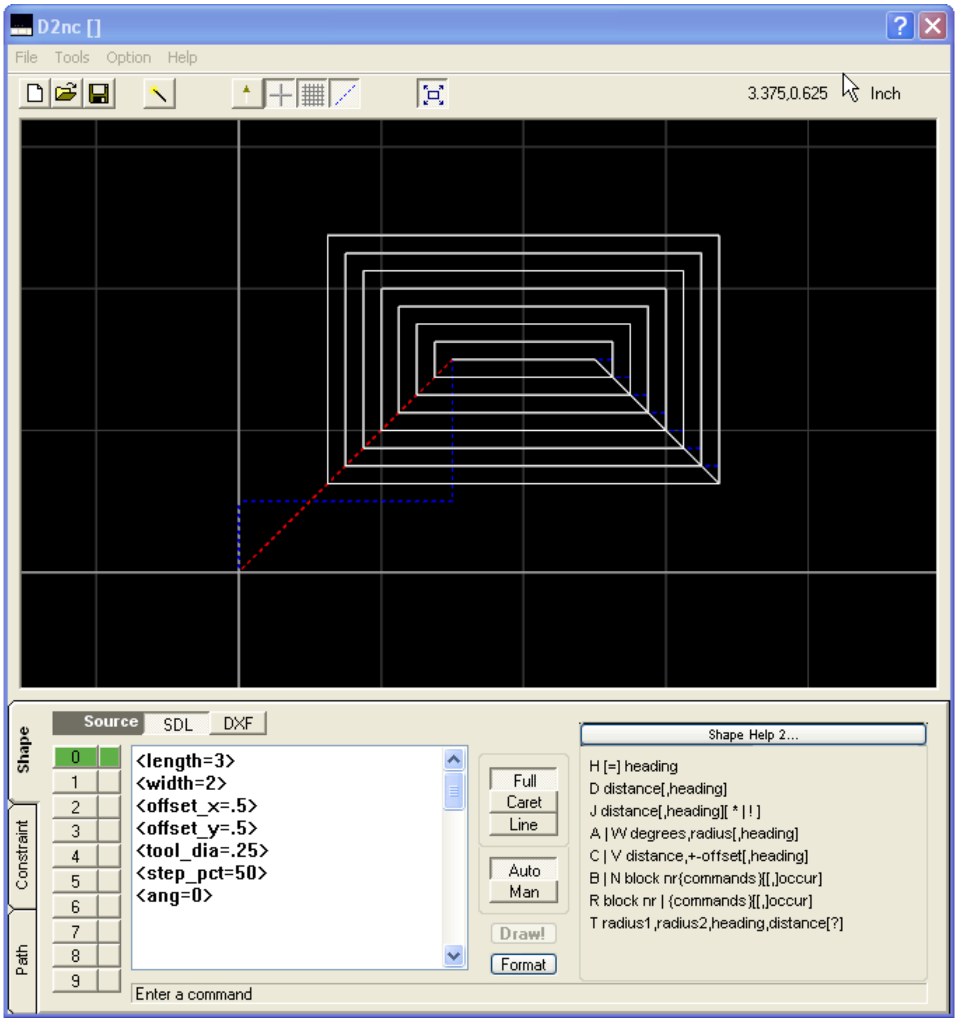
# D2nc
# User
# Manual

# Table of Contents

# 1 Introducing D2nc

**D2nc** is used to describe shapes with a Shape Description Language or extract them from DXF files. Shapes represent the tool path a tool must follow to produce a part. The shapes are converted to g-code by setting machine, material and tool constraints.

New in V2.0 are SDL Wizards 23 for generating shapes for common tasks.




D2nc can be run as a stand alone application or launched from a button on the Mach3 screen. To activate this button in Mach3, refer to the installation 5 steps for post install.

To learn to use D2nc it's best to follow the tutorials 34 but at the very least review the basic concepts 34 page.

The Shape Description Language 91 can be simple to use. There are only three basic commands used to produce most shapes. They are Jump, Draw and Arc. SDL can also be used as a programming language for writing SDL wizards and can produce complex shapes. All of the wizards in D2nc are written in SDL.

This help file last updated for release 2.2.0

## 1.1 Why D2nc?

D2nc originated from my need to create g-code in the machine shop, not at some remote CAD/CAM workstation, for simple tasks that occur so frequently. The Mach3 wizards only went so far and it was a case of, "More Power, Scotty!". D2nc has grown into a fairly complex program which I rely on in my own shop.

D2nc creates shapes from a language I've named Shape Description Language (SDL). SDL can be quite simple to use for creating basic shapes. The addition of variables and math functions to SDL in version 2 of D2nc has greatly enhanced its ability. A wizard interface allows for the rapid creation of g-code for some of the more common operations. For complex shapes, the option exists to import a DXF file and extract a shape from that.

D2nc will not suit everybody, nothing ever can. There is a built in 15 day trial key to allow you test its suitability for the way you work.

D2nc
Designed and written by
Graham Hollis

Enjoy!

## 1.2 Installation

The installation process of D2nc consists of running an installation program and following the on screen prompts. The first time you install D2nc you should reboot your system. For install patches or upgrade releases it should not be necessary to reboot unless otherwise instructed.

D2nc is a stand alone program which can be run from a D2nc icon placed on the desktop or launched using a button within Mach3.

One of the options presented during the install is installing the D2nc Mach3 screen sets. Selecting this option will install screen sets into the Mach3 directory to allow D2nc to be launched from within Mach3.

Launch the install program and follow the installation program prompts.

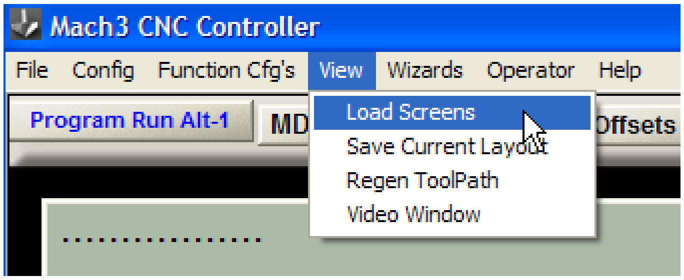
A D2nc program icon will be placed on the desktop and a program group created. D2nc will be installed into the directory C:\D2nc

**Mach3 Integration**

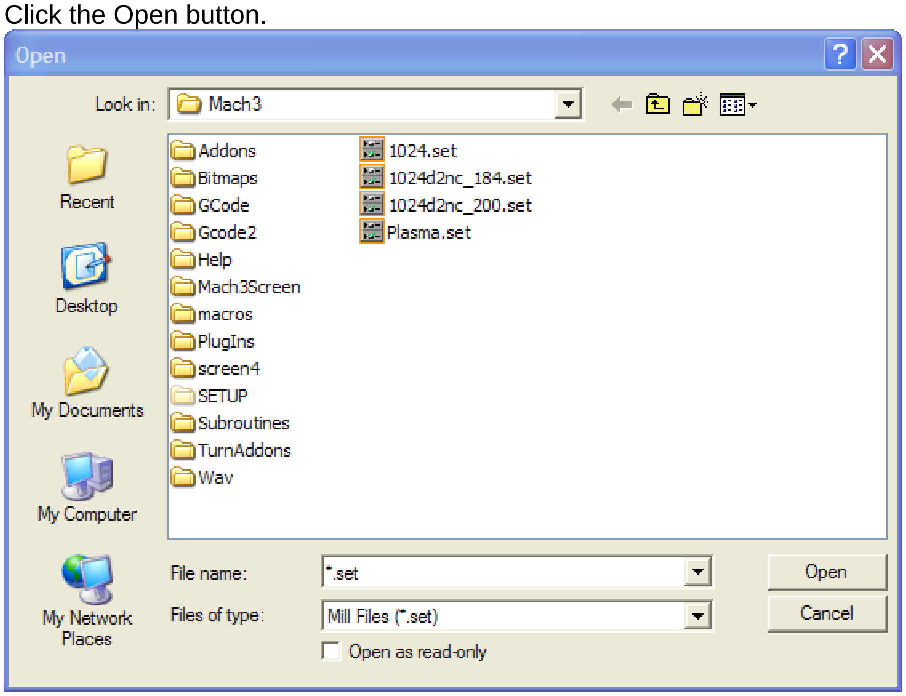
Two modified mill screen sets, 1024d2nc_200.set and 1024d2nc_300.set, are placed in the C:\Mach3 directory. These are the standard 1024.set screen sets which have been modified by adding a single button to the "Program Run" screen. The button is labeled D2nc.


To add the button to Mach3 you need to change the standard screen set to the modified one.

From the Mach3 menu select View -> Load Screens



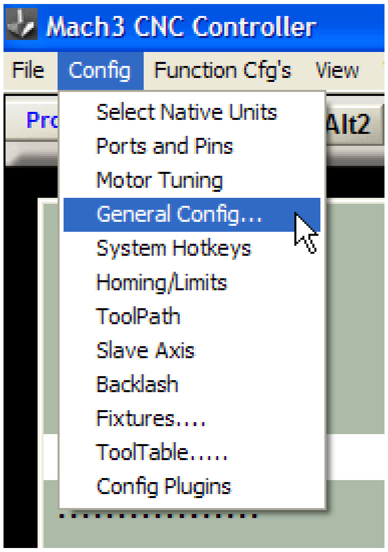

If you're running Mach3 v2.x, select the 1024d2nc_200.set file so its name appears in the File name box.
If you're running Mach3 v3.x, select the 1024d2nc_300.set file as this uses the built in sleep call.
Click the Open button.



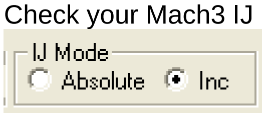

Mach3 will now have the D2nc program button ready for use.

Additionally you should check the following settings in Mach3.

These are found on the Mach3 menu:
Config -> General Config...

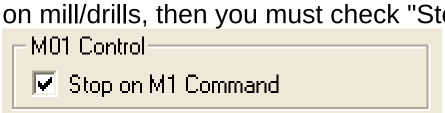



Check your Mach3 IJ Mode setting.

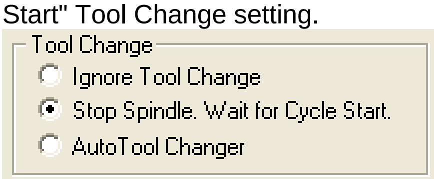



D2nc needs to be set the same way Mach3 is set. Both programs default to incremental. In D2nc the setting is on the Options -> Settings 30 menu.

If you plan on using D2nc's position/pause mode which allows for using the quill for drilling and tapping on mill/drills, then you must check "Stop on M1 Command".




To make Mach3 stop and wait when a tool number changes, set the "Stop spindle. Wait for Cycle Start" Tool Change setting.



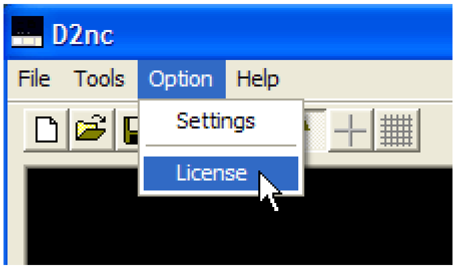

## 1.3 How to buy D2nc

D2nc has a built in 15 day trial key that activates when the program is first used. The trial period should allow sufficient time to determine if the program is useful to you. If the trial expires and you feel you need extra time to evaluate it, contact support@d2nc.com and request a trial key extension. These will be provided on a case by case basis.

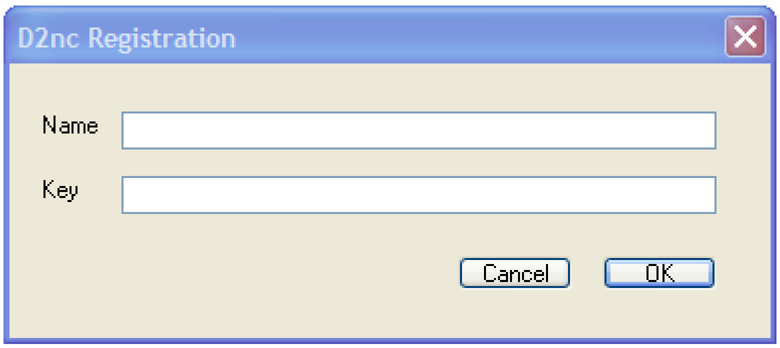
When the trial period expires, you may still use the program to define shapes but the "Show G-Code" and "Generate G-Code to File" buttons will cease to work.

Visit www.d2nc.com to purchase a license key. The key will be emailed to you shortly after payment is received.

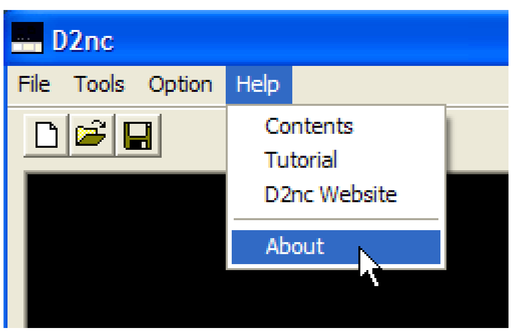
Select the menu Option->License




Enter the Name and Key as provided to you in the email




Confirm your registration by viewing the Help->About screen

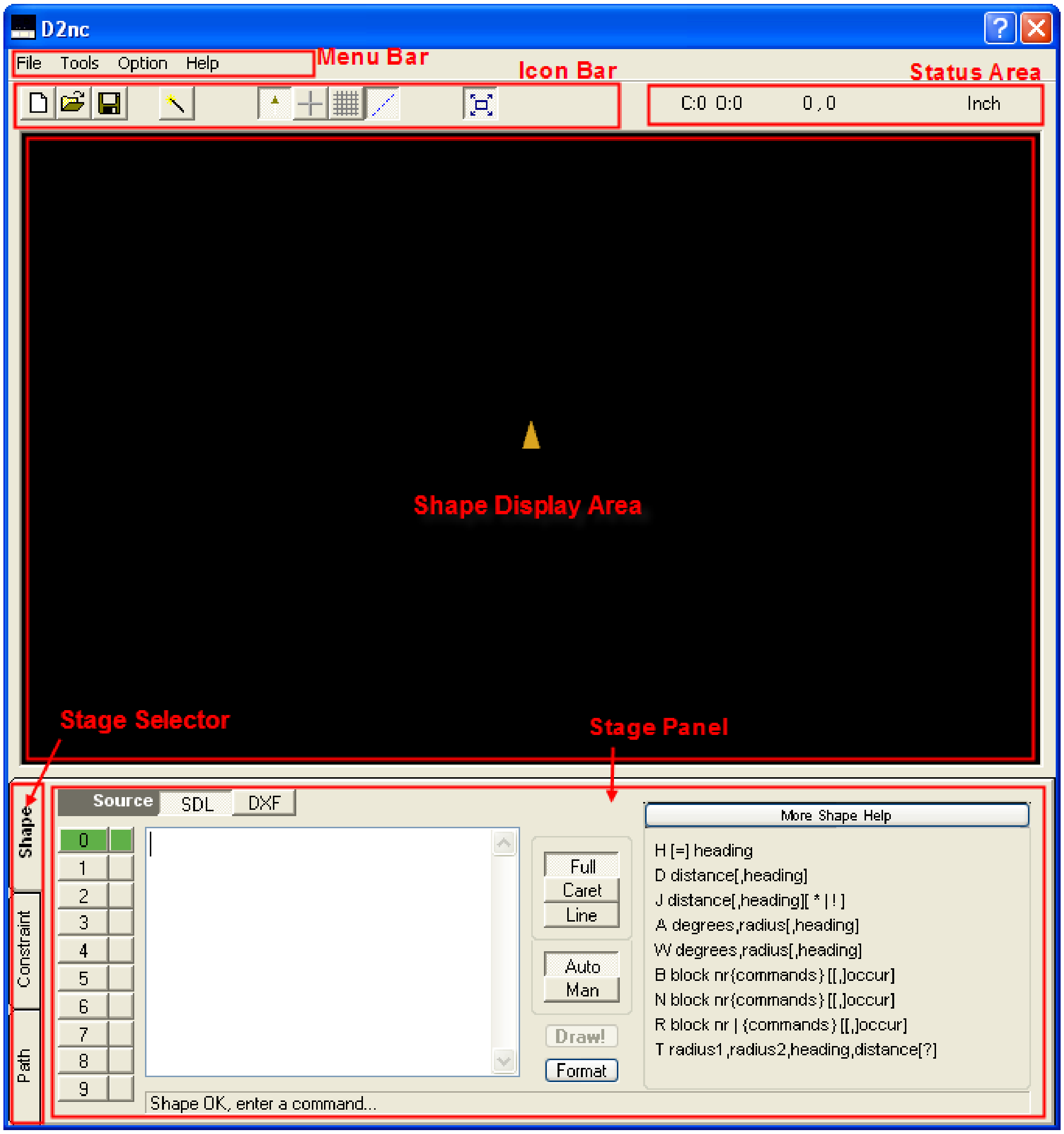



## 1.4 Copyright

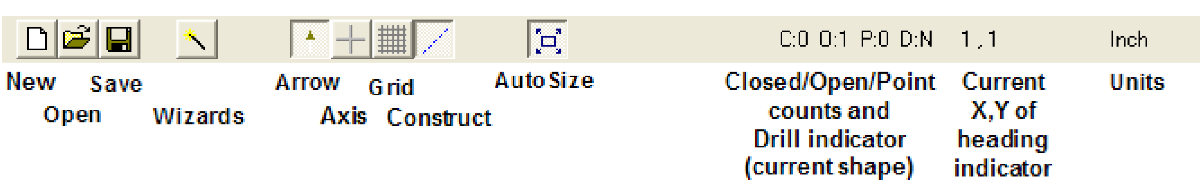
D2nc is copyright Graham Hollis. www.d2nc.com

Mach3 is copyright Artsoft. www.machsupport.com

# 2 The User Interface (Overview)




**Menu Bar**
Program Menu 22




**Icon Bar**
*New*
*Open*

*Save*
*Wizards* loads the SDL wizard selection screen.
*Arrow* shows/hides the yellow arrow heading indicator.
*Axis* shows/hides the X and Y axis and the zero intersect.
*Grid* shows/hides the display grid lines.
*Construct* shows/hides the red and blue jump and construction lines.
*Autosize* Auto/Manual resize of the shape in the display area.

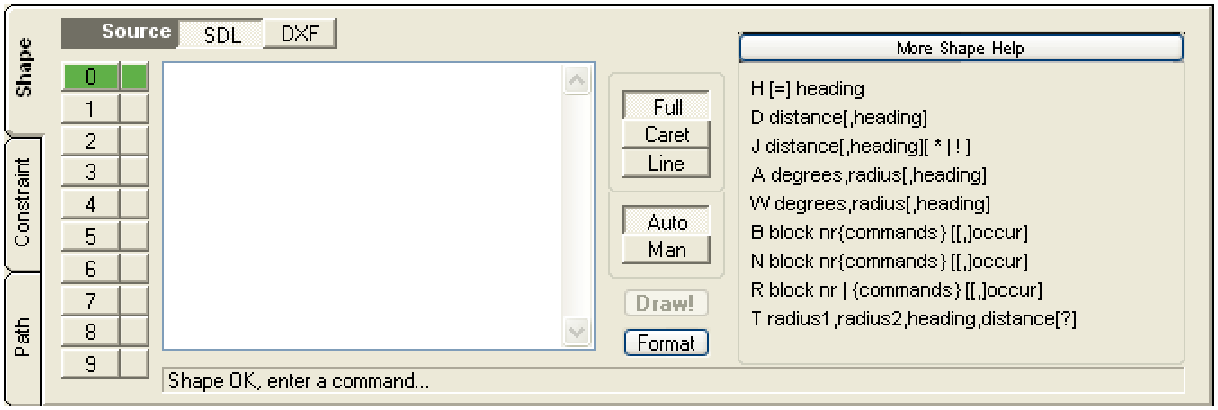
**Status Area**
C: shows the number of closed chains in the current shape.
O: shows the number of open chains in the current shape.
P: shows the number of points in the current shape.
(NOTE! : The first 3 counts are important as only a single closed shape with no open shapes and points can be offset using G41/G42. That is; offset buttons will only be enabled if the status is C:1 O:0 P:0)
D: indicator Y/N shows if the shape can be used for drilling. The drill operation buttons are only enabled if this is a Y.
X,Y shows the current location of the heading indicator in the shape display area.
Units shows the current working units. Inch or Metric. This is changeable in the settings 30 menu.

**Shape Display Area**
The black area is where shapes are displayed. This is an output only area. Shapes are defined using the Shape Description Language 91 while in the Shape 11 mode.
Using the Mouse.
*Right drag* - Hold the right mouse button down to drag the shape around.
*Shift left drag* or *scroll wheel* - Hold the shift key and left mouse button down and move the mouse up to zoom in or down to zoom out. This can also be done by clicking on the display area and rolling the mouse scroll wheel.

**Stage Selector**
There are three program stages which need to be followed to generate g-code. The stages are:
Describe Shapes 11
Set Constraints 14
Define Paths 15

**Stage Panel**
Activities for the current stage are carried out in this area. See *Stage Selector* above for descriptions of the stages.

## 2.1 Shape - SDL




**Shape Selector**
You can describe multiple shapes (up to 10 are supported) using the shape selector. Each shape described is separate from any other. The selector is in two parts, Active and Visible. Only one shape can be active at a time but any number can be visible. The visibility selector can be in three states:
green - visible
blue - hidden
red - hidden with parsing errors.

**Shape Description Area**
The shape description area is where you enter commands from the Shape Description Language to describe a shape. No formatting is required and all commands can run together. For readability and problem solving its recommended to enter one command per line. The control-f key sequence will reformat the area placing one command per line.

**Full/Caret/Line**
Each keystroke is evaluated and the resulting shape is displayed. By default the full shape is processed. By selecting Caret, the shape is displayed only up to the current cursor position. The Line option will display the shape up to the end of the current cursor line. These two options are a great way to debug your shape description if its not producing the shape you think it should.

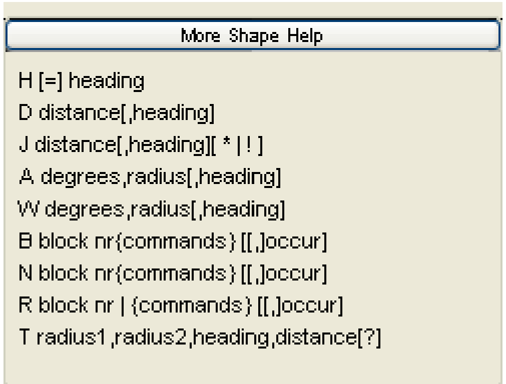
**Auto/Manual**
In auto mode the shape is redrawn after each keystroke. In manual mode the shape is redrawn only when the Draw! button is pressed.

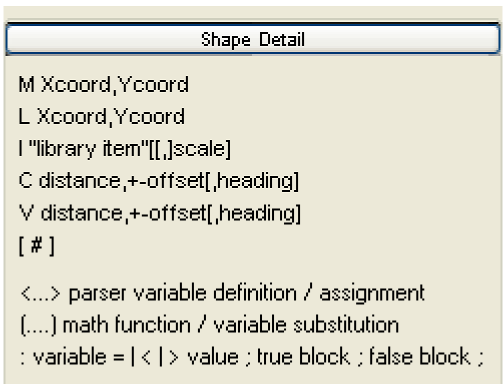
**Draw!**
When in manual mode, the Draw! button processes the full shape description.

**Format**
The format button restructures the SDL input to one command per line.

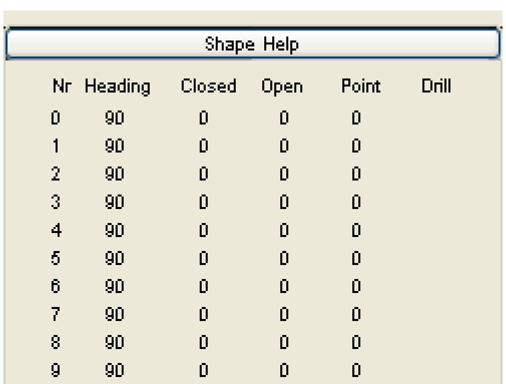
**Panel Selector button (Shape Detail / Shape Help)**
The selector button cycles through the following panels:

**Shape Help**
A quick reference to the complete Shape Description Language. For more detailed help on any one command, click on that command.



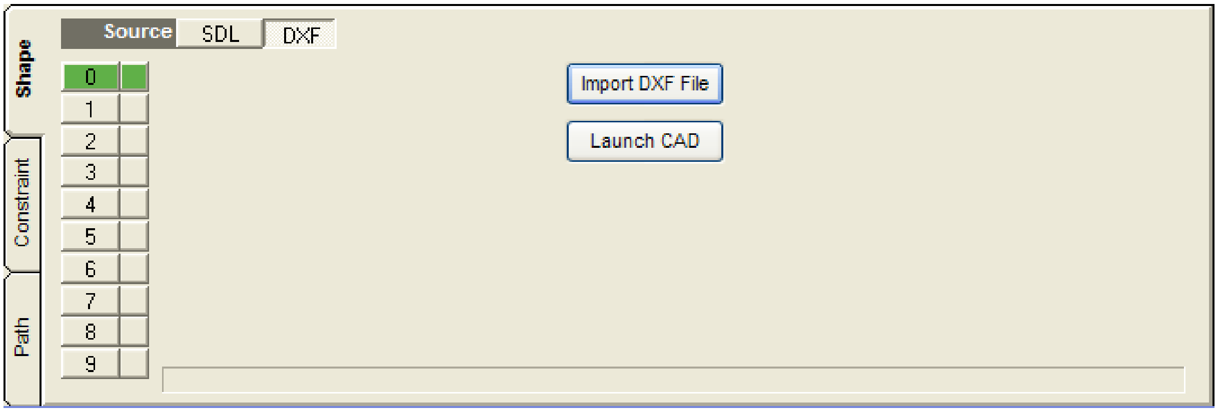

**Shape Detail**
For each shape, the table shows the absolute heading in the unit circle, the number of closed and open chains in the shape, the number of points in the shape and if the shape can be used for drilling. For the rules on how shapes can be used see the path [15] help .


Shape Help

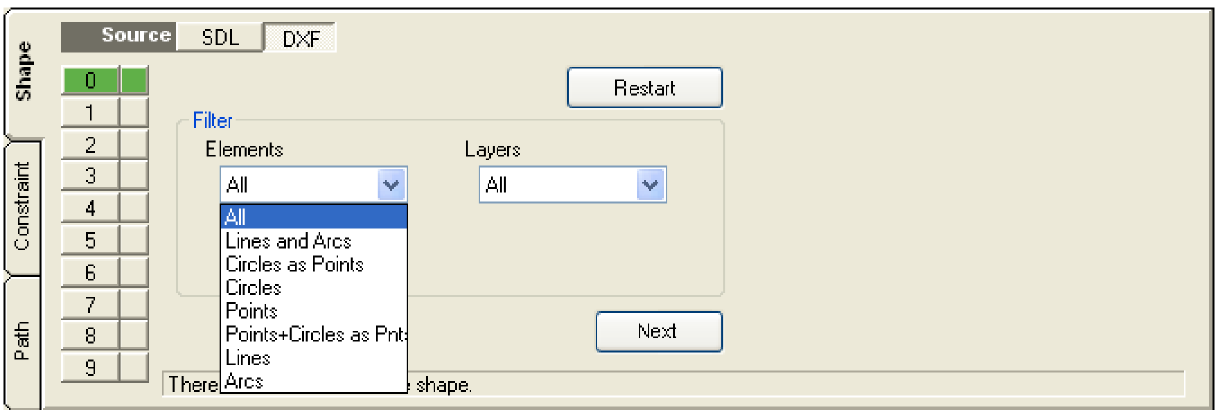
| Nr | Heading | Closed | Open | Point | Drill |
|---|---|---|---|---|---|
| 0 | 90 | 0 | 0 | 0 | |
| 1 | 90 | 0 | 0 | 0 | |
| 2 | 90 | 0 | 0 | 0 | |
| 3 | 90 | 0 | 0 | 0 | |
| 4 | 90 | 0 | 0 | 0 | |
| 5 | 90 | 0 | 0 | 0 | |
| 6 | 90 | 0 | 0 | 0 | |
| 7 | 90 | 0 | 0 | 0 | |
| 8 | 90 | 0 | 0 | 0 | |
| 9 | 90 | 0 | 0 | 0 | |

**Message Line**
The message line will give information about any error in the SDL syntax.

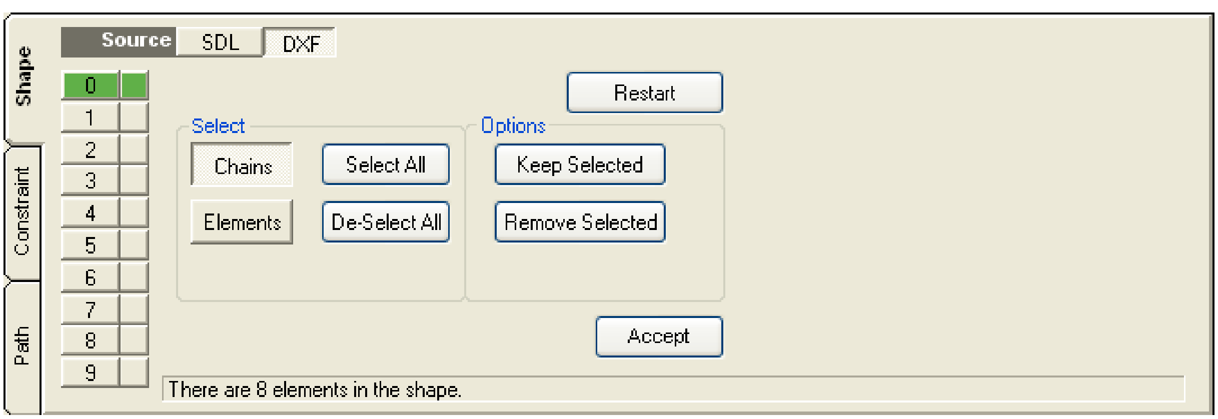
## 2.2 Shape - DXF



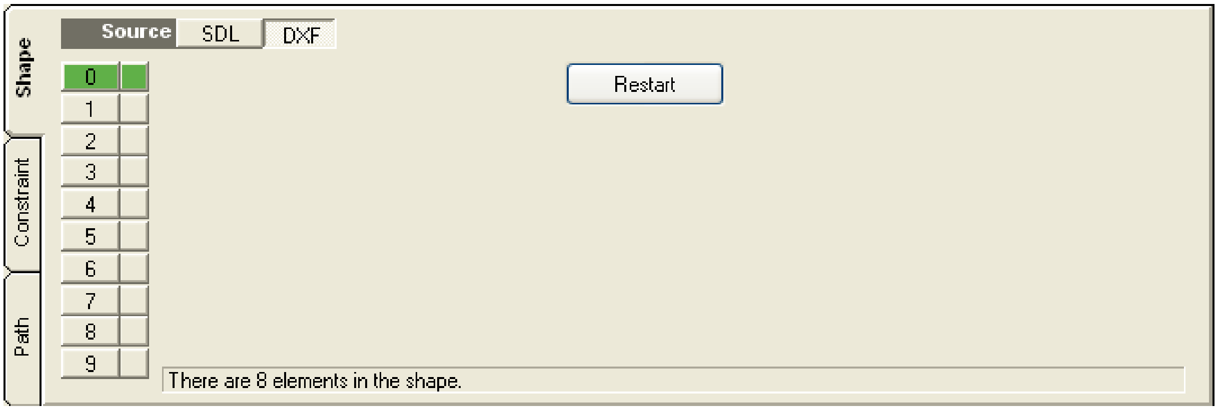

The DXF Shape source initial screen. Either import a DXF file or launch your CAD program to create or modify a DXF file. The CAD launch button can be enabled in the settings 30.




DXF element filter screen. Filters what elements and from what layers elements are imported.

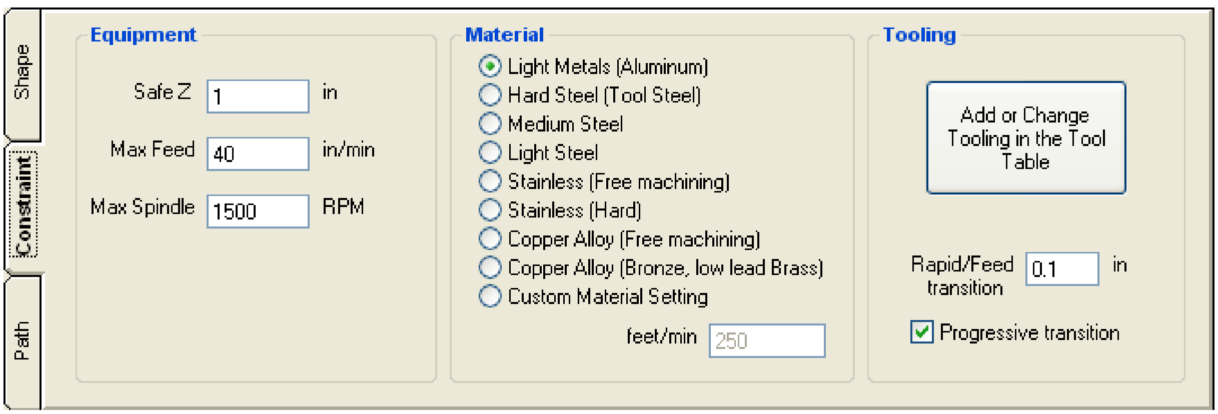



Chain and element selection screen allowing you to keep or delete selected items. Click on accept to lock the shape in.




Once a shape has been locked in you may restart the DXF import from the beginning

## 2.3 Constraint




There are three types of constraints.

**Equipment**

The equipment constraints describe the limits of your machine in feed rate and spindle speed. These are used to limit the maximum speed or feed calculated in the g-code during the next step.
The safe Z is the height that the tool will be for all rapid moves. The top of the material to be cut is always at a Z axis value of zero. Any cutting moves at feed rate will be with Z less than zero. Rapid moves will occur with Z greater than zero at the safe Z height.

**Material**

There is list of pre-defined materials, each of which have a predefined surface feet or meters per minute as a cutting speed. It is also possible to select the "Custom Material Setting" and enter your own surface speed for the material you are cutting. The surface speed is used to calculate the speed and feed in the g-code during the next step.

**Tooling**

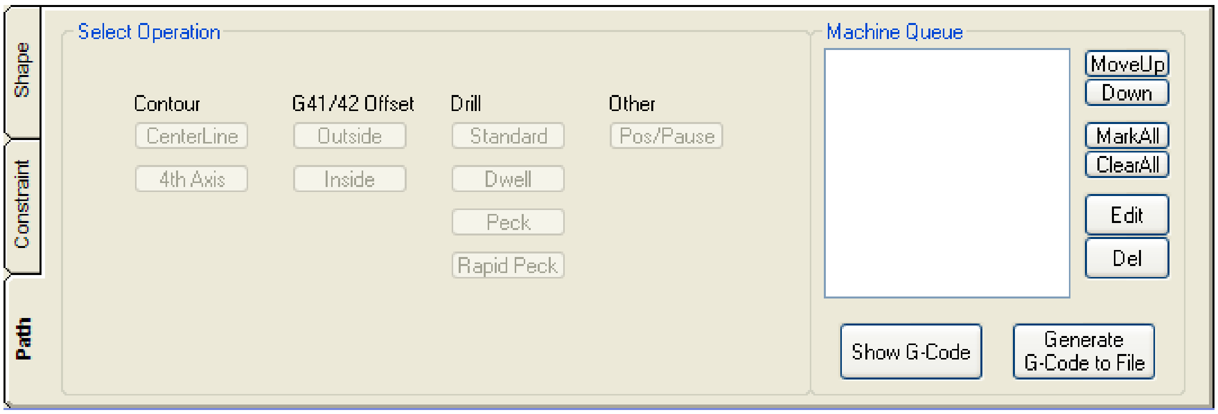
D2nc supports a tool table 28 to build a library of tools. Tools added into the tool table can be selected during the next step when defining paths. Various aspects of the tool described in the tool table are used to calculate the speed and feed and what type of operation the tool is suitable for.
While defining tools is optional it is recommended for the following reasons:

1. If there is no tool defined in the table you are asked to enter a tool diameter which is assigned to the default tool 0. If you define several machine operations and each operation uses a different diameter, the g-code will have the same tool 0 for each operation making tool changes a problem.

2. Tool 0 uses a generic set of parameters and may not produce the optimal speed and feed.

Rapid feed transition is the height, during a move downwards, at which a Z axis will change from the rapid rate to the plunge rate that is defined in the path definition. If progressive transition is checked, the transition height will move down by the step size on each subsequent pass.

## 2.4 Path




On the left side of the panel you will see buttons for each possible operation which can be used to define a path. On the right side of the panel is the Machine Queue 20.

You will see some are greyed out and some are selectable. Those for which there is no shape described that would allow their use are greyed out.

The rules for greyed out buttons are as follows:

- The center contour can be used for all shapes that have any number of open and closed entities in them.
- The G41/G42 offset can only be used on shapes comprised of a single closed chain with no points. When defining a shape the status must be C:1 O:0 P:0 to use offsetting.
- The drill group of buttons can only be used on shapes which have no linear movement. A drill shapes consists of a series of jumps and/or warps with "d0" (draw point) cutting strokes.

### Operations

#### Contour Operations

**Centerline** 18
Centerline has the center of the tool follow the shape outline. The diameter of the tool used must be taken into account when defining the shape.

**4th Axis** 18
Wrap either the X or Y axis of a shape (SDL or DXF sourced) around the A axis with optional scaling and aspect ratio adjustment.

#### G41/G42 Offset Operations

**Outside** 18
The path is offset to the outside of the shape using G41/G42 offsetting. The offset is the radius of the tool plus any stock allowance.

**Inside** 19
The path is offset to the inside of the shape using G41/G42 offsetting. The offset is the radius of the tool plus any stock allowance.

**Drilling Operations**

**Standard** 19
G81 canned cycle generated g-code

**Dwell** 19
G82 canned cycle generated g-code

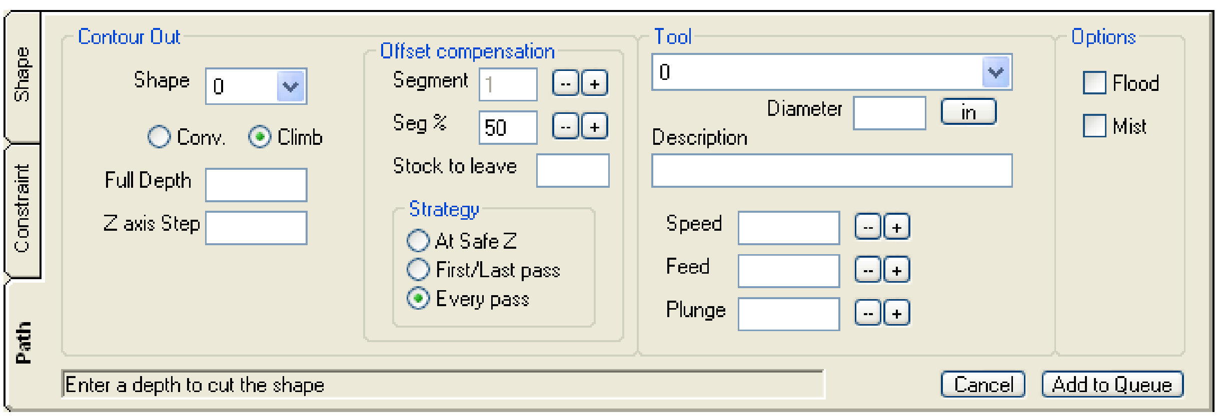
**Peck** 19
G83 canned cycle generated g-code

**Rapid Peck** 20
G73 canned cycle generated g-code

**Other Operations**

**Position and Pause** 20
Creates a series of G00 and M01 in g-code for each point in a drill shape. Useful for mill/drill machines with a quill allowing the positioning over a hole for manual drilling or tapping using a tapping head. No Z axis moves are generated for this path.

## 2.5 Operations




Each operation has its own unique panel. The contour Out panel is shown above.

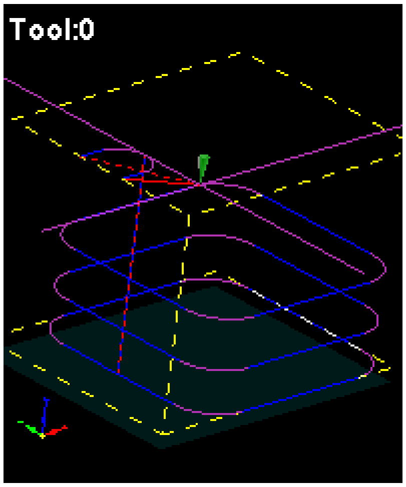
Generally the panels are divided into two parts. The operation parameters on the left and the tool parameters on the right

Once all entries have been made, click on the Add to Queue (or Update Queue if in edit mode) to add the shape to the Machine Queue 20

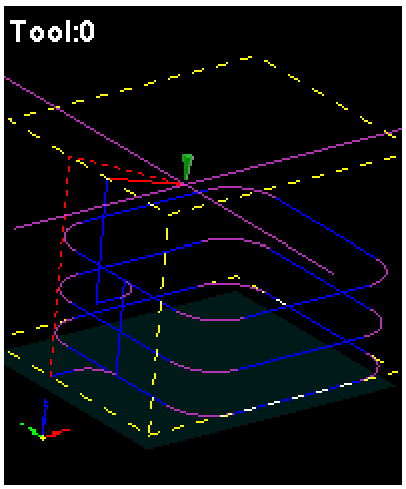
**Offset compensation Strategy**
The offset compensation strategy is only available for the G41/G42 inside and outside offsetting operations and has the following three selectable options:

**At Safe Z**

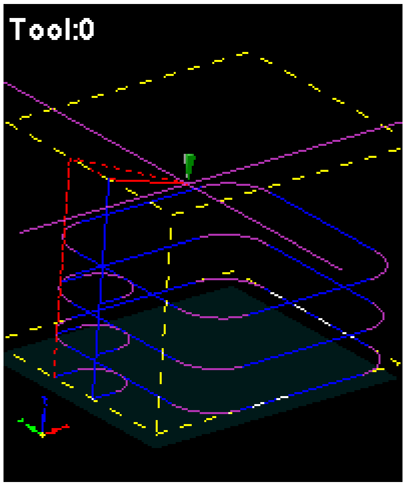
The compensation occurs at Safe Z height. When the tool plunges into the material it is in an already compensated state. Once the final pass has completed, the tool retracts to Safe Z before cancelling offset compensation.



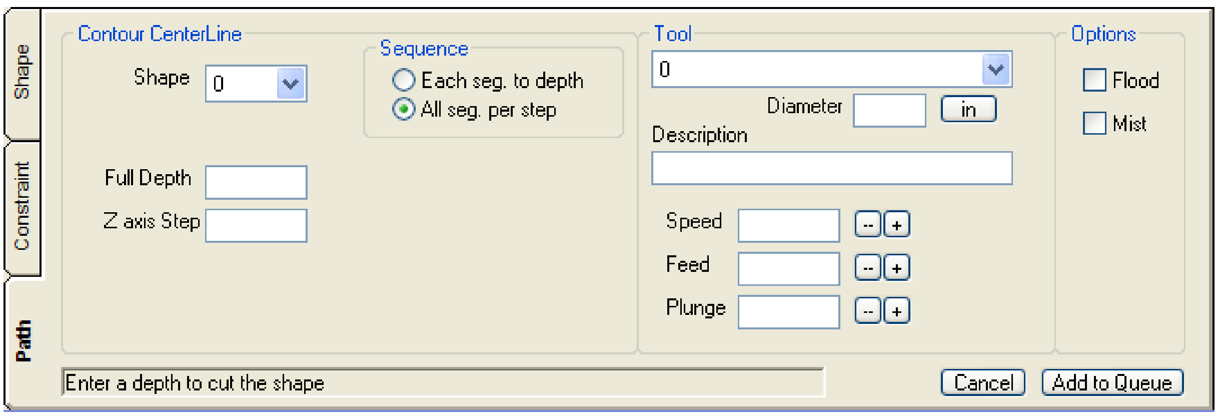

**First/Last pass**
The compensation moves occur on the first and last pass only and at the material level. This should not be used where these is a possibility of the exit compensation move occurring at full depth into material. That would include any inside offsets and outside where the contour was not on the edge of the material.



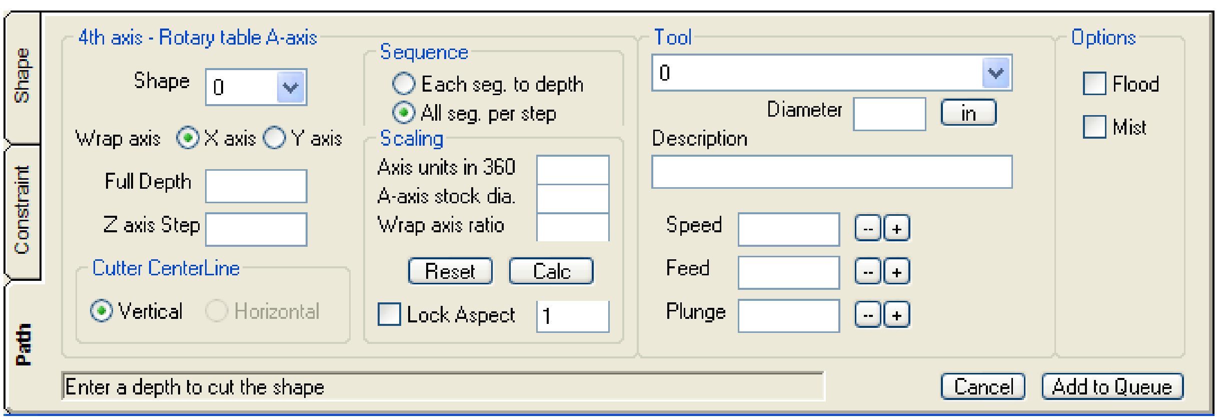

**Every pass**
This is the default setting and although slower due to the repeated lift and plunge, it should leave the cleanest finish.



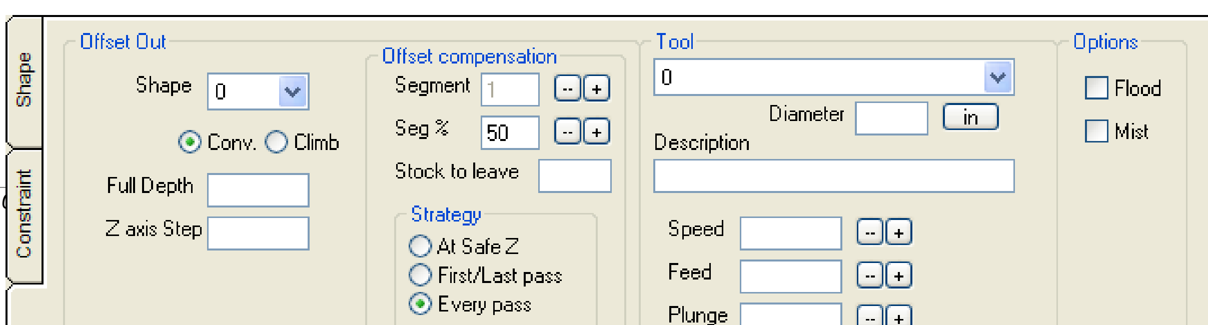

## 2.5.1 Contour Centerline



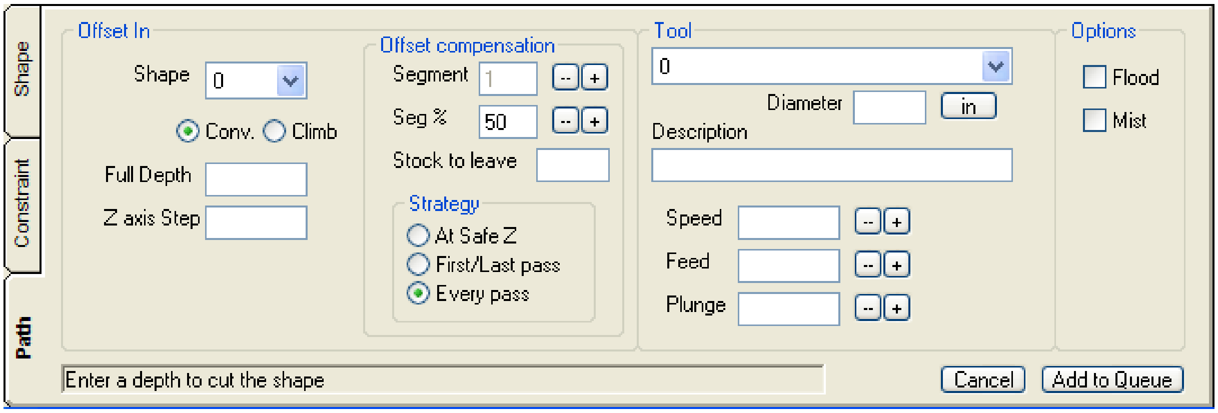

## 2.5.2 Contour 4th Axis



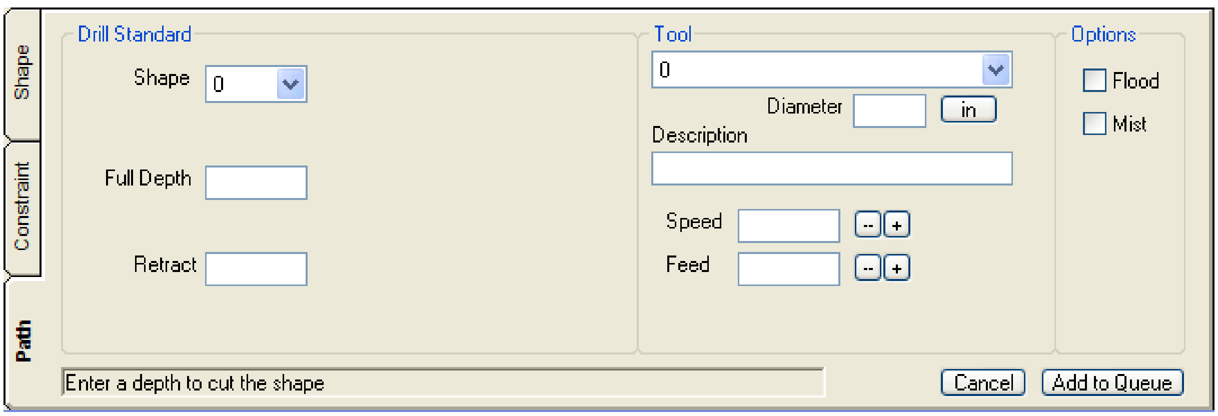

## 2.5.3 G41/G42 Offset Outside



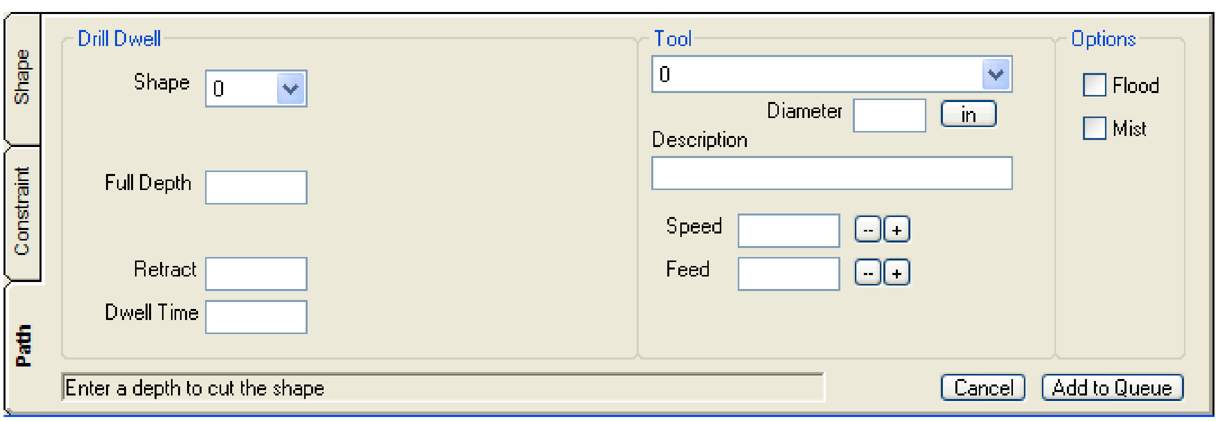
### 2.5.4 G41/G42 Offset Inside



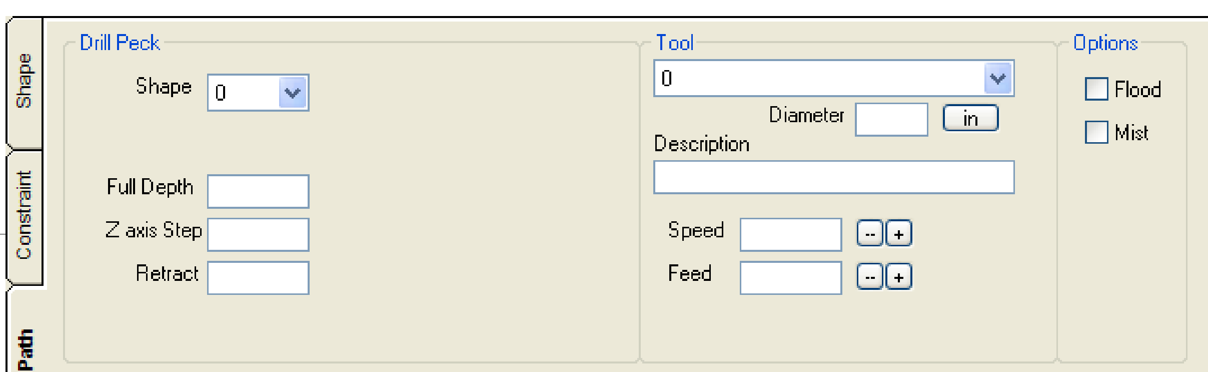

### 2.5.5 Drill Standard



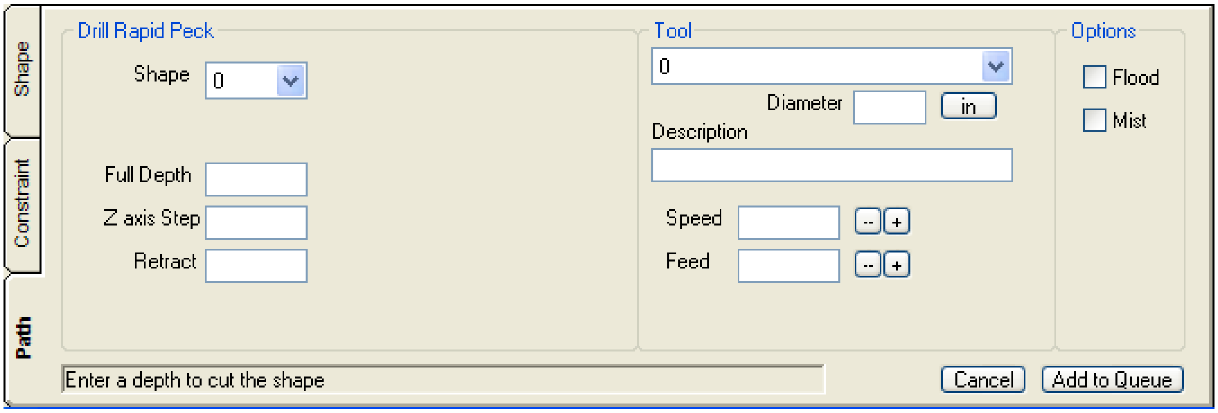

### 2.5.6 Drill Dwell



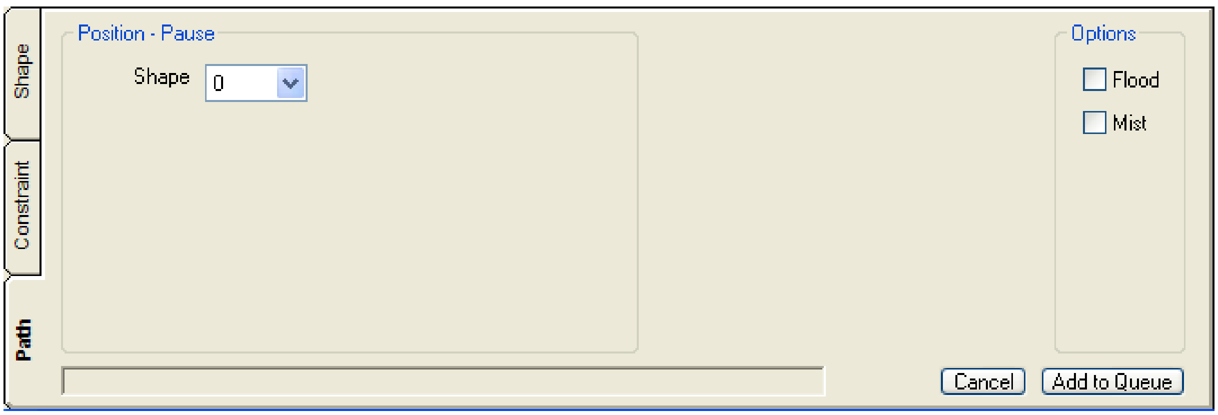

### 2.5.7 Drill Peck



### 2.5.8 Drill Rapid Peck

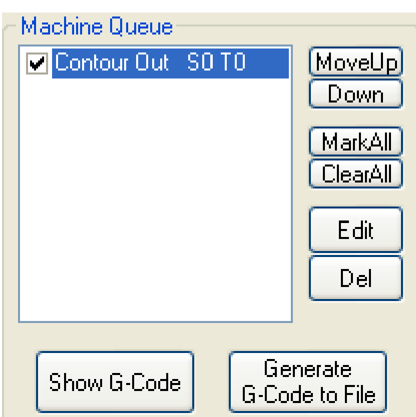



### 2.5.9 Other Position and Pause




Used for hand operation of the quill. X and Y axis will position to the hole and then pause while you hand operate the quill for center drilling, drilling, reaming or tapping head operations. Start will then position to the next hole in the sequence.

## 2.6 Machine Queue




The Machine Queue allows the possibility to perform multiple operations in one g-code program. Either as multiple operations on one shape or several operations on multiple shapes to produce one part.

An example of the former is a contour roughing pass. You may leave a few thousandths allowance in the first queue operation and  in a second, make a finishing pass at full depth with no allowance.

An example of the latter would be three operations on three shapes to produce a drilled flange. For example, an inside contour for the cut on shape 1, a drill cycle on a bolt circle shape 2 and finally an outside contour of shape 0.

Entries in the Machine Queue show three items of information:
Operation type, Shape Nr, Tool Nr

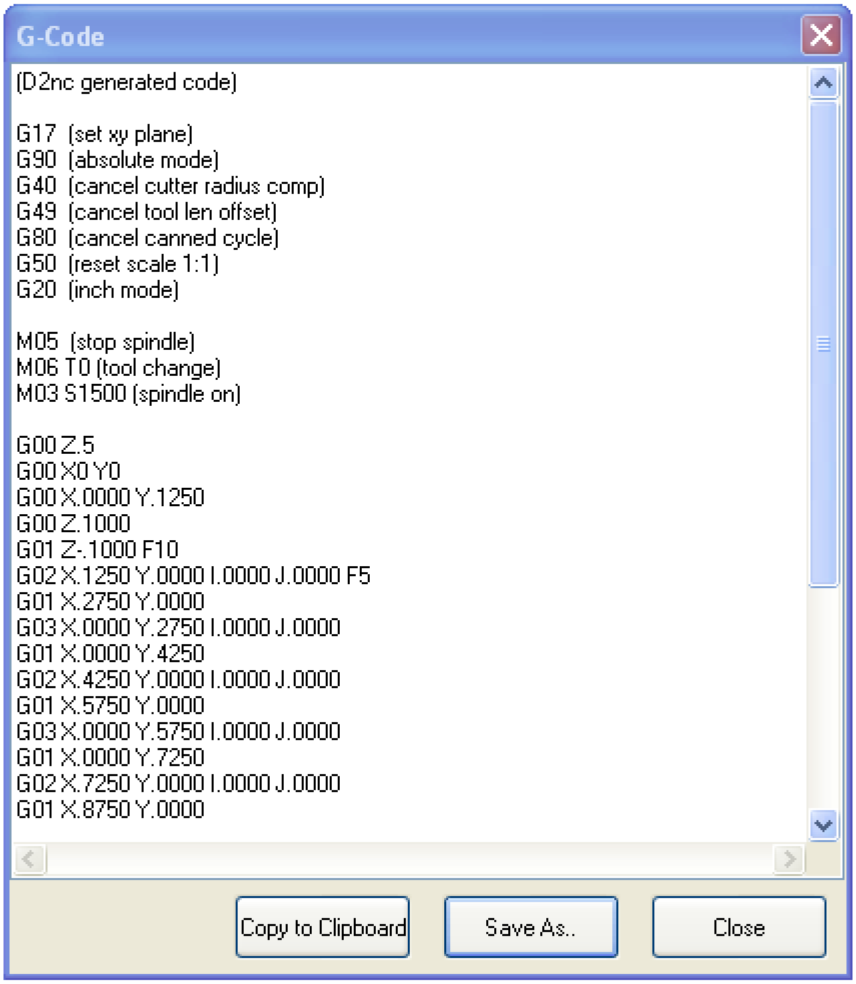
Only the machine queue items that are checked on the left will be included in the g-code generated. You may define several items in the queue and then machine them sequentially one at a time by checking and unchecking the appropriate queue entries.

Entries in the queue can be reordered by highlighting the entry to move and click the "MoveUp" and "Down" buttons.

**Generate G-code to File**

This button generates g-code for each entry checked in the machine queue and writes it to a file. The output file name is set in the menu item Settings30. This button will not work if a license is not installed or the trial period is over.

**Show G-Code**




This button generates g-code for each entry checked in the machine queue and displays it in the g-code window. From here you can make adjustments to editable g-code and then copy it to the windows clipboard or save it to a new file. This button will not work if a license is not installed or the trial period is over.

## 2.7 Menu

This is the full menu structure in D2nc. Click on the links below to learn about those features where available.

**File**

- **New**
- **Open..**
- **Save**
- **Save As...**
- **Export DXF...** 33
- **Exit**

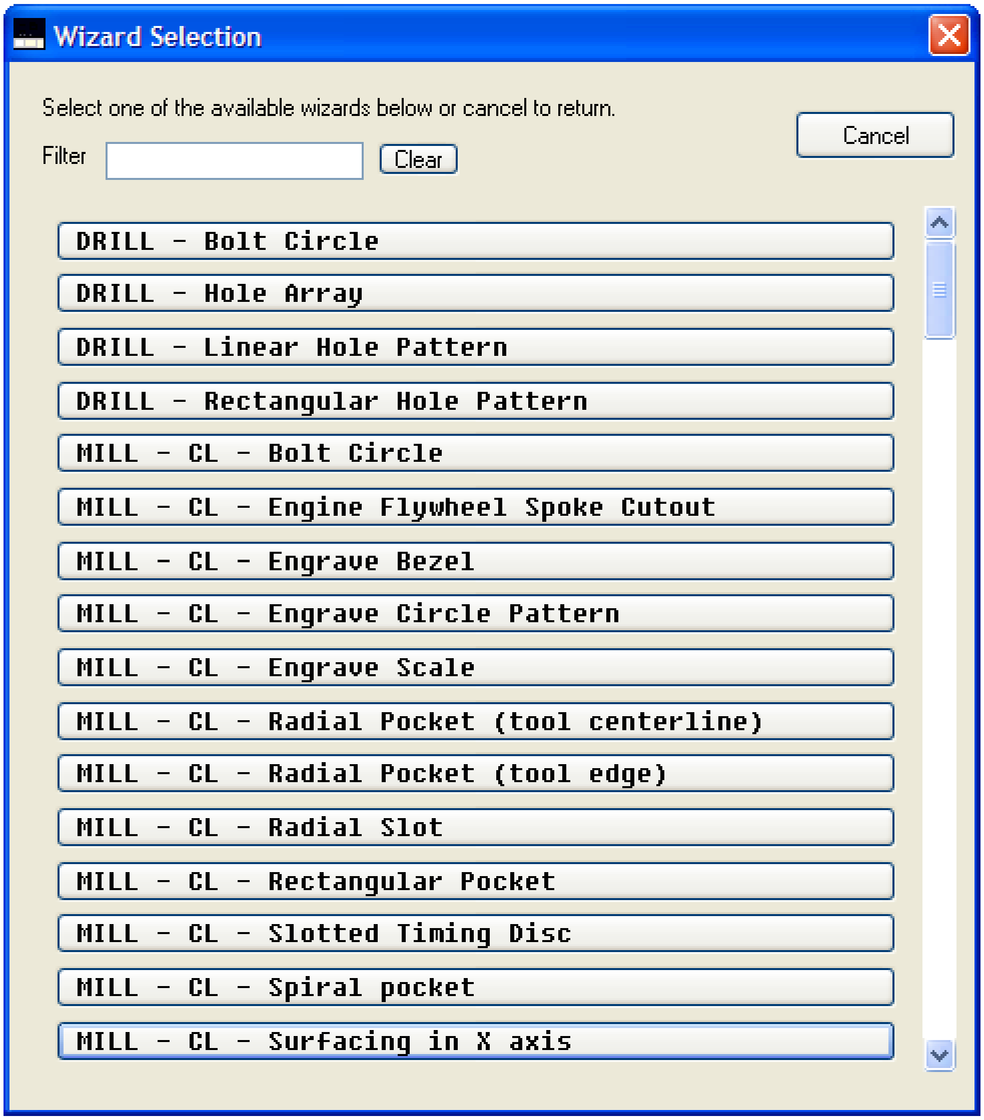
**Tools**

- **SDL Wizards** 23

**Shape Library** 26
**Tool Table** 28
**Material Settings** 33
**Option**
**Settings** 30
**License** 7
**Help**
**Contents**
**Tutorial**
**D2nc Website**
**About**

### 2.7.1 SDL Wizards

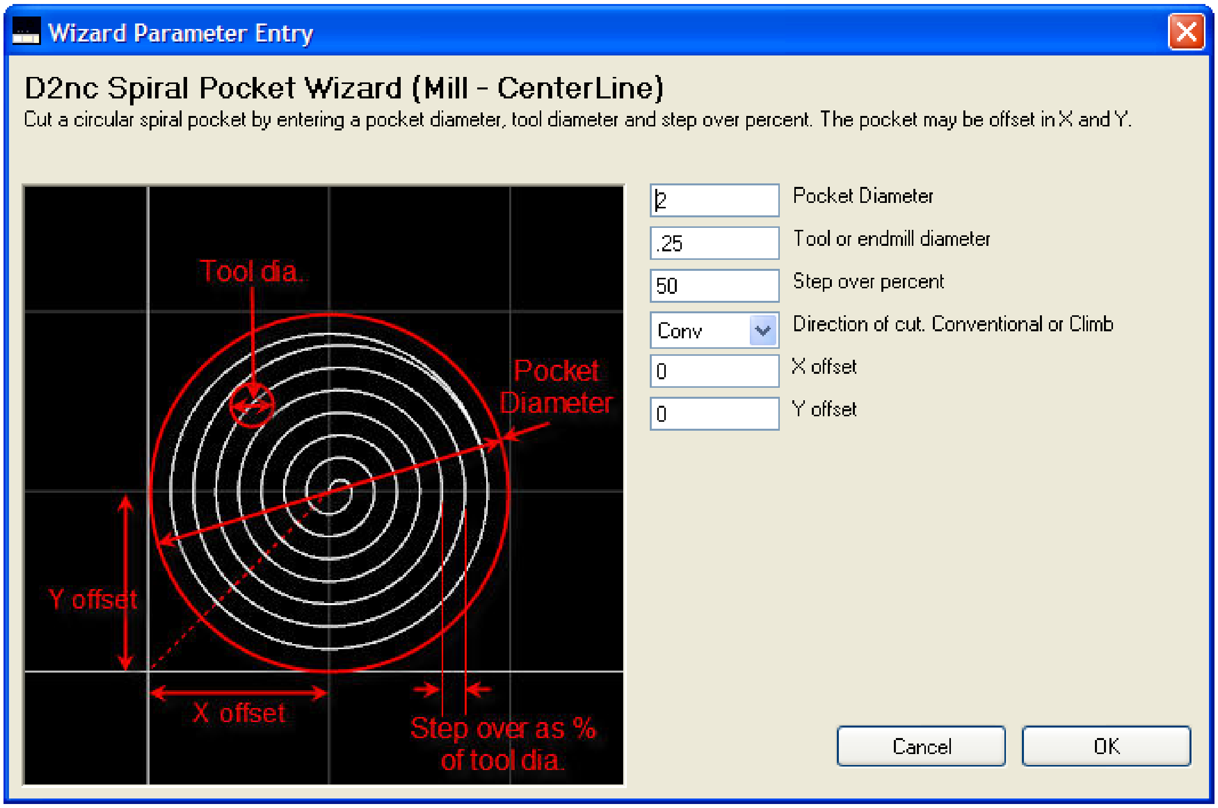
The SDL Wizards in D2nc take advantage of D2nc version 2's SDL extensions of variables and functions.



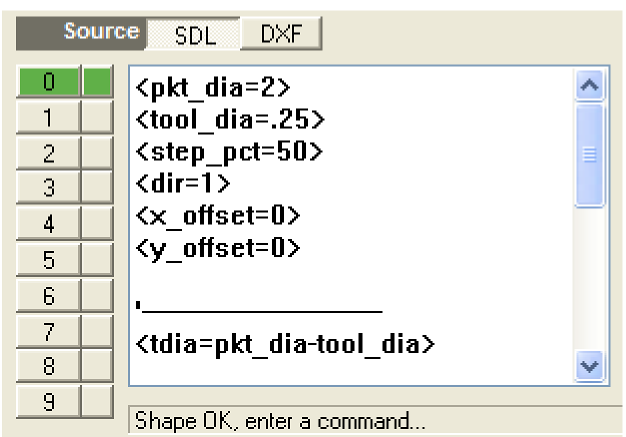

A complete list of available wizards and sample output can be seen in the Wizard Directory 50

Each wizard is assigned its own button. If there are more wizards than can fit on the screen, the list can be scrolled top to bottom. To find a particular wizard, start typing any sequence of letters you know occur in the wizard name and the list will dynamically update and only show wizards containing that string of letters. You can click the Clear button to remove the filter and show all the wizards.

Wizards are easy to create if you have worked with a programming language and understand XML. Refer to the guide to creating your own wizard 46. Click on a wizard in the selection screen to load the parameter entry screen.

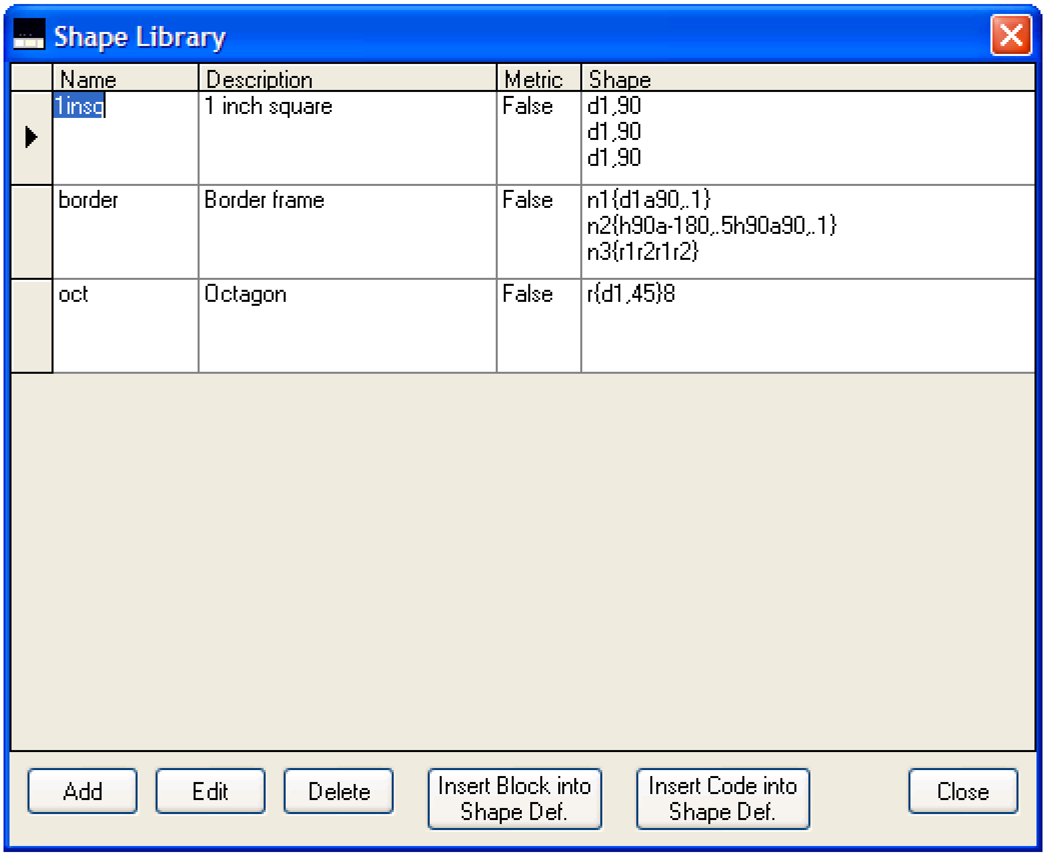


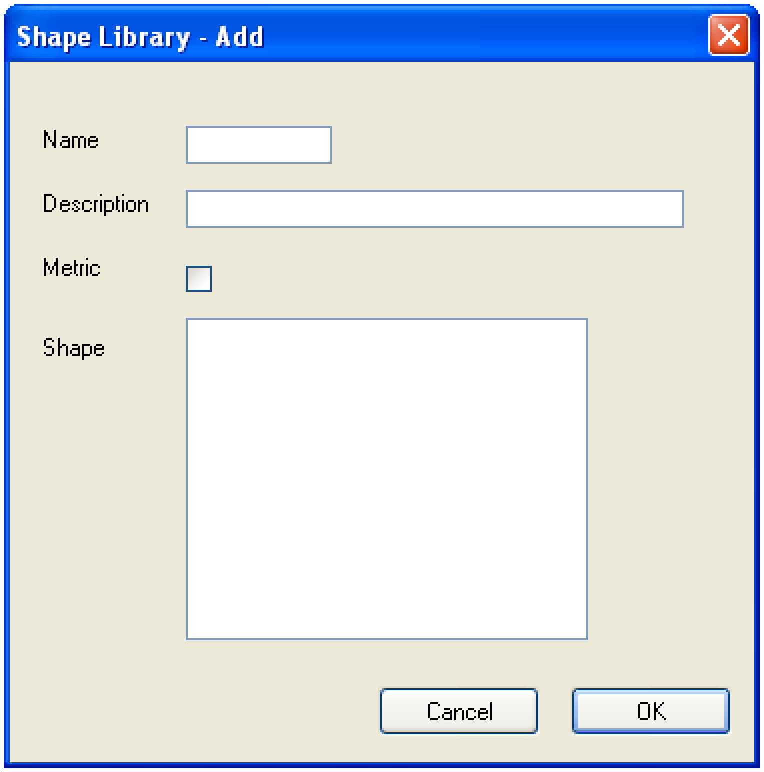

Enter the desired values for the available parameters of the wizards you selected. The wizard may display a graphic illustrating the basic wizard function and the meaning behind the individual input parameters. Once you satisfied with the input, click the "OK" button to generate the SDL into the shape description area.

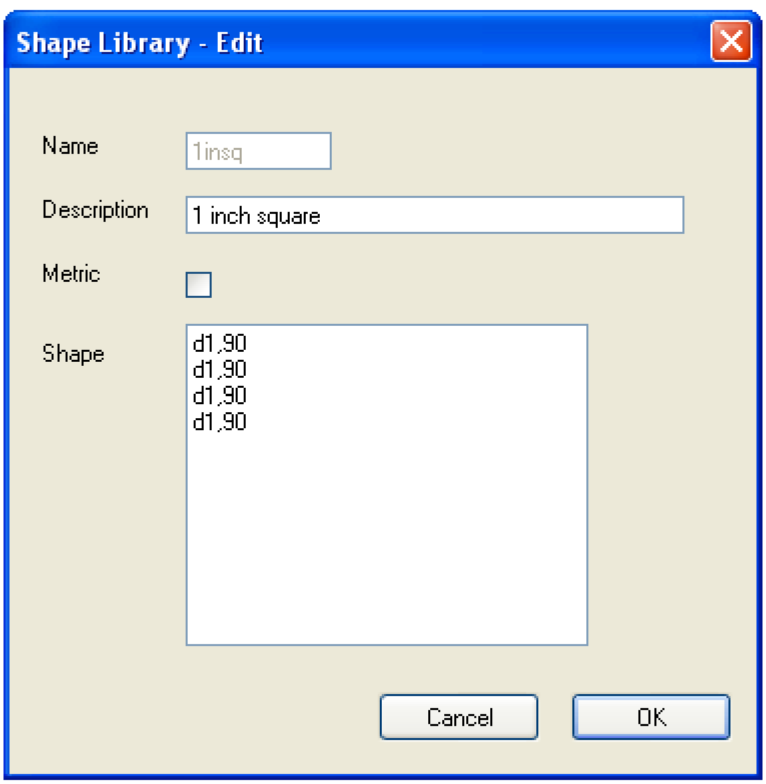



The SDL program in the shape description generated by the wizard is a live program and may be adjusted before proceeding to set the constraints 40 and define a tool path 40. The convention for wizards is to assign 91 the value input to a variable, one input parameter per line. The input variables are separated from the static part of the SDL program by a dashed comment line. You may adjust the values of the variables to make changes to the displayed shape or you can re-run the wizard to input new values.

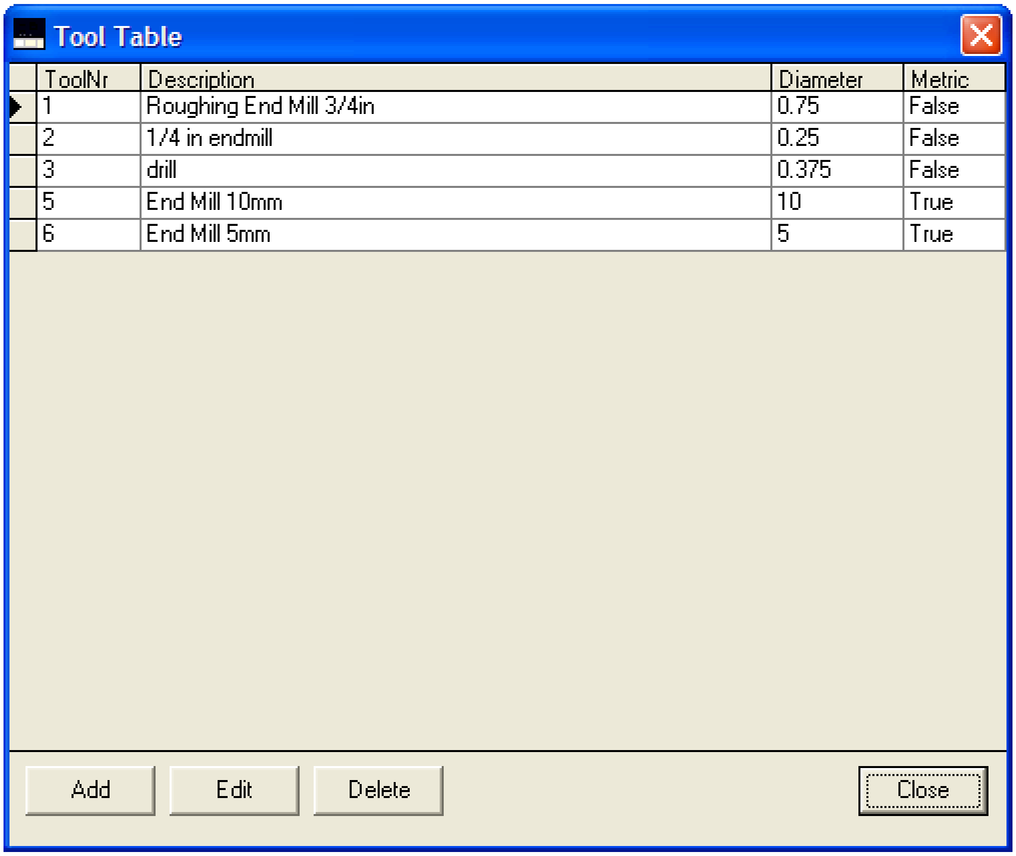
## 2.7.2 Shape Library

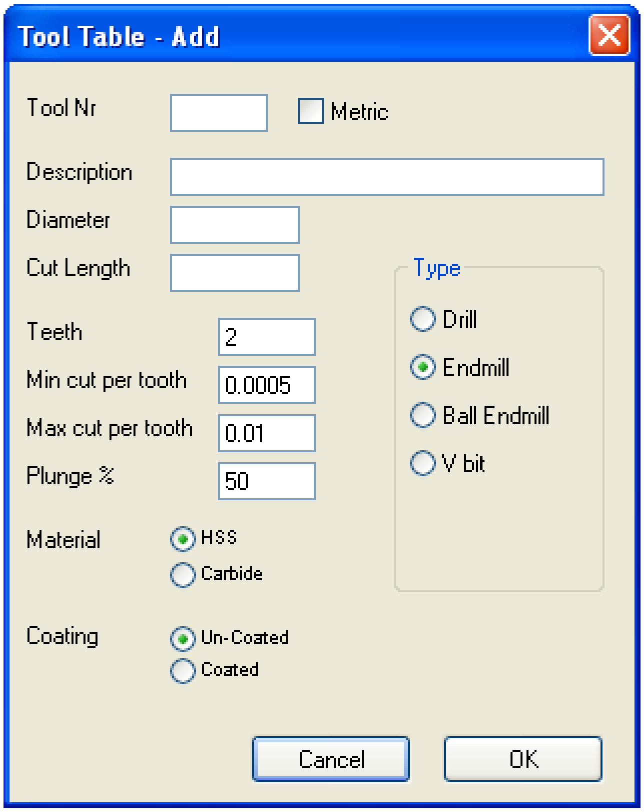


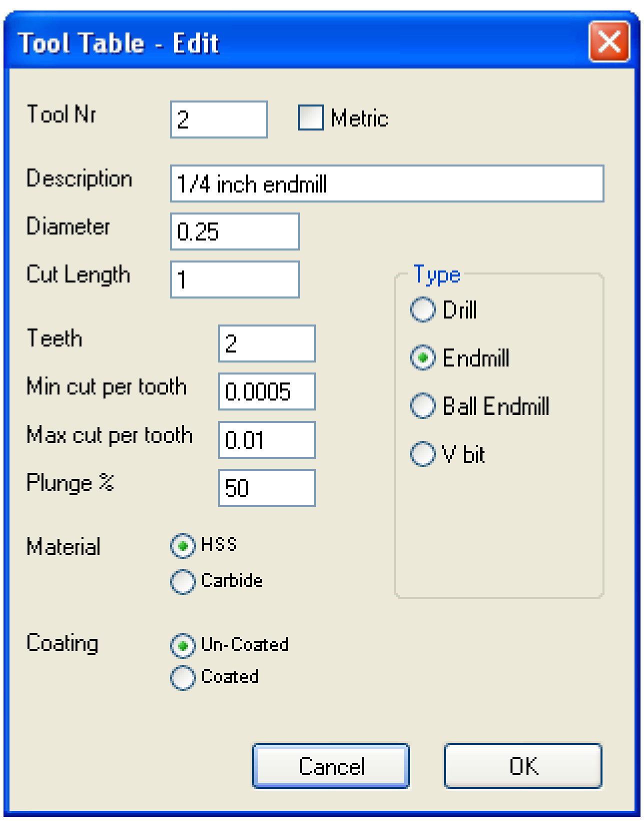

Shape Library - Add
Name
Description
Metric
Shape
Cancel
OK


Shape Library - Edit
Name
1insq
Description
1 inch square
Metric
Shape
d1,90
d1,90
d1,90
d1,90
Cancel
OK

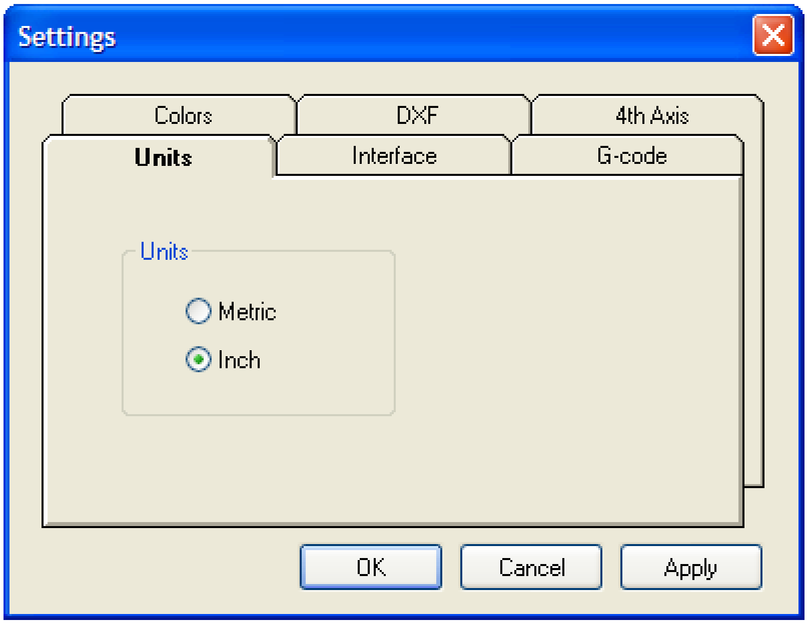
### 2.7.3 Tool Table



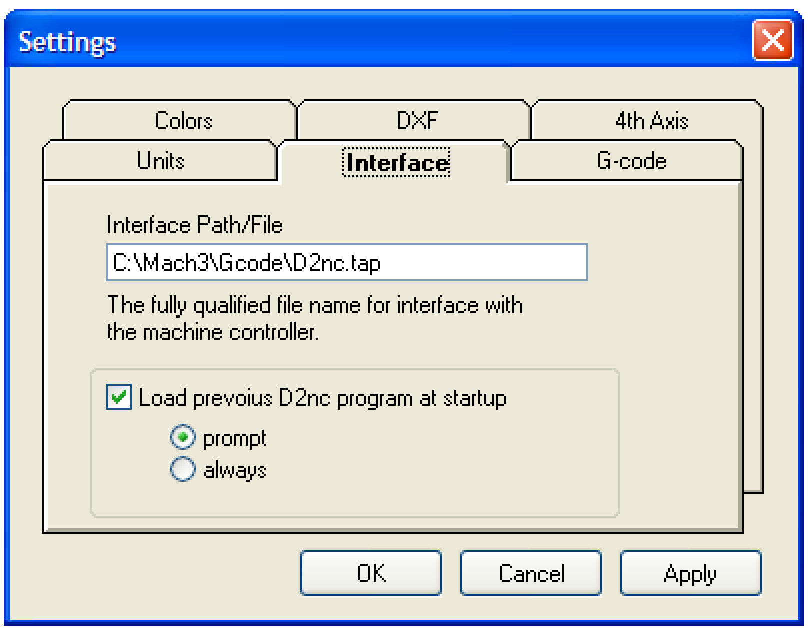

Tool Table - Add
Tool Nr
Metric
Description
Diameter
Cut Length
Type
Drill
Endmill
Ball Endmill
V bit
Teeth
2
Min cut per tooth
0.0005
Max cut per tooth
0.01
Plunge %
50
Material
HSS
Carbide
Coating
Un-Coated
Coated
Cancel
OK


Tool Table - Edit
Tool Nr
2
Metric
Description
1/4 inch endmill
Diameter
0.25
Cut Length
1
Type
Drill
Endmill
Ball Endmill
V bit
Teeth
2
Min cut per tooth
0.0005
Max cut per tooth
0.01
Plunge %
50
Material
HSS
Carbide
Coating
Un-Coated
Coated
Cancel
OK

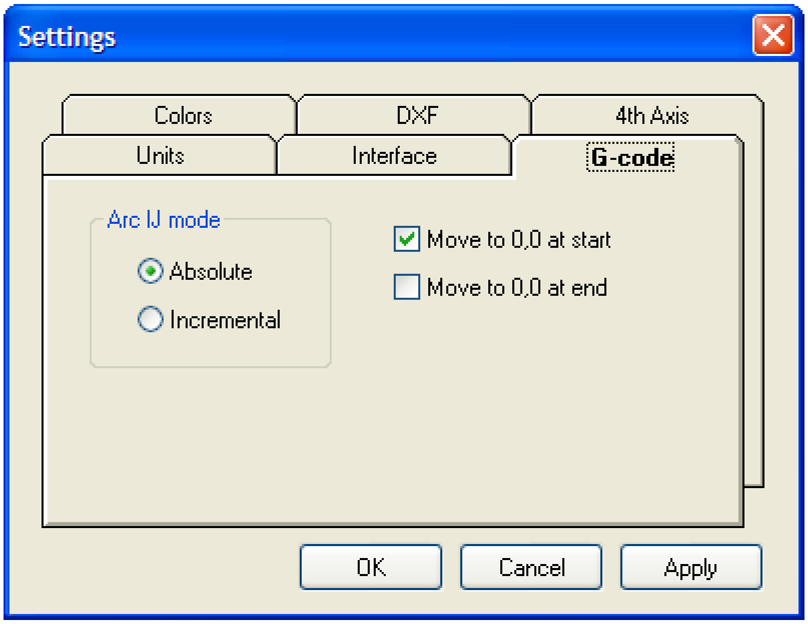
### 2.7.4 Settings



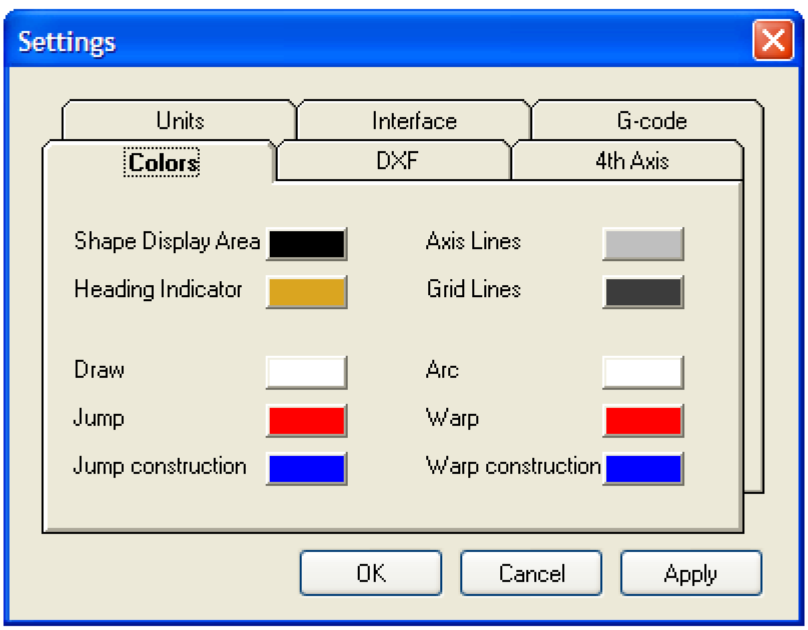

Select the units you will working in.



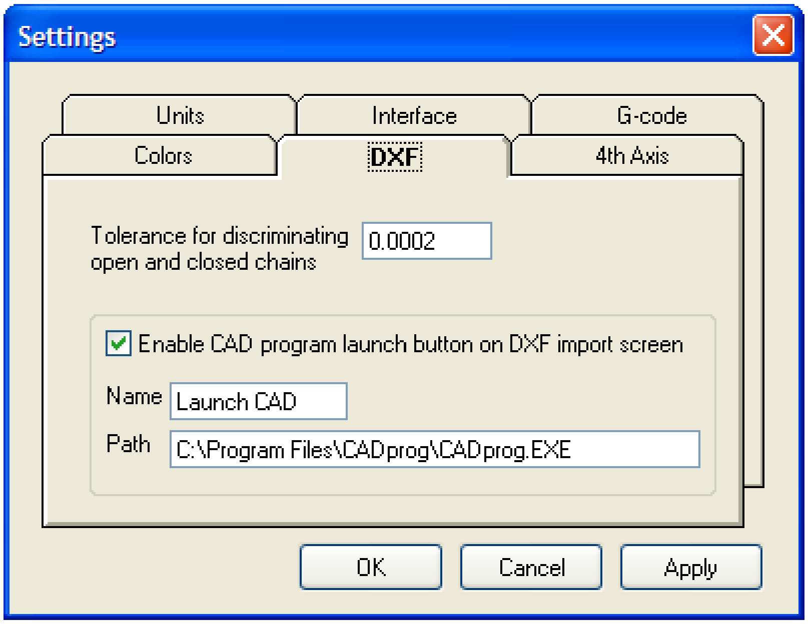

The file name that the g-code is saved to when you click the "Generate G-Code to File" button
Set the option of loading the previous program you were working on at startup. Additionally you can be prompted to make a Yes or No decision to load the previous program or just load it every time.



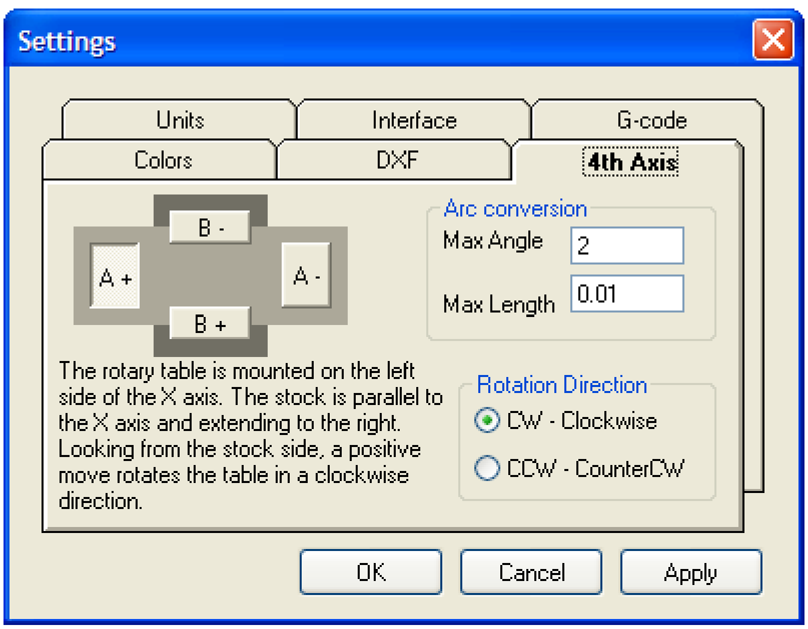

This option sets how the I and J components of G02 and G03 are generated. Either as an absolute coordinate or incremental to the X and Y coordinate of the G02 or G03 move. This setting **must** match the same setting in Mach3. See installation 5 for further information. If these settings are different all arcs will look strange and may manifest itself by showing arcs going in the wrong direction or as mirror images of what they should be. Be aware that some of the Mach3 built-in wizards may change the setting in Mach3 and cause it to be out of sync with the setting in D2nc.
Move to 0,0 at start and at end will include these moves in the generated g-code if checked.




You may customize the colors by clicking on a swatch and selecting a new color from the picker presented.



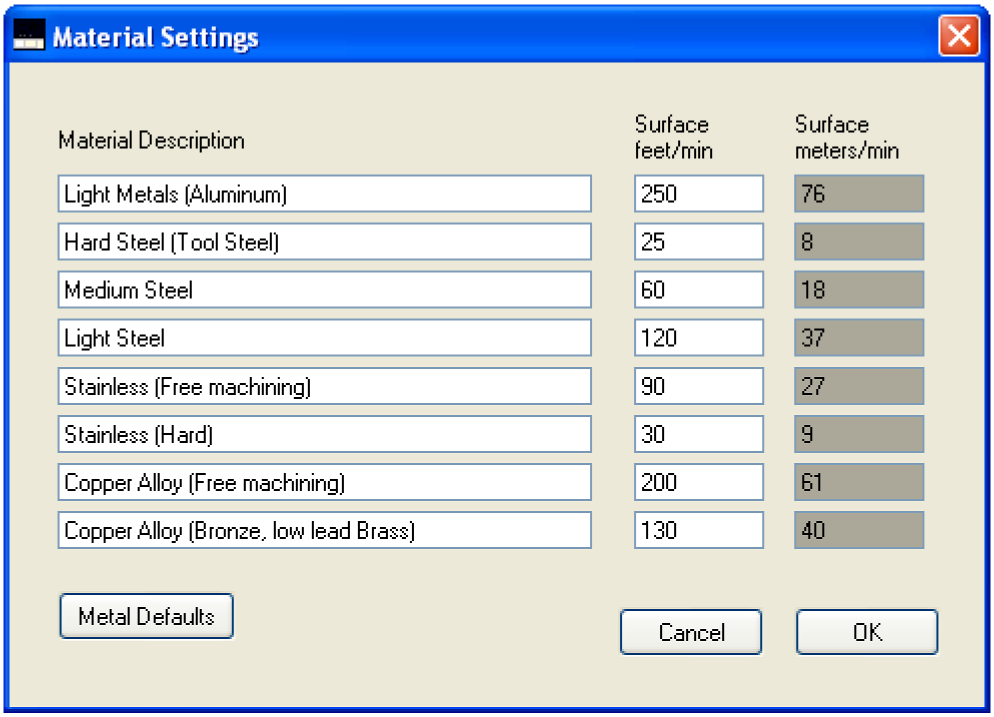

When importing DXF files, any element endpoint's within proximity to each other by the defined tolerance, are considered joined together.
A CAD program launch button can be enabled allowing for easier creation or modification of DXF files. Change the name of the button and the path to the CAD program executable for your environment.




Specify the location and rotational direction of the 4th axis. For each setting the narrative will change describing the setup.
The arc conversion converts all arc's to line segments the smaller of a max angle or a segment length. For a particular radius, an angle of 90 degrees will be generated with either 45 line segments (spanning 2 degrees each) or as many .010 segments fit the curve, whichever is greater, given the default values above.

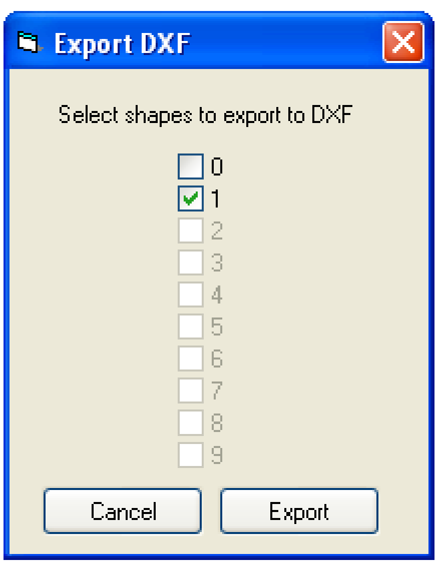
## 2.7.5 Material Settings

Material settings allows you to customize the material description and the surface speed for that material. If D2nc is in inch mode, you will be able to set the feet/min and meters/min will be calculated and visa versa.




Clicking on the Metal Defaults button will return the list of material to their original installation defaults.

## 2.7.6 Export DXF




The menu item "Export DXF..." allows any shape from any shape source to be exported to a DXF file. By default the current shape is checked for export. You may select additional or alternative shapes to be exported. All shapes selected for export are sent to a single layer 0 in the DXF file. Clicking on the

"Export" button prompts for a DXF file name and location.

# 3 Using D2nc - Basic Concepts

Understanding a few basic concepts will go a long way to helping you use D2nc.

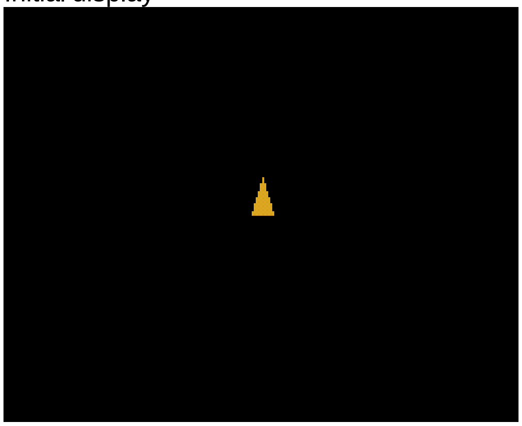
There are three stages to producing g-code with D2nc.
1. Describe a shape or import and extract a shape from a DXF file.
2. Set machine, material and tool constraints.
3. Define operations which are added to a machine queue.

D2nc is based on a simple Shape Description Language [91] (SDL). When you initially load D2nc, you will see an arrow in the black shape display. You type commands in the shape description area and the result is reflected in the display area.

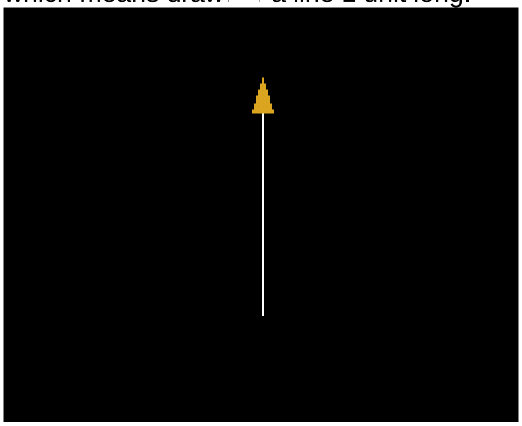
The process of describing a shape can be thought of as navigating along a course or driving a vehicle. Each move or turn you make is a continuation from your last move.

The current heading or direction is the key to describing a shape. The heading indicator is the gold arrowhead in the shape display area. It is used as an aid to track the current heading. The arrow indicates the point at which the next command will start and direction it which it will go.

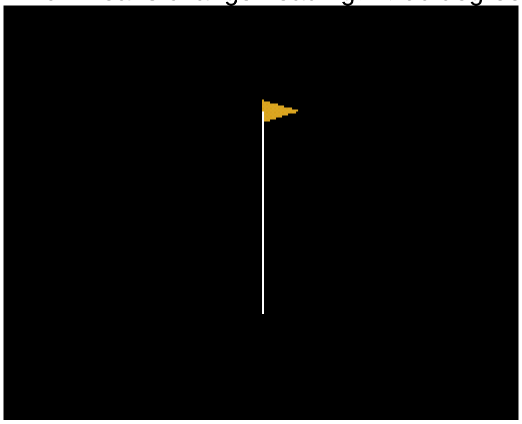
Initial display



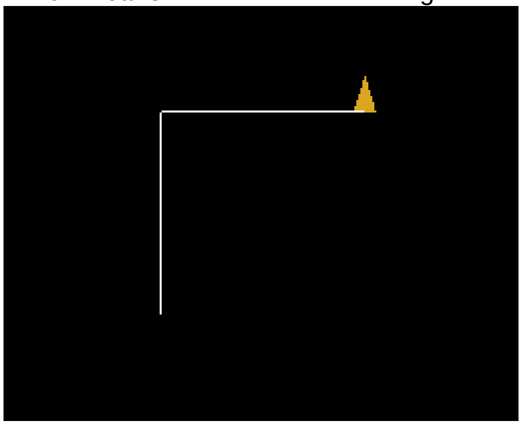
Enter the following in the Shape Description Area [11]:
`d1`
which means draw [97] a line 1 unit long.



Now enter:
`h90`
which means change heading 95 90 degrees right



Continue to enter the following:
`d1h-90`
`which means` draw a line 1 unit long then change heading 90 degrees left



As can be seen from this simple sequence, the D1 in step 2 and the D1 in step 4 both drew a line in the direction of the heading indicated by the yellow heading indicator.

Now consider that the shape represents the outline of a machine part in plan view. This is converted into g-code by defining machine operations (contouring, drilling ...) in subsequent steps. Operations can be performed on single or multiple shapes that are added to a machining queue. The queue is processed into g-code and loaded into Mach3 with a single click of the "Generate G-Code to File" button.

## 3.1 End-to-End Process

In the end-to-end process look at the use of D2nc from describing a part to the output of g-code.

First we describe a shape, set the constraints, define a path which we add to the machine queue. Clicking on "Generate G-Code to File" button will then generate the g-code and save it in the interface file. If you started D2nc from within Mach3 it will additionally load the g-code into Mach3 so that its ready to run.

To start, load D2nc from the desktop icon or from the D2nc button within Mach3. If the Mach3 button does not exist, see the installation 5 guide for instruction on how to enable it.

### 3.1.1 Describe a Shape using SDL

After D2nc starts, you are presented with a clear black Shape Display Area and empty Shape Description Area below. The first task in D2nc is to describe a shape using Shape Description Language [91]. This is accomplished by entering commands into the Shape Description Area. These commands are interpreted and the resulting shape is displayed in the Shape Display Area.

D2nc supports multiple shape sources. Currently there are two shape sources,  SDL and DXF. Think of a shape source as a provider of geometry to D2nc's CAM engine. Once you have sourced a shape, steps 2 and 3, setting constraints and defining a tool path are the same.


As an alternative to this tutorial for SDL shape entry, you can follow the tutorial Importing a shape from DXF [37].

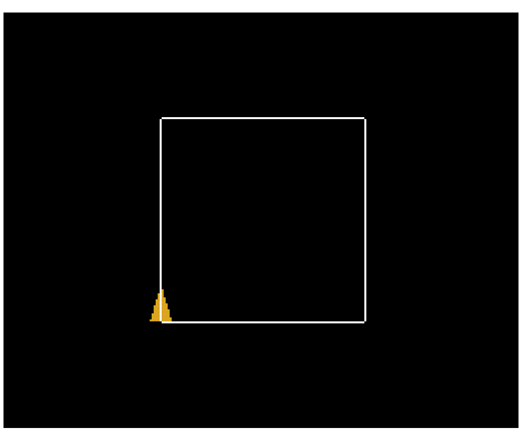
We begin this first shape by introducing two commands. They are Draw [97] and Heading [95]. With these two commands we will describe the profile of a one inch or millimeter cube depending on your native units. The native units are displayed top right above the shape display area.


All the following examples are sized appropriate for inches. *If you're using metric you can switch to inch [30] for this tutorial or substitute a "d25" wherever there is a "d1" used below.*

Enter the following into the Shape Description Area [11]:

```
d 1
h 90
d 1
h 90
d 1
h 90
d 1
h 90
```

which will produce the following display in the Shape Display Area [9].



Let's examine what happened here. The first "d1" drew a line in the direction of the heading indicator. The "h90" caused the heading indicator to turn 90 degrees clockwise. The next "d1" drew a line in the direction of the heading indicator. This second line is at a right angle to the first line due to the change in heading. The second line started from where the first line ended.

Congratulations! You have completed your first shape. For more insight to SDL, consider the following:

The shape description you just entered can be optimized by using features of the language. The following shape descriptions describe the same shape you've just completed, a one unit square.

The setting of the heading can be specified using the optional heading parameter of the Draw [97] command instead of a separate heading command.

```
d1,90
d1,90
d1,90
d1,90
```

The Repeat [106] command has an inline block definition further simplifying the shape definition. Take the block enclosed by braces { } and repeat it four times

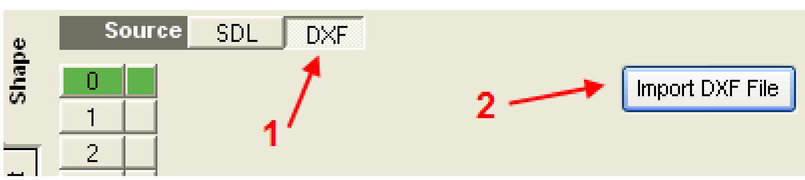
```
r{d1,90}4
```

There are many way to achieve the same thing in SDL as it is a language.

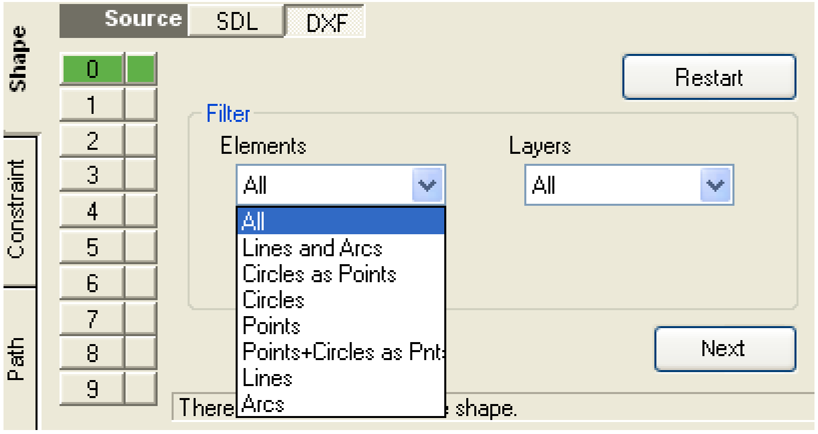
The next steps will be to define the constraints and a tool path to generate g-code. You may now proceed to the setting constraints [40] or take a look at Importing a shape from DXF [37].

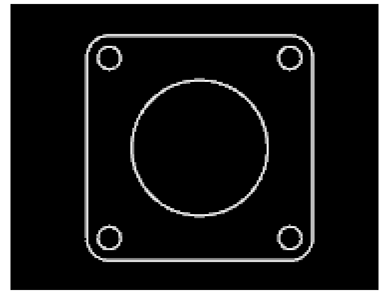
## 3.1.2 Import a Shape from DXF

DXF import was added to D2nc as a way of getting more complex shapes which cannot be described with SDL, into D2nc. I like to refer to it as shape extraction where a shape is extracted from the DXF file. Once you have extracted a shape, the steps required to generate g-code, that is the setting of constraints and defining tool paths, are the same as if the shape had been described with SDL.




To import a DXF file first select the DXF shape source and then click the **Import** button. In the open file dialog, browse to the directory containing your DXF file, select the file you wish to import and then click the **Open** button. You will find a sample DXF file in C:\D2nc directory with the name D2ncsample.dxf which I will use in this tutorial.

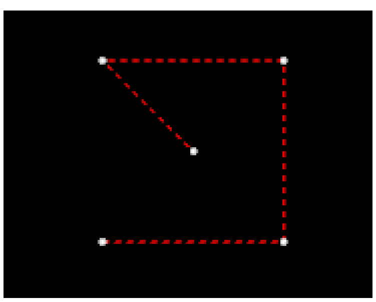


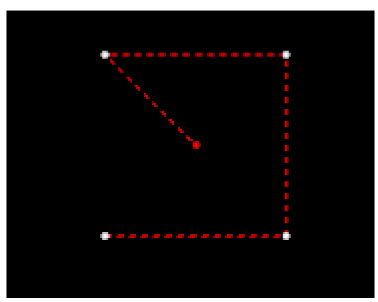



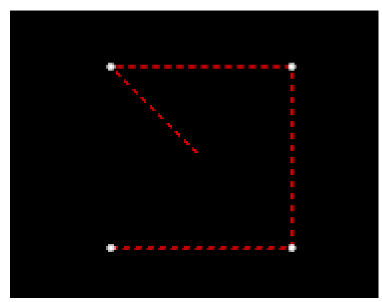
The filter panel is the first of two panels used during shape extraction.  The **Elements**, defaulted to Lines and Arcs, control which elements in the DXF are extracted into the shape. The **Layers** determine from which layer those elements are extracted. The choices are either the default All layers or any other named layer in the DXF file.

**Note!**
*The idea is to import just the elements needed for one machining operation into one shape. Change the filter to All elements and All Layers to reveal the full contents of the sample DXF file. This file contains a flange requiring three operations. One, drill the four corner holes. Two, cut the center hole offset to the inside and three, cut out the part from the stock by offsetting to the outside. The three operations we've now determined will require importing the DXF three times and each time extracting different elements into a new shape.*

To create our first shape to drill the four corner holes, change the filter to **Circles as Points** and click the **Next** button. In D2nc, only points can be used for defining a drilling operation.



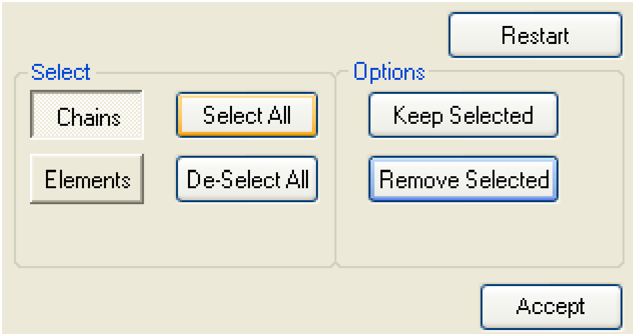




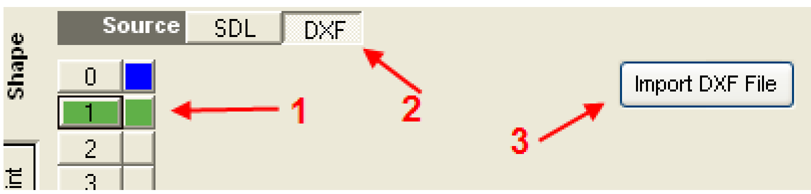
The five circles in the DXF file have been converted to points and imported. The points are indicated by the white dots in the display area and the read dashed lines the tool movement. All other elements in the DXF file have been ignored.

We have a point in the center of the shape originating from the large center circle. It too was converted to a point but as we don't want to drill there, we need to remove it. Place the mouse pointer over the white dot and click to select it. Once selected it will turn red.

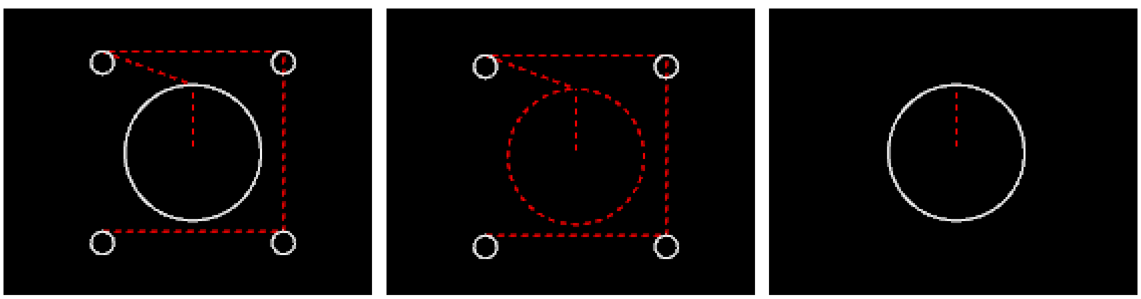
**NOTE!**
Some older graphic card that do not support OpenGL V2 will not be able to click select chains and elements with the mouse. An alternate method is to use the left and right arrow keys to cycle through chains or elements (depend on the selected option). One one chain or element can be selected at a time. Use the DEL key to remove the selected item.




Click the **Remove Selected** button to eliminate the selected elements. Click the **Accept** button to lock in the shape.

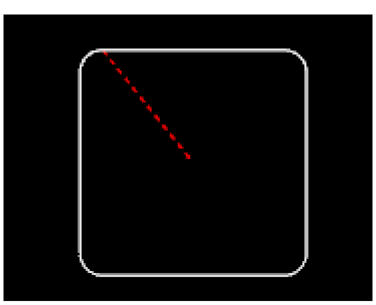



Now that we have shape 0 locked in, select shape 1 then the DXF source and then **Import**. Select the same DXF to import and click **Open**. Set the filter to **Circles** and click **Next**.



The five circles in the DXF file have been imported. Click on the large center circle to select it. Now click the **Keep Selected** button which will remove all elements that are not selected leaving us with the shape for the inside offset. Click the **Accept** button to lock in the shape.

The last shape we need is that of the outside contour. We have shape 0 and 1 locked in so select shape 2 then the DXF source and then **Import**. Select the same DXF to import and click **Open**. Set the filter to **Lines and Arcs** and click **Next**.



This shape needs no adjustment so we just click **Accept** to lock it in.

Congratulations, you have successfully extracted the three shapes from the DXF file which are required for the three machining operations needed on this particular part. You may now proceed to set the constraints 40, define the tool paths 40 and then generate g-code 42.

## 3.1.3 Set Constraints

Now that we have the shape defined, lets proceed and setup to do an outside offset contour and generate the g-code into Mach3.

Click on the Constraint 14 tab to show the constraint panel.

Enter the following values:

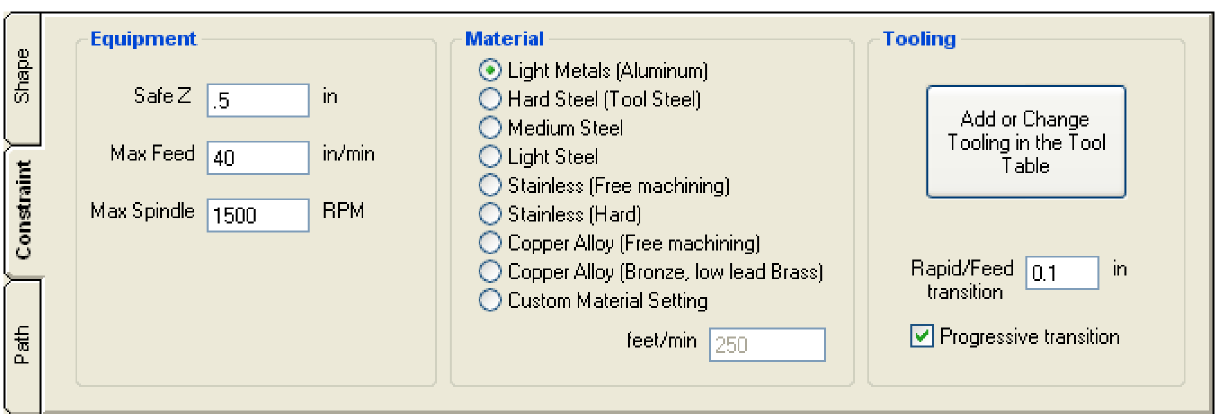
**Equipment**
SafeZ .5
Max Feed 40
Max spindle 1500

**Material**
Select Light metals (Aluminum)

**Tooling**
For this exercise we will use tool zero. Tool zero cannot be defined in the tool table and its parameters are entered during the definition of the tool path in the next step. Refer to the tool table 28 help on how to effectively use the tooling.

Set the Rapid/Feed transition to 0.1. This is the height at which a Z axis move down will change from rapid rate to plunge rate as defined in the path definition. Check the progressive transition check box. This will cause the transition height to move down by the step size on each subsequent pass.

The constraint section will now look like this:

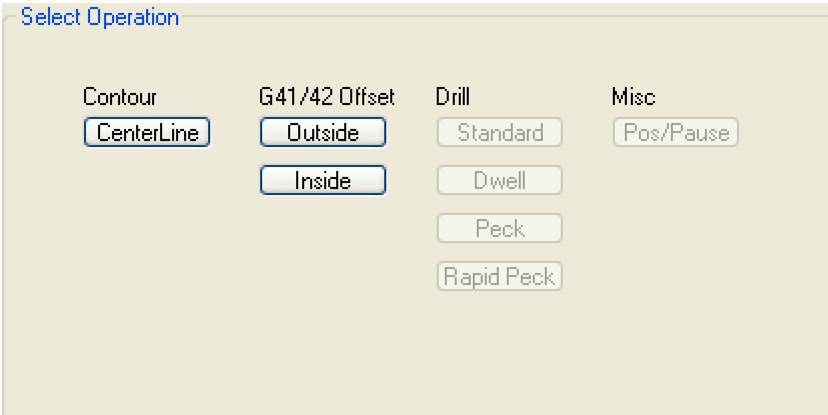



## 3.1.4 Define Path

You are now ready to define a path and generate g-code.

Click on the Path 15 tab to show the path panel.

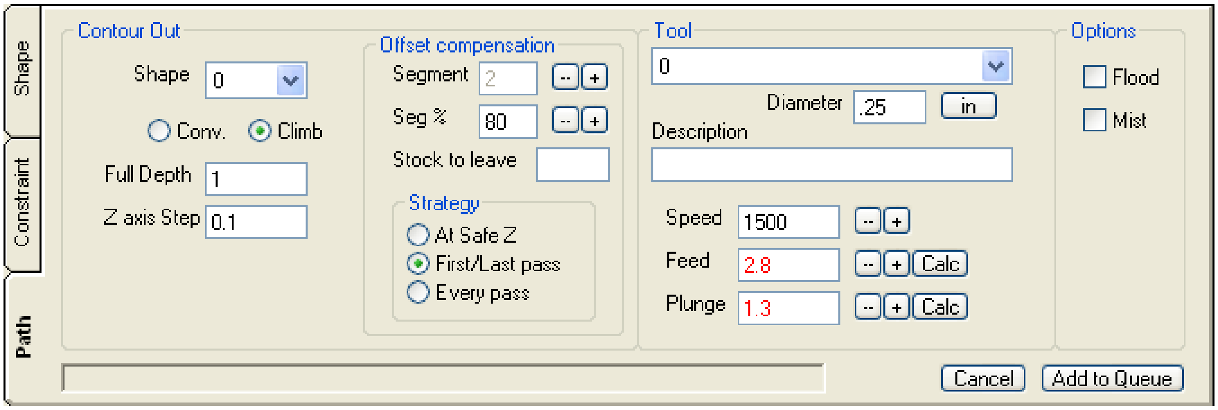
In the Select operation panel you should see the three buttons, Centerline, Outside and Inside enabled and the remaining buttons greyed out. Only the operations for which there are eligible shapes are enabled. If this is not the case, go back to the Describe a Shape step and check your shape definition.




Select the Offset Outside button.

- The shape to use for this contour will be preselected to Shape 0 as this is the only shape available.
- The Conv. or Climb option sets the direction the tool will move around the shape. A climbing cut outside would be in a clockwise direction.
- Full Depth: enter 1 to cut the shape to a depth of 1 inch.
- Z Axis Step: enter .1 for each pass to cut a maximum of 1/10 of an inch on each pass.
- Tool: enter a diameter of .25 for a 1/4 inch end mill. As soon as you enter the tool diameter, the lead in/out moves are calculated and shown in the display area as well as the speed, feed and plunge.
- Enter "1/4 end mill" as a description. This description will be included in the generated g-code as a reminder for what tool the code was generated.
- For off set compensation, select the second segment by clicking the plus to increase the segment number which contains the compensation moves. You will see the compensation move relocate to the second segment. The Seg% moves the location of the compensation move from 0% to 99% along the selected segment. The compensation move is only shown after the tool has been selected or the diameter entered. The compensation move is always 150% the tool diameter.
- Leave "Stock to leave" blank for now. It is possible to define a finishing pass allowance and then add another "contour left" operation to the machine queue at say full depth in one pass with no allowance.
- The compensation strategy selects where the compensation moves occur. Select "First/Last pass" to have the compensation occur only on the first and last pass. The other options "At Safe Z" makes the moves at Safe Z height so it will not remove stock and "Every pass" makes the moves at every Z axis step.
- Increase the feed and plunge if necessary by clicking on the plus next to these settings.

Your input should look as follows.

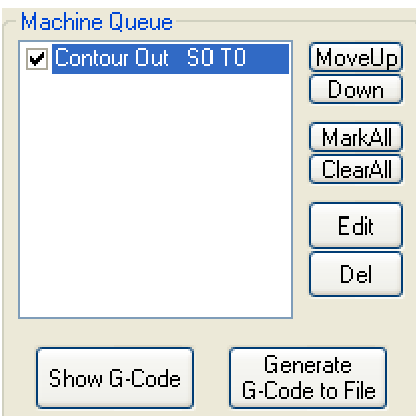



Click the "Add to Queue" button.

The path is now defined and you ready to generate g-code.

### 3.1.5 Generate Gcode

The Machine Queue now has the entry "Contour Outside Shape 0 Tool 0" in it:

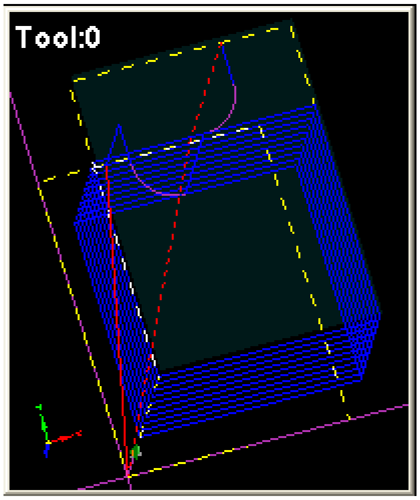



Click the "Generate G-code to File" button.

D2nc will now generate the g-code and close. If you have several items in the machine queue, only those items that are checked will have the g-code generated. This allows you to define several shapes with paths in creating a single part and to create code for each operation separately.

If you started D2nc from the desktop icon or other shortcut, it will have created a g-code file as specified in the interface file in D2nc settings 30.

If you started D2nc from within Mach3 using the integration button, Mach3 will now load the generated g-code. If you grab and tilt the Mach3 display with your mouse you will see the shape with compensation moves ready to cut.




When D2nc closed, it not only generated the g-code file, but also saved the shape and machine queue in case any changes are necessary.

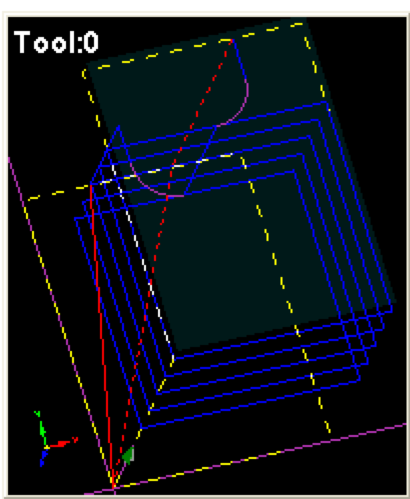
If you load D2nc again, it will prompt you (by default) if you would like to reload the last shape. This is a settable feature in the D2nc settings 30.

If you do that now and click "Yes" the shape will reload. Click the path tab and the machine queue is already set. Click on the queue entry to select it and then click the edit button below the queue. You will see all your previous settings are preset.

- Go ahead and change the Z axis step to .2
- Click the "Update Queue" button.

Click the "Generate G-code to File" button.

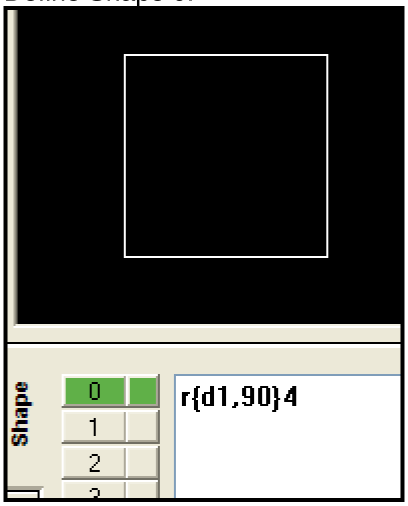
Look at the Mach3 display and you will see the number of passes have been reduced from 10 to 5.



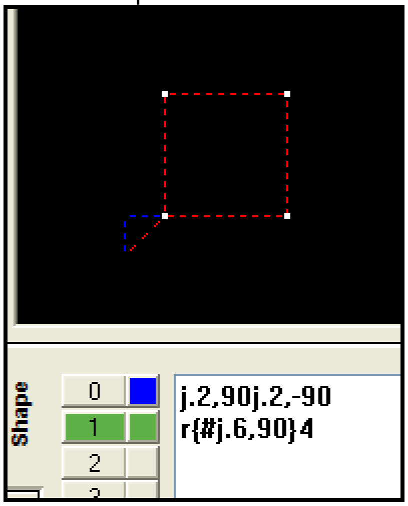
## 3.2 What's a Shape?

A shape is defined using the Shape Description Language.
Up to 10 shapes can be defined in the same program numbered from 0 to 9.
The shape selector buttons select the active shape or the shape you currently working with.
The small shape display buttons alongside the selector enable you to show or hide that particular shape.

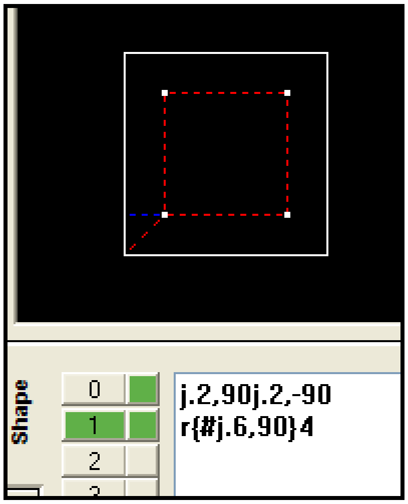
Define Shape 0.




Define Shape 1. You can see that Shape 0 is defined but hidden by the blue shape display button.



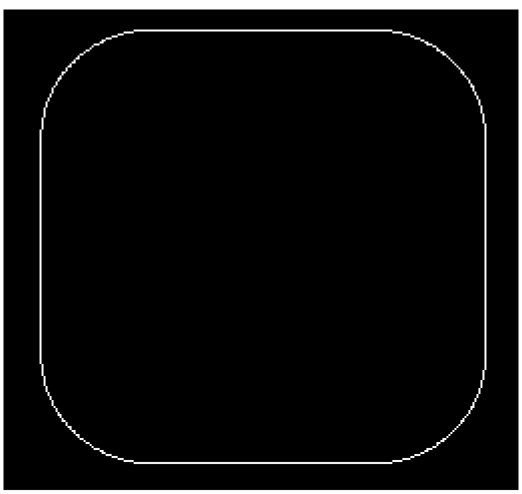

Click on the Shape 0 display button to toggle the display of shape 0 on so that it is displayed along with the active Shape 1.




Later on during path definition 40 each of the operations, for example contouring or drilling, are performed on an entire shape definition. In this example Shape 0 could be left contoured while Shape 1 could be peck drilled.

## 3.3 Simple Shape

This quick "one line" shape may help you grasp Shape Description Language.



```
d1a90,.5
d1a90,.5
d1a90,.5
d1a90,.5
```

or

```
r{d1a90,.5}4
```

`r` repeat
`{` open inline block

`d1` draw line one inch long
`a90,.5` arc 90 degrees with a radius of .5 inch
`}` close inline block
`4` repeat inline block 4 times

If you are having trouble understanding how this sequence of commands produces the shape shown, please refer to the basic concepts 34.

## 3.4 Creating your own SDL Wizards

SDL Wizards are easy to create. Lets walk step by step through the process of creating a simple SDL wizard.

1. Create a variable driven SDL program.

Remember the four sided square. We can change that into a wizard where you can specify the length of the side and the number of sides.

r{d1,90}4

the variable components here are:

1 = length of side
90 = heading change
4 = occurrences

The 90 heading change is really 360 degrees divided by the occurrences and so can be calculated. The other 2 are input variables so lets define them. The syntax for variable definition is in the SDL reference under assignment 91.

<len=1>
<side=4>

The variable names **len** and **side** can be anything you like as long as they start with an alpha character. Now we can use those variables in SDL as follows:

r{d(len),(360/side)}(side)

You can see the variables have been inserted into the SDL command by way of functions 93. By changing the values of **len** and **side** you can create any size equilateral polygon. The completed variable driven SDL program is the foundation of a SDL Wizard.

<len=1>
<side=4>
r{d(len),(360/side)}(side)

2. Create the SDL Wizard from the program

This step is planned to be automated in a future version of D2nc. For now, this has to be done manually. The wizards are kept in the sub-directory C:\D2nc\Wizards

Wizards are XML files with a .d2w extension. The format for the XML file is provided in a template in this directory called C:\D2nc\Wizards\Sample.d2w.txt
Copy this template to another file, giving it the name of the template as you would like it to appear in the wizard selection screen. In this example copy it to a file "My polygon wizard.d2w". The wizard selection screen will only show files with the extension .d2w
Edit this file you created in notepad or some other text editor.

The contents of the new file you copied from the template should be:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<wizard>
<title></title>
<desc></desc>
<graphic></graphic>
<parameters>
<param>
<prompt></prompt>
<variable></variable>
<value></value>
<mvalue></mvalue>
</param>
<param>
<prompt></prompt>
<variable></variable>
<value></value>
<mvalue></mvalue>
</param>
</parameters>
<SDL>
<shape>
<![CDATA[

]]>
</shape>
</SDL>
</wizard>
```

Adapt the contents and save the file.

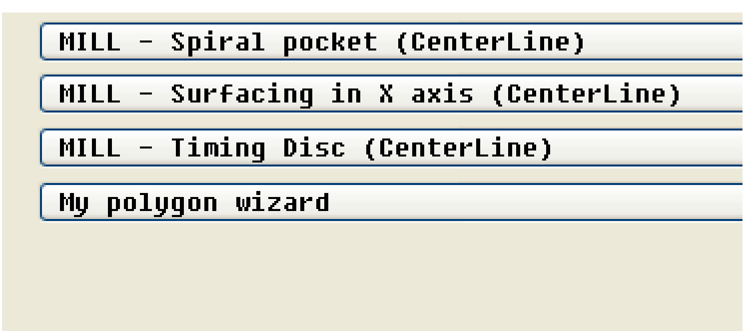
<title> and <desc>
This text is shown at the top of the wizard parameter entry screen as a way to explain how the wizard works.
<graphic>
The name of a .jpg file used to illustrate the wizard. If no name is entered or the name entered is not found, a default "no image found" will be displayed. The image can be created with any screen capture and paint application that saves jpg files. The image will scale to fit in the graphic window but the actual size is **340**x**340** pixels. It is always better to create graphics exactly this size as scaling causes loss of quality.
<parameters><param>.....</param><param>.....</param>...</parameters>
This <param> section contains the four fields <prompt>, <variable>, <value> and <mvalue>. The <param> section can be repeated up to a maximum of 12 times within the <parameters> section.
<prompt>
Text describing the input parameter.
<variable>
The name of the variable used in the static part of the SDL program.
*Note! A variable named tool_dia has special meaning when used in a wizard. The value entered for tool_dia is passed as a hint to the centerline tool path definition and pre-fills the tool 0 dia with this*

*value.*
<value> and <mvalue>
The default value entered in the parameter entry screen. The <mvalue> is the default value if D2nc is in metric mode.
<SDL> <shape>
The <SDL> section contains <shape> parameter which contains the static part of the SDL program. The SDL program must be placed within the XML <![CDATA[ and ]]> so that it is not interpreted as XML code.

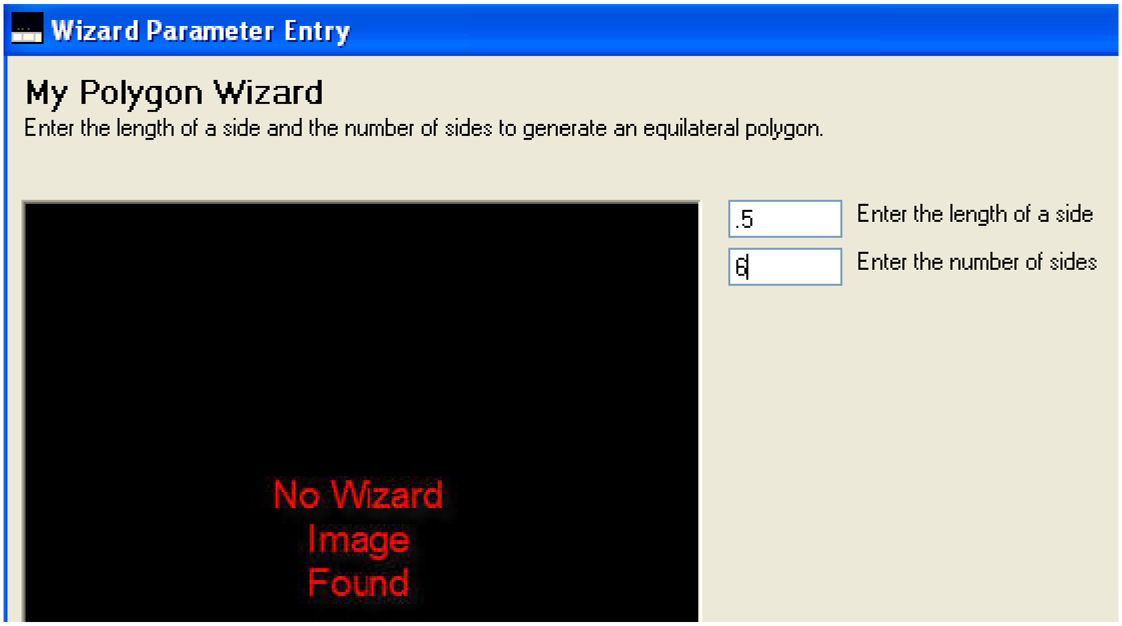
```
<?xml version="1.0" encoding="ISO-8859-1"?>

<wizard>
<title>My Polygon Wizard</title>
<desc>Enter the length of a side and the number of sides to generate an equilateral polygon.</desc>
<graphic>polygon.jpg</graphic>
<parameters>
<param>
<prompt>Enter the length of a side</prompt>
<variable>len</variable>
<value>1</value>
<mvalue>25</mvalue>
</param>
<param>
<prompt>Enter the number of sides</prompt>
<variable>side</variable>
<value>4</value>
<mvalue>4</mvalue>
</param>
</parameters>
<SDL>
<shape>
<![CDATA[
r{d(len),(360/side)}(side)
]]>
</shape>
</SDL>
</wizard>
```

3. Test your new wizard!

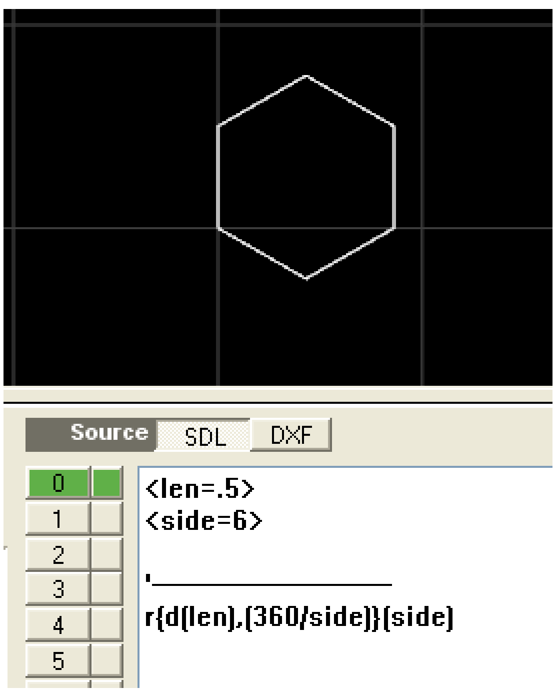
Load D2nc and click on the wizard icon. You will find your newly created wizard shown in the list of available wizards.



Click on the "My polygon wizard" button to show the Wizard Parameter Entry screen.




Change the length to 0.5 and the number of sides to 6 and then click the OK button.




In the shape description area you will see the generated SDL. In the shape display area, the resultant shape.

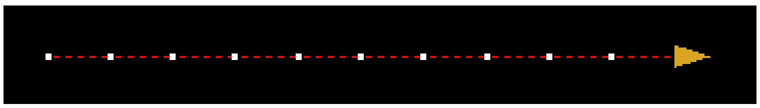
Congratulations on creating your first SDL Wizard!

## 3.5 Rotary using X or Y

D2nc only generates g-code for 2 1/2d. While there is no ability to generate g-code for a 4th axis and a rotary table, some rotary operations can be done by using the rotary table connected to either the X or Y motor controller. To do this you need to calculate what linear move on the X or Y axis produces one rotation of the rotary table.

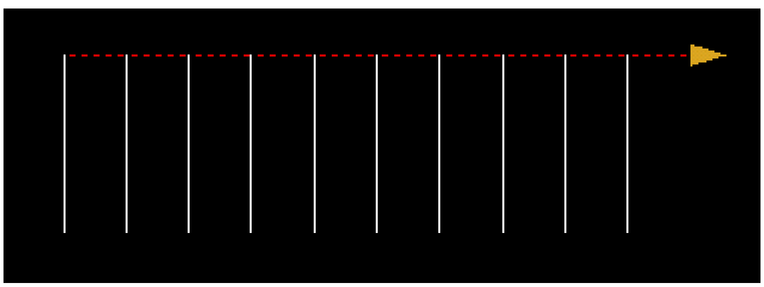
Once you calculate this distance (3.5 inches in the examples below) you divide it by the number of holes or teeth you need to drill or cut and use that as the jump "step over" as per the examples.

**Radial Drilling or Spoking**
Use to drill a series of holes or with a position/pause operation to position for manual drilling or boring.



```
h90
r{d0j.35}10
```

**Gear Cutting**
For gear cutting in this example the rotary is in the X axis drive and Y is driven to cut the teeth. The SafeZ, Depth and Step all need to be set to zero and a center line contour path made.



```
h90
r{
h90
d1,180
j1,90@
j.35
}10
```

# 4 Wizard Directory

The naming convention for the wizards are as follows:

Primary function

Drill - Drilling holes with a drill bit.
Mill - Cutting using an end mill.

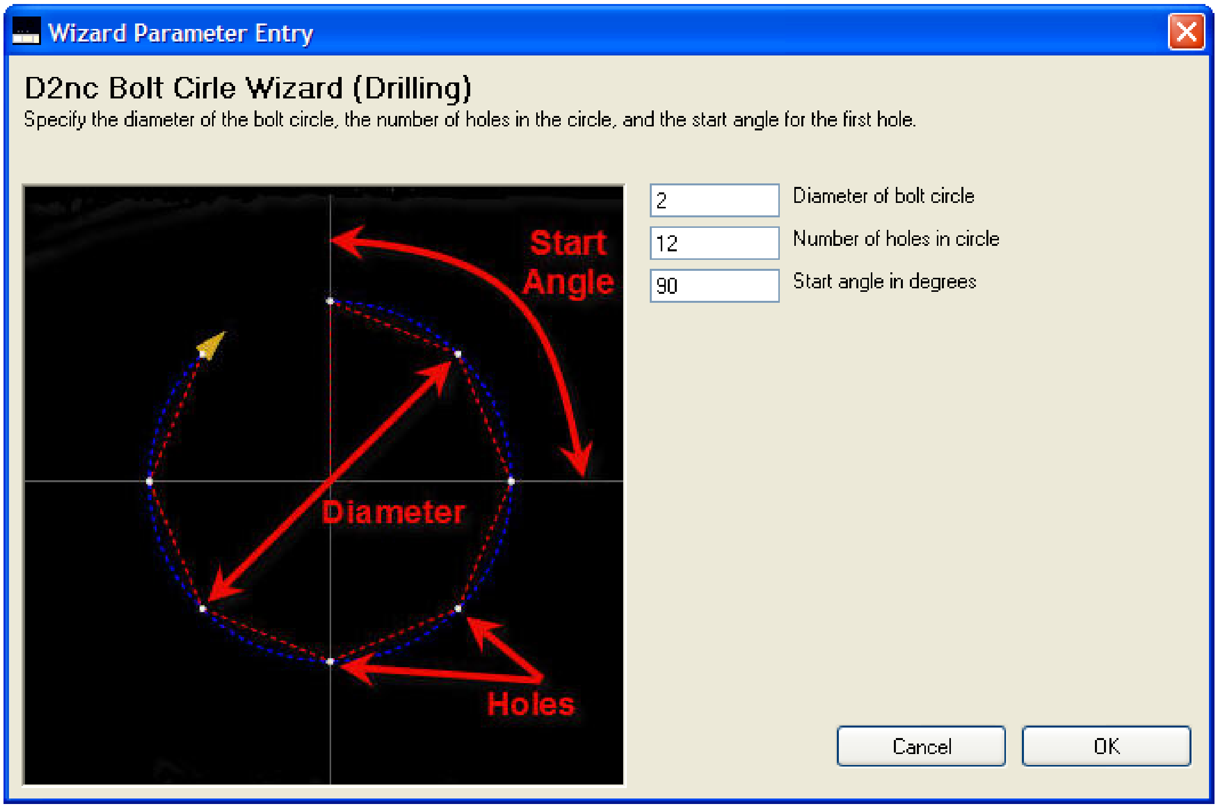
Contour type (for mill)

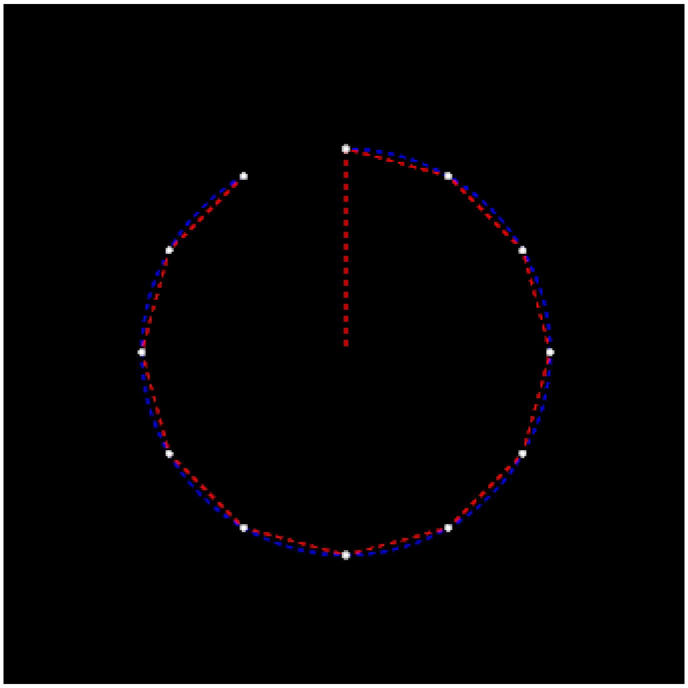
CL - Centerline contour generated. Shape already compensated for tool diameter.
Off - Offset contour generated. Shape is actual part dimension and needs to be offset.
Multi - Advanced wizard creating multiple shapes for the part being machined.

Name

Name describing the wizards main function.

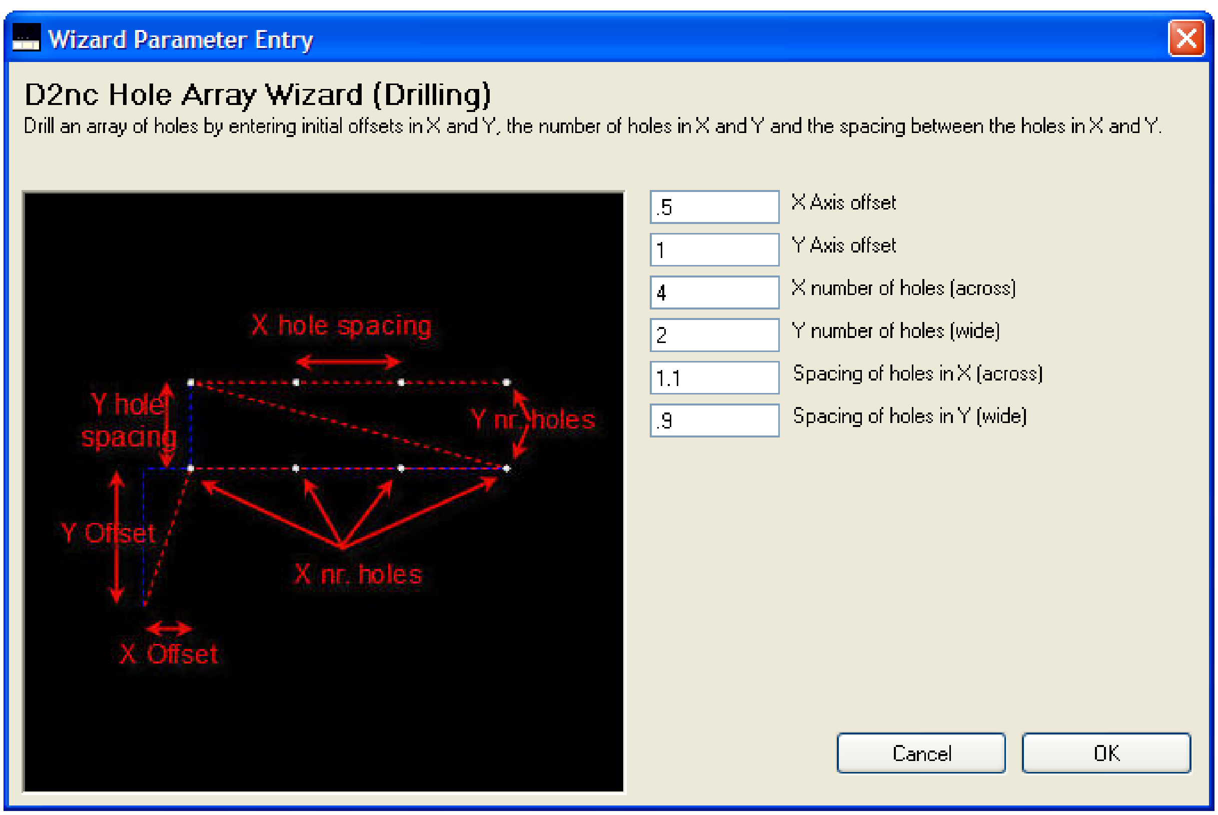
## 4.1 Drill - Bolt Circle

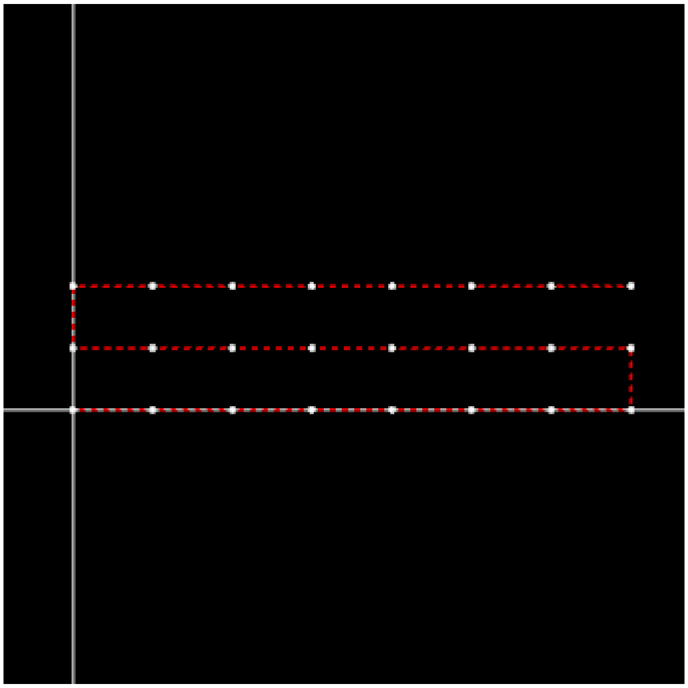




<circle_dia=2>
<nr_holes=12>
<head=90>

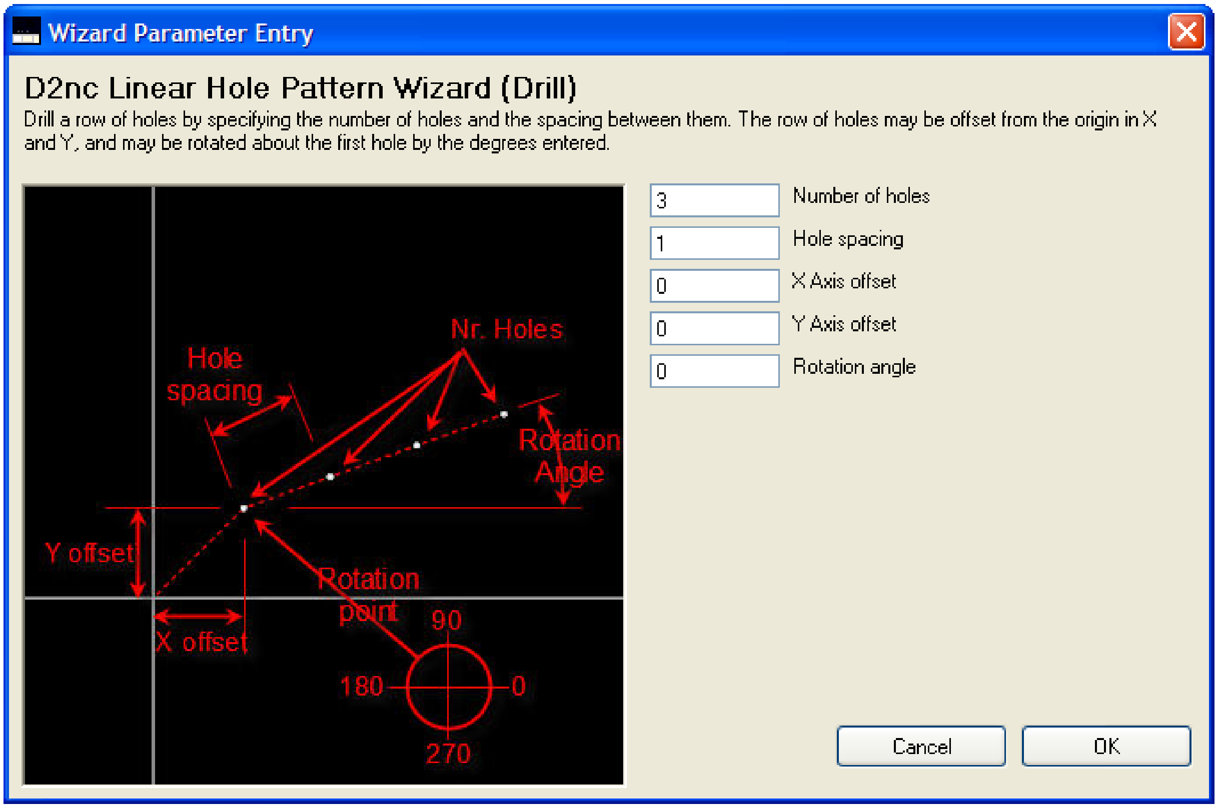
## 4.2 Drill - Hole Array

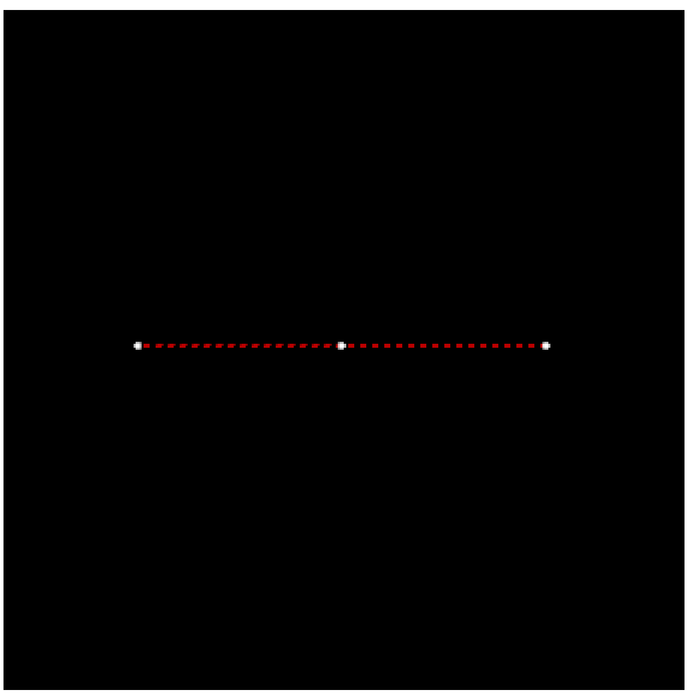




<x_offset=0>
<y_offset=0>
<x_number=8>
<y_number=3>
<x_spacing=.9>
<y_spacing=.7>

## 4.3 Drill - Linear Hole Pattern

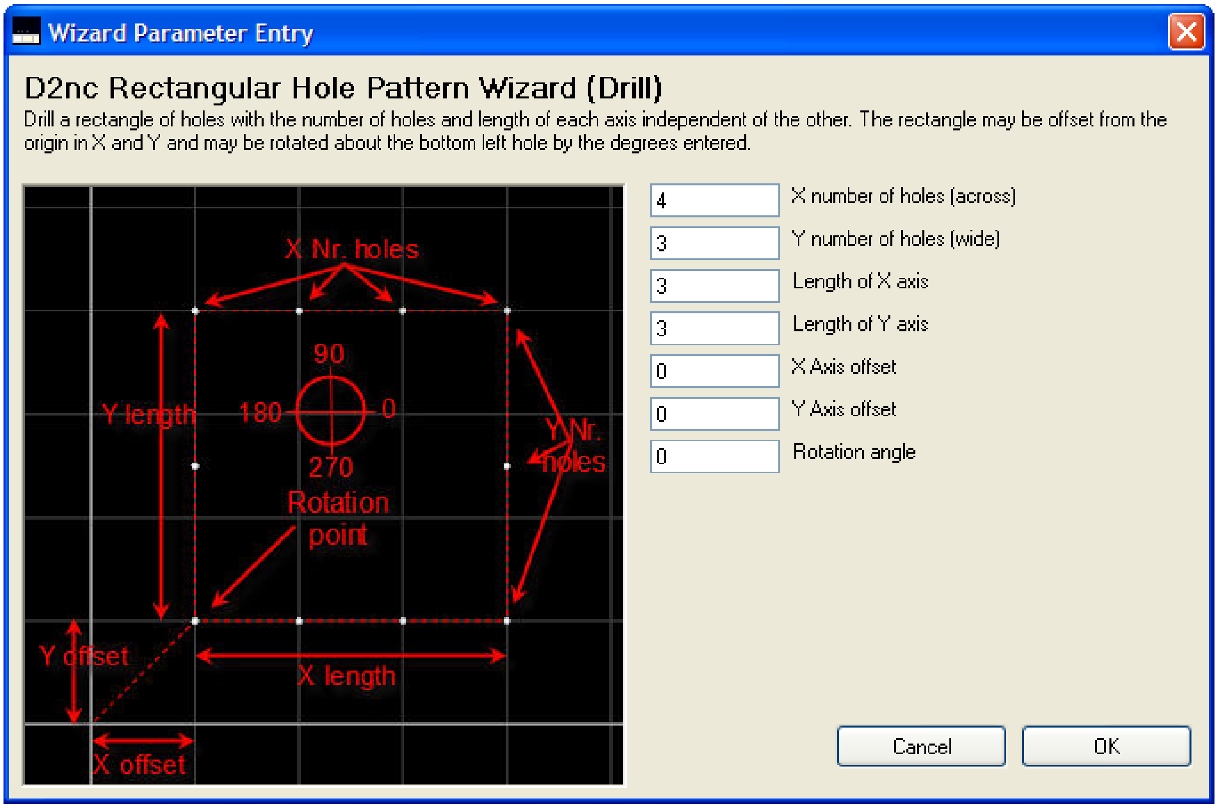


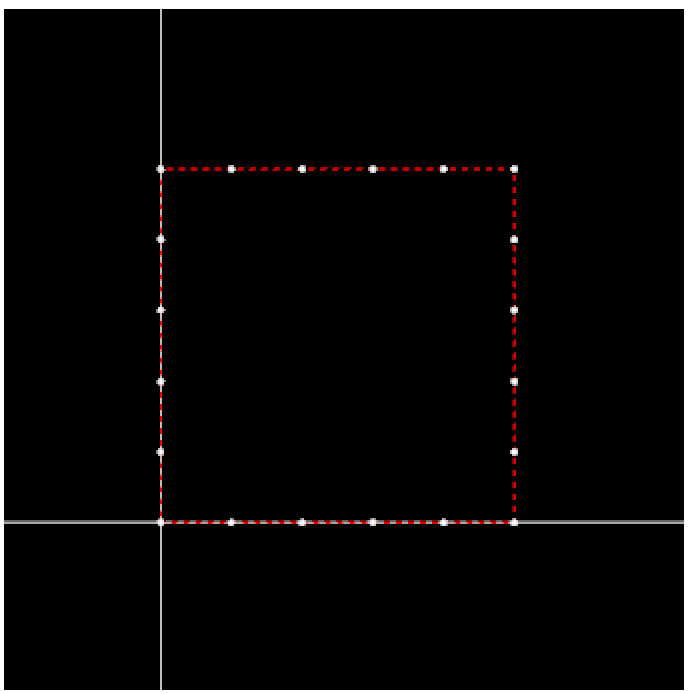



<nr_holes=3>
<hole_spacing=1>
<x_offset=0>
<y_offset=0>

<angle=0>

## 4.4 Drill - Rectangular Hole Pattern

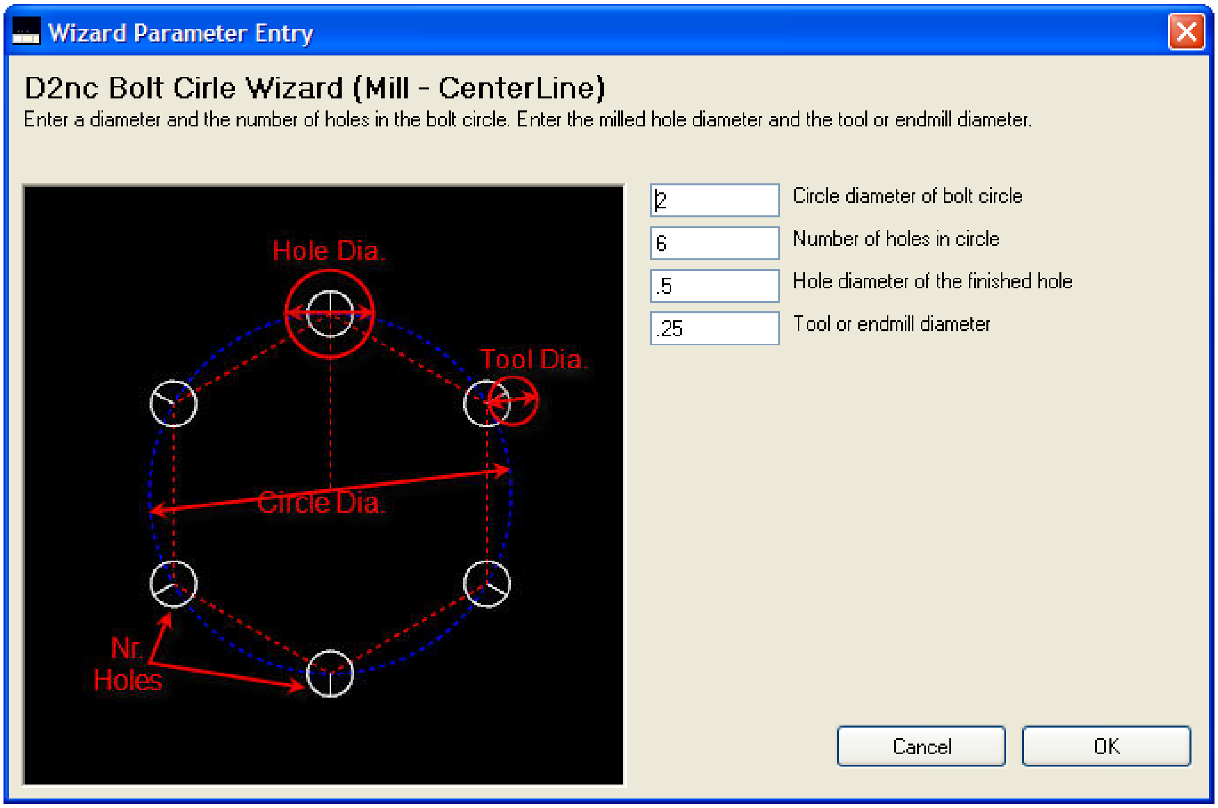


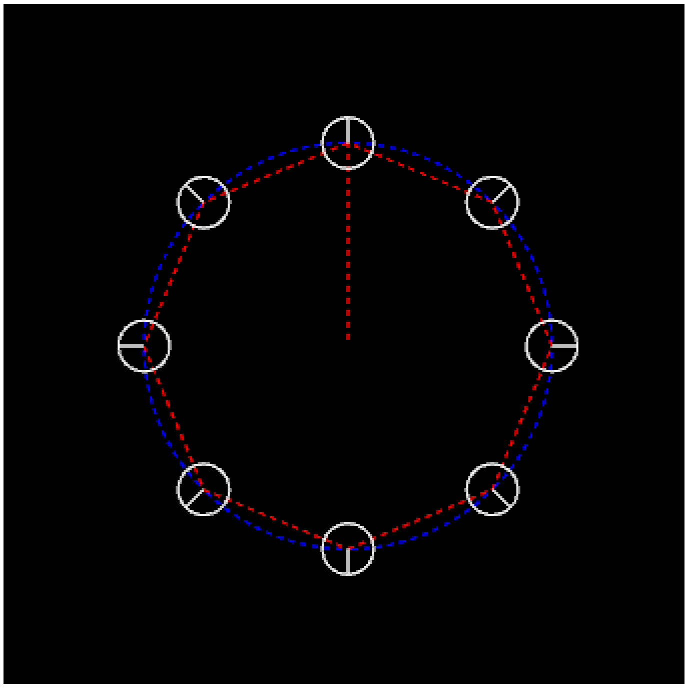



<x_nr_holes=6>

<y_nr_holes=6>
<length_x=4>
<Length_y=4>
<x_offset=0>
<y_offset=0>
<angle=0>

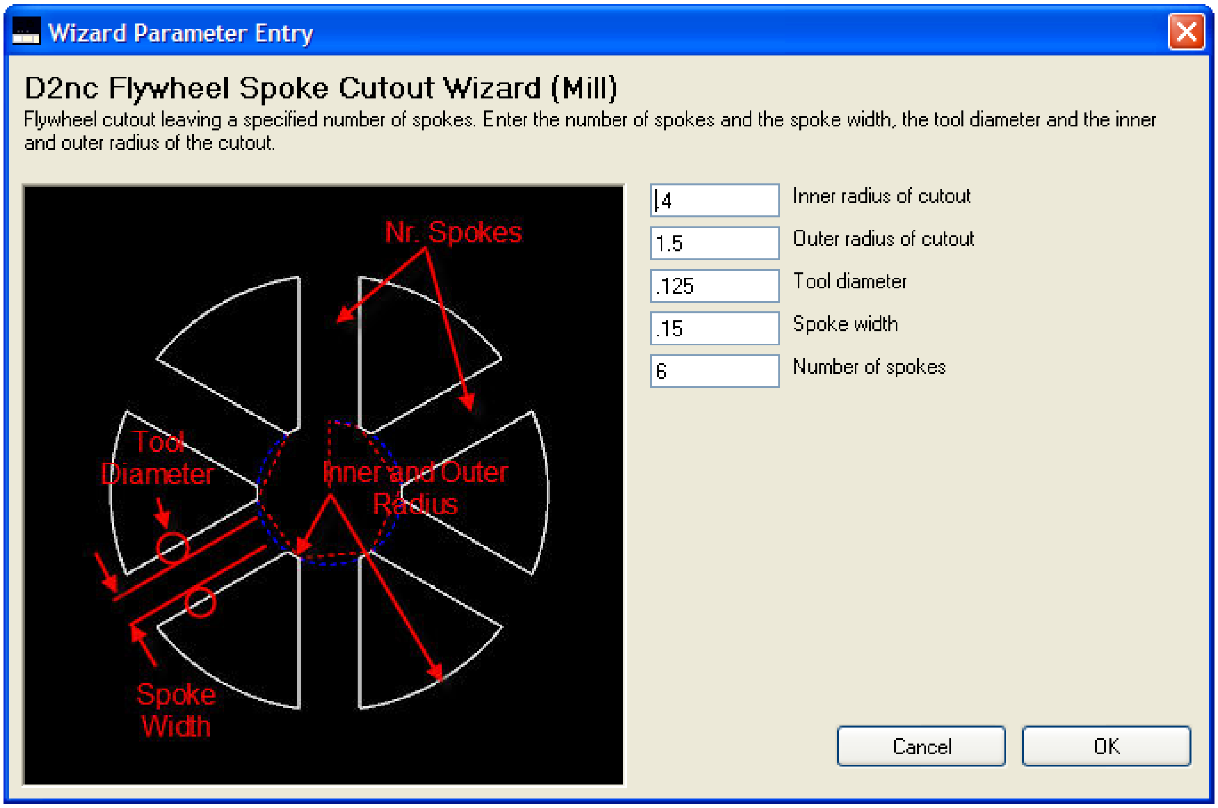
## 4.5 Mill - CL - Bolt Circle



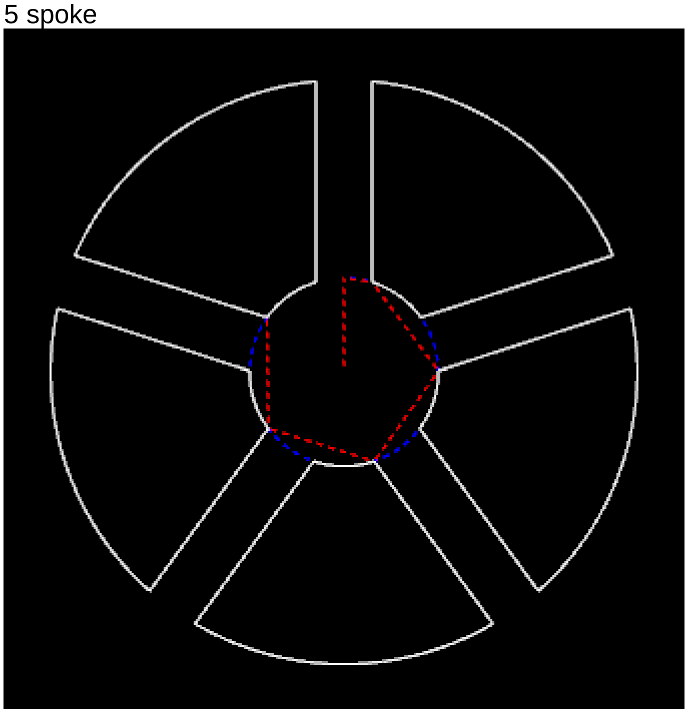


<circle_dia=2>
<nr_holes=8>
<hole_dia=.5>
<tool_dia=.25>

## 4.6 Mill - CL - Engine Flywheel Spoke Cutout



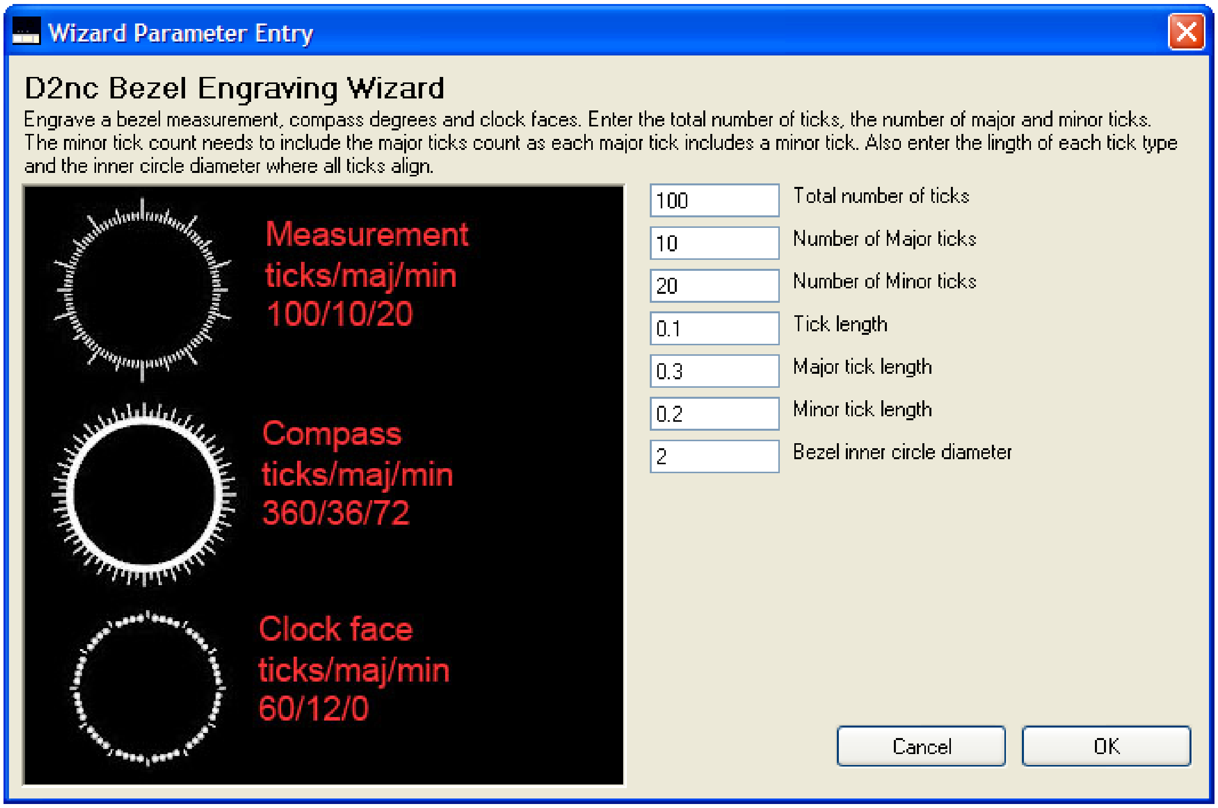

5 spoke



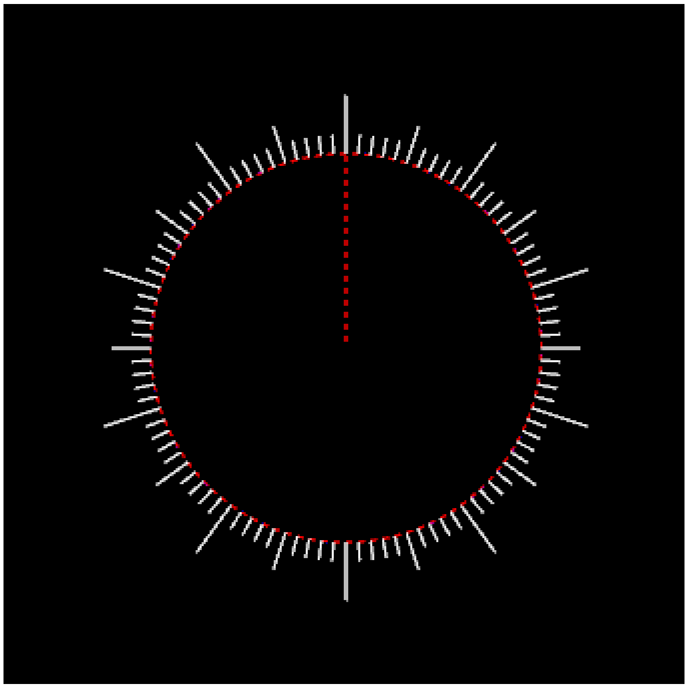
<in_rad=.4>
<out_rad=1.5>
<tool_dia=.125>

<swidth=.15>
<nr_spokes=5>

## 4.7 Mill - CL - Engrave Bezel

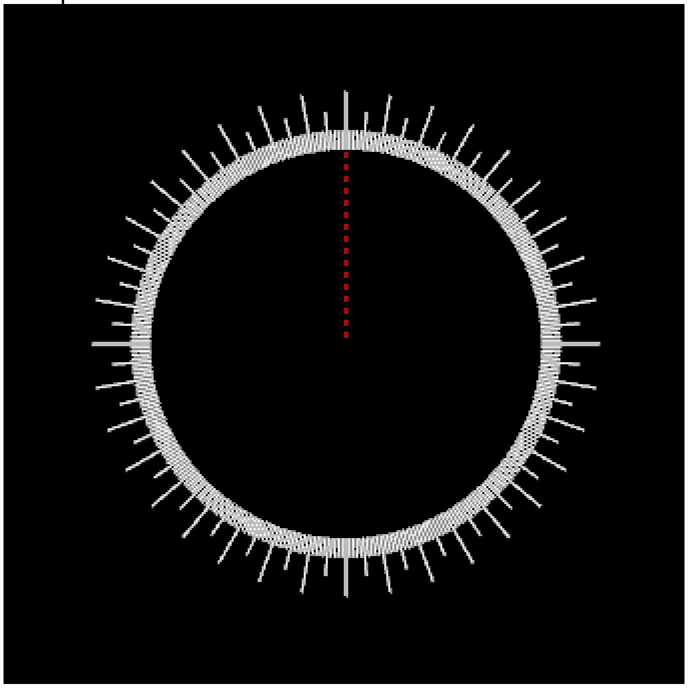



100 Divisions



<ticks=100>
<majt=10>
<mint=20>
<tlen=0.1>
<majtlen=0.3>
<mintlen=0.2>
<dia=2>

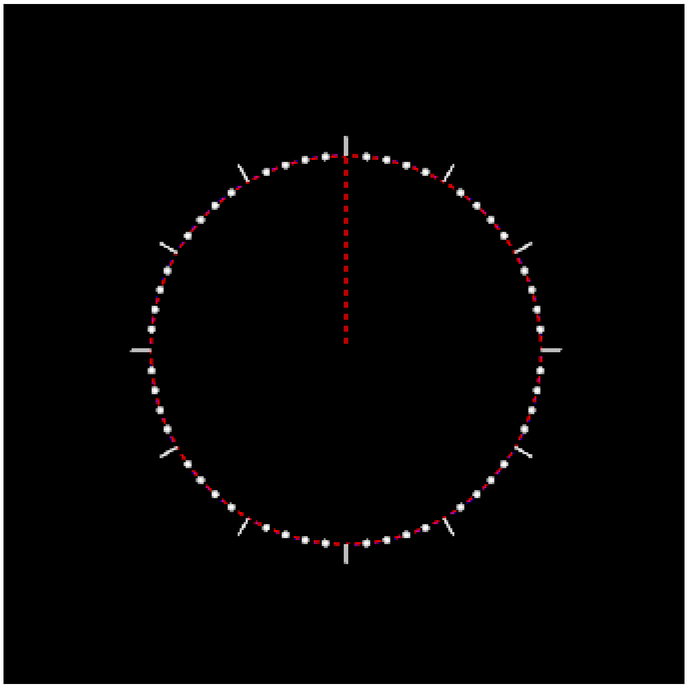
Compass Rose



<ticks=360>

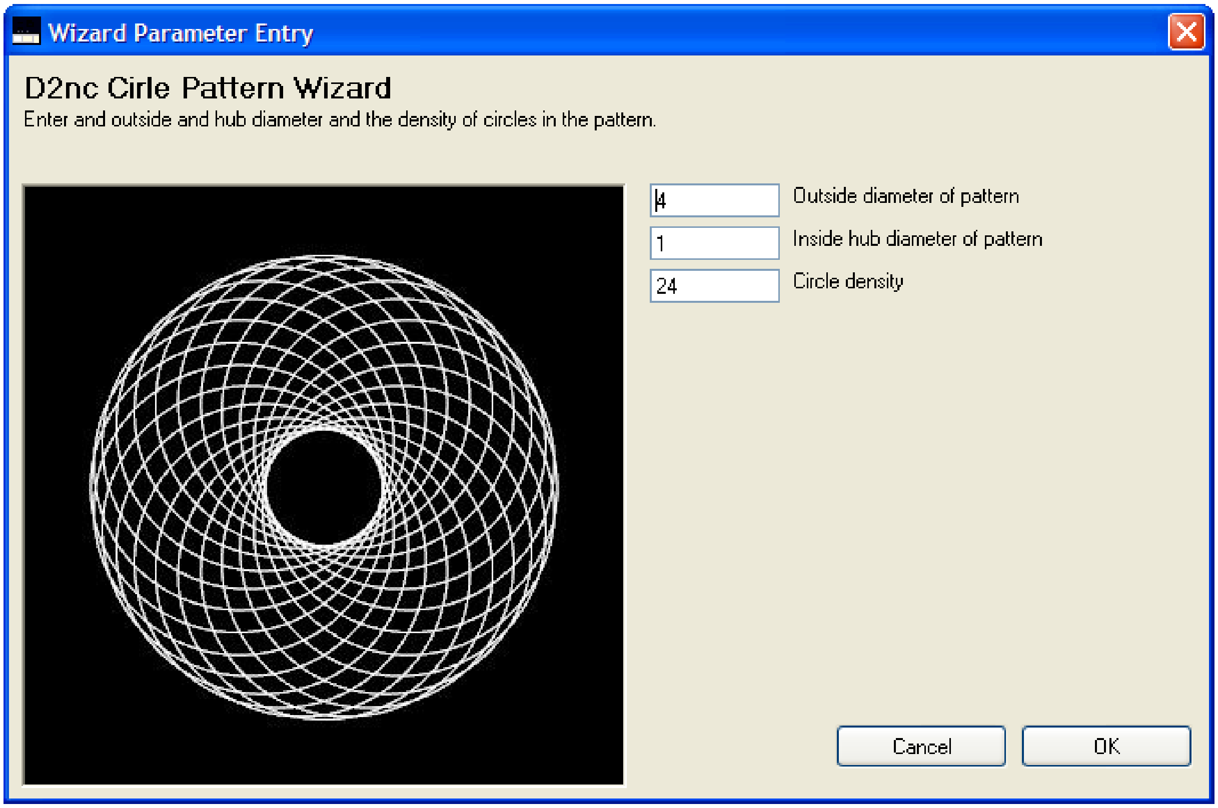
<majt=36>
<mint=72>
<tlen=0.1>
<majtlen=0.3>
<mintlen=0.2>
<dia=2>

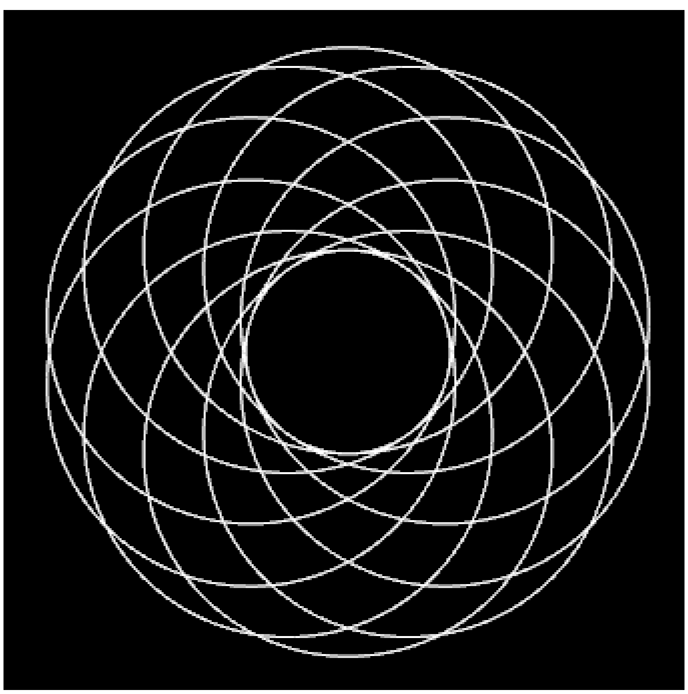
Clock Face



<ticks=60>
<majt=12>
<mint=0>
<tlen=0>
<majtlen=.1>
<mintlen=0>
<dia=2>

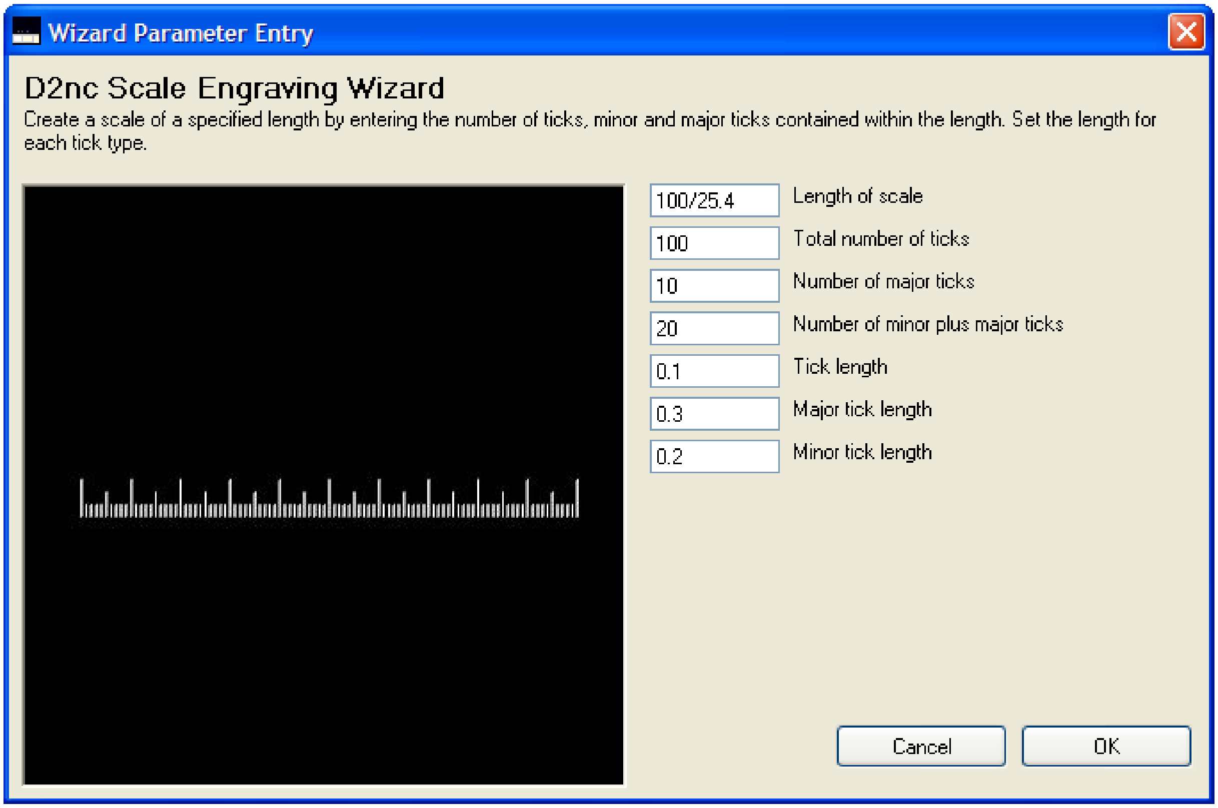
## 4.8 Mill - CL - Engrave Circle Pattern



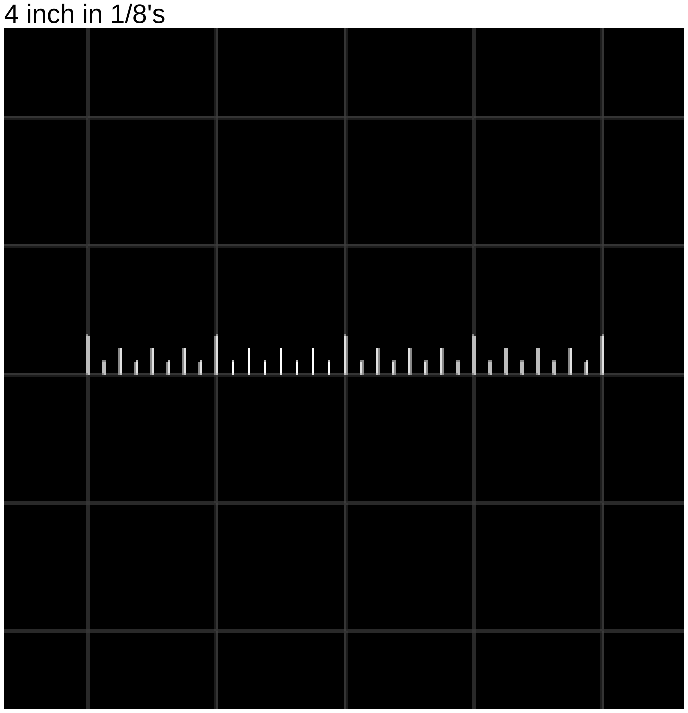



<out_dia=3>
<hub_dia=1>
<density=10>

## 4.9 Mill - CL - Engrave Scale



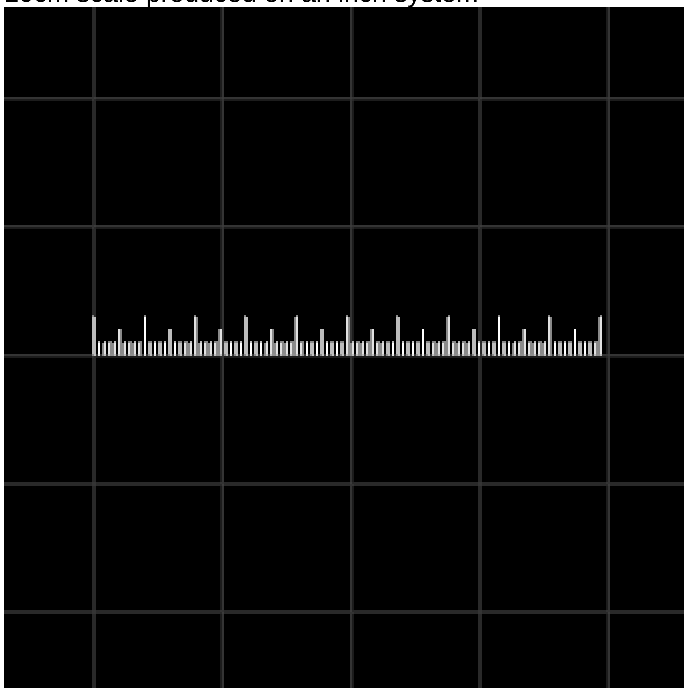

4 inch in 1/8's



<len=4>
<ticks=32>
<majt=4>

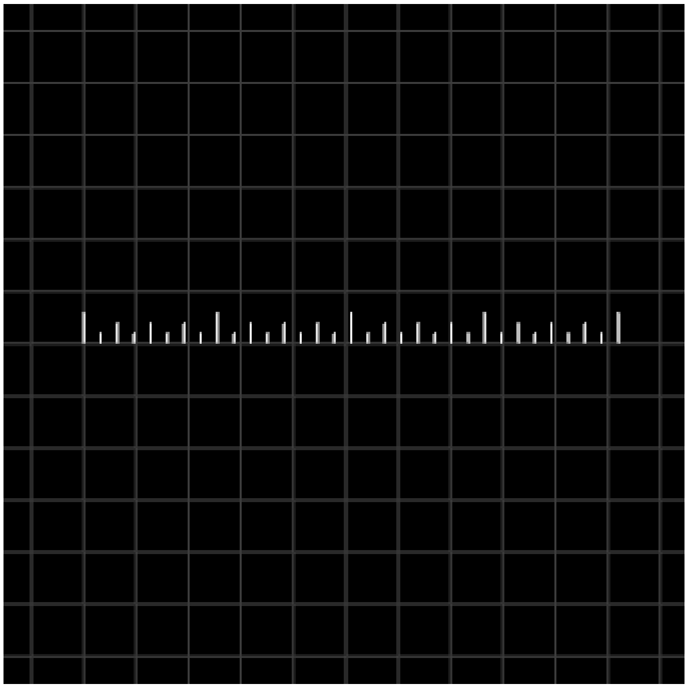
<mint=16>
<tlen=0.1>
<majtlen=0.3>
<mintlen=0.2>

10cm scale produced on an inch system



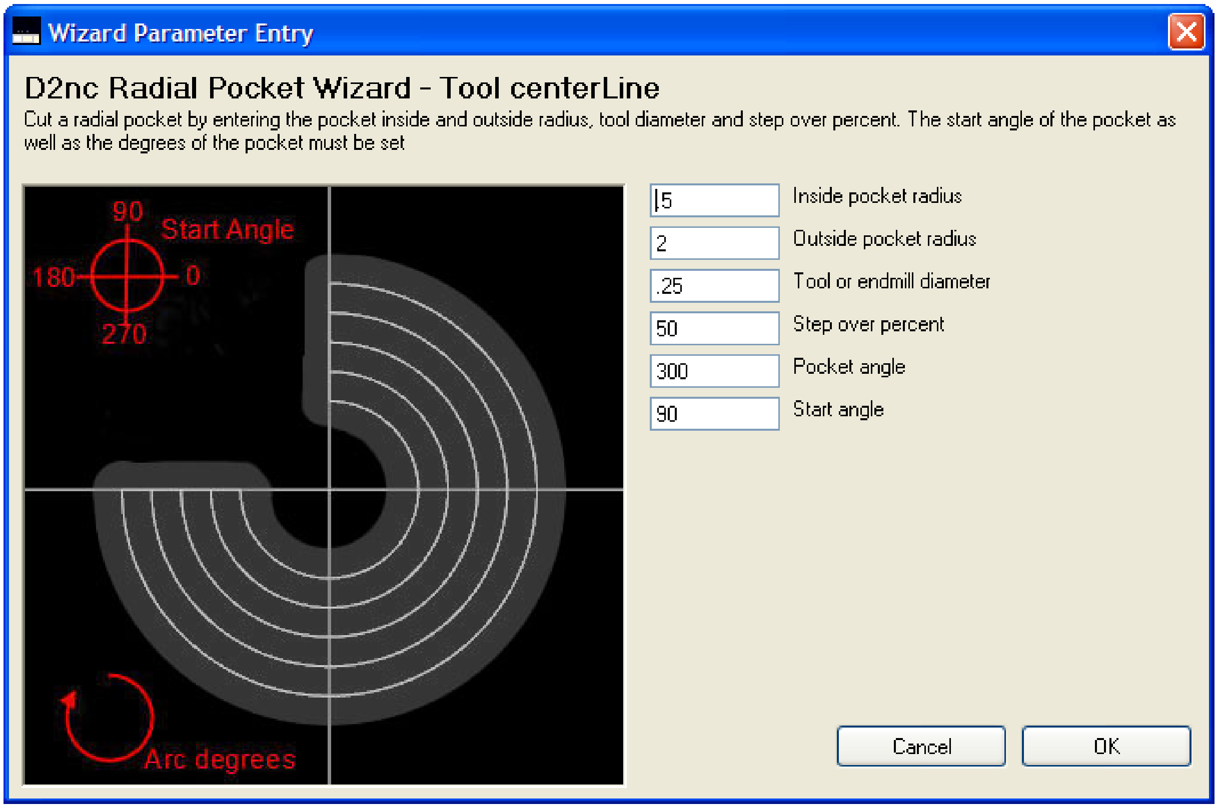
<len=100/25.4>
<ticks=100>
<majt=10>
<mint=20>
<tlen=0.1>
<majtlen=0.3>
<mintlen=0.2>

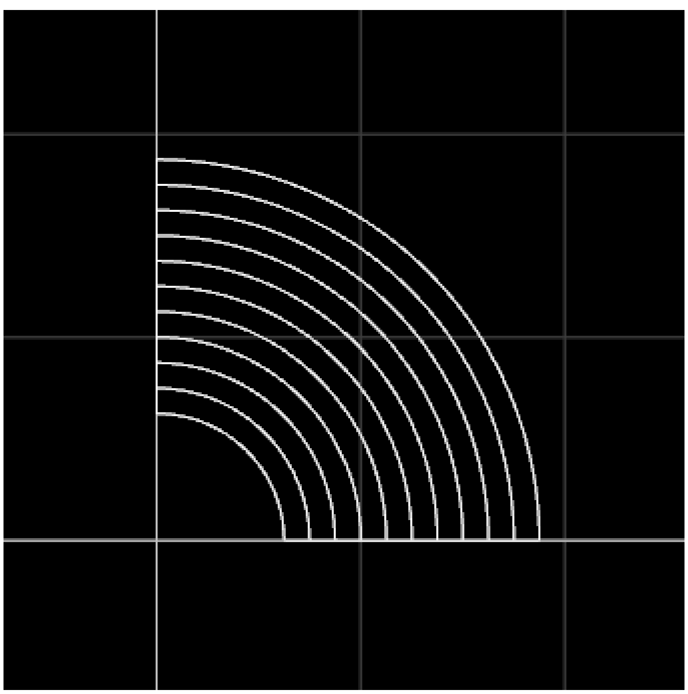
4 inch scale produced on a metric system



<len=4*25.5>
<ticks=32>
<majt=4>
<mint=16>
<tlen=2>
<majtlen=6>
<mintlen=4>

## 4.10 Mill - CL - Radial Pocket (tool centerline)

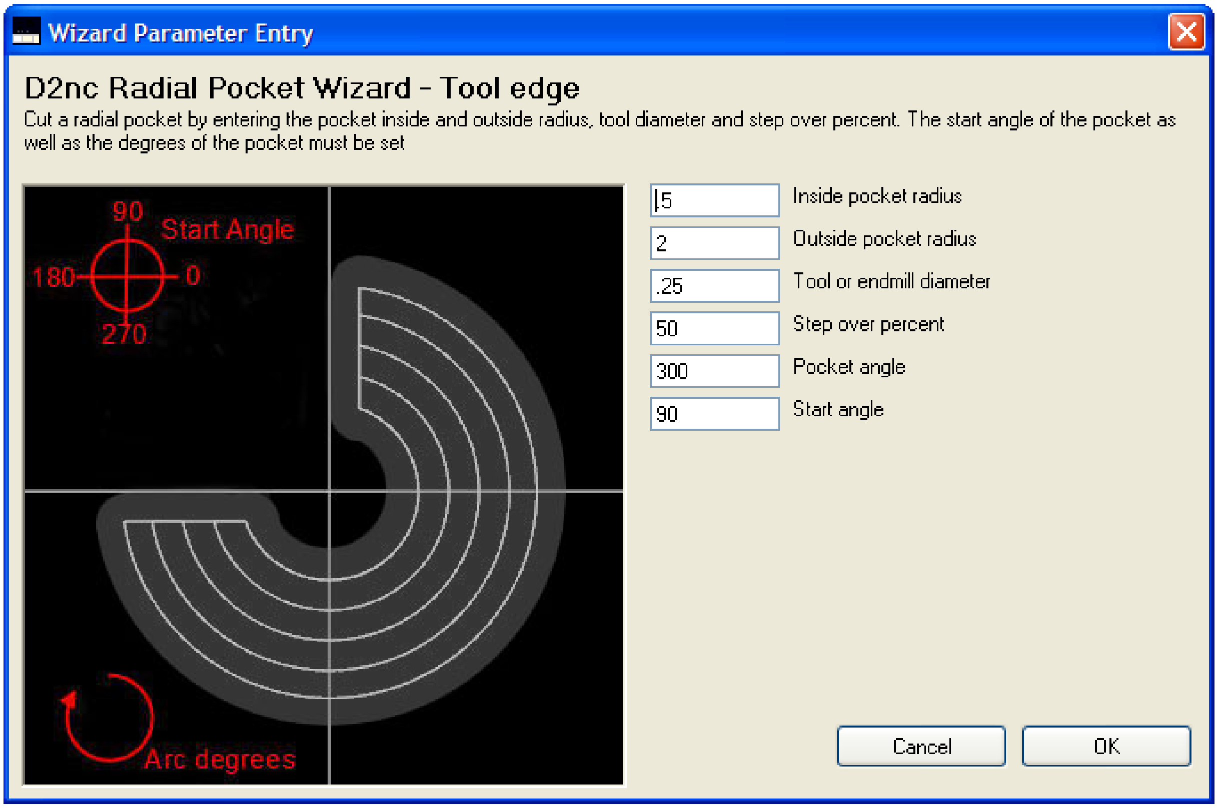


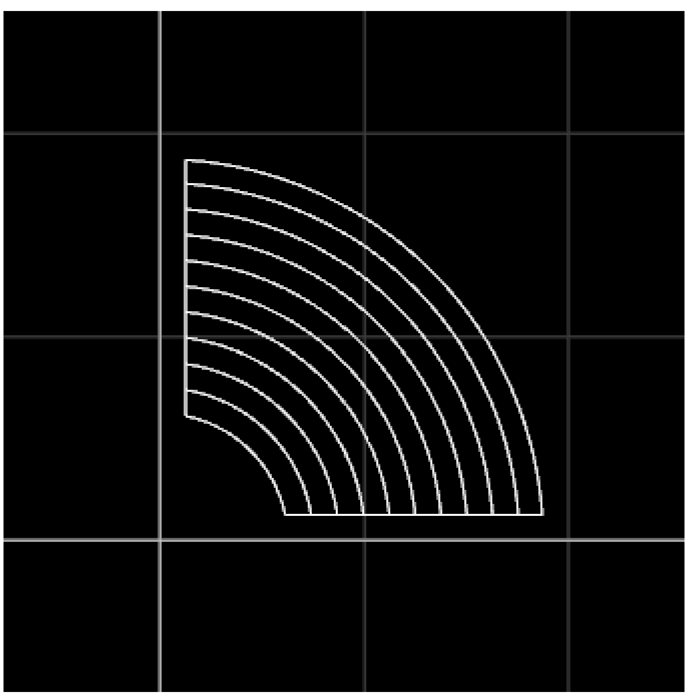



<in_rad=.5>
<out_rad=2>
<tool_dia=.25>
<step_pct=50>

<arc_deg=90>
<start_ang=90>

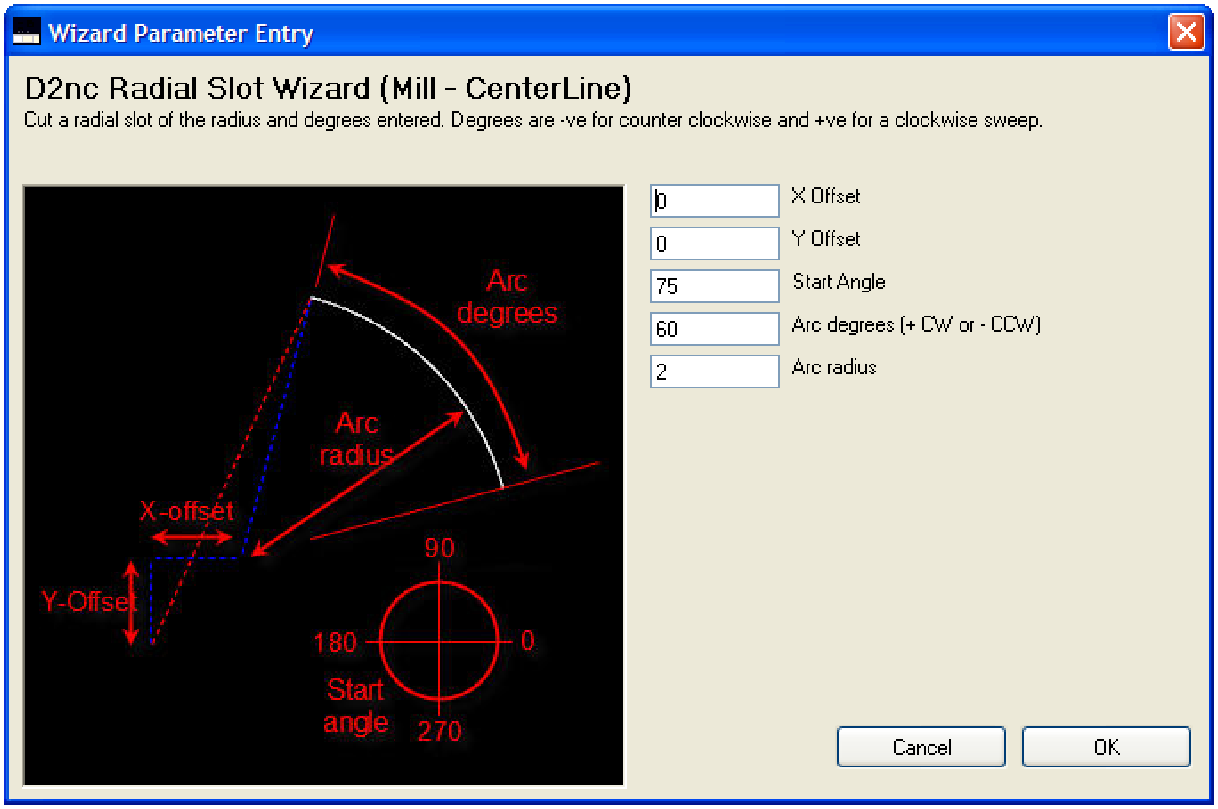
# 4.11 Mill - CL - Radial Pocket (tool edge)

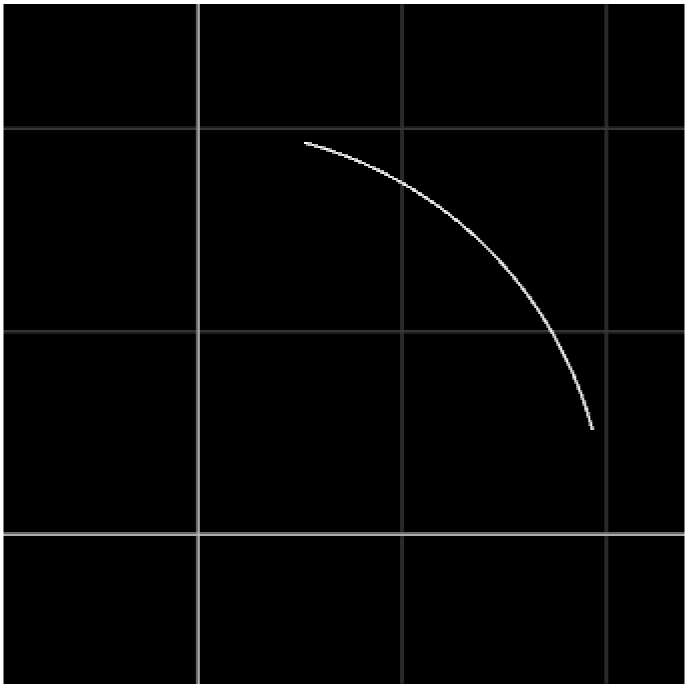


<in_rad=.5>
<out_rad=2>
<tool_dia=.25>
<step_pct=50>
<arc_deg=90>
<start_ang=90>

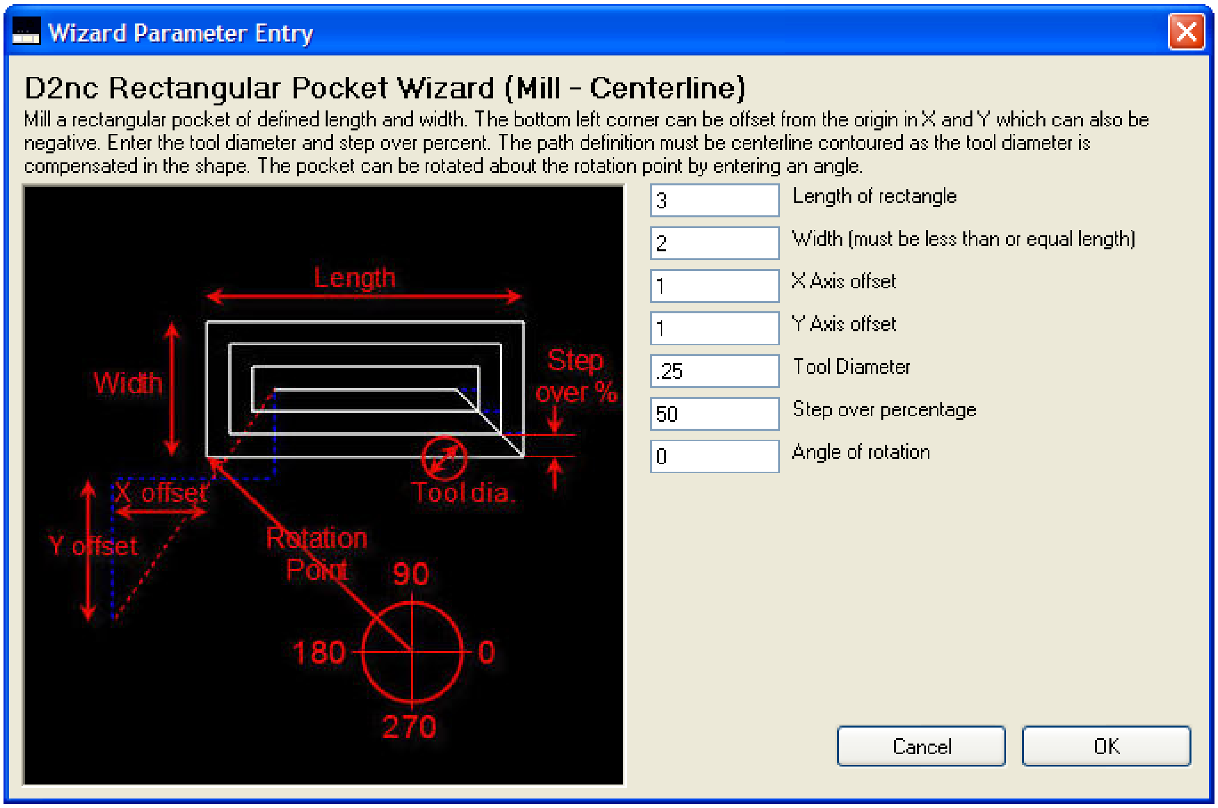
## 4.12 Mill - CL - Radial Slot

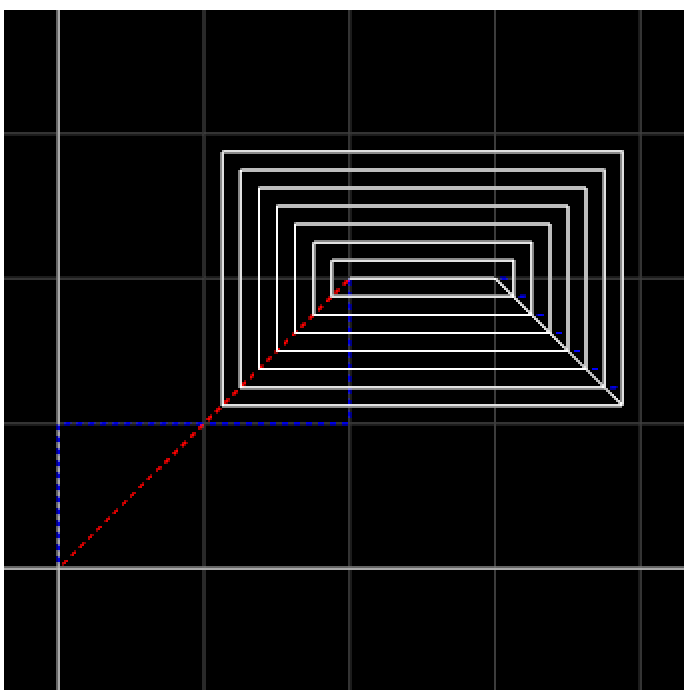




<x_offset=0>
<y_offset=0>
<start_angle=75>
<arc_deg=60>
<arc_rad=2>

## 4.13 Mill - CL - Rectangular Pocket

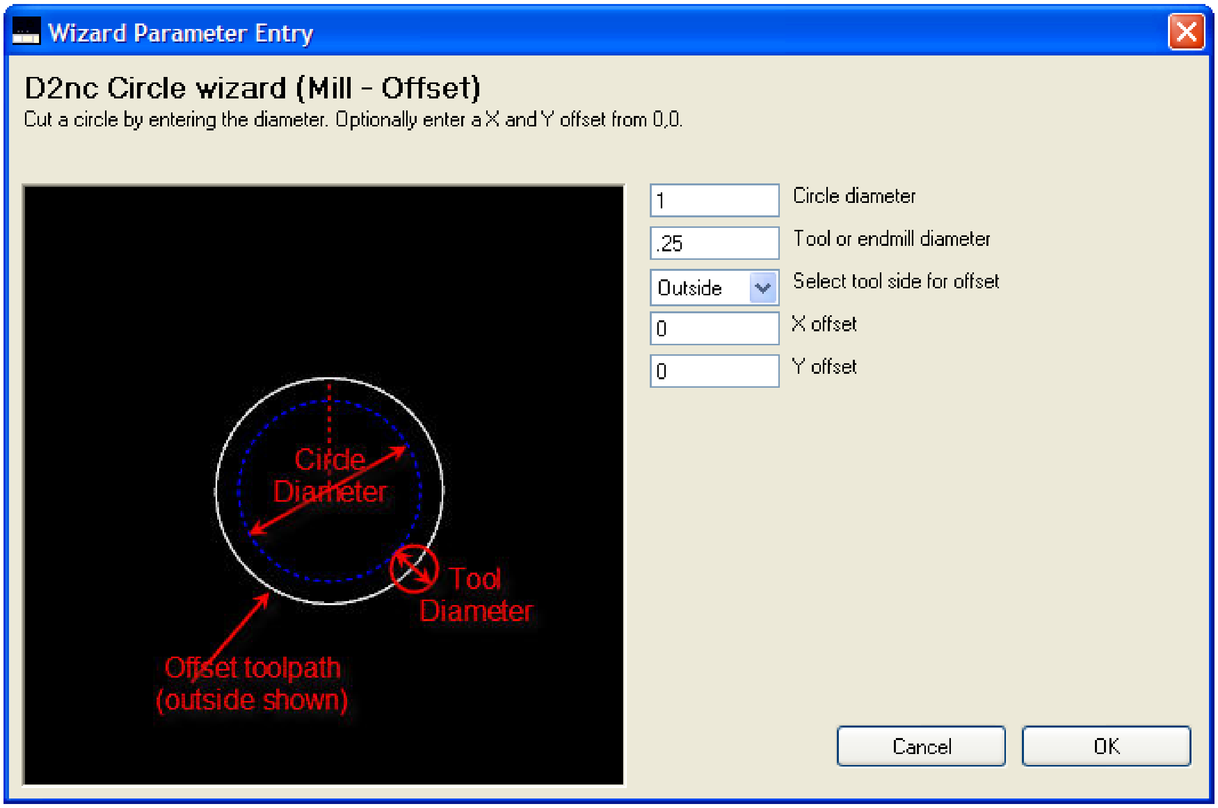


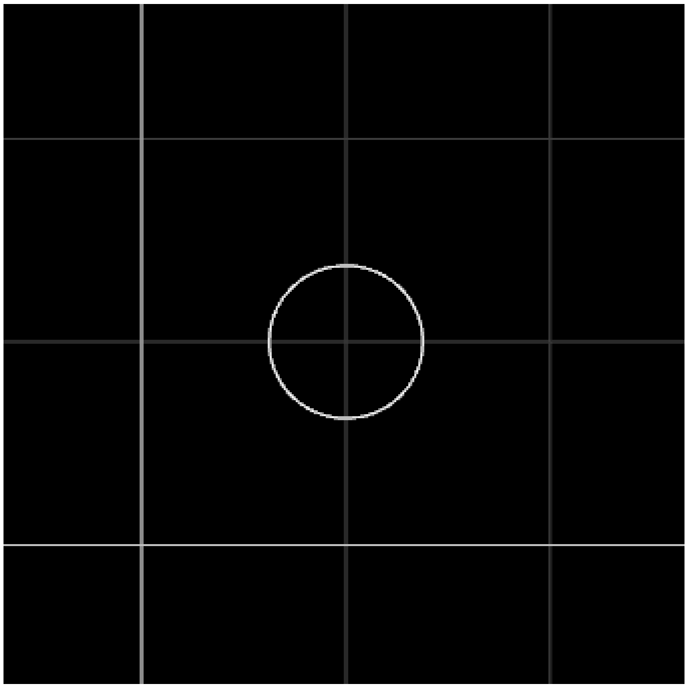
<length=3>
<width=2>
<offset_x=1>
<offset_y=1>

<tool_dia=.25>
<step_pct=50>
<ang=0>

## 4.14 Mill - CL - Shape - Circle

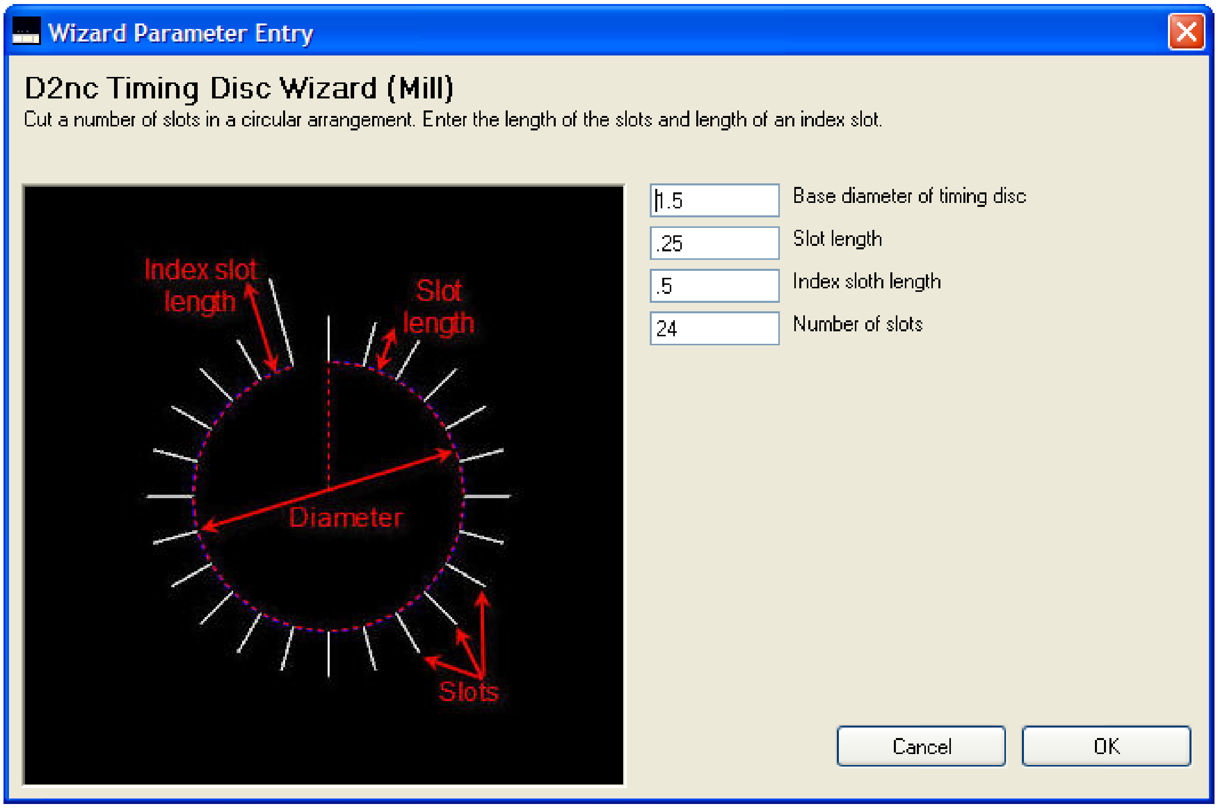



1 inch circle compensated inside for a 1/4 in tool

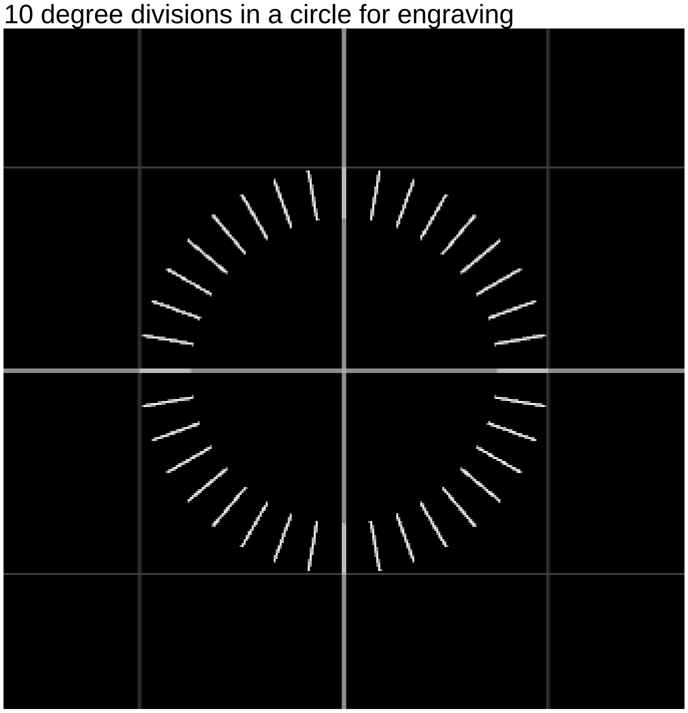


<dia=1>
<tool_dia=.25>
<side=2>
<x_offset=1>
<y_offset=1>

## 4.15 Mill - CL - Slotted Timing Disk



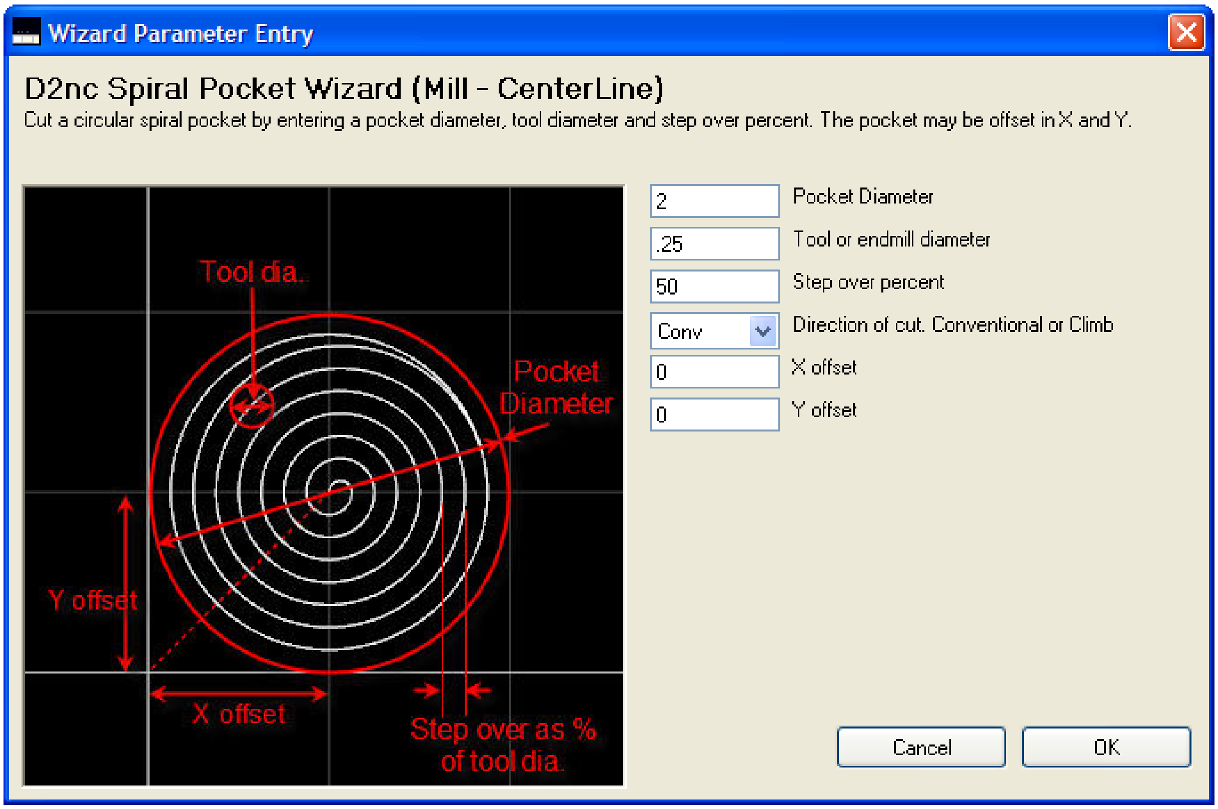

10 degree divisions in a circle for engraving



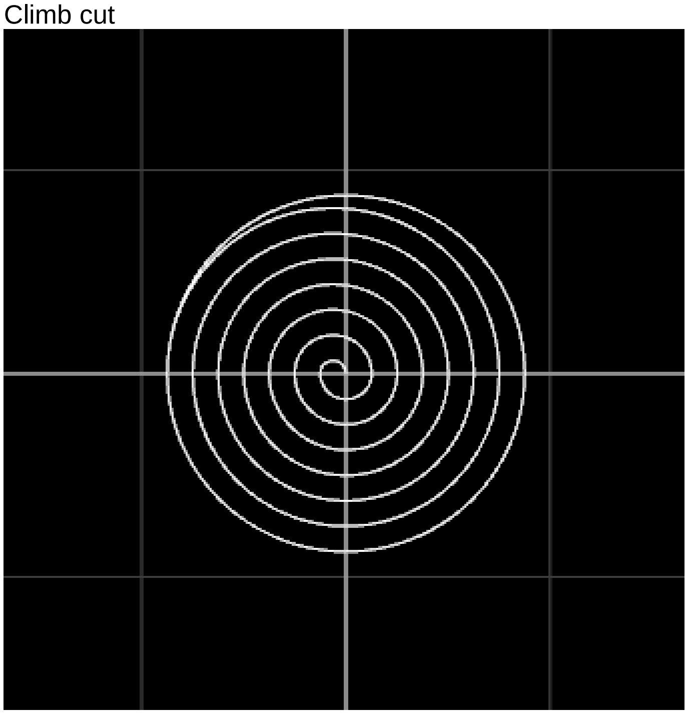
<base_dia=1.5>
<slot_len=.25>
<idx_len=.25>

<nr_slots=36>

## 4.16 Mill - CL - Spiral Pocket

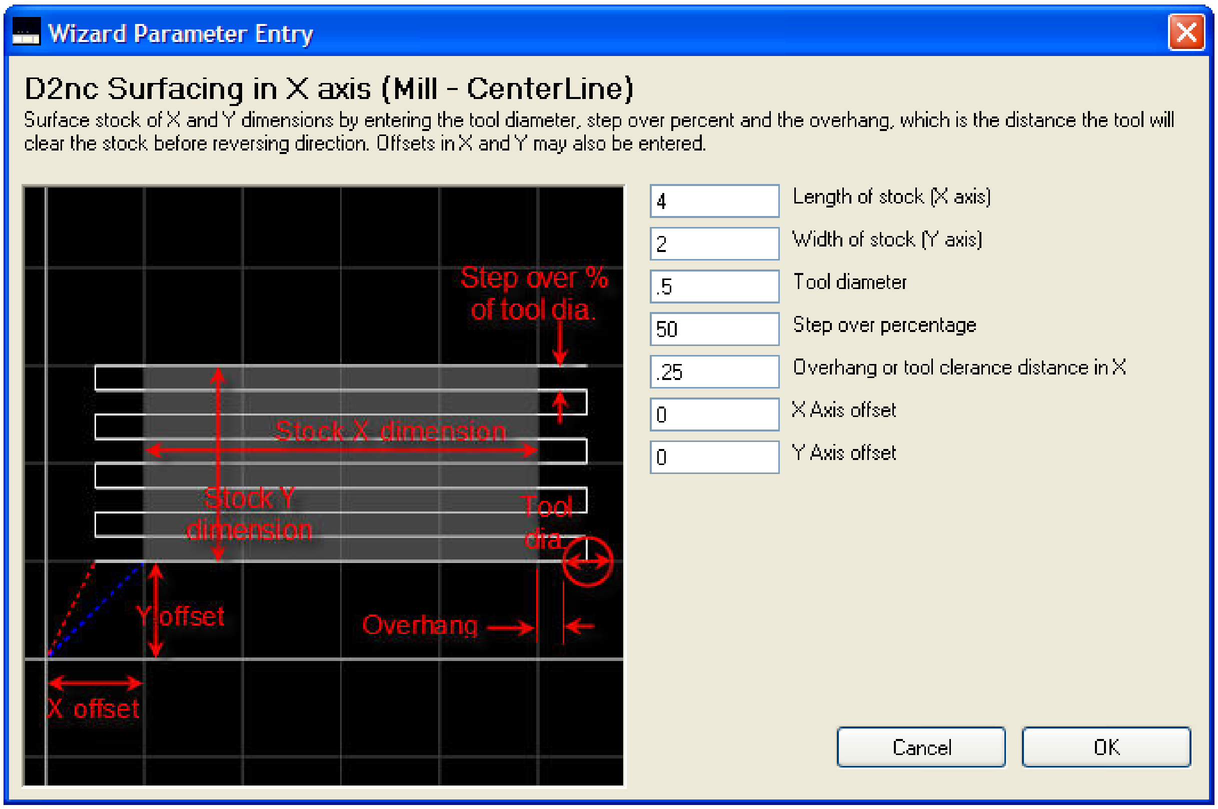


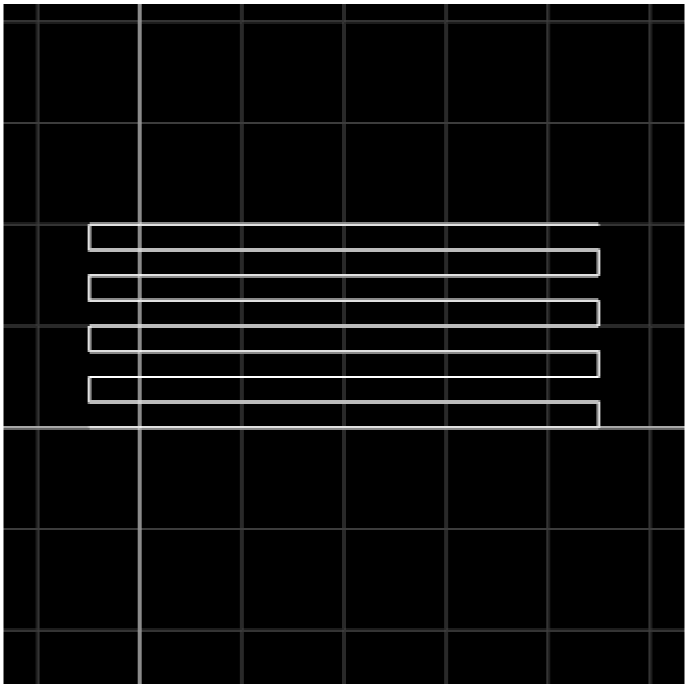

Climb cut



<pkt_dia=2>
<tool_dia=.25>
<step_pct=50>
<dir=-1>
<x_offset=0>
<y_offset=0>

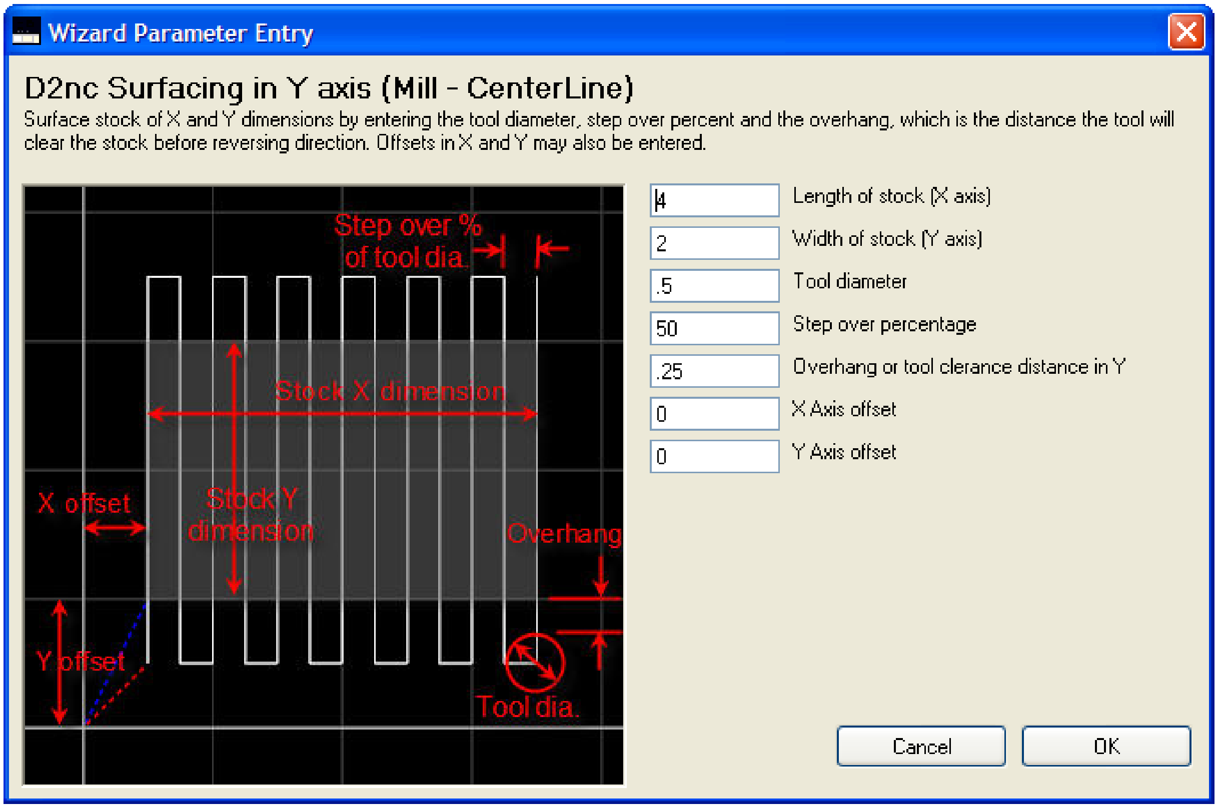
## 4.17 Mill - CL - Surfacing in X axis

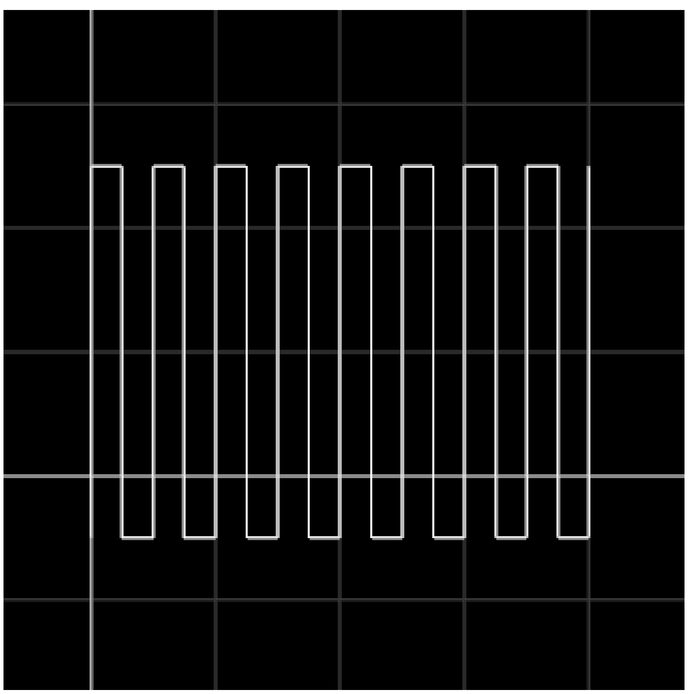




<stock_x=4>
<stock_y=2>
<tool_dia=.5>
<stepover=50>
<x_over=.25>
<x_offset=0>
<y_offset=0>

## 4.18 Mill - CL - Surfacing in Y axis

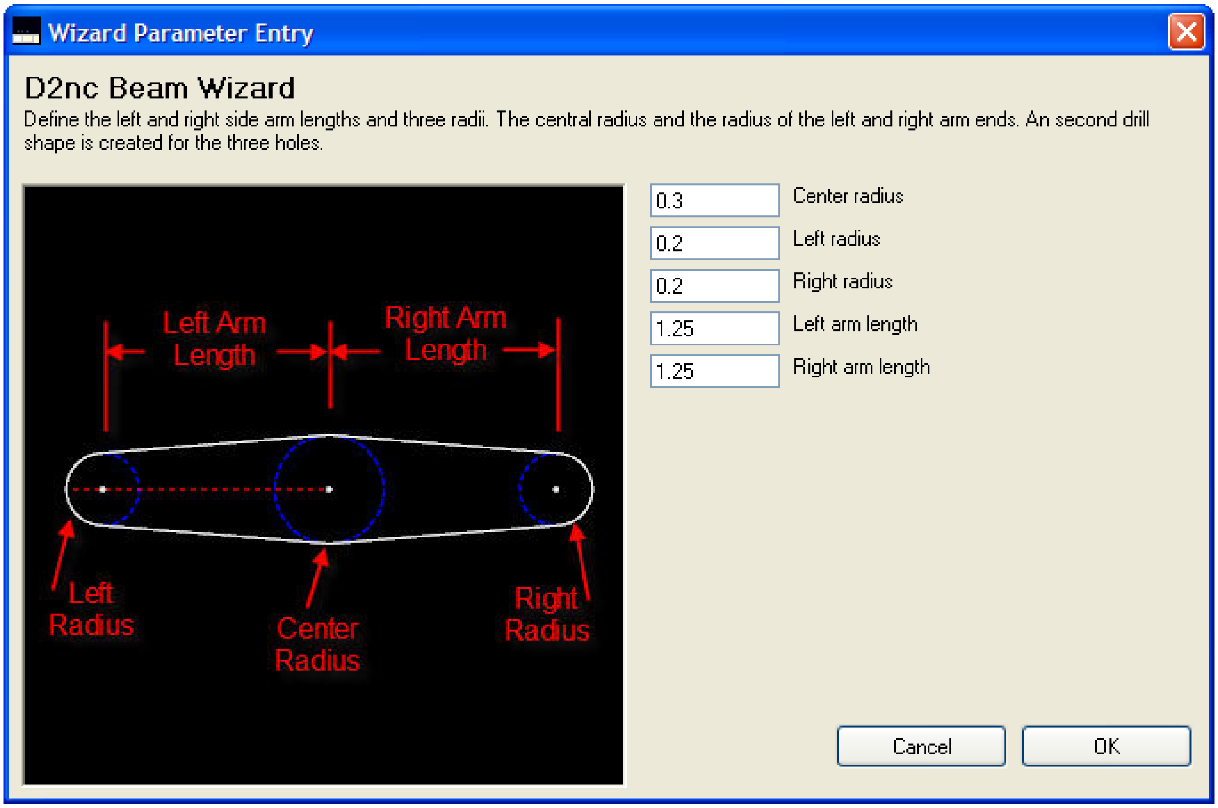


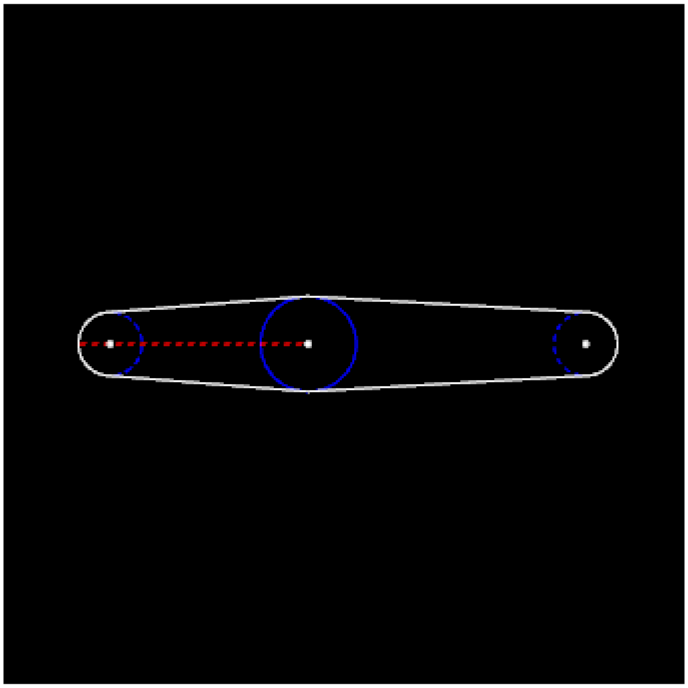
<stock_x=4>
<stock_y=2>
<tool_dia=.5>
<stepover=50>

<y_over=.25>
<x_offset=0>
<y_offset=0>

## 4.19 Mill - Multi - Engine Beam

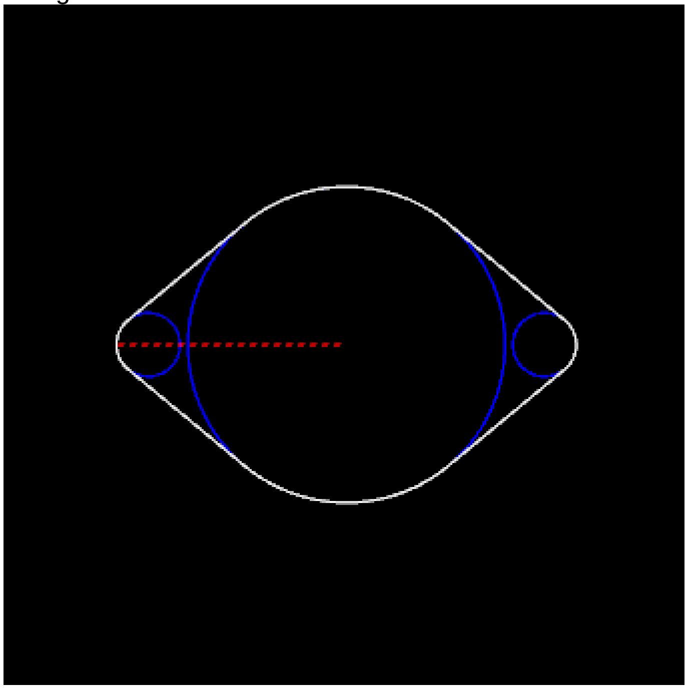



Unequal arm lengths



<c_rad=0.3>
<l_rad=0.2>
<r_rad=0.2>
<l_len=1.25>
<r_len=1.75>

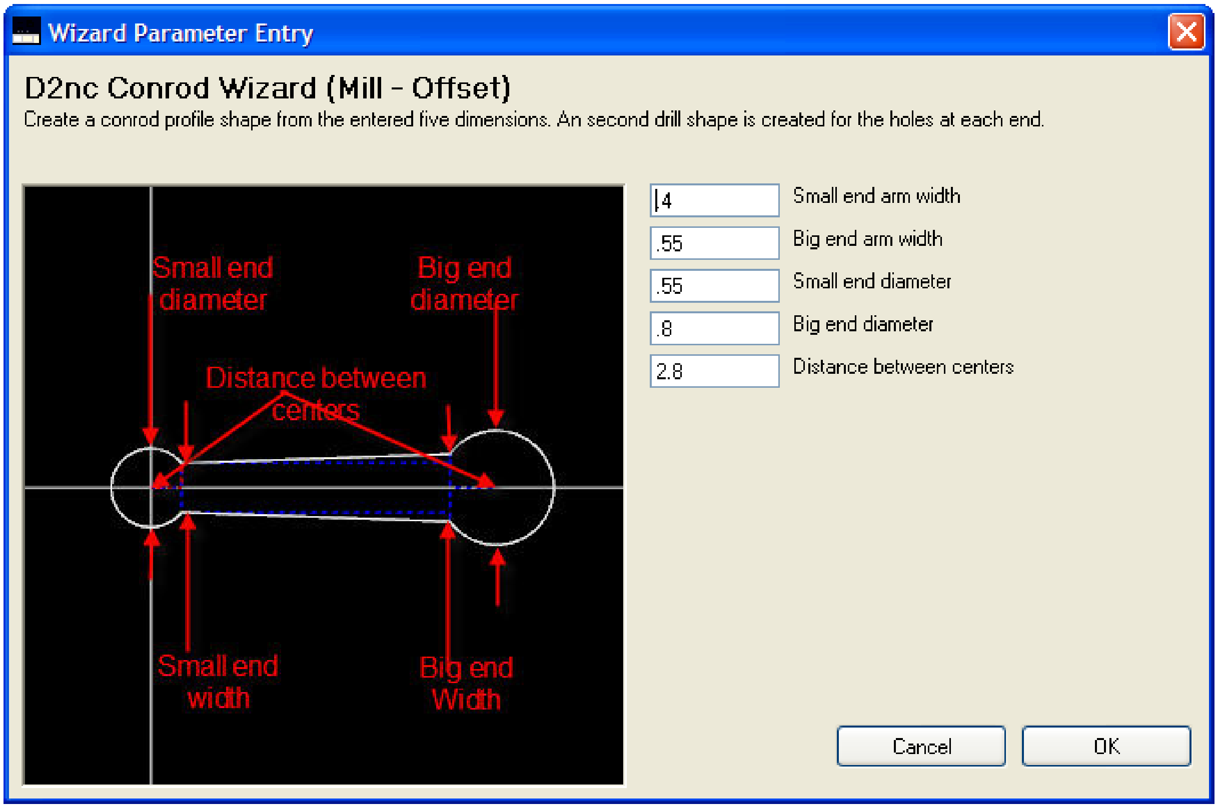
Flange cutout

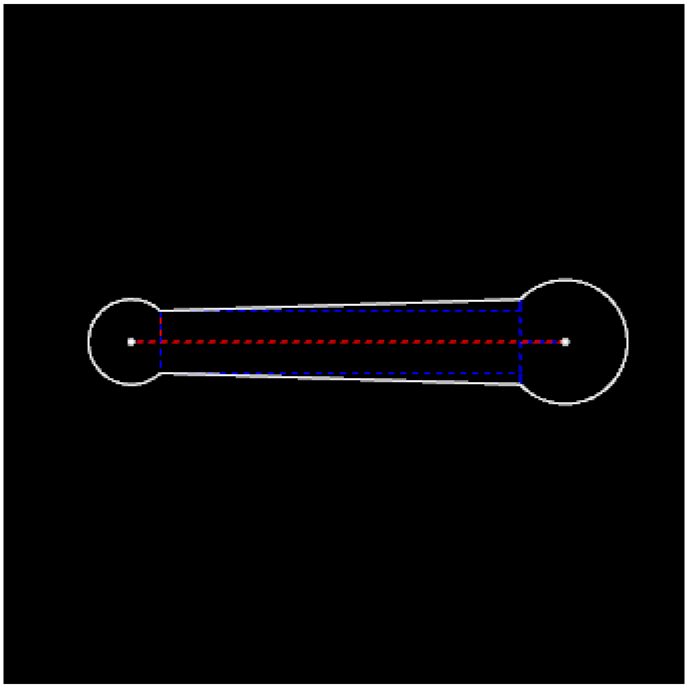


<c_rad=1>
<l_rad=0.2>
<r_rad=0.2>
<l_len=1.25>

<r_len=1.25>

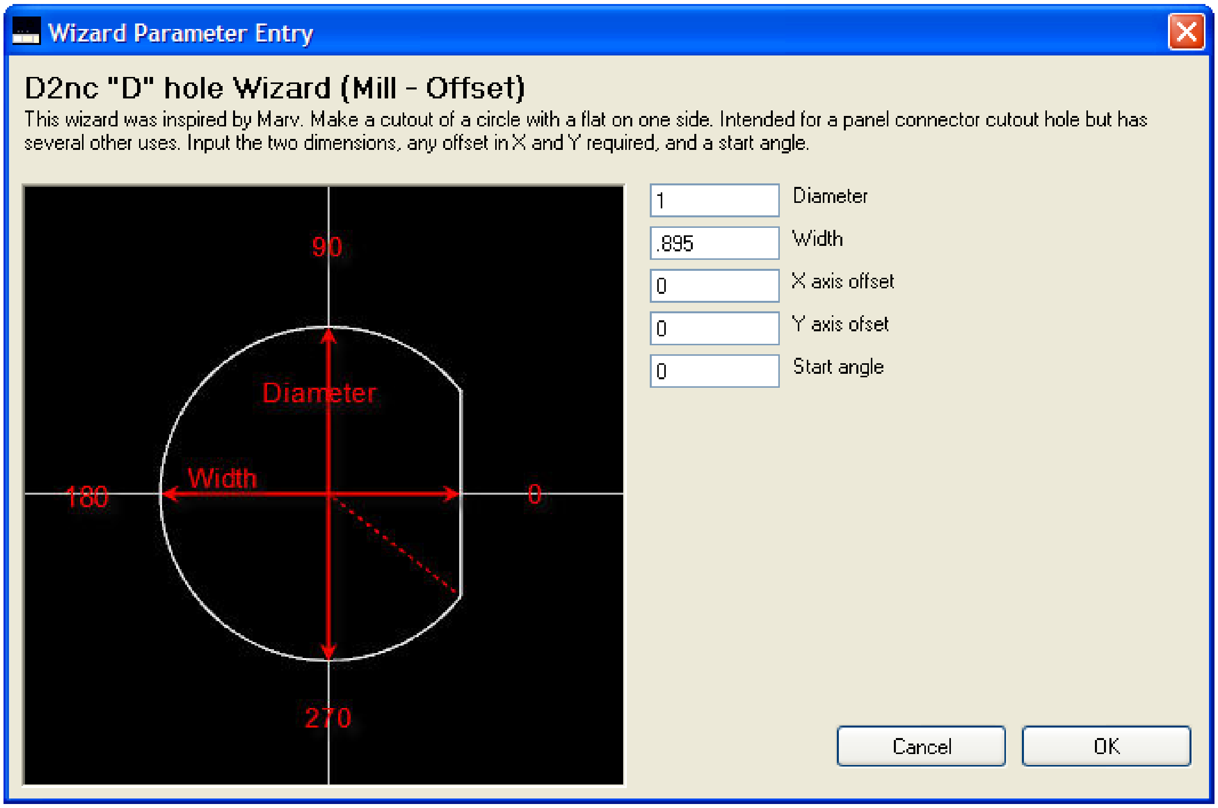
## 4.20 Mill - Multi - Engine Conrod

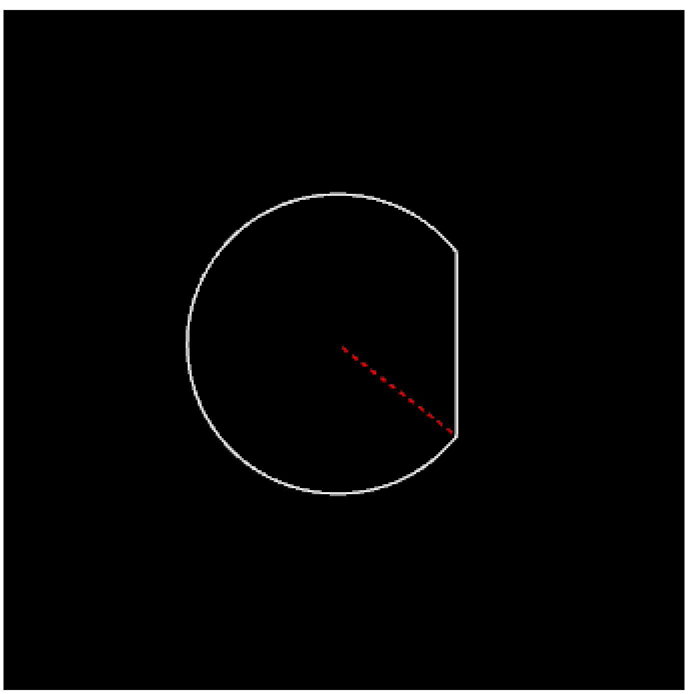




<s_arm=.4>
<b_arm=.55>
<s_dia=.55>
<b_dia=.8>
<dbc=2.8>

## 4.21 Mill - Off - 'D' Hole

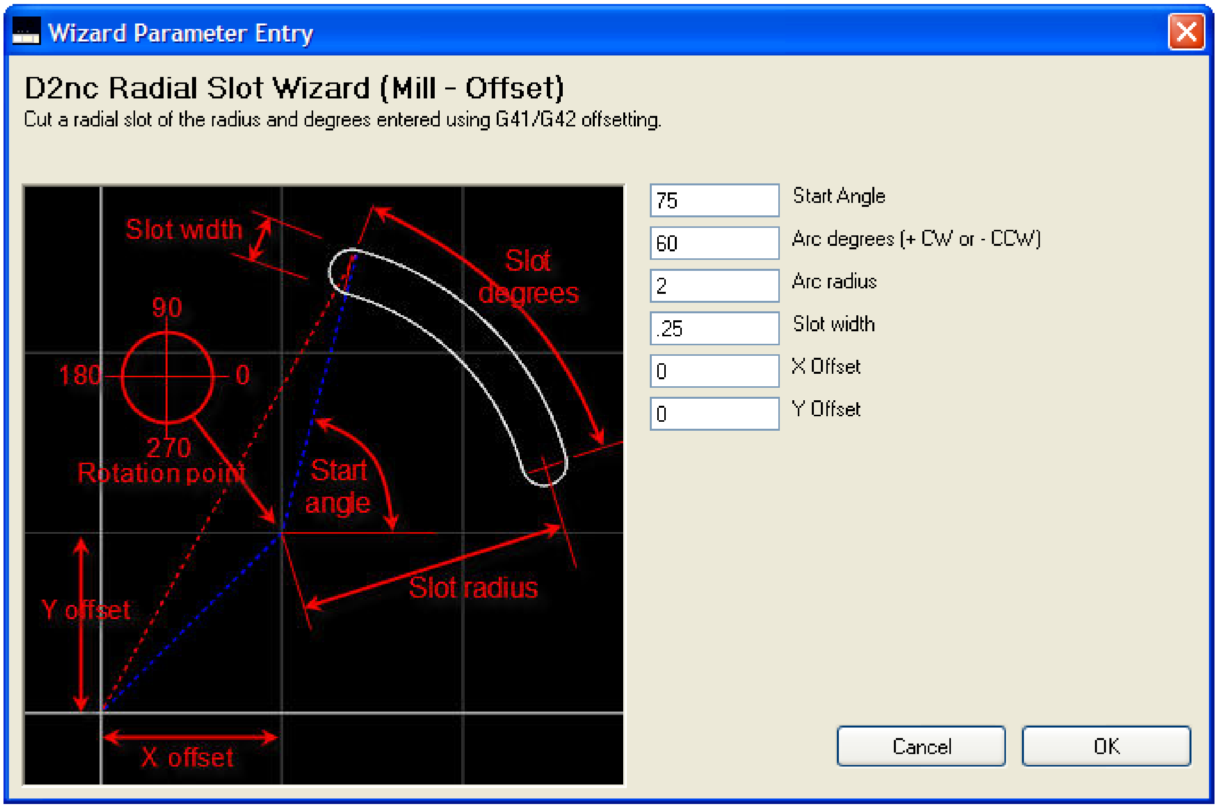


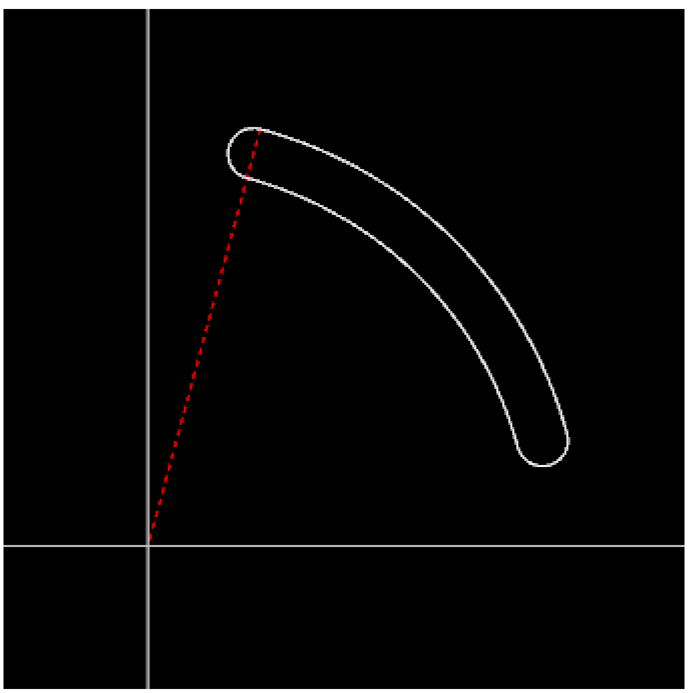



<dia=1>
<acc=.895>
<x_offset=0>
<Y_offset=0>

<start_ang=0>

## 4.22 Mill - Off - Radial Slot

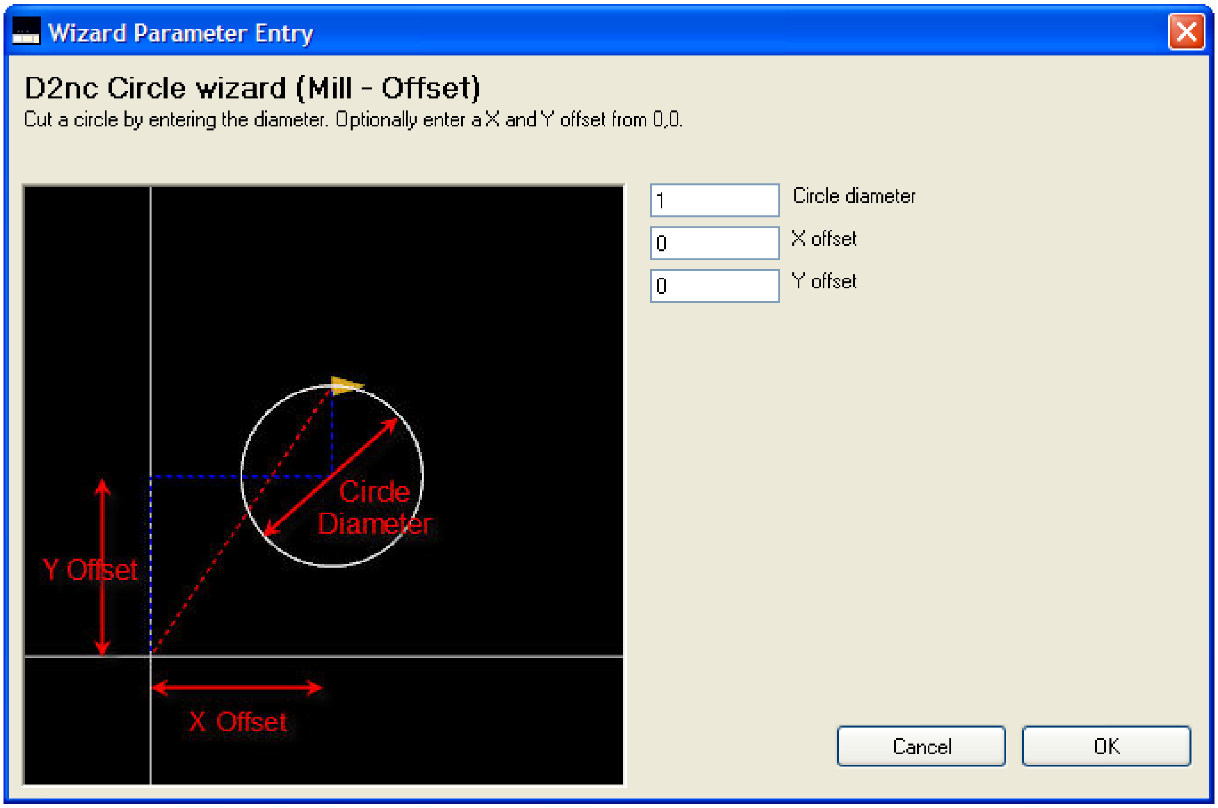


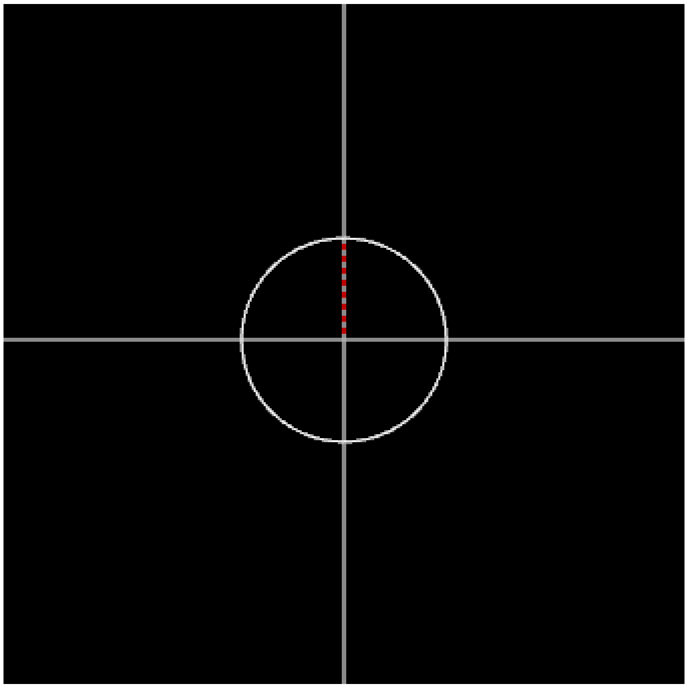



<start_angle=75>
<arc_deg=60>

<arc_rad=2>
<slot_width=.25>
<x_offset=0>
<y_offset=0>

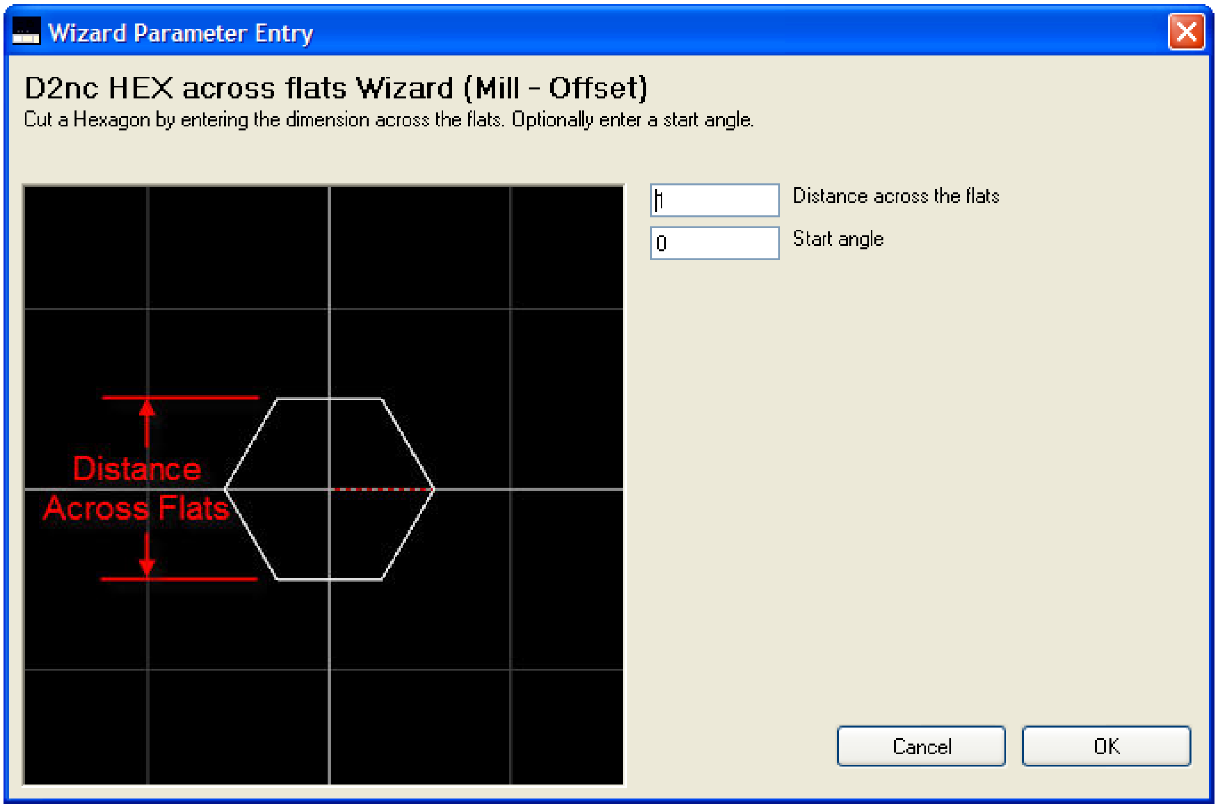
## 4.23 Mill - Off - Shape - Circle

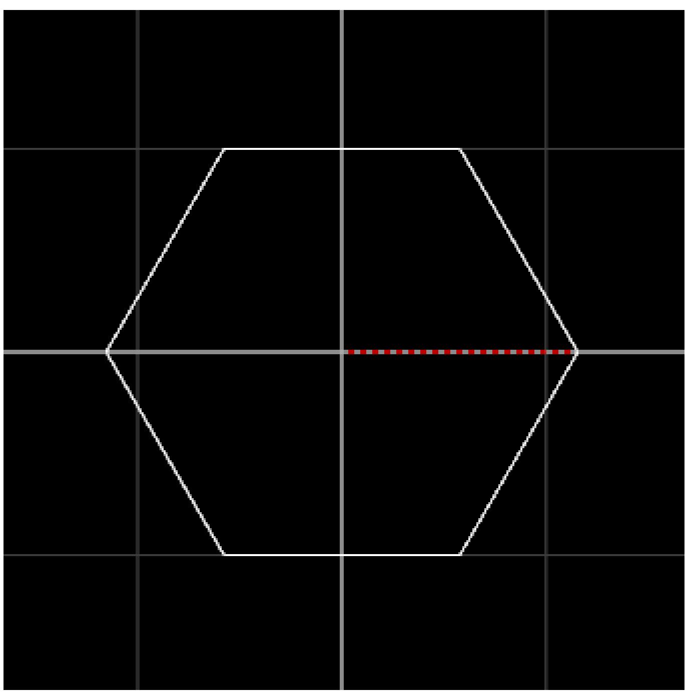




<dia=1>
<x_offset=0>
<y_offset=0>

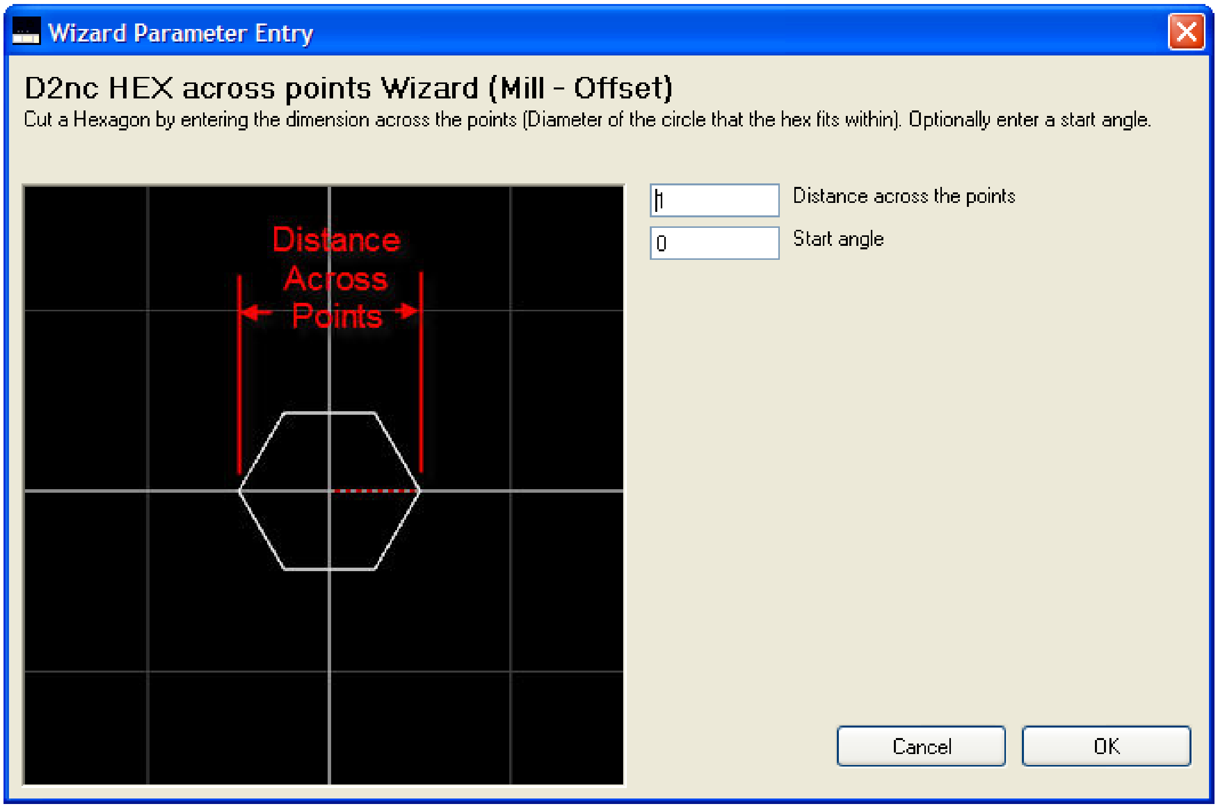
## 4.24 Mill - Off - Shape - HEX - across flats

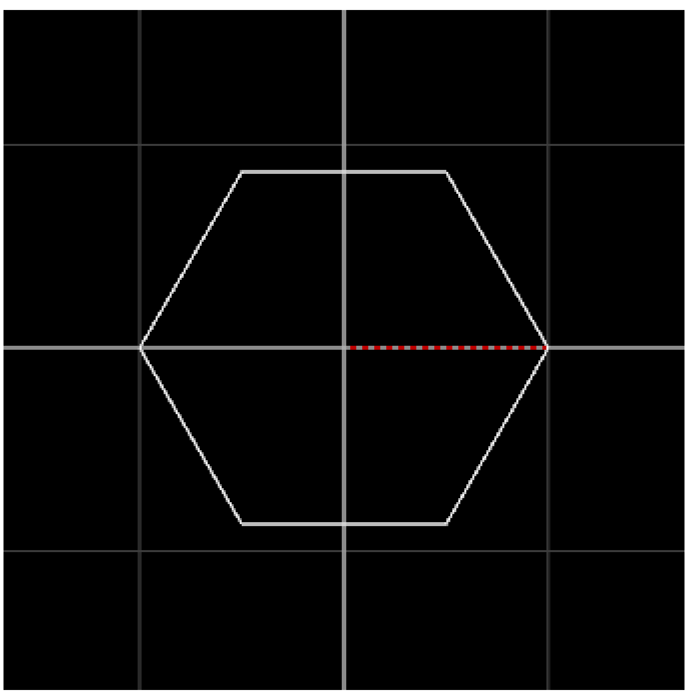


<acc_flat=2>
<start_ang=0>

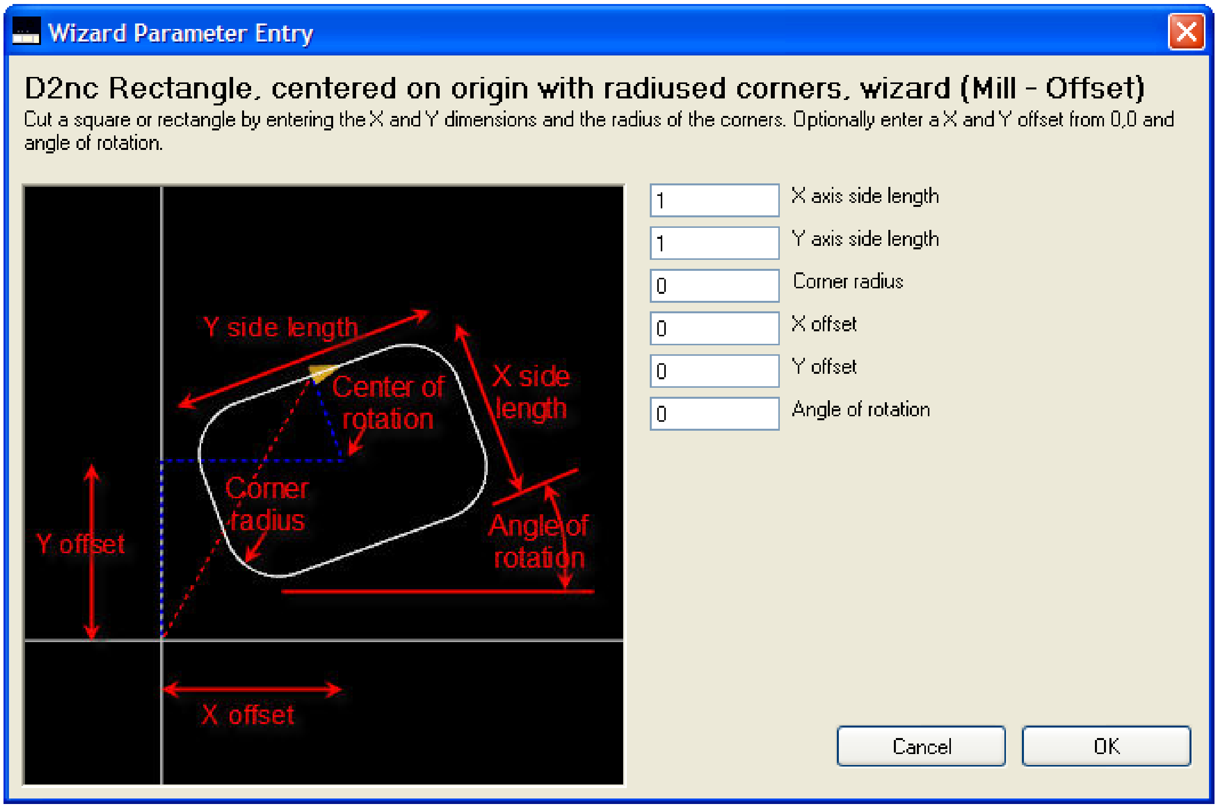
## 4.25 Mill - Off - Shape - HEX - across points

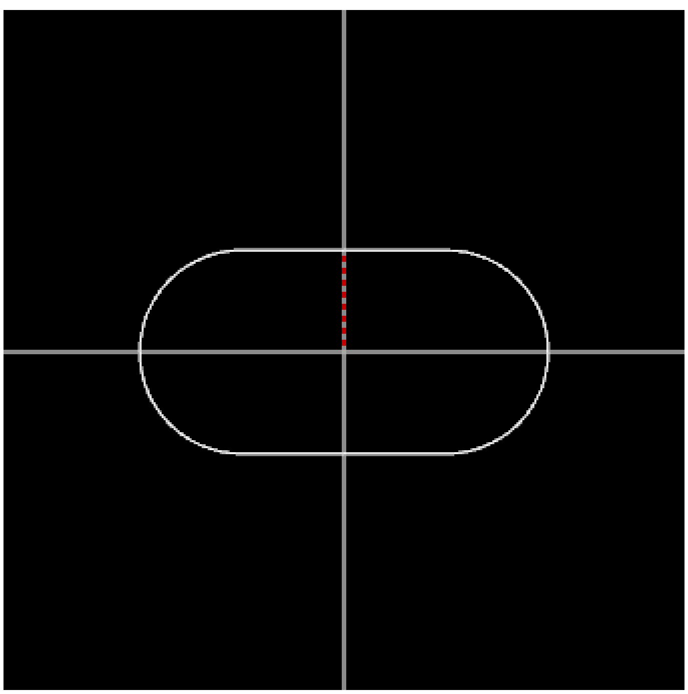


<cir_dia=2>
<start_ang=0>

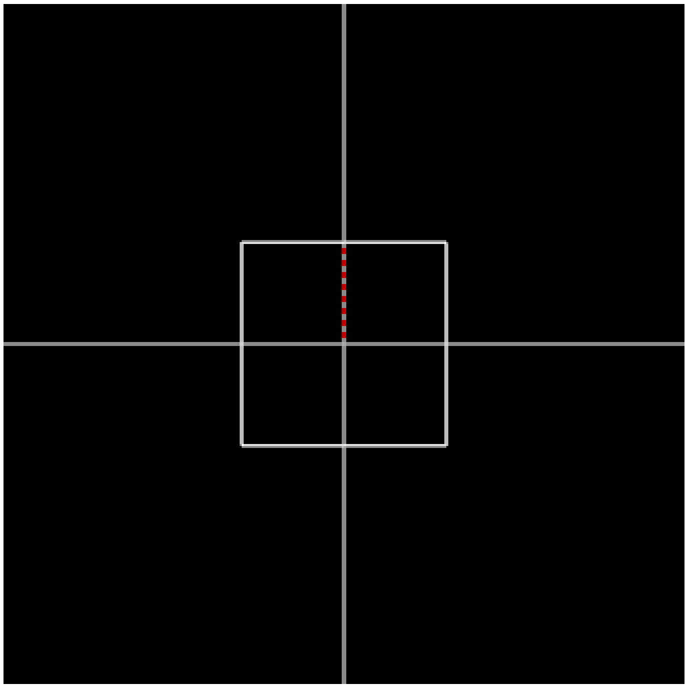
## 4.26 Mill - Off - Shape - Rectangle Centered Radius



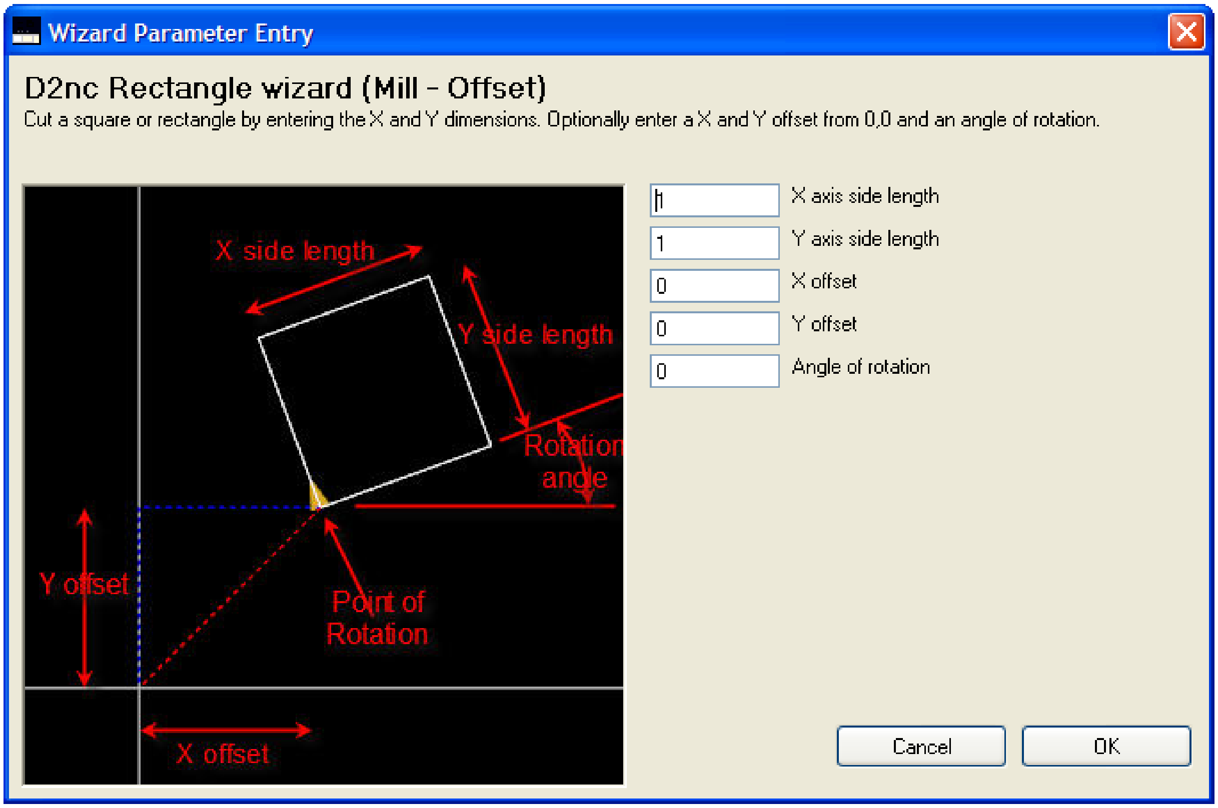
<x_side=2>
<y_side=1>
<rad=.5>
<x_offset=0>

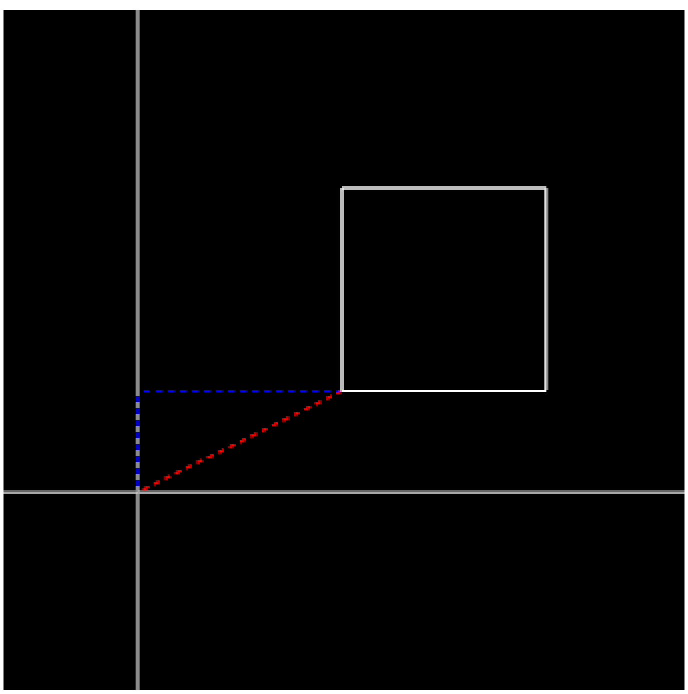
<y_offset=0>
<rot=0>



<x_side=1>
<y_side=1>
<rad=0>
<x_offset=0>
<y_offset=0>
<rot=0>

## 4.27 Mill - Off - Shape - Rectangle



<x_side=1>
<y_side=1>
<x_offset=1>
<y_offset=.5>

<rot=0>

# 5 Shape Description Language

The Shape Description Language definition.

## 5.1 Assignment

**Definition**

**<** varname = value **|** *varname* | mathfunction >

**<** varname = *varname* *operator* *value* **|** *varname* **|** *mathfunction* **>**

**Varname**
Variable name may be any alphanumeric name starting with the letter A thru Z but must not be:
R0 thru R9 which are reserved as a form of the repeat command and,
must not be any of the math function names.

**Operators**

| | |
|---|---|
| + | Used to sum two numbers |
| - | Used to find the difference between two numbers |
| / | Used to divide two numbers |
| * | Used to multiply two numbers |
| ^ | Used to raise a number to the power of an exponent |
| % | Used to divide two numbers and return only the remainder |

**Math functions**

| | |
|---|---|
| ABS(x) | Returns the absolute value of a number<br>The absolute value of a number is its unsigned magnitude.<br>For example, ABS(-1) and ABS(1) both return 1. |
| ACOS(x) | Inverse Cosine<br>ACOS(x) = ATAN(-x / SQR(-x * x + 1)) + 2 * ATAN(1) |
| ASIN(x) | Inverse Sine<br>ASIN(x) = ATAN(x / SQR(-x * x + 1)) |
| ATAN(x) | Returns the arctangent of a number.<br>The Atan function takes the ratio of two sides of a right triangle (number) and returns the corresponding angle in radians. The ratio is the length of the side opposite the angle divided by the length of the side adjacent to the angle.<br>The range of the result is -pi/2 to pi/2 radians. |
| COS(x) | Returns the cosine of an angle x.<br>The angle must be expressed in radians.<br>The Cos function takes an angle and returns the ratio of two sides of a right triangle. The ratio is the length of the side adjacent to the angle divided by the length of the hypotenuse.<br>The result lies in the range -1 to 1. |
| COT(x) | Cotangent<br>COT(x) = 1 / TAN(x) |
| CSC(x) | Cosecant<br>CSC(x) = 1 / SIN(x) |
| DTR(x) | Converts x from degrees to radians.<br>To convert degrees to radians, the degrees are multiply by pi/180 |
| EXP(x) | Returns e (the base of natural logarithms) raised to a power.<br>If the value of number exceeds 709.782712893, an error occurs. The constant e is approximately 2.718282.<br>The Exp function complements the action of the Log function and is sometimes referred to as the antilogarithm. |
| INT(x) | Returns the integer portion of x.<br>Int returns the first negative integer less than or equal to number.<br>For example, Int converts -7.4 to -8 |
| LN(x) | Returns the natural logarithm of a number.<br>The natural logarithm is the logarithm to the base e. The constant e is approximately 2.718282. |
| LOG(x,y) | Returns the logarithm of a number x in the base y. Also see LN<br>The logarithm in base y for any number x is calculated by dividing the natural logarithm of x by the natural logarithm of y as follows:<br>Log(x) = LN(x) / LN(y) |
| RTD(x) | Converts x from radians to degrees.<br>To convert radians to degrees, the radians are multiply by 180/pi. |
| SEC(x) | Secant<br>SEC(x) = 1 / COS(x) |
| SGN(x) | Returns an integer indicating the sign of x.<br>If x is greater that zero, SGN returns 1.<br>If x is equal to zero, SGN returns 0.<br>If x is less that zero, SGN returns -1. |
| SIN(x) | Returns the sine of an angle x<br>The angle must be expressed in radians.<br>The Sin function takes an angle and returns the ratio of two sides of a right triangle. The ratio is the length of the side opposite the angle divided by the length of the hypotenuse.<br>The result lies in the range -1 to 1. |
| SQR(x) | Returns the square root of a number.<br>The value of x must be greater than or equal to zero. |
| TAN(x) | Returns a Double specifying the tangent of an angle.<br>The angle must be expressed in radians.<br>Tan takes an angle and returns the ratio of two sides of a right triangle. The ratio is |

## 5.2 Functions

**Definition**

**(** *varname* **)**

**(** *varname* *operator* *value* **|** *varname* **|** *mathfunction* **)**

**(** varname = *varname* *operator* *value* **|** *varname* **|** *mathfunction* **)**

**Varname**
Variable name may be any alphanumeric name starting with the letter A thru Z but must not be:
R0 thru R9 which are reserved as a form of the repeat command and,
must not be any of the math function names.

Operators

| | |
|---|---|
| + | Used to sum two numbers |
| - | Used to find the difference between two numbers |
| / | Used to divide two numbers |
| * | Used to multiply two numbers |
| ^ | Used to raise a number to the power of an exponent |
| % | Used to divide two numbers and return only the remainder |

**Math functions**

| | |
|---|---|
| ABS(x) | Returns the absolute value of a number<br>The absolute value of a number is its unsigned magnitude.<br>For example, ABS(-1) and ABS(1) both return 1. |
| ACOS(x) | Inverse Cosine<br>ACOS(x) = ATAN(-x / SQR(-x * x + 1)) + 2 * ATAN(1) |
| ASIN(x) | Inverse Sine<br>ASIN(x) = ATAN(x / SQR(-x * x + 1)) |
| ATAN(x) | Returns the arctangent of a number.<br>The Atan function takes the ratio of two sides of a right triangle (number) and returns the corresponding angle in radians. The ratio is the length of the side opposite the angle divided by the length of the side adjacent to the angle.<br>The range of the result is -pi/2 to pi/2 radians. |
| COS(x) | Returns the cosine of an angle x.<br>The angle must be expressed in radians.<br>The Cos function takes an angle and returns the ratio of two sides of a right triangle. The ratio is the length of the side adjacent to the angle divided by the length of the hypotenuse.<br>The result lies in the range -1 to 1. |
| COT(x) | Cotangent<br>COT(x) = 1 / TAN(x) |
| CSC(x) | Cosecant<br>CSC(x) = 1 / SIN(x) |
| DTR(x) | Converts x from degrees to radians.<br>To convert degrees to radians, the degrees are multiply by pi/180 |
| EXP(x) | Returns e (the base of natural logarithms) raised to a power.<br>If the value of number exceeds 709.782712893, an error occurs. The constant e is approximately 2.718282.<br>The Exp function complements the action of the Log function and is sometimes referred to as the antilogarithm. |
| INT(x) | Returns the integer portion of x.<br>Int returns the first negative integer less than or equal to number.<br>For example, Int converts -7.4 to -8 |
| LN(x) | Returns the natural logarithm of a number.<br>The natural logarithm is the logarithm to the base e. The constant e is approximately 2.718282. |
| LOG(x,y) | Returns the logarithm of a number x in the base y. Also see LN<br>The logarithm in base y for any number x is calculated by dividing the natural logarithm of x by the natural logarithm of y as follows:<br>Log(x) = LN(x) / LN(y) |
| RTD(x) | Converts x from radians to degrees.<br>To convert radians to degrees, the radians are multiply by 180/pi. |
| SEC(x) | Secant<br>SEC(x) = 1 / COS(x) |
| SGN(x) | Returns an integer indicating the sign of x.<br>If x is greater that zero, SGN returns 1.<br>If x is equal to zero, SGN returns 0.<br>If x is less that zero, SGN returns -1. |
| SIN(x) | Returns the sine of an angle x<br>The angle must be expressed in radians.<br>The Sin function takes an angle and returns the ratio of two sides of a right triangle. The ratio is the length of the side opposite the angle divided by the length of the hypotenuse.<br>The result lies in the range -1 to 1. |
| SQR(x) | Returns the square root of a number.<br>The value of x must be greater than or equal to zero. |
| TAN(x) | Returns a Double specifying the tangent of an angle.<br>The angle must be expressed in radians.<br>Tan takes an angle and returns the ratio of two sides of a right triangle. The ratio is |

## 5.3 Conditional test

**Definition**

**:** conditional test (=,<,>)**;** true commands **;** false commands **;**

**Usage**

The conditional test works like the IIF function from visual basic. A good reference on the use of IIF can be found in the [wikipedia IIF]

**:** The colon indicates the start of the conditional test
**;** The first semi-colon is the start of the true SDL
**;** The second semi-colon is the start of the false SDL
**;** The third semi-colon ends the false code and is the end of the conditional statement

All four, the colon and three semi-colons must be present to form a valid conditional statement.

**Example**

To toggle a heading between 90 and -90 degrees based on its current setting:

**:** a = 90 **;** <a=-90> **;** < a=90> **;**

Test **:** Is a = 90 **;** If true set a = -90 **;** If false set a = 90 **;**

## 5.4 H - Heading

**Definition**

H [=] *heading*

| | |
|---|---|
| H | Heading command |
| = | *optional* modifier to set an absolute heading with zero being a positive direction on the x axis. |
| *heading* | Range -360 to [+]360 sets the heading relative to the current heading. |

**Usage**

Use to change the current heading. Any subsequent command starts on the new current heading.

**Example**

Any change in heading is relative to the current heading. A positive change is clockwise and a negative, counter clockwise. Every shape 115 has its own heading. Every shape starts with a heading straight up or north or, in absolute terms, a heading of 90 degrees. When the D2nc starts, shape 0 is active by default and the heading indicator is active in the shape display area 9.

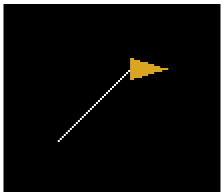
The heading is changed by this H command or it can also be changed by the optional heading qualifier to the Draw 97, Jump 97, Arc 100, Warp 101, Convex 117 and Concave 118 commands.

The heading has a optional *absolute* qualifier which is invoked by setting the heading with the equal sign. It is recommended that you avoid using an absolute setting for the heading as it breaks the relative nature of the language and causes problems with rotating shapes with a warp. Having said that, it does provide function for some difficult shapes and has been made available for those instances. The absolute setting must be followed by a heading of 0 to <360 degrees and is not available as a qualifyer on those commands that support the heading qualifyer.

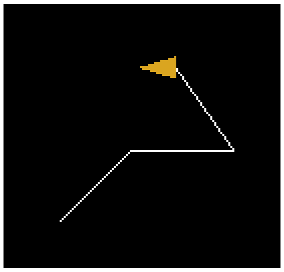
We take the following program:

```
h45
d.5,45
d.5,-125
d.5
h=180
d.2
&
```

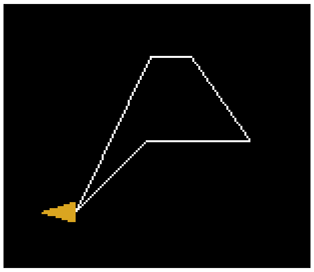
and step through it:



| | |
|---|---|
| `h45` | Set heading 45 degrees clockwise from the initial 90 setting |
| `d.5,45` | Draw line .5 long then set heading 45 degrees clockwise |



| | |
|---|---|
| `d.5,-125` clockwise | Draw line .5 long then set heading 125 degrees counter |
| `d.5` | Draw line .5 long |
| `H=180` | Set the heading to an absolute heading of 180 degrees. Disregard current heading |



| | |
|---|---|
| `d.2` | Draw a line .2 long |
| `&` | Close the shape leaving the heading unchanged |

## 5.5 D - Draw

**Definition**

D *distance*[,*heading*]

| | |
|---|---|
| D | Draw command |
| *distance* | Distance in working units to draw line in the direction of the current heading. |
| *heading* | *optional* range -360 to [+]360 sets the heading relative to the current heading. |

**Usage**

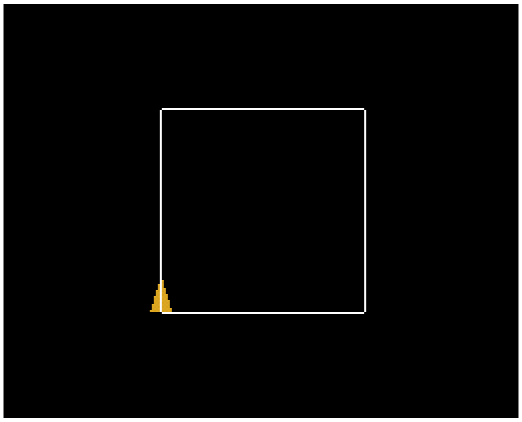
Draw straight lines on the current heading of the specified distance. Draw is used to cut material and create G01 moves when translated to g-code.

**Example**

Enter the following into the Shape Description Area 11:

```
d 1
h 90
d 1
h 90
d 1
h 90
d 1
h 90
```

which will produce the following display in the Shape Display Area 9.



Let's examine what happened here. The first "d1" drew a line in the direction of the heading indicator. The "h90" caused the heading indicator to turn 90 degrees clockwise. The next "d1" drew a line in the direction of the heading indicator. This second line is at a right angle to the first line due to the change in heading. The second line started from where the first line ended.

## 5.6 J - Jump

**Definition**

J *distance*[,*heading*] [ **!** | @ ]

| | |
|---|---|
| J | Jump command |
| *distance* | Distance in working units to jump in the direction of the current heading. |
| *heading* | *optional* range -360 to [+]360 sets the heading relative to the current heading. |
| **!** (option 1) | *optional* line converter. Cause a line to be drawn to this point. Any sequence of jumps and warps will be converted into a cutting line from the last cut or anchor point. |
| @ (option 2) | *optional* anchor this point. Excludes this jump being converted to construction line for a subsequent jump or to a cut on a subsequent jump or warp option 1. |

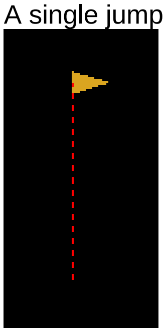
**Usage**

Jump on the current heading for the specified distance. Jump is used to position the tool with G00 moves when translated to g-code.

**Example**

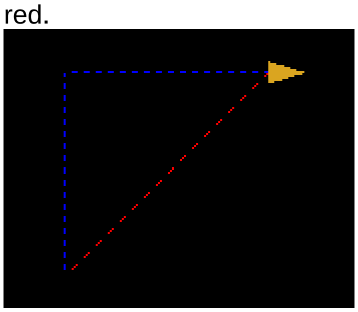
The jump 97 works as you would expect but has a few qualifiers and some behaviour that needs further explanation. The jump, when translated to g-code, equates to the tool making rapid moves above the cutting plane at the safe Z height as set in constraints 14 . In the shape display area the jump is represented by a red dashed line.

A single jump with a heading change produces the following:



```
j1,90
```

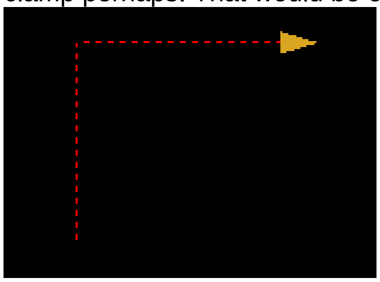
Consider a subsequent jump. Normally you should expect the tool to make a rapid move on the shortest possible path. To meet this expectation, when there are two or more jumps in a row, they are converted to construction lines (blue) and the resulting real jump or tool move calculated and shown in red.



```
j1,90
j1
```

There are two qualifiers on the jump command, an @ (anchor) and ! (draw line to here).

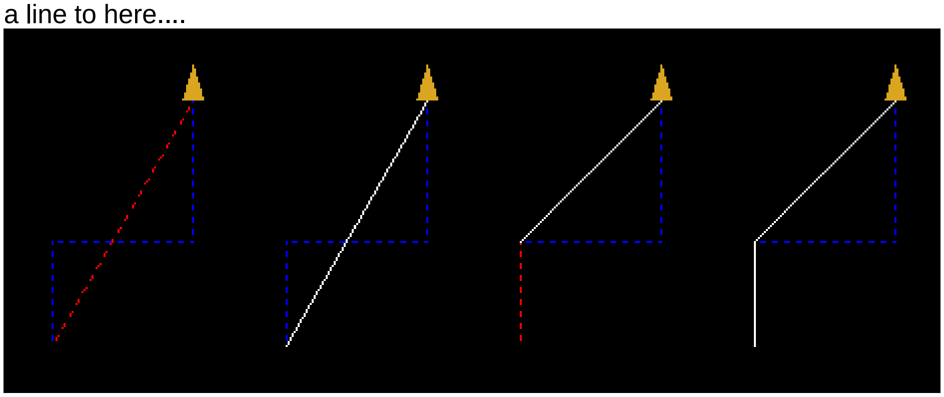
There may have been a reason to have the tool follow both jumps in the previous example, to avoid a clamp perhaps. That would be one of the uses for the anchor.



```
j1,90@
j1
```

**Positioning with Jump (or the closest thing to relative coordinates)**

Jump can also be used to reach a position where you cannot possibly know the correct heading and line distance to that point. This is most useful when working from a drawing.  Up .3 right .4 up.4 create a line to here....



First follow the directions:

```
j.3,90
j.4,-90
j.4
```

Second convert the jump to a line:

```
j.3,90
j.4,-90
j.4!
```

Third, lets assume the first jump was a positioning not a cutting move so lets anchor it:

```
j.3,90@
j.4,-90
j.4!
```

Fourth, if you actually wanted a cutting move instead of a positioning move you would use the draw 97

```
d.3,90
j.4,-90
j.4!
```

## 5.7 A - Arc

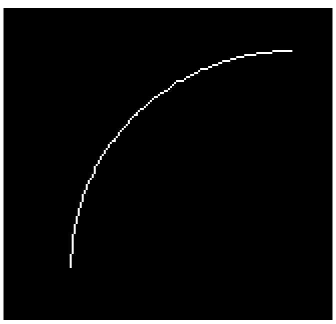
**Definition**

A *angle*,*radius*[,*heading*]

| | |
|---|---|
| A | Arc command |
| *angle* | Range -360 to [+]360. Angle of the arc.<br>-ve moves in a CCW direction and +ve in a CW direction. |
| *radius* | Radius of the arc in working units |
| *heading* | *optional* range -360 to [+]360 sets the heading relative to the current heading. |

**Usage**

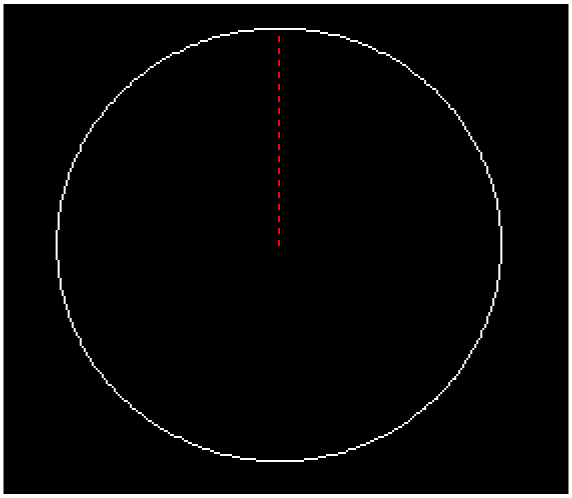
The arc parameter angle sets the direction of the arc: clockwise is a positive angle and counter clockwise is a negative angle. The radius sets a point to the right or left of the starting point depending on whether the angle is positive or negative. Arc is used to cut material and create G02 and G03 moves when translated to g-code.



`a90,1`

**Example**

In this next example, a jump and cw heading change is made from the origin point. The arc of 360 degrees has the same radius as the jump which puts the start position, a default of x,y coordinate of 0,0 at the center of the circle.



```
j1,90
a360,1
```

## 5.8 W - Warp

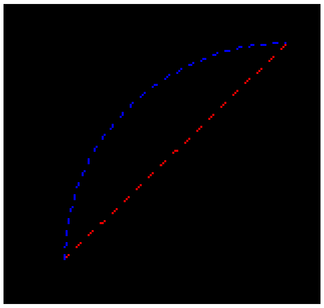
**Definition**

W *angle*,*radius*[,*heading*] [ **!** **|** @ ]

| | |
|---|---|
| W | Warp command |
| *angle* | Range -360 to [+]360. Angle of the warp.<br>-ve moves in a CCW direction and +ve in a CW direction. |
| *radius* | Radius of the warp in working units |
| *heading* | Range -360 to [+]360 sets the heading relative to the current heading. |
| **!** (option 1) | *optional* line converter. Cause a line to be drawn to this point. Any sequence of jumps and warps will be converted into a cutting line from the last cut or anchor point. |
| @ (option 2) | *optional* anchor this point. Excludes this warp and any jumps or warps before it from being converted to a line on a subsequent jump or warp option 1. |

**Usage**

Warp is used for making arc type movements above the cutting plane and translate to G00 moves in a straight line. The blue line in the graphic below is the arc created by the warp and the red line is the path the tool would take in positioning to the end of the warp move.

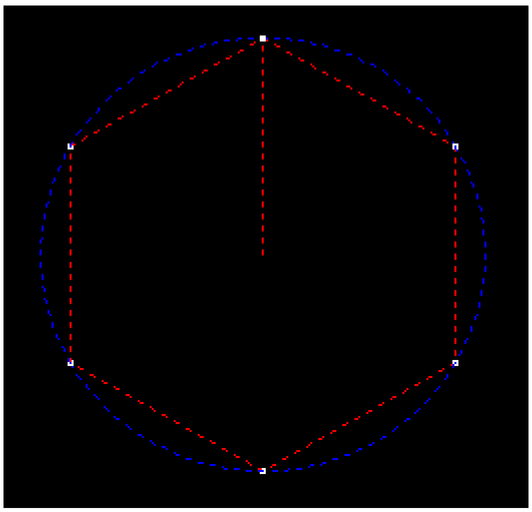


```
w90,1
```

**Example**

Warp is most useful for rotating a shape about a point. Creating bolt circle uses warp in the following way.

1. A Jump 97, for the distance of the radius, from the center of the circle to the circumference and setting the heading in the direction of rotation.
2. A Warp 101 along the circumference by the number of degrees equal to *360 / number of holes* or in this example 60 with a radius the same as the initial jump from the center
3. Draw 97 line of zero length which makes a point. When defining a path, even for drilling operations, material removal is only possible for segments of a shape created with the Draw 97, Arc 100 and the ConVex 117, ConCave 118 commands.
4. Repeat steps 2 & 3 a total of the number of holes.



```
j1,90
w60,1d0
w60,1d0
w60,1d0
w60,1d0
w60,1d0
w60,1d0
```

or

```
j1,90
```

```
r{
w60,1
d0
}6
```

## 5.9 B - Block

**Definition**

B *block number*{ *commands* }[[,]*occurs*]

| | |
|---|---|
| B | Block command |
| *block number* | A number ranging from 0 to 9. The block number must be unique between both N and B type blocks and can only be used once across all shapes in a program. |
| *commands* | Any commands in combination. |
| *occurs* | Default 1. The number of times the block of commands enclosed in braces will be repeated. The occurs is ignored when the block number is referenced by the repeat command. |

**Usage**

The block and procedure are identical with one minor difference. The block (B) has a default occurrence of 1 while the procedure (N) has a default occurrence of 0. This allows the procedure to define a shape without that shape being represented in the shape until referenced by a repeat (R) command.

The Block 103 command along with its occurs option is used to enclose a series of commands and have those commands used in the shape. Repeating a Block or Procedure 105 is the function of the Repeat 106 command.

The following are all equivalent:

```
b1{d1,90}

n1{d1,90}
r1

n1{d1,90}1

b1{d1,90}0
r1

r{d1,90}
```

In the last example above, the repeat uses the inline block specification.

The difference to using a numbered block vs an inline block, is that the numbered block can be repeated later in the same shape definition with a reference to it by the Repeat command.

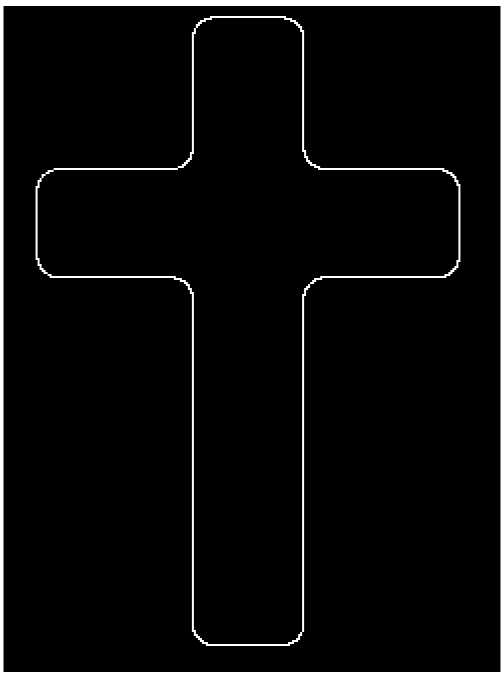
The block number is referenced by the Repeat 106 command to repeat the commands enclosed by the { }. The Repeat 106 may optionally specify a number of occurrences the commands are to be repeated.

Repeat block one, four times would be written:

r1,4

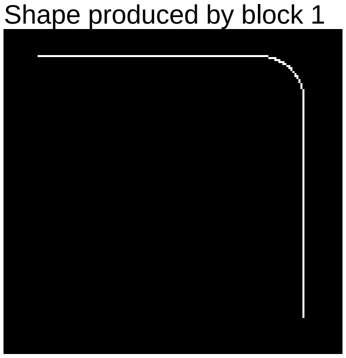
**Example**

Defining a few shape elements in blocks and then repeating the blocks in an inline block of the repeat command can be quite useful. The blocks are referenceable anywhere in the code and is how the long arm of the cross is made.

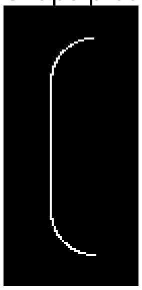


```
r{
b1{d.5 a-90,.1 d.5}
b2{a90,.1 d.3 a90,.1}
}3
r1d1r2d1
```

Shape produced by block 1



Shape produced by block 2



## 5.10 N - Procedure

**Definition**

N *block number*{ *commands* }[[,]*occurs*]

| | |
|---|---|
| N | Procedure command |
| *block number* | A number ranging from 0 to 9. The block number must be unique between both N and B type blocks and can only be used once across all shapes in a program. |
| *commands* | Any commands in combination. |
| *occurs* | Default 0<br>The number of times the block of commands enclosed in braces will be repeated. The occurs is ignored when the block number is referenced by the repeat command. |

**Usage**

The block and procedure are identical with one minor difference. The block (B) has a default occurrence of 1 while the procedure (N) has a default occurrence of 0. This allows the procedure to define a shape without that shape being represented in the shape until referenced by a repeat (R) command.

The Block 103 command along with its occurs option is used to enclose a series of commands and have those commands used in the shape. Repeating a Block or Procedure 105 is the function of the Repeat 106 command.

The following are all equivalent:

```
b1{d1,90}

n1{d1,90}
r1

n1{d1,90}1

b1{d1,90}0
r1

r{d1,90}
```

In the last example above, the repeat uses the inline block specification.

The difference to using a numbered block vs an inline block, is that the numbered block can be repeated later in the same shape definition with a reference to it by the Repeat command.

The block number is referenced by the Repeat 106 command to repeat the commands enclosed by the { }. The Repeat 106 may optionally specify a number of occurrences the commands are to be repeated.

Repeat block one, four times would be written:

r1,4

## 5.11 R - Repeat

**Definition**

R *block number* | { *commands* }[[,]*occurs*]

| | |
|---|---|
| R | Repeat command |
| *block number* | A number ranging from 0 to 9. The block number must be defined in the code with either a N or B type blocks specification.<br>Mutually exclusive with inline block of commands. |
| *commands* | Any commands in combination.<br>Mutually exclusive with blackener. |
| *occurs* | Default 1.<br>The number of times the referenced block number or the commands enclosed in braces will be repeated. |

**Usage**

The block and procedure are identical with one minor difference. The block (B) has a default occurrence of 1 while the procedure (N) has a default occurrence of 0. This allows the procedure to define a shape without that shape being represented in the shape until referenced by a repeat (R) command.

The Block 103 command along with its occurs option is used to enclose a  series of commands and have those commands used in the shape. Repeating a Block or Procedure 105 is the function of the Repeat 106 command.

The following are all equivalent:

```
b1{d1,90}

n1{d1,90}
r1

n1{d1,90}1

b1{d1,90}0
r1
```

r{d1,90}

In the last example above, the repeat uses the inline block specification.

The difference to using a numbered block vs an inline block, is that the numbered block can be repeated later in the same shape definition with a reference to it by the Repeat command.

The block number is referenced by the Repeat106 command to repeat the commands enclosed by the { }. The Repeat106 may optionally specify a number of occurrences the commands are to be repeated.

Repeat block one, four times would be written:

r1,4

## 5.12 T- Tangent

**Definition**

*T [-]radius1,[-]radius2,heading,distance[?]*

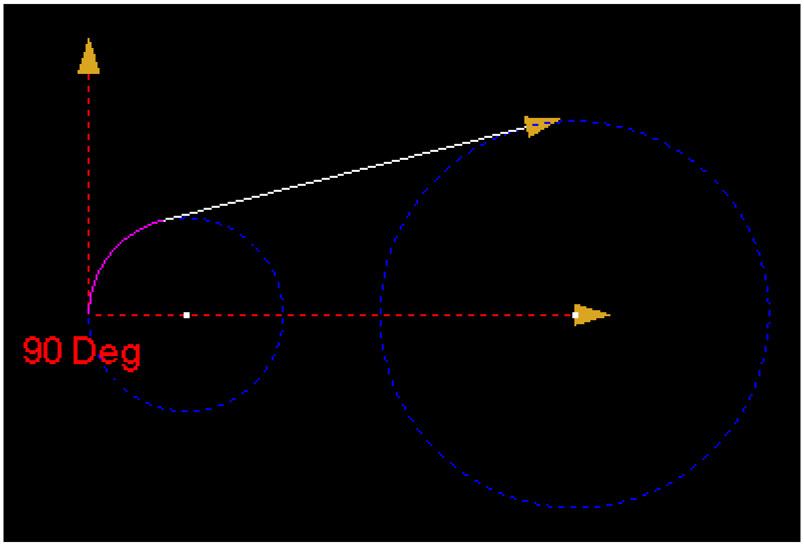
| | |
|---|---|
| T | tangent command |
| *[-]radius1* | Radius of the first circle in working units. A positive value for CW and a negative for CCW |
| *[-]radius2* | Radius of the second circle in working units. A positive value for outside tangent and a negative for inside tangent. |
| *heading* | Heading in degrees relative to the current heading indicator of the vector connecting the center of the two circles. |
| *distance* | Distance between the center of the two circles. |
| *?* | Complete the arc to the next quadrant crossing from the tangent point along the arc of the second circle. |

**Usage**

Understanding the heading can be a bit tricky. Remember the heading is the angle formed between initial heading and a line connecting the circle centers.

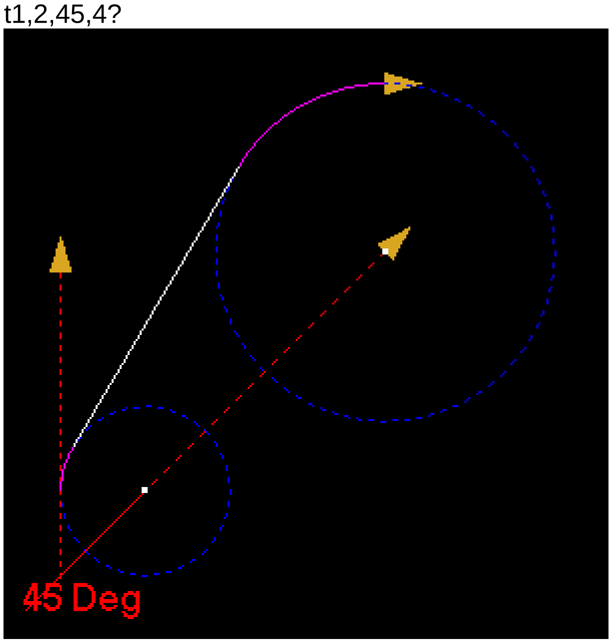
A 90 degree angle is formed between two imaginary lines. The first extending the initial heading and the second passing through the two circle centers.

t1,2,90,4



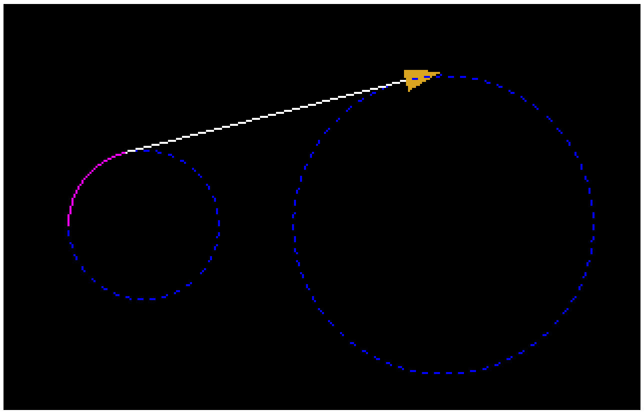

In this example a 45 degree angle is formed. The quadrant completion (?) has been set to extend the tangent line around the target arc to the next quadrant boundary.

t1,2,45,4?

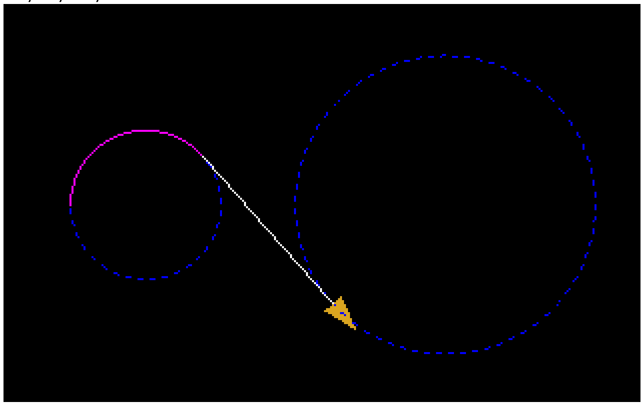



There are four combinations of clockwise/counterclockwise and inside/outside as follows:

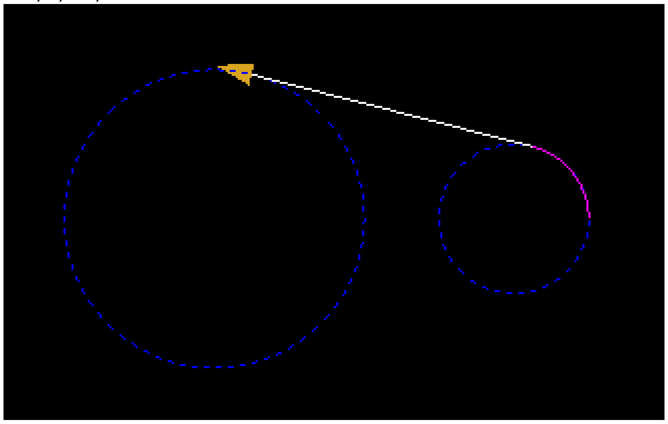
CW/outside
t1,2,90,4



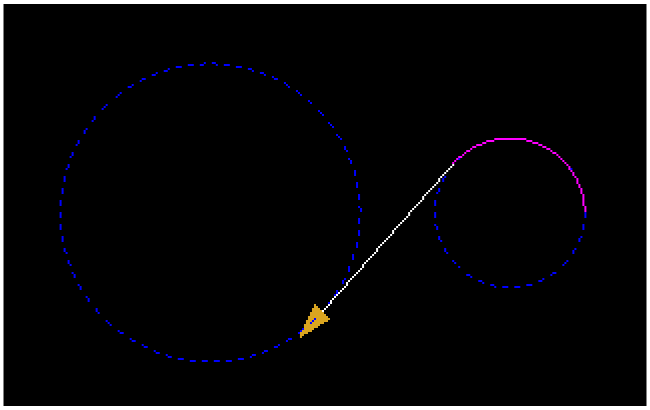
CW/inside
t1,-2,90,4



CCW/outside
t-1,2,90,4

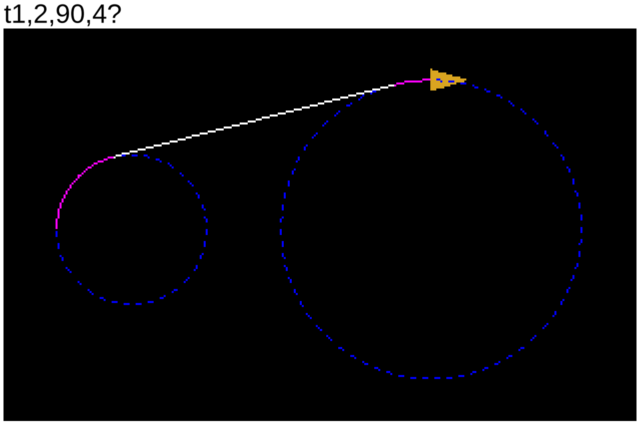


CCW/inside
t-1,-2,90,4

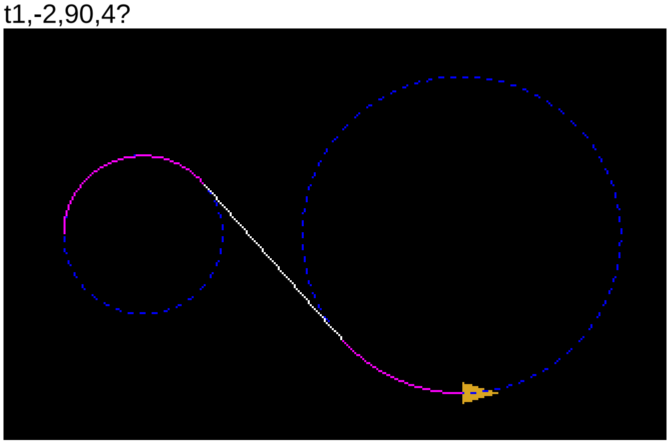


The same examples with quadrant completion (?) set:

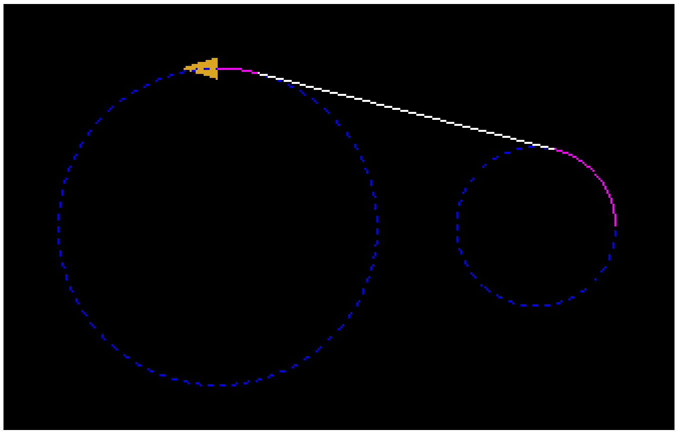
CW/outside
t1,2,90,4?



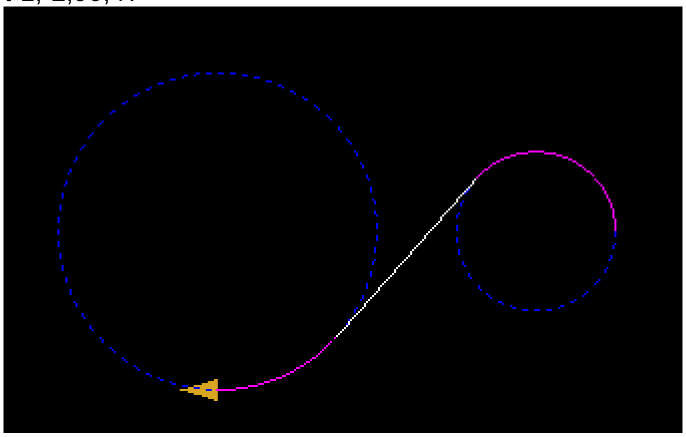
CW/inside
t1,-2,90,4?



CCW/outside
t-1,2,90,4?

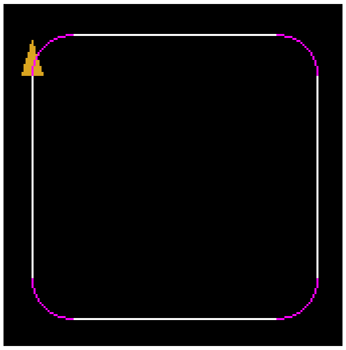


CCW/inside
t-1,-2,90,4?

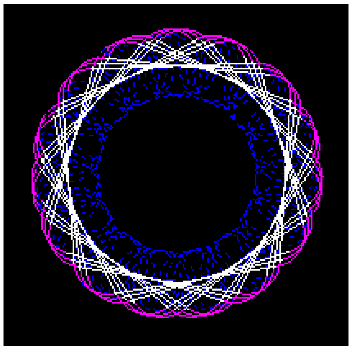


**Example 1**

The tangent command is perhaps the most powerful D2nc command. It opens up the possibility of countless shapes not possible before in the previous versions.



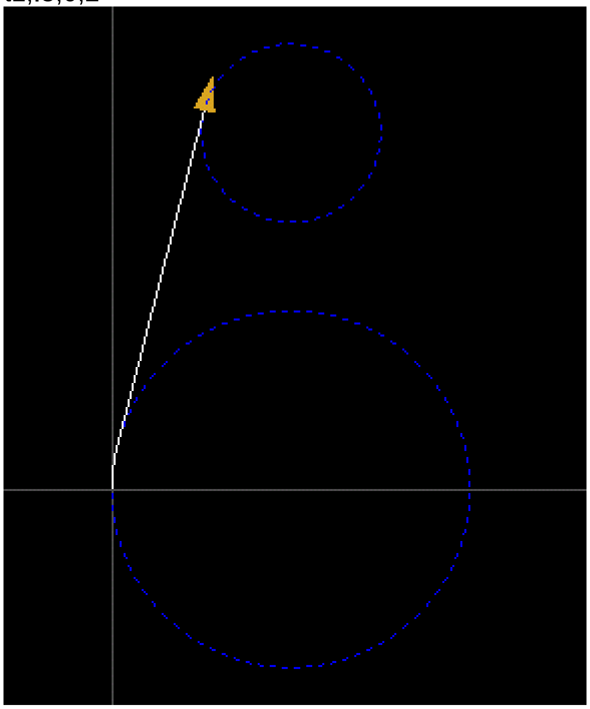
r{t.2,.2,90,1}4&



r{t.2,.1,90,1}45

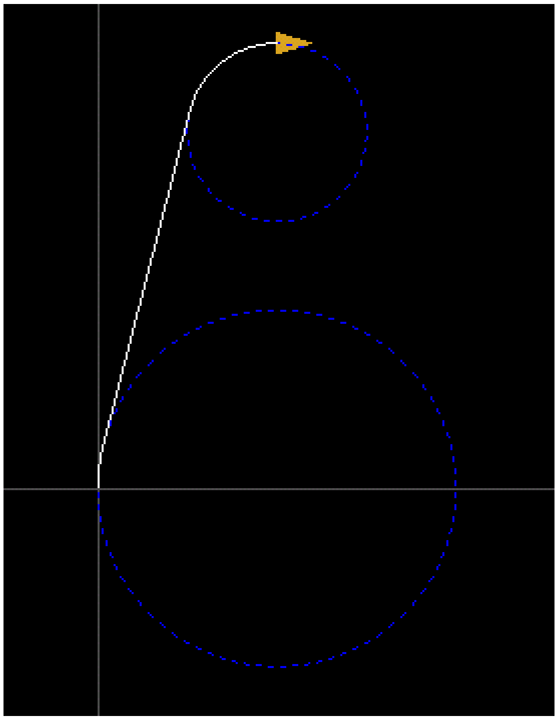
In this example, a bracket is formed by two tangent lines on two circles with a box shape below. The bottom circle is 2 inches in diameter and the top 1 inch. The center of the circles are 2 inches apart. The heading of the line connecting the circle centers is on the same heading as the initial heading so they are zero degrees apart. A detailed explanation of the heading can be found in the tangent reference section 107.

t1,.5,0,2



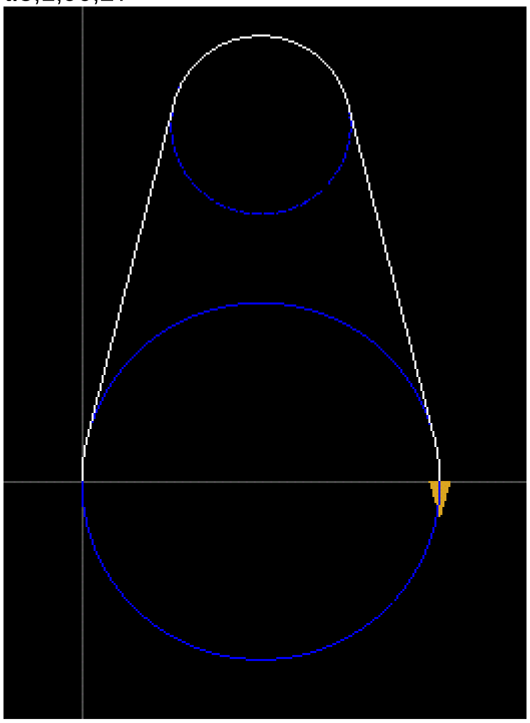
The shape created is the initial part of the arc to connect to the tangent line and the tangent line itself. This leaves the heading at some value we can get by looking at the Shape detail display which shows 75.52 degrees. We could draw an arc of 75.52 degrees with a radius of .5 (a75.52,.5) or we could just specify the quadrant completion (?) setting of the tangent command. The quadrant completion will extend the tangent line around the second circle until a quadrant boundary is reached.

t1,.5,0,2?



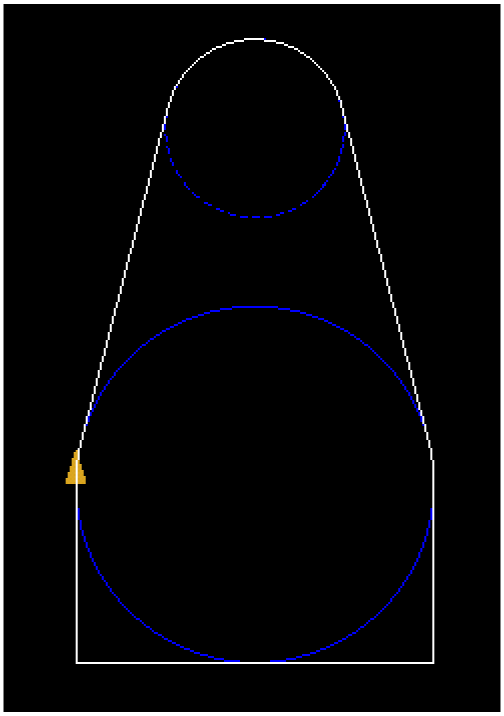
Now we can specify another tangent this time from the small circle to the large. Notice this time the heading will be 90 degrees between the current heading and circle centers. We also need to extend to the next quadrant. The program so far will now look line this:

t1,.5,0,2?
t.5,1,90,2?



The shape can now be finished off with the bottom box shape

t1,.5,0,2?
t.5,1,90,2?
d1,90
d2,90
&

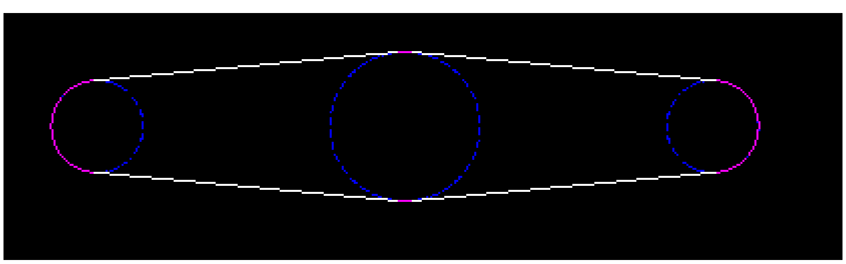


There you have the complete bracket outline ready to define a path and machine.

**Example 2**

The rocker arm illustrates the use of the tangent command and is a simple shape made up of four tangents. This shape comes from a live project and is the reason behind the creation of the tangent command. While the shape could have been define previously it required some math to calculate the arc angles and line distances. Too much to do in the machine shop and a simple command was needed.

The rocker arm is three circles spaced .67 apart the outer two circle have a radius of .1 and the center circle a radius of .1625. Given those dimensions, the following four tangent commands create the shape.



```
t.1,.1625,90,.670?
t.1625,.1,0,.670?
t.1,.1625,90,.670?
t.1625,.1,0,.670?
```

or

```
r{
t.1,.1625,90,.670?
t.1625,.1,0,.670?
}2
```

## 5.13 M - Move

**Definition**

M Xcoord,Ycoord

| | |
|---|---|
| M | Move command |
| *Xcoord* | X axis absolute coordinate for destination of the move |
| *Ycoord* | Y axis absolute coordinate for destination of the move |

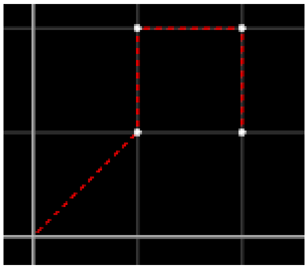
**Usage**

The M move command is coordinate based. Using this command can reduce the flexibility of SDL as the shape cannot be rotated. This could limit its use in repeated blocks and procedures.

Move is from the current position to the X and Y location specified in the move.

**Example**

Move to four specific X,Y coordinates and drill a hole there.



```
m1,1
d0
m1,2
d0
m2,2
d0
m2,1
d0
```

## 5.14 E - Etch

**Definition**

E Xcoord,Ycoord

| | |
|---|---|
| E | Etch command |
| *Xcoord* | X axis coordinate for destination of the move |
| *Ycoord* | Y axis coordinate for destination of the move |

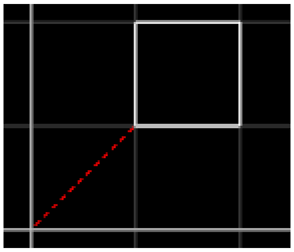
**Usage**

The E etch command is coordinate based. Using this command can reduce the flexibility of SDL as the shape cannot be rotated. This could limit its use in repeated blocks and procedures.

Etch is from the current position to the X and Y location specified in the etch.

**Example**

Move to a start coordinate 1,1 and etch a 1 inch square.



```
m1,1
e1,2
e2,2
e2,1
e1,1
```

## 5.15 I - Library

**Definition**

I "libraryitem"[[,]scale]

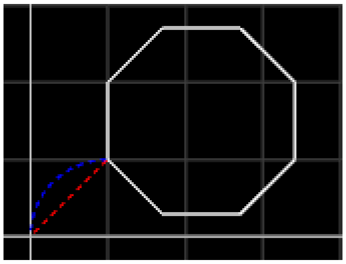
| | |
|---|---|
| I | Library command (letter i) |
| libraryitem | Name of a shape contained in the Shape Library. |
| scale | A scaling factor to reduce or increase the shape by. A factor less than 1 will reduce the size and larger than 1 will increase the size. The default value is 1 if left unspecified. |

**Usage**

Used to insert predefined shapes contained in the shape library 26 into a shape.

**Example**

Insert library shape "oct" after a positioning warp move



```
w90,1,-90
i"oct"
```

## 5.16 V - ConVex

**Definition**

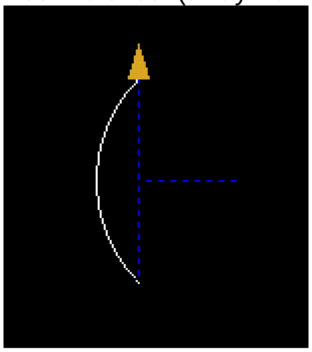
V *distance*,+-*offset*[,*heading*]

| | |
|---|---|
| V | conVex command |
| *distance* | Distance in working units to move in the current heading direction to set the end point of the convex arc. |
| *offset* | Chord perpendicular to and at the mid point of the distance set in the first parameter. Endpoint of the cord is the center of the arc. A positive offset sets the chord away from the arc and negative towards the arc. |
| *heading* | Range -360 to [+]360 sets the heading relative to the current heading. |

**Usage**

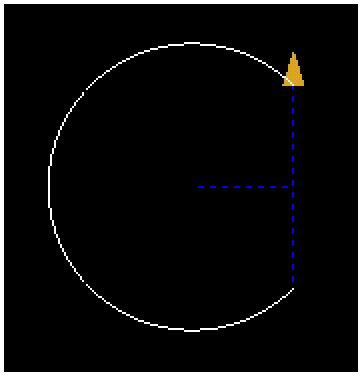
Convex allows a linear move with an arc connecting the start and end points. The offset can be either side of the linear move controlled by a negative or positive offset.

Positive offset (away from arc)



```
v1,.5
```

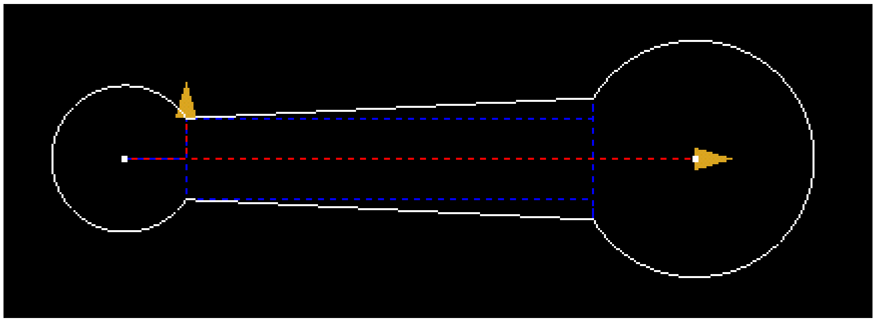
Negative offset (towards arc)



```
v1,-.5
```

**Example**

This con rod outline is an example of using the convex shape.



For the outline of the con rod enter the following in an empty shape:

```
h90
j.3,-90
j.2,90@
j2,-90
j.1,180!
v.6,-.5,180
j.1,-90
j2,90!
v.4,-.3
```

To define points to drill the centers select a different shape and enter:

```
d0,90
j2.8
d0
```

## 5.17 C - ConCave

**Definition**

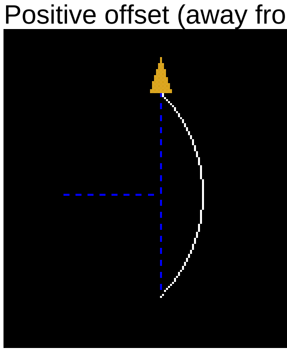
C *distance*,+-*offset*[,*heading*]

| | |
|---|---|
| C | conCave command |
| *distance* | Distance in working units to move in the current heading direction to set the end point of the concave arc. |
| *offset* | Chord perpendicular to and at the mid point of the distance set in the first parameter. Endpoint of the cord is the center of the arc. A positive offset sets the chord away from the arc and negative towards the arc. |
| *heading* | Range -360 to [+]360 sets the heading relative to the current heading. |

**Usage**

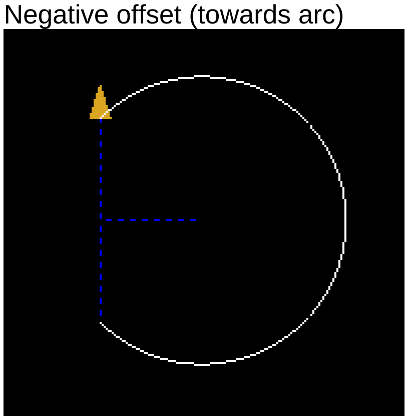
Concave allows a linear move with an arc connecting the start and end points. The offset can be either side of the linear move controlled by a negative or positive offset.

Positive offset (away from arc)



`c1,.5`

Negative offset (towards arc)



`c1,-.5`

**Example**

ConCave is the same as ConVex except is used for going in the reverse direction.
See example from ConVex 117

## 5.18 & - Close

**Definition**

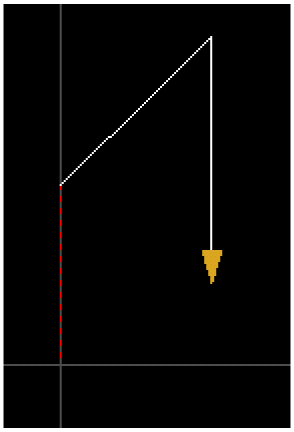
&

| | |
|---|---|
| & | Close command |

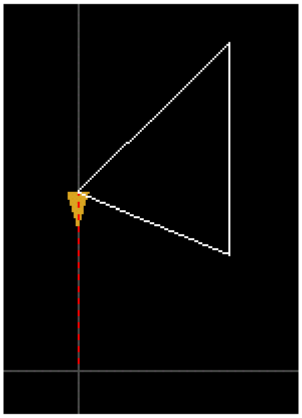
**Usage**

Draw to close will close the currently open shape with a Draw 97 move to the start of the first segment of the current shape. If no shape is currently open it will draw back to itself effectively creating a point. The heading 95 remains unchanged by the & command.

**Example**



```
j1,45
d1.2,135
d1.2
```



```
j1,45
d1.2,135
d1.2
&
```

## 5.19 % - Reflect

**Definition**

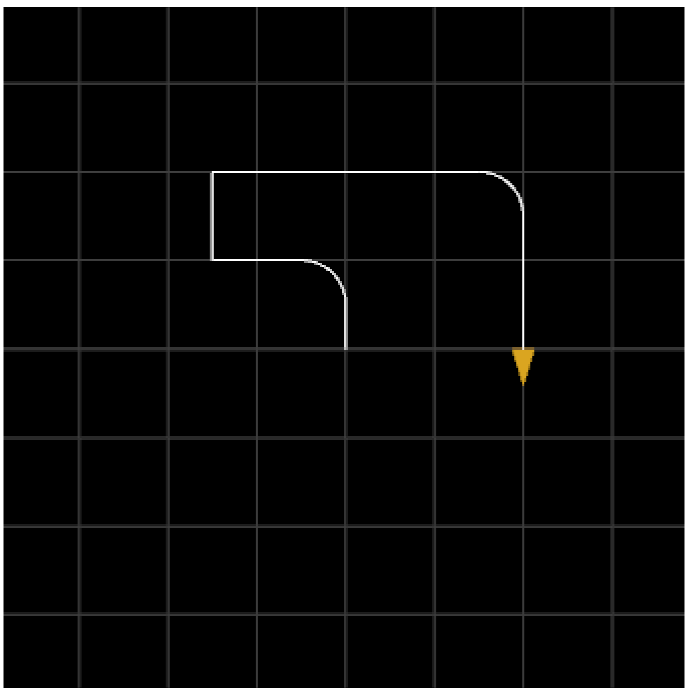
%

| | |
|---|---|
| % | Reflect command |

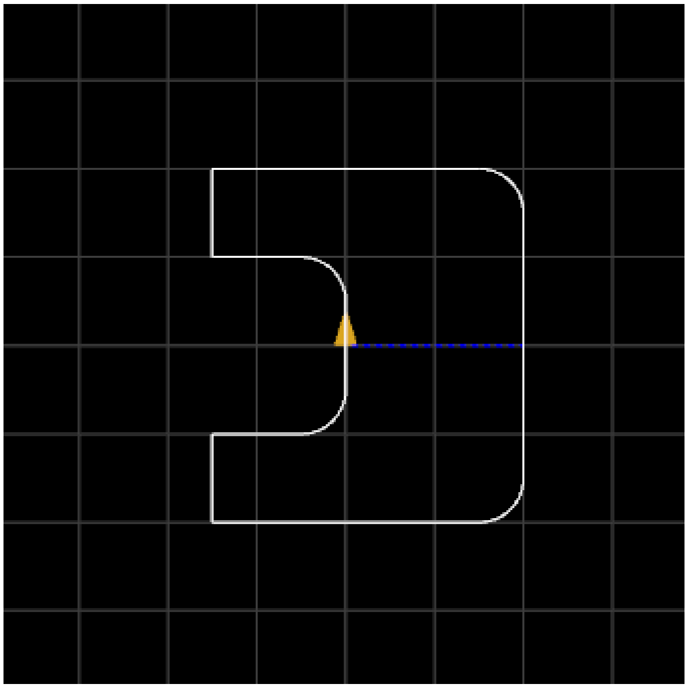
**Usage**

Reflect will close the currently open shape with a reflection of the shape across and imaginary axis from the current point to the first point in the current contiguous line. The imaginary axis will be shown as a construction line in the shape display area. Useful for describing any symmetrical irregular shape.

**Example**



```
d.5
a-90,.5
d1,90
d1,90
d3
a90,.5
d1.5
```



```
d.5
a-90,.5
d1,90
d1,90
d3
a90,.5
d1.5
%
```