Notation legend

Class::method(object, numeric, string, array, bool , object, numeric, string, array, bool default); — optional

xx = Snippet code

Get numerical point data

x Returns the x-coordinate of point name
Part::x(name);

y Returns the y-coordinate of point name
Part::y(name);

dx Returns the x-axis delta between points name1 and name2
Part::deltaX(name1, name2);

dy Returns the x-axis delta between points name1 and name2
Part::deltaY(name1, name2);

a Returns the angle made by a line from points name1 to name2
Part::angle(name1, name2);

cl Returns the length of curve {start,cp1,cp2,end}
Part::curveLen(start, cp1, cp2, end);

d Returns the distance between points name1 and name2
Part::distance(name1, name2);

Various

Return point name
Part::loadPoint(name);

Returns true if point name exists in the part
Part::isPoint(name);

Returns the part title
Part::getTitle( );

Sets the part render flag to bool
Part::setRender(bool);

u Returns distance as a formatted string with the correct units
Part::unit(distance);

Generates a new unique id with optional prefix
Part::newId(prefix);

Adding points

ap Adds point as name, description is optional
Part::addPoint(name, point, description);

np Adds point name with coordinates x-coord and y-coord, description is optional
Part::newPoint(name, x-coord, y-coord, description);

Adding points based on other points

cp Clones point source into point name
Part::clonePoint(source, name);

fx Mirror point name around x-coord
Part::flipX(name, x-coord);

fy Mirror point name around y-coord
Part::flipY(name, y-coord);

r Rotate point moon angle degrees around point sun
Part::rotate(moon, sun, angle);

s Shift point name distance mm under angle degrees
Part::shift(name, angle, distance);

st Shift distance mm from origin towards direction
Part::shiftTowards(origin, direction, distance);

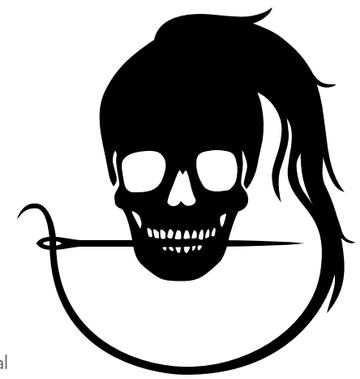
sft Shift a fraction from origin towards direction
Part::shiftFractionTowards(origin, direction, distance);

so Shift distance mm from origin passed direction
Part::shiftOutwards(origin, direction, distance);

sa Shift point distance mm along curve {start,cp1,cp2,end}
Part::shiftAlong(start, cp1, cp2, end, distance);

sfa Shift point a fraction of the curve length along curve {start,cp1,cp2,end}
Part::shiftFractionAlong(start, cp1, cp2, end, fraction);

Adding points based on lines/curves/circles



lc Add point at intersection of line segments {fromA,toA} and {fromB,toB}
Part::linesCross(fromA, toA, fromB, toB);

bc Add point at intersection of lines {fromA,toA} and {fromB,toB}
Part::beamsCross(start1, end1, start2, end2);

cer Add point at edge edge of curve {start,cp1,cp2,end}
edge is one of: left,right,top,bottom
Part::curveEdge(start, cp1, cp2, end, edge);

ccx Add points crossing curve {start,cp1,cp2,end} at x-coord, prefix is optional
Part::curveCrossesX(start, cp1, cp2, end, x-coord, prefix);

ccy Add points crossing curve {start,cp1,cp2,end} at y-coord, prefix is optional
Part::curveCrossesY(start, cp1, cp2, end, x-coord, prefix);

ccl Add points at intersections between curve {start,cp1,cp2,end} and line {from,to}, prefix is optional
Part::curveCrossesLine(start, cp1, cp2, end, from, to, prefix);

cc Add points at intersections between curves {startA,cp1A,cp2A,endA} and {startB,cp1B,cp2B,endB}, prefix is optional
Part::curvesCross(startA, cp1A, cp2A, endA, startB, cp1B, cp2B, endB, prefix);

sc Add points to split curve {start,cp1,cp2,end} in two halves at split, prefix and splitOnDelta are optional
If splitOnDelta is true, split must be a value between 0 and 1. If not, it's the name of the point to split on.
Part::splitCurve(nameStart, nameCp1, nameCp2, nameEnd, nameSplit, prefix , splitOnDelta false);

cic Add points at intersection of circle with center c1 and radius r1 and circle with center c2 and radius r2, prefix and sort are optional
Part::circlesCross(c1, r1, c2, r2, prefix, sort);

cil Add points at intersection of circle with center c1 and radius r1 and line from start to end, prefix and sort are optional
Part::circlesesLine(c1, r1, start, end, prefix, sort);

Adding non-points

text Adds message as text name anchored on anchor, attributes are optional
Part::newText(name, anchor, message, attributes);

p Adds pathstring as path name, attributes is optional
Part::newPath(name, patstring, attributes);

textop Adds message as textOnPath name along pathstring, attributes are optional
Part::newTextOnPath(name, pathstring, message, attributes);

note Adds message as note name anchored on anchor, hour, length, offset, and attributes are optional
Part::newNote(name, anchor, message, hour, length, offset, attributes);

snip Adds snippet name with defs id reference anchored on anchor, attributes are optional
Part::newSnippet(name, reference, anchor, attributes);

Adds include name with svg code svg
Part::newInclude(name, svg);

gl Adds a grainline path between from and to, text is optional
Part::newGrainline(from, to, text);

cof Adds a cut-on-fold path between from and to, text and offset is optional
Part::newCutonfold(from, to, text, offset);

notch Places a notch at each point in array points
Part::notch(points);

tit Adds title with number, title, and message anchored on anchor in optional mode
Mode is one of: default, vertical, horizontal, small, vertical-small, or horizontal-small
Part::addTitle(anchor, number, title, message, mode);

Adding dimensions

All these methods take 3 extra optional parameters at the end:
pathAttributes, labelAttributes, and leaderAttributes

dw Adds a width dimension from from to to at y-coord, text is optional
Part::newWidthDimension(from, to, y-coord, text);

dh Adds a height dimension from from to to at x-coord, text is optional
Part::newHeightDimension(from, to, x-coord, text);

dl Adds a linear dimension from from to to at offset, text is optional
Part::newLinearDimension(from, to, offset, text);

dc Adds a curved dimension at offset from pathstring, text is optional
Part::newCurvedDimension(pathstring, offset, text);

dws Adds a small width dimension from from to to at y-coord, text is optional
Part::newWidthDimensionSm(from, to, y-coord, text);

dhs Adds a small height dimension from from to to at x-coord, text is optional
Part::newHeightDimensionSm(from, to, x-coord, text);

dls Adds a small linear dimension from from to to at offset, text is optional
Part::newLinearDimensionSm(from, to, offset, text);

Path offset

op Offset path source as new path name at offset, render and attributes are optional
Part::offsetPath(name, source, offset, render false, attributes);

ops Offset pathstring as new path name at offset, render and attributes are optional
Part::offsetPathString(name, pathstring, offset, render false, attributes);

Pattern methods

osi Set option name to value unless it is already set
Pattern::setOptionIfUnset(name, value);

os Set option name to value
Pattern::setOption(name, value);

o Returns option name
Pattern::getOption(name);

o Returns option name - Alias of getOption
Pattern::o(name);

vsi Set value name to value unless it is already set
Pattern::setValueIfUnset(name, value);

vs Set value name to value
Pattern::setValue(name, value);

v Returns value name - Alias of getValue
Pattern::getValue(name);

v Returns value name
Pattern::v(name);

t Translate message
Pattern::t(message);

cps Clone points from part from into part into
Pattern::clonePoints(from, into);

Add a new part with name name
Pattern::newPart(name);

Add message to the pattern messages
Pattern::msg(message);

Add message to the pattern debug messages
Pattern::dbg(message);

Returns true if this is a paperless pattern
Pattern::isPaperless( );

Converts a stretch option to a scale factor and returns it
Pattern::stretchToScale(stretch);

Model methods

m Returns measurement name
Model::getMeasurement(name);

m Returns measurement name - alias for getMeasurement
Model::m(name);

Sets measurement name to value
Model::setMeasurement(name, value);

BezierToolbox methods

circ Returns control point offset to mimic a circle with radius
Methos is static, no BezierToolbox object needed
BezierToolbox::bezierCircle(radius);

Freesewing cheat sheet

freesewing.org