example where you need to specify the `startnode`.

Ex. 9.3



```

\begin{pspicture}(2.1,3)
  \pnodes(0,0){A}(2,0){B}(0.3,3){C}(1.7,3){D}
  \psset{fiber=none, arrowscale=1.2, arrowinset=0}
  \optcoupler[fiber=l,
    addtoFiberIn1={angleA=0, ArrowInside=->},
    addtoFiberIn2={angleA=180, arrows=<-},
    abspos=0.5, compname=Cpl](A)(B)(C)(D)
  \optfiber[compname=Hnlf, abspos=1](C)(C|A)
  \optamp[abspos=2, compname=Amp](D|B)(D)
  \drawfiber{Cpl}{Hnlf}
  \drawfiber[startnode=1, ncurv=1.2]{Hnlf}{Amp}
  \drawfiber{Cpl}{Amp}
\end{pspicture}

```

9.6.2. Fiber appearance

Fiber `\psstyle`

The appearance of fibers is controlled with this style. The optional argument of `\drawfiber` can be used as well, these options can overwrite settings from the `Fiber` style.

addtoFiber= $\langle list \rangle$

The **Fiber** style is extended by locally adding the parameters contained in $\langle list \rangle$. The $\langle list \rangle$ must be surrounded by curly braces. See for example Ex. 9.4.

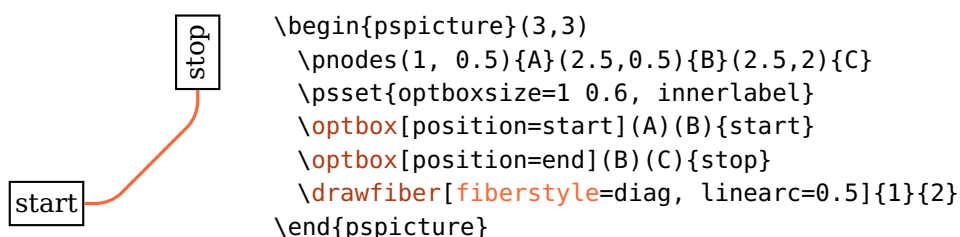
newFiber= $\langle list \rangle$

Similar to **addtoFiber**, but an existing **Fiber** style is overwritten with the new parameter set.

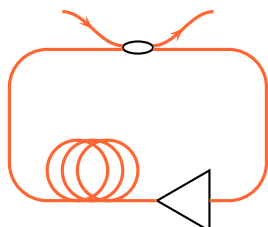
fiberstyle= $\langle string \rangle$

default: curve

The fibers are usually drawn as `\nccurve`, but any type of node connection `\nc $\langle string \rangle$` from `pst-node` can be selected with this parameter, see the `pst-node` documentation³ for possible values. The following examples show possible use cases, Ex. 11.13 contains another typical use case.



Ex. 9.4



```

\begin{pspicture}(4,2.5)
  \pnodes(1,2.5){In}(3,2.5){Out}(0,2){A}(4,2){B}(4,0){C}(0,0){D}
  \optcoupler[fiber=t, addtoFiber={ArrowInside=->}, 2
    coupleralign=b](In)(A)(Out)(B)
  \psset{fiber=none}
  \optamp[position=0.35](C)(D)
  \optfiber[position=0.35](D)(C)
  \drawfiber{2}{3}
  \addtopsstyle{Fiber}{fiberstyle=bar, linearc=0.5, armA=1.5}
  \drawfiber[stopnode=1]{1}{2}
  \drawfiber[stopnode=1]{1}{3}
\end{pspicture}

```

9.7. Drawing wires

All available components can be connected with wire connections, the wire components in Sec. 6 are connected automatically. The `wire*` parameters are equiv-

³<http://mirror.ctan.org/help/Catalogue/entries/pst-node.html>

alent to their fiber* counterparts, see their respective documentation for details.

`\drawwire`[$\langle options \rangle$]{ $\langle obj_1 \rangle$ }{ $\langle obj_2 \rangle$ }...

This command can take a variable number of arguments (minimum of two) which can be either an ID, an ID range, a `compname`, or a PSTricks nodes (see Sec. 9.1). To distinguish between nodes and components, nodes must be either enclosed in parenthesis within the brackets, or enclosed only in parenthesis: `\drawwire{(node)}{comp}`, or `\drawwire(node){comp}`. If you specify more than two arguments, you get connections $1 \rightarrow 2$, $2 \rightarrow 3$, and so on.

The automatic connections are drawn internally with `\drawwire` commands from each reference node to the component.

`wirealign`=rel, relative, center, abs, absolute default: relative

See `fiberalign`.

`wireangleA`= $\langle num \rangle$ default: 0

See `fiberangleA`.

`wireangleB`= $\langle num \rangle$ default: 0

See `fiberangleB`.

`wirestyle`= $\langle string \rangle$ default: angle

See `fiberstyle`, the default value is angle.

`addtoWire`= $\langle list \rangle$

The `Wire` style is extended by locally adding the parameters contained in $\langle list \rangle$. The $\langle list \rangle$ must be surrounded by curly braces.

`newWire`= $\langle list \rangle$

Similar to `addtoWire`, but an existing `Wire` style is overwritten with the new parameter set.

`Wire` $\langle psstyle \rangle$

The appearance of wires is controlled with this style, compare with Sec. 9.6.2. The optional argument of `\drawwire` can be used as well, these options can overwrite settings from the `Wire` style.

9.8. Automatic fiber and wire connections

All fiber components described in Sec. 5 are connected automatically to their reference nodes with `\drawfiber`, their behaviour is described in Sec. 9.6.1. The electrical components (Sec. 6) are accordingly connected with `\drawwire`. The connections can also be controlled in a number of ways, without explicitly using these macros. The style of each connection can be configured independently, see Sec. 9.8.1.

`fiber`=[*+]*none*, *all*, *i*, *o*, *<refpoint>* default: *all*

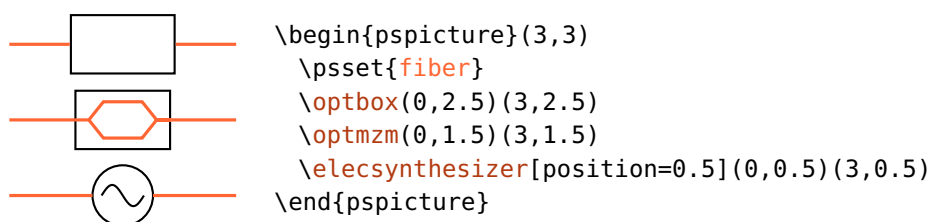
This parameter selects which fiber connections are drawn. The value of this parameter can affect either only the fiber components (if the value is prefixed with *), only free-ray and electrical components (prefixed with +), or all component types otherwise. Typical free-ray components which are often used with fibers as well are `\optbox` and `\optdetector`. Please note, that electrical components cannot be connected automatically with fibers, in these cases you must use `\drawfiber` explicitly.

You can select to connect all nodes, none, or select distinct connections to draw. Value *i* (*l* is equivalent) draws all input fibers, *o* (*r* is equivalent) draws all output fibers. These values are exclusive, that means *io* does not select all fibers, but none.

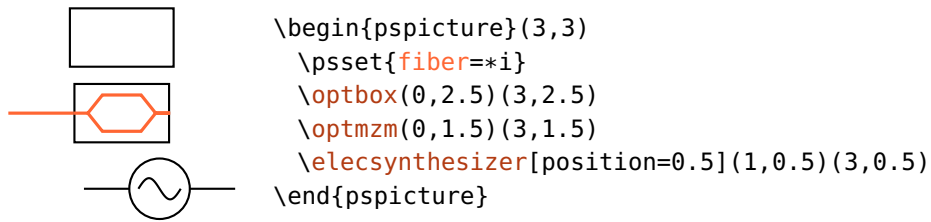
Components which have more than one input or output fiber (`\couplers` and `\optcirculator`) allow also values *t* and *b*, possibly in combination with *i*, *o*, *r*, or *l*.

See the following examples for some variants.

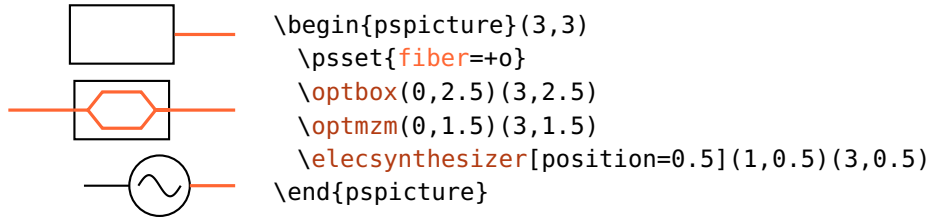
1. Enable all fiber connections all components.



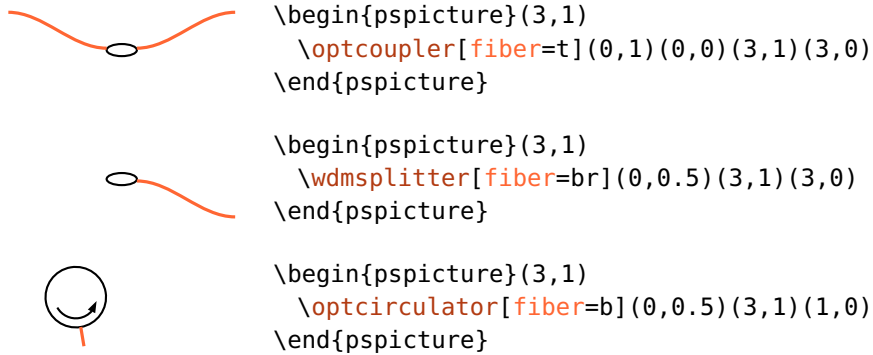
2. Draw only the input fiber of a fiber component (`\optmzm`), the other components are not affected.



3. All connections of the fiber component are drawn, the other components get only the output fiber.



4. The values `t` and `b` affect only components which have more than two fibers, i.e. `\optcoupler` and the couplers (Sec. 5.10).

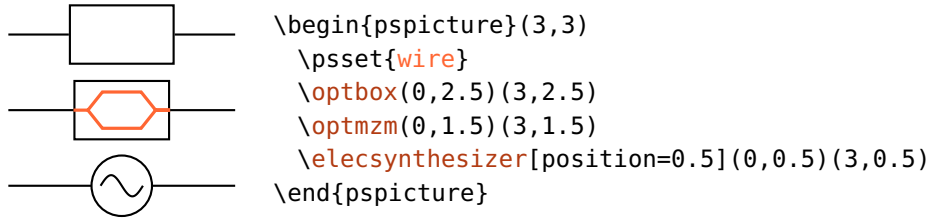


`wire=[*+]none, all, i, o, \langle refpoint \rangle`

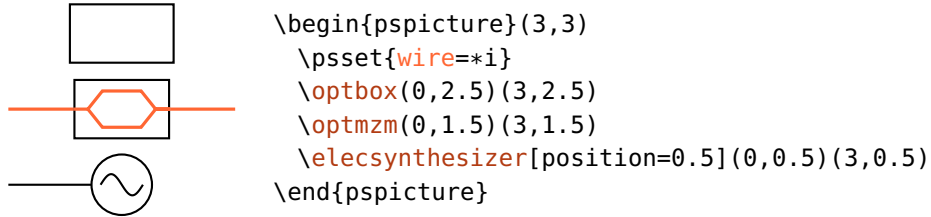
default: all

This parameter selects which wire connections are drawn. The value of this parameter can affect either only the electrical components (if the value is prefixed with *), only free-ray and fiber components (prefixed with +), or all components otherwise.

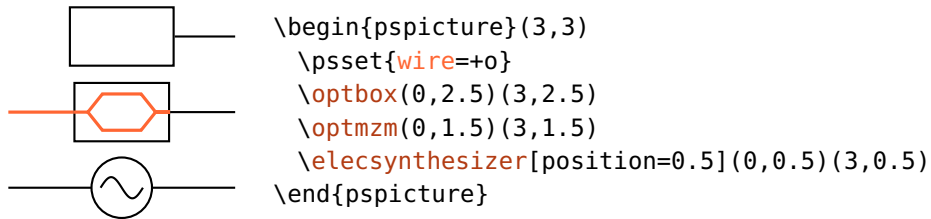
1. Enable all wire connections for all components.



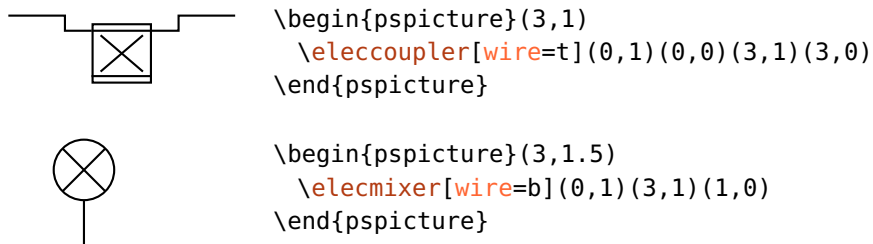
2. Draw only the input wire of an electrical component (`\elecsynthesizer`), the other components are not affected.



3. Draw only the output wire of the free-ray and fiber components, but all connections of the wire component.



4. The values `t` and `b` affect only components which have more than two connections, i.e. `\elecmixer` and `\eleccoupler`.



9.8.1. Appearance of automatic connections

The styles of all automatic fiber connections can be configured independently, they are all derived from the basic `Fiber` style which is described in Sec. 9.6.2.

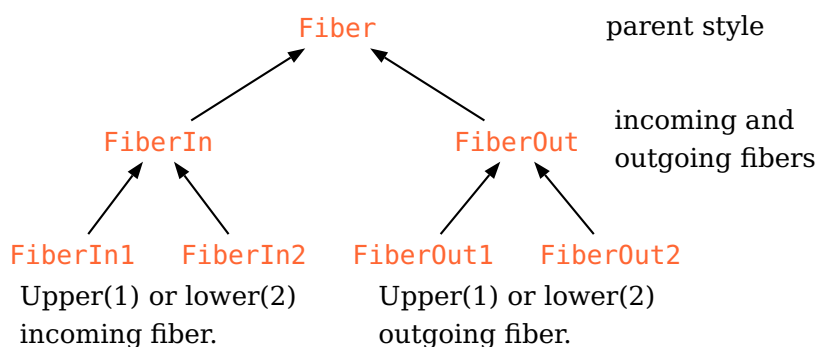
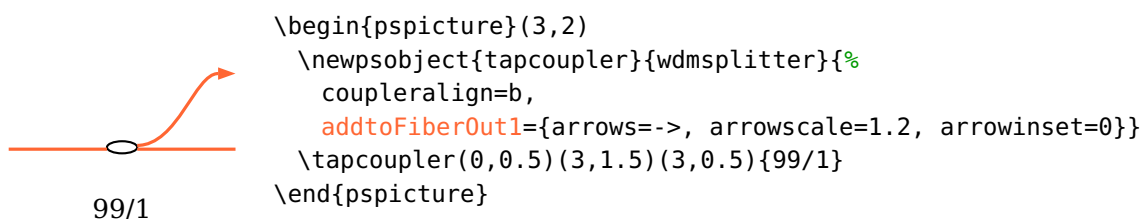


Figure 9.1.: Inheritance diagram for the psstyles used for the automatic fiber connections. These styles should be changed with `\addtopstyle` or the respective `addto<style>` options to preserve the inheritance. An identical inheritance tree applies to the respective `Wire...` styles.

FiberIn Style **FiberIn** applies to all input fibers, it inherits from **Fiber**. If the component **FiberIn1** has more than one input fiber they use the styles **FiberIn1** and **FiberIn2** which **FiberIn2** inherit from **FiberIn**. All output fibers use the style **FiberOut**, in case of two **FiberOut** output fibers they use the styles **FiberOut1** and **FiberOut2** which inherit from **FiberOut1** **FiberOut**. See Fig. 9.1 for an inheritance diagram of the styles. You must **FiberOut2** have in mind, that this inheritance is effective only if you use `\addtopstyle` to change the styles (see also below).

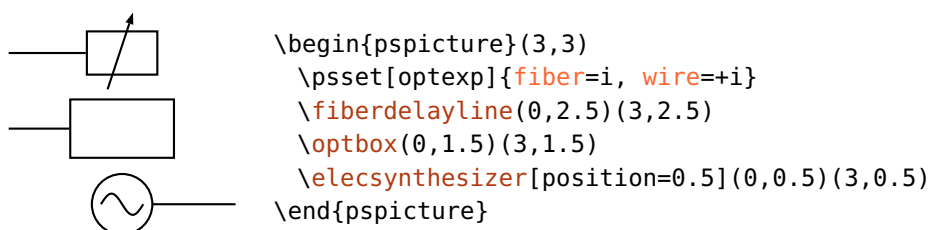
WireIn Style **WireIn** applies to all input wires, it inherits from **Wire**. If the component **WireIn1** has more than one input wire they use the styles **WireIn1** and **WireIn2** which **WireIn2** inherit from **WireIn**. All output wires use the style **WireOut**, in case of two output **WireOut** wires they use the styles **WireOut1** and **WireOut2** which inherit from **WireOut**. **WireOut1** See Fig. 9.1 for an inheritance diagram of the styles. You must have in mind, **WireOut2** that this inheritance is effective only if you use `\addtopstyle` to change the styles (see also below).

new<style> For every style two appropriate keys **new<style>** and **addto<style>** are provided **addto<style>** which can be used to change the styles for single objects. This can be used to define own components (Sec. 10.1) with respectively changed connections or to avoid explicit grouping.



9.8.2. Known limitations

The combination of `fiber` and `wire` does not allow any arbitrary combination of all parameter values in the current implementation (version v5.1). Most combinations do work, but the following examples shows a variant, which does not work correctly (the `\elecsynthesizer` should have an input fiber):



In such special cases one must resort to manual connections.

9.9. Layers



The necessary order of constructing a drawing is to define the components first before you can connect them. This implies that all connections are drawn on top of the components, which might be unwanted.

To circumvent this, we provide the `optexp` environment, inside which all connections are drawn behind the components.

```

\begin{optexp}
\end{optexp}

```

	<code>\begin{pspicture}(3,4)</code>
no layers	<code>\psset{beam, fillstyle=solid, fillcolor=black}</code>
	<code>\optbox(0,3.5)(3,3.5){no layers}</code>
	<code>\begin{optexp}</code>
	<code>\optbox(0,1.5)(3,1.5){with layers}</code>
with layers	<code>\end{optexp}</code>
	<code>\end{pspicture}</code>

Beware, that this is achieved by executing twice all code included in the environment. This works fine for all `pst-optexp` commands which take care of not drawing stuff twice but only in a changed order. All other text and graphics are drawn twice which leads to ugly results because of altered anti-aliasing in the viewer.

To avoid this, you can either move all other code outside the environment, or use the following two commands to define code parts which are executed only once.


`\backlayer{<code>}`

Execute `<code>` only once in the first pass.

`\frontlayer{<code>}`

Execute `<code>` only once in the second pass.

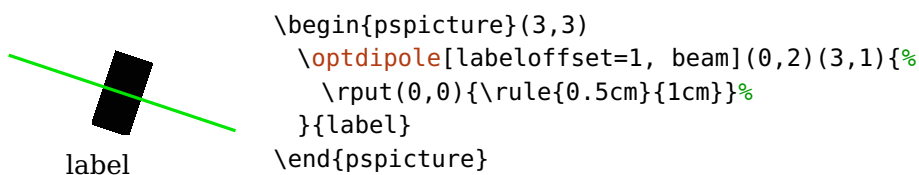
In this example, the text is put on top of the frame although the text is defined first:

	<code>\begin{pspicture}(3,1)</code>
	<code>\begin{optexp}</code>
	<code>\frontlayer{\rput(1.5,0.5){front}}</code>
	<code>\backlayer{\psframe*[linecolor=D0range!50](1,0)(2,1)}</code>
	<code>\end{optexp}</code>
	<code>\end{pspicture}</code>

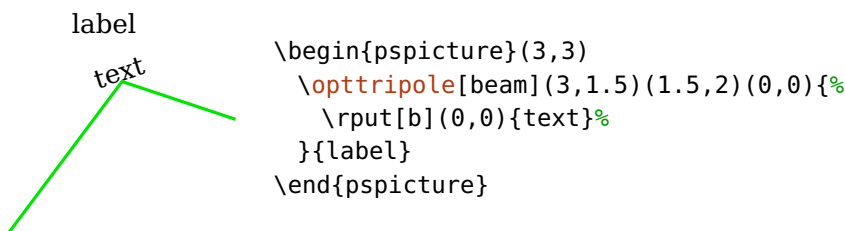
10. Custom components

The `pst-optexp` package provides two commands which can use anything as optical components. This includes e.g. external images or your own drawings.

`\optdipole`[$\langle options \rangle$]($\langle in \rangle$)($\langle out \rangle$){ $\langle comp \rangle$ }{ $\langle label \rangle$ }



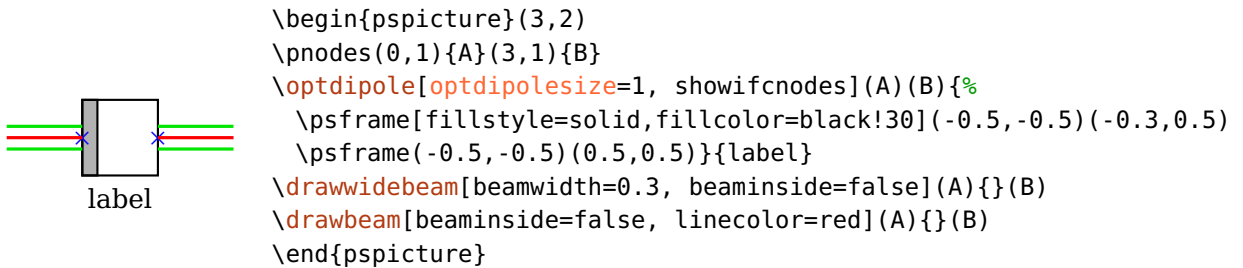
`\opttripole`[$\langle options \rangle$]($\langle in \rangle$)($\langle center \rangle$)($\langle out \rangle$){ $\langle comp \rangle$ }{ $\langle label \rangle$ }



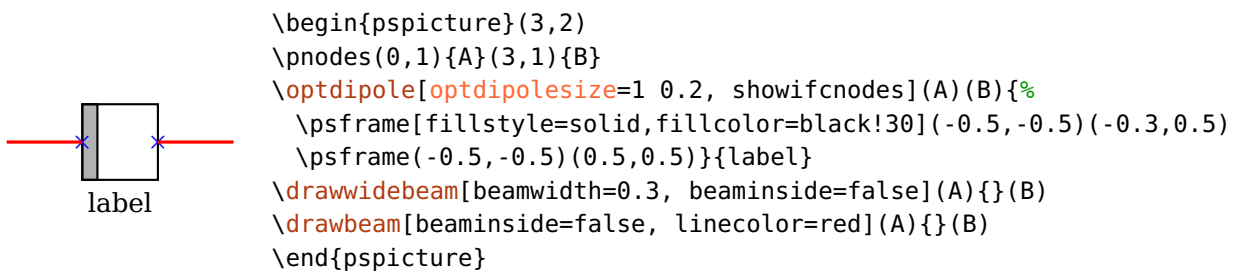
These components can also be connected with beams and fibers, but do not provide full access to all options. `\opttripole` defines a single reflective interface, whereas `\optdipole` can have two transmittive interfaces:

`optdipolesize`= $\langle width \rangle$ [$\langle height \rangle$] default: 0 0

With this option you can define two transmittive interfaces for `\optdipole`, which are placed at $(\pm 0.5\langle width \rangle, 0)$.



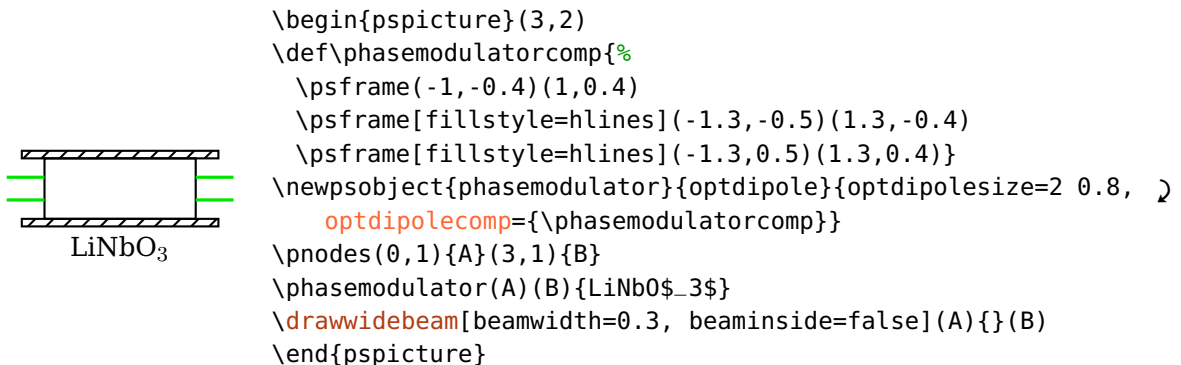
If also a $\langle height \rangle$ greater than zero is set, this is used as size of the numerical aperture for ray tracing (see `useNA`). In the following example the height is set to 0.2 which is smaller than the wide beam which is not drawn, in contrast to the previous example.



Further examples are Ex. 11.11 and Ex. 11.24.

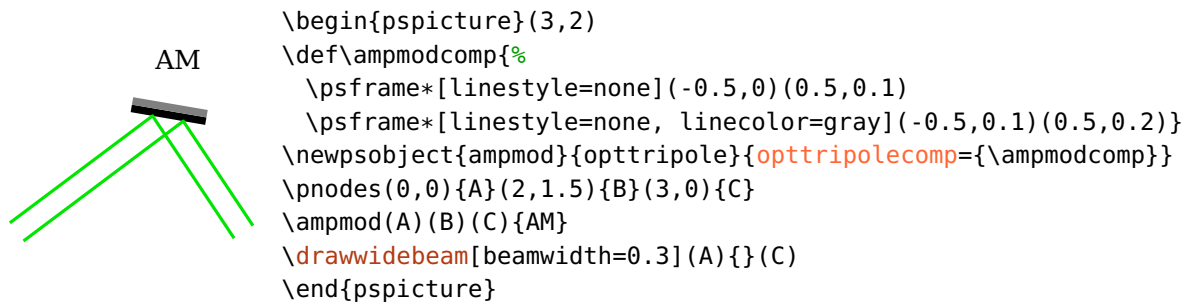
`optdipolecomp`= $\langle macros \rangle$

This option defines the drawing of a component, and can be used to define a new component based in `\optdipole` with `\newpsobject`.



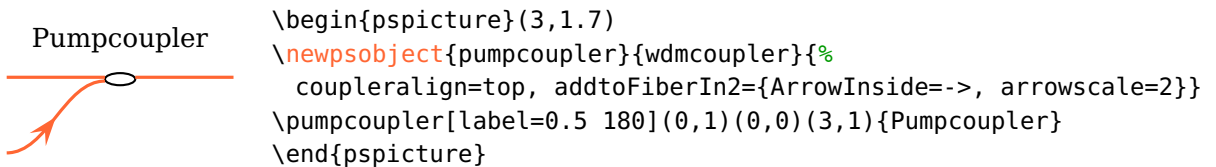
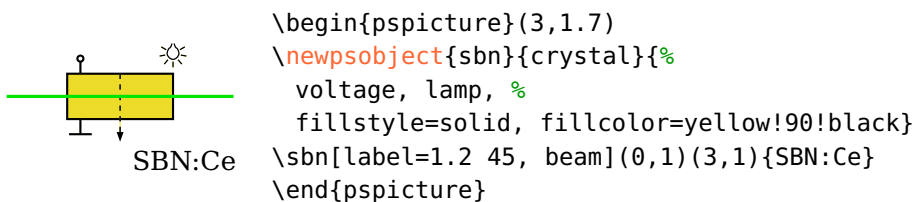
`opttripolecomp`= $\langle macros \rangle$

This option defines the drawing of a component, and can be used to define a new component based in `\opttripole` with `\newpsobject`.

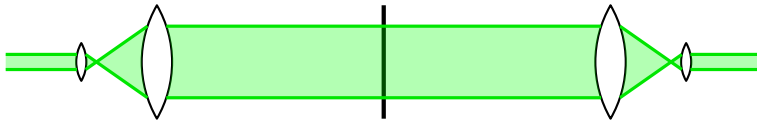


10.1. Customized versions of existing components

You can define a customized version of an existing component using the `\newpsobject` macro. With this you can define a new component using pre-defined objects with a set of options. These options serve only as default values and can be overridden when calling the macro.



Ex. 10.1



```

\begin{pspicture}(10,2)
  \newpsobject{MOLensIn}{lens}{lens=0.5 0.5 0.5}
  \newpsobject{MOLensOut}{lens}{lens=1.5 1.5 1.5}
  \pnodes(0,1){A}(10,1){B}
  \MOLensIn[abspos=1](A)(B)\MOLensOut[abspos=2](A)(B)
  \optplate[plateheight=1.5](A)(B)
  \MOLensOut[abspos=8](A)(B)\MOLensIn[abspos=9](A)(B)
  \addtopsstyle{Beam}{n=1, fillstyle=solid, fillcolor=green, opacity=0.3}
  \psset{loadbeampoints}
  \drawwidebeam[beamwidth=0.2, stopinside](A){1}
  \drawwidebeam[beamdiv=-70]{1}{2}
  \drawwidebeam[stopinside]{2-4}
  \drawwidebeam[beamdiv=-70]{4}{5}
  \drawwidebeam{5}(B)
\end{pspicture}

```

10.2. Defining new objects

pst-optexp provides some high-level macros to allow very convenient definition of your own custom components.

Note, that this part of the documentation is not complete. It roughly describes the steps to define own components since version 3.0. Some parts may be subject to change which are not backward compatible.

```
\newOptexpDipole[⟨fixopt⟩]{⟨name⟩}{⟨dftopt⟩}
```

```
\newOptexpTripole[⟨fixopt⟩]{⟨name⟩}{⟨dftopt⟩}
```

```
\newOptexpFiberDipole[⟨fixopt⟩]{⟨name⟩}{⟨dftopt⟩}
```

```
\newOptexpElecDipole[⟨fixopt⟩]{⟨name⟩}{⟨dftopt⟩}
```

These macros generate all organizing code for the component `\⟨name⟩` which handle the positioning, label placement, rotation and shifting, and the layering. In the simplest case you only have to define the actual drawing of the component outline.

With the argument $\langle fixopt \rangle$ you can specify argument settings which will not be allowed to be overwritten, with $\langle dftopt \rangle$ you can specify a set of default arguments which may be overwritten with the components optional argument. Both arguments $\langle fixopt \rangle$ and $\langle dftopt \rangle$ are optional and may be omitted.

The macros `\newOptexpFiberDipole` and `\newOptexpElecDipole` only differ from the generic `\newOptexpDipole` in the connection settings, like the components in Sec. 5 and Sec. 6, respectively.

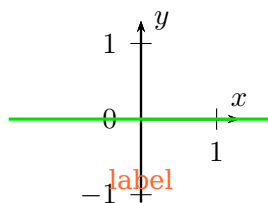
The actual process of creating a new component is split up in different parts, depending on the component complexity and user requirements:

1. The component drawing (see Sec. 10.2.1), this is the only required part.
2. Support for raytracing (optional)
3. Optional support for `rotateref`, `extnode` and `position=start|end` (optional).

The following section show the development of a simple example component `\testcomp`.

10.2.1. The component drawing

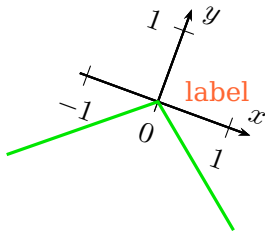
You must define a macro which is called e.g. `\...@comp`, which contains the component drawings. See the following two examples for dipole and tripole which show the orientation and position of the component coordinate system.



```

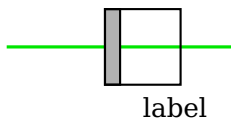
\begin{pspicture}(3.5,2.6)
\newOptexpDipole{mydipole}
\makeatletter
\def\mydipole@comp{%
  \psaxes[arrows=->](0,0)(0,-1.1)(1.3,1.3)[$x$,90][$y$,0]
}%
\makeatother
\mydipole(0,1)(3.5,1){\color{spot}label}
\drawbeam(0,1){}(3.5,1)
\end{pspicture}

```



```
\begin{pspicture}(3.5,3.7)
\newOptexpTripole{mytripole}{labelangle=-60}
\makeatletter
\def\mytripole@comp{%
\psaxes[arrows=->](0,0)(-1.1,0)(1.3,1.3)[$x$,90][$y$,0]
}%
\makeatother
\mytripole(0,1)(2,1.7)(3,0){\color{spot}label}
\drawbeam(0,1){}(3,0)
\end{pspicture}
```

Our example component and its drawing are defined as follows:



```
\begin{pspicture}(3, 2)
\newOptexpDipole{testcomp}
\makeatletter
\def\testcomp@comp{%
\psframe[fillstyle=solid, 2
fillcolor=black!30](-0.5,-0.5)(-0.3,0.5)
\psframe(-0.5,-0.5)(0.5,0.5)
}%
\makeatother
\begin{optexp}
\testcomp[position=0.6, labelalign=l](0,1)(3,1){label}
\drawbeam[beaminside=false](0,1){}(3,1)
\end{optexp}
\end{pspicture}
```

The positioning of the component and the label, and the layering is working already.

10.2.2. Support for raytracing

To add support for ray-tracing, you must define the interface with their properties. The definitions are nested in a macro `\<name>@nodes` in order to be accessible in the different layers.

The definitions are done with the macro `\newOptexpComp`

```
\def\testcomp@nodes{%
\newOptexpComp{%
% definition of the first interface (IF)
{-0.5 0} % x- and y-coordinate of the IF center
```

```

{0 1} % direction vector of the IF (x and y)
-0.5 0.5 % limits use for the NA as factor for IF vector.
trans % behavior of the IF, can be 'trans', 'refl' or 'abs'
{PlainIfc} % type of IF
% the same for the right IF, only the center is different
{0.5 0} {0 1} -0.5 0.5 trans {PlainIfc}
\POE@key@n % refractive index, could be any number
}%
}%

```

So, in condensed form, our example component looks like this, and the beam is not drawn inside the component any more:



```

\begin{pspicture}(3, 2)
\newOptexpDipole{testcomp}
\makeatletter
\def\testcomp@comp{%
  \psframe[fillstyle=solid, fillcolor=black!30](-0.5,-0.5)(-0.3,0.5)
  \psframe(-0.5,-0.5)(0.5,0.5)}%
\def\testcomp@nodes{%
  \newOptexpComp{%
    {-0.5 0} {0 1} -0.5 0.5 trans {PlainIfc}
    {0.5 0} {0 1} -0.5 0.5 trans {PlainIfc} \POE@key@n }}%
\makeatother
\begin{optexp}
\testcomp[position=0.6, labelalign=l](0,1)(3,1){label}
\drawbeam[beaminside=false](0,1){}(3,1)
\end{optexp}
\end{pspicture}

```

10.2.3. Support for external nodes

To enable, as last step, support for external nodes with `extnode`, positioning at the start or end with `position=start|end`, and setting the rotation reference node with `rotateref`, we need to provide the external dimensions of the component. For this we need to define a new macro `\<name>@ref`, which defines at Postscript level the values `@@x` (half width), `@@y` (half height), and optionally `@@x0` and `@@y0` (offset of the center).

```

\def\testcomp@ref{%
  \POE@setref{ /@@x 0.5 def /@@y 0.5 def }%
}%

```

This completes the code for our example component `\testcomp`, which now looks like this:

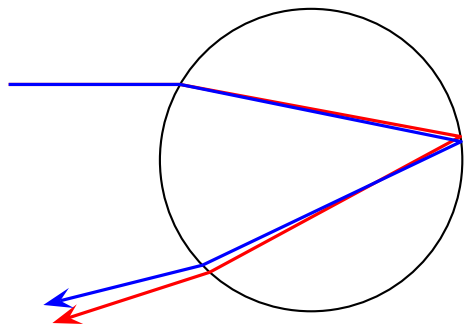


```

\begin{pspicture}(3, 2)
\newOptexpDipole{testcomp}
\makeatletter
\def\testcomp@comp{%
  \psframe[fillstyle=solid, 2]
    {fillcolor=black!30}(-0.5,-0.5)(-0.3,0.5)
  \psframe(-0.5,-0.5)(0.5,0.5)}%
\def\testcomp@nodes{%
  \newOptexpComp{%
    {-0.5 0} {0 1} -0.5 0.5 trans {PlainIfc}
    {0.5 0} {0 1} -0.5 0.5 trans {PlainIfc} \POE@key@n }}%
\def\testcomp@ref{%
  \POE@setref{ /@@x 0.5 def /@@y 0.5 def }}%
\makeatother
\begin{optexp}
\begin{optexp}
\testcomp[position=0.6, labelalign=l, extnode=tl](0,1)(3,1){label}
\drawbeam[beaminside=false](0,1){}(3,1)
\psdot(\oencodeExt{})
\end{optexp}
\end{pspicture}

```

10.2.4. Examples of custom components



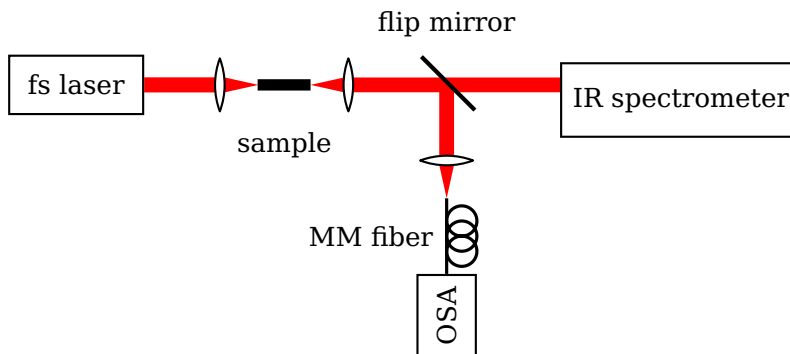
```

\begin{pspicture}(6,4)
\newOptexpTripole{raindrop}
\makeatletter
\def\raindrop@nodes{%
  \newOptexpComp{%
    {0 -2} {0 2} -2 2 trans {CurvedIfc}
    {0 2} {0 -2} -2 2 refl {CurvedIfc}
    {0 -2} {0 2} -2 2 trans {CurvedIfc}
    \POE@key@n }}%
\def\raindrop@comp{\pscircle(0,0){2}}%
\raindrop(0,4)(4,2)(0,0)
\psset{beamalign=abs, arrows=->, arrowscale=2}
\drawbeam[linecolor=red](0,3){}(0,1)
\drawbeam[n=*n*1.05, linecolor=blue](0,3){}(0,1)
\end{pspicture}

```

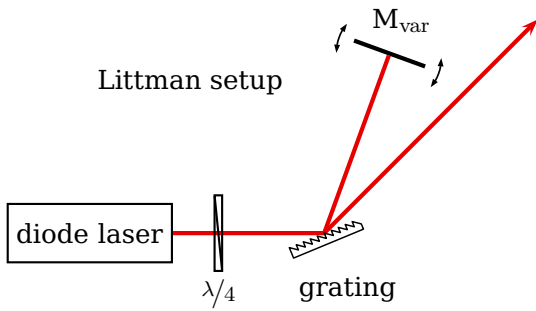
11. Examples

Ex. 11.1



```
\begin{pspicture}(0.1,0.4)(10,4.5)
  \psset{optexp}{lens=1 1 0.7, loadbeampoints}
  \pnodes(2,4){L}([Xnodesep=4]L){M}([Xnodesep=1.5]M){IRSpec}([offset=-2.5]M){OSA}
  \begin{optexp}
    \optbox[position=start, innerlabel, optboxwidth=1.8](L)(M){fs laser}
    \lens[abspos=1](L)(M)
    \optbox[abspos=1.85, optboxsize=0.7 0.15, fillstyle=solid, 
      fillcolor=black](L)(M){sample}
    \lens[abspos=2.7](L)(M)
    \mirror[labelangle=45](L)(M)(OSA){flip mirror}
    \optbox[position=end, optboxsize=3.2 1, innerlabel, 
      compshift=-0.2](M)(IRSpec){IR spectrometer}
    \lens[abspos=1](M)(OSA)
    \optplane[angle=90]([offset=-1.5]M)
    \optfiber[fiberloopradius=0.2, label=1, 
      newFiber={linewidth=1.5\pslinewidth}]([offset=-1.5]M)(OSA){MM fiber}
    \optbox[position=end, innerlabel, optboxwidth=1.1](M)(OSA){OSA}
    \newpsstyle{Beam}{linestyle=none, fillcolor=red, fillstyle=solid, raytrace=false}
    \drawwidebeam[beamwidth=0.2, stopinside]{1-2}
    \drawwidebeam[beamdiv=-25]{2-3}
    \drawwidebeam[loadbeampoints=false, beamdiv=25]{3-4}
    \drawwidebeam[startinside]{4-5}
    \drawwidebeam[savebeampoints=false, beamalign=abs]{5-6}
    \drawwidebeam[stopinside]{5}{7}
    \drawwidebeam[beamdiv=-25]{7-8}
  \end{optexp}
\end{pspicture}
```


Ex. 11.2

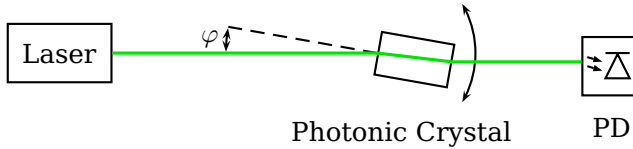


```

\begin{pspicture}(-4.2,-1)(3,3)
\nodes(-2,0){L0ut}(0,0){Grat}(4;45){0ut}(2.5;70){Mvar}
\newsstyle{Beam}{linewidth=2\pslinewidth, linecolor=red!90!black}
\begin{optexp}
\optbox[optboxwidth=2.2, labeloffset=0, position=start](L0ut)(Grat){diode laser}
\mirror[variable, label=0.5](Grat)(Mvar)(Grat){M$_{\mathrm{var}}$}
\optretplate[position=0.3](L0ut)(Grat){$\frac{\lambda}{4}$}
\optgrating(L0ut)(Grat)(0ut){grating}
\drawbeam[arrows=->]{1}{3}{4}(0ut)
\drawbeam{2}{4}
\end{optexp}
\end{pspicture}
\end{pspicture}

```

Ex. 11.3

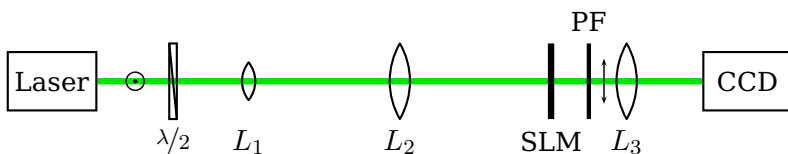


```

\begin{pspicture}(8.5,1.9)
\nodes(1.4,1.3){Laser}(7.6,1.3){Diode}
\optbox[position=start, labeloffset=0](Laser)(Diode){Laser}%
\optbox[abspos=4, optboxsize=1 0.6, labeloffset=1, n=3,
comname=PC, angle=-10, rotateref=l](Laser)(Diode){Photonic Crystal}
\optdetector[detttype=diode](\oencodeOut{PC})(Diode|\oencodeOut{PC}){PD}
\nodexn{(\oencodeIn{PC}) + (2;170)}{Angle1}
\psline[linestyle=dashed](\oencodeIn{PC})(Angle1)
\psarc<->(\oencodeIn{PC}){1.3}{330}{30}
\psarc[arcsep=1pt]{<->(\oencodeIn{PC}){2}{170}{180}
\uput{2.1}[175](\oencodeIn{PC}){\small $\varphi$}
\drawbeam{-}
\end{pspicture}

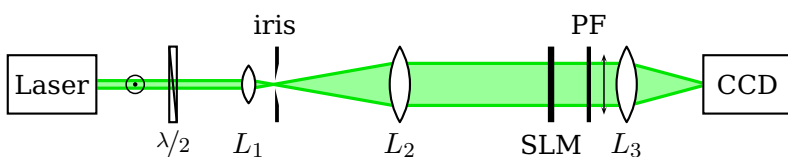
```

Ex. 11.4

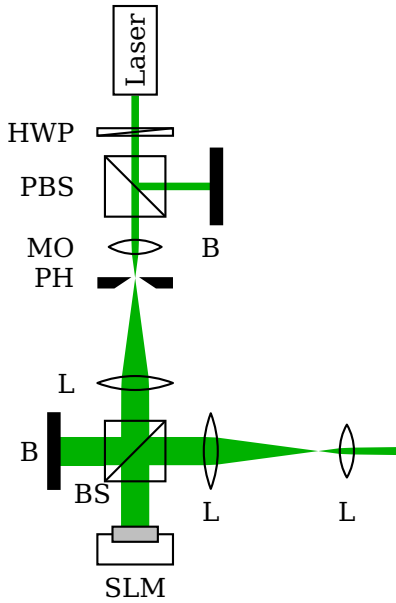


```
\begin{pspicture}(10.4,1.7)
\pnodes(1.2,1){Start}(9.2,1){CCD}
\begin{optexp}
\optbox[position=end, label=0, optboxwidth=1.2](CCD)(Start){Laser}
\optbox[position=end, label=0, optboxwidth=1.2](Start)(CCD){CCD}
\polarization[poltype=perp,abspos=0.5](Start)(CCD)
\optretplate[abspos=1](Start)(CCD){$\frac{\lambda}{2}$}
\lens[lens=0.4 0.4 0.5,abspos=2](Start)(CCD){$L_1$}
\lens[abspos=4](Start)(CCD){$L_2$}
\optplate[abspos=6, platelinewidth=3\pslinewidth](Start)(CCD){SLM}
\optplate[abspos=6.5, labelangle=180](Start)(CCD){PF}
\polarization[abspos=6.7](Start)(CCD)
\lens[abspos=7](Start)(CCD){$L_3$}
\drawbeam[linewidth=2\pslinewidth]{1}{2}
\end{optexp}
\end{pspicture}
```

Ex. 11.5



```
\begin{pspicture}(10.4,1.7)
\pnodes(1.2,1){Start}(9.2,1){CCD}
\begin{optexp}
\optbox[position=start, label=0, optboxwidth=1.2](Start)(CCD){Laser}
\polarization[poltype=perp,abspos=0.5](Start)(CCD)
\optretplate[abspos=1](Start)(CCD){$\frac{\lambda}{2}$}
\lens[lens=0.4 0.4 0.5,abspos=2](Start)(CCD){$L_1$}
\pinhole[pewidth=0.05, abspos=2.35, labelangle=180](Start)(CCD){iris}
\lens[abspos=4](Start)(CCD){$L_2$}
\optplate[abspos=6,platelinewidth=3\pslinewidth](Start)(CCD){SLM}
\optplate[abspos=6.5,labelangle=180](Start)(CCD){PF}
\lens[abspos=7](Start)(CCD){$L_3$}
\optbox[position=end, label=0, optboxwidth=1.2](Start)(CCD){CCD}
\polarization[abspos=6.7, polsize=0.8](Start)(CCD)
\addtopsstyle{Beam}{fillstyle=solid, fillcolor=green!40!white, loadbeampoints}
\drawwidebeam[beamwidth=0.1, stopinside]{1}{4}
\drawwidebeam[beamdiv=-20]{4}{6}
\drawwidebeam[stopinside]{6-9}
\drawwidebeam[beamdiv=-30]{9-10}
\end{optexp}
\end{pspicture}
```

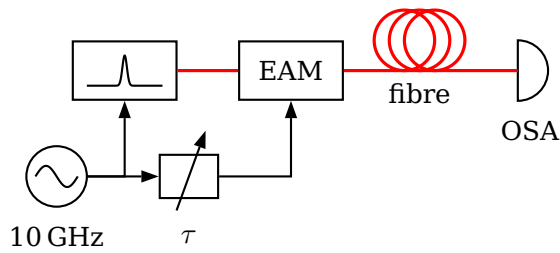
Ex. 11.6: Adapted from [arXiv:1112.5270v2](https://arxiv.org/abs/1112.5270v2)

```

\begin{pspicture}(0.3,0.4)(5.5,8.3)
\pnodes(2,7){L}(2,5.8){Pbs}([Xnodesep=1]Pbs){Blk1}(2,2.3){Bs}
\pnodes([Xnodesep=-1]Bs){Blk2}([Xnodesep=3.5]Bs){Out}(2,1.3){Slm}
\begin{optexp}
\optbox[position=end, innerlabel, optboxsize=1.2 0.6](Pbs)(L){Laser}
\psset{labelalign=r, mirrortype=extended}
\newpsstyle{ExtendedMirror}{fillstyle=solid, fillcolor=black}
\optretplate[position=0.4](L)(Pbs){HWP}
\beamsplitter[labelangle=-90](L)(Pbs)(Blk1){PBS}
\mirror[label=, -90 c](Pbs)(Blk1)(Pbs){B}
\lens[abspos=0.8, lens=0.7 0.7 0.7, n=2](Pbs)(Bs){M0}
\pinhole[pewidth=0.15, abspos=1.2](Pbs)(Bs){PH}
\lens[abspos=2.6, lensradius=1.4](Pbs)(Bs){L}
\psset{labelalign=c}
\beamsplitter[labelangle=-135](Pbs)(Bs)(Blk2){BS}
\mirror[labeloffset=0.4](Bs)(Blk2)(Bs){B}
\lens[abspos=1, lensradius=1.4](Bs)(Out){L}
\lens[abspos=2.8, lens=0.7 0.7 0.7, n=2](Bs)(Out){L}
\opttripole(Bs)(Slm)(Bs){%
\psframe[fillstyle=solid,fillcolor=gray!50](-0.3,0)(0.3,0.2)
\psline(-0.3,0.1)(-0.5,0.1)(-0.5,0.5)(0.5,0.5)(0.5,0.1)(0.3,0.1)}{SLM}
\addtopsstyle{Beam}{linestyle=none, fillstyle=solid, fillcolor=green!70!black}
\drawwidebeam[beamwidth=0.1]{1-4}
\drawwidebeam[beamwidth=0.1, beaminsidefirst, beaminsidelast]{3}{5-8}
\psset{savebeampoints=false, loadbeampoints}
\drawwidebeam{8-11}(Out)\drawwidebeam[beaminsidefirst]{8}{12}
\end{optexp}
\end{pspicture}

```

Ex. 11.7

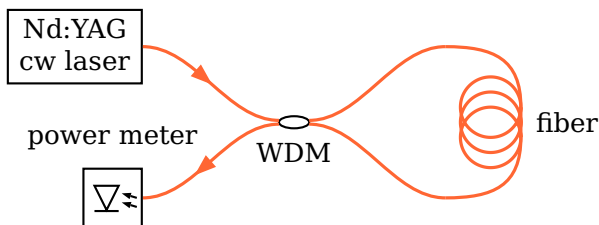


```

\begin{pspicture}(6.4,3.2)
\addtopsstyle{Fiber}{linecolor=red}\psset{fiber=none}
\pnodes(2.3,2.3){Lin}([Xnodesep=4.5]Lin){Det}
\optbox[label=0.2, position=start, compname=L, extnode=b](Lin)(Det){%
\psGauss[yunit=0.03,sigma=0.03]{-0.5}{0.5}}
\optbox[label=0, compname=EAM, extnode=b, abspos=1.5](Lin)(Det){EAM}
\optfiber[labeloffset=0.3, abspos=3.2](Lin)(Det){fibre}
\optdetector(Lin)(Det){OSA}
\pnode([Xnodesep=-0.5,offset=-1]\oenableExt{L}){Osc}
\elecsynthesizer[wire=none](Osc)(\oenableExt{EAM}|Osc){10\,GHz}
\optarrowcomp[arrowcompsize=0.8 0.6, 2
arrowscale=1.5](Osc)(\oenableExt{EAM}|Osc){\tau}
\drawfiber{1-4}
\addtopsstyle{Wire}{arrows=->, arrowinset=0, arrowscale=1.5}
\drawwire{5}{6}\psset[optexp]{wireangleB=90}
\drawwire{6}(\oenableExt{EAM})\drawwire{5}(\oenableExt{L})
\end{pspicture}

```

Ex. 11.8

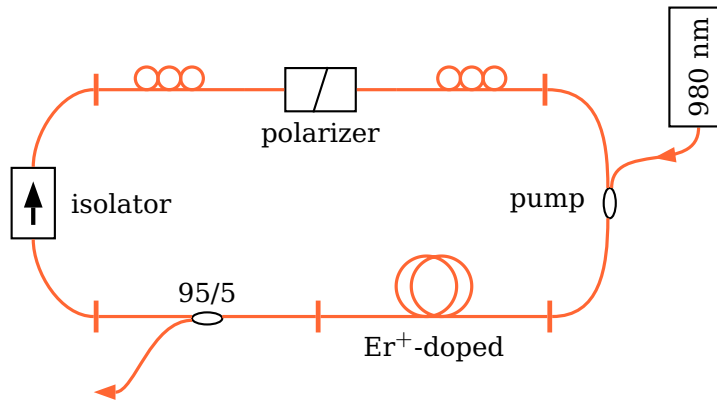


```

\begin{pspicture}(0.2,0.6)(8.2,3.5)
\pnodes(2,3){Laser}(2,1){PwMeter}(6,3){CplTop}(6,1){CplBot}
\psset{arrowscale=1.5, arrowinset=0}
\optbox[position=start, optboxsize=1.8 1, 2
labeloffset=0](Laser)([Xnodesep=0.1]Laser){%
\begin{tabular}{@{}c@{}}Nd:YAG\\[-0.4ex]cw laser\end{tabular}}
\optcoupler[addtoFiberIn1={ArrowInside=->}, addtoFiberIn2={ArrowInside=-<},
labeloffset=0.4](Laser)(PwMeter)(CplTop)(CplBot){WDM}
\optfiber[addtoFiberOut={ncurv=1, angleB=0}, addtoFiberIn={ncurv=1, angleA=0},
compshift=-1, label=0.2 . l](CplBot)(CplTop){fiber}
\optdetector[detttype=diode]([Xnodesep=0.1]PwMeter)(PwMeter){power meter}
\end{pspicture}

```

Ex. 11.9

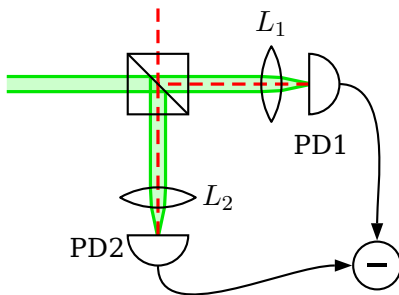


```

\begin{pspicture}(0.9,0.9)(10.4,6.1)
\psset{arrowscale=1.5, arrowinset=0}
\nodes(2,5){PC1in}(4,5){PC1out}(6,5){PC2in}(8,5){PC2out}(2,2){CplSig}
\nodes(5,2){CplIn}(2,1){CplOut}(10,4.5){Pump}(8,2){PumpSig}
\optisolator[compshift=0.8, addtoFiberIn={angleA=180},
             addtoFiberOut={angleB=180}, label=0.5 . l]%
             (CplSig)(PC1in){isolator}
\polcontrol[addtoFiberIn={arrows=|-}](PC1in)(PC1out)
\optfiberpolarizer[labeloffset=0.6](PC1out)(PC2in){polarizer}
\polcontrol[addtoFiberOut={arrows=-|}](PC2in)(PC2out)
\wdmsplitter[labeloffset=0.3, coupleralign=bottom, addtoFiberIn={arrows=|-},
             addtoFiberOut1={arrows=->}, addtoFiberOut2={arrows=-|}]%
             (CplIn)(CplOut)(CplSig){95/5}
\wdmcoupler[addtoFiberIn1={ArrowInside=->}, addtoFiberIn2={angleA=0},
            addtoFiberOut={angleB=0,arrows=-|}, ncurv=0.9,
            coupleralign=bottom, compshift=0.8](Pump)(PC2out)(PumpSig){pump}
\optbox[position=start, innerlabel, 2
        optboxwidth=1.6](Pump) ([offset=-0.1]Pump){980~nm}
\optfiber[fiberloops=2, labeloffset=0.4](CplIn)(PumpSig){Er$^+$-doped}
\end{pspicture}

```

Ex. 11.10

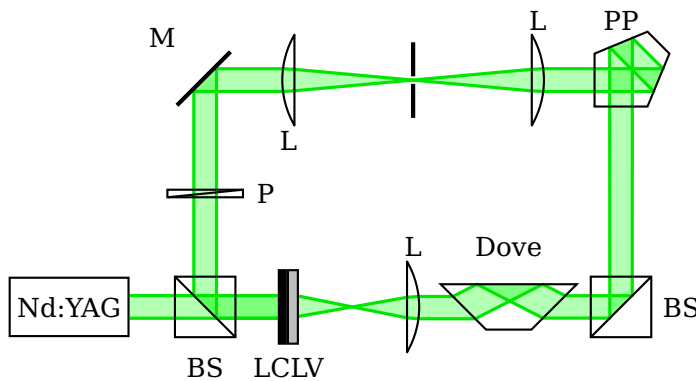


```

\begin{pspicture}(1,1.2)(6.2,5)
\pnodes(1,4){SigIn}(3,4){BS}(3,5){L0}(5,4){Det1}(3,2){Det2}
\begin{optexp}
  \optplane[compname=L0, angle=90](L0)
  \beamsplitter[compname=BS](SigIn)(BS)(Det2)
  \lens[abspos=0.5, compname=L2, n=2.1](Det2)(BS){$L_2$}
  \lens[abspos=0.5, compname=L1, n=2.1](Det1)(BS){$L_1$}
  \psset[optexp]{extnodealign=rel, extnode=r}
  \optdetector[compname=Det1](BS)(Det1){PD1}
  \optdetector[compname=Det2](BS)(Det2){PD2}
  \addtopsstyle{Beam}{beamwidth=0.2, fillstyle=solid,
    fillcolor=green, opacity=0.2}
  \drawwidebeam(SigIn){BS}{L2}{Det2}
  \drawwidebeam[beaminsidefirst]{BS}{L1}{Det1}
  \newpsstyle{Beam}{linecolor=red, linestyle=dashed, linewidth=1.5\pslinewidth}
  \drawbeam{L0}{BS}{L1}{Det1}
  \drawbeam[beaminsidefirst]{BS}{L2}{Det2}
\end{optexp}
\cnodeput[framesep=5pt]([Xnodesep=0.5, 2
  offset=-0.5]\oenodeExt{Det1}|\oenodeExt{Det2}){M}{\rule{2.5mm}{1pt}}
\psset{arrows=<- , arrowscale=1.5, arrowinset=0}
\ncurve[angleA=90]{M}{\oenodeExt{Det1}}
\ncurve[angleA=180, angleB=-90]{M}{\oenodeExt{Det2}}
\end{pspicture}

```

Ex. 11.11

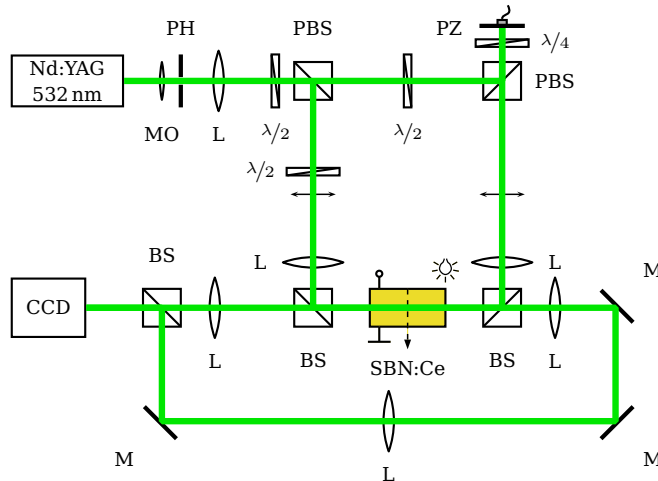


```

\begin{pspicture}(-0.2,0)(9,5)
\psset{optexp}{lens=1.2 0 1.2, n=1.72}
\addtopsstyle{Beam}{fillstyle=solid, fillcolor=green, opacity=0.3}
\nodes(2.4,1){BS1}([offset=3]BS1){M1}([Xnodesep=5.5]M1){PP}(PP|BS1){BS2}
\begin{optexp}
\optbox[label=0, position=start, optboxwidth=1.6]([Xnodesep=-1]BS1)(BS1){Nd:YAG}
\beamsplitter[comname=BS](BS2)(BS1)(M1){BS}
\optdipole[position=0.2, comname=LCLV, optdipolesize=0.28 1](BS1)(BS2){%
\psframe[fillstyle=solid,fillcolor=black,dimen=outer](-0.14,-0.5)(-0.02,0.5)
\psframe[fillstyle=solid,fillcolor=gray!50,dimen=outer](-0.02,-0.5)(0.14,0.5)
}{LCLV}
\optretplate(BS1)(M1){P}
\mirror(BS1)(M1)(PP){M}
\lens[position=0.2](M1)(PP){L}
\pinhole(M1)(PP)
\lens[position=0.2](PP)(M1){L}
\pentaprism(M1)(PP)(BS2){PP}
\beamsplitter(PP)(BS2)(BS1){BS}
\doveprism[comname=Dove, position=0.27, n=2.3](BS2)(BS1){Dove}
\lens[n=2.5](BS2)(BS1){L}
\drawwidebeam[beamwidth=0.3]{1-3}
\drawwidebeam[loadbeampoints]{3}{2}{4-}{3}
\end{optexp}
\end{pspicture}

```

Ex. 11.12

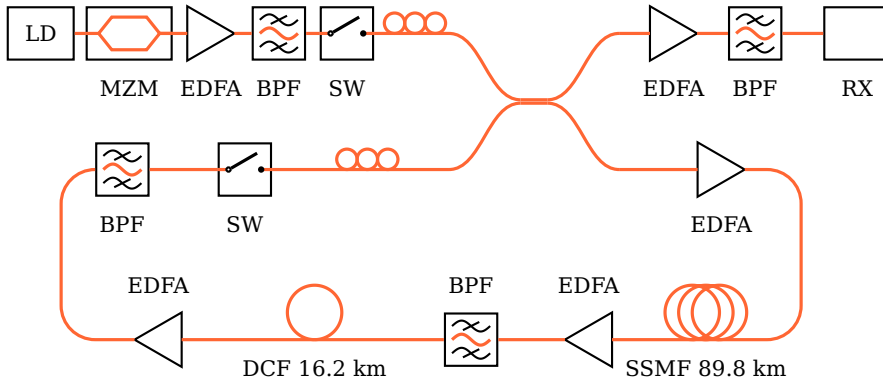


```

\begin{pspicture}(0,-0.2)(8.6,5.6)
\pnodes(1.5,5){Laser}(4,5){PBS}(6.5,5){PBS2}(6.5,5.7){piezo}(4,2){BSFwd}
\pnodes(6.5,2){BSBwd}(2,2){BS4f}(2,0.5){M4f3}(8,2){M4f1}(8,0.5){M4f2}(1,2){CCD}
\psset{mirrorwidth=0.6, plateheight=0.7, outerheight=0.7, labeloffset=0.7, }
\labelstyle=\scriptsize, lens=1.2 1.2 0.8, bssize=0.5}
\optbox[position=start, optboxsize=1.5 0.7, innerlabel]%
(Laser)(PBS){\parbox{1.5cm}{\centering Nd:YAG\ 532\,nm}}
\lens[lensheight=0.5, position=0.2](Laser)(PBS){M0}
\pinhole[position=0.3, labelangle=180](Laser)(PBS){PH}
\lens[position=0.5](Laser)(PBS){L}
\optretplate[position=0.8](Laser)(PBS){$\frac{\lambda}{2}$}
\beamsplitter(Laser)(PBS)(BSFwd){PBS}
\optretplate[position=0.4](PBS)(BSFwd){$\frac{\lambda}{2}$}
\polarization(PBS)(BSFwd)\polarization(PBS2)(BSBwd)
\lens[position=0.8](PBS)(BSFwd){L}
\optretplate(PBS)(PBS2){$\frac{\lambda}{2}$}
\beamsplitter(PBS)(PBS2)(piezo){PBS}
\optretplate[abspos=0.5](PBS2)(piezo){$\frac{\lambda}{4}$}
\mirror[mirrortype=piezo, labelangle=90](PBS2)(piezo){PZ}
\lens[position=0.8, labelangle=180](PBS2)(BSBwd){L}
\crystal[crystalsize=1 0.5, voltage, lamp, fillstyle=solid, ]
fillcolor=yellow!90!black, labeloffset=0.8](BSFwd)(BSBwd){SBN:Ce}
\beamsplitter(PBS)(BSFwd)(BSBwd){BS}
\beamsplitter[labelangle=-90](PBS2)(BSBwd)(BSFwd){BS}
\mirror(BSBwd)(M4f1)(M4f2){M}\mirror(M4f1)(M4f2)(M4f3){M}
\lens[labelangle=180](M4f2)(M4f3){L}\mirror(M4f2)(M4f3)(BS4f){M}
\beamsplitter(M4f3)(BS4f)(CCD){BS}
\optbox[position=end, label=0, optboxwidth=1](BS4f)(CCD){CCD}
\lens[abspos=0.7](BS4f)(BSFwd){L}\lens[abspos=0.7](BSBwd)(M4f1){L}
\addtopsstyle{Beam}{linewidth=2\pslinewidth, linejoin=2}
\drawbeam{1-6}{11-14}{13}{12}{15}{18}{16}{17}{25}{23}{24}
\drawbeam[beaminsidefirst]{6}{7}{10}{17}{16}{18}{26}{19-24}
\end{pspicture}

```


Ex. 11.13

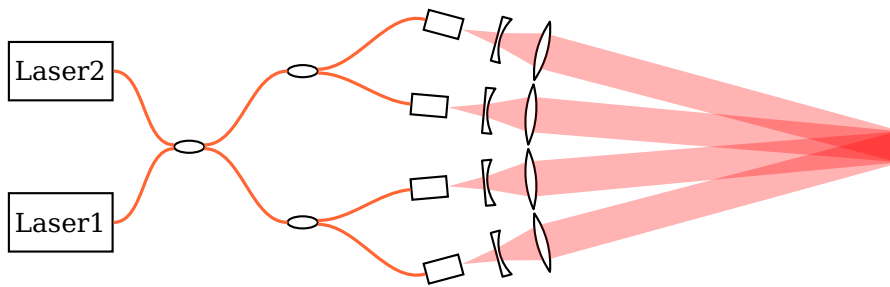


```

\begin{pspicture}(13,5.5)
\psset{optboxwidth=1, labelstyle=\footnotesize, fiber=none}
\pnodes(1,5){LD}([Xnodesep=5.5]LD){CPLin1}([offset=-2]CPLin1){CPLin2}
\pnodes([Xnodesep=2.5]CPLin1){CPLout1}([Xnodesep=2.5]CPLin2){CPLout2}
\pnode([Xnodesep=3]CPLout1){RX}
\optbox[position=start, label=0](LD)(CPLin1){LD}
\optmzm[abspos=0.8](LD)(CPLin1){MZM}
\optamp[abspos=2](LD)(CPLin1){EDFA}
\optfilter[abspos=3](LD)(CPLin1){BPF}
\optswitch[abspos=4](LD)(CPLin1){SW}
\polcontrol[abspos=5, fiber=out](LD)(CPLin1)
\drawfiber{1-6}
\optcoupler[couplertype=none, fiber](CPLin1)(CPLin2)(CPLout1)(CPLout2)
\optamp[abspos=0.8, fiber=in](CPLout1)(RX){EDFA}
\optfilter[abspos=2](CPLout1)(RX){BPF}
\optbox[position=end](CPLout1)(RX){RX}
\drawfiber{8-10}
\pnodes([Xnodesep=-0.5]LD|CPLin2){TL}(RX|TL){TR}([offset=-2.5]TR){BR}(TL|BR){BL}
\optamp[fiber=i](CPLout2)(TR){EDFA}
\optfiber[label=0.3 . t, position=0.85](BL)(BR){SSMF 89.8~km}
\drawfiber[fiberstyle=angle, arm=1.2, lineararc=0.5, startnode=N]{11}{12}
\optamp[position=0.3](BR)(BL){EDFA}
\optfilter[position=0.55, labelangle=180](BL)(BR){BPF}
\optfiber[fiberloops=1, label=0.3 . t, position=0.35](BL)(BR){DCF 16.2~km}
\optamp[position=0.85](BR)(BL){EDFA}
\optfilter[position=0.2](TL)(CPLin2){BPF}
\optswitch(TL)(CPLin2){SW}
\polcontrol[position=0.8, fiber=out](TL)(CPLin2)
\drawfiber{12-16}\drawfiber{17-19}
\drawfiber[fiberstyle=angle, arm=0.5, lineararc=0.5, stopnode=1]{16}{17}
\end{pspicture}

```

Ex. 11.14

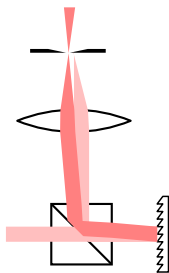


```

\begin{pspicture}(12, 4)
\psset{labeloffset=0.3, fiber=none}
\nodes(1.6,2){In}(12, 2){Ref}
\optlate[linestyle=dashed, plateheight=3, position=1, compname=RefPlane](In)(Ref)
\addtopsstyle{Beam}{linestyle=none, beamdiv=20, beaminside=false, }
\fillstyle=solid, fillcolor=red, opacity=0.3}
\multido{\i=1+1, \ii=165+10}{4}{%
\rput(Ref){\pnode(6;\ii){A\i}}
\optbox[optboxsize=0.5 0.3, position=end, compname=Box\i, extnode=l](Ref)(A\i)
\lens[lens=0 -0.6 0.6, abspos=0.5, compname=L\i](A\i)(Ref)
\lens[lens=1.2 1.2 0.8, abspos=1.1, n=1.65, compname=LC\i](A\i)(Ref)
\drawwidebeam{Box\i}{L\i}{LC\i}{RefPlane}
}%
\optcoupler[abspos=1](In)(In)(Ref)(Ref)
\wdmsplitter[abspos=2.5, compshift=1](In)(Ref)(Ref)
\wdmsplitter[abspos=2.5, compshift=-1](In)(Ref)(Ref)
\optbox[compshift=-1, position=start, label=0](In)(Ref){Laser1}
\optbox[compshift=1, position=start, label=0](In)(Ref){Laser2}
\drawfiber{17}{14}{15}{Box1}\drawfiber{15}{Box2}
\drawfiber{18}{14}{16}{Box3}\drawfiber{16}{Box4}
\end{pspicture}

```

Ex. 11.15

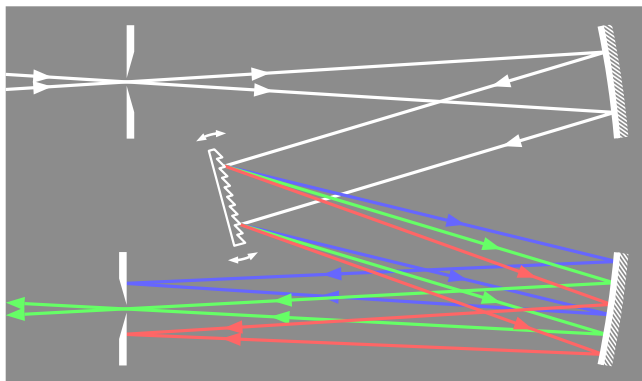


```

\begin{pspicture}(0.5,0)(2.2,4)
\nodes(0,1){A}(1,1){BS}(2,1){G}(1,4){B}
\beamsplitter(G)(BS)(B) \optgrating(BS)(G)(BS)
\lens[lens=2 2 1.5, compshift=0.1, n=2.25](BS)(B)
\pinhole[phwidth=0.05, innerheight=0.05, position=0.8, }
compshift=0.18](BS)(B)
\optplane[angle=90](B)
\addtopsstyle{Beam}{linestyle=none, beamwidth=0.2, fillstyle=solid}
\drawwidebeam[fillcolor=red!25!white](A){1-2}{1}{3-5}
\drawwidebeam[fillcolor=red!50!white, beamangle=-5]{2}{1}{3-5}
\end{pspicture}

```

Ex. 11.16: Adapted from Wikipedia

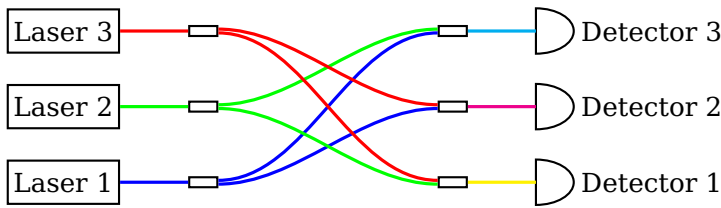


```

\begin{pspicture}(0,-1)(8.5,4)
\psframe[fillstyle=solid,fillcolor=gray!90,linestyle=none](0,-1)(8.5,4)
\addtopsstyle{OptComp}{linecolor=white}
\newpsstyle{Beam}{ArrowInside=->, linejoin=2, arrowscale=1.3, arrowinset=0}
\addtopsstyle{ExtendedMirror}{hatchcolor=white, hatchsep=0.5\pslinewidth}
\pnodes(0,3){A}(8,3){B}(3,1.5){C}(8,0){D}(0,0){E}%
\psset{linewidth=1.5\pslinewidth, mirrorradius=13, mirrorwidth=1.5, 2
    gratingwidth=1.3, mirrortype=extended, phwidth=0.1, outerheight=1.5}%
\begin{optexp}
  \pinhole[position=0.2](A)(B)%
  \mirror(A)(B)(C)
  \optgrating[reverse, angle=15, variable](B)(C)(D)
  \mirror(C)(D)(E)
  \pinhole[position=0.8](D)(E)
  \drawwidebeam[ArrowInsidePos=0.3, beamwidth=0.2, linecolor=white, 2
    beamdiv=-7.15](A){1-3}
  \psset{loadbeampoints, savebeampoints=false}%
  \drawwidebeam[ArrowInsidePos=0.6, beamangle=3, linecolor=blue!60!white]{3-5}(E)
  \drawwidebeam[ArrowInsidePos=0.7, linecolor=green!60!white, arrows=->, 2
    ArrowInsideMinLength=2]{3-5}(E)
  \drawwidebeam[ArrowInsidePos=0.8, beamangle=-3, linecolor=red!60!white]{3-5}(E)
\end{optexp}
\end{pspicture}

```

Ex. 11.17

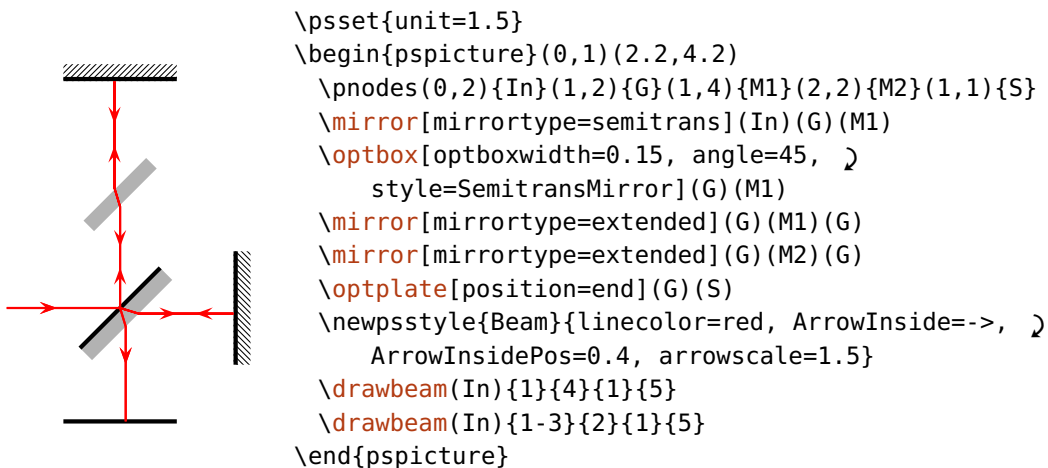


```

\begin{pspicture}(0,0.7)(9.4,3.3)
\psset{optexp}{couplertype=rectangle, fiber=none, detsize=0.5 0.6}
\multido{\i=1+1}{3}{%
\optbox[optboxsize=1.5 0.6, innerlabel, position=start](1.5,\i)(7,\i){Laser \i}
\wdmsplitter[position=0.2](1.5,\i)(7,\i)(7,\i)
\wdmcoupler[position=0.8](1.5,\i)(1.5,\i)(7,\i)
\optdetector[label=0.4 90 l](1.5,\i)(7,\i){Detector \i}
}%
\drawfiber[linecolor=blue, startnode=N]{1}{2}{7}
\drawfiber[linecolor=blue]{2}{11}
\drawfiber[linecolor=green, stopnode=1]{5}{6}{11}
\drawfiber[linecolor=green, stopnode=2]{6}{3}
\drawfiber[linecolor=red]{9}{10}{3}
\drawfiber[linecolor=red, startnode=2]{10}{7}
\drawfiber[linecolor=cyan]{11}{12}
\drawfiber[linecolor=magenta]{7}{8}
\drawfiber[linecolor=yellow]{3}{4}
\end{pspicture}

```

Ex. 11.18

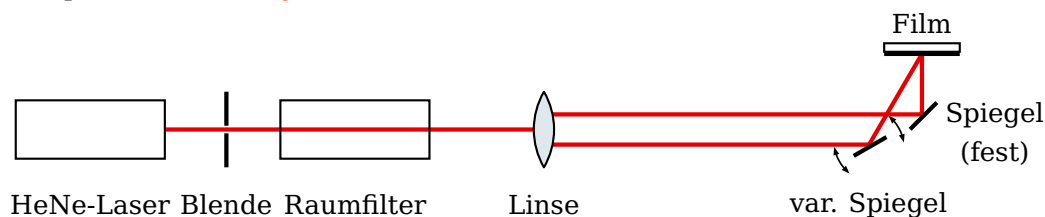


```

\psset{unit=1.5}
\begin{pspicture}(0,1)(2.2,4.2)
\pnodes(0,2){In}(1,2){G}(1,4){M1}(2,2){M2}(1,1){S}
\mirror[mirrortype=semitrans](In)(G)(M1)
\optbox[optboxwidth=0.15, angle=45, 2
style=SemitransMirror](G)(M1)
\mirror[mirrortype=extended](G)(M1)(G)
\mirror[mirrortype=extended](G)(M2)(G)
\optplate[position=end](G)(S)
\newpsstyle{Beam}{linecolor=red, ArrowInside=->, 2
ArrowInsidePos=0.4, arrowscale=1.5}
\drawbeam(In){1}{4}{1}{5}
\drawbeam(In){1-3}{2}{1}{5}
\end{pspicture}

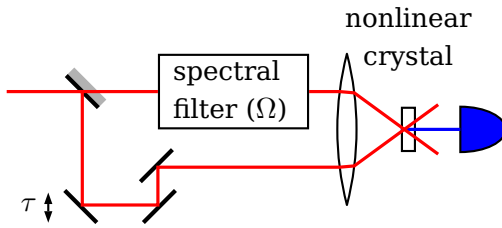
```

Ex. 11.19: Adapted from semibyte.de



```
\begin{pspicture}(13.6,2.6)
\psset{optexp}{labeloffset=1, mirrorwidth=0.5, optboxwidth=2}
\nodes(2,1){Laser}(12,2){Film}(Film|Laser){Ref}
\begin{optexp}
\optbox[position=start](Laser)(Ref){HeNe-Laser}
\pinhole[abspos=0.8](Laser)(Ref){Blende}
\optbox[abspos=2.5](Laser)(Ref){Raumfilter}
\lens[fillstyle=solid, fillcolor={rgb}{0.87,0.91,0.93}](Laser)(Ref){Linse}
\mirror[label=1 30]([offset=0.2]Laser)([offset=0.2]Ref)(Film){%
\begin{tabular}{c}Spiegel\\(fest)\end{tabular}}
\opttripole[label=0.4](Ref)(Film)(Ref){%
\psframe(0.5,1\pslinewidth)(-0.5,5\pslinewidth)
\psline[linewidth=2\pslinewidth](0.5,0)(-0.5,0)
}{Film}
\newpsstyle{Beam}{linewidth=2\pslinewidth, linecolor=red!90!black, 2
beamalign=absolute}
\drawbeam{1-4}
\drawbeam[beampos=0.2]{4-6}
\mirror[variable, label=0.8 -30]([offset=-0.2]Laser)([offset=-0.2, 2
Xnodesep=-0.7]Ref)(Film){var. Spiegel}
\drawbeam[beampos=-0.2]{4}{7}{6}
\end{optexp}
\end{pspicture}
```

Ex. 11.20

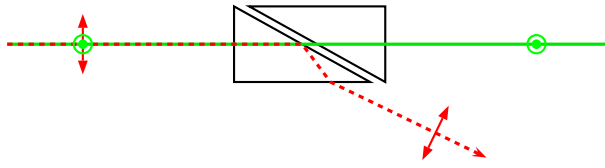


```

\begin{pspicture}(0,0.2)(6.6,3.1)
\psset{mirrorwidth=0.6, useNA=false, beaminsidfirst}
\pnodes(0,2){In}(1,2){BS}(1,0.5){M1}(2,1){M2}(6,1.5){Det}
\mirror[mirrortype=semitrans, n=1](In)(BS)(M1)
\mirror[label=0.6 180 . absolute, compname=M](BS)(M1)(M2|M1)
\mirror(M1)(M2|M1)(M2)
\mirror(M2|M1)(M2)([Xnodesep=1]M2)
\optbox[optboxsize=2 1, abspos=2, label=0, 2
allowbeaminside=false](BS)([offset=0.5]Det){%
\parbox{1.6\psxunit}{spectral\filter ($\Omega$)}}
\lens[lens=4 4 2, n=3.3]([Xnodesep=-3]Det)(Det)
\optbox[optboxsize=0.2 0.6, n=1, label=-1.2](\oencodeOut{6})(Det){%
\begin{tabular}{@{}c@{}}nonlinear\crystal\end{tabular}}
\optplate[linestyle=none](\oencodeOut{7})(Det)
\optdetector[detsize=0.6 0.6, fillstyle=solid, fillcolor=blue](\oencodeOut{6})(Det)
\newsstyle{Beam}{linewidth=1.5\pslinewidth}
\drawbeam[linecolor=blue]{7-9}
\addtopsstyle{Beam}{linecolor=red}
\drawbeam(In){1-4}{6-8}
\drawbeam{1}{5-8}
\rput[r](\oencodeLabel{M}){$\tau$}
\psline[arrows=<->, arrowinset=0](0.15,0.25)(0.15,-0.15)}
\end{pspicture}

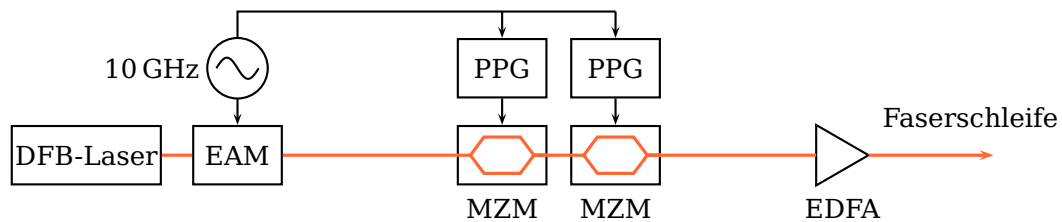
```

Ex. 11.21



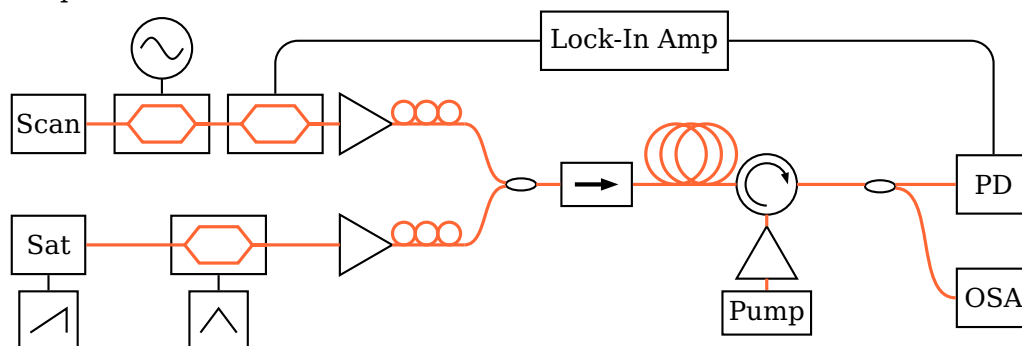
```
\begin{pspicture}(-1,-0.5)(8,1.5)
  \newpsobject{perppol}{polarization}{poltype=perp, linecolor=green, polsize=1}
  \newpsobject{parpol}{polarization}{poltype=parallel, linecolor=red, polsize=0.8}
  \newpsstyle{Polarization}{arrowinset=0, arrowscale=1.2}
  \pnodes(0,1){A}(8,1){B}(4,-0.5){C}
  \glanthompson[glanthompsongap=0.2, glanthompsonsize=2 1](A)(B)
  \optplane(B)\optplane[angle=90](C)
  \drawbeam(A){1-2}
  \drawbeam[linestyle=dashed, dash=2pt 2pt, linecolor=red, arrows=->](A){1}{3}
  \parpol[abspos=1](A)(B)\perppol[abspos=1](A)(B)
  \nodexn{(\oenameBeam)}-1.5(!\oeBeamVec){}{D}
  \perppol[abspos=7](A)(B)\parpol(D)(\oenameBeam){}
\end{pspicture}
```

Ex. 11.22



```
\begin{pspicture}(13.7,2.9)
  \newpsstyle{Wire}{arm=0.1, arrows=->}
  \psset[optexp]{optboxsize=1.2 0.8, optmzmsize=1.2 0.8, fiber=none, label=0.7}
  \pnodes(2,1){L}([Xnodesep=11]L){A}
  \optbox[position=start, optboxwidth=2, label=0](L)(A){DFB-Laser}
  \optbox[abspos=1, label=0, compname=Eam, extnode=t](L)(A){EAM}
  \optmzm[abspos=4.5, compname=Mzm1, extnode=t](L)(A){MZM}
  \optmzm[abspos=6, compname=Mzm2, extnode=t](L)(A){MZM}
  \optamp[abspos=9](L)(A){EDFA}
  \drawfiber{1-5}\drawfiber[arrows=->]{}(A)
  \pnode([offset=1.5]\oenameExt{Eam}){Osci}
  \elec synthesizer[compname=Syn, position=0.5, wire=o, label=0.5 . 2]
    r](Osci)(\oenameExt{Eam}){10\,GHz}
  \psset[optexp]{label=0, wire, optboxsize=0.8 1.2}
  \optbox[compname=Ppg1](\oenameExt{Mzm1}|Osci)(\oenameExt{Mzm1}){PPG}
  \optbox[compname=Ppg2](\oenameExt{Mzm2}|Osci)(\oenameExt{Mzm2}){PPG}
  \drawwire[arrows=-, wireangleB=-90]{Syn}(\oenameExt{Mzm2}|Osci)
  \rput([offset=0.5, Xnodesep=-0.3]A){Faserschleife}
\end{pspicture}
```

Ex. 11.23: Adapted from DOI:10.1364/OL.37.000930

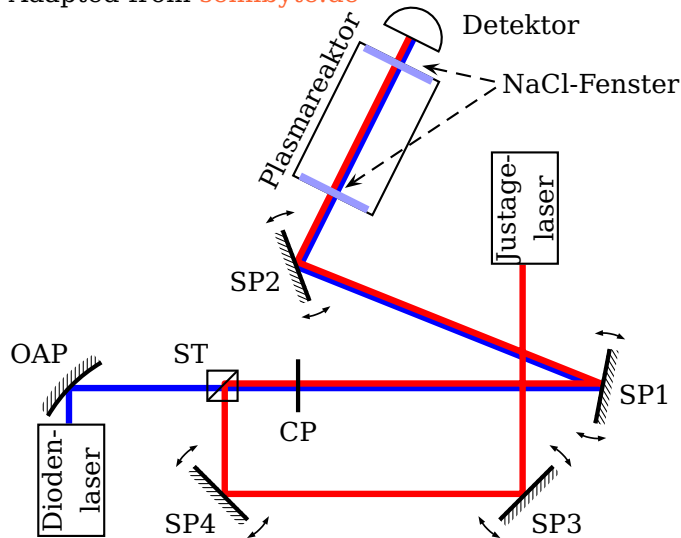


```

\begin{pspicture}(0,0.2)(13.5,4.7)
\psset{optexp}{optboxsize=1 0.8, label=0, fiber=none}
\nodes(1,3.2){Scan}\nnode([offset=-1.6]Scan){Sat}
\node([Xnodesep=5]Scan){ScanCpl}\nnode(ScanCpl|Sat){SatCpl}
\node([Xnodesep=1.5,offset=-0.8]ScanCpl){Cpl}\nnode([Xnodesep=5]Cpl){PD}
\node([Xnodesep=2.5, offset=-1.4]Cpl){Pump}
\optbox[position=start](Scan)(ScanCpl){Scan}
\optmzm[abspos=1, compname=M1, extnode=t](Scan)(ScanCpl)
\optmzm[abspos=2.5, compname=M2, extnode=t](Scan)(ScanCpl)
\optamp[abspos=3.7](Scan)(ScanCpl)
\polcontrol[abspos=4.5, fiber=o](Scan)(ScanCpl)
\drawfiber{1-5}
\optbox[position=start, compname=S, extnode=b](Sat)(SatCpl){Sat}
\optmzm[abspos=1.75, compname=M3, extnode=b](Sat)(SatCpl)
\optamp[abspos=3.7](Sat)(SatCpl)
\polcontrol[abspos=4.5, fiber=o](Sat)(SatCpl)
\drawfiber{6-9}
\wdmcoupler[fiber=i](ScanCpl)(SatCpl)(Cpl)
\optisolator[position=0.1](Cpl)(Pump|Cpl)
\optfiber[position=0.6](Cpl)(Pump|Cpl)
\addtopsstyle{OptCircArrow}{arrows=<-}
\optcirculator[optcircangle=0 -90](Cpl)(PD)(Pump)
\wdmsplitter[position=0.8, coupleralign=t](Cpl)(PD)
\optbox[position=end, compname=Pd, extnode=t](Cpl)(PD){PD}
\optbox[position=end, compshift=-1.5](Cpl)(PD){OSA}
\optamp[fiber](Pump)(\oencodeIfc{2}{13})
\optbox[position=start, optboxsize=0.6 1.2](Pump)(Pump|Cpl){Pump}
\drawfiber{10-15}\drawfiber{14}{16}
\optbox[compshift=0.7, optboxsize=2.5 0.8](\oencodeExt{M2})(\oencodeExt{Pd}){\oencodeExt{M2}}{Lock-In Amp}
\addtopsstyle{Wire}{arm=0.1}
\drawwire[wireangleA=90, linearc=0.3](\oencodeExt{M2}){}
\drawwire[wireangleB=90, linearc=0.3](\oencodeExt{Pd}){}
\elecsynthesizer([offset=0.2]\oencodeExt{M1})(\oencodeExt{M1})
\psset{optexp}{synthshape=rectangle}
\elecsynthesizer[synthtype=sawtooth](\oencodeExt{S}|Pump)(\oencodeExt{S})
\elecsynthesizer[synthtype=triangle](\oencodeExt{M3}|Pump)(\oencodeExt{M3})
\end{pspicture}

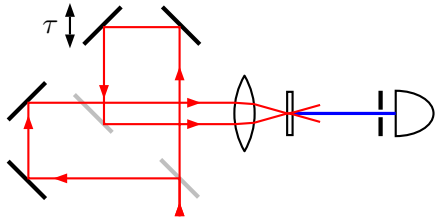
```


Ex. 11.24: Adapted from semibyte.de



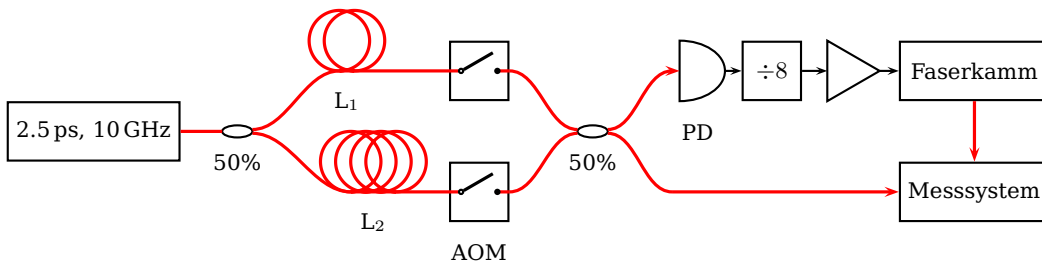
```
\begin{pspicture}(0.2,0.5)(9,7.4)
  \pnode(1,2){DL}\pnode([offset=0.5]DL){OAP}\pnode([Xnodesep=2]OAP){ST}
  \pnode([offset=-1.4]ST){SP4}\pnode([Xnodesep=4]SP4){SP3}
  \pnode([offset=3]SP3){JL}\pnode([Xnodesep=1]SP3|ST){SP1}
  \pnode([Xnodesep=1]ST|JL){SP2}\pnode([Xnodesep=1.5,offset=3]SP2){Det}
  \psset{optexp}{optboxsize=1.5 0.9, labeloffset=0.6}
  \begin{optexp}
    \optbox[position=start, innerlabel](DL)(OAP){\parbox{1.5\psunit}{\centering }
      Dioden-\\laser}}
    \psset{optexp}{mirrortype=extended}
    \mirror[mirrorradius=2](DL)(OAP)(ST){OAP}
    \psset{optexp}{variable}
    \beamsplitter[bssize=0.4, labelangle=-45](SP4)(ST)(SP1){ST}
    \optplate[abspos=1, plateheight=0.7](ST)(SP1){CP}
    \mirror(ST)(SP1)(SP2){SP1}\mirror(SP1)(SP2)(Det){SP2}
    \optdipole[label=-0.7 . b relative, optdipolesize=2, position=0.6](SP2)(Det){%
      \psframe[dimen=outer](-1,-0.6)(1,0.6)
      \psset{fillstyle=solid, fillcolor=blue!40, linestyle=none}
      \psframe(-1,-0.5)(-0.9,0.5)\psframe(0.9,-0.5)(1,0.5)}{Plasmareaktor}
    \optdetector[label=. 30 l](SP2)(Det){Detektor}
    \optbox[position=end, innerlabel](SP3)(JL){\parbox{1.5\psunit}{\centering }
      Justage-\\laser}}
    \mirror(SP4)(SP3)(JL){SP3}\mirror(ST)(SP4)(SP3){SP4}
    \psset{linewidth=2\pslinewidth}
    \drawbeam[linecolor=blue, beampos=0.03]{1-2}
    \drawbeam[beamalign=abs, loadbeampoints, linecolor=blue]{2-8}
    \drawbeam[linecolor=red, beampos=-0.03]{9-11}{3-8}
  \end{optexp}
  \pnode([Xnodesep=1.2,offset=-0.6]\oenodeIn{8}){NaCl}
  \psset{linestyle=dashed, arrows=->, arrowscale=1.5}
  \ncline[nodesepA=0.1, nodesepB=0.2]{NaCl}{\oenodeIn{7}}
  \ncline[nodesepA=0.1, nodesepB=0.3]{NaCl}{\oenodeOut{7}}
  \rput[l](NaCl){NaCl-Fenster}
\end{pspicture}
```

Ex. 11.25: Adapted from DOI:10.1364/AOP.1.000308, Fig. 8



```
\begin{pspicture}(5.7,2.8)
\psset{mirrorwidth=0.7, bsstyle=plate, bssize=0.5, arrowinset=0, arrowscale=1.2}
\newpsobject{splitterplate}{beamsplitter}{linewidth=2\pslinewidth, }
\psnode(2.3,0){In}([offset=0.5]In){STM1}([Xnodesep=-2]STM1){M1}
\psnodes([offset=1]M1){M2}([offset=2]STM1){M3}([Xnodesep=-1]M3){M4}(M4|M2){STM2}
\splitterplate(In)(STM1)(M1)
\mirror(STM1)(M1)(M2)
\mirror(M1)(M2)(M3|M2)
\mirror(STM1)(M3)(M4)
\mirror[label=0.6 180 . absolute](M3)(M4)(STM2)
\rput[r](\oencodeLabel){}\tau$
\psline[arrows=<->](0.15, 0.4)(0.15, -0.2)
\splitterplate[compoffset=0.2, compname=STM2](M4)(STM2)([Xnodesep=1]STM2)
\psnode(\oencodeCenter{STM2}){STM2C}
\psnode([Xnodesep=4]STM2C){Out}
\lens[n=2](STM2C)(Out)
\optbox[position=0.65, optboxsize=0.1 0.6](STM2C)(Out)
\optplate[linestyle=none, position=0.75](STM2C)(Out)
\pinhole[position=0.95, outerheight=0.6](STM2C)(Out)
\optdetector[detsize=0.5 0.6](STM2C)(Out)
\drawbeam[linecolor=blue]{8-}
\newpsstyle{Beam}{ArrowInside=>, linecolor=red, ArrowInsideMinLength=1, arrows=>-}
\drawbeam[ArrowInsidePos=0.835](In){1-3}{7-9}
\drawbeam[ArrowInsidePos=0.74](In){1}{4-9}
\end{pspicture}
```

Ex. 11.26

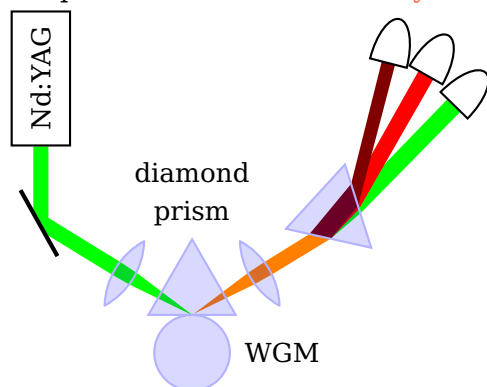


```

\begin{pspicture}(13.8,3.8)
  \node(2.3,2){Signal}
  \node([Xnodesep=1.5,offset=0.8]Signal){UpIn}
  \node([Xnodesep=1.5, offset=-0.8]Signal){LowIn}
  \node([Xnodesep=3.5]UpIn){UpOut}
  \node(UpOut|LowIn){LowOut}
  \node([Xnodesep=1.5]UpOut){UpIn2}
  \node([Xnodesep=3]UpIn2){UpOut2}
  \node([Xnodesep=1.5]LowOut){LowIn2}
  \node(UpOut2|LowIn2){LowOut2}
  \psset{labelstyle=\footnotesize}
  \psset{fiber=none}\addtopsstyle{Fiber}{linecolor=red}
  \psset{wire=none}\addtopsstyle{Wire}{arm=0.1, arrows=->}
  \optbox[position=start, label=0, optboxsize=2.3]{
    0.8](Signal)([Xnodesep=0.1]Signal){2.5\,ps, 10\,GHz}
  }
  \wdmsplitter[label=0.4](Signal)(UpIn)(LowIn){50\%}
  \optfiber[fiberloops=4, label=0.4, position=0.3](LowIn)(LowOut){L$_2$}
  \optswitch[position=0.7](LowIn)(LowOut){AOM}
  \optfiber[fiberloops=2, label=0.4, position=0.2](UpIn)(UpOut){L$_1$}
  \optswitch[position=0.7](UpIn)(UpOut)
  \optcoupler[label=0.4, position=0.3](UpOut)(LowOut)(UpIn2)(LowIn2){50\%}
  \optbox[optboxsize=2 0.8, position=end, label=0,]{
    extnode=t](LowIn2)(LowOut2){Messsystem}
  }
  \drawfiber{1-4}{7}(LowIn2)\drawfiber[arrows=->](LowIn2){8}
  \optdetector[abspos=0.4, detsize=0.6 0.8](UpIn2)(UpOut2){PD}
  \optbox[optboxsize=0.8 0.8, label=0, position=0.44](UpIn2)(UpOut2){$\div 8$}
  \optamp[position=0.8](UpIn2)(UpOut2)
  \optbox[position=end, extnode=b, label=0, optboxsize=2]{
    0.8](UpIn2)(UpOut2){Faserkamm}
  }
  \drawfiber{2}{5-7}(UpIn2)\drawfiber[arrows=->](UpIn2){9}
  \drawwire{9-12}
  \drawfiber[arrows=->](\oencodeExt{12})(\oencodeExt{8})
\end{pspicture}

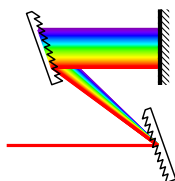
```

Ex. 11.27: Adapted from DOI:10.1103/PhysRevLett.105.263904



```
\begin{pspicture}(0.6,2.8)(6.8,7.8)
\nodes(1,6){Laser}(1,5){M}(3,4){PR}(5,5){PT}(6,7){D1}
\newsstyle{Glass}{fillcolor=blue, fillstyle=solid, opacity=0.15,
  \linecolor=blue!30!white}
\begin{optexp}
\optbox[position=start, innerlabel, optboxwidth=1.8](Laser)(M){Nd:YAG}
\optprism[style=Glass, compname=P, n=1, prismtype=reflective, prismangle=60,
  \labeloffset=1.2](M)(PR)(PT){\begin{tabular}{@{}c@{}}diamond\\
  prism\end{tabular}}
\mirror(Laser)(M){\oenableIfc{2}{P}}%
\lens[abspos=1.05, style=Glass]{\oenableIfc{2}{P}}(M)%
\optprism[compname=PDet, style=Glass]{\oenableIfc{2}{P}}(PT)(D1)%
\lens[abspos=1.05, style=Glass]{\oenableIfc{2}{P}}{\oenableOut{PDet}}%
\newsstyle{Beam}{linestyle=none, fillstyle=solid}%
\drawwidebeam[fillcolor=green, beamwidth=0.2, beaminsidelast,
  \beampathcount=5]{1}{3}{4}{2}%
\drawwidebeam[fillcolor=orange, beaminsidefirst, startinsidecount=1, loadbeam,
  \beaminsidelast]{2}{6}%
\psset{savebeam=false, loadbeam, beamnodealign=vector}%
\drawwidebeam[fillcolor=orange]{6}{5}%
\drawwidebeam[fillcolor=green, n=1.2, beampathskip=1, savebeam=2]{6}{5}(D1)
\drawwidebeam[fillcolor=red, beampathskip=1, n=1.4, savebeam=3]{6}{5}(D1)
\drawwidebeam[fillcolor=red!50!black, beampathskip=1, n=1.57, savebeam=4]{6}{5}(D1)
\multido{\i=2+1}{3}{%
  \backlayer{%
    \pnodex(!\oeBeamCenter{\i}){Det\i}
    \nodexn{(Det\i)-(!\oeBeamVecMedian{\i})}{Det\i'}}
  \optdetector[detsize=0.6 0.5](Det\i')(Det\i)}
\end{optexp}
\pscircle[style=Glass]([offset=-0.5]\oenableIfc{2}{P}){0.5}
\rput[l]([offset=-0.5,Xnodesep=0.7]\oenableIfc{2}{P}){WGM}
\end{pspicture}
```

Ex. 11.28

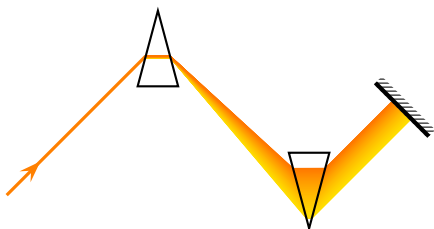


```

\begin{pspicture}(2.2,2.2)
\definecolor[ps]{bl}{rgb}{tx@addDict begin Red Green Blue end}%
\pnode(0,0.5){In}\pnode(2,0.5){G1}
\pnode([Xnodesep=-1.5, offset=1.3]G1){G2}\pnode(G1|G2){MG}
\begin{optexp}
\optgrating[angle=40, gratingalign=c](In)(G1)(G2)
\optgrating[reverse, angle=40, gratingalign=c](G1)(G2)(MG)
\mirror[mirrorwidth=1, mirrortype=extended](G2)(MG)(G2)
\drawbeam[linecolor=red](In){1}
\multido{\i=0+1}{40}{%
\pstVerb{\i\space 650 400 sub 39 div mul 400 add
tx@addDict begin wavelengthToRGB end }%
\drawbeam[linecolor=bl, beamangle=\i\space 16 sub 0.2 mul]{1-2}%
\drawbeam[linecolor=bl, loadbeampoints]{2-3}
}%
\end{optexp}
\end{pspicture}

```

Ex. 11.29

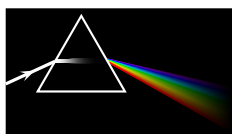


```

\begin{pspicture}(5.6,3)
\pnodes(0,0.5){S1}(2,2.5){S2}(3,1.5){S3}(4,0.5){S4}(4.5,1){S5}
\psset{prismangle=30, mirrortype=extended}
\definecolor[ps]{bl}{rgb}{tx@addDict begin Red Green Blue end}%
\begin{optexp}
\optprism(S1)(S2)(S3)\optprism(S3)(S4)(S5)
\mirror[compshift=-0.07](\oenableIn{2})([offset=1,Xnodesep=1]\oenableIn{2})(\oenableIn{2})
\drawbeam[linecolor=orange, ArrowInside=->, ArrowInsidePos=0.2, ↘
arrowscale=1.5](S1){1}
\addtopsstyle{Beam}{linecolor=bl, linewidth=0.35\pslinewidth, beampathskip=1}
\multido{\r=0+0.5}{120}{%
\pstVerb{\r\space 620 590 sub 59 div mul 590 add
tx@addDict begin wavelengthToRGB end }%
\drawbeam[n=3.42 -0.002 \r\space mul add, beampos=\r\space 60 sub 0.0002 ↘
mul](S1){1-3}
}%
\end{optexp}
\end{pspicture}

```

Ex. 11.30

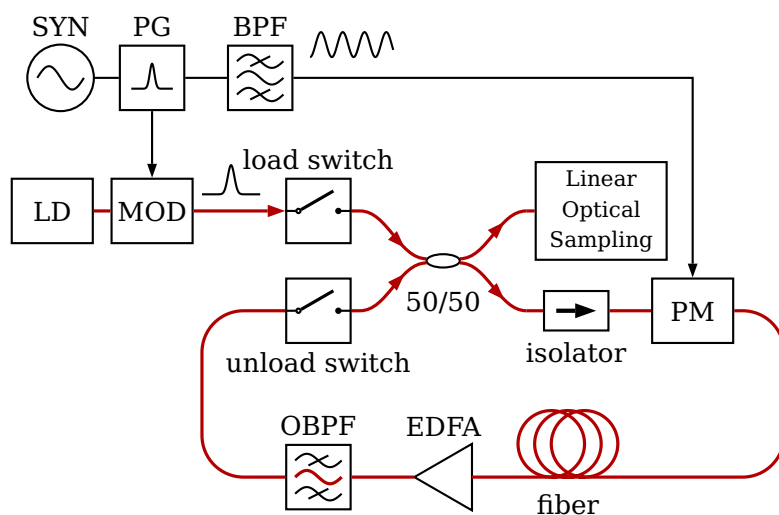


```

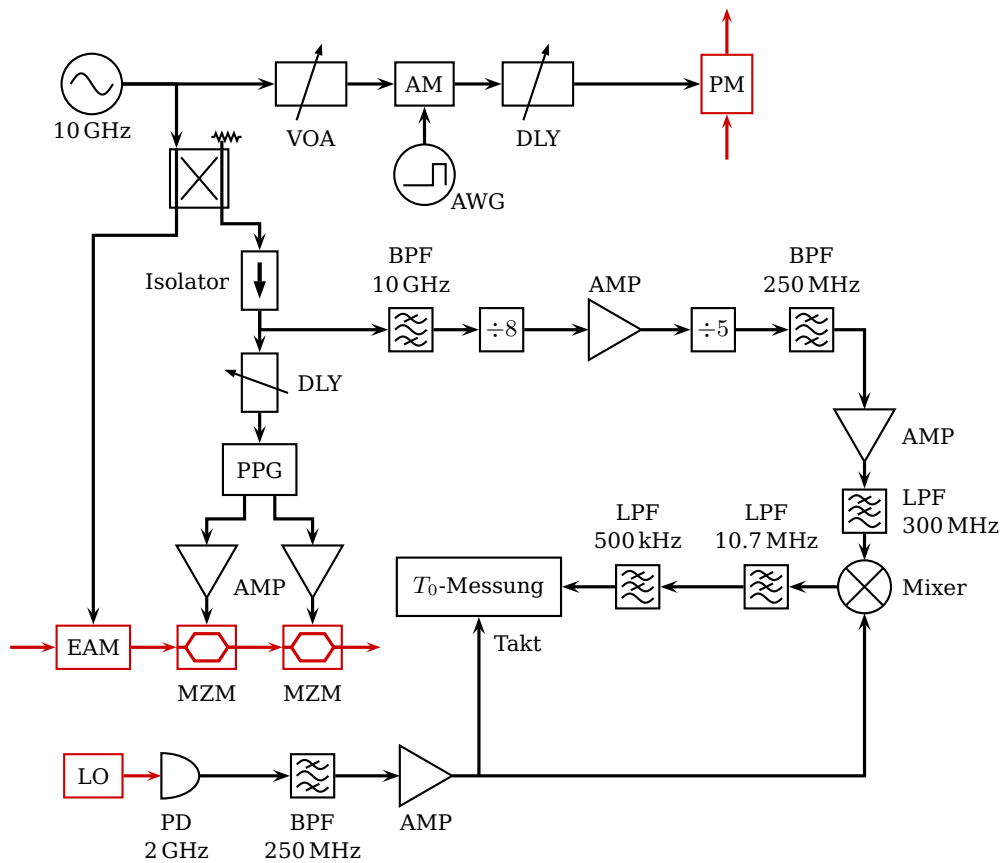
\begin{pspicture}(3,1.7)
\pnodes(-1,0){A}(1,1){B}(3,0){C}
\psframe*(0,0)(3,1.7)
\begin{optexp}
\optplane(0,0.7)
\optprism[prismalign=center, linecolor=white](A)(B)(C)
\optplane(C)
\addtopsstyle{Beam}{linecolor=white}
\drawbeam[ArrowInside=->]{1-2}
\definecolor[ps]{bl}{rgb}{%
tx@addDict begin Red Green Blue end}%
\addtopsstyle{Beam}{linestyle=fade, beaminsidelast, fadeto=black}
\multido{\i=0+1}{60}{%
\drawbeam[n=1.5 \i\space 0.003 mul add, linecolor=white, beampathskip=1, 2
fadefuncname=gauss]{1-2}
\pstVerb{\i\space 650 400 sub 59 div mul 400 add
tx@addDict begin wavelengthToRGB end }%
\drawbeam[n=1.5 \i\space 0.003 mul add, linecolor=bl, beampathskip=2, 2
linewidth=0.3\pslinewidth]{-}%
}%
\end{optexp}
\end{pspicture}

```

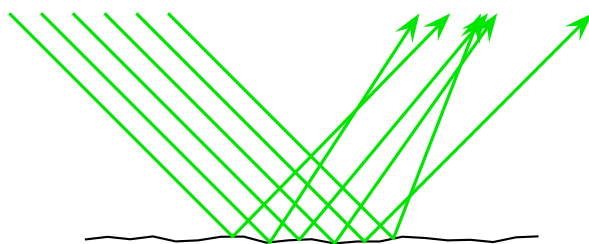
Ex. 11.31



Ex. 11.34



Ex. 11.35



```

\begin{pspicture}(10,3)
\makeatletter
\newOptexpTripole{roughsurface}
\def\roughsurface@nodes{%
  \pssavepath[linestyle=none, arrows=-,ArrowInside=-]{\oename@Path{A}}{%
    \code{0 srand}
    \moveto(!-3 Rand 0.5 sub 0.1 mul)
    \multido{\r=-2.7+0.3}{20}{%
      \lineto(!\r\space Rand 0.5 sub 0.1 mul)}}%
}%
\newOptexpComp{%
  {0 0} tx@IntersectDict /\PIT@name{\oename@Path{A}} get 0 0 refl {PathIfc}
  1 }%
}%
\def\roughsurface@comp{%
  \pstracecurve{\oename@Path{A}}
}%
\makeatother
\optplane[angle=90](1,3)
\roughsurface(0,3)(5,0)(10,3)
\addtopsstyle{Beam}{beamalign=abs, beamangle=-45, arrows=->, arrowscale=2}
\multido{\r=0+0.3}{6}{%
  \drawbeam[beampos=\r]{1}{2}{1}}
\end{pspicture}

```

12. Additional information

12.1. Internal component structure

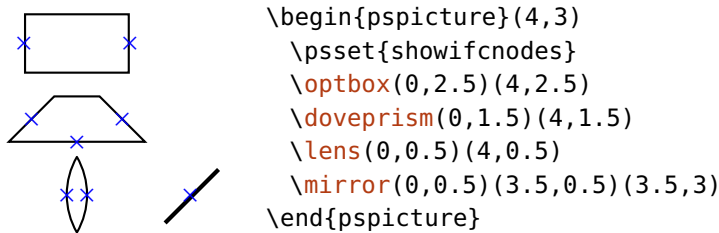
`showifcnodes=true, false`

default: false

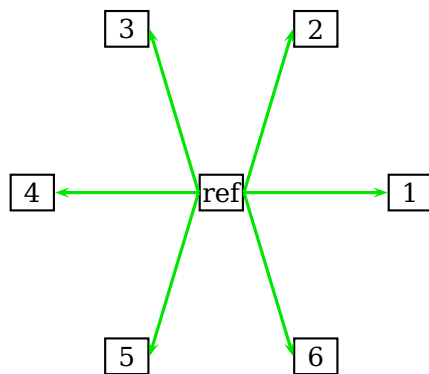
`IfcNodeStyle<psstyle>`

default: `dotstyle=x, dotscale=1.5, linecolor=blue`

Every component consists internally of interfaces which are defined by a node on the optical axis, a plane vector or a curvature radius, and by its optical characteristics (reflective or transmittive, or both e.g. for `\beamsplitter`). The interface nodes can be visualized for each component with the `showifcnodes` parameter.



`\draw*beam` calculates the distance between the interface nodes of two components and connects the two nearest to each other.

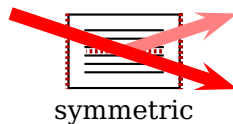


```
\begin{pspicture}(-3,-3)(3,3)
\psset{optboxsize=0.6 0.5, labeloffset=0}
\multido{\i=0+60,\ii=1+1}{6}{%
\pnode(2.5;\i){A}
\optbox([Xnodesep=-1]A)([Xnodesep=1]A){\ii}
}
\optbox[comname=ref](-1,0)(1,0){ref}
\addtopsstyle{Beam}{arrows=->}
\multido{\i=1+1}{6}{\drawbeam{ref}{\i}}
\end{pspicture}
```

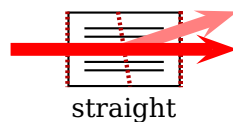
`showinterfaces=true, false`

default: false

Visualizes all interfaces of a component.



```
\begin{pspicture}(3,1.5)
\psset[optexp]{showinterfaces, aomgratingcount=6}
\addtopsstyle{IfcStyle}{linewidth=2\pslinewidth}
\pnodes(0,1.5){A}(3,0.5){B}(3,1.5){C}
\optaom[aomalign=symmetric](A)(B)(C){symmetric}
\addtopsstyle{Beam}{arrows=->, linewidth=4\pslinewidth}
\drawbeam[linecolor=red!50](A){}(C)
\drawbeam[linecolor=red](A){}(B)
\end{pspicture}
```



```
\begin{pspicture}(3,1.5)
\psset[optexp]{showinterfaces, aomgratingcount=6}
\addtopsstyle{IfcStyle}{linewidth=2\pslinewidth}
\pnodes(0,1){A}(3,1){B}(3,1.5){C}
\optaom[aomalign=straight](A)(B)(C){straight}
\addtopsstyle{Beam}{arrows=->, linewidth=4\pslinewidth}
\drawbeam[linecolor=red!50](A){}(C)
\drawbeam[linecolor=red](A){}(B)
\end{pspicture}
```

`IfcStyle`(*psstyle*)

default: `linestyle=dashed, dash=1pt 1pt,`
`linecolor=red!70!black`

Determines the appearance of the interfaces, which are shown with `showinterfaces`.

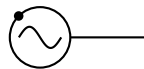
12.2. Overview of special nodes

In this section you find overviews of all special nodes which each component provides. The possible positions for external and rotation reference nodes, and the interface nodes.

12.2.1. External and rotation reference nodes

Here, you find all possible external (Sec. 8.5) and rotation reference nodes (Sec. 8.7) for all components. Displayed are all possible combinations of t, c, or b with l, c, or r. Not all components support all combinations, e.g. a plain `\mirror` ignores t and b and an `\optplate` ignores l and r.

Instead of these values also fractional parts can be used: t corresponds to $+1 \cdot \Delta y$, b corresponds to $-1 \cdot \Delta y$, l corresponds to $-1 \cdot \Delta x$, and r corresponds to $+1 \cdot \Delta x$. This can be especially useful for round outer shapes, as the following examples shows.

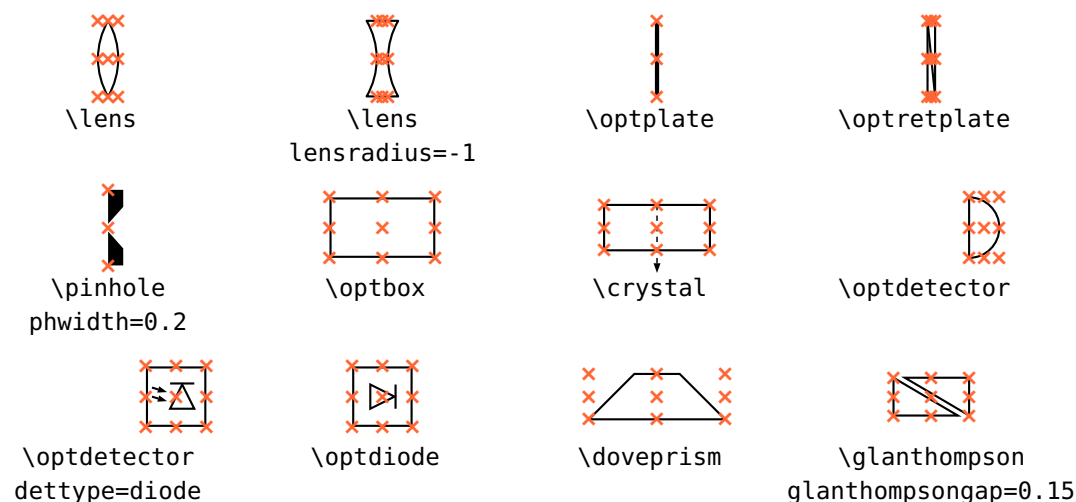


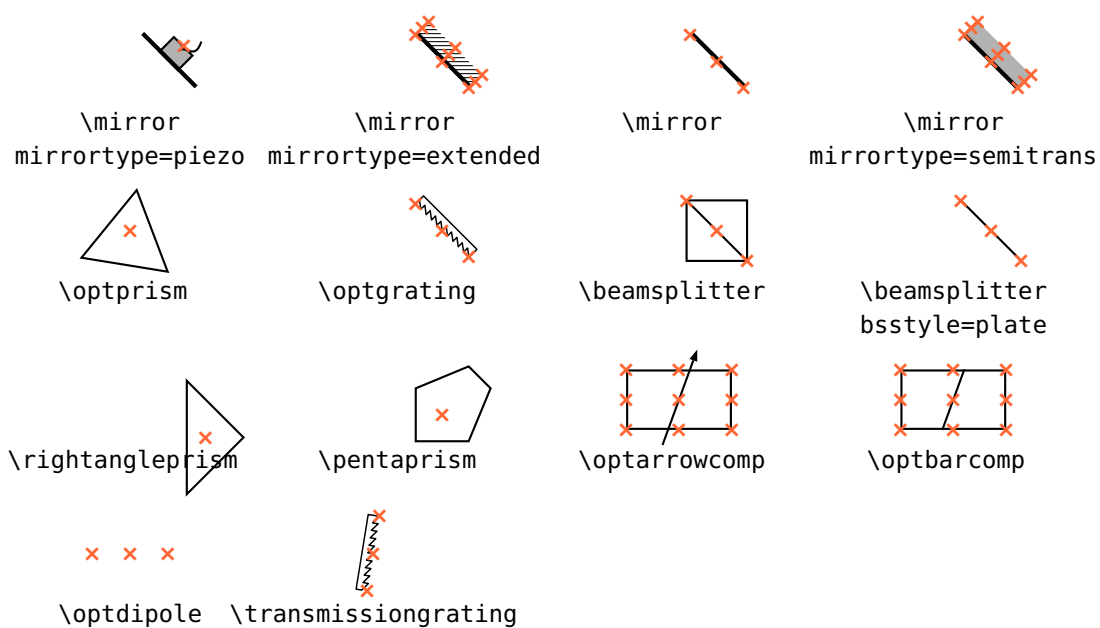
```

\begin{pspicture}(2,1)
  \elecsize[extnode={-0.707,0.707}](1,0.5)(2,0.5)
  \psdot(\oextnode)
\end{pspicture}

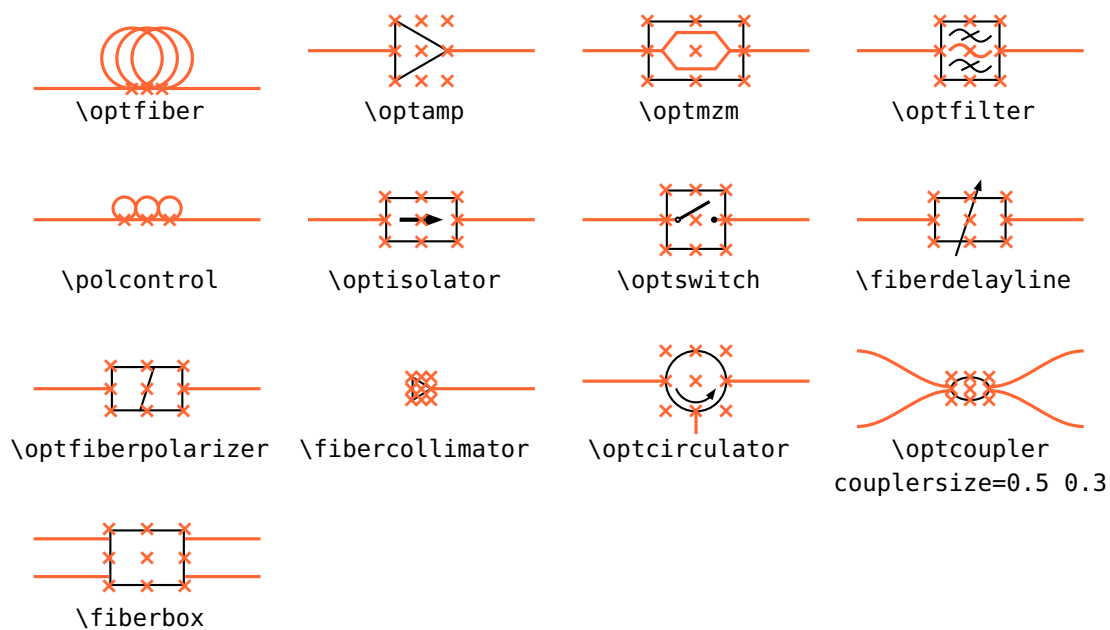
```

Free-ray components





Fiber components



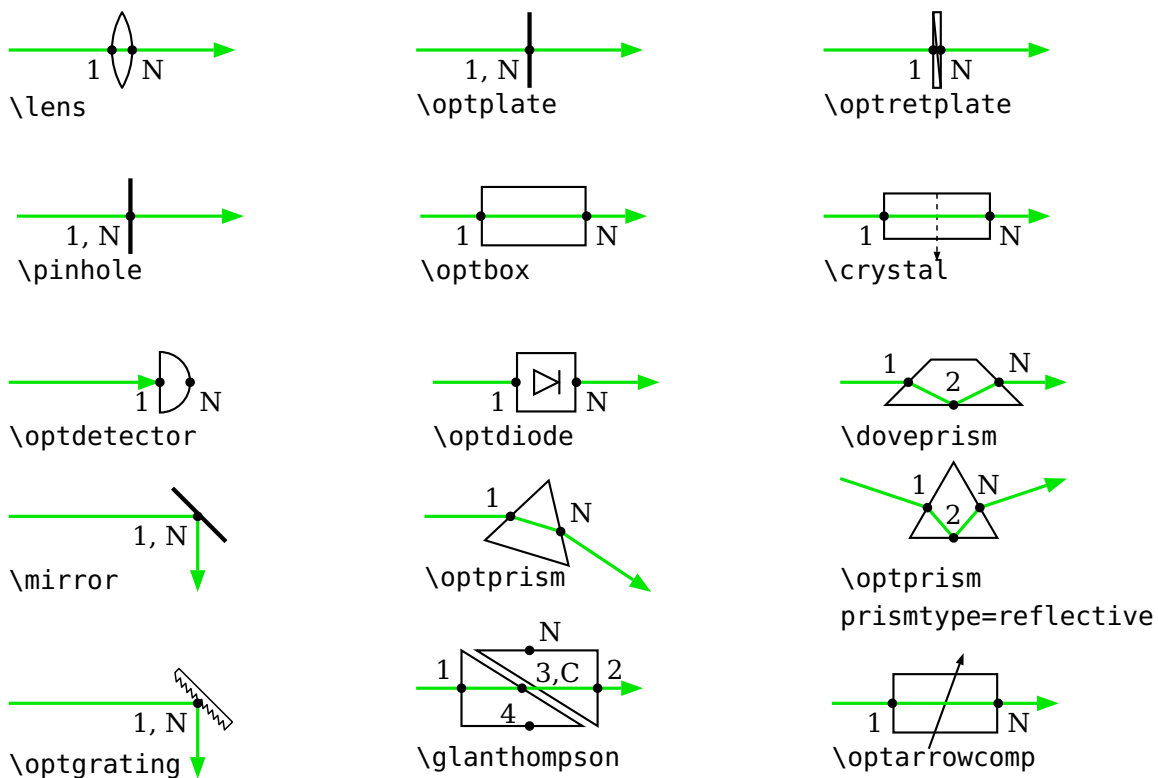
Electrical components

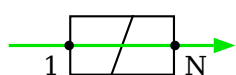


12.2.2. Interface nodes

Here, all available components and their interface nodes are listed. If a node is labeled as "1, N", it means that both nodes are equal.

Free-ray components

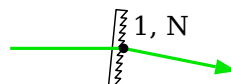




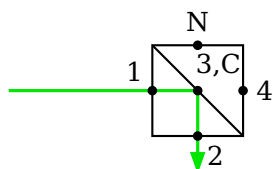
\optbarcomp



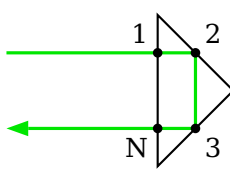
\optdipole



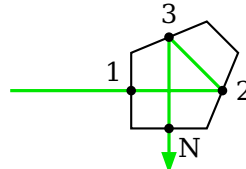
\transmissiongrating



\beamsplitter

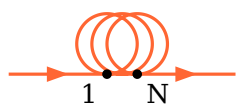


\rightangleprism



\pentaprism

Fiber components



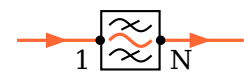
\optfiber



\optamp



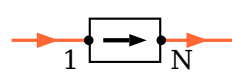
\optmzm



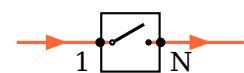
\optfilter



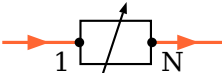
\polcontrol



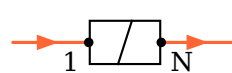
\optisolator



\optswitch



\fiberdelayline



\optfiberpolarizer



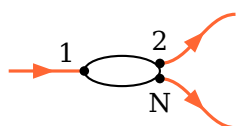
\fibercollimator



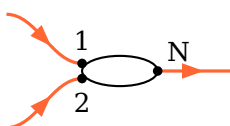
\optcirculator



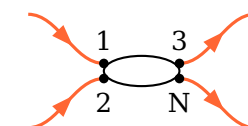
\fiberbox



\wdmsplitter

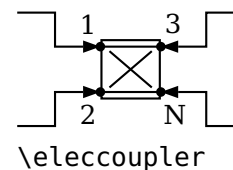
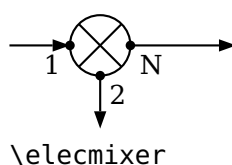
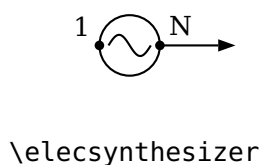


\wdmcoupler



\optcoupler

Electrical components



12.3. Backward compatibility

In this section you find a comprehensive listing of the behaviour which changed in a version compared to the previous one and which is not backward compatible. Here you find also information on what you must note when upgrading and how you must change old documents to work correctly with the newer version.

12.3.1. Version 4.3a

`prismtype` The option `prismtype` was still not correctly implemented, the automatic alignment was somewhat arbitrary. This should be finally fixed, an example is shown for option `prismalign`.

`\optprism` The positioning of the label was not consistent with other components, the reference point was not the center point and did also depend on the component rotation. This is fixed now, but could mean little corrections for existing images.

12.3.2. Version 4.3

`\rightangleprism` The alignment of the `\rightangleprism` and the positioning of the label was wrong. This became apparent only if the input and output vectors had very different lengths.

12.3.3. Version 4.2

`prismtype` The option `prismtype`, introduced in version 4.1, always needed an additional angle rotation of 180°. This is not necessary anymore, the component behavior was corrected, but this may require adjustment of drawings created with version 4.1.

12.3.4. Version 4.0

The option `conn` and the old, buggy code for beam connections was finally removed. To update older documents, see Sec. 12.3.6. With this change also option `connjoin` became useless.

The appearance of `\optfiber` changed slightly, because the old implementation was buggy and the separation between the loops depended on the number of loops (see example below). Note, that to achieve a similar default look, the default value for `fiberloopsep` was changed from 0.3 to 0.2. Therefore, the interface nodes are a bit closer together.



The macro `\newOptexpDipoleNoLabel` was removed and can be replaced by `\newOptexpDipole` without any further changes.

The old syntax of defining own components with `\mycomp@iii` is not supported anymore.

12.3.5. Version 3.3

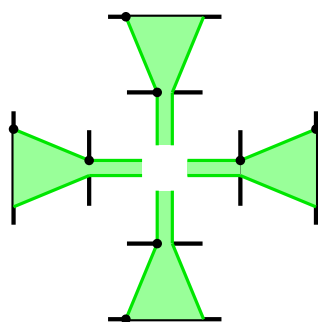
`\opttripole` Changed the reference angle for the label of the `\opttripole` to be consistent with the other tripoles.

Beam nodes

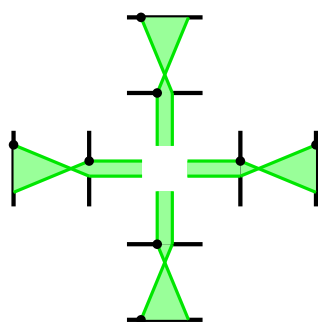
Before version 3.3 the beam nodes `\oenodeBeamUp` and `\oenodeBeamLow` were not swapped correctly and the orientation of “upper” and “lower” was only relative to the beam direction (equivalent to `extnodealign=relative`).

This could lead to problems when using `loadbeampoints` with `beamdiv`. In that case the sign of `beamdiv` was not unambiguously associated with contracting or expanding beam.

As of version 3.3 the beam nodes are arranged according to `extnodealign=absolute` and the association of negative `beamdiv` with a contracting and positive `beamdiv` with an expanding beam is always valid. In the following two examples all `\oenodeBeamUp` are marked.



```
\begin{pspicture}(-2,-2)(2,2)
\addtopsstyle{Beam}{fillstyle=solid, fillcolor=green!40}
\multido{\i=0+90,\ia=1+2,\ib=2+2}{4}{%
\optplate[position=end](0.3;\i)(1;\i)
\optplate[position=end, plateheight=1.5](1;\i)(2;\i)
\drawwidebeam[beamwidth=0.2](0.3;\i){\ia}
\node(\oenodeBeamUp){Up}
\drawwidebeam[loadbeampoints, beamdiv=45]{\ia-\ib}
\psdot(Up)\psdot(\oenodeBeamUp)}
}
\end{pspicture}
```



```
\begin{pspicture}(-2,-2)(2,2)
\addtopsstyle{Beam}{fillstyle=solid, fillcolor=green!40}
\multido{\i=0+90,\ia=1+2,\ib=2+2}{4}{%
\optplate[position=end](0.3;\i)(1;\i)
\optplate[position=end](1;\i)(2;\i)
\drawwidebeam[beamwidth=0.2](0.3;\i){\ia}
\node(\oenodeBeamUp){Up}
\drawwidebeam[loadbeampoints, beamdiv=-45]{\ia-\ib}
\psdot(Up)\psdot(\oenodeBeamUp)}
}
\end{pspicture}
```

12.3.6. Version 3.0

The changes in version 3.0 compared to version 2.1 which are not backward compatible.

General parameters

`namingscheme=old, new`

default: new

In version 2.1 special component nodes had to be accessed by their explicit name. If you need the old naming scheme, because you accessed internal nodes directly, then you must set `namingscheme=old`. Since version 3.0 explicit macros are provided to access all special component nodes, so that the actual naming scheme does not matter, see Sec. 8.

`endbox` These options now affect all components.
`angle`
`rotateref`

extnode Some components now provide more possible positions for **extnode**. **\optdetector** had only one possible external node, which was accessible for any value of **extnode**, now this node can be accessed only with **extnode=r**. All other values give different node positions.

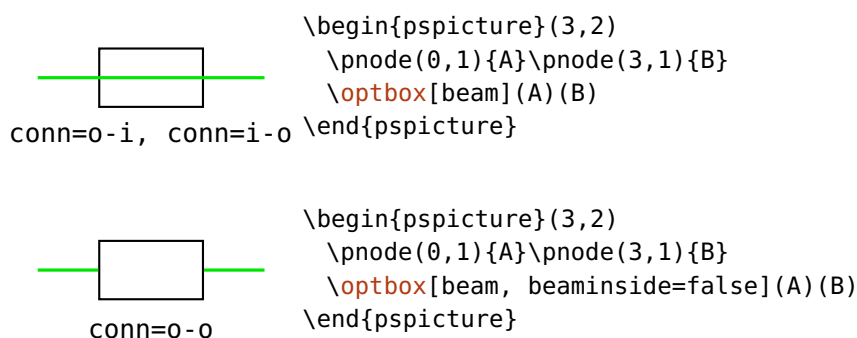
\optbox An **\optbox** does always have the input and output nodes at the respective interfaces, whereas in version 2.1 the input and output nodes were at the same position for **endbox=true**.

\fibercollimator The **\fibercollimator** has only a fiber connection by default, the beam must be drawn manually or with **beam** option.

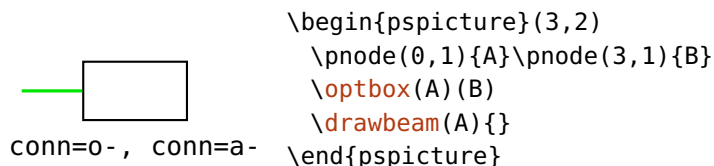
Beam connections


In version 2.1, beam connections could be internal, that is the connections were specified together with the component (**beam** or **conn**), or external with an additional call of **\drawbeam**. The main reason for internal connections was the aesthetic aspect of drawing the beam behind the component, but also convenience played a role. Drawing the beam behind the component is possible with layering (Sec. 9.9) since 3.0.

Some of the variants of internal beam connections can be used also in 3.0:



Other internal connections need an additional **\drawbeam** call, and if you want to move the beam behind the component, the code must be wrapped in a **optexp** environment. Except for this, external connections are equivalent:






conn=-o, conn=-b

```

\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \optbox(A)(B)
  \drawbeam{}(B)
\end{pspicture}

```




conn=i-, A-

```

\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \begin{optexp}
    \optbox(A)(B)
    \drawbeam[beaminsidelast](A){}
  \end{optexp}
\end{pspicture}

```



conn=-i, conn=-B

```

\begin{pspicture}(3,2)
  \pnode(0,1){A}\pnode(3,1){B}
  \begin{optexp}
    \optbox(A)(B)
    \drawbeam[beaminsidfirst]{}(B)
  \end{optexp}
\end{pspicture}

```

Documentation index

A

abspos, 9, 10, 13–17, 22, **22**, 38, 73, 80, 82, 89–91, 93, 96, 97, 102–104, 106, 107, 111, 123, 128–132, 134, 136–138, 141–145, 147, 148

addtoBeam, **98**, **98**

addtoFiber, **111**, 112

addtoFiberIn, 117, 132, 133

addtoFiberIn1, 111, 117, 132, 133

addtoFiberIn2, 111, 117, 122, 132, 133

addtoFiberOut, 117, 132, 133

addtoFiberOut1, 117, 118, 133

addtoFiberOut2, 117, 133

addtoOptComp, 25, **25**, 26

\addtopsstyle, 24, 117

addtoWire, 113, **113**

addtoWireIn, 117

addtoWireIn1, 117

addtoWireIn2, 117

addtoWireOut, 117

addtoWireOut1, 117

addtoWireOut2, 117

align, **60**

allowbeaminside, 35, 69, **94**, 95, 142

angle, **22**, 23, 37, 40, 46, 65, 72, 74, 78, 79, 81–83, 92, 93, 97, 108–110, 128, 129, 134, 138–140, 143, 149, 153

aomalign, 47, **47**, 48, 155

aomcomp, 47, **48**

aomgratingcount, 47, 155

aomheight, 47, **47**

aomreflalign, 48, **48**, 49, 50

aomsize, **47**, 49, 50

aomwidth, 47, **47**

arrowcompangle, **31**

arrowcompheight, [31](#), [31](#)
arrowcompshape, [31](#), [31](#)
arrowcompsize, [31](#), [132](#)
ArrowCompStyle, [31](#), [32](#), [57](#)
arrowcompwidth, [31](#), [31](#)
ArrowInside, [98](#), [99](#)
ArrowInsideMaxLength, [99](#), [99](#)
ArrowInsideMinLength, [98](#), [98](#), [99](#), [139](#), [146](#)

B

\backlayer, [82](#), [119](#), [148](#)
barcompangle, [32](#), [32](#), [33](#)
barcompheight, [32](#), [32](#)
barcompshape, [32](#), [32](#), [33](#)
barcompsize, [32](#)
BarCompStyle, [32](#), [33](#)
barcompwidth, [32](#), [32](#)
Beam, [14](#), [15](#), [17](#), [37](#), [42](#), [46–50](#), [69](#), [80](#), [82](#), [88–90](#), [92](#), [96–98](#), [98](#), [99](#), [101](#), [102](#),
[104](#), [105](#), [123](#), [128–131](#), [134–136](#), [138–142](#), [146](#), [148–150](#), [153](#), [155](#),
[162](#)
beam, [21](#), [24–30](#), [34–41](#), [43–46](#), [50–52](#), [69](#), [70](#), [72](#), [98](#), [120](#), [122](#), [163](#)
beamalign, [33](#), [42](#), [81](#), [82](#), [90–92](#), [92](#), [127](#), [128](#), [141](#), [145](#), [153](#)
beamangle, [11](#), [33](#), [46](#), [87](#), [88](#), [90](#), [91](#), [91](#), [92](#), [103](#), [105](#), [138](#), [139](#), [149](#), [153](#)
beamdiffractionorder, [49](#), [49](#), [50](#)
beamdiv, [33](#), [92](#), [101](#), [101](#), [103](#), [104](#), [123](#), [128](#), [130](#), [138](#), [139](#), [161](#), [162](#)
beaminside, [11](#), [93](#), [94–97](#), [121](#), [125–127](#), [138](#), [163](#)
beaminsidefirst, [80](#), [91](#), [94](#), [94](#), [95](#), [106](#), [131](#), [134](#), [136](#), [142](#), [164](#)
beaminsidelast, [79](#), [81](#), [94](#), [94](#), [95](#), [103](#), [131](#), [148](#), [164](#)
beammode, [49](#), [96](#)
beamnodealign, [82](#), [96](#), [97](#), [148](#)
beampathcount, [93](#), [93](#), [148](#)
beampathskip, [82](#), [92](#), [93](#), [148–150](#)
beamos, [10](#), [11](#), [37](#), [40](#), [42](#), [43](#), [81](#), [87](#), [89](#), [91](#), [91](#), [94](#), [96](#), [97](#), [100–103](#), [106](#),
[107](#), [141](#), [145](#), [149](#), [153](#)
\beamsplitter, [17](#), [20](#), [43](#), [43](#), [46](#), [85](#), [96](#), [106](#), [131](#), [134–136](#), [138](#), [145](#), [146](#),
[154](#)
beamwidth, [11](#), [14](#), [15](#), [17](#), [33](#), [42](#), [45](#), [47–51](#), [69](#), [80](#), [82](#), [93](#), [96](#), [97](#), [99](#), [100](#),
[101](#), [102](#), [104](#), [105](#), [121–123](#), [128](#), [130](#), [131](#), [134](#), [135](#), [138](#), [139](#), [148](#),
[162](#)
bssize, [43](#), [85](#), [136](#), [145](#), [146](#)

bsstyle, 17, 43, **43**, 146

C

caxisinv, **34**

caxislength, 34, **34**

compname, 17, 18, 73, **73**, 84–86, 89, 93, 96, 107, 111, 113, 129, 132, 134, 135, 138, 142–144, 146, 148, 155

compoffset, 23, **23**, 24, 146

components

- \beamsplitter, 17, 20, 43, **43**, 46, 85, 96, 106, 131, 134–136, 138, 145, 146, 154

- \crystal, 34, **34**, 35, 122, 136, 142

- \doveprism, 36, **36**, 86, 87, 89, 135, 154

- \eleccoupler, 64, **64**, 65, 116

- \elecmmixer, **66**, 67, 109, 116

- \elecsynthesizer, 24, 65, **65**, 66, 114–116, 118, 132, 143, 144, 156

- \fiberbox, **61**, 62, 63

- \fibercollimator, 68, 69, **69**, 70, 163

- \fiberdelayline, 31, 56, **56**, 103, 118

- \glanthompson, 36, **36**, 37, 93, 143

- \lens, **27**

- \mirror, 10, 17, 23–26, 38, **38**, 39–41, 43, 72, 76, 77, 80, 88, 91, 92, 99, 101, 102, 106, 128, 129, 131, 135, 136, 139–142, 145, 146, 148, 149, 154, 156

- \oapmirror, 42, **42**, 43

- \optamp, 54, **54**, 111, 112, 137, 143, 144, 147

- \optaom, 46, **46**, 47–50, 155

- \optarrowcomp, 31, **31**, 32, 132

- \optbarcomp, 32, **32**, 33

- \optbox, 10–17, 19–23, 30, **30**, 63, 73–77, 79, 81–85, 91, 92, 94, 95, 101, 108–112, 114–116, 118, 119, 128–133, 135–138, 140–148, 154, 155, 163, 164

- \optcirculator, 57, **57**, 58, 67, 75, 109, 114, 115, 144

- \optcoupler, **58**, 59, 60, 65, 75, 111, 112, 115, 132, 137, 138, 147

- \optdetector, 13, 16, 17, 68, 70, **70**, 71, 75, 82, 114, 129, 132, 134, 140, 142, 145–148, 163

- \optdiode, 35, **35**, 94

- \optdipole, 24, 120, **120**, 121, 135, 145

- \optfiber, 53, **53**, 54, 111, 112, 128, 132, 133, 137, 144, 147, 161

- \optfiberpolarizer, 32, 57, **57**, 133

- \optfilter, 24, 53, 68, **68**, 69, 94, 137

- \optgrating, 25, 44, **44**, 45–47, 90, 105, 129, 138, 139, 149
- \optgrid, **44**
- \optisolator, 55, **55**, 56, 133, 144
- \optmzm, 15–17, 53, 55, **55**, 114–116, 137, 143, 144
- \optplane, 33, 36, 37, 42, 50, 82, 87, 89, 90, 93, 97, 103, 106, 107, 128, 134, 138, 143, 150, 153
- \optplate, 13, 29, **29**, 90–92, 101–103, 106, 107, 123, 130, 138, 140, 142, 145, 146, 156, 162
- \optprism, 50, **50**, 51, 82, 86, 87, 90, 93, 96, 99, 103, 148–150
- \optretplate, 13, 29, **29**, 94, 129–131, 135, 136
- \optsource, 33, **33**
- \optswitch, 56, **56**, 137, 147
- \opttripole, 120, **120**, 121, 122, 131, 141, 161
- \parabolicmirror, 41, **41**, 42
- \pentaprism, 52, **52**, 78, 94, 135
- \pinhole, 30, **30**, 90, 130, 131, 135, 136, 138, 139, 141, 146
- \polarization, 37, **37**, 38, 82, 93, 130, 136, 143
- \polcontrol, 55, **55**, 133, 137, 144
- \rightangleprism, 51, **51**, 52, 160
- \transmissiongrating, **45**, 46
- \wdmcoupler, 59, **59**, 60, 61, 122, 133, 140, 144
- \wdmsplitter, 16, 17, 59, **59**, 60, 61, 115, 118, 133, 138, 140, 144, 147
- compshift, 17, 23, **23**, 74, 83, 128, 132, 133, 138, 144, 149
- conn, 161, 163
- connjoin, 161
- coupleralign, 60, **60**, 61, 75, 112, 118, 122, 133, 144
- couplersep, 59, **59**, 65, 75
- couplersize, 59, **59**, 65, 75
- couplertype, 59, **59**, 60, 65, 75, 137, 140
- \crystal, 34, **34**, 35, 122, 136, 142
- CrystalCaxis, 35, **35**
- crystalheight, 34, **34**
- CrystalLamp, 35, **35**
- crystalsize, 34, **34**, 136
- crystalwidth, 34, **34**

D

- DetectorStyle, 71, **71**
- detsize, **70**, 82, 140, 142, 146–148
- detype, 13, 17, 71, **71**, 129, 132

diffractionorders, [48](#), [49](#), [50](#)
\doveprism, [36](#), [36](#), [86](#), [87](#), [89](#), [135](#), [154](#)
doveprismsize, [36](#), [36](#), [87](#)
\drawbeam, [7](#), [10](#), [11](#), [13](#), [20](#), [33](#), [36](#), [37](#), [39](#), [40](#), [42](#), [43](#), [46](#), [78](#), [79](#), [81](#), [82](#),
[84–99](#), [102–107](#), [121](#), [124–127](#), [129](#), [130](#), [134](#), [136](#), [140–143](#), [145](#), [146](#),
[149](#), [150](#), [153–155](#), [163](#), [164](#)
\drawfiber, [12](#), [16](#), [17](#), [61](#), [107–112](#), [114](#), [132](#), [137](#), [138](#), [140](#), [143](#), [144](#), [147](#)
\drawwidebeam, [11](#), [14](#), [15](#), [17](#), [33](#), [42](#), [45](#), [47–51](#), [69](#), [79](#), [80](#), [82](#), [86](#), [88](#), [92](#), [93](#),
[96](#), [97](#), [99–105](#), [121–123](#), [128](#), [130](#), [131](#), [134](#), [135](#), [138](#), [139](#), [148](#), [154](#),
[162](#)
\drawwire, [113](#), [114](#), [132](#), [143](#), [144](#), [147](#)

E

\eleccoupler, [64](#), [64](#), [65](#), [116](#)
eleccouplerinput, [64](#), [64](#)
eleccouplersep, [64](#)
eleccouplersize, [64](#), [64](#)
eleccouplertype, [64](#), [64](#)
\elecmixer, [66](#), [67](#), [109](#), [116](#)
elecmixersize, [67](#)
\elecsynthesizer, [24](#), [65](#), [65](#), [66](#), [114–116](#), [118](#), [132](#), [143](#), [144](#), [156](#)
endbox, [22](#), [163](#)
environ, [6](#)
environments
 – optexp, [11](#), [18](#), [45](#), [118](#), [163](#)
 – pspicture, [73](#), [80](#)
ExtendedMirror, [24](#), [39](#), [41](#), [41](#), [76](#), [77](#), [131](#), [139](#)
extnode, [40](#), [75](#), [76–78](#), [83](#), [124](#), [126](#), [127](#), [132](#), [134](#), [138](#), [143](#), [144](#), [147](#), [156](#),
[163](#)
extnodealign, [76](#), [76](#), [77](#), [134](#), [161](#)
extnodes, [77](#), [77](#)

F

fade, [99](#), [100](#)
fadefunc, [99](#), [100](#), [100](#)
fadefuncname, [99](#), [100](#), [100](#), [150](#)
fadepoints, [99](#), [99](#), [100](#)
fadeto, [99](#), [99](#), [100](#), [150](#)
FdlArrow, [57](#)
fdlsize, [56](#), [103](#)

Fiber, 53, 57, 58, 65, 108, 109, 111, **111**, 112, 116, 117, 132, 147
fiber, 15–17, 25, 32, 33, 61, 63, 69, 103, 109, 111, 112, 114, **114**, 115, 118,
128, 132, 137, 138, 140, 143, 144, 147
fiber*, 113
fiberalign, 109, **109**, 110, 113
fiberangleA, 110, **110**, 113
fiberangleB, 110, **110**, 113
\fiberbox, **61**, 62, 63
fiberboxcount, 62, 63, **63**
fiberboxheight, **62**, 63
fiberboxsepin, 62, **63**
fiberboxsepout, 62, 63, **63**
fiberboxsize, 63, **63**
fiberboxwidth, **62**, 63
\fibercollimator, 68, 69, **69**, 70, 163
fibercolsize, **70**
\fiberdelayline, 31, 56, **56**, 103, 118
FiberIn, 117
FiberIn1, 117
FiberIn2, 117
fiberloopradius, 54, **54**, 128
fiberloops, **53**, 54, 133, 137, 147
fiberloopsep, 54, **54**, 161
FiberOut, 117
FiberOut1, 57, 117
FiberOut2, 57, 117
fiberpolsize, **57**
fiberstyle, 12, 65, 112, **112**, 113, 137
fillstyle, 11
filterangle, 69, **69**
filtersize, **68**
FilterStyle, 69, **69**
filtertype, 68, **68**
forcebeaminside, 95, **95**
\frontlayer, 119

G

\glanthompson, 36, **36**, 37, 93, 143
glanthompsongap, **36**, 37, 143
glanthompsonheight, 36, **36**

glanthompsonsize, **36**, 37, 143
glanthompsonwidth, 36, **36**
gratingalign, 45, **45**, 149
gratingcount, 44, **44**
gratingdepth, 44, **44**
gratingheight, **44**
gratinglinewidth, **45**
gratingtype, 44, **44**
gratingwidth, **44**, 139

I

IfcNodeStyle, **154**
IfcStyle, 155, **155**
innercompalign, 24, **24**, 69
innerheight, **30**, 90, 138
innerlabel, **21**, 23, 76, 77, 112, 136, 145
IsolatorArrow, 56, **56**
isolatorsize, **55**

L

label, 15–17, 19, 21, **21**, 34, 35, 53, 54, 63, 76, 77, 111, 120–122, 124–133, 135–138, 140–147
labelalign, 19, **19**, 21, 125–127, 131
labelangle, 10, **19**, 20, 21, 24, 125, 128, 130, 131, 136, 137, 145
labeloffset, 10, 13–15, 17, 19, **19**, 20, 21, 39, 45, 59, 60, 75, 84, 85, 120, 129, 131–133, 136, 138, 141, 145, 148, 155
labelref, 10, 20, **20**, 21, 76
labelstyle, 15–17, **19**, 136, 137, 147
lamp, **34**
lampscale, **35**
\lens, **27**
lens, 9–11, 13–15, 17, 22, 23, 25, 27, 28, **28**, 33, 51, 80, 82, 89, 91, 93, 96, 97, 102–104, 106, 107, 123, 128, 130, 131, 134–136, 138, 141, 142, 146, 148, 154
lensheight, 11, 14, 15, 27, **27**, 136
lensradius, 11, 13–15, 28, **28**, 80, 131
lensradiusleft, 27, **27**, 28
lensradiusright, **27**, 28
lenswidth, 28, **28**
linestyle, 11

loadbeam, 80–82, 96, 105, **105**, 106

loadbeampoints, 104, **104**, 105, 123, 128, 130, 131, 135, 149, 161

M

macros

- \addtopsstyle, 24, 117
- \backlayer, 82, 119, 148
- \drawbeam, 7, 10, 11, 13, 20, 33, 36, 37, 39, 40, 42, 43, 46, 78, 79, 81, 82, 84–99, 102–107, 121, 124–127, 129, 130, 134, 136, 140–143, 145, 146, 149, 150, 153–155, 163, 164
- \drawfiber, 12, 16, 17, 61, 107–112, 114, 132, 137, 138, 140, 143, 144, 147
- \drawwidebeam, 11, 14, 15, 17, 33, 42, 45, 47–51, 69, 79, 80, 82, 86, 88, 92, 93, 96, 97, 99–105, 121–123, 128, 130, 131, 134, 135, 138, 139, 148, 154, 162
- \drawwire, 113, 114, 132, 143, 144, 147
- \frontlayer, 119
- \nccurve, 12, 107, 112
- \ncput, 21
- \newOptexpComp, 125
- \newOptexpDipole, 123–127, 161
- \newOptexpDipoleNoLabel, 161
- \newOptexpElecDipole, 123, 124
- \newOptexpFiberDipole, 123, 124
- \newOptexpTripole, 123, 125, 127, 153
- \newpsobject, 25, 121, 122
- \newpsstyle, 8, 24
- \nodexn, 82
- \oeBeamCenter, 82, 148
- \oeBeamVec, 81, 82, 93, 143
- \oeBeamVecLow, 82
- \oeBeamVecMedian, 82, 97, 148
- \oeBeamVecUp, 82
- \oenode, 49, 77
- \oenodeBeam, 79–82, 93, 143
- \oenodeBeam*, 80
- \oenodeBeamLow, 82, 161
- \oenodeBeamUp, 82, 161, 162
- \oenodeCenter, 73, 75, 146
- \oenodeExt, 76, 77, 127, 132, 134, 143, 144, 147, 156

- \oenodeIfc, 51, 61, 78, 93, 96, 144, 148
- \oenodeIn, 73, 78, 129, 145, 149
- \oenodeLabel, 142, 146
- \oenodeOut, 73, 78, 129, 142, 145, 148
- \oenodeRefA, 21, 74, 78, 98
- \oenodeRefB, 21, 74, 98
- \oenodeRotref, 79
- \oenodeTrefA, 74
- \oenodeTrefB, 74
- \psbezier, 69, 70

\mirror, 10, 17, 23–26, 38, **38**, 39–41, 43, 72, 76, 77, 80, 88, 91, 92, 99, 101, 102, 106, 128, 129, 131, 135, 136, 139–142, 145, 146, 148, 149, 154, 156

mirrordepth, 17, 39, 40, **40**, 41, 43, 76, 77

mirrorlinewidth, **38**

mirrorradius, **38**, 39, 91, 139, 145

mirrortype, 17, 24, 39, **39**, 40, 41, 43, 76, 77, 80, 91, 92, 99, 101, 106, 131, 136, 139, 140, 142, 145, 149

mirrorwidth, **38**, 80, 99, 101, 102, 136, 139, 141, 142, 146, 149

multido, 6

N

n, 10, 11, 36, 80–82, 86–88, **88**, 89, 90, 96, 97, 102, 103, 106, 107, 123, 127, 129, 131, 134, 135, 138, 142, 146, 148–150

namingscheme, 72, 162

\nccurve, 12, 107, 112

\ncput, 21

newBeam, **98**

newFiber, **112**, 128

newFiberIn, 117

newFiberIn1, 117

newFiberIn2, 117

newFiberOut, 117

newFiberOut1, 117

newFiberOut2, 117

newOptComp, 25, 26, **26**

\newOptexpComp, 125

\newOptexpDipole, 123–127, 161

\newOptexpDipoleNoLabel, 161

\newOptexpElecDipole, 123, 124

`\newOptexpFiberDipole`, 123, 124
`\newOptexpTripole`, 123, 125, 127, 153
`\newsobject`, 25, 121, 122
`\newsstyle`, 8, 24
`newWire`, 113
`newWireIn`, 117
`newWireIn1`, 117
`newWireIn2`, 117
`newWireOut`, 117
`newWireOut1`, 117
`newWireOut2`, 117
`\nodexn`, 82
`npos`, 21

O

`\oapmirror`, 42, 42, 43
`oapmirroraperture`, 42, 43
`\oeBeamCenter`, 82, 148
`\oeBeamVec`, 81, 82, 93, 143
`\oeBeamVecLow`, 82
`\oeBeamVecMedian`, 82, 97, 148
`\oeBeamVecUp`, 82
`\oenode`, 49, 77
`\oenodeBeam`, 79–82, 93, 143
`\oenodeBeam*`, 80
`\oenodeBeamLow`, 82, 161
`\oenodeBeamUp`, 82, 161, 162
`\oenodeCenter`, 73, 75, 146
`\oenodeExt`, 76, 77, 127, 132, 134, 143, 144, 147, 156
`\oenodeIfc`, 51, 61, 78, 93, 96, 144, 148
`\oenodeIn`, 73, 78, 129, 145, 149
`\oenodeLabel`, 142, 146
`\oenodeOut`, 73, 78, 129, 142, 145, 148
`\oenodeRefA`, 21, 74, 78, 98
`\oenodeRefB`, 21, 74, 98
`\oenodeRotref`, 79
`\oenodeTrefA`, 74
`\oenodeTrefB`, 74
`\optamp`, 54, 54, 111, 112, 137, 143, 144, 147
`optampsize`, 54

`\optaom`, 46, **46**, 47–50, 155
`\optarrowcomp`, 31, **31**, 32, 132
`\optbarcomp`, 32, **32**, 33
`\optbox`, 10–17, 19–23, 30, **30**, 63, 73–77, 79, 81–85, 91, 92, 94, 95, 101, 108–112, 114–116, 118, 119, 128–133, 135–138, 140–148, 154, 155, 163, 164
`optboxheight`, **30**, 31
`optboxsize`, 11, 12, 20–22, **31**, 63, 73, 76, 77, 81, 82, 85, 112, 128, 129, 131, 132, 136, 138, 140, 142–147, 155
`optboxwidth`, 13, 15–17, 20, **30**, 31, 94, 95, 128–130, 133, 135–137, 140, 141, 143, 148
`optcircangle`, **57**, 58, 144
`optcircangleA`, **57**
`optcircangleB`, **57**
`OptCircArrow`, 58, **58**, 144
`optcircsize`, **57**
`\optcirculator`, 57, **57**, 58, 67, 75, 109, 114, 115, 144
`OptComp`, 25, **25**, 26, 69, 71, 139
`\optcoupler`, **58**, 59, 60, 65, 75, 111, 112, 115, 132, 137, 138, 147
`\optdetector`, 13, 16, 17, 68, 70, **70**, 71, 75, 82, 114, 129, 132, 134, 140, 142, 145–148, 163
`\optdiode`, 35, **35**, 94
`optdiodesize`, **35**
`\optdipole`, 24, 120, **120**, 121, 135, 145
`optdipolecomp`, 121, **121**
`optdipolesize`, **120**, 121, 135, 145
`optexp`, 11, 18, 45, 118, 163
`\optfiber`, 53, **53**, 54, 111, 112, 128, 132, 133, 137, 144, 147, 161
`\optfiberpolarizer`, 32, 57, **57**, 133
`\optfilter`, 24, 53, 68, **68**, 69, 94, 137
`\optgrating`, 25, 44, **44**, 45–47, 90, 105, 129, 138, 139, 149
`\optgrid`, 44
`optgridcount`, 45
`optgriddepth`, 45
`optgridheight`, 45
`optgridlinewidth`, 45
`optgridtype`, 45
`optgridwidth`, 45
`optional`, 25, **26**
`OptionalStyle`, 25, **25**, 26

\optisolator, 55, **55**, 56, 133, 144
\optmzm, 15–17, 53, 55, **55**, 114–116, 137, 143, 144
optmzsize, **55**, 143
\optplane, 33, 36, 37, 42, 50, 82, 87, 89, 90, 93, 97, 103, 106, 107, 128, 134, 138, 143, 150, 153
\optplate, 13, 29, **29**, 90–92, 101–103, 106, 107, 123, 130, 138, 140, 142, 145, 146, 156, 162
\optprism, 50, **50**, 51, 82, 86, 87, 90, 93, 96, 99, 103, 148–150
\optretplate, 13, 29, **29**, 94, 129–131, 135, 136
\optsource, 33, **33**
\optswitch, 56, **56**, 137, 147
\opttripole, 120, **120**, 121, 122, 131, 141, 161
opttripolecomp, **121**, 122
outerheight, **30**, 136, 139, 146

P

packages

- environ, 6
- multido, 6
- pst-eucl, 6
- pst-intersect, 6
- pst-node, 6, 21, 112
- pst-optexp, 6, 24, 84, 119
- pst-plot, 6
- pstricks, 6
- pstricks-add, 6, 98

\parabolicmirror, 41, **41**, 42

parameters

- abspos, 9, 10, 13–17, 22, **22**, 38, 73, 80, 82, 89–91, 93, 96, 97, 102–104, 106, 107, 111, 123, 128–132, 134, 136–138, 141–145, 147, 148
- addtoBeam, 98, **98**
- addtoFiber, **111**, 112
- addtoFiberIn, 117, 132, 133
- addtoFiberIn1, 111, 117, 132, 133
- addtoFiberIn2, 111, 117, 122, 132, 133
- addtoFiberOut, 117, 132, 133
- addtoFiberOut1, 117, 118, 133
- addtoFiberOut2, 117, 133
- addtoOptComp, 25, **25**, 26
- addtoWire, 113, **113**

- addtoWireIn, 117
- addtoWireIn1, 117
- addtoWireIn2, 117
- addtoWireOut, 117
- addtoWireOut1, 117
- addtoWireOut2, 117
- align, 60
- allowbeaminside, 35, 69, 94, 95, 142
- angle, 22, 23, 37, 40, 46, 65, 72, 74, 78, 79, 81–83, 92, 93, 97, 108–110, 128, 129, 134, 138–140, 143, 149, 153
- aomalign, 47, 47, 48, 155
- aomcomp, 47, 48
- aomgratingcount, 47, 155
- aomheight, 47, 47
- aomreflalign, 48, 48, 49, 50
- aomsize, 47, 49, 50
- aomwidth, 47, 47
- arrowcompangle, 31
- arrowcompheight, 31, 31
- arrowcompshape, 31, 31
- arrowcompsize, 31, 132
- arrowcompwidth, 31, 31
- ArrowInside, 98, 99
- ArrowInsideMaxLength, 99, 99
- ArrowInsideMinLength, 98, 98, 99, 139, 146
- barcompangle, 32, 32, 33
- barcompheight, 32, 32
- barcompshape, 32, 32, 33
- barcompsize, 32
- barcompwidth, 32, 32
- beam, 21, 24–30, 34–41, 43–46, 50–52, 69, 70, 72, 98, 120, 122, 163
- beamalign, 33, 42, 81, 82, 90–92, 92, 127, 128, 141, 145, 153
- beamangle, 11, 33, 46, 87, 88, 90, 91, 91, 92, 103, 105, 138, 139, 149, 153
- beamdiffractionorder, 49, 49, 50
- beamdiv, 33, 92, 101, 101, 103, 104, 123, 128, 130, 138, 139, 161, 162
- beaminside, 11, 93, 94–97, 121, 125–127, 138, 163
- beaminsidefirst, 80, 91, 94, 94, 95, 106, 131, 134, 136, 142, 164
- beaminsidelast, 79, 81, 94, 94, 95, 103, 131, 148, 164
- beammode, 49, 96
- beamnodealign, 82, 96, 97, 148

- beampathcount, 93, **93**, 148
- beampathskip, 82, **92**, 93, 148–150
- beampos, 10, 11, 37, 40, 42, 43, 81, 87, 89, 91, **91**, 94, 96, 97, 100–103, 106, 107, 141, 145, 149, 153
- beamwidth, 11, 14, 15, 17, 33, 42, 45, 47–51, 69, 80, 82, 93, 96, 97, 99, **100**, 101, 102, 104, 105, 121–123, 128, 130, 131, 134, 135, 138, 139, 148, 162
- bssize, **43**, 85, 136, 145, 146
- bsstyle, 17, **43**, **43**, 146
- caxisinv, **34**
- caxislength, 34, **34**
- compname, 17, 18, 73, **73**, 84–86, 89, 93, 96, 107, 111, 113, 129, 132, 134, 135, 138, 142–144, 146, 148, 155
- compoffset, 23, **23**, 24, 146
- compshift, 17, **23**, **23**, 74, 83, 128, 132, 133, 138, 144, 149
- conn, 161, 163
- connjoin, 161
- coupleralign, 60, **60**, 61, 75, 112, 118, 122, 133, 144
- couplersep, 59, **59**, 65, 75
- couplersize, 59, **59**, 65, 75
- couplertype, 59, **59**, 60, 65, 75, 137, 140
- crystalheight, 34, **34**
- crystalsize, 34, **34**, 136
- crystalwidth, 34, **34**
- detsize, **70**, 82, 140, 142, 146–148
- dettype, 13, 17, 71, **71**, 129, 132
- diffractionorders, **48**, 49, 50
- doveprismsize, 36, **36**, 87
- eleccouplerinput, 64, **64**
- eleccouplersep, **64**
- eleccouplersize, 64, **64**
- eleccouplertype, 64, **64**
- elecmixersize, **67**
- endbox, **22**, 163
- extnode, 40, **75**, 76–78, 83, 124, 126, 127, 132, 134, 138, 143, 144, 147, 156, 163
- extnodealign, 76, **76**, 77, 134, 161
- extnodes, 77, **77**
- fade, 100
- fadefunc, 99, 100, **100**

- fadefuncname, 99, 100, **100**, 150
- fadepoints, 99, **99**, 100
- fadeto, 99, **99**, 100, 150
- fdlsiz, **56**, 103
- fiber, 15–17, 25, 32, 33, 61, 63, 69, 103, 109, 111, 112, 114, **114**, 115, 118, 128, 132, 137, 138, 140, 143, 144, 147
- fiber*, 113
- fiberalign, 109, **109**, 110, 113
- fiberangleA, 110, **110**, 113
- fiberangleB, 110, **110**, 113
- fiberboxcount, 62, 63, **63**
- fiberboxheight, **62**, 63
- fiberboxsepin, 62, **63**
- fiberboxsepout, 62, 63, **63**
- fiberboxsize, 63, **63**
- fiberboxwidth, **62**, 63
- fibercolsize, **70**
- fiberloopradius, 54, **54**, 128
- fiberloops, **53**, 54, 133, 137, 147
- fiberloopsep, 54, **54**, 161
- fiberpolsize, **57**
- fiberstyle, 12, 65, 112, **112**, 113, 137
- fillstyle, 11
- filterangle, 69, **69**
- filtersize, **68**
- filtertype, 68, **68**
- forcebeaminside, 95, **95**
- glanthompsongap, **36**, 37, 143
- glanthompsonheight, 36, **36**
- glanthompsonsize, **36**, 37, 143
- glanthompsonwidth, 36, **36**
- gratingalign, 45, **45**, 149
- gratingcount, 44, **44**
- gratingdepth, 44, **44**
- gratingheight, **44**
- gratinglinewidth, **45**
- gratingtype, 44, **44**
- gratingwidth, **44**, 139
- innercompalign, 24, **24**, 69
- innerheight, **30**, 90, 138

- innerlabel, [21](#), [23](#), [76](#), [77](#), [112](#), [136](#), [145](#)
- isolatorsize, [55](#)
- label, [15-17](#), [19](#), [21](#), [21](#), [34](#), [35](#), [53](#), [54](#), [63](#), [76](#), [77](#), [111](#), [120-122](#), [124-133](#), [135-138](#), [140-147](#)
- labelalign, [19](#), [19](#), [21](#), [125-127](#), [131](#)
- labelangle, [10](#), [19](#), [20](#), [21](#), [24](#), [125](#), [128](#), [130](#), [131](#), [136](#), [137](#), [145](#)
- labeloffset, [10](#), [13-15](#), [17](#), [19](#), [19](#), [20](#), [21](#), [39](#), [45](#), [59](#), [60](#), [75](#), [84](#), [85](#), [120](#), [129](#), [131-133](#), [136](#), [138](#), [141](#), [145](#), [148](#), [155](#)
- labelref, [10](#), [20](#), [20](#), [21](#), [76](#)
- labelstyle, [15-17](#), [19](#), [136](#), [137](#), [147](#)
- lamp, [34](#)
- lampscale, [35](#)
- lens, [9-11](#), [13-15](#), [17](#), [22](#), [23](#), [25](#), [27](#), [28](#), [28](#), [33](#), [51](#), [80](#), [82](#), [89](#), [91](#), [93](#), [96](#), [97](#), [102-104](#), [106](#), [107](#), [123](#), [128](#), [130](#), [131](#), [134-136](#), [138](#), [141](#), [142](#), [146](#), [148](#), [154](#)
- lensheight, [11](#), [14](#), [15](#), [27](#), [27](#), [136](#)
- lensradius, [11](#), [13-15](#), [28](#), [28](#), [80](#), [131](#)
- lensradiusleft, [27](#), [27](#), [28](#)
- lensradiusright, [27](#), [28](#)
- lenswidth, [28](#), [28](#)
- linestyle, [11](#)
- loadbeam, [80-82](#), [96](#), [105](#), [105](#), [106](#)
- loadbeampoints, [104](#), [104](#), [105](#), [123](#), [128](#), [130](#), [131](#), [135](#), [149](#), [161](#)
- mirrordepth, [17](#), [39](#), [40](#), [40](#), [41](#), [43](#), [76](#), [77](#)
- mirrorlinewidth, [38](#)
- mirrorradius, [38](#), [39](#), [91](#), [139](#), [145](#)
- mirrorrtype, [17](#), [24](#), [39](#), [39](#), [40](#), [41](#), [43](#), [76](#), [77](#), [80](#), [91](#), [92](#), [99](#), [101](#), [106](#), [131](#), [136](#), [139](#), [140](#), [142](#), [145](#), [149](#)
- mirrorwidth, [38](#), [80](#), [99](#), [101](#), [102](#), [136](#), [139](#), [141](#), [142](#), [146](#), [149](#)
- n, [10](#), [11](#), [36](#), [80-82](#), [86-88](#), [88](#), [89](#), [90](#), [96](#), [97](#), [102](#), [103](#), [106](#), [107](#), [123](#), [127](#), [129](#), [131](#), [134](#), [135](#), [138](#), [142](#), [146](#), [148-150](#)
- namingscheme, [72](#), [162](#)
- newBeam, [98](#)
- newFiber, [112](#), [128](#)
- newFiberIn, [117](#)
- newFiberIn1, [117](#)
- newFiberIn2, [117](#)
- newFiberOut, [117](#)
- newFiberOut1, [117](#)
- newFiberOut2, [117](#)

- newOptComp, 25, 26, **26**
- newWire, **113**
- newWireIn, 117
- newWireIn1, 117
- newWireIn2, 117
- newWireOut, 117
- newWireOut1, 117
- newWireOut2, 117
- npos, 21
- oapmirroraperture, **42**, 43
- optampsize, **54**
- optboxheight, **30**, 31
- optboxsize, 11, 12, 20–22, **31**, 63, 73, 76, 77, 81, 82, 85, 112, 128, 129, 131, 132, 136, 138, 140, 142–147, 155
- optboxwidth, 13, 15–17, 20, **30**, 31, 94, 95, 128–130, 133, 135–137, 140, 141, 143, 148
- optcircangle, **57**, 58, 144
- optcircangleA, **57**
- optcircangleB, **57**
- optcirsize, **57**
- optdiodesize, **35**
- optdipolecomp, 121, **121**
- optdipolesize, **120**, 121, 135, 145
- optgridcount, 45
- optgriddepth, 45
- optgridheight, 45
- optgridlinewidth, 45
- optgridtype, 45
- optgridwidth, 45
- optional, 25, **26**
- optmzmsize, **55**, 143
- opttripolecomp, **121**, 122
- outerheight, **30**, 136, 139, 146
- parmirrorheight, **41**, 42
- parmirrorwidth, **41**, 42
- pentaprismsize, **52**, 78
- phlinewidth, 30, **30**
- phwidth, 30, **30**, 90, 130, 131, 138, 139
- plateheight, 29, **29**, 90–92, 101, 102, 106, 107, 123, 136, 138, 145, 162
- platelinewidth, **29**, 130

- platesize, **29**
- platewidth, **29, 29**
- polcontrolsize, **55**
- polcontroltype, **55, 55**
- pollinewidth, **37**
- polsize, **37, 130, 143**
- poltype, **38, 38, 93, 130, 143**
- position, **10–17, 20, 21, 21, 22, 33, 34, 58, 61, 65, 70, 79, 81, 84, 85, 90–92, 94, 95, 101–103, 106–108, 112, 114–116, 118, 124–133, 135–141, 143–148, 162**
- prismalign, **51, 90, 150, 160**
- prismangle, **50, 50, 90, 93, 96, 148, 149**
- prismsize, **50, 87**
- prismtype, **50, 51, 93, 96, 148, 160**
- pswarning, **49, 103**
- raprismalign, **52**
- raprismsize, **52**
- raytrace, **36, 86, 87, 98, 99, 128**
- refractiveindex, **90**
- reverse, **45, 46, 90**
- rotateref, **23, 23, 78, 79, 83, 92, 124, 126, 129**
- savebeam, **81, 82, 105, 105, 106, 107, 148**
- savebeampoints, **79, 104, 105, 128, 131, 139**
- showifcnodes, **62, 92, 121, 154, 154**
- showinterfaces, **155, 155**
- showoptdots, **52, 72**
- sourceheight, **33, 33**
- sourcesize, **33**
- sourcewidth, **33, 33**
- startinside, **91, 95, 106, 106, 107, 128**
- startinsidecount, **92, 95, 96, 148**
- startnode, **110, 110, 111, 137, 140**
- stopinside, **81, 95, 104, 106, 107, 123, 128, 130**
- stopinsidecount, **42, 95, 96**
- stopnode, **110, 110, 111, 112, 137, 140**
- switchsize, **56**
- switchstyle, **56, 56**
- synthshape, **66, 66, 144**
- synthsize, **65**
- synthtype, **65, 66, 144**

- thicklens, **28**
- usefiberstyle, **53, 53**
- useNA, **87, 88, 121, 142**
- usewirestyle, **53, 53**
- variable, **17, 40, 40, 60, 139, 145**
- voltage, **34**
- wire, **31, 115, 116, 118, 132, 143, 147**
- wire*, **112**
- wirealign, **113**
- wireangleA, **113, 144**
- wireangleB, **113, 132, 143, 144**
- wirestyle, **65, 113**

parmirrorheight, **41, 42**

parmirrorwidth, **41, 42**

\pentaprism, **52, 52, 78, 94, 135**

pentaprismsize, **52, 78**

phlinewidth, **30, 30**

phwidth, **30, 30, 90, 130, 131, 138, 139**

PiezoMirror, **39, 41, 41**

\pinhole, **30, 30, 90, 130, 131, 135, 136, 138, 139, 141, 146**

plateheight, **29, 29, 90–92, 101, 102, 106, 107, 123, 136, 138, 145, 162**

platelinewidth, **29, 130**

platesize, **29**

platewidth, **29, 29**

Polarization, **38, 38, 143**

\polarization, **37, 37, 38, 82, 93, 130, 136, 143**

\polcontrol, **55, 55, 133, 137, 144**

polcontrolsize, **55**

polcontroltype, **55, 55**

polllinewidth, **37**

polsize, **37, 130, 143**

poltype, **38, 38, 93, 130, 143**

position, **10–17, 20, 21, 21, 22, 33, 34, 58, 61, 65, 70, 79, 81, 84, 85, 90–92, 94, 95, 101–103, 106–108, 112, 114–116, 118, 124–133, 135–141, 143–148, 162**

prismalign, **51, 90, 150, 160**

prismangle, **50, 50, 90, 93, 96, 148, 149**

prismsize, **50, 87**

prismtype, **50, 51, 93, 96, 148, 160**

\psbezier, **69, 70**

pspicture, 73, 80
pst-eucl, 6
pst-intersect, 6
pst-node, 6, 21, 112
pst-optexp, 6, 24, 84, 119
pst-plot, 6
pstricks, 6
pstricks-add, 6, 98
pswarning, 49, 103

R

raprismalign, 52
raprismsize, 52
raytrace, 36, 86, 87, 98, 99, 128
refractiveindex, 90
reverse, 45, 46, 90
\rightangleprism, 51, 51, 52, 160
rotateref, 23, 23, 78, 79, 83, 92, 124, 126, 129

S

savebeam, 81, 82, 105, 105, 106, 107, 148
savebeampoints, 79, 104, 105, 128, 131, 139
SemitransMirror, 39, 41, 140
showifcnodes, 62, 92, 121, 154, 154
showinterfaces, 155, 155
showoptdots, 52, 72
sourceheight, 33, 33
sourcesize, 33
sourcewidth, 33, 33
startinside, 91, 95, 106, 106, 107, 128
startinsidecount, 92, 95, 96, 148
startnode, 110, 110, 111, 137, 140
stopinside, 81, 95, 104, 106, 107, 123, 128, 130
stopinsidecount, 42, 95, 96
stopnode, 110, 110, 111, 112, 137, 140
styles
– ArrowCompStyle, 31, 32, 57
– BarCompStyle, 32, 33
– Beam, 14, 15, 17, 37, 42, 46–50, 69, 80, 82, 88–90, 92, 96–98, 98, 99, 101, 102, 104, 105, 123, 128–131, 134–136, 138–142, 146, 148–150, 153, 155, 162

- CrystalCaxis, 35, **35**
- CrystalLamp, 35, **35**
- DetectorStyle, 71, **71**
- ExtendedMirror, 24, 39, 41, **41**, 76, 77, 131, 139
- FdlArrow, **57**
- Fiber, 53, 57, 58, 65, 108, 109, 111, **111**, 112, 116, 117, 132, 147
- FiberIn, 117
- FiberIn1, 117
- FiberIn2, 117
- FiberOut, 117
- FiberOut1, 57, 117
- FiberOut2, 57, 117
- FilterStyle, 69, **69**
- IfcNodeStyle, **154**
- IfcStyle, 155, **155**
- IsolatorArrow, 56, **56**
- OptCircArrow, 58, **58**, 144
- OptComp, 25, **25**, 26, 69, 71, 139
- OptionalStyle, 25, **25**, 26
- PiezoMirror, 39, 41, **41**
- Polarization, 38, **38**, 143
- SemitransMirror, 39, **41**, 140
- SynthStyle, 66, **66**
- VariableCoupler, **60**
- VariableStyle, 25, 40
- Wire, 53, 65, 67, 113, **113**, 117, 132, 143, 144, 147
- WireIn, 117
- WireIn1, 117
- WireIn2, 117
- WireOut, 117
- WireOut1, 67, 117
- WireOut2, 67, 117

switchsize, **56**

switchstyle, 56, **56**

synthshape, 66, **66**, 144

synthsize, **65**

SynthStyle, 66, **66**

synthtype, **65**, 66, 144

T

thicklens, **28**

\transmissiongrating, **45, 46**

U

usefiberstyle, **53, 53**

useNA, **87, 88, 121, 142**

usewirestyle, **53, 53**

V

variable, **17, 40, 40, 60, 139, 145**

VariableCoupler, **60**

VariableStyle, **25, 40**

voltage, **34**

W

\wdmcoupler, **59, 59, 60, 61, 122, 133, 140, 144**

\wdmsplitter, **16, 17, 59, 59, 60, 61, 115, 118, 133, 138, 140, 144, 147**

Wire, **53, 65, 67, 113, 113, 117, 132, 143, 144, 147**

wire, **31, 115, 116, 118, 132, 143, 147**

wire*, **112**

wirealign, **113**

wireangleA, **113, 144**

wireangleB, **113, 132, 143, 144**

WireIn, **117**

WireIn1, **117**

WireIn2, **117**

WireOut, **117**

WireOut1, **67, 117**

WireOut2, **67, 117**

wirestyle, **65, 113**

A. Revision history

This revision history is a list of changes relevant to users of this package. Changes of a more technical nature which do not affect the user interface or the behavior of the package are not included in the list. If an entry in the revision history states that a feature has been *improved* or *extended*, this indicates a modification which either does not affect the syntax and behavior of the package or is syntactically backwards compatible (such as the addition of an optional argument to an existing command). Entries stating that a feature has been *deprecated*, *modified*, *fixed*, *renamed*, or *removed* demand attention. They indicate a modification which may require changes to existing documents. The numbers on the right indicate the relevant section of this manual, more information regarding backward compatibility can be found in Sec. 12.3.

5.1 2014-11-19

Added component `\oapmirror`.

Added support for filled beams with path interfaces.

5.0 2014-10-30

Added support for arbitrary interface shapes (experimental, requires the `pst-intersect` package and `pstricks` version 2.53).

Added component `\parabolicmirror`.

4.10 2014-06-04

Added value `firstcomp` to parameter `beamalign` 9.2.3

Modified `\optsource` to handle also the beam alignment 4.8

4.9 2014-06-03

Added new component `\optsource` 4.8

Improved naming of the last component name, which can now be accessed also by its number besides using `N`.

Fixed bug related to `extnodes` together with `compname`

4.8 2014-02-28

Added component <code>\optaom</code>	4.20
Added option <code>beammode</code>	9.2.4
Added option <code>showinterfaces</code>	12.1

4.7 2014-01-12

Added syntax <code><num></code> for accessing relative component IDs	8.1
--	-----

4.6 2013-11-27

Added option <code>extnodes</code>	8.5
Added option <code>optdipolecomp</code>	10
Added option <code>opttripolecomp</code>	10

4.5 2013-07-21

Added option <code>usewirestyle</code>	5
Added line style <code>fade</code> , with options <code>fadeto</code> , <code>fadepoints</code> , <code>fadefunc</code> , and <code>fadefuncname</code>	9.2.7 .

4.4 2013-07-14

Added component <code>\transmissiongrating</code>	4.19
Modified <code>innerlabel</code> to explicitly set <code>labelalign=c</code>	
Fixed trailing spaces in <code>savebeampoints</code> (and <code>savebeam</code>).	
Allow Postscript code in <code>angle</code> .	

4.3a 2013-07-02

Fixed tripole alignment bugs introduced in version 4.3.	
Again fixed the alignment of reflective <code>\optprism</code>	4.21
Fixed improper alignment of <code>\optprism</code> label	12.3.1

4.3 2013-06-28

Added option <code>raprismalign</code>	4.22
Fixed alignment of <code>\rightangleprism</code> and rotation of its label.	

4.2 2013-06-07

Added option startinsidecount	9.2.4
Added option stopinsidecount	9.2.4
Added option beampathcount	9.2.3
Added option beamnodealign	9.2.5
Added \oeBeamVecMedian	8.9
Added \oeBeamCenter	8.8
Modified the orientation of a reflective \optprism	12.3.3
Fixed bug of \optprism and \rightangleprism inside an optexp environment.	
Fixed bug of wrong \rightangleprism rotation in special cases.	
Fixed buggy handling of multiple saved beams	9.5

4.1 2013-05-21

Added option prismtype	4.21
----------------------------------	------

4.0 2013-04-15

Added electrical components	6
Added wire connections with \drawwire	9.7
Added component \optarrowcomp	4.6
Added component \optbarcomp	4.7
Improved \optdetector	7.3
Extended \drawfiber to support component ranges	9.6
Extended couplersep to support automatic calculation	5.10
Fixed appearance of \optfiber	12.3.4
Fixed extnodealign to affect also multipoles	8.5
Fixed pswarning.	
Removed old, buggy beam connection code	12.3.4
Removed option conn	12.3.4
Removed option connjoin	12.3.4
Removed \newOptexpDipoleNolabel	12.3.4

3.6 2013-03-20

Added option compoffset	3.3
Added option beampathskip	9.2.3
Added option innercompalign	3.3
Fixed some bugs in the raytracing and beam code.	

3.5 2013-02-22

Added option `filterangle` 7.1
 Fixed wrong output fiber style in `\optcoupler`
 Fixed strange Postscript error which occurred with some interpreters.

3.4 2013-02-03

Fixed bugs when using `fillstyle` for some components.
 Extended option `switchsize` 5.6
 Modified coupler center node to be on the base line 8.3
 Extended error checking for `\drawbeam` and `\drawfiber` 9.1
 Fixed some bugs in the connection code.

3.3a 2012-09-18

Fixed bug which was exposed by an update of `pst-node`.
 Fixed trailing spaces.

3.3 2012-08-17

Extended `\opttripole` and `\optdipole` 10
 Modified reference angle for label of `\opttripole` 12.3.5
 Added option `optdipolesize` to support two interfaces for `\optdipole` . 10
 Added option `gratingalign` 4.18
 Added option `forcebeaminside` 9.2.4
 Extended option `mirrorradius` 4.14
 Improved `\drawbeam` and `\drawwidebeam` to be able to use `ArrowInside`
 without `linestyle` 9.2.7
 Fixed orientation of `\oenodeBeamUp` and `\oenodeBeamLow` 8.8
 Fixed a bug related to `beaminside` and ambiguous components.
 Fixed some trailing spaces.
 Added more examples 11

3.2 2012-07-26

Added component `\glanthompson` 4.12
 Added access to beam vectors with `\oeBeamVec`, `\oeBeamVecUp` and
 `\oeBeamVecLow` 8.9
 Fixed wrong computation of node distance in `\fiberbox`
 Fixed bug in `\wdmsplitter` for `coupleralign=b` and `couplertype=none`
 Fixed bug when filling components placed with `position=end/start`
 Added value `absolute` to parameter `labelref` 3.1

Added examples	11
--------------------------	----

3.1 2012-07-17

Added component \fiberbox	5.11
Extended connection macros to not required curly braces around node parenthesis any more	8.1
Modified \fibercollimator to have allowbeaminside=false by default	7.2
Extended fiber couplers to allow using only two nodes	5.10
Fixed a bug concerning node expressions with \drawfiber	

3.0 2012-07-09

Modified beam connections with \drawbeam to support raytracing	9.2
Added wide beams with \drawwidebeam	9.3
Added \drawfiber for fiber connections	9.6
Added optexp environment for layering of components and connections	9.9
Added german documentation.	
Modified naming of component nodes	8
Modified extnode to work with more components	8.5
Modified angle and rotateref to affect all components	3.3
Modified endbox to affect all components	3.2
Extended position by values start and end	3.2
Extended abspos by values start and end	3.2
Removed deprecated lens code which used lenswidth and lensheight for construction	4.1
Added option platesize	4.3
Added option phwidth	4.4
Modified option caxislength	4.9
Deprecated option lampscale	4.9
Added style CrystalCaxis	4.9
Added style CrystalLamp	4.9
Added option optboxsize	4.5
Extended option detsize	7.3
Added style DetectorStyle	7.3
Removed deprecated \detector, use \optdetector	7.3
Extended option doveprismsize	4.11
Added style Polarization	4.13
Deprecated option pollinewidth	4.13
Removed option polwidth	4.13
Removed option pol	4.13

Added style VariableStyle	4.14
Added mirror type semitrans	4.14
Renamed \optgrid to \optgrating	4.18
Renamed optgridwidth to gratingwidth	4.18
Renamed optgridheight to gratingheight	4.18
Renamed optgriddepth to gratingdepth	4.18
Renamed optgridcount to gratingcount	4.18
Renamed optgridtype to gratingtype	4.18
Renamed optgridlinewidth to gratinglinewidth	4.18
Added option prismalign	4.21
Modified faulty alignment of \rightangleprism	4.22
Extended option optampsize	5.2
Extended option optmzsize	5.3
Added option polcontroltype	5.4
Extended option isolatorsize	5.5
Added style IsolatorArrow	5.5
Extended option fdlsiz	5.7
Added style FdlArrow	5.7
Extended option fiberpolsize	5.8
Added component \optcirculator	5.9
Extended option couplersize	5.10
Extended option couplertype	5.10
Renamed option align to coupleralign	5.10
Added style VariableCoupler	5.10
Added style FilterStyle	7.1
Extended option fibercolsize	7.2
Removed deprecated option labelrelative	3.1
Removed deprecated option iwidth	4.4
Removed deprecated option owidth	4.4
Removed deprecated option bswidth	4.17
Deprecated \newOptexpDipoleNoLabel, use \newOptexpDipole	10.2
Deprecated option refractiveindex	9.2.2
Deprecated option conn	9.2
Extended option fiber	9.8

2.1 2009-11-05

Added component \optfiberpolarizer	5.8
Added option compshift	3.3
Added option label	3.1

Added option connjoin	
Added options addtoBeam and newBeam	9.2
Added style OptComp and related options addtoOptComp and newOptComp	3.5
Added option bsstyle	4.17
Extended \fibercollimator to use up to four reference nodes	7.2
Improved thicklens to work also with plain lenses	4.1
Use pst-doc class for the documentation.	

2.0 2008-07-27

Added fiber-optical components	5
Added component \optdiode	4.10
Added component \pentaprism	4.23
Added component \rightangleprism	4.22
Added component \doveprism	4.11
Added component \optprism	4.21
Added \drawbeam	9.2
Added component connections (options fiber, conn and beam)	9
Added option compname	8.1
Added option extrnode	8.5
Renamed \detector to \optdetector	7.3

1.2 2008-06-17

Modified lens design to use interface curvatures	4.1
Added options lensradiusleft and lensradiusright	4.1
Added option thicklens	4.1
Added option lenstype	4.1
Added option mirrorradius (curved mirrors)	4.14
Added option optgridtype (binary gratings)	4.18
Added \newOptexpDipole	10.2
Added \newOptexpDipoleNolabel	10.2
Added \newOptexpTripole	10.2
Added \newOptexpFiberDipole	10.2
General improvements of T _E X and Postscript code	

1.1 2007-09-06

Improved labeling features	3.1
Added parameter labelref	3.1
Replaced labelrelative by labelref=relative	3.1
Renamed \polarisation to \polarization	4.13

Renamed polwidth to polsize	4.13
Renamed pol to poltype	4.13
Renamed bswidth to bssize	4.17
Renamed iwidth to innerheight	4.4
Renamed owidth to outerheight	4.4
Added support for fillstyle for all components	

1.0 2007-07-18

First CTAN version