

# The `flowchart` package\*

## Flowchart Shapes for `TikZ`

Adrian P. Robson<sup>†</sup>

28 January 2013

### 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.

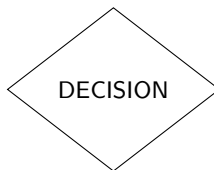
This package requires that `makeshape` [1] and of course `PGF/TikZ` [2] are also installed .

### 2 The Symbols

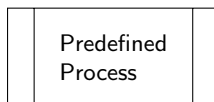
The package provides the following symbols as defined in the ISO standard:



**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.

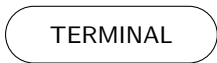


**Storage** – Input or output using any kind of online storage.

---

\*This document corresponds to `flowchart` 3.2, dated 2013/01/28.

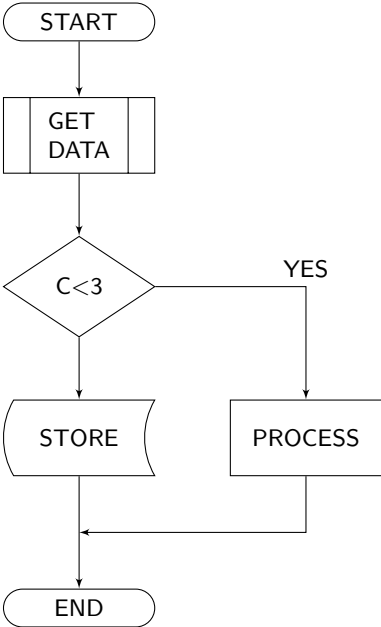
<sup>†</sup>`adrian.robson@nepsweb.co.uk`



**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 TikZ:



It needs the following in the document’s preamble:

```

\usepackage{flowchart}
\usetikzlibrary{arrows}

```

The TikZ 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:

```

1 \begin{tikzpicture}[>=latex',font={\sf \small}]
2
3 \def\smbwd{2cm}
4
5 \node (terminal1) at (0,0) [draw, terminal,
6   minimum width=\smbwd,
7   minimum height=0.5cm] {START};
8
9 \node (predproc1) at (0,-1.5) [draw, predproc, align=left,
10  minimum width=\smbwd,
11  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:

```
1 \RequirePackage{makeshape}
2 \RequirePackage{tikz}
3 \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:

```
4 \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.

```
5 \def\predprocAnchorpath{
```

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

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

Correct for minimum dimensions and outer separation:

```

9   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
10  \advance\pgf@xa\pgfshapeouterxsep
11  \mincorrect{\pgf@ya}{\pgfshapeminheight}
12  \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:

```

13  \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
14  \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
15  \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
16  \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
17  \pgfpathclose
18 }

```

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

```

19 \def\predprocBackground{

```

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

```

20  \pgf@xa=\ctbnex
21  \pgf@ya=\ctbney
22  \advance\pgf@xa by \band

```

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

```

23  \mincorrect{\pgf@xa}{\pgfshapeminwidth}
24  \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:

```

25  \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
26  \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
27  \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
28  \pgfpathlineto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
29  \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:

```

30  \advance\pgf@xa by -\band
31  \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}}
32  \pgfpathlineto{\pgfpoint{\pgf@xa}{-\pgf@ya}}
33  \pgfpathmoveto{\pgfpoint{-\pgf@xa}{\pgf@ya}}
34  \pgfpathlineto{\pgfpoint{-\pgf@xa}{-\pgf@ya}}
35 }

```

## 4.2.2 Predproc shape declaration

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

```

predproc 36 \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.

```

37  \setpaths{\predprocAnchorpath}{\predprocBackground}

```

`\savedanchor` 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`.

```

38 \savedanchor{\northeast}{
39   \pgf@x = \ctbnex
40   \advance\pgf@x by \band
41   \mincorrect{\pgf@x}{\pgfshapeminwidth}
42   \advance\pgf@x\pgfshapeouterxsep
43   \pgf@y = \ctbney
44   \mincorrect{\pgf@y}{\pgfshapeminheight}
45   \advance\pgf@y\pgfshapeouterysep
46 }

```

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

```

north east 47 \anchor{north}{ \northeast \pgf@x=0pt }
east        48 \anchor{north east}{ \northeast }
south east  49 \anchor{east}{ \northeast \pgf@y=0pt }
south       50 \anchor{south east}{ \northeast \pgf@y=-\pgf@y }
south west  51 \anchor{south}{ \northeast \pgf@x=0pt \pgf@y=-\pgf@y }
west        52 \anchor{south west}{ \northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y }
north west  53 \anchor{west}{ \northeast \pgf@x=-\pgf@x \pgf@y=0pt }
            54 \anchor{north west}{ \northeast \pgf@x=-\pgf@x }
            55 }

```

## 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 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.

```

56 \def\storagepath{
57   \pgf@xa=\pgf@x \pgf@ya=\pgf@y
58   \pgf@xb=-\pgf@xa \pgf@yb=-\pgf@ya
59   \advance\pgf@xb by \pgf@xc

```

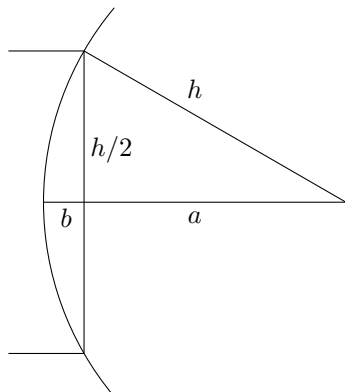
The shape is drawn from its SW corner moving counter clockwise. The radius for the arcs is the height.

```

60   \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}
61   \pgfpatharc{210}{150}{2*\pgf@ya}
62   \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@ya}}
63   \pgfpatharc{150}{210}{2*\pgf@ya}
64   \pgfpathclose
65 }

```

`\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.



$$h^2 = a^2 + \left(\frac{h}{2}\right)^2$$

$$a^2 = h^2 - \frac{h^2}{4} = \frac{3h^2}{4}$$

$$a = h\sqrt{3/4} \quad (1)$$

$$b = h - a = h - h\sqrt{3/4}$$

$$= h(1 - \sqrt{3/4}) \approx 0.134h \quad (2)$$

The macro's parameters are:

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

Equation 2 given above is implemented as follows:

```
66 \def\arcoffset#1#2{
67   \pgfmathsetlength#1{0.134*2*#2}
68 }
```

`\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:

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

Correct for minimum height but not for outer separation.

```
72   \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:

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

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

```
75   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
76   \pgf@x=\pgf@xa
77   \pgf@y=\pgf@ya
78 }
```

`\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:

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

Correct for minimum height and outer separation:

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

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

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

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

```
86   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
87   \advance\pgf@xa\pgfshapeouterxsep
88   \pgf@x=\pgf@xa
89   \pgf@y=\pgf@ya
90 }
```

### 4.3.2 Anchor and background paths

The anchor and background path macros both use `\storagepath`, but 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.

```
91 \def\storageAnchorpath{
92   \storageParamsOuter
93   \storagepath
94 }
```

`\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.

```
95 \def\storageBackground{
96   \storageParams
97   \storagepath
98 }
```

### 4.3.3 Storage shape declaration

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

```
storage 99 \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:

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



`\savedanchor` 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.

```

101   \savedanchor{\northeast}{
102     \storageParamsOuter
103   }

```

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

```

\northeast 104   \savedanchor{\northeastArc}{
105     \storageParamsOuter
106     \advance\pgf@x by -\pgf@xc
107   }

```

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

```

north east 108   \anchor{north}{ \northeast \pgf@x=0pt }
east       109   \anchor{north east}{ \northeast }
south east 110   \anchor{east}{ \northeastArc \pgf@y=0pt }
south      111   \anchor{south east}{ \northeast \pgf@y=-\pgf@y }
south west 112   \anchor{south}{ \northeast \pgf@x=0pt \pgf@y=-\pgf@y }
west       113   \anchor{south west}{ \northeastArc \pgf@x=-\pgf@x \pgf@y=-\pgf@y }
north west 114   \anchor{west}{ \northeast \pgf@x=-\pgf@x \pgf@y=0pt }
           115   \anchor{north west}{ \northeastArc \pgf@x=-\pgf@x }

```

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

```

north      116   \anchor{east r}{ \northeast \pgf@y=0pt }
north east 117   \anchor{north west r}{ \northeast \pgf@x=-\pgf@x }
east       118   \anchor{south west r}{ \northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y }

119 }

```

## 4.4 Process Shape

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

```

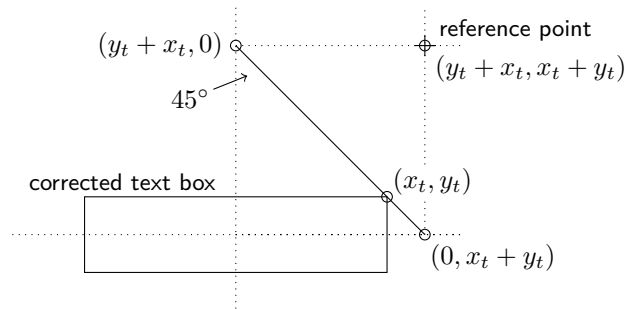
120 \pgfdeclareshape{process}{
121   \inheritsavedanchors[from=rectangle]
122   \inheritanchor[from=rectangle]{center}
123   \inheritanchor[from=rectangle]{text}
124   \inheritanchor[from=rectangle]{north}
125   \inheritanchor[from=rectangle]{north east}
126   \inheritanchor[from=rectangle]{east}
127   \inheritanchor[from=rectangle]{south east}
128   \inheritanchor[from=rectangle]{south}
129   \inheritanchor[from=rectangle]{south west}
130   \inheritanchor[from=rectangle]{west}
131   \inheritanchor[from=rectangle]{north west}
132   \inheritbackgroundpath[from=rectangle]
133   \inheritanchorborder[from=rectangle]
134 }

```

## 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:

```

135 \def\decisionrefout{
136   \pgf@xa=\ctbnex
137   \pgf@ya=\ctbney
138   \pgf@xb=\ctbnex
139   \pgf@yb=\ctbney

```

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

```

140   \advance\pgf@xa by \pgf@yb
141   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
142   \advance\pgf@xa by \pgfshapeouterxsep

```

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

```

143   \advance\pgf@ya by \pgf@xb
144   \mincorrect{\pgf@ya}{\pgfshapeminheight}
145   \advance\pgf@ya by \pgfshapeouterysep
146 }

```

`\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:

```

147 \def\decisionref{
148   \pgf@xa=\ctbnex
149   \pgf@ya=\ctbney
150   \pgf@xb=\ctbnex
151   \pgf@yb=\ctbney
152   \advance\pgf@xa by \pgf@yb
153   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
154   \advance\pgf@ya by \pgf@xb

```

```

155 \mincorrect{\pgf@ya}{\pgfshapeminheight}
156 }

```

`\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:

```

157 \def\decisionpath{
158   \def\refx{\pgf@xa}
159   \def\refy{\pgf@ya}
160   \pgfpathmoveto{\pgfpoint{0}{\refy}}
161   \pgfpathlineto{\pgfpoint{\refx}{0}}
162   \pgfpathlineto{\pgfpoint{0}{-\refy}}
163   \pgfpathlineto{\pgfpoint{-\refx}{0}}
164   \pgfpathclose
165 }

```

#### 4.5.2 Anchor and background paths

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

```

166 \def\decisionanchor{
167   \decisionrefout
168   \decisionpath
169 }

```

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

```

170 \def\decisionborder{
171   \decisionref
172   \decisionpath
173 }

```

#### 4.5.3 Decision shape declaration

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

```

decision 174 \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:

```

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

```

\north 176   \savedanchor{\north}{
177     \decisionrefout
178     \pgf@x = 0pt
179     \pgf@y = \pgf@ya
180   }

```

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

```

\ north 181 \savedanchor{\east}{
182 \decisionrefout
183 \pgf@x = \pgf@xa
184 \pgf@y = 0pt
185 }

```

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

```

186 \savedanchor{\northeast}{
187 \decisionrefout
188 \divide\pgf@xa by 2
189 \divide\pgf@ya by 2
190 \pgf@x = \pgf@xa
191 \pgf@y = \pgf@ya
192 }

```

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

```

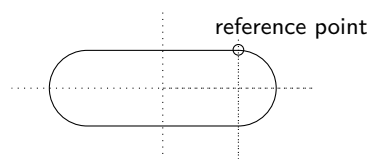
north 193 \anchor{north}{ \north }
north east 194 \anchor{north east}{ \northeast }
east 195 \anchor{east}{ \east }
south east 196 \anchor{south east}{ \northeast \pgf@y=-\pgf@y }
south 197 \anchor{south}{ \north \pgf@y=-\pgf@y }
south west 198 \anchor{south west}{ \northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y }
west 199 \anchor{west}{ \east \pgf@x=-\pgf@x }
north west 200 \anchor{north west}{ \northeast \pgf@x=-\pgf@x }
201 }

```

## 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 added to the x-coordinate to get the bounding dimensions:

```

202 \def\terminalrefneout{
203 \pgf@xa=\ctbnex
204 \pgf@ya=\ctbney
205 \advance\pgf@xa by \pgf@ya

```

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

```
206 \mincorrect{\pgf@xa}{\pgfshapeminwidth}
207 \advance\pgf@xa\pgfshapeouterxsep
208 \mincorrect{\pgf@ya}{\pgfshapeminheight}
209 \advance\pgf@ya\pgfshapeouterysep
```

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

```
210 \advance\pgf@xa by -\pgf@ya
211 }
```

`\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:

```
212 \def\terminalrefne{
213   \pgf@xa=\ctbnex
214   \pgf@ya=\ctbney
215   \advance\pgf@xa by \pgf@ya
216   \mincorrect{\pgf@xa}{\pgfshapeminwidth}
217   \mincorrect{\pgf@ya}{\pgfshapeminheight}
218   \advance\pgf@xa by -\pgf@ya
219 }
```

`\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:

```
220 \def\terminalpath{
221   \def\refx{\pgf@xa}
222   \def\refy{\pgf@ya}
223   \def\radius{\refy}
224   \pgfpathmoveto{\pgfpoint{\refx}{\refy}}
225   \pgfpathlineto{\pgfpoint{-\refx}{\refy}}
226   \pgfpatharc{90}{270}{\radius}
227   \pgfpathlineto{\pgfpoint{\refx}{-\refy}}
228   \pgfpatharc{270}{360}{\radius}
229   \pgfpatharc{0}{90}{\radius}
230   \pgfpathclose
231 }
```

#### 4.6.2 Anchor and background paths

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

```
232 \def\terminalanchor{
233   \terminalrefneout
234   \terminalpath
235 }
```

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

```
236 \def\terminalborder{
237   \terminalrefne
238   \terminalpath
239 }
```

### 4.6.3 Terminal shape declaration

```

\pgfdeclareshape This is the \pgfdeclareshape declaration for the terminal shape.
terminal 240 \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:
    241   \setpaths{\terminalanchor}{\terminalborder}

\saveanchor There are two saved anchors defined for the terminal shape. The \northeast
\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.
    242   \saveanchor{\northeast}{
    243       \terminalrefneout
    244       \pgf@x = \pgf@xa
    245       \pgf@y = \pgf@ya
    246   }

\saveanchor The \northeastBB macro gives the coordinates of the NE corner of the shape's
\northeastBB bounding rectangle in \pgf@x and \pgf@y. It corrects for inner and outer sepa-
ration, and minimum dimensions. It is obtain by adding the width of the shape's
round end to the x-coordinate of the reference point:
    247   \saveanchor{\northeastBB}{
    248       \terminalrefneout
    249       \advance \pgf@xa by \pgf@ya
    250       \pgf@x = \pgf@xa
    251       \pgf@y = \pgf@ya
    252   }

\anchor The standard anchors are defined. These are based on the \northeast and
north \northeastBB saved anchors:
north east 253   \anchor{north}{\northeast \pgf@x=0pt }
    east 254   \anchor{north east}{\northeast }
south east 255   \anchor{east}{\northeastBB \pgf@y=0pt }
    south 256   \anchor{south east}{\northeast \pgf@y=-\pgf@y }
south west 257   \anchor{south}{\northeast \pgf@x=0pt \pgf@y=-\pgf@y }
    west 258   \anchor{south west}{\northeast \pgf@x=-\pgf@x \pgf@y=-\pgf@y }
north west 259   \anchor{west}{\northeastBB \pgf@x=-\pgf@x \pgf@y=0pt }
    260   \anchor{north west}{\northeast \pgf@x=-\pgf@x }

\anchor Four additional anchors are defined that follow the shape's bounding rectangle.
north east r 261   \anchor{north east r}{\northeastBB}
south east r 262   \anchor{south east r}{\northeastBB \pgf@y=-\pgf@y}
south west r 263   \anchor{south west r}{\northeastBB \pgf@x=-\pgf@x \pgf@y=-\pgf@y}
north west r 264   \anchor{north west r}{\northeastBB \pgf@x=-\pgf@x}

    265 }

```

## 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`.