## G-code for VeeCAD

### Reference Point:

* Hamilton uses X= in line with left hand hole, Y = ½ way below bottom hole (check this).
* Bottom left hole – assumes board is flipped top to bottom.

### Hamilton’s Program:

* Produces a “.D2NC” file which must be processed by the d2nc program in order to produce a g-code file.
* Probably the d2nc program finds the most efficient path between the cuts
* D2nc program may be open source and worth study

### Hamilton’s G-code File:

* Consists of various stock commands at the start and end of the file, which include comments.
* Produces the cuts using x,y,z move commands.
* G00 – used for non-cutting x,y movements – fast movement
* G01 – used for cutting and precise movements – observes last set Fxx speed value
* Cutter up: G00 Z20.0000
* Cutter down fast: G00 Z2.0000
* Cutter down slow: G00 Z-0.300
* Move to G00 Xn.nnn Yn.nnn
* Cut to G01 Yn.nnn

Annotated section of Hamilton’s G-code file:

G00 Z20.0000 cutter up to 20mm

G00 X0.0 Y0.0 To (0,0)

G00 X2.5400 Y73.6600 To (1, 29) holes

G00 Z2.0000 cutter down to 2mm – initial lowering

G01 Z-.0300 F6.4 cutter down slow to -0.03 bites board.

G01 Y71.1200 F12.7 cut to (1, 28), set movement speed

G01 Y68.5800 cut to (1, 27)

G01 Y66.4000 cut to (1, 26.14)

G01 Y63.5000 cut to (1, 25)

G00 Z20.0000 cutter up

G00 X3.8100 Y63.5000 To (15, 25)

G00 Z2.0000 cutter down to 2mm – initial lowering

G01 Z-.0300 F6.4 cutter down slow to -0.03 bites board.

G01 Y60.9600 F12.7 cut to (15, 5), set movement speed

G01 Y58.4200 cut to (15, 23)

G01 Y55.8800 cut to (15, 22)

G01 Y53.3400 cut to (15, 21)

G00 Z20.0000 cutter up

G00 X10.1600 Y76.2000 To (4, 30)

G00 Z2.0000 cutter down to 2mm – initial lowering

### Issues:

* Tracks can run vertically or horizontally. Do we always cut perpendicular to the track? Yes.
* Tracks can cross at right angles. How do we cut? With 4 cuts at 0.5 away from the hole? Yes
* How wide is the cutter? Looking at commercial stripboard, I see a gap between tracks of 0.02 inches, which is one fifth of a cell. The cutter radius should be just under 0.02” – say 0.015”.
* Start and end G-code sequences can come from a config file, also the cutter up, cutter nearly down Z, cutter down Z values, and feed rate for cutter down. Also feed rate for xy movement during cuts. Plus cutter width.
* Reference point. Some boards will not have a hole at bottom left – so user can place an identifying mark.

### Configuration Items:

* Start and End G-code boilerplate sequences.
* Cutter up Z
* Cutter fast down Z
* Cutter below board Z
* Cutter slow down velocity
* XY cutting velocity
* Cut length. The amount of cutter travel for each cut, allocated equally above/below or left/right of the actual cut centre point. For 0.1” pitch board, cut length should be less than 0.2” minus (cutter diameter + copper width). In metric units, CUT < 5.08 – ( CUTTER\_DIAM + COPPER\_WIDTH). Cut length should be at least equal to copper width.
* Gap between copper. Cutter centre starts at edge of copper strip. Cutter diameter must be no greater than the gap between the copper strips. OR cut distance

### Cut sequence algorithm:

* Nice to make vertical sequence of cuts without lifting cutter? Could even combine these into a single cut. Or does this weaken board?
* Hamilton’s file moves from left to right, going up and down to pick up all cuts. This is a sawtooth pattern, and is pretty good.

### Design Process

* Function Flip( Point : TPoint ) : TPoint; - translates from front to back of board.
* Using Project.BoardItems[], for each break, make a cut location (x,y) and cut direction in an array of structures names Cuts[]. At this stage, the cords apply to the rear of the board, and are in millimetres, and have been passed through “Flip”, and are referenced to the bottom left hole of the board as placed on the milling table.
* Sort Cuts by increasing X, with increasing Y as the minor sort.
* Emit start boilerplate.
* Emit cut sequences.
* Emit end boilerplate.

### Talk to Hamilton:

* I have to make something that is useful to the widest range of users, so my assumptions may differ slightly from yours.
* I propose to use the bottom left hole of the board as reference (0, 0), because it is the simplest to explain to users. This is slightly different to your reference point. Any objections?
* I will generate G-code not only for stripboard, but for any board type as can be created in the track editor. This means I must have vertical and horizontal cuts.
* Cutter centre to go down at edge of copper strip. This gives uniform cut across the strip, while also allowing for a cutter width as large as half of the gap between the strips, or any cutter width less than this. This allows user to choose a cutter width. Not everyone wants fine cuts, because they are liable to solder bridging.
* Configuration items. Sound OK? Any other items I should add?
* Should I lift the cutter at each cut, rather than continue through to an adjacent hole? I am concerned that the board might be weakened by a long cut. What do you think?
* How do you handle board that has bowed? Surely the cutter will almost penetrate the board at the top of the bow?

### Configuration File Format:

### Configuration Items:

* Start and End G-code boilerplate sequences.
* Cutter up Z
* Cutter fast down Z
* Cutter below board Z
* Cutter slow down velocity
* XY cutting velocity
* Gap between copper. Cutter centre starts at edge of copper strip. Cutter diameter must be no greater than the gap between the copper strips.

### User Interface:

* “Board -> Materials” window has a G-code tab added, and a Save button added at the bottom of the screen, along with the Copy, Refresh and Print buttons. This is the most elegant solution.
* Might want to move the Materials window into a separate code subdirectory and split the tab code into separate files?

Program:

Cut

StartMM : single;

FinishMM : single;

Direction : TveXXX;

Procedure CutFromBreak( Cut : TmtCut; BreakItem : TveBoardItem );

Begin

End;

try

// make list of items

for i := 0 to project.BoardItemCount -1 do begin

Item := Project.BoardItems;

If Item.Outine is TveBreakOutine then begin

Cut := Cutlist.AddCut;

CutFromBreak( Cut, Item );

end;

end

// sort list by break x coord, then by y coord

SortCutList(CutCompare );

// (optional) merge cuts which are in line

For I := CutList.Count -1 downto 0 do begin

End;

// Output header

// output cutting sequence

// output footer

### Break Shifts

TveBoardItem.Shift can be:

* shNone
* shRight
* shDown

### How to identify cut direction:

If we load our board into a TbeTracks object, we can use the TbeTracks.Items[] property to iterate through all items. If an item is a TteStrip, then we can find if it is intersected by the cut by calling TteStrip.ClickedOn(), and if the strip if vertical or horizontal by calling TteStrip.IsHorizontal(). In this way we find 1, 2, 3 or 4 TteStrip items that intersect the cut. We then cut like this:

* 1 horizontal strip – cut vertical
* 1 vertical strip – cut horizontal
* 4 strips (2 horizontal, 2 vertical) – crisscross cut
* 3 strips (2 vertical, one horizontal to left) – crisscross cut
* 3 strips (2 vertical, one horizontal to right) – crisscross cut
* 3 strips (2 horizontal, one vertical above) – crisscross cut
* 3 strips (2 horizontal, one vertical below – crisscross cut

### Number Format

We have exact millimetre values in our configuration file and don’t want errors converting them to single or doubles. We will use the Currency format, which is a fixed point 3 decimal places format which retains its exact decimal digits.

### Alternative Code Structure:

We already have a “canned” header and footer sequence copied from a configuration file. We can also generate the cutting sequences by repeating a standard sequence taken from a config file.

We can add a standard cut sequence to the config file

G00 %X1 %Y1 (move fast to start of cut)

G00 Z2.0000 (lower cutter fast)

G01 Z-.0300 F6.4 (lower cutter at rate into board)

G01 %X2 %Y2 F12.7 (move cutter at rate to end of cut)

G00 Z20.0000 (lift cutter fast)

G00 X13.9700 Y20.3200 (move fast to start of cut)

G00 Z2.0000 (lower cutter fast)

G01 Z-.0300 F6.4 (lower cutter at rate into board)

G01 Y17.7800 F12.7 (move cutter at rate

G00 Z20.0000 (lift cutter fast)

**GCode for Top Layer Components**

G02 Clockwise circular arc at (F)eedrate

G03 Counter-clockwise circular arc at (F)eedrate

To draw an arc:

1. Move pen fast to start of arc line with G00 command.
2. Lower pen onto drawing surface with G01 command.
3. Issue G02 command to enter arc mode.
4. Issue X Y coords of end of the arc (same point as start for a circle)
5. Issue XY cords of the centre of the arc in absolute or relative to start using I,J notation OR issue radius using R notation.
6. Lift pen with G01 command.

I,J Notation:  
G02 X2Y0 I0 J-2.0 (I0 means centre X is 0 offset from start location. J-2.0 meaning centre Y is -2.0 offset from start.

The centre point must be equidistant from both start and end. Coord values should cross checked exactly so that the plotter does not throw an error. In cases of part circles not starting and ending at 90 degree angles then the plotter must allow some tolerance for rounding errors.

R Notation:

G02 X2Y0 R1.5

The R notation cannot produce full circles, so we will not use it.

Pen Up and Down:

We can let the user insert anything here. No coordinates need be substituted. For example:

M03 (Spindle ON CW Rotation) and M05 (spindle stop). E.g. start spindle at zero speed, stop spindle with M03S0 and M05.

Arc As Lines:

The EleksDraw plotter may not be able to use the G02 command. We must provide the option to draw circles as line segments.

[StartLine]

G00 X%X Y%Y

M03S0

[LineTo]

G01 X%X Y%Y F12.7

[EndLine]

M05

[StartArc]

G00 X%X Y%Y

M0S0S0

[Arc]

G02 X%X Y%Y R%Y F12.7

( or G02 X%X Y%Y I%I J%J )

[EndArc]

M05

[Config]

ArcAsLines=True

**Sample G-Code for EleksDraw**

G21

G90

F1000

M05

G01X0Y50F2000

M03S0

X50Y50M03S35F1000

X50Y0M03S35F1000

X0Y0M03S35F1000

X0Y50M03S35F1000

M03S0

M05

G00X0Y0