The gcodepreview OpenSCAD library*

Author: William F. Adams willadams at aol dot com

2024/04/12

Abstract

The gcode preview library allows using OpenSCAD to move a tool in lines and output dxf and G-code files so as to work as a CAD/CAM program for $\rm CNC$

Contents

1	read	dme.md	
2	gco	depreview	
	$\overline{2.1}$	Position and Variables	
	2.2	Tools and Changes	
	2.3	File Handling	1
	2.4	Movement and Cutting	4
3	gco	depreview_template.scad	2

1 readme.md

 $16 \langle rdm \rangle$

1 (rdm)	> # gcodepreview
2 (rdm)	
3 (rdm)	OpenSCAD library for moving a tool in lines and arcs so as to model how a part would be cut
4 (rdm)	using G-Code, so as to allow OpenSCAD to function as a compleat CAD/CAM solution for
5 (rdm)	subtractive CNC (mills and routers).
6 (rdm)	
7 (rdm)	![OpenSCAD Cut Joinery Module](https://raw.githubusercontent.com/WillAdams/gcodepreview/mai
8 (rdm)	
9 (rdm)	Updated to make use of Python in OpenSCAD:
10 (rdm)	
11 (rdm)	http://www.guenther-sohler.net/openscad/
12 (rdm)	
13 (rdm)	(previous versions had used RapCAD)
14 (rdm)	
15 (rdm)	A BlockSCAD file for the main modules is available at:

^{*}This file (gcodepreview.dtx) has version number v0.2, last revised 2024/04/12.

2 1 readme.md

```
17 (rdm) https://www.blockscad3d.com/community/projects/1244473
18 (rdm)
19 \langle rdm \rangle The project is discussed at:
20 (rdm)
{\tt 21}~{\tt (rdm)}~{\tt https://forum.makerforums.info/t/g-code-preview-using-openscad-rapcad/85729}
22 (rdm)
23 (rdm) and
24 (rdm)
25 (rdm) https://forum.makerforums.info/t/openscad-and-python-looking-to-finally-be-resolv
26 (rdm)
27 (rdm) and
28 \langle rdm \rangle
29 (rdm) https://willadams.gitbook.io/design-into-3d/programming
30 (rdm)
31 (rdm) Usage is:
32 (rdm)
33 (rdm) Place the file in C:\Users\\~\Documents\OpenSCAD\libraries (C:\Users\\\~\Documents\OpenSCAD\libraries)
34 \langle \mathsf{rdm} 
angle deprecated since RapCAD is not longer needed since Python is now used for writing
35 (rdm)
36 \left< \mathsf{rdm} \right> (While it was updated for use w/ RapCAD, so as to take advantage of the writeln of
37 \langle rdm \rangle it was possible to write that in Python)
38 (rdm)
39 (rdm)
              use <gcodepreview.py>;
40 (rdm)
              use <pygcodepreview.scad>;
41 (rdm)
              include <gcodepreview.scad>;
42 (rdm)
43 (rdm) Note that it is necessary to use the first two files (this allows loading
44 \langle \mathsf{rdm} \rangle the Python commands and then wrapping them in OpenSCAD commands) and then
_{45} \langle \mathsf{rdm} \rangle include the last file (which allows using OpenSCAD variables to selectively
46 \langle \mathsf{rdm} \rangle implement the Python commands via their being wrapped in OpenSCAD modules)
47 (rdm)
48 \langle \mathsf{rdm} \rangle and define variables which match the project and then use commands such as:
49 (rdm)
50 (rdm)
              opengcodefile(Gcode_filename);
51 (rdm)
              opendxffile(DXF_filename);
52 (rdm)
53 (rdm)
              difference() {
54 (rdm)
                  setupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorig
55 (rdm)
56 (rdm)
              movetosafez();
57 (rdm)
58 (rdm)
              toolchange(squaretoolno,speed * square_ratio);
59 (rdm)
              begintoolpath(0,0,0.25);
60 (rdm)
              beginpolyline(0,0,0.25);
61 (rdm)
62 (rdm)
              cutoneaxis_setfeed("Z",-1,plunge*square_ratio);
63 (rdm)
              addpolyline(stocklength/2,stockwidth/2,-stockthickness);
64 (rdm)
65 (rdm)
66 (rdm)
              cutwithfeed(stocklength/2,stockwidth/2,-stockthickness,feed);
67 (rdm)
68 (rdm)
              endtoolpath();
              endpolyline();
69 (rdm)
70 (rdm)
```

```
71 (rdm)
72 (rdm)
73 (rdm)
             closegcodefile();
74 (rdm)
             closedxffile();
75 (rdm)
76 (rdm) Tool numbers match those of tooling sold by Carbide 3D (ob. discl., I work for them).
77 (rdm) Comments are included in the G-code to match those expected by CutViewer.
78 (rdm)
79 \langle \mathsf{rdm} \rangle A complete example file is: gcodepreview_template.scad another is
80 (rdm) openscad_gcodepreview_cutjoinery.tres.scad which is made from an
81 (rdm) OpenSCAD Graph Editor file:
82 (rdm)
        ![OpenSCAD Graph Editor Cut Joinery File](https://raw.githubusercontent.com/WillAdams/gcode
83 (rdm)
84 (rdm)
85 (rdm) Version 0.1 supports setting up stock, origin, rapid positioning, making cuts,
86 (rdm) and writing out matching G-code, and creating a DXF with polylines.
87 (rdm)
88 (rdm) Added features since initial upload:
89 (rdm)
90 (rdm)
          - endpolyline(); --- this command allows ending one polyline so as to allow multiple lines
          - separate dxf files are written out for each tool where tool is ball/square/V and small/l
91 (rdm)
          - re-writing as a Literate Program using the LaTeX package docmfp (begun 4/12/24)
92 (rdm)
93 (rdm)
94 (rdm) Not quite working feature:
95 (rdm)
96 (rdm)
          - exporting SVGs --- these are written out upside down due to coordinate differences between
97 (rdm)
98 (rdm) Possible future improvements:
99 (rdm)
100 (rdm)
         - G-code: support for G2/G3 arcs
101 (rdm) - DXF support for curves and the 3rd dimension
102 \, \langle \mathsf{rdm} 
angle - G-code: import external tool libraries and feeds and speeds from JSON or CSV files --- n
103 (rdm)
          - support for additional tooling shapes such as dovetail tools, or roundover tooling
          - general coding improvements --- current coding style is quite prosaic
104 (rdm)
          - generalized modules for cutting out various shapes/geometries --- a current one is to cu
105 (rdm)
106 %
```

As noted above, this library works by using Python code as a back-end so as to persistently store and access variables, and to write out files. Doing so requires a total of three files:

- A Python file: gcodepreview.py (gcpy)
- An OpenSCAD file: gcodepreview.scad (gcpscad)
- An OpenSCAD file which connects the other two files: pygcodepreview.scad (pyscad)

Each file will begin with a suitable comment indicating the file type:

```
107 \; \langle \text{gcpy} \rangle #!/usr/bin/env python 108 \; \langle \text{gcpy} \rangle
```

```
109 \( \text{pyscad} \) //!OpenSCAD

110 \( \text{pyscad} \)

111 \( \text{gcpscad} \) //!OpenSCAD

112 \( \text{gcpscad} \)

113 \( \text{gcpscad} \) //gcodepreview 0.1

114 \( \text{gcpscad} \) //

115 \( \text{gcpscad} \) //used via use \( \text{gcodepreview.py}; \)

116 \( \text{gcpscad} \) // use \( \text{pygcodepreview.scad}; \)

117 \( \text{gcpscad} \) // include \( \text{gcodepreview.scad}; \)

118 \( \text{gcpscad} \) //
```

writeln The original implementation in RapSCAD used a command writeln — fortunately, this command is easily re-created in Python:

```
119 \langle gcpy \rangle def writeln(*arguments):
120 \langle gcpy \rangle line_to_write = ""
121 \langle gcpy \rangle for element in arguments:
122 \langle gcpy \rangle line_to_write += element
123 \langle gcpy \rangle f.write(line_to_write)
124 \langle gcpy \rangle f.write("\n")
```

which command will accept a series of arguments and then write them out to a file object.

2.1 Position and Variables

In modeling the machine motion and G-code it will be necessary to have the machine track several variables. This will be done using paired functions (which will return the matching variable) and a matching (global) variable, as well as additional functions for setting the matching variable.

The first such variables are for XYZ position:

```
    mpx
    mpx
    mpy
    mpz
    mpz
```

It will further be necessary to have a variable for the current tool:

currenttool • currenttool

For each command it will be necessary to implement an appropriate aspect in each file. The Python file wil manage the Python variables and handle things which can only be done in Python, while there will be two OpenSCAD files as noted above, one which calls the Python code (this will be used), while the other will be included and will be able to access and use OpenSCAD variables, as well as implement Customizer options.

The first such routine will be appropriately enough, to set up the stock, and psetupstock perform other initializations.

```
125 \(\sqrt{gcpy}\) def psetupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin

126 \(\sqrt{gcpy}\) global mpx

127 \(\sqrt{gcpy}\) mpx = float(0)

128 \(\sqrt{gcpy}\) global mpy
```

```
mpy = float(0)
              129 (gcpy)
                             global mpz
              130 (gcpy)
                             mpz = float(0)
              131 (gcpy)
                             global currenttool
              132 \langle \mathsf{gcpy} \rangle
              133 (gcpy)
                             currenttool = 102
              134 (gcpy)
osetupstock
              135 (pyscad) module osetupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin) {
                               psetupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin);
              136 (pyscad)
              137 (pyscad) }
              138 (pyscad)
 setupstock
              139 (gcpscad) module setupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin) {
              140 (gcpscad)
                              osetupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin);
              141 (gcpscad) //initialize default tool and XYZ origin
              142 (gcpscad)
                              osettool(102);
                              oset(0,0,0);
              143 (gcpscad)
                              if (zeroheight == "Top") {
              144 (gcpscad)
                                if (stockorigin == "Lower-Left") {
              145 (gcpscad)
                                translate([0, 0, (-stockthickness)]){
              146 (gcpscad)
                                cube([stocklength, stockwidth, stockthickness], center=false);
              147 (gcpscad)
              148 \langle gcpscad \rangle if (generategcode == true) {
              149 (gcpscad) // owriteone("(setupstock)");
              150 (gcpscad) owritethree("(stockMin:0.00mm, 0.00mm, -",str(stockthickness),"mm)");
              151 \(\rangle gcpscad \rangle \) owritefive("(stockMax:",str(stocklength),"mm, ",str(stockwidth),"mm, 0.00mm)");
              152 (gcpscad)
                                owritenine("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(stockthic
              153 (gcpscad) }
              154 (gcpscad) }
              _{155} \langle \text{gcpscad} \rangle }
                                 else if (stockorigin == "Center-Left") {
              156 \; \langle \mathsf{gcpscad} \rangle
                                translate([0, (-stockwidth / 2), -stockthickness]){
              157 (gcpscad)
              158 \langle \mathsf{gcpscad} \rangle
                                  cube([stocklength, stockwidth, stockthickness], center=false);
              159 (gcpscad)
                                if (generategcode == true) {
              160 (gcpscad) // owriteone("(setupstock)");
              161 \( gcpscad \) owritefive("(stockMin:0.00mm, -",str(stockwidth/2), "mm, -",str(stockthickness), "mm)");
              162 \(\rangle gcpscad \rangle \) owritefive("(stockMax:",str(stocklength),"mm, ",str(stockwidth/2),"mm, 0.00mm)");
                                owriteeleven("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(stockth
              163 (gcpscad)
              164 (gcpscad)
              165 (gcpscad)
              166 (gcpscad)
                                } else if (stockorigin == "Top-Left") {
                                translate([0, (-stockwidth), -stockthickness]){
              167 (gcpscad)
                                  cube([stocklength, stockwidth, stockthickness], center=false);
              168 (gcpscad)
              169 (gcpscad) if (generategcode == true) {
              170 (gcpscad) // owriteone("(setupstock)");
              171 \(\rangle gcpscad \rangle \) owritefive("(stockMin:0.00mm, -",str(stockwidth),"mm, -",str(stockthickness),"mm)");
              172 (gcpscad) owritethree("(stockMax:",str(stocklength),"mm, 0.00mm, 0.00mm)");
              173 (gcpscad) owriteeleven("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(stockthickn
              174 (gcpscad)
                                }
                                }
              175 (gcpscad)
                                }
              176 (gcpscad)
              177 \( gcpscad \) else if (stockorigin == "Center") {
              178 (gcpscad) //owritecomment("Center");
              179 (gcpscad)
                                translate([(-stocklength / 2), (-stockwidth / 2), -stockthickness]){
```

```
6
```

```
cube([stocklength, stockwidth, stockthickness], center=false);
 180 (gcpscad)
 181 \langle gcpscad \rangle if (generategcode == true) {
 182 \( gcpscad \rangle // owriteone("(setupstock)");
 183 (gcpscad) owriteseven("(stockMin: -",str(stocklength/2),", -",str(stockwidth/2),"mm, -"
 184 (gcpscad) owritefive("(stockMax:",str(stocklength/2),"mm, ",str(stockwidth/2),"mm, 0.00
 185 (gcpscad) owritethirteen("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",st
 186 (gcpscad) }
 187 (gcpscad) }
 188 (gcpscad) }
 189 (gcpscad) } else if (zeroheight == "Bottom") {
 190 (gcpscad) //owritecomment("Bottom");
                                                    if (stockorigin == "Lower-Left") {
 191 (gcpscad)
 192 (gcpscad)
                                                    cube([stocklength, stockwidth, stockthickness], center=false);
 193 (gcpscad) if (generategcode == true) {
 194 (gcpscad) // owriteone("(setupstock)");
 195 (gcpscad) owriteone("(stockMin:0.00mm, 0.00mm, 0.00mm)");
 196 (gcpscad) owriteseven("(stockMax:",str(stocklength),"mm, ",str(stockwidth),"mm, ",str(s
 197 (gcpscad) owriteseven("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(s
 198 (gcpscad)
 199 (gcpscad) } else if (stockorigin == "Center-Left") {
                                                    translate([0, (-stockwidth / 2), 0]){
200 (gcpscad)
                                                          cube([stocklength, stockwidth, stockthickness], center=false);
201 (gcpscad)
202 (gcpscad) if (generategcode == true) {
203 (gcpscad) // owriteone("(setupstock)");
204 \(\rangle gcpscad \rangle \) owritethree("(stockMin:0.00mm, -",str(stockwidth/2),"mm, 0.00mm)");
205 \(\rangle gcpscad \rangle \) owriteseven("(stockMax:",str(stocklength),"mm, ",str(stockwidth/2),"mm, ",str
206 (gcpscad) owritenine("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(st
207 (gcpscad)
                                                   }
208 (gcpscad)
                                                   }
209 (gcpscad) } else if (stockorigin == "Top-Left") {
210 (gcpscad)
                                                  translate([0, (-stockwidth), 0]){
                                                          cube([stocklength, stockwidth, stockthickness], center=false);
211 (gcpscad)
212 (gcpscad)
213~\langle \texttt{gcpscad} \rangle if (generategcode == true) {
214 \( gcpscad \) // owriteone("(setupstock)");
215 \(\rangle gcpscad \rangle \) owritethree("(stockMin:0.00mm, -",str(stockwidth),"mm, 0.00mm)");
216 \(gcpscad\) owritefive("(stockMax:",str(stocklength),"mm, 0.00mm, ",str(stockthickness),"
217 \(\rangle gcpscad \rangle \) owritenine("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(stocklength),", ",str(stockwidth),", ",str(stocklength),", ",str(sto
218 \langle \mathsf{gcpscad} \rangle }
219~\langle \texttt{gcpscad} \rangle } else if (stockorigin == "Center") {
                                                    translate([(-stocklength / 2), (-stockwidth / 2), 0]){
220 (gcpscad)
221 (gcpscad)
                                                          cube([stocklength, stockwidth, stockthickness], center=false);
222 (gcpscad)
223 (gcpscad) if (generategcode == true) {
224 (gcpscad) // owriteone("(setupstock)");
225 \ \langle \texttt{gcpscad} \rangle \ \texttt{owritefive("(stockMin:-",str(stocklength/2),", -",str(stockwidth/2),"mm, 0.000)} \rangle
226 \(\rangle gcpscad \rangle \) owriteseven("(stockMax:",str(stocklength/2),"mm, ",str(stockwidth/2),"mm, ",str(stockwid
227 \(\rangle gcpscad \rangle \) owriteeleven("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(stoc
228 (gcpscad) }
229 (gcpscad) }
230 (gcpscad) }
231 (gcpscad) if (generategcode == true) {
232 \langle gcpscad \rangle owriteone("G90");
233 (gcpscad) owriteone("G21");
```

```
234 (gcpscad) // owriteone("(Move to safe Z to avoid workholding)");
              235 (gcpscad) // owriteone("G53G0Z-5.000");
              236 (gcpscad) }
              237 (gcpscad) //owritecomment("ENDSETUP");
              238 (gcpscad) }
              239 (gcpscad)
                    It will be necessary to have Python functions which return the current values
      xpos
      ypos of the machine position in Cartesian coordinates:
              240 \langle \text{gcpy} \rangle def xpos():
              241 \langle \mathsf{gcpy} \rangle
                                  global mpx
              242 \langle \mathsf{gcpy} \rangle
                                 return mpx
              243 (gcpy)
              244 \langle \text{gcpy} \rangle def ypos():
              245 \langle \mathsf{gcpy} \rangle
                                  global mpy
              246 \langle \mathsf{gcpy} \rangle
                                 return mpy
              247 \langle \mathsf{gcpy} \rangle
              248 \langle \text{gcpy} \rangle def zpos():
              249 \langle \mathsf{gcpy} \rangle
                                 global mpz
              250 \langle \mathsf{gcpy} \rangle
                                 return mpz
              251 \langle \mathsf{gcpy} \rangle
psetxpos and in turn, functions which set the positions:
psetypos
              252 \langle \text{gcpy} \rangle def psetxpos(newxpos):
psetzpos
              253 (gcpy)
                                 global mpx
                                 mpx = newxpos
              254 \langle \mathsf{gcpy} \rangle
              255 \langle \mathsf{gcpy} \rangle
              256 \langle \text{gcpy} \rangle def psetypos(newypos):
                                 global mpy
              257 \langle \mathsf{gcpy} \rangle
                                 mpy = newypos
              258 \langle \mathsf{gcpy} \rangle
              259 \langle \mathsf{gcpy} \rangle
              260 \langle \text{gcpy} \rangle def psetzpos(newzpos):
              261 (gcpy)
                                 global mpz
              262 \langle gcpy \rangle
                                 mpz = newzpos
              263 (gcpy)
 getxpos and as noted above, there will need to be matching OpenSCAD versions:
 getypos
              264 (pyscad) function getxpos() = xpos();
 getzpos
              265 (pyscad) function getypos() = ypos();
 setxpos 266 \langle pyscad \rangle function getzpos() = zpos();
 setypos 267 (pyscad)
 setzpos 268 (pyscad) module setxpos(newxpos) {
              269 (pyscad) psetxpos(newxpos);
              270 (pyscad) }
              271 (pyscad)
              272 (pyscad) module setypos(newypos) {
              273 (pyscad) psetypos(newypos);
              274 (pyscad) }
              275 (pyscad)
              276 \langle pyscad \rangle module setzpos(newzpos) {
              277 \langle pyscad \rangle psetzpos(newzpos);
              278 (pyscad) }
              279 (pyscad)
      oset
```

```
280 \(\partial \text{gcpscad} \rangle \text{ module oset(ex, ey, ez) } \{ 281 \(\partial \text{gcpscad} \rangle \text{ setxpos(ex);} \)
282 \(\partial \text{gcpscad} \rangle \text{ setzpos(ey);} \)
283 \(\partial \text{gcpscad} \rangle \text{ setzpos(ez);} \)
284 \(\partial \text{gcpscad} \rangle \}
285 \(\partial \text{gcpscad} \rangle \)
```

2.2 Tools and Changes

Similarly Python functions and variables will be used to track and set and return psettool the current tool:

```
pcurrenttool
              286 (gcpy) def psettool(tn):
               287 (gcpy)
                             global currenttool
               288 (gcpy)
                             currenttool = tn
               289 (gcpy)
               290 (gcpy) def pcurrent_tool():
               291 (gcpy)
                             global currenttool
               292 (gcpy)
                             return currenttool
               293 (gcpy)
    osettool and matching OpenSCAD modules set and return the current tool:
 currenttool
              294 \langle pyscad \rangle module osettool(tn){
              295 (pyscad) psettool(tn);}
               296 (pyscad)
               297 (pyscad) function current_tool() = pcurrent_tool();
               298 (pyscad)
  toolchange and apply the appropriate commands for a toolchange.
               299 \langle gcpscad \rangle module toolchange(tool_number,speed) {
               300 (gcpscad)
                               osettool(tool_number);
               301 \langle gcpscad \rangle if (generategcode == true) {
               302 (gcpscad) writecomment("Toolpath");
               303 (gcpscad) owriteone("MO5");
               304 \langle \text{gcpscad} \rangle // writecomment("Move to safe Z to avoid workholding");
               305 (gcpscad) // owriteone("G53G0Z-5.000");
               306 (gcpscad) // writecomment("Begin toolpath");
               307 (gcpscad)
                            if (tool_number == 201) {
               308 (gcpscad) writecomment("TOOL/MILL, 6.35, 0.00, 0.00, 0.00");
               309 (gcpscad)
                             } else if (tool_number == 202) {
              310 (gcpscad) writecomment("TOOL/MILL, 6.35, 3.17, 0.00, 0.00");
              311 (gcpscad)
                             } else if (tool_number == 102) {
              312 (gcpscad) writecomment("TOOL/MILL, 3.17, 0.00, 0.00, 0.00");
              313 (gcpscad)
                             } else if (tool_number == 101) {
              314 (gcpscad) writecomment("TOOL/MILL, 3.17, 1.58, 0.00, 0.00");
                             } else if (tool_number == 301) {
              315 (gcpscad)
              316 (gcpscad) writecomment("TOOL/MILL, 0.03, 0.00, 6.35, 45.00");
              317 (gcpscad)
                             } else if (tool_number == 302) {
              318 (gcpscad) writecommment("TOOL/MILL, 0.03, 0.00, 10.998, 30.00");
                             } else if (tool_number == 390) {
              319 (gcpscad)
                            writecomment("TOOL/MILL,0.03, 0.00, 1.5875, 45.00");
              320 (gcpscad)
              321 (gcpscad)
              322 (gcpscad)
                                select_tool(tool_number);
               323 \(\rangle gcpscad\)\) owritetwo("M6T",str(tool_number));
               324 (gcpscad) owritetwo("MO3S",str(speed));
```

```
325 \langle \mathsf{gcpscad} \rangle \ \}
                      326 (gcpscad) }
                      327 (gcpscad)
                         There must also be a module for selecting tools: select_tool:
         selecttool
       tool_number
                      328 (gcpscad) module select_tool(tool_number) {
                      329 \(\rangle gcpscad \rangle \) //echo(tool_number);
                      330 (gcpscad)
                                       if (tool_number == 201) {
                                         gcp_endmill_square(6.35, 19.05);
                      331 (gcpscad)
                      332 (gcpscad)
                                       } else if (tool_number == 202) {
                                         gcp_endmill_ball(6.35, 19.05);
                      333 (gcpscad)
                                       } else if (tool_number == 102) {
                      334 (gcpscad)
                                         gcp_endmill_square(3.175, 19.05);
                      335 (gcpscad)
                      336 (gcpscad)
                                       } else if (tool_number == 101) {
                      337 (gcpscad)
                                         gcp_endmill_ball(3.175, 19.05);
                                       } else if (tool_number == 301) {
                      338 (gcpscad)
                                         gcp_endmill_v(90, 12.7);
                      339 (gcpscad)
                                       } else if (tool_number == 302) {
                      340 (gcpscad)
                      _{341}~\langle \mathsf{gcpscad} \rangle
                                         gcp_endmill_v(60, 12.7);
                      _{342}~\langle \mathsf{gcpscad} \rangle
                                       } else if (tool_number == 390) {
                      343 (gcpscad)
                                         gcp_endmill_v(90, 3.175);
                      344 (gcpscad)
                      345 (gcpscad) }
                      346 (gcpscad)
                         Each tool must be modeled in 3D using an OpenSCAD module:
gcp_endmill_square
                      347 \langle gcpscad \rangle module gcp_endmill_square(es_diameter, es_flute_length) {
                                       cylinder(r1=(es_diameter / 2), r2=(es_diameter / 2), h=es_flute_length, center=false);
                      348 (gcpscad)
                      349 (gcpscad) }
                      350 \langle gcpscad \rangle
  gcp_endmill_ball
                      351~\langle \texttt{gcpscad} \rangle module gcp_endmill_ball(es_diameter, es_flute_length) {
                                       translate([0, 0, (es_diameter / 2)]){
                      352 (gcpscad)
                      353 (gcpscad)
                                         union(){
                                            sphere(r=(es_diameter / 2));
                      354 (gcpscad)
                      355 (gcpscad)
                                            cylinder(r1=(es_diameter / 2), r2=(es_diameter / 2), h=es_flute_length, center=fal
                      356 (gcpscad)
                      357 (gcpscad)
                      358 (gcpscad) }
                      359 (gcpscad)
     gcp_endmill_v
                      360 (gcpscad) module gcp_endmill_v(es_v_angle, es_diameter) {
                      361 (gcpscad)
                      362 (gcpscad)
                                         cylinder(r1=0, r2=(es_diameter / 2), h=((es_diameter / 2) / tan((es_v_angle / 2))),
                                         translate([0, 0, ((es_diameter / 2) / tan((es_v_angle / 2)))]){
                      363 (gcpscad)
                                            cylinder(r1=(es_diameter / 2), r2=(es_diameter / 2), h=((es_diameter * 8)), cente
                      364 (gcpscad)
                      365 (gcpscad)
                      366 \langle gcpscad \rangle
                      367 (gcpscad) }
                      368 (gcpscad)
```

2.3 File Handling

For writing to files it will be necessary to have commands for each step of working with the files.

popengcodefile There is a separate function for each type of file, and for DXFs, there are popendxffile multiple file instances, one for each type of different type and size of tool which it popendxlgblffile is expected a project will work with.

```
popendxflgsqfile
                     369 (gcpy) def popengcodefile(fn):
 popendxflgVfile
                    370 (gcpy)
                                    global f
popendxfsmblfile
                                    f = open(fn, "w")
                    371 (gcpy)
popendxfsmsqfile
                    372 (gcpy)
 popendxfsmVfile
                    373 (gcpy) def popendxffile(fn):
    popensvgfile
                                    global dxf
                    374 (gcpy)
                                    dxf = open(fn, "w")
                    375 (gcpy)
                    376 (gcpy)
                    377 (gcpy) def popendxlgblffile(fn):
                                    global dxflgbl
                    378 (gcpy)
                    379 (gcpy)
                                    dxflgbl = open(fn, "w")
                    380 (gcpy)
                               def popendxflgsqfile(fn):
                    381 (gcpy)
                    382 (gcpy)
                                    global dxfldsq
                                    dxflgsq = open(fn, "w")
                    383 (gcpy)
                    384 (gcpy)
                    385 \langle gcpy \rangle def popendxflgVfile(fn):
                                    global dxflgV
                    386 (gcpy)
                                    dxflgV = open(fn, "w")
                    387 (gcpy)
                    388 (gcpy)
                    389 (gcpy) def popendxfsmblfile(fn):
                                    global dxfsmbl
                    390 (gcpy)
                                    dxfsmbl = open(fn, "w")
                    391 (gcpy)
                    392 (gcpy)
                    393 \langle gcpy \rangle def popendxfsmsqfile(fn):
                    394 (gcpy)
                                    global dxfsmsq
                                    dxfsmsq = open(fn, "w")
                    395 (gcpy)
                    396 (gcpy)
                    397 \langle \texttt{gcpy} \rangle def popendxfsmVfile(fn):
                                    global dxfsmV
                    398 (gcpy)
                                    dxfsmV = open(fn, "w")
                    399 (gcpy)
                    400 (gcpy)
                    401 \langle \text{gcpy} \rangle def popensvgfile(fn):
                                    global svg
                     402 (gcpy)
                    403 (gcpy)
                                    svg = open(fn, "w")
                     404 (gcpy)
                         There will need to be matching OpenSCAD modules for the Python functions.
  oopengcodefile
    oopensvgfile
                    405 (pyscad) module oopengcodefile(fn) {
    oopendxffile
                    406 (pyscad) popengcodefile(fn);
                     407 (pyscad) }
                     408 (pyscad)
                    409 (pyscad) module oopensvgfile(fn) {
                    410 (pyscad) popensvgfile(fn);
                    411 (pyscad) }
                    412 (pyscad)
```

413 (pyscad) module oopendxffile(fn) {

```
414 \langle pyscad \rangle
                                    echo(fn);
                 415 \langle pyscad \rangle popendxffile(fn);
                 416 \langle pyscad \rangle  }
                 417 (pyscad)
                 418 (pyscad) module oopendxflgblfile(fn) {
                 419 \( pyscad \) popendxflgblfile(fn);
                 420 (pyscad) }
                 421 (pyscad)
                 422 (pyscad) module oopendxflgsqfile(fn) {
                 423 \( pyscad \) popendxflgsqfile(fn);
                 424 \langle pyscad \rangle  }
                 425 (pyscad)
                 426 <pyscad> module oopendxflgVfile(fn) {
                 427 (pyscad)
                              popendxflgVfile(fn);
                 428 (pyscad) }
                 429 (pyscad)
                 430 (pyscad) module oopendxfsmblfile(fn) {
                 431 (pyscad)
                               popendxfsmblfile(fn);
                 432 \langle \mathsf{pyscad} \rangle  }
                 433 (pyscad)
                 434 (pyscad) module oopendxfsmsqfile(fn) {
                 435 (pyscad)
                                   echo(fn);
                 436 \langle pyscad \rangle popendxfsmsqfile(fn);
                 437 (pyscad) }
                 438 (pyscad)
                 439 (pyscad) module oopendxfsmVfile(fn) {
                 440 (pyscad) popendxfsmVfile(fn);
                 441 (pyscad) }
                 442 (pyscad)
opengcodefile
  opensvgfile
                 443 (gcpscad) module opengcodefile(fn) {
  opendxffile
                 444 \langle gcpscad \rangle if (generategcode == true) {
                 445 (gcpscad) oopengcodefile(fn);
                 446 (gcpscad)
                                     echo(fn);
                 447 (gcpscad)
                                     owritecomment(fn);
                 448 (gcpscad) }
                 449 (gcpscad) }
                 450 (gcpscad)
                 451 (gcpscad) module opensvgfile(fn) {
                 452 \langle gcpscad \rangle if (generatesvg == true) {
                 453 (gcpscad) oopensvgfile(fn);
                 454 (gcpscad)
                                     echo(fn);
                                     svgwriteone(str("<?xml version=",chr(34),"1.0",chr(34)," encoding=",chr(34),"UTF-8",
                 455 (gcpscad)
                 456 \langle gcpscad \rangle // writesvglineend();
                 457 (gcpscad) svgwriteone(str("<svg version=",chr(34),"1.1",chr(34)," xmlns=",chr(34),"http://www.w3.
                 458 (gcpscad) //<path d="M755.906 0 L755.906 377.953 L0 377.953 L0 0 L755.906 0 Z " stroke="black" str
                 459 \(\rangle gcpscad \rangle \) svgwriteone(str("\(\rangle path d=\),chr(34),\"M\",stocklength\(\fint 3.77953,\" 0 L\",stocklength\(\fint 3.77953,\" \)
                 460 (gcpscad)
                                     }
                 461 (gcpscad) }
                 462 (gcpscad)
                 463 (gcpscad) module opendxffile(fn) {
                 464 \langle gcpscad \rangle if (generatedxf == true) {
                 465 (gcpscad) oopendxffile(str(fn,".dxf"));
                 466 (gcpscad) //
                                      echo(fn);
```

```
467 (gcpscad)
                                    dxfwriteone("0");
                468 (gcpscad)
                                    dxfwriteone("SECTION");
                                    dxfwriteone("2");
                469 (gcpscad)
                                    dxfwriteone("ENTITIES");
                470 (gcpscad)
                471 (gcpscad)
                                    dxfwriteone("0");
                472 (gcpscad) if (large_ball_tool_no > 0) { oopendxflgblfile(str(fn,".",large_ball_tool_no
                473 (gcpscad)
                                    dxfpreamble(large_ball_tool_no);
                474 (gcpscad) }
                475 \(\rangle gcpscad \rangle \) if (large_square_tool_no > 0) { oopendxflgsqfile(str(fn,".",large_square_tool_no > 0) }
                                    dxfpreamble(large_square_tool_no);
                476 (gcpscad)
                477 (gcpscad) }
                478 \(\rangle gcpscad\) if (large_V_tool_no > 0) { oopendxflgVfile(str(fn,".",large_V_tool_no,".dxf"
                                    dxfpreamble(large_V_tool_no);
                479 (gcpscad)
                480 (gcpscad) }
                481 \(\rangle gcpscad \rangle \) if \((\small_ball_tool_no > 0\) \(\rangle \) opendxfsmblfile\((\str(fn, ".", small_ball_tool_no \)
                                    dxfpreamble(small_ball_tool_no);
                482 (gcpscad)
                483 (gcpscad) }
                 484 (gcpscad) if (small_square_tool_no > 0) { oopendxfsmsqfile(str(fn,".",small_square_tool
                                       echo(str("tool no",small_square_tool_no));
                485 (gcpscad) //
                                    dxfpreamble(small_square_tool_no);
                486 (gcpscad)
                487 (gcpscad) }
                488 \( gcpscad \) if \( (small_V_tool_no > 0 \) \( \) oopendxfsmVfile(str(fn,".",small_V_tool_no,".dxf"
                489 (gcpscad)
                                    dxfpreamble(small_V_tool_no);
                490 (gcpscad) }
                491 (gcpscad) }
                492 (gcpscad) }
                493 (gcpscad)
    writedxf
                      Once files have been opened they may be written to.
writedxflgbl
                494~\langle \texttt{gcpy} \rangle def writedxf(*arguments):
writedxflgsq
                                 line_to_write = ""
                495 \langle \mathsf{gcpy} \rangle
                496 \langle \mathsf{gcpy} \rangle
                                 for element in arguments:
                497 \langle \mathsf{gcpy} \rangle
                                      line_to_write += element
                498 \langle gcpy \rangle
                                 dxf.write(line_to_write)
                                 dxf.write("\n")
                499 (gcpy)
                500 (gcpy)
                501 (gcpy) def writedxflgbl(*arguments):
                                 line_to_write = ""
                502 (gcpy)
                                 for element in arguments:
                503 (gcpy)
                                      line_to_write += element
                504 (gcpy)
                505 (gcpy)
                                 dxflgbl.write(line_to_write)
                                 print(line_to_write)
                506 \langle \mathsf{gcpy} \rangle
                507 (gcpy)
                                 dxflgbl.write("\n")
                508 (gcpy)
                509 (gcpy) def writedxflgsq(*arguments):
                                 line_to_write = ""
                510 (gcpy)
                511 (gcpy)
                                 for element in arguments:
                512 (gcpy)
                                      line_to_write += element
                513 (gcpy)
                                 dxflgsq.write(line_to_write)
                                 print(line_to_write)
                514 (gcpy)
                                 dxflgsq.write("\n")
                515 (gcpy)
                516 \langle \mathsf{gcpy} \rangle
                517~\langle \texttt{gcpy} \rangle def writedxflgV(*arguments):
                518 \langle gcpy \rangle
                                 line_to_write = ""
```

519 **(gcpy**)

for element in arguments:

```
line_to_write += element
                  520 (gcpy)
                                   dxflgV.write(line_to_write)
                  521 (gcpy)
                                  print(line_to_write)
                  522 (gcpy)
                                   dxflgV.write("\n")
                  523 (gcpy)
                  524 (gcpy)
                  525 (gcpy) def writedxfsmbl(*arguments):
                                  line_to_write = ""
                  526 (gcpy)
                                   for element in arguments:
                  527 \langle \mathsf{gcpy} \rangle
                                       line_to_write += element
                  528 (gcpy)
                                   dxfsmbl.write(line_to_write)
                  529 (gcpy)
                                   print(line_to_write)
                  530 \langle gcpy \rangle
                                   dxfsmbl.write("\n")
                  531 (gcpy)
                  532 (gcpy)
                  533 \langle gcpy \rangle def writedxfsmsq(*arguments):
                                   line_to_write = ""
                  534 (gcpy)
                                   for element in arguments:
                  535 (gcpy)
                  536 (gcpy)
                                       line_to_write += element
                                   dxfsmsq.write(line_to_write)
                  537 (gcpy)
                                   print(line_to_write)
                  538 (gcpy)
                  539 \langle \mathsf{gcpy} \rangle
                                   dxfsmsq.write("\n")
                  540 (gcpy)
                  541 \; \langle \texttt{gcpy} \rangle def writedxfsmV(*arguments):
                  542 \langle \mathsf{gcpy} \rangle
                                  line_to_write = ""
                  543 (gcpy)
                                  for element in arguments:
                                       line_to_write += element
                  544 (gcpy)
                  545 (gcpy)
                                   dxfsmV.write(line_to_write)
                  546 (gcpy)
                                  print(line_to_write)
                  547 (gcpy)
                                   dxfsmV.write("\n")
                  548 (gcpy)
                  549~\langle \text{gcpy} \rangle def writesvg(*arguments):
                                  line_to_write = ""
                  550 (gcpy)
                                  for element in arguments:
                  551 (gcpy)
                  552~\langle \mathsf{gcpy} \rangle
                                       line_to_write += element
                  553 \langle \mathsf{gcpy} \rangle
                                   svg.write(line_to_write)
                  554 (gcpy)
                                  print(line_to_write)
                  555 (gcpy)
                  556 (gcpy) def pwritesvgline():
                  557 (gcpy)
                                   svg.write("\n")
                  558 (gcpy)
owritecomment
  dxfwriteone
                  559 (pyscad) module owritecomment(comment) {
 dxfwritelgbl
                  560 (pyscad) writeln("(",comment,")");
                  561 (pyscad) }
                  562 (pyscad)
                  563 (pyscad) module dxfwriteone(first) {
                  564 (pyscad) writedxf(first);
                  565 (pyscad) // writeln(first);
                  566 (pyscad) //
                                       echo(first);
                  567 \langle pyscad \rangle }
                  568 \langle \mathsf{pyscad} \rangle
                  569 /pyscad module dxfwritelgbl(first) {
                  570 (pyscad) writedxflgbl(first);
```

```
571 (pyscad) }
572 (pyscad)
573 (pyscad) module dxfwritelgsq(first) {
574 (pyscad) writedxflgsq(first);
575 (pyscad) }
576 (pyscad)
577 (pyscad) module dxfwritelgV(first) {
578 (pyscad) writedxflgV(first);
579 (pyscad) }
580 (pyscad)
581 (pyscad) module dxfwritesmbl(first) {
582 \(\rangle pyscad\rangle \) writedxfsmbl(first);
583 (pyscad) }
584 (pyscad)
585 (pyscad) module dxfwritesmsq(first) {
586 \(\rangle \text{pyscad} \rangle \text{ writedxfsmsq(first);}\)
587 \langle \mathsf{pyscad} \rangle }
588 (pyscad)
589 (pyscad) module dxfwritesmV(first) {
590 (pyscad) writedxfsmV(first);
591 (pyscad) }
592 (pyscad)
593 (pyscad) module svgwriteone(first) {
594 (pyscad) writesvg(first);
595 (pyscad) }
596 (pyscad)
597 (pyscad) module writesvglineend(first) {
598 (pyscad) pwritesvgline();
599 (pyscad) }
600 (pyscad)
601 (pyscad) module owriteone(first) {
602 (pyscad) writeln(first);
603 (pyscad) }
604 (pyscad)
605 \; \langle {\sf pyscad} \rangle \; {\sf module} \; {\sf owritetwo(first, second)} \; \{
606 (pyscad) writeln(first, second);
607 (pyscad) }
608 (pyscad)
609 (pyscad) module owritethree(first, second, third) {
610 (pyscad) writeln(first, second, third);
611 (pyscad) }
612 \langle \mathsf{pyscad} \rangle
613 \langle pyscad \rangle module owritefour(first, second, third, fourth) {
614 (pyscad) writeln(first, second, third, fourth);
615 (pyscad) }
616 (pyscad)
617 (pyscad) module owritefive(first, second, third, fourth, fifth) {
618 (pyscad) writeln(first, second, third, fourth, fifth);
619 (pyscad) }
621 \langle pyscad \rangle module owritesix(first, second, third, fourth, fifth, sixth) {
622 (pyscad) writeln(first, second, third, fourth, fifth, sixth);
623 (pyscad) }
624 (pyscad)
```

```
625 \(\rangle pyscad \rangle \) module owriteseven(first, second, third, fourth, fifth, sixth, seventh) {
               626 (pyscad) writeln(first, second, third, fourth, fifth, sixth, seventh);
               627 (pyscad) }
               628 (pyscad)
               629 (pyscad) module owriteeight(first, second, third, fourth, fifth, sixth, seventh, eighth) {
               630 (pyscad) writeln(first, second, third, fourth, fifth, sixth, seventh, eighth);
               631 (pyscad) }
               632 (pyscad)
               633 (pyscad) module owritenine(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth) {
               634 (pyscad) writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth);
               635 \langle pyscad \rangle }
               636 (pyscad)
               637 \langle pyscad \rangle module owriteten(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tent
               638 (pyscad) writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth);
               639 (pyscad) }
               640 (pyscad)
               641 (pyscad) module owriteeleven(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, t
               642\ \langle \mathsf{pyscad} \rangle writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleve
               643 \langle \mathsf{pyscad} \rangle }
               644 (pyscad)
               645~{
m \langle pyscad 
angle} module owritetwelve(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, t
               646 (pyscad) writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleve
               647 (pyscad) }
               648 (pyscad)
               649 (pyscad) module owritethirteen(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth,
               650 (pyscad) writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleve
               651 (pyscad) }
               652 (pyscad)
    dxfwrite
 dxfpreamble 653 \langle gcpscad \rangle module dxfwrite(tn,arg) {
writesvgline 654 \langle gcpscad \rangle if (tn == large_ball_tool_no) {
                                 dxfwritelgbl(arg);}
               655 (gcpscad)
               656 (gcpscad) if (tn == large_square_tool_no) {
               657 (gcpscad)
                                 dxfwritelgsq(arg);}
               658 (gcpscad) if (tn == large_V_tool_no) {
                                 dxfwritelgV(arg);}
               659 (gcpscad)
               660 (gcpscad) if (tn == small_ball_tool_no) {
               661 (gcpscad)
                                 dxfwritesmbl(arg);}
               662 (gcpscad) if (tn == small_square_tool_no) {
                                 dxfwritesmsq(arg);}
               663 (gcpscad)
               664 \; \langle \texttt{gcpscad} \rangle if (tn == small_V_tool_no) {
                                 dxfwritesmV(arg);}
               665 (gcpscad)
               666 (gcpscad) }
               667 (gcpscad)
               668 (gcpscad) module dxfpreamble(tn) {
                                   echo(str("dxfpreamble",small_square_tool_no));
               669 (gcpscad) //
                                 dxfwrite(tn,"0");
               670 (gcpscad)
               671 (gcpscad)
                                 dxfwrite(tn, "SECTION");
               672 (gcpscad)
                                 dxfwrite(tn,"2");
                                 dxfwrite(tn,"ENTITIES");
               673 (gcpscad)
                                 dxfwrite(tn,"0");
               674 (gcpscad)
               675 (gcpscad) }
               676 (gcpscad)
               677 (gcpscad) module writesvgline(bx,by,ex,ey) {
```

16

```
678 \langle gcpscad \rangle if (generatesvg == true) {
679 (gcpscad)
                  svgwriteone(str("<path d=",chr(34),"M",bx*3.77953," ",by*3.77953," L",ex*
680 (gcpscad)
681 (gcpscad) }
682 (gcpscad)
683 (gcpscad) module dxfbpl(tn,bx,by) {
684 (gcpscad)
                  dxfwrite(tn,"POLYLINE");
685 (gcpscad)
                  dxfwrite(tn,"8");
                  dxfwrite(tn,"default");
686 (gcpscad)
687 (gcpscad)
                  dxfwrite(tn, "66");
688 (gcpscad)
                  dxfwrite(tn,"1");
689 (gcpscad)
                  dxfwrite(tn,"70");
690 (gcpscad)
                  dxfwrite(tn,"0");
691 (gcpscad)
                  dxfwrite(tn,"0");
                  dxfwrite(tn,"VERTEX");
692 (gcpscad)
693 (gcpscad)
                  dxfwrite(tn,"8");
694 (gcpscad)
                  dxfwrite(tn,"default");
695 (gcpscad)
                  dxfwrite(tn,"70");
                  dxfwrite(tn,"32");
696 (gcpscad)
                  dxfwrite(tn,"10");
697 (gcpscad)
698 (gcpscad)
                  dxfwrite(tn,str(bx));
699 (gcpscad)
                  dxfwrite(tn,"20");
700 (gcpscad)
                  dxfwrite(tn,str(by));
                  dxfwrite(tn,"0");
701 (gcpscad)
702 (gcpscad) }
703 (gcpscad)
704 (gcpscad) module beginpolyline(bx,by,bz) {
705 (gcpscad) if (generatedxf == true) {
706 (gcpscad)
                  dxfwriteone("POLYLINE");
707 (gcpscad)
                  dxfwriteone("8");
                  dxfwriteone("default");
708 (gcpscad)
709 (gcpscad)
                  dxfwriteone("66");
710 (gcpscad)
                  dxfwriteone("1");
                  dxfwriteone("70");
711 (gcpscad)
712 (gcpscad)
                  dxfwriteone("0");
713 (gcpscad)
                  dxfwriteone("0");
714 (gcpscad)
                  dxfwriteone("VERTEX");
715 \langle \mathsf{gcpscad} \rangle
                  dxfwriteone("8");
716 (gcpscad)
                  dxfwriteone("default");
717 (gcpscad)
                  dxfwriteone("70");
718 (gcpscad)
                  dxfwriteone("32");
719 (gcpscad)
                  dxfwriteone("10");
720 (gcpscad)
                  dxfwriteone(str(bx));
                  dxfwriteone("20");
721 (gcpscad)
722 (gcpscad)
                  dxfwriteone(str(by));
723 (gcpscad)
                  dxfwriteone("0");
724 (gcpscad)
                  dxfbpl(current_tool(),bx,by);}
725 (gcpscad) }
726 (gcpscad)
727 (gcpscad) module dxfapl(tn,bx,by) {
728 (gcpscad)
                  dxfwrite(tn,"VERTEX");
                  dxfwrite(tn,"8");
729 (gcpscad)
                  dxfwrite(tn,"default");
730 (gcpscad)
731 (gcpscad)
                  dxfwrite(tn,"70");
```

```
dxfwrite(tn,"32");
732 (gcpscad)
733 (gcpscad)
                   dxfwrite(tn,"10");
734 (gcpscad)
                   dxfwrite(tn,str(bx));
735 (gcpscad)
                   dxfwrite(tn,"20");
736 (gcpscad)
                   dxfwrite(tn,str(by));
737 (gcpscad)
                   dxfwrite(tn,"0");
738 (gcpscad) }
739 (gcpscad)
740 (gcpscad) module addpolyline(bx,by,bz) {
741 \langle gcpscad \rangle if (generatedxf == true) {
742 (gcpscad)
                   dxfwriteone("VERTEX");
743 \langle \mathsf{gcpscad} \rangle
                   dxfwriteone("8");
744 \langle \mathsf{gcpscad} \rangle
                   dxfwriteone("default");
745 (gcpscad)
                  dxfwriteone("70");
746~\langle \mathsf{gcpscad} \rangle
                  dxfwriteone("32");
747 (gcpscad)
                  dxfwriteone("10");
748 (gcpscad)
                   dxfwriteone(str(bx));
749 (gcpscad)
                   dxfwriteone("20");
750 (gcpscad)
                   dxfwriteone(str(by));
751 (gcpscad)
                   dxfwriteone("0");
752 (gcpscad)
                   dxfapl(current_tool(),bx,by);
753 (gcpscad)
754 (gcpscad) }
755 (gcpscad)
756 (gcpscad) module dxfcpl(tn) {
757 (gcpscad)
                   dxfwrite(tn, "SEQEND");
758 (gcpscad)
                   dxfwrite(tn,"0");
759 (gcpscad) }
760 (gcpscad)
761 (gcpscad) module closepolyline() {
762 \langle gcpscad \rangle if (generatedxf == true) {
763 (gcpscad)
                  dxfwriteone("SEQEND");
764 (gcpscad)
                  dxfwriteone("0");
765 (gcpscad)
                   dxfcpl(current_tool());
766 (gcpscad)
767 (gcpscad) }
768 (gcpscad)
769 (gcpscad) module writecomment(comment) {
770 (gcpscad) if (generategcode == true) {
771 (gcpscad) owritecomment(comment);
772 (gcpscad) }
773 (gcpscad) }
774 (gcpscad)
```

pclosegcodefile At the end of the project it will be necessary to close each file. In some pclosesygfile instances it will be necessary to write additional information, depending on the pclosedxffile file format.

```
775 \( \text{gcpy} \) \text{ def pclosegcodefile():}
776 \( \text{gcpy} \) \text{ f.close()}
777 \( \text{gcpy} \) \text{ def pclosesvgfile():}
778 \( \text{gcpy} \) \text{ def pclosesvgfile():}
779 \( \text{gcpy} \) \text{ svg.close()}
780 \( \text{gcpy} \) \text{ def pclosedxffile():}
```

```
dxf.close()
                        782 \langle \mathsf{gcpy} \rangle
                        783 (gcpy)
                        784 \langle \text{gcpy} \rangle def pclosedxflgblfile():
                                          dxflgbl.close()
                        785 (gcpy)
                        786 (gcpy)
                        787 (gcpy) def pclosedxflgsqfile():
                        788 (gcpy)
                                          dxflgsq.close()
                        789 (gcpy)
                        790~\langle \texttt{gcpy} \rangle def pclosedxflgVfile():
                                          dxflgV.close()
                        791 (gcpy)
                        792 (gcpy)
                        793 \langle \text{gcpy} \rangle def pclosedxfsmblfile():
                                          dxfsmbl.close()
                        794 (gcpy)
                        795 \langle gcpy \rangle
                        796 \langle \mathsf{gcpy} \rangle def pclosedxfsmsqfile():
                                          dxfsmsq.close()
                        797 (gcpy)
                        798 (gcpy)
                        799 (gcpy) def pclosedxfsmVfile():
                                          dxfsmV.close()
                        800 (gcpy)
                        801 (gcpy)
  oclosegcodefile
     oclosedxffile
                        802 <pyscad > module oclosegcodefile() {
oclosedxflgblfile
                        803 (pyscad) pclosegcodefile();
                        804 (pyscad) }
                        805 (pyscad)
                        806 (pyscad) module oclosedxffile() {
                        807 (pyscad) pclosedxffile();
                        808 \langle pyscad \rangle  }
                        809 (pyscad)
                        810 \(pyscad\)\) module oclosedxflgblfile() {
                        811 \langle pyscad \rangle pclosedxflgblfile();
                        812 \langle pyscad \rangle  }
                        813 (pyscad)
                        814 (pyscad) module oclosedxflgsqfile() {
                        815 (pyscad) pclosedxflgsqfile();
                        816 \langle \mathsf{pyscad} \rangle }
                        817 \langle \mathsf{pyscad} \rangle
                        818 (pyscad) module oclosedxflgVfile() {
                        819 (pyscad) pclosedxflgVfile();
                        820 (pyscad) }
                        821 (pyscad)
                        822 <pyscad > module oclosedxfsmblfile() {
                        823 \langle \mathsf{pyscad} \rangle pclosedxfsmblfile();
                        824 (pyscad) }
                        825 (pyscad)
                        826 <pyscad > module oclosedxfsmsqfile() {
                        827 (pyscad) pclosedxfsmsqfile();
                        828 \langle pyscad \rangle  }
                        829 (pyscad)
                        830 <pyscad> module oclosedxfsmVfile() {
                        831 (pyscad) pclosedxfsmVfile();
                        832 \langle pyscad \rangle  }
                        833 (pyscad)
```

```
834 (pyscad) module oclosesvgfile() {
                  835 \langle pyscad \rangle pclosesvgfile();
                  836 \langle pyscad \rangle  }
                  837 (pyscad)
closegcodefile
  dxfpostamble 838 \langle \texttt{gcpscad} \rangle module closegcodefile() {
  closedxffile 839 (gcpscad) if (generategcode == true) {
                                     owriteone("M05");
                  840 (gcpscad)
                                     owriteone("M02");
                  841 (gcpscad)
                  842 (gcpscad) oclosegcodefile();
                  843 (gcpscad) }
                  844 \langle gcpscad \rangle  }
                  845 (gcpscad)
                  846 (gcpscad) module dxfpostamble(arg) {
                  847 (gcpscad)
                                     dxfwrite(arg,"ENDSEC");
                                     dxfwrite(arg, "0");
                  848 (gcpscad)
                                     dxfwrite(arg,"EOF");
                  849 (gcpscad)
                  850 (gcpscad) }
                  851 (gcpscad)
                  852 (gcpscad) module closedxffile() {
                  853 (gcpscad) if (generatedxf == true) {
                                     dxfwriteone("ENDSEC");
                  854 (gcpscad)
                  855 (gcpscad)
                                     dxfwriteone("0");
                  856 (gcpscad)
                                     dxfwriteone("EOF");
                  857 (gcpscad) oclosedxffile();
                  858 (gcpscad)
                                     echo("CLOSING");
                  859 \( gcpscad \) if (large_ball_tool_no > 0) { dxfpostamble(large_ball_tool_no);
                  860 (gcpscad)
                                     oclosedxflgblfile();
                  861 (gcpscad) }
                  862 \(\rangle gcpscad \rangle \) if (large_square_tool_no > 0) { dxfpostamble(large_square_tool_no);
                  863~\langle \mathsf{gcpscad} \rangle
                                     oclosedxflgsqfile();
                  864 (gcpscad) }
                  865 \langle gcpscad \rangle if (large_V_tool_no > 0) { dxfpostamble(large_V_tool_no);
                                     oclosedxflgVfile();
                  866 (gcpscad)
                  867 (gcpscad) }
                  868 (gcpscad) if (small_ball_tool_no > 0) { dxfpostamble(small_ball_tool_no);
                  869 (gcpscad)
                                     oclosedxfsmblfile();
                  870 (gcpscad) }
                  871 \(\)gcpscad\(\) if \(\)(small_square_tool_no > 0) \(\){ \(\)dxfpostamble(small_square_tool_no);}
                  872 (gcpscad)
                                     oclosedxfsmsqfile();
                  873 (gcpscad) }
                  874 (gcpscad) if (small_V_tool_no > 0) { dxfpostamble(small_V_tool_no);
                  875 (gcpscad)
                                     oclosedxfsmVfile();
                  876 (gcpscad) }
                  877 (gcpscad)
                  878 (gcpscad) }
                  879 (gcpscad)
                  880 (gcpscad) module closesvgfile() {
                  881 (gcpscad) if (generatesvg == true) {
                                     svgwriteone("</svg> ");
                  882 (gcpscad)
                                oclosesvgfile();
                  883 (gcpscad)
                  884 (gcpscad)
                                     echo("CLOSING SVG");
                  885 \langle gcpscad \rangle
                  886 (gcpscad) }
```

887 (gcpscad)

2.4 Movement and Cutting

otm With all the scaffolding in place, it is possible to model tool movement and ocut cutting and to write out files which represent the desired machine motions.

```
orapid
         888 (gcpscad) module otm(ex, ey, ez, r,g,b) {
         889 (gcpscad) color([r,g,b]) hull(){
                          translate([xpos(), ypos(), zpos()]){
         890 (gcpscad)
         891 (gcpscad)
                             select_tool(current_tool());
         892 (gcpscad)
         893 (gcpscad)
                          translate([ex, ey, ez]){
         894 (gcpscad)
                             select_tool(current_tool());
         895 (gcpscad)
         896 (gcpscad)
                        }
         897 (gcpscad) oset(ex, ey, ez);
         898 (gcpscad) }
         899 (gcpscad)
        900 \langle gcpscad \rangle module ocut(ex, ey, ez) {
        901 \(\rangle gcpscad \rangle \) //color([0.2,1,0.2]) hull(){
        902 \langle gcpscad \rangle otm(ex, ey, ez, 0.2,1,0.2);
        903 (gcpscad) }
        904 (gcpscad)
         905 (gcpscad) module orapid(ex, ey, ez) {
         906 (gcpscad) //color([0.93,0,0]) hull(){
         907 (gcpscad) otm(ex, ey, ez, 0.93,0,0);
         908 (gcpscad) }
         909 \langle gcpscad \rangle
        910 (gcpscad) module rapidbx(bx, by, bz, ex, ey, ez) {
         911 \langle gcpscad \rangle // writeln("GO X",bx," Y", by, "Z", bz);
         912 (gcpscad) if (generategcode == true) {
        913 (gcpscad) writecomment("rapid");
        914 (gcpscad) owritesix("GO X",str(ex)," Y", str(ey), " Z", str(ez));
        915 (gcpscad) }
         916 (gcpscad)
                           orapid(ex, ey, ez);
         917 (gcpscad) }
         918 (gcpscad)
        919 (gcpscad) module rapid(ex, ey, ez) {
        920 (gcpscad) // writeln("GO X",bx," Y", by, "Z", bz);
        921 \langle gcpscad \rangle if (generategcode == true) {
        922 (gcpscad) writecomment("rapid");
        923 (gcpscad) owritesix("GO X",str(ex)," Y", str(ey), " Z", str(ez));
        924 (gcpscad) }
        925 (gcpscad)
                           orapid(ex, ey, ez);
        926 (gcpscad) }
        927 (gcpscad)
        928 (gcpscad) module movetosafez() {
        929 (gcpscad) //this should be move to retract height
        930 (gcpscad) if (generategcode == true) {
        931 (gcpscad) writecomment("Move to safe Z to avoid workholding");
        932 (gcpscad)
                           owriteone("G53G0Z-5.000");
         933 (gcpscad) }
         934 (gcpscad)
                          orapid(getxpos(), getypos(), retractheight+55);
```

```
935 (gcpscad) }
936 \langle \mathsf{gcpscad} \rangle
937 (gcpscad) module begintoolpath(bx,by,bz) {
938 \langle gcpscad \rangle if (generategcode == true) {
939 (gcpscad) writecomment("PREPOSITION FOR RAPID PLUNGE");
940 (gcpscad)
                  owritefour("GOX", str(bx), "Y",str(by));
941 (gcpscad)
                  owritetwo("Z", str(bz));
942 (gcpscad)
                  orapid(bx,by,bz);
943 (gcpscad)
944 (gcpscad) }
945 (gcpscad)
946 (gcpscad) module movetosafeheight() {
947 \langle gcpscad \rangle //this should be move to machine position
948 \langle gcpscad \rangle if (generategcode == true) {
949 (gcpscad) // writecomment("PREPOSITION FOR RAPID PLUNGE"); Z25.650
950~\langle \texttt{gcpscad} \rangle //G1Z24.663F381.0 ,"F",str(plunge)
951 (gcpscad) if (zeroheight == "Top") {
952 (gcpscad)
                  owritetwo("Z",str(retractheight));
953 (gcpscad) }
954 (gcpscad) }
955 (gcpscad)
                  orapid(getxpos(), getypos(), retractheight+55);
956 (gcpscad) }
957 (gcpscad)
958 (gcpscad) module cutoneaxis_setfeed(axis,depth,feed) {
959 (gcpscad) if (generategcode == true) {
960 (gcpscad) // writecomment("PREPOSITION FOR RAPID PLUNGE"); Z25.650
961 (gcpscad) //G1Z24.663F381.0 ,"F",str(plunge) G1Z7.612F381.0
962 (gcpscad) if (zeroheight == "Top") {
963 (gcpscad)
                  owritefive("G1",axis,str(depth),"F",str(feed));
964 (gcpscad) }
965 (gcpscad) }
966 (gcpscad) if (axis == "X") {setxpos(depth);}
967 (gcpscad) if (axis == "Y") {setypos(depth);}
968 (gcpscad) if (axis == "Z") {setzpos(depth);}
969 (gcpscad) }
970 (gcpscad)
971 (gcpscad) module cut(ex, ey, ez) {
972 (gcpscad) // writeln("GO X",bx," Y", by, "Z", bz);
973 (gcpscad) if (generategcode == true) {
974 (gcpscad) // writecomment("rapid");
975 \langle gcpscad \rangle owritesix("G1 X",str(ex)," Y", str(ey), " Z", str(ez));
976 (gcpscad) }
977 \langle gcpscad \rangle if (generatesvg == true) {
978 \langle gcpscad \rangle // owritesix("G1 X",str(ex)," Y", str(ey), " Z", str(ez));
979 (gcpscad) // orapid(getxpos(), getypos(), retractheight+5);
980 (gcpscad)
                  writesvgline(getxpos(),getypos(),ex,ey);
981 (gcpscad) }
982 (gcpscad) ocut(ex, ey, ez);
983 (gcpscad) }
984 (gcpscad)
985 \langle gcpscad \rangle module cutwithfeed(ex, ey, ez, feed) {
986 \(\rangle gcpscad \rangle \) \( \text{writeln("GO X",bx," Y", by, "Z", bz);} \)
987 \langle gcpscad \rangle if \langle generategcode == true \rangle {
988 (gcpscad) // writecomment("rapid");
```

```
989 \( \text{gcpscad} \rangle \) owriteeight("G1 X",str(ex)," Y", str(ey), " Z", str(ez),"F",str(feed));
990 \( \text{gcpscad} \rangle \) }
991 \( \text{gcpscad} \rangle \) ocut(ex, ey, ez);
992 \( \text{gcpscad} \rangle \) }
993 \( \text{gcpscad} \rangle \) module endtoolpath() \{
995 \( \text{gcpscad} \rangle \) if \( (\text{generategcode} == \text{true} \rangle \) \{
996 \( \text{gcpscad} \rangle \) //Z31.750
997 \( \text{gcpscad} \rangle \) // owriteone("G53G0Z-5.000");
998 \( \text{gcpscad} \rangle \) owritetwo("Z",str(retractheight));
999 \( \text{gcpscad} \rangle \) orapid(getxpos(),getypos(),retractheight);
1000 \( \text{gcpscad} \rangle \) orapid(getxpos(),getypos(),retractheight);
```

3 gcodepreview_template.scad

```
1002 (gcptmpl) //!OpenSCAD
1003 (gcptmpl)
1004 (gcptmpl) use <gcodepreview.py>;
1005 (gcptmpl) use <pygcodepreview.scad>;
1006 (gcptmpl) include <gcodepreview.scad>;
1007 (gcptmpl)
1008 \langle gcptmpl \rangle  $fa = 2;
1009 \langle gcptmpl \rangle  $fs = 0.125;
1010 (gcptmpl)
1011 (gcptmpl) /* [Export] */
1012 (gcptmpl) Base_filename = "export";
1013 (gcptmpl)
1014 (gcptmpl) /* [Export] */
1015 \langle \text{gcptmpl} \rangle generatedxf = true;
1016 (gcptmpl)
1017 (gcptmpl) /* [Export] */
1018 (gcptmpl) generategcode = true;
1019 (gcptmpl)
1020 (gcptmpl) /* [Export] */
1021 (gcptmpl) generatesvg = false;
1022 (gcptmpl)
1023 (gcptmpl) /* [CAM] */
1024 (gcptmpl) toolradius = 1.5875;
1025 (gcptmpl) /* [CAM] */
1026 \; \langle \texttt{gcptmpl} \rangle \; \; \texttt{large\_ball\_tool\_no} \; = \; 0; \; // \; \; [0:0,111:111,101:101,202:202]
1027 (gcptmpl)
1028 (gcptmpl) /* [CAM] */
1029 (gcptmpl) large_square_tool_no = 0; // [0:0,112:112,102:102,201:201]
1030 (gcptmpl)
1031 (gcptmpl) /* [CAM] */
1032 (gcptmpl) large_V_tool_no = 0; // [0:0,301:301,690:690]
1033 (gcptmpl)
1034 (gcptmpl) /* [CAM] */
1035 (gcptmpl) small_ball_tool_no = 0; // [0:0,121:121,111:111,101:101]
1036 (gcptmpl)
1037 (gcptmpl) /* [CAM] */
1038 (gcptmpl) small_square_tool_no = 102; // [0:0,122:122,112:112,102:102]
```

```
1039 (gcptmpl)
1040~\langle \text{gcptmpl} \rangle /* [CAM] */
1041 (gcptmpl) small_V_tool_no = 0; // [0:0,390:390,301:301]
1042 (gcptmpl)
1043 (gcptmpl) /* [Feeds and Speeds] */
1044 (gcptmpl) plunge = 100;
1045 (gcptmpl) /* [Feeds and Speeds] */
1046 \langle \text{gcptmpl} \rangle feed = 400;
1047 \; \langle \text{gcptmpl} \rangle /* [Feeds and Speeds] */
1048 \langle \text{gcptmpl} \rangle speed = 16000;
1049 (gcptmpl) /* [Feeds and Speeds] */
1050~\langle \text{gcptmpl} \rangle square_ratio = 1.0; // [0.25:2]
1051 \; \langle \text{gcptmpl} \rangle /* [Feeds and Speeds] */
1052 (gcptmpl) small_V_ratio = 0.75; // [0.25:2]
1053 (gcptmpl) /* [Feeds and Speeds] */
1054 (gcptmpl) large_V_ratio = 0.875; // [0.25:2]
1055 (gcptmpl)
1056 (gcptmpl) /* [Stock] */
1057 \langle \text{gcptmpl} \rangle stocklength = 219;
1058 (gcptmpl) /* [Stock] */
1059 \langle \text{gcptmpl} \rangle stockwidth = 150;
1060 \langle \mathsf{gcptmpl} \rangle /* [Stock] */
1061 (gcptmpl) stockthickness = 8.35;
1062 (gcptmpl) /* [Stock] */
1063 (gcptmpl) zeroheight = "Top"; // [Top, Bottom]
1064 (gcptmpl) /* [Stock] */
1065 (gcptmpl) stockorigin = "Center"; // [Lower-Left, Center-Left, Top-Left, Center]
1066 (gcptmpl) /* [Stock] */
1067 \langle gcptmpl \rangle retractheight = 9;
1068 (gcptmpl)
1069 (gcptmpl) filename_gcode = str(Base_filename, ".nc");
1070 (gcptmpl) filename_dxf = str(Base_filename);
1071 \; \langle \texttt{gcptmpl} \rangle filename_svg = str(Base_filename, ".svg");
1072 (gcptmpl)
1073 (gcptmpl) opengcodefile(filename_gcode);
1074 \(\langle gcptmpl \rangle \) opendxffile(filename_dxf);
1075 (gcptmpl)
1076 (gcptmpl) difference() {
1077 (gcptmpl) setupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin);
1078 (gcptmpl)
1079 (gcptmpl) movetosafez();
1080 (gcptmpl)
1081 \(\rangle gcptmpl \rangle \) toolchange(small_square_tool_no,speed * square_ratio);
1082 (gcptmpl)
1083 \langle gcptmpl \rangle begintoolpath(0,0,0.25);
1084 (gcptmpl) beginpolyline(0,0,0.25);
1085 (gcptmpl)
1086 (gcptmpl) cutoneaxis_setfeed("Z",-1,plunge*square_ratio);
1087 (gcptmpl)
1088 (gcptmpl) cutwithfeed(stocklength/2,stockwidth/2,-stockthickness,feed);
1089 \(\rangle gcptmpl \rangle \) addpolyline(stocklength/2,stockwidth/2,-stockthickness);
1090 (gcptmpl)
1091 \langle \texttt{gcptmpl} \rangle endtoolpath();
1092 (gcptmpl) closepolyline();
```

24 Index

```
\begin{array}{c} 1093 \; \langle \text{gcptmpl} \rangle \; \\ 1094 \; \langle \text{gcptmpl} \rangle \\ 1095 \; \langle \text{gcptmpl} \rangle \; \text{closegcodefile();} \\ 1096 \; \langle \text{gcptmpl} \rangle \; \text{closedxffile();} \end{array}
```

References

```
[RS274] Thomas R. Kramer, Frederick M. Proctor, Elena R. Messina. 
https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id= 823374
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

${f Symbols}$	${f M}$	owritecomment (rou-
\\ 33	mpx (variable) 4	tine) 13
\~ 33	mpy (variable) 4	
	mpz (variable) 4	P
\mathbf{C}		pclosedxffile (rou-
closedxffile (rou-	N	tine) 17
tine) 19	\n 124, 499,	pclosegcodefile (rou-
closegcodefile (rou-	507, 515, 523,	tine) 17
tine) 19	531, 539, 547, 557	pclosesvgfile (rou-
currenttool (routine) . 8		tine) 17
currenttool (variable) 4	Ο	pcurrenttool (rou-
_	oclosedxffile (rou-	tine) 8
D	tine) 18	popendxffile (rou-
\Documents 33	oclosedxflgblfile	tine) 10
dxfpostamble (rou-	(routine) 18	popendxflgsqfile
tine) 19	oclosegcodefile (rou-	(routine) 10
dxfpreamble (routine) 15	tine) 18	popendxflgVfile (rou-
dxfwrite (routine) 15	ocut (routine) 20	tine) 10
dxfwritelgbl (rou-	oopendxffile (rou-	popendxfsmblfile
tine) 13	tine) 10	(routine) 10
dxfwriteone (routine) 13	oopengcodefile (rou-	popendxfsmsqfile
	tine) 10	(routine) 10
${f G}$	oopensvgfile (rou-	popendxfsmVfile (rou-
gcp_endmill_ball	tine) 10	tine) 10
$(routine) \dots 9$	opendxffile (routine) 11	popendxlgblffile
<pre>gcp_endmill_square</pre>	opengcodefile (rou-	$(\text{routine}) \dots 10$
$(routine) \dots 9$	tine)	popengcodefile (rou-
gcp_endmill_v (rou-	\OpenSCAD 33	tine) $\dots \dots 10$
tine) $\dots \dots g$	opensvgfile (routine) 11	popensygfile (rou-
getxpos (routine) 7	orapid (routine) 20	tine) 10
getypos (routine) 7	oset (routine) 7	psettool (routine) 8
getzpos (routine) 7	osettool (routine) 8	psetupstock (routine) . 4
${f L}$	osetupstock (routine) 5	psetxpos (routine) 7
-		psetypos (routine) 7
\libraries 33	otm (routine) 20	psetzpos (routine) 7

Index 25

${f R}$	pclosegcodefile 17	setypos (routine) 7
\RapCAD 33	pclosesvgfile 17	setzpos (routine) 7
routines:	pcurrenttool \ldots 8	
closedxffile 19	popendxffile 10	${f T}$
closegcodefile . 19	popendxflgsqfile 10	tool_number (variable) 9
currenttool 8	popendxflgVfile 10	toolchange (routine) . 8
dxfpostamble 19	popendxfsmblfile 10	
$\mathtt{dxfpreamble}$ 15	popendxfsmsqfile 10	\mathbf{U}
$\mathtt{dxfwrite}$ 15	popendxfsmVfile 10	\Users 33
$\mathtt{dxfwritelgbl}$ 13	popendxlgblffile 10	
$\mathtt{dxfwriteone}$ 13	popengcodefile . 10	${f V}$
${\tt gcp_endmill_ball}$. 9	popensvgfile 10	variables:
<pre>gcp_endmill_square</pre>	psettool 8	currenttool \dots 4
9	psetupstock 4	mpx
$\texttt{gcp_endmill_v}$ 9	psetxpos γ	mpy
getxpos γ	psetypos $\ldots 7$	mpz
getypos γ	psetzpos γ	$\verb"tool_number" \dots \dots 9$
$\texttt{getzpos} \dots \gamma$	$selecttool \dots 9$	
oclosedxffile 18	$\mathtt{setupstock}$ 5	\mathbf{W}
oclosedxffile 18 oclosedxflgblfile	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	W writedxf (routine) 12
	-	• •
oclosedxflgblfile	$\verb"setxpos" \dots \dots$	writedxf (routine) 12
oclosedxflgblfile	setxpos	writedxf (routine) 12 writedxflgbl (rou-
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7	writedxf (routine) 12 writedxflgbl (routine) 12
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8	writedxf (routine) 12 writedxflgbl (routine) 12 writedxflgsq (routine)
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12	<pre>writedxf (routine) 12 writedxflgbl (routine) 12 writedxflgsq (routine) 12</pre>
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12	<pre>writedxf (routine) 12 writedxflgbl (routine) 12 writedxflgsq (routine) 12 writeln (routine) 4</pre>
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12 writedxflgsq 12	<pre>writedxf (routine) 12 writedxflgbl (routine) 12 writedxflgsq (routine) 12 writeln (routine) 4 writesvgline (routine)</pre>
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12 writedxflgsq 12 writeln 4	<pre>writedxf (routine) 12 writedxflgbl (rou-</pre>
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12 writedxflgsq 12 writeln 4 writesvgline 15	<pre>writedxf (routine) 12 writedxflgbl (rou-</pre>
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12 writedxflgsq 12 writeln 4 writesvgline 15 xpos 7	<pre>writedxf (routine) 12 writedxflgbl (rou-</pre>
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12 writedxflgsq 12 writeln 4 writesvgline 15 xpos 7 ypos 7 zpos 7	<pre>writedxf (routine) 12 writedxflgbl (rou-</pre>
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12 writedxflgsq 12 writeln 4 writesvgline 15 xpos 7 ypos 7 zpos 7	<pre>writedxf (routine) 12 writedxflgbl (rou-</pre>
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12 writeln 4 writesvgline 15 xpos 7 ypos 7 zpos 7 selecttool (routine) 9	writedxf (routine) 12 writedxflgbl (routine) 12 writedxflgsq (routine) writeln (routine) .
oclosedxflgblfile	setxpos 7 setypos 7 setzpos 7 toolchange 8 writedxf 12 writedxflgbl 12 writedxflgsq 12 writeln 4 writesvgline 15 xpos 7 ypos 7 zpos 7	<pre>writedxf (routine) 12 writedxflgbl (rou-</pre>