The `flowchart` package[*]
Flowchart Shapes for Ti*k*Z

Adrian P. Robson[†]

19 March 2015

1 Introduction

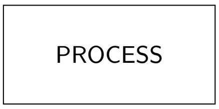
This package provides shapes for drawing program flowcharts. They are based on the classic *IBM Flowcharting Template*, which conforms to ISO 1028:1973, with some IBM extensions. (this has since been revised by ISO 5807:1985).

At the moment, there is only a limited selection of the standard symbols, but other symbols might be added in the future.

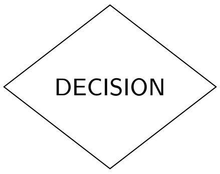
This package requires that `makeshape` [1] and of course PGF/Ti*k*Z [2] are also installed .

2 The Symbols

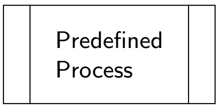
The package provides the following symbols as defined in the ISO standard:



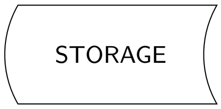
Process – Any processing function; or defined operations causing change in value, form or location of information.



Decision – A decision or switching-type operation that determines which of a number of alternative paths are followed.



Predefined Process – One or more named operations or program steps specified in a subroutine or another set of flowcharts.

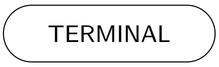


Storage – Input or output using any kind of online storage.

---

[*] This document corresponds to flowchart 3.3, dated 2015/03/19.

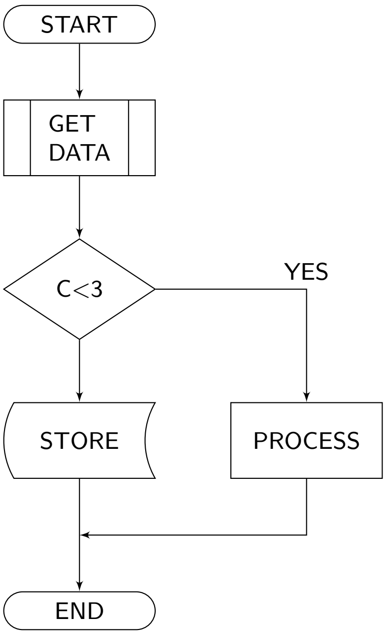
[†] `adrian.robson@nepsweb.co.uk`



Terminal – A terminal point in a flowchart: start, stop, halt delay or interrupt. It may show exit from a closed subroutine.

3 Usage

The example below uses all of the symbols given in §2, and shows how they can be laid out and connected with Ti*k*Z:



It needs the following in the document's preamble:

```
\usepackage{flowchart}
\usetikzlibrary{arrows}
```

The Ti*k*Z package is included in `flowchart`, so it does not have to be explicitly loaded. However, any `\usetikzlibrary` commands that are needed must be placed after `\loadpackage{flowchart}`.

The flowchart above is produced by the following `tikzpicture` environment:

```
\begin{tikzpicture}[>=latex',font={\sf \small}]

\def\smbwd{2cm}

\node (terminal1) at (0,0) [draw, terminal,
   minimum width=\smbwd,
   minimum height=0.5cm] {START};

\node (predproc1) at (0,-1.5) [draw, predproc, align=left,
   minimum width=\smbwd,
   minimum height=1cm] {GET\\ DATA};
```

```
12
13  \node (decide1) at (0,-3.5) [draw, decision,
14     minimum width=\smbwd,
15     minimum height=1cm] {C$<$3};
16
17  \node (storage1) at (0,-5.5) [draw, storage,
18     minimum width=\smbwd,
19     minimum height=1cm] {STORE};
20
21  \node (process1) at (3,-5.5) [draw, process,
22     minimum width=\smbwd,
23     minimum height=1cm] {PROCESS};
24
25  \coordinate (point1) at (0,-6.75);
26
27  \node (terminal2) at (0,-7.75) [draw, terminal,
28     minimum width=\smbwd,
29     minimum height=0.5cm] {END};
30
31  \draw[->] (terminal1) -- (predproc1);
32  \draw[->] (predproc1) -- (decide1);
33  \draw[->] (decide1) -| node[above]{YES} (process1);
34  \draw[->] (decide1) -- (storage1);
35  \draw[->] (process1) |- (point1);
36  \draw[->] (storage1) -- (point1) -- (terminal2);
37
38  \end{tikzpicture}
```

3.1 Symbols

The flow chart symbols are created as nodes, as shown in lines 5-7, which defines a `terminal` shape. The minimum dimension keys should be used to create consistently sized symbols. In particular, a defined value should be used for the width in all symbols. So `\smbwd` is defined on line 3, and used in lines 6, 10, 14, 18, 22, and 28.

3.2 Layout

In this example, we have used absolute coordinates to position the diagram's nodes. This works well for small flowcharts, but relative positioning might be better for larger diagrams.

3.3 Connectors

The shapes are connected by drawing lines between them as shown in lines 31-36 of the example. Here, unqualified node names are used, but explicit anchor names such as `(nodename.north east)` could be used instead.

It can sometimes be convenient to place a connector at an arbitrary point on a shape. The `(nodename.45)` notation achieves this, where the number gives the angle from the centre of the shape to the connecting point on its boundary.

Right angled connections are traditionally used in flowcharts, and these are created with the `-|` and `|-` notations shown in lines 33 and 35 of the example.

Joining connectors is best done by declaring a named coordinate, and using it as the meeting point. In the example, a coordinate called `point1` is declared on line 25, and then used in line 35 and 36 to connect `process1` and `storage1` to `terminal2`.

4 Implementation

The implementation of `flowchart.sty` uses the `makeshape` package, which provides support for custom PGF shapes. With this, we have only have to create boundary and anchor path macros, and anchor points for each shape.

There are three pairs of keys that have to be accommodated by PGF shapes: inner and outer separation, and minimum dimensions. The `makeshape` package has corrected text box macros `\ctbnex` and `\ctbney`, which automatically handle inner separation; and the PGF keys `\pgfshapeouterxsep`, `\pgfshapeouterysep`, `\pgfshapeminheight` and `\pgfshapeminwidth`, which give the outer separation and minimum dimensions of the shape.

4.1 Preamble

The `makeshape` package provides the `tikz` package. However, the `shapes` library is also needed:

```
\RequirePackage{makeshape}
\RequirePackage{tikz}
\usetikzlibrary{shapes}
```

4.2 Predproc Shape

This is the Predefined Process symbol.

4.2.1 Anchor and background paths

These macros define the paths that are used by the `makeshape setpaths` command in the shape's `\pgfdeclareshape` macro, which is described in §4.2.2.

`\band` The shape's side band size is an internal constant:

```
\def\band{10pt}
```

`\predprocAnchorpath` The `\predprocAnchorpath` macro defines the shape's *anchor path*. It 'draws' the path on which the shape's calculated anchor points lay. It is very similar to first part of `\predproc@shape` that draws the outer path, but it corrects for outer separation and does not draw any side bands.

```
\def\predprocAnchorpath{
```

First, get the corrected text box's NE corner using `\ctbnex` and `\ctbney`, then make room for the side band.

```
    \pgf@xa=\ctbnex
    \pgf@ya=\ctbney
    \advance\pgf@xa by \band
```

Correct for minimum dimensions and outer separation:

```
  \mincorrect{\pgf@xa}{\pgfshapeminwidth}
  \advance\pgf@xa\pgfshapeouterxsep
  \mincorrect{\pgf@ya}{\pgfshapeminheight}
  \advance\pgf@ya\pgfshapeouterysep
```

Finally, draw the anchor path, which is a rectangle, using the values in `\pgf@xa` and `\pgf@ya` that were calculated above:

```
  \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
  \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
  \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
  \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
  \pgfpathclose
}
```

`\predprocBackground`

The `\predprocBackground` macro draws the shape's path, including its side band. It is used in the shape's `\backgroundpath` macro.

```
\def\predprocBackground{
```

First, get the corrected text box's NE corner using `\ctbnex` and `\ctbney`, then make room for the side band.

```
  \pgf@xa=\ctbnex
  \pgf@ya=\ctbney
  \advance\pgf@xa by \band
```

Correct for minimum dimensions but *do not add* outer separation:

```
  \mincorrect{\pgf@xa}{\pgfshapeminwidth}
  \mincorrect{\pgf@ya}{\pgfshapeminheight}
```

Finally, draw the outer shape, which is a rectangle, using the values in `\pgf@xa` and `\pgf@ya` that were calculated above:

```
  \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
  \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
  \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
  \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
  \pgfpathclose
```

Finally, we draw the inner shape, which completes the shape with its side bands. The x-coordinate is aligned on the right side band position, then the side bands are drawn:

```
  \advance\pgf@xa by -\band
  \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
  \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
  \pgfpathmoveto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
  \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
}
```

4.2.2 Predproc shape declaration

`\pgfdeclareshape`
`predproc`

This is the `\pgfdeclareshape` declaration for the `predproc` shape.

```
\pgfdeclareshape{predproc}{
```

The path macros defined in §4.2.1 are used as follows with the `setpaths` command provided by the `makeshape` package to draw the shape and make boundary intersection calculations.

```
  \setpaths{\predprocAnchorpath}{\predprocBackground}
```

`\savedanchor` `\northeast`

The `\northeast` saved anchor is used to define the position of the NE corner of the shape. The calculation is similar that used in the anchor path described in §4.2.1, and corrects for inner and outer separation, and minimum dimensions. It returns the coordinates of the point in `\pgf@x` and `\pgf@y`.

```
   \savedanchor{\northeast}{
      \pgf@x = \ctbnex
      \advance\pgf@x by \band
      \mincorrect{\pgf@x}{\pgfshapeminwidth}
      \advance\pgf@x\pgfshapeouterxsep
      \pgf@y = \ctbney
      \mincorrect{\pgf@y}{\pgfshapeminheight}
      \advance\pgf@y\pgfshapeouterysep
   }
```

`\anchor` `north` `north east` `east` `south east` `south` `south west` `west` `north west`

There are some standard anchors, which are all based on the `\northeast` saved anchor:

```
   \anchor{north}{ \northeast \pgf@x=0pt }
   \anchor{north east}{ \northeast }
   \anchor{east}{ \northeast \pgf@y=0pt }
   \anchor{south east}{ \northeast \pgf@y=-\pgf@y }
   \anchor{south}{ \northeast \pgf@x=0pt \pgf@y=-\pgf@y }
   \anchor{south west}{ \northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y }
   \anchor{west}{ \northeast \pgf@x=-\pgf@x \pgf@y=0pt }
   \anchor{north west}{ \northeast \pgf@x=-\pgf@x }

}
```

4.3 Storage Shape

4.3.1 Support Macros

`\storagepath`

The `storage` shape's background path is defined in `\storagepath`. It requires the following register to be set:

`\pgf@x`	x coordinate of NE corner excluding outer separation
`\pgf@y`	y coordinate of NE corner excluding outer separation
`\pgf@xc`	arc offset for y coordinate

The NE corner is stored in `\pgf@xa` and `\pgf@ya` and and the SW corner is put in `\pgf@xb` and `\pgf@yb`. The SW x-coordinate has to be moved right by the arc offset to compensate for the curve of the shapes west side.

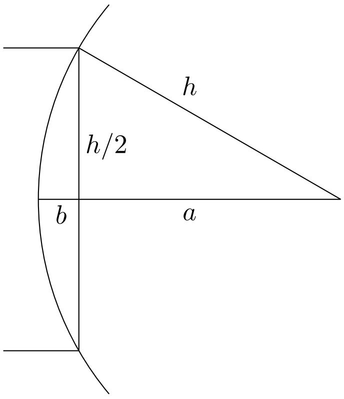
```
\def\storagepath{
   \pgf@xa=\pgf@x \pgf@ya=\pgf@y
   \pgf@xb=-\pgf@xa \pgf@yb=-\pgf@ya
   \advance\pgf@xb by \pgf@xc
```

The shape is drawn from its SW corner moving counter clockwise. The radius for the arcs is the height.

```
   \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}
   \pgfpatharc{210}{150}{2*\pgf@ya}
   \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@ya}}
   \pgfpatharc{150}{210}{2*\pgf@ya}
   \pgfpathclose
}
```

`\arcoffset`

The `storage` shape's arc offset is calculated by the `\arcoffset` macro. The required arc offset is b, and the shape's height is h in the diagram below.



$$\begin{aligned} h^2 &= a^2 + \left(\frac{h}{2}\right)^2 \\ a^2 &= h^2 - \frac{h^2}{4} = \frac{3h^2}{4} \\ a &= h\sqrt{3/4} \qquad (1) \\ b &= h - a = h - h\sqrt{3/4} \\ &= h(1 - \sqrt{3/4}) \approx 0.134h \qquad (2) \end{aligned}$$

The macro's parameters are:

`#1`	the calculated arc offset
`#2`	half the height

Equation 2 given above is implemented as follows:

```
\def\arcoffset#1#2{
    \pgfmathsetlength#1{0.134*2*#2}
}
```

`\storageParams`

The `\storageParams` macro calculates reference values for the shape with no outer separation. The following have values assigned after it is called:

`\pgf@x`	x-coordinate of NE corner excluding outer separation
`\pgf@y`	y-coordinate of NE corner excluding outer separation
`\pgf@xc`	arc offset for y-coordinate

First get the shape's corrected text box:

```
\def\storageParams{
    \pgf@xa=\ctbnex
    \pgf@ya=\ctbney
```

Correct for minimum height but not for outer separation.

```
    \mincorrect{\pgf@ya}{\pgfshapeminheight}
```

Calculate the room needed for the side arc, which is one of the macro's outputs, and use it to change the x-coordinate:

```
    \arcoffset{\pgf@xc}{\pgf@ya}
    \advance\pgf@xa by \pgf@xc
```

Finally, correct for minimum width and set the output registers:

```
    \mincorrect{\pgf@xa}{\pgfshapeminwidth}
    \pgf@x=\pgf@xa
    \pgf@y=\pgf@ya
}
```

`\storageParamsOuter`

The `\storageParamsOuter` macro calculates reference values for the shape with its outer separation included. The following have values assigned after it is called:

`\pgf@x`	x coordinate of NE corner including outer separation
`\pgf@y`	y coordinate of NE corner including outer separation
`\pgf@xc`	arc offset for y coordinate

Its implementation has a lot in common with `\storageParams`. First, get the NE corner of the corrected text box:

```
\def\storageParamsOuter{
   \pgf@xa=\ctbnex
   \pgf@ya=\ctbney
```

Correct for minimum height and outer separation:

```
   \mincorrect{\pgf@ya}{\pgfshapeminheight}
   \advance\pgf@ya\pgfshapeouterysep
```

Calculate the arc offset, which is an output, and make room for the side curve:

```
   \arcoffset{\pgf@xc}{\pgf@ya}
   \advance\pgf@xa by \pgf@xc
```

Finally, correct for minimum width and outer separation. Then set the output registers:

```
   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
   \advance\pgf@xa\pgfshapeouterxsep
   \pgf@x=\pgf@xa
   \pgf@y=\pgf@ya
}
```

4.3.2 Anchor and background paths

The anchor and background path macros both use `\storagepath`, but with with different parameters. The output registers of the macros `\storageParams` and `\storageParamsOuter` are compatible with the inputs of `\storagepath`.

`\storageAnchorpath`

The `\storageAnchorpath` macro defines the shape's *anchor path*. It 'draws' the path on which the shape's calculated anchor points lay. This is similar to the background path but it is corrected for outer separation and minimum dimensions.

```
\def\storageAnchorpath{
   \storageParamsOuter
   \storagepath
}
```

`\storageBackground`

The `\storageBackground` macro draws the path that is the outer boundary of the `storage` shape. This excludes outer separation but corrects for minimum height and width.

```
\def\storageBackground{
   \storageParams
   \storagepath
}
```

4.3.3 Storage shape declaration

`\pgfdeclareshape`
`storage`

This is the `\pgfdeclareshape` declaration for the `storage` shape.

```
\pgfdeclareshape{storage}{
```

The path macros defined in §4.3.2 are used with the `setpaths` command provided by the `makeshape` package to draw the shape and make boundary intersection calculations:

```
   \setpaths{\storageAnchorpath}{\storageBackground}
```

`\savedanchor`
`\northeast`

There are two saved anchors defined for the `storage` shape. The `\northeast` saved anchor is used to define the position of the NE corner of the shape. The calculation is similar that used in the anchor path described in §4.3.2, and corrects for inner and outer separation, and minimum dimensions. It returns the coordinates of the point in `\pgf@x` and `\pgf@y`, and its implementation is trivial since `\storageParamsOuter` does the required work.

```
  \savedanchor{\northeast}{
    \storageParamsOuter
  }
```

`\savedanchor`
`\northeast`

The `\northeastArc` saved anchor is similar, but corrects for the arc offset.

```
  \savedanchor{\northeastArc}{
    \storageParamsOuter
    \advance\pgf@x by -\pgf@xc
  }
```

`\anchor`
`north`
`north east`
`east`
`south east`
`south`
`south west`
`west`
`north west`

The standard anchors are defined. These are based on the `\northeast` and `\northeastArc` saved anchors:

```
  \anchor{north}{ \northeast \pgf@x=0pt }
  \anchor{north east}{ \northeast }
  \anchor{east}{ \northeastArc \pgf@y=0pt }
  \anchor{south east}{ \northeast \pgf@y=-\pgf@y }
  \anchor{south}{ \northeast \pgf@x=0pt \pgf@y=-\pgf@y }
  \anchor{south west}{ \northeastArc \pgf@x=-\pgf@x \pgf@y=-\pgf@y }
  \anchor{west}{ \northeast \pgf@x=-\pgf@x \pgf@y=0pt }
  \anchor{north west}{ \northeastArc \pgf@x=-\pgf@x }
```

`\anchor`
`north`
`north east`
`east`

Three additional anchors are defined that follow the shape's bounding rectangle.

```
  \anchor{east r}{ \northeast \pgf@y=0pt }
  \anchor{north west r}{ \northeast \pgf@x=-\pgf@x }
  \anchor{south west r}{ \northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y }

}
```

4.4 Process Shape

`\pgfdeclareshape`
`process`

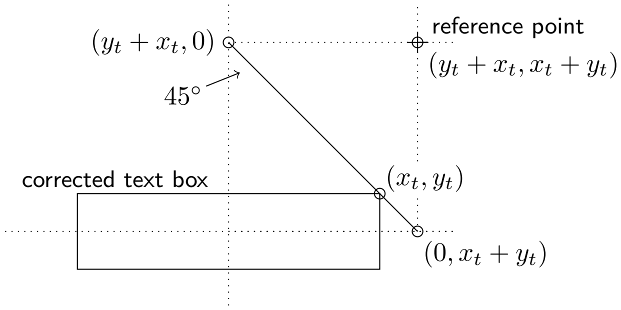
The `process` shape is simply implemented by inheriting relevant anchors and paths from the standard `rectangle` shape without any changes:

```
\pgfdeclareshape{process}{
  \inheritsavedanchors[from=rectangle]
  \inheritanchor[from=rectangle]{center}
  \inheritanchor[from=rectangle]{text}
  \inheritanchor[from=rectangle]{north}
  \inheritanchor[from=rectangle]{north east}
  \inheritanchor[from=rectangle]{east}
  \inheritanchor[from=rectangle]{south east}
  \inheritanchor[from=rectangle]{south}
  \inheritanchor[from=rectangle]{south west}
  \inheritanchor[from=rectangle]{west}
  \inheritanchor[from=rectangle]{north west}
  \inheritbackgroundpath[from=rectangle]
  \inheritanchorborder[from=rectangle]
}
```

4.5 Decision Shape

4.5.1 Support Macros

Two macros `\decisionref` and `\decisionrefout` give a reference point for the `decision` shape. The default shape is a rotated square which touches the corrected text box. The location and calculation of the default reference point is illustrated below:



`\decisionrefout` The `\decisionrefout` macro gives a reference point on the bounding rectangle of the `decision` shape. Inner and outer separation are included, and it is corrected for the shape's minimum dimensions. On completion, the coordinates of the point are in `\pgf@xa` and `\pgf@ya`.

First, the coordinates of the corrected text box are put into some registers:

```
\def\decisionrefout{
   \pgf@xa=\ctbnex
   \pgf@ya=\ctbney
   \pgf@xb=\ctbnex
   \pgf@yb=\ctbney
```

The reference point's x-coordinates is calculated and corrected for minimum width, and outer x-separation:

```
   \advance\pgf@xa by \pgf@yb
   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
   \advance\pgf@xa by \pgfshapeouterxsep
```

Then the calculation and correction is repeated for the y-coordinate:

```
   \advance\pgf@ya by \pgf@xb
   \mincorrect{\pgf@ya}{\pgfshapeminheight}
   \advance\pgf@ya by \pgfshapeouterysep
}
```

`\decisionref` The `\decisionref` macro is almost the same as `\decisionrefout` but has no correction for outer separation. Again, the coordinates of the point are in `\pgf@xa` and `\pgf@ya` on completion:

```
\def\decisionref{
   \pgf@xa=\ctbnex
   \pgf@ya=\ctbney
   \pgf@xb=\ctbnex
   \pgf@yb=\ctbney
   \advance\pgf@xa by \pgf@yb
   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
   \advance\pgf@ya by \pgf@xb
   \mincorrect{\pgf@ya}{\pgfshapeminheight}
}
```

`\decisionpath` The `\decisionpath` macro draws the `\decision` shape's path. It expects the reference point coordinates to be in `\pgf@xa` and `\pgf@ya`. The path is drawn clockwise starting at the top:

```
\def\decisionpath{
   \def\refx{\pgf@xa}
   \def\refy{\pgf@ya}
   \pgfpathmoveto{\pgfpoint{0}{\refy}}
   \pgfpathlineto{\pgfpoint{\refx}{0}}
   \pgfpathlineto{\pgfpoint{0}{-\refy}}
   \pgfpathlineto{\pgfpoint{-\refx}{0}}
   \pgfpathclose
}
```

4.5.2 Anchor and background paths

`\decisionanchor` The `\decisionanchor` macro is the anchor path for the shape's `setpath`:

```
\def\decisionanchor{
   \decisionrefout
   \decisionpath
}
```

`\decisionborder` The `\decisionborder` macro is the border path for the shape's `setpath`:

```
\def\decisionborder{
   \decisionref
   \decisionpath
}
```

4.5.3 Decision shape declaration

`\pgfdeclareshape` `decision` This is the `\pgfdeclareshape` declaration for the `decision` shape.

```
\pgfdeclareshape{decision}{
```

The path macros defined in §4.6.2 are used with the `setpaths` command provided by the `makeshape` package to draw the shape and make boundary intersection calculations:

```
   \setpaths{\decisionanchor}{\decisionborder}
```

There are three saved anchors defined for the `decision` shape. They have to correct for inner and outer separation, and minimum dimensions; and return the coordinates of the point in `\pgf@x` and `\pgf@y`. However, this is simplified because `\decisionrefout` does the required work.

`\savedanchor` `\north` The `\north` saved anchor uses the y-coordinate of the reference point:

```
   \savedanchor{\north}{
      \decisionrefout
      \pgf@x = 0pt
      \pgf@y = \pgf@ya
   }
```

`\savedanchor` `\north`

The `\east` saved anchor uses the x-coordinate of the reference point:

```
   \savedanchor{\east}{
      \decisionrefout
      \pgf@x = \pgf@xa
      \pgf@y = 0pt
   }
```

`\savedanchor` `\northeast`

The `\northeast` saved anchor uses the both coordinates of the reference point to calculate the required point:

```
   \savedanchor{\northeast}{
      \decisionrefout
      \divide\pgf@xa by 2
      \divide\pgf@ya by 2
      \pgf@x = \pgf@xa
      \pgf@y = \pgf@ya
   }
```

`\anchor` `north` `north east` `east` `south east` `south` `south west` `west` `north west`

The standard anchors are defined. These are based on the `\north` `\east` and `\northeast` saved anchors:

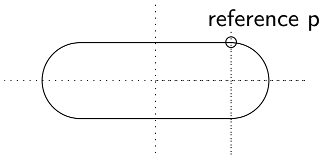
```
   \anchor{north}{ \north }
   \anchor{north east}{ \northeast }
   \anchor{east}{ \east }
   \anchor{south east}{ \northeast \pgf@y=-\pgf@y }
   \anchor{south}{ \north \pgf@y=-\pgf@y }
   \anchor{south west}{ \northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y }
   \anchor{west}{ \east \pgf@x=-\pgf@x }
   \anchor{north west}{ \northeast \pgf@x=-\pgf@x }

}
```

4.6 Terminal Shape

4.6.1 Support Macros

Two macros `\terminalrefneout` and `\terminalrefne` give a reference point on the boundary surface of the `terminal` shape.



`\terminalrefneout`

The `\terminalrefneout` macro gives a reference point on the boundary surface of the `terminal` shape. Inner and outer separation are included, and it is corrected for the shape's minimum dimensions. On completion, the coordinates of the point are in `\pgf@xa` and `\pgf@ya`.

First, the coordinates of the corrected text box are obtained, and then the width of the rounded end is add to the x-coordinate to get the bounding dimensions:

```
\def\terminalrefneout{
   \pgf@xa=\ctbnex
   \pgf@ya=\ctbney
   \advance\pgf@xa by \pgf@ya
```

Then these bounding coordinates are corrected for minimum dimensions, and outer separation:

```
    \mincorrect{\pgf@xa}{\pgfshapeminwidth}
    \advance\pgf@xa\pgfshapeouterxsep
    \mincorrect{\pgf@ya}{\pgfshapeminheight}
    \advance\pgf@ya\pgfshapeouterysep
```

Finally, the corrected x-coordinate is reduced by the width of the rounded end to give the required reference point:

```
    \advance\pgf@xa by -\pgf@ya
}
```

`\terminalrefne`

The `\terminalrefne` macro is almost the same as `\terminalrefneout` but has no correction for outer separation. Again, the coordinates of the point are in `\pgf@xa` and `\pgf@ya` on completion:

```
\def\terminalrefne{
    \pgf@xa=\ctbnex
    \pgf@ya=\ctbney
    \advance\pgf@xa by \pgf@ya
    \mincorrect{\pgf@xa}{\pgfshapeminwidth}
    \mincorrect{\pgf@ya}{\pgfshapeminheight}
    \advance\pgf@xa by -\pgf@ya
}
```

`\terminalpath`

The `\terminalpath` macro draws the `terminal` shape's path. It expects the reference point coordinates to be in `\pgf@xa` and `\pgf@ya`. The path is drawn anticlockwise:

```
\def\terminalpath{
    \def\refx{\pgf@xa}
    \def\refy{\pgf@ya}
    \def\radius{\refy}
    \pgfpathmoveto{\pgfpoint{\refx}{\refy}}
    \pgfpathlineto{\pgfpoint{-\refx}{\refy}}
    \pgfpatharc{90}{270}{\radius}
    \pgfpathlineto{\pgfpoint{\refx}{-\refy}}
    \pgfpatharc{270}{360}{\radius}
    \pgfpatharc{0}{90}{\radius}
    \pgfpathclose
}
```

4.6.2 Anchor and background paths

`\terminalanchor`

The `\terminalanchor` macro is the anchor path for the shape's `setpath`:

```
\def\terminalanchor{
    \terminalrefneout
    \terminalpath
}
```

`\terminalborder`

The `\terminalborder` macro is the border path for the shape's `setpath`:

```
\def\terminalborder{
    \terminalrefne
    \terminalpath
}
```

4.6.3 Terminal shape declaration

`\pgfdeclareshape`
`terminal`

This is the `\pgfdeclareshape` declaration for the `terminal` shape.

```
\pgfdeclareshape{terminal}{
```

The path macros defined in §4.6.2 are used with the `setpaths` command provided by the `makeshape` package to draw the shape and make boundary intersection calculations:

```
    \setpaths{\terminalanchor}{\terminalborder}
```

`\savedanchor`
`\northeast`

There are two saved anchors defined for the `terminal` shape. The `\northeast` saved anchor gives the position of the shape's 'reference point' used above. It corrects for inner and outer separation, and minimum dimensions. It returns the coordinates of the point in `\pgf@x` and `\pgf@y`, and its implementation is simple since `\terminalrefneout` does the required work.

```
    \savedanchor{\northeast}{
       \terminalrefneout
       \pgf@x = \pgf@xa
       \pgf@y = \pgf@ya
    }
```

`\savedanchor`
`\northeastBB`

The `\northeastBB` macro gives the coordinates of the NE corner of the shape's bounding rectangle in `\pgf@x` and `\pgf@y`. It corrects for inner and outer separation, and minimum dimensions. It is obtain by adding the width of the shape's round end to the x-coordinate of the reference point:

```
    \savedanchor{\northeastBB}{
       \terminalrefneout
       \advance \pgf@xa by \pgf@ya
       \pgf@x = \pgf@xa
       \pgf@y = \pgf@ya
    }
```

`\anchor`
`north`
`north east`
`east`
`south east`
`south`
`south west`
`west`
`north west`

The standard anchors are defined. These are based on the `\northeast` and `\northeastBB` saved anchors:

```
    \anchor{north}{ \northeast \pgf@x=0pt }
    \anchor{north east}{ \northeast }
    \anchor{east}{ \northeastBB \pgf@y=0pt }
    \anchor{south east}{ \northeast \pgf@y=-\pgf@y }
    \anchor{south}{ \northeast \pgf@x=0pt \pgf@y=-\pgf@y }
    \anchor{south west}{ \northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y }
    \anchor{west}{ \northeastBB \pgf@x=-\pgf@x \pgf@y=0pt }
    \anchor{north west}{ \northeast \pgf@x=-\pgf@x }
```

`\anchor`
`north east r`
`south east r`
`south west r`
`north west r`

Four additional anchors are defined that follow the shape's bounding rectangle.

```
    \anchor{north east r}{\northeastBB}
    \anchor{south east r}{\northeastBB \pgf@y=-\pgf@y}
    \anchor{south west r}{\northeastBB \pgf@x=-\pgf@x \pgf@y=-\pgf@y}
    \anchor{north west r}{\northeastBB \pgf@x=-\pgf@x}

}
```

References

[1] Adrian P. Robson, *The makeshape package and a method for creating custom shapes in PGF*, 2013. Available as `makeshape.pdf` from `ctan.org`.

[2] Till Tantau, *The TikZ and PGF Packages, Manual for version 2.10*, 2010. Available as `pgfmanual.pdf` from `ctan.org`.