The PCB-GCODE User's Manual

Version 3.6.1

John T. Johnson

pcbgcode@pcbgcode.org

January 29, 2014

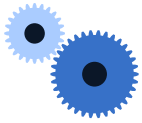
About this book

This book is typeset with LaTeX using the Computer Modern and Palatino fonts.

The LaTeX distribution used is MacTeX. The editor is Texpad for Mac OS X and iOS. When doing academic writing, I use BibDesk on Mac OS X and PocketBib on iOS.

Some words, such as MacTeX in the previous paragraph, are links to websites. In this manual, all links are colored gray. The reader will note that other words are links to figures, tables, or sections in this manual. These are also colored gray.

Throughout this book there are icons in the margins that will help the reader navigate the book and find material at an appropriate skill level. These icons are shown below.



Sections marked with this icon provide background on the operation of pcb-gcode. This information is for curious users and can safely be skipped.

Some parts of the manual are for intermediate or advanced users. This icon alerts the reader to this fact.

Supporters

There is a great community that has built up around pcb-gcode where users freely offer advice and help solve problems for other users. Despite the fact that pcb-gcode has always been free and always will be, a few outstanding individuals have donated time or money (which goes towards paying for web hosting for `pcbgcode.org`). Your generosity warms my heart, and I thank you all!

- Art Eckstein is the tireless moderator of the Yahoo! group.
- Translations
  - Maurício Dias — Portuguêse
  - Wolfgang Schmidt — German
- Monetary Donations (If I missed you, *please* let me know.)
  - Anonymous (2007), Greg Brill, Paul Kiedrowski, Steven Manzer, Barry Ward, Ward Elder, Art Eckstein, World of ElectroMechanica, Austin Valley Software Corporation, webstudio.co.uk, Juan Posada, Richard McCarthy, Keeley Electronics
- Code Contributions
  - Martin L. Marriott - optimizer for milling g-code

Contents

1 Introduction **1**
- 1.1 Purpose . . . 1
- 1.2 Features . . . 1
- 1.3 How it Works . . . 2
  - 1.3.1 Overview . . . 2
  - 1.3.2 Isolation . . . 3
  - 1.3.3 Drilling . . . 4

2 Setup **5**
- 2.1 EAGLE compatibility . . . 5
- 2.2 Installation . . . 5
  - 2.2.1 Downloading and unarchiving . . . 5
  - 2.2.2 Configuring EAGLE . . . 6
  - 2.2.3 Selecting g-code style . . . 6
  - 2.2.4 Previewer Setup . . . 8
- 2.3 Machine Setup . . . 8
- 2.4 Generation Options . . . 9
- 2.5 GCode Options . . . 12

3 Using pcb-gcode **15**
- 3.1 Running pcb-gcode . . . 15
- 3.2 Using EAGLE's DRC . . . 16
- 3.3 Previewer . . . 18
- 3.4 Saving and Loading Settings . . . 22
  - 3.4.1 Overview . . . 22
  - 3.4.2 Saving Settings . . . 24
  - 3.4.3 Loading Settings . . . 24

4 Customizing **25**
- 4.1 G-Code . . . 25
- 4.2 Profiles . . . 29

4.3 Drill Rack Files 29
4.4 User GCode 31

A Sample Mach3 Profile **35**

List of Figures

1.1 Preview showing color-coded multiple passes. 3
1.2 Preview showing a zoomed version of the color-coded multiple passes. Brown is the first pass, red the second, orange the third, etc. 4

2.1 The proper directory structure after uncompressing the archive. 6
2.2 Add the path to pcb-gcode to the User Language Programs option. 7
2.3 Select the style g-code pcb-gcode should produce. 7
2.4 Overwrite warning for gcode-defaults.h. 8
2.5 Machine options. 9
2.6 Options available when generating a board. 10
2.7 Options for generating g-code files. 12

3.1 EAGLE shortcut key assignments. 15
3.2 Two components in the board layout editor. 16
3.3 Pcb-gcode settings for the DRC example. 17
3.4 The pads are far enough apart to allow them to be isolated. 17
3.5 The two pads are too close together and cannot be isolated. A bridge is formed. 17
3.6 EAGLE's DRC indicating the two pads are too close together. 18
3.7 The previewer showing a multi-pass file for the bottom side of an example PCB included with pcb-gcode. Each × represents a drill hole in the drill file. 19
3.8 The sample board `docs/examples/04151_lcdi2c.brd` with wires and mirrored text on the Milling layer. 20
3.9 Milling generated for milling from the bottom side of `docs/examples/04151_lcdi2c.brd`. 21
3.10 Text engraving generated for cutting on the bottom side of `docs/examples/04151_lcdi2c.brd`. 21
3.11 The object info for the mirrored text on the Milling layer. Note that the Mirror checkbox is on. This indicates to pcb-gcode that the text should be engraved on the bottom side of the board. 22
3.12 The Plugins tab showing where settings can be saved and loaded. 23

List of Tables

3.1 Keys available in previewer 18

Chapter 1

Introduction

1.1 Purpose

Pcb-gcode is a User Language Program (ULP) for EAGLE PCB design software produced by CadSoft. Pcb-gcode allows one to make printed circuit boards by using a CNC router or milling machine to cut the traces out of the copper on the board. It also produces files for drilling holes. Two-sided boards are supported. By "mechanically etching" the boards, no toxic chemicals are needed – making the process more environmentally friendly. Turn-around times and costs are much reduced from ordering a prototype from a board house.

1.2 Features

Though no program can be all things to all people, pcb-gcode has a lot of features to help make it useful.

One or two sided boards There are checkboxes for selecting whether to generate files for the top and/or bottom sides of the boards.

Outlines Generate gcode for cutting around the tracks of the PCB. Multiple passes are possible, which helps eliminate the small slivers that may be left behind. There is also an option to make only one outline pass.

Drills Generate gcode for drilling component and mounting holes. Tool changes are supported, as well as a drill rack file.

Preview A preview for outlines is available.

Milling Milling code can be generated for any wires drawn on the Milling (46) layer.

Text Generates gcode to engrave any vector text that is on the Milling (46) layer.

Spot drill holes Holes can be spot drilled with the outlines tool to help the drill bits to start straight when drilling holes.

Tool change position Where the machine should go so that the tool can be changed.

Drill rack files Allows using one drill bit for a range of hole sizes in the board.

Profile Starting settings for particular styles of gcode, for example, Mach3 or EMC, among others.

Embedding comments Comments documenting the settings a file was created with can be inserted into the gcode.

User gcode For users that need to generate gcode for unusual situations.

File naming Several options exist for naming files according to the conventions of the user's controller, and their local language.

Plugins Allow for future expansion.

1.3 How it Works

1.3.1 Overview

After designing your board in Eagle's board editor, the ULP pcb-gcode-setup is run and options are set (See Figure 2.6 on page 10). Pcb-gcode will generate a set of files that will cut out the tracks, pads, pours and vias (hereinafter called tracks) for the top and/or bottom of the board. Pcb-gcode can also generate files to drill holes from the top and/or bottom of the board. Since the holes usually go all the way through, they only need to be drilled from one side, although some users have drilled slightly more than half way from both sides for a cleaner finish. The user may also choose to create milling wires on the milling layer of the board. This can be used to cut out sections of the board, or cut the perimeter of the board out. There is also an option for engraving vector text that the user places on the milling layer.

After the files are generated, they are transferred to the control software for the CNC router or milling machine. The PCB is mounted on the machine. The origin is set to the lower left for the top side of the board and the top files are run. The board is then flipped in the `X` axis (i.e. around the `Y` axis), the origin is set at the lower right, and the bottom files are run. After minimal inspection and cleanup, the board is ready to be loaded with components.

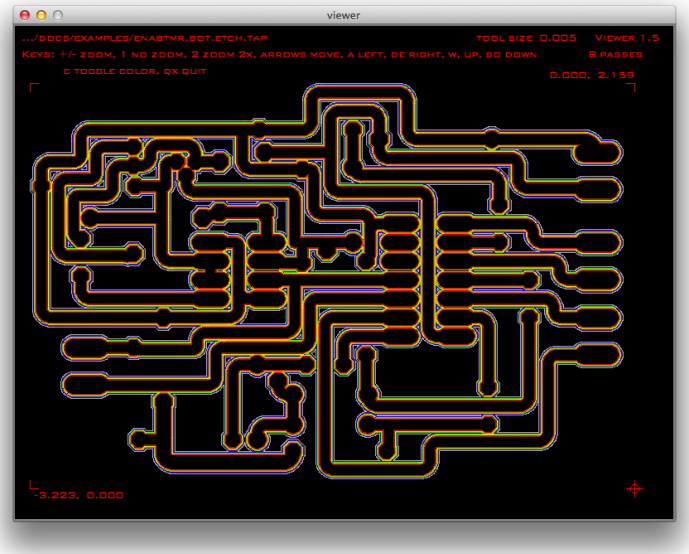


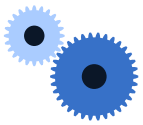
Figure 1.1: *Preview showing color-coded multiple passes.*

1.3.2 Isolation

Pcb-gcode has the ability to generate g-code that cuts out tracks with increasing amounts of isolation. This is helpful because it can help eliminate small slivers of copper left behind by the cutting process.

The isolation begins at a minimum amount and is increased by a step size until a maximum amount is reached. The builtin previewer can be used to see how this isolation works. See Figure 1.1. From the zoomed view shown in Figure 1.2 on the following page you can see that the passes start out close to the track, then move out by a step size. The colors of the preview tracks follow the standard resistor color codes, so brown is the first pass, red is the second, orange the third, etc.

For the curious reader, the isolation is calculated as shown in Equation 1.1. You do not need to know this to use pcb-gcode. Note that this formula is erroneous and that the *EtchingToolSize* should be divided by 2 to yield the offset. Changing this now would break many user's setups, so it will be left as is. Most users arrive at the setting for the etching tool size by trial and error, so it is likely a moot point.



$$isolation = EtchingToolSize + MinimumIsolation + PassNumber * StepSize \quad (1.1)$$

Figure 1.2: *Preview showing a zoomed version of the color-coded multiple passes. Brown is the first pass, red the second, orange the third, etc.*

1.3.3 Drilling

Compared to the isolation passes and files, drill file creation is relatively straight forward. Each hole in the board is sorted according to size and distance from other holes of the same size. G-code is created to position and drill each hole. Tool changes can be included in the file so that the user or an automated tool changer can change the bit. Optionally a rack file can be used.

Rack files contain a list of drills the user has available and the size holes they can be used for. For example, if a library part has a 1mm hole, and another part has a 1.1mm hole, a 1.1mm drill bit could be used for both. For more information on rack files, see Section 4.3 on page 29.

Chapter 2

Setup

2.1 EAGLE compatibility

Pcb-gcode is compatible with EAGLE versions 5 and 6[1]. For versions of EAGLE before version 5, pcb-gcode version 3.3.3 is still available in the Yahoo! group. This manual does not apply to version 3.3.3. Please see the documentation included with version 3.3.3.

2.2 Installation

2.2.1 Downloading and unarchiving

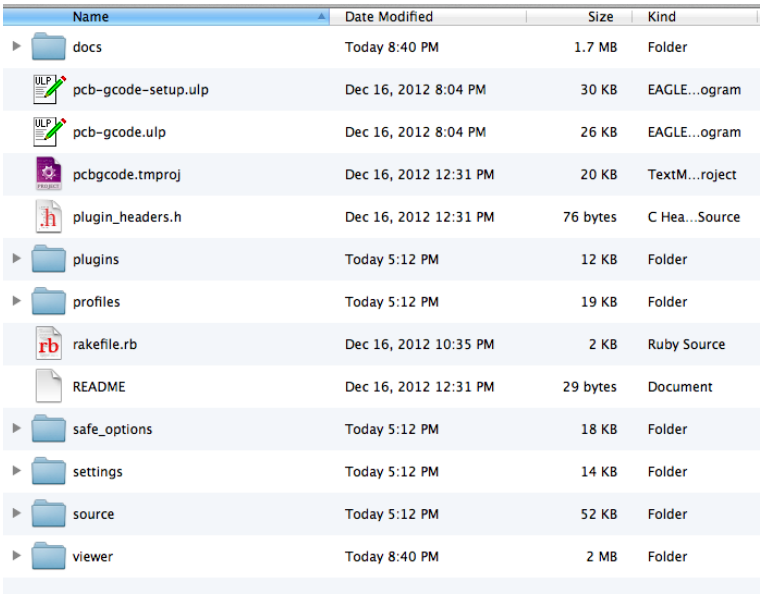
Pcb-gcode can be downloaded from the Yahoo! group's software folder. Unzip the archive into a place where the operating system will allow files to be saved. For Windows, this should be somewhere inside your Documents[2] folder. For Mac OS X, it could be, for instance, `~/Documents/Eagle/pcbgcode`, and for Linux, somewhere off your home folder. Be sure to preserve the directory structure in the archive. See Figure 2.1 on the next page.

Windows Note

Note that if you are using Windows, odd things might happen if you install pcb-gcode outside the Documents folder. This is due to Windows File Protection or Windows Resource Protection, which try to keep files outside the Documents folder from being changed. This could result in, for instance, the previewer always showing the same preview. If you install outside the Documents folder and have trouble, try installing under your folder inside Documents.

[1] CadSoft changed the way numbers were represented internally with their release of version 6. This effectively broke parts of pcb-gcode. Version 3.6 incorporates changes for compatibility with version 5 or 6.

[2] For Windows XP: "C:\Documents and Settings", for Windows 7: "C:\Users\[yourname]\My Documents"



Name	Date Modified	Size	Kind
docs	Today 8:40 PM	1.7 MB	Folder
pcb-gcode-setup.ulp	Dec 16, 2012 8:04 PM	30 KB	EAGLE...ogram
pcb-gcode.ulp	Dec 16, 2012 8:04 PM	26 KB	EAGLE...ogram
pcbgcode.tmproj	Dec 16, 2012 12:31 PM	20 KB	TextM...roject
plugin_headers.h	Dec 16, 2012 12:31 PM	76 bytes	C Hea...Source
plugins	Today 5:12 PM	12 KB	Folder
profiles	Today 5:12 PM	19 KB	Folder
rakefile.rb	Dec 16, 2012 10:35 PM	2 KB	Ruby Source
README	Dec 16, 2012 12:31 PM	29 bytes	Document
safe_options	Today 5:12 PM	18 KB	Folder
settings	Today 5:12 PM	14 KB	Folder
source	Today 5:12 PM	52 KB	Folder
viewer	Today 8:40 PM	2 MB	Folder

Figure 2.1: *The proper directory structure after uncompressing the archive.*

2.2.2 Configuring EAGLE

Now that pcb-gcode is uncompressed, Eagle must know where it is located. In Eagle's `Control Panel`, click `Options` | `Directories`, then put the path to pcb-gcode in the `User Language Programs` field. See Figure 2.2 on the facing page.

2.2.3 Selecting g-code style

To complete the setup, pcb-gcode must be told which type of g-code it should generate. Open a board in Eagle, then click `File` | `Run...`. Locate the folder where pcb-gcode is and select pcb-gcode-setup.ulp. You will see the screen in Figure 2.3 on the next page. Select the style g-code that most closely matches your controller.

You will receive the warning shown in Figure 2.4 on page 8. If this is the first time pcb-gcode has been run, just click Yes and skip the rest of this paragraph. If this is an existing installation of pcb-gcode and gcode-defaults.h has been modified, make note of the modifications before clicking Yes, then make those modifications as needed to the new gcode-defaults.h file.

After clicking Yes, `pcb-gcode-setup` will be run again, and you will see the screen shown in Figure 2.6 on page 10.

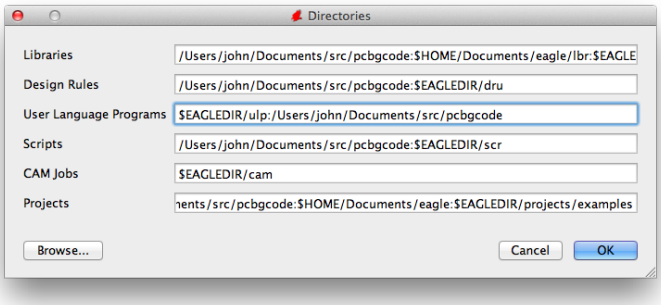


Figure 2.2: *Add the path to pcb-gcode to the User Language Programs option.*

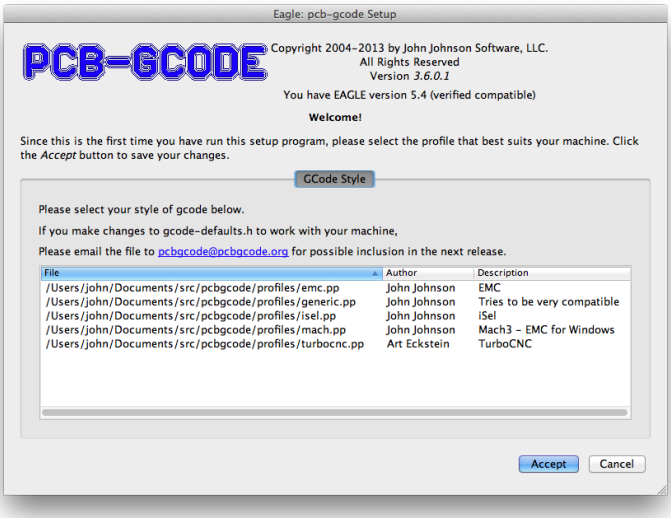


Figure 2.3: *Select the style g-code pcb-gcode should produce.*

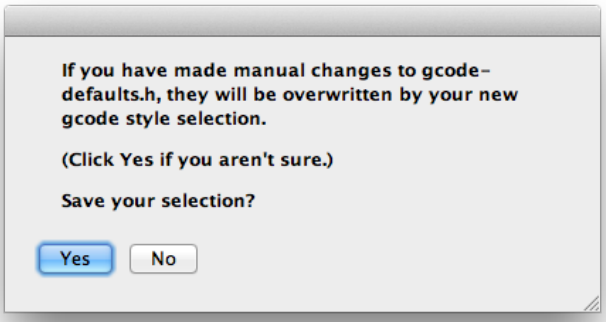


Figure 2.4: *Overwrite warning for gcode-defaults.h.*

2.2.4 Previewer Setup

The previewer (See section 3.3 on page 18) is written in Processing and requires javaw.exe to run. You can download a Java runtime environment (JRE) at java.com. You will find installation instructions there.

If you choose not to use the previewer, simply turn it off. See Figure 2.6 on page 10.

2.3 Machine Setup

Click the `Machine` tab to view the machine options as shown in Figure 2.5 on the facing page. First select the preferred unit of measure by selecting it under `Units`.

Now set the settings for the Z axis. `Z High` should be high enough to clear any clamps or fixtures that hold the PCB down. Set `Z Up` high enough to clear the board when moving from location to location. Set `Z Down` to the depth into the board that the tool should cut when etching. `Drill Depth` should be set deep enough to drill through the PCB. `Drill Dwell` is the time, in seconds, that the drill should pause at the bottom of the hole.

The `Tool Change` options are the position where the tool should be moved when a new tool is required.

The `Spin Up Time` in the `Spindle` box should be set to the length of time in seconds that it takes the spindle to come up to speed. If the spindle is manually controlled, this can be set to `1`.

The `Feed Rates` should be set for `X Y` moves as well as `Z` moves. Rates here will usually be quite low unless the machine has a very fast spindle. See a machinist's reference on how to calculate the optimal feed rate, use trial and error, or post to the Yahoo! group

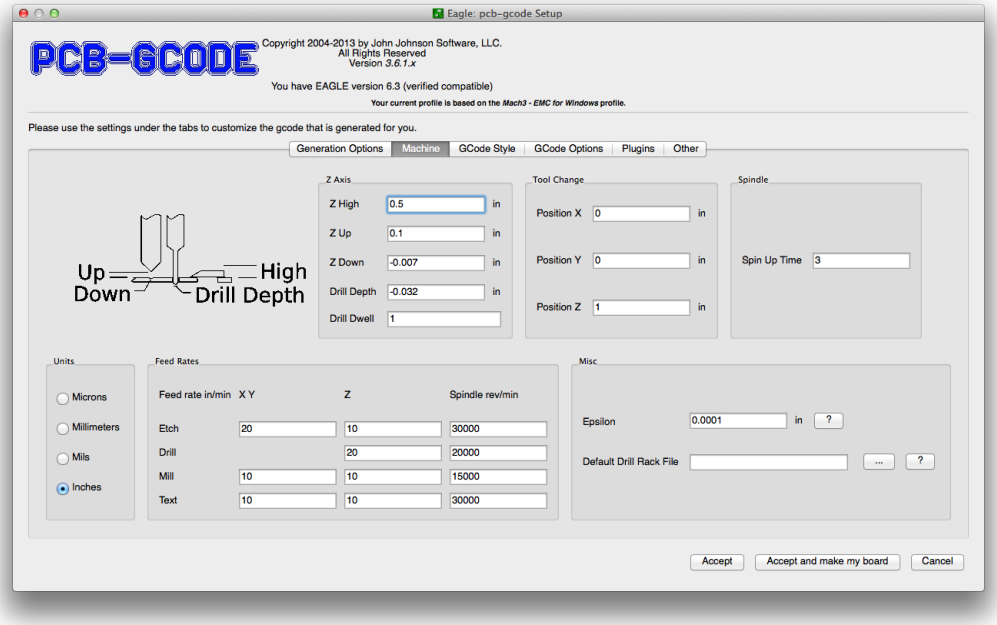


Figure 2.5: *Machine options.*

email list for advice. Note that there are separate feed rates for `X Y`, and `Z`, and there is a spindle speed setting for each of the four operations pcb-gcode can generate files for.

`Epsilon` is the minimum move that will be written to the g-code file. For instance, if `Epsilon` is set to 0.0001″ then the g-code file will not contain movements less than 0.0001″. This option will rarely need to be changed.

The `Default Drill Rack File` option allows for the selection of a rack file to be used if one has not been setup for a particular board. In most cases this can be left blank to start with. See Section 4.3 on page 29 for more information about setting up rack files.

2.4 Generation Options

Now that reasonable values have been set for the machine, click the `Generation Options` tab (See Figure 2.6 on the following page). This is where the various files produced by pcb-gcode can be selected, and common options can be set. A description of the options follows:

Top Side Options having to do with the tracks on the top of the board, and drill holes made from the top side of the board.

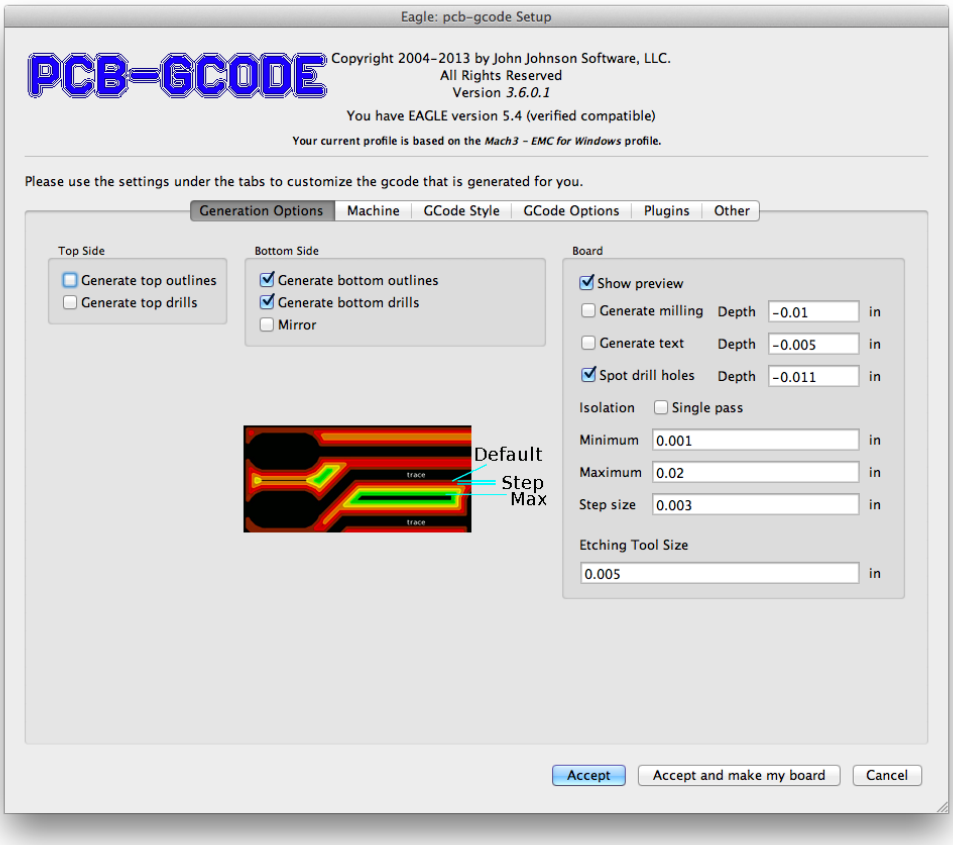


Figure 2.6: *Options available when generating a board.*

Generate top outlines Generate g-code to cut out the tracks, pads, pours and vias on the top side of the board.

Generate top drills Generate g-code to drill holes from the top side of the board.

Bottom Side Options having to do with the tracks on the bottom of the board, and drill holes made from the bottom side of the board.

Generate bottom outlines Generate g-code to cut out the tracks, pads, pours and vias on the bottom side of the board.

Generate bottom drills Generate g-code to drill holes from the bottom side of the board.

Mirror X coordinates for the bottom of the board are usually negative. This makes setting the origin for a two-sided board easier. Turning this option on causes the X coordinates to be positive, however, the bottom tracks will be a mirror image of what they should be. So in general, leave this option off.

Board Options that apply to the board in general.

Show preview Use the previewer in pcb-gcode to preview the g-code generated.

Generate milling Generate g-code for any wires the user has drawn on the `Milling` (46) layer. `Depth` sets the milling depth.

Generate text Generate g-code to engrave any vector text the user may have placed on the `Milling` layer. `Depth` sets the engraving depth. Not that text on the top or bottom layers will be outlined just as the tracks are, whereas text on the milling layer is engraved. That is, the tool along the center of the lines that make up the letter.

Spot drill holes Spot drilling helps the drill bits center themselves and helps prevent "walking." Depth sets the spot drill depth.

Isolation The cutting tool can make several passes around the tracks at an increasing distance each time. This helps eliminate slivers of copper that remain.

Single pass When turned on, only a single pass will be made around the tracks on the board.

Minimum The minimum distance the cutting tool will be away from tracks. That is, the starting isolation amount.

Maximum The maximum distance the cutting tool will be away from tracks. The maximum isolation amount.

Step size The amount the isolation increases with each pass.

Etching Tool Size The size of the cutter used to cut around tracks on the board.

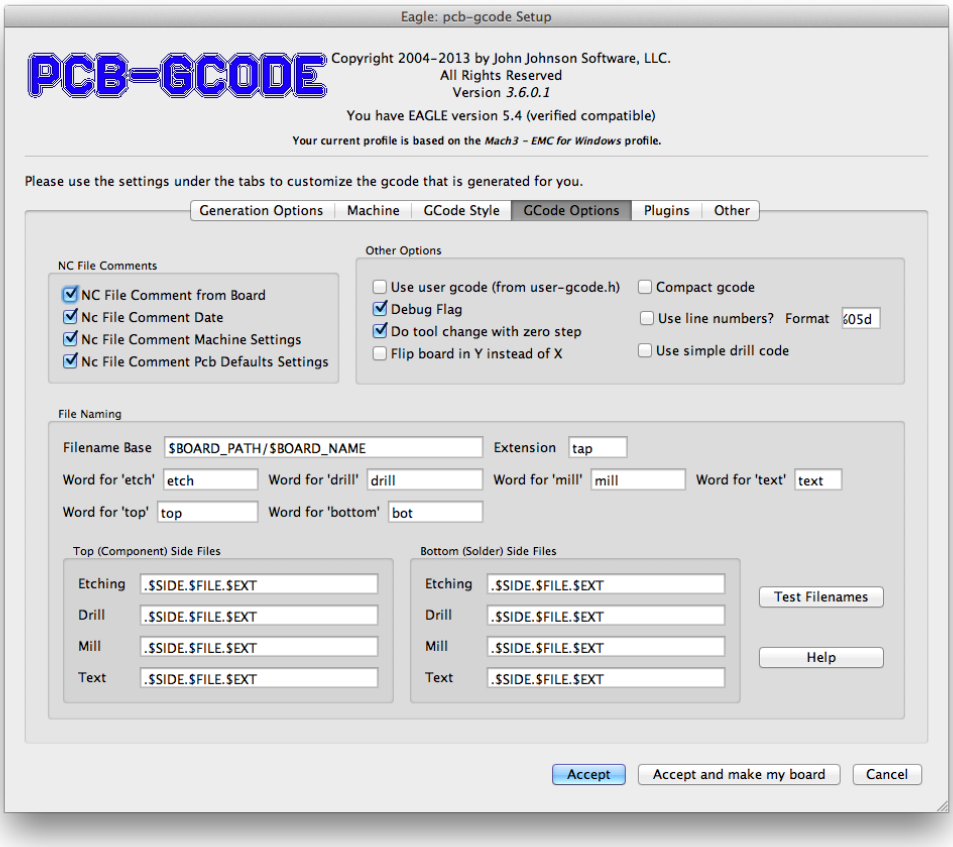


Figure 2.7: *Options for generating g-code files.*

2.5 GCode Options

The options under the GCode Options tab allows the customization of some of the g-code file's content, as well as how the files are named.

NC File Comments Comments added to the g-code file.

NC File Comment from Board adds a comment with the name of the board file.

NC File Comment Date adds the date the g-code file is created.

NC File Comment Machine Settings adds settings related to the machine. Tool size, Z axis settings, spindle on time, milling depth, text depth, tool change position, XY feed rate, Z feed rate.

NC File Comment PCB Defaults Settings adds comments with the isolation settings: min, max, and step size, which files were selected to be produced, and the unit of measure.

Other Options Options affecting how the g-code is generated.

Use user gcode (from user-gcode.h) inserts user g-code into the generated file. See Section 4.4 on page 31 for more details.

Debug Flag sets the global debug flag. Used for development and troubleshooting.

Do tool change with zero step after moving to the tool change position and pausing for the operator to change the bit, the Z axis will move to Z0.000 and pause. This allows the operator to adjust the bit length to touch the surface of the PCB. Note that the tool should initially be set high into the spindle so that it won't dig into the PCB when it moves to Z0.

Flip board in Y instead of X changes the code generated so that after one side of the board is cut, the board should be flipped in the Y axis to complete cutting on the other side. The default is for the board to be flipped in the X axis.

Compact gcode eliminates some redundant commands in the g-code file, such as having G01 on every line.

Use line numbers? inserts line numbers into the g-code file. The format shown `%05d` will insert 5 digit numbers with leading zeroes.

Use simple drill code uses XYZ movements to create drill holes. Usually `DRILL_FIRST_HOLE` and `DRILL_HOLE` are used, but some controllers don't understand the command (typically `G82`).

File naming every option one could want for naming files.

Macros the following can be used in the file name to create the final file name. Note that paths do not include a / at the end. Also note that / is always the path delimiter, even on Windows. Eagle handles the conversion automatically.

$PROJECT_PATH[n] Project paths as set in the Eagle Control Panel. n begins at zero for the first entry.

$ULP_PATH[n] ULP paths as set in the Control Panel.

$CAM_PATH[n] CAM paths as set in the Control Panel.

$BOARD_PATH path to the board file.

$BOARD_NAME the file name of the board file with the extension removed.

$SIDE the side of the board being generated. Defaults are 'bot' and 'top'.

$FILE the file being generated. Defaults are 'etch', 'drill', 'mill' and 'text'.

$EXT the extension set in `Extension` on this screen.

Test Filenames click the button to see how the file names will look.

Help gives a list of the macros defined above and tips on creating file names.

Using the options shown in 2.7, here is how the filename for the top etching file will be created:

- The board path and board name will be used. (`Filename Base`)
- The word for 'top' will be substituted for `$SIDE`. (See `Etching` under `Top (Component) Side Files`.
- The word for 'etch' will be substituted for `$FILE`.
- The `Extension` will be substituted for `$EXT`.

Using examples for the board path and name, the final file name would be:
`/Users/john/Documents/pcbcode/examples/enabtmr.top.etch.tap`

Chapter 3

Using pcb-gcode

3.1 Running pcb-gcode

Run pcb-gcode by selecting File | Run... from EAGLE's board editor. Find the file `pcb-gcode-setup.ulp` and click the Open button. The setup screen will be shown where options can be changed (See Figure 2.6). When the options are correct, click the `Accept and make my board` button. To save the options without generating files for the board, click the `Accept` button.

Pcb-gcode is actually two programs. The first, pcb-gcode-setup, allows for setting options and changing the way NC files are created. The second program is pcb-gcode, which actually creates the files. Most people run pcb-gcode-setup, check their settings, then click the `Accept and make my board` button to generate files. If settings don't need to be changed, pcb-gcode can be run directly and all the file selections and options from the last time pcb-gcode-setup was run will be used.

To make it easier to run pcb-gcode and pcb-gcode-setup, use EAGLE's Assign function to assign a shortcut key to `run pcb-gcode` and `run pcb-gcode-setup` (See Figure 3.1).

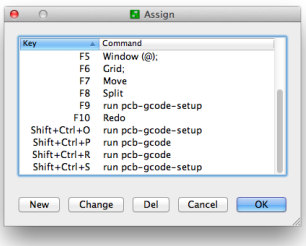


Figure 3.1: *EAGLE shortcut key assignments.*

3.2 Using EAGLE's DRC

When creating files to etch a board, it is usually the case that the tracks should be made wide to allow for easier machining and to account for tolerances in the machine such as backlash and spindle runout. EAGLE's Design Rule Check (DRC) can be used to help ensure that all tracks on the board will be cut out, and no bridges will be left.

A bridge is formed when two parts (tracks, pads, vias) on the board are too close together for the etching tool to pass between them. An example is shown in Figure 3.2. For this example, Single Pass isolation is selected, minimum isolation is set to $0.010''$, and the etching tool size is set to $0.005''$. These setting are shown in Figure 3.3 on the facing page.

If the pads for the leads for the resistors that are close together are more than $0.010'' + 0.010'' + 0.005'' = 0.025''$ inches apart, the pads will be properly isolated as shown in Figure 3.4 on the next page. However, if the two pads are less than $0.025''$ apart, they cannot be isolated, and a bridge will be formed as shown in Figure 3.5 on the facing page.

To help ensure that this does not happen, EAGLE's DRC can be used. From the board editor in EAGLE, click the Tools menu, then click DRC. Click the Clearance tab and set all clearances to 25mil. (A mil is $0.001''$, so this equals $0.025''$.) When the Check button is clicked, any distances less than 0.025" will be marked with a red mark, and a list of errors will be shown as in Figure 3.6 on page 18.

Once the DRC clearances are set up, it is a simple matter to click the DRC button or run a DRC check from the Tools menu before running pcb-gcode to generate files for the board. This provides a good way to help ensure that components on the board are far enough apart to be properly isolated.

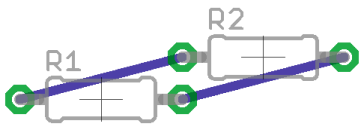


Figure 3.2: *Two components in the board layout editor.*

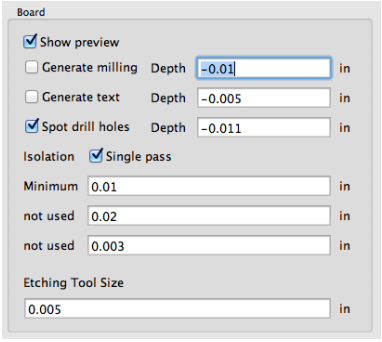


Figure 3.3: *Pcb-gcode settings for the DRC example.*

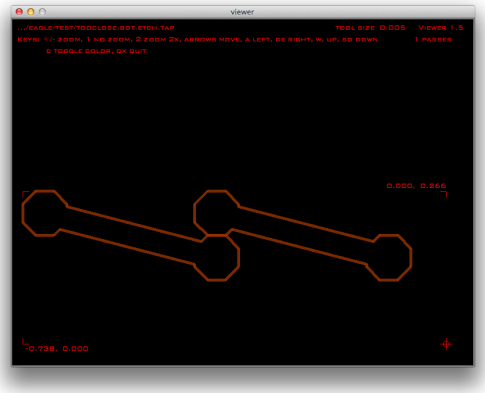


Figure 3.4: *The pads are far enough apart to allow them to be isolated.*

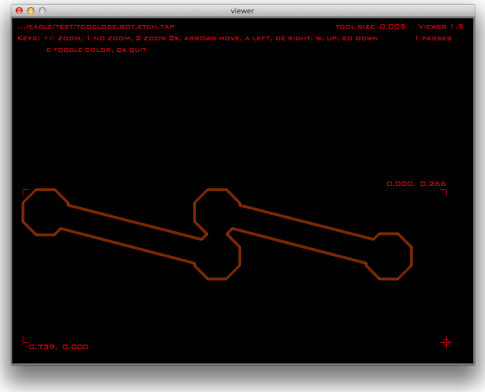


Figure 3.5: *The two pads are too close together and cannot be isolated. A bridge is formed.*

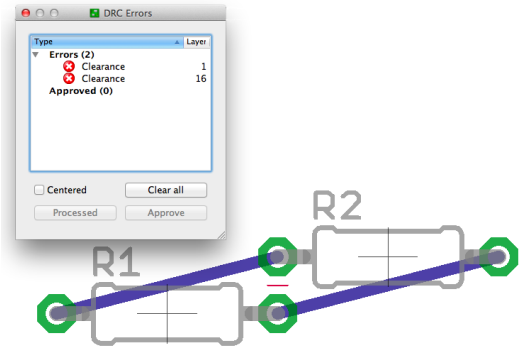


Figure 3.6: *EAGLE's DRC indicating the two pads are too close together.*

3.3 Previewer

The previewer included in pcb-gcode shows a quick preview of the tool movements that are sent to the NC file. The lines are color coded using the standard resistor color codes. The first pass of the tool is drawn in brown, the second is red, the third is orange, etc.

The preview is enabled by turning on the `Show preview` option under the `Generation Options` tab (See Figure 2.6 on page 10). After pcb-gcode creates the isolation or milling for the current layer, a preview of the results will be shown in the previewer. Several keys can be used to change the view, See Table 3.1. If you wonder about some of the unusual keys, such as why `e` can be used to pan right, it is because of the Dvorak keyboard layout that some use, including the author.

Table 3.1: *Keys available in previewer*

Key	Function
1	Set zoom to 1x (no zoom)
2	Set zoom to 2x
+ =	Zoom in
- _	Zoom out
a ←	Pan left
d e →	Pan right
w , ↑	Pan up
s o ↓	Pan down
c	Color on / off
q x	Quit preview

The previewer does not read and interpret the NC files directly, but uses an internal

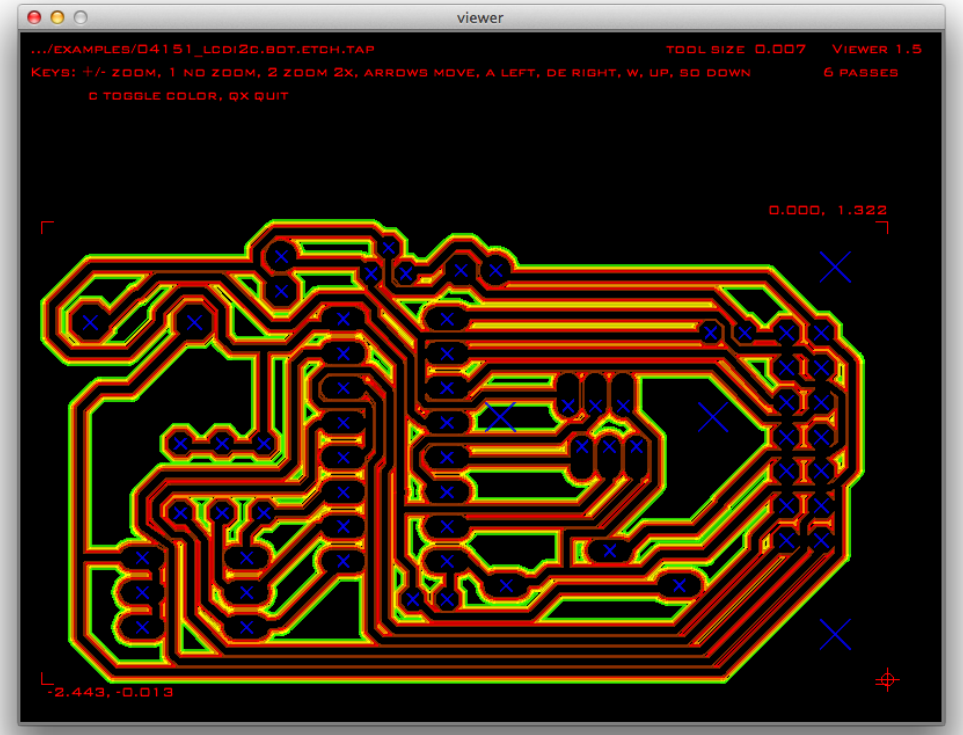


Figure 3.7: *The previewer showing a multi-pass file for the bottom side of an example PCB included with pcb-gcode. Each × represents a drill hole in the drill file.*

representation of the tool movements. This gives an accurate representation of the tool's movements and size, without the overhead of interpreting several different styles of g-code.

As can be seen in Figure 3.7, additional information is provided by the preview. In the upper-left corner is the name of the board file that was processed to produce this preview. At the upper-right, the viewer version number can be found. Just to the left of that is the etching tool size set in pcb-gcode-setup. Just below the version number is the number of passes generated. Back on the left-hand side near the top is an overview of the keys available.

The four extents of the board are marked with red corners. At the upper right corner, the X and Y coordinates are given. The X and Y coordinates are also given at the lower-left corner. Note that in the figure, the X coordinate is negative because this is the bottom of the board. In the lower right corner, a circle and crosshairs mark the origin point for the

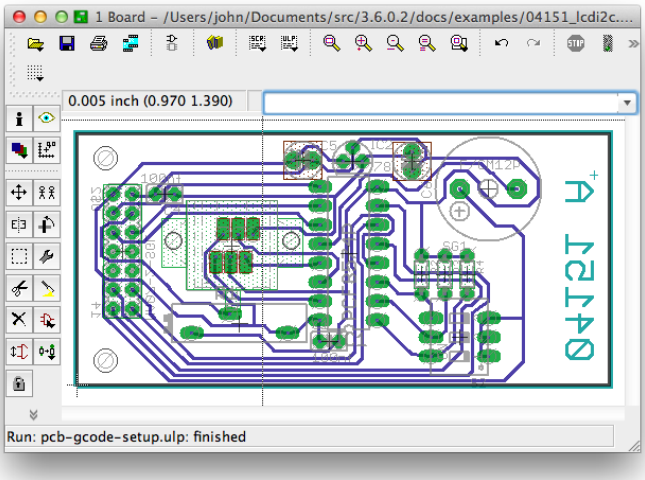


Figure 3.8: *The sample board* `docs/examples/04151_lcdi2c.brd` *with wires and mirrored text on the Milling layer.*

board.

If the milling and text options are turned on, the previewer will show any files generated for the top and bottom milling and text. Since the Milling (46) layer is not a top or bottom layer (it can be thought of as going through the board), NC files are generated to mill any lines drawn on it from either the top or bottom side. So in other words, if `Generate milling` is selected, both top and bottom milling files will be generated. The same for text placed on the Milling layer. If `Generate text` is selected, both top and bottom text engraving files will be generated. There is one caveat: if the text is mirrored, it will be output in the bottom milling file. If the text is not mirrored, it will be output in the top milling file. This is similar to the way text placed on the `Top` and `Bottom` layers looks. The following example will help clarify this.

The example board `docs/examples/04151_lcdi2c.brd` shown in Figure 3.8 includes wires (shown in blue) drawn on the Milling layer, as well as mirrored vector text (also shown in blue) placed on the Milling layer. A prewiew of the bottom milling file is shown in Figure 3.9 on the facing page. A preview of the bottom text engraving file is shown in Figure 3.10 on the next page. You can see that the text is oriented properly for engraving on the bottom side of the board. The object info for the text is shown in Figure 3.11 on page 22.

The top milling file preview looks just like the bottom preview, and is not shown. The top text engraving file was empty since there is no non-mirrored text on the Milling layer, so its preview has also been omitted.

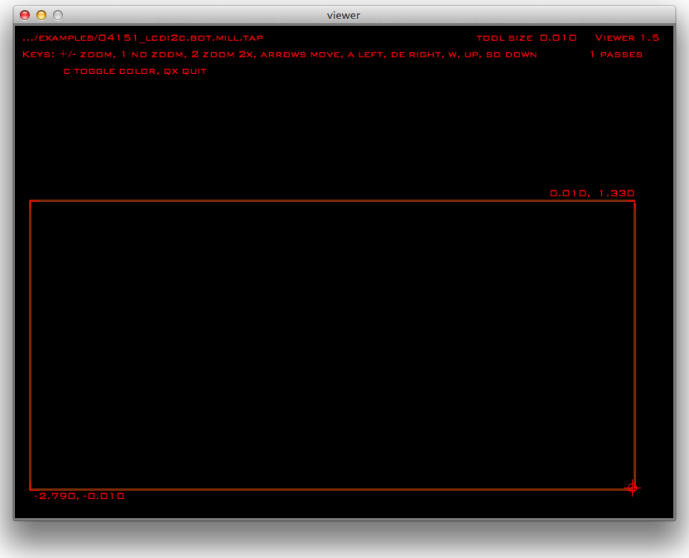


Figure 3.9: *Milling generated for milling from the bottom side of* `docs/examples/04151_lcdi2c.brd`.

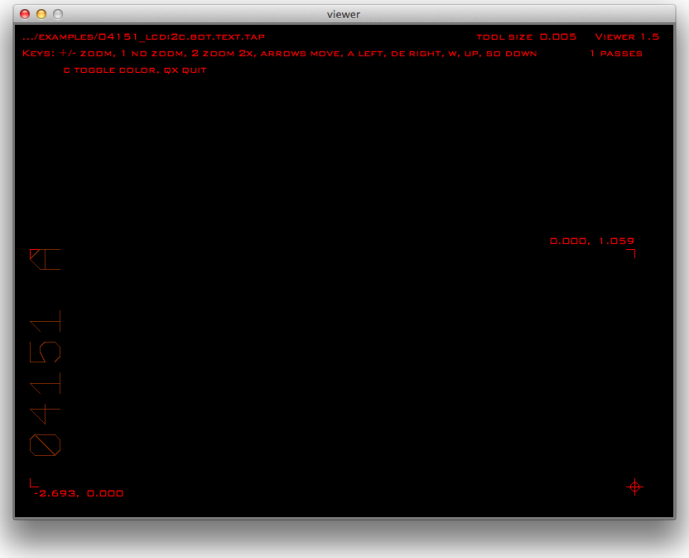


Figure 3.10: *Text engraving generated for cutting on the bottom side of* `docs/examples/04151_lcdi2c.brd`.

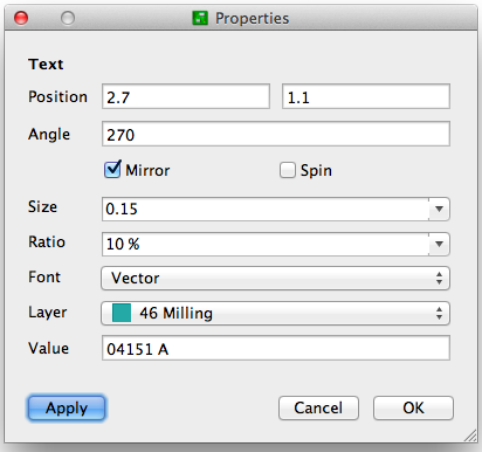
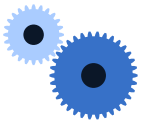


Figure 3.11: *The object info for the mirrored text on the Milling layer. Note that the Mirror checkbox is on. This indicates to pcb-gcode that the text should be engraved on the bottom side of the board.*



For the curious reader, the previewer is written using a language called Processing, which is somewhat Java-like. Three versions of the previewer are included in pcb-gcode, one for each operating system supported: Mac OS X, Linux, and Windows. When the previewer is enabled, pcb-gcode detects which operating system it is running on, and runs the appropriate viewer.

3.4 Saving and Loading Settings

3.4.1 Overview

Settings may be loaded and saved to settings files. This can be found on the Plugins tab, and is shown in figure 3.12 on the next page. This allows the operator to set up groups of settings for different PC boards, save them, and load them when manufacturing the same board later. The settings saved include those on the Generation Options tab (shown in figure 2.6 on page 10), as well as those on the Machine tab (see figure 2.5 on page 9) except for the Units setting.

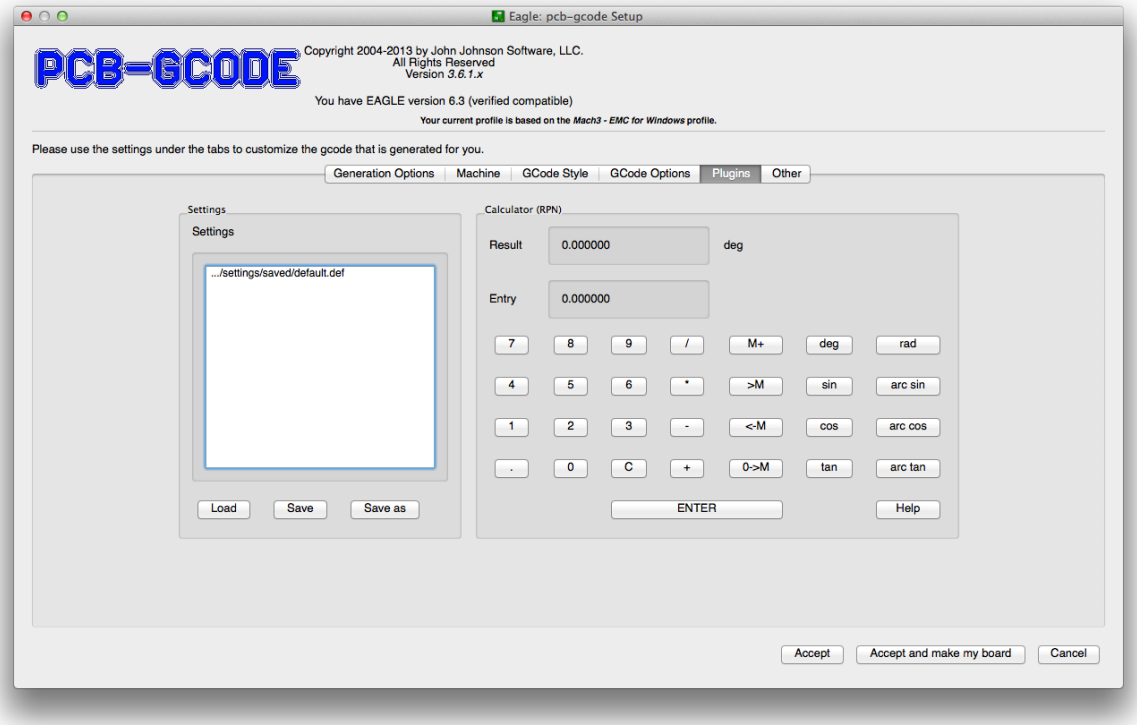


Figure 3.12: *The Plugins tab showing where settings can be saved and loaded.*

3.4.2 Saving Settings

First, set up all settings on the Generation Options tab, and the Machine tab. When the settings are working as desired, select the Plugins tab, then click the `Save as` button. A standard file save dialog will appear where you can enter a file name. A confirmation dialog will appear telling you your settings have been saved, then setup will reload.

If you would like to replace a set of settings that have already been saved, click the filename in the list of Settings, then click the Save button. A dialog will open confirming that your settings have been saved.

Eagle Bug

Note that settings must be saved in the `<pcb-gcode path>/settings/saved` **folder.** There is a bug on some platforms that prevents pcb-gcode from setting the proper directory for the save dialog. Please ensure the directory is set correctly, otherwise your saved files will not appear in the list of Settings.

3.4.3 Loading Settings

To load a set of settings you have saved before, go to the Plugins tab, select the file from the Settings list, and click the Load button. The settings will be loaded, and the setup program will reload.

Intermediate

The settings from the Generation Options tab are saved in `pcb-defaults.h`, and the settings from the Machine tab are saved in `pcb-machine.h`. When the settings are saved from the Plugins tab, they are copied to `filename.def` and `filename.mac` files. When loading, the opposite operation takes place.

Chapter 4

Customizing

4.1 G-Code

When the g-code style is initially selected at installation as discussed in Section 2.2.3 on page 6, the profile (`.pp`) file selected is copied from the `profiles` folder to the `settings` folder and to the file `gcode-defaults.h`. This is the file that pcb-gcode uses when generating g-code files.

Intermediate

The listing found in Appendix A on page 35 can alse be used as a reference for editing `gcode-defaults.h`. If there is a need to change the g-code generated by pcb-gcode, the gcode-defaults.h file can be edited.

The definitions in the file are flexible in some aspects, and restricted in others. The main restriction is that a definition that expects a certain number of parameters must be given that number of parameters. Unused parameters can be passed as comments in the g-code.

For example, if the controller does not understand the DWELL command, it can be changed to a comment. In the listing, DWELL is defined to be:

```
46 string DWELL    = "G04 " + PARAM + "%f" + EOL;
```

This can be made into a comment the controller will ignore by surrounding it with COMMENT_BEGIN and COMMENT_END:

```
46 string DWELL    = COMMENT_BEGIN + "G04 " + PARAM + "%f" + COMMENT_END
     + EOL;
```

The `%f` that pcb-gcode needs is still there, but the command is now just a comment as far as the controller is concerned.

The definitions use previous definitions as much as possible. This helps make the files more readable, and makes future changes easier as well. For instance, `EOL` is defined in line 22 to be `"\n"`. Changing the line ending of all generated code would be a simple matter of changing the single definition of `EOL`, rather than editing every line.

This is only an example. EAGLE handles line endings automatically depending on the operating system it is running on. If line endings in generated NC files need to be changed, a conversion program should be used.

The first option on line 11, `FILENAMES_8_CHARACTERS`, tells pcb-gcode whether it should limit filenames to 8 characters. This is used for DOS control software such as TurboCNC.

Misc defines Comments and line ending

COMMENT_BEGIN, COMMENT_END If your controller doesn't understand beginning and ending characters for comments, as on lines 16-17, just make `COMMENT_END` empty.

EOL is the character added to the end of every line.

Formats Parameter formats.

PARAM different controllers use different characters to introduce a parameter. Mach3 uses `P`, while others use `#`.

FORMAT is the floating point format used for coordinates. The default value `%-7.4f` means a leading negative sign will be output for negative coordinates (very important), the number will be 7 digits long, and 4 digits will be to the right of the decimal point. The `f` indicates it is a floating point (real) number.

FR_FORMAT is the format used to insert feedrate parameters into the g-code. In the example, the leading `F` indicates this is a feedrate parameter. The rest of the format is similar to `FORMAT` — leading negative sign possible, 5 digits wide, no digits to the right of the decimal point.

IJ_FORMAT is used to output `I J` coordinates to the g-code file. You can see by the definition on line 26 that this format reuses the `FORMAT` definition defined earlier.

R_FORMAT is used by commands that require an `R` parameter, usually the radius for an arc command.

Modes Inch, metric, etc. modes

INCH_MODE used to set the controller to inch mode.

INCH_MODE_COMMENT a comment inserted in the g-code indicating that inch mode is being set.

METRIC_MODE used to set the controller to Metric mode.

METRIC|_MODE_COMMENT a comment inserted in the g-code indicating that metric mode is being set.

MIL_MODE used to set the controller to mil mode. Currently undefined in all profiles.

MICRON_MODE used to set the controller to micron mode. Currently undefined in all profiles.

ABSOLUTE_MODE would be used to set the controller to absolute coordinates mode. Currently just a comment.

G Codes Basic g-code defines for movements.

RAPID for rapid moves with the cutting tool out of the material.

FEED for movements with the cutting tool in the material.

ARC_CW for cutting an arc clockwise.

ARC_CCW for cutting an arc counter-clockwise.

DWELL pause for a number of seconds. Number of seconds (a float) is passed.

M Codes M-code definitions.

SPINDLE_ON turns the spindle on. Takes `DWELL` as a parameter.

SPINDLE_OFF turns the spindle off.

END_PROGRAM signals the end of the gcode program.

OPERATOR_PAUSE pauses for the operator to do something, like change the tool.

Coordinates Definitions for coordinate parameters.

MOVE_X X axis movement. Passed an `X` coordinate as a parameter.

MOVE_Y Y axis movement. Passed a `Y` coordinate as a parameter.

MOVE_XY XY axis movement. Passed an `X` and `Y` coordinate as parameters.

MOVE_Z Z axis movement. Passed a `Z` coordinate as a parameter.

MOVE_XYZ XYZ axis movement. Passed `X Y Z` coordinates as parameters.

Rapids Combinations of `RAPID` and the above coordinates.

RAPID_MOVE_X rapid X axis movement. Passed an `X` coordinate as a parameter.

RAPID_MOVE_Y rapid Y axis movement. Passed a `Y` coordinate as a parameter.

RAPID_MOVE_XY rapid XY axis movement. Passed `X Y` coordinates as parameters.

RAPID_MOVE_XY_HOME rapid XY axis movement to `X0 Y0`.

RAPID_MOVE_Z rapid Z axis movement. Passed a `Z` coordinate as a parameter.

RAPID_MOVE_XYZ rapid XYZ axis movement. Passed `X Y Z` coordinates as parameters.

Feeds Movements at cutting speed, uses `FEED` and the coordinate definitions above.

FEED_MOVE_X feed X axis movement. Passed an `X` coordinate as a parameter.

FEED_MOVE_Y feed Y axis movement. Passed a `Y` coordinate as a parameter.

FEED_MOVE_XY feed XY axis movement. Passed `X Y` coordinates as parameters.

FEED_MOVE_XY_WITH_RATE feed XY axis movement. Passed `X Y` coordinates and the feed rate.

FEED_MOVE_Z feed Z axis movement. Passed a `Z` coordinate as a parameter.

FEED_MOVE_Z_WITH_RATE feed Z axis movement. Passed `Z` coordinate and a feed rate as parameters.

FEED_MOVE_XYZ feed XYZ axis movement. Passed an `X Y Z` coordinates as parameters.

Drilling holes Definitions for drilling holes.

DRILL_CODE the gcode instruction to drill a hole.

RELEASE_PLANE the Z position to move the drill to after drilling. Takes a `Z` coordinate as a parameter.

DWELL_TIME the time to dwell in the bottom of the hole. Takes a floating point (real) argument.

DRILL_FIRST_HOLE generated to drill the first hole. Takes `X Y Z`, feed rate, release `Z` plane and dwell time as parameters.

DRILL_HOLE generated for subsequent holes. Takes `X Y` as parameters.

Tool change Definitions for changing tools.

TOOL_CODE the tool selection code. Passed the tool number (an integer) as a parameter.

TOOL_MM_FORMAT a tool size formatted for millimeters. Passed a tool size (float).

TOOL_INCH_FORMAT a tool size formatted for inches. Passed a tool size (float).

TOOL_CHANGE the command issued when a tool is to be changed. Takes tool number and tool size as arguments.

TOOL_CHANGE_TABLE_HEADER the tool table header comment inserted in the g-code.

TOOL_CHANGE_TABLE_FORMAT generates an entry for the tool table. Note that this is a function.

Circles / Arcs Arc and circle commands.

ARC_TOP is an arc on the top of the board. Takes `X Y R` as parameters.

ARC_BOTTOM an arc on the bottom of the board. Takes `X Y R` as parameters.

4.2 Profiles

Advanced

Profiles, which are found in the `profiles` folder, control the format that pcb-gcode uses when writing g-code. The files ending in `.pp` are the list of files shown when pcb-gcode is initially set up (See Section 2.2 on page 5), and also in the list of profiles in the `GCode Styles` tab. When a profile is selected, it is copied to `settings/gcode-defaults.h`. A sample profile is shown in Appendix A on page 35.

To create a custom profile, such as for a controller that is not already supported, begin with a profile that most closely matches the g-code the controller supports. Select this profile in `GCode Styles` and `Accept` the change. This will copy the profile to `settings/gcode-defaults.h`. Generate code for a test board and open the generated files in an editor. Find commands that the controller does not support, and edit `gcode-defaults.h` to generate the proper code. When testing is complete, copy `settings/gcode-defaults.h` into the `profiles` folder, renaming it with a descriptive name and the extension `.pp`. Edit the file and change the author and description fields. And of course, save a backup somewhere outside the pcb-gcode folder heirarchy. To share this profile with other users of this controller, upload the profile file to the Profiles folder on the Yahoo! group.

For information on editing `gcode-defaults.h`, See Section 4.1 on page 25.

4.3 Drill Rack Files

Rack files allow the substitution of one drill size for a range of sizes that may be found in the board. For instance, a 0.031″ drill might be used for hole sizes 0.025″ – 0.032″. This cuts down on the number of drills that must be kept on hand, and the number of tool changes needed to drill a board. An example drill rack file is shown in Listing 4.1. **Please heed the warning about using a tab character** between entries on a line. Otherwise, your rack file will not work.

As can be seen, drills with different units of measure are supported. This includes inches, millimeters, mils, or wire gage sizes. See Listing 4.2 for examples that work. The algorithm tries to be intelligent and assumes, for example, that 0.1 is in inches, whereas 0.6 is in millimeters. To be safe, add the unit of measure after the number.

Looking at Listing 4.1, the meaning of the fields are as follows:

tool The tool number to use. This is somewhat arbitrary unless the machine has a tool changer or you have offsets or lengths set up for different tool numbers. Tool numbers can begin at any reasonable number, such as T01, T16, etc.

drill_size The size of the actual drill. These should be in ascending order from smallest to largest.

minimum The smallest hole size this drill should be used for.

maximum The largest hole this drill should be used for.

length Currently not used. Leave set to 1.5in.

Taking tool `T01` as an example. It is a 0.500mm drill, and it will be used for all holes from `0.000in` up to `0.025in`. Meaning, if there is a 0.015″ hole in the board, it will be drilled with this bit. If there is a 0.045″ hole, it will not be drilled with drill `T01`, but another drill will be used, if a good match is available.

Looking at the table, it can be seen that all hole sizes from 0.000in up to 0.125in have been accounted for. If, say, a 0.130″ hole is in the board, an error message will be given saying a drill is not available for that size hole.

Rack file are selected by the following method: first, if there is a rack file with the same name as the board, but with the extension `.drl`, it will be used. Next, a default rack file will be used if it has been set in `pcb-gcode-setup` (See Figure 2.5 on page 9). Finally, if neither of those is available, the rack file `settings/default.drl` will be used from the `pcb-gcode` directory. If all these attempts fail and no rack file can be found, a table of drill sizes will be written to the drill file. In most cases this works well, but sometimes it can result in, for instance, drilling ten holes with a 0.031″, then stopping and asking for a 0.032″ bit. Obviously, the same bit could have been used for both sets of holes. That is why rack files exist.

Listing 4.1: *Sample Rack File*

```
1  #
2  # Please note that you must use a TAB character
3  # between each setting on a line.
4  #
5  # Tip: Set the TAB size of your editor to 12 characters.
6  #
7  tool    drill_size      minimum maximum length
8  T01     0.500mm         0.000in 0.025in 1.5in
9  T02     0.032in         0.025in 0.035in 1.5in
10 T03     0.040in         0.035in 0.045in 1.5in
11 T04     0.050in         0.045in 0.055in 1.5in
12 T05     0.062in         0.055in 0.070in 1.5in
13 T06     0.085in         0.070in 0.125in 1.5in
```

Listing 4.2: *Sample entries for rack files*

```
0.032in    0.032 inches
62ml       62 mils, 0.062 inches
0.43mm     0.43 millimeters
1500mc     1500 microns, 1.500 millimeters
60#        60 wire gage drill (0.040'' or 1.016mm)
0.12       0.12 inches
0.60       0.60 millimeters
43         43 wire gage drill
```

4.4 User GCode

Advanced

The pcb-gcode ULP allows you to customize the g-code created for your boards to a great degree. If you don't see an option in the profile that suits your needs, you can add your code to the `user-gcode.h` file. To enable user g-code, run `pcb-gcode-setup`, click the `GCode Options` tab, then turn the `Use user gcode...` option on. Generate a set of NC files for a board. Let's say, for example, that after you change the tool when you're drilling from the bottom of the board, you want the tool to move to X5 Y5 Z5, turn the spindle off, then turn it back on. Since this has to do with drilling the bottom of the board, we should look at the ...bot.drill.tap (bottom drill) file. An excerpt from a file is shown in Listing 4.3.

Listing 4.3: *Bottom drill file before adding user g-code.*

```
G90
(Tool Change Begin)
(Bottom Tool Change Begin)
M05
G00 X0.0000 Y0.0000 Z2.0000
M06 T01  ; 0.0236
(Bottom Tool changed)
(Tool changed)
G00 Z0.0200
M03
G04 P3.000000
(Bottom Tool Change End)
(Tool Change End)
G82 X-1.6200 Y1.2900 Z-0.1000 F9.80  R0.0200 P0.250000
G82 X-1.8800 Y0.5900
G82 X-1.9500 Y1.4900
G82 X-1.9500 Y1.8100
```

We want to add our commands after the tool is changed when drilling the bottom of the board. Looking at the sample above, you will find this line:

```
(Bottom Tool Change End)
```

That's where we want our code to go. Now you can open user-gcode.h in your favorite editor, and use the Search or Find feature to find the line with Bottom Tool Change End. Here's an excerpt from the user-gcode.h file:

```
1 TOOL_ZERO_BEGIN[BOTTOM] = "(Bottom Tool zero begin)\n";
2 TOOL_ZERO_END[BOTTOM]   = "(Bottom Tool zero end)\n";
3 TOOL_CHANGE_END[BOTTOM] = "(Bottom Tool Change End)\n";
4 TOOL_CHANGE_BEGIN[TOP]  = "(Top Tool Change Begin)\n";
```

The 3^{rd} line is the one we're interested in:

```
3 TOOL_CHANGE_END[BOTTOM]   = "(Bottom Tool Change End)\n";
```

Change the line so that it looks like this:

```
3 TOOL_CHANGE_END[BOTTOM] = "(Bottom Tool Change End)\n"
4                           "G00 X5 Y5 Z5\n"
5                           "M05 (spindle off)\n"
6                           "G04 P3.000000 (wait 3 seconds)\n"
7                           "M03 (spindle on)\n";
8                           "G04 P3.000000 (wait 3 more seconds\n";
```

Notice that all the lines have a `\n` before the last `"` and that the last line is the only one that ends with a semi-colon `;`. Generate the files again. Open the bottom drill file in the editor and have a look. Here's how the sample looks now:

```
1  G90
2  (Tool Change Begin)
3  (Bottom Tool Change Begin)
4  M05
5  G00 X0.0000 Y0.0000 Z2.0000
6  M06 T01  ; 0.0236
7  (Bottom Tool changed)
8  (Tool changed)
9  G00 Z0.0200
10 M03
11 G04 P3.000000
12 (Bottom Tool Change End)
13 G00 X5 Y5 Z5
14 M05 (spindle off)
15 G04 P3.000000 (wait 3 seconds)
16 M03 (spindle on)
17 G04 P3.000000 (wait 3 more seconds)
18 (Tool Change End)
19 G82 X-1.6200 Y1.2900 Z-0.1000 F9.80  R0.0200 P0.250000
20 G82 X-1.8800 Y0.5900
21 G82 X-1.9500 Y1.4900
22 G82 X-1.9500 Y1.8100
```

Note that the lines added to the user-gcode.h file are now in the generated g-code from lines 13-17.

Since we put our code in the `TOOL_CHANGE_END[BOTTOM]` definition, it will only be put in files for the bottom side of the board. So the code will be in the bot.drill file. If we only wanted our code in files for the top side, we would have put the code in `TOOL_CHANGE_END[TOP]`. You can probably guess that if we wanted the code in both the top and bottom files, we would have put the code in `TOOL_CHANGE_END[TOOL_CHANGE_END[ALL]`. To conclude, the steps to follow are:

1. Generate a set of files.
2. Find the file that you want the code to be put in (bot, top, etc.).
3. Find the location in the file that you want the code.
4. Find the comment near that location.
5. Find the comment in the user-gcode.h file.
6. Insert your code after the comment.
7. Generate the files again and check to be sure it is correct.

Appendix A

Sample Mach3 Profile

```
//
// Options for pcb-gcode.ulp.
// Often used options are at the top of the file.
// Copied to gcode-defaults.h by the setup program.

//
// author=John Johnson
// description=Mach3 - EMC for Windows
//

int FILENAMES_8_CHARACTERS = NO;

//
// Comments.
//
string COMMENT_BEGIN = "(";
string COMMENT_END   = ")";

//
// Format strings for coordinates, etc.
//
string EOL        = "\n";             /* standard line ending */
string PARAM      = "P";              /* some use P, some # for parameters */
string FORMAT     = "%-7.4f ";        /* coordinate format */
string FR_FORMAT  = "F%-5.0f ";       /* feedrate format */
string IJ_FORMAT  = "I" + FORMAT + "J" + FORMAT;
string R_FORMAT   = "R" + FORMAT;

//
// Modes
//
string INCH_MODE            = "G20" + EOL;
string INCH_MODE_COMMENT    = COMMENT_BEGIN + "Inch Mode" + COMMENT_END + EOL;
string METRIC_MODE          = "G21" + EOL;
string METRIC_MODE_COMMENT  = COMMENT_BEGIN + "Metric Mode" + COMMENT_END + EOL;
string MIL_MODE             = "M02 (Please setup MIL_MODE in gcode-defaults.h)" + EOL;
string MICRON_MODE          = "M02 (Please setup MICRON_MODE in gcode-defaults.h)" + EOL;
string ABSOLUTE_MODE        = COMMENT_BEGIN + "Absolute Coordinates" + COMMENT_END + EOL + "G90" + EOL;

//
// G codes
//
string RAPID   = "G00 ";
string FEED    = "G01 ";
string ARC_CW  = "G02 ";
string ARC_CCW = "G03 ";
string DWELL   = "G04 " + PARAM + "%f" + EOL;

//
// M codes
//
string SPINDLE_ON  = "M03" + EOL + DWELL;
string SPINDLE_OFF = "M05" + EOL;
```

```
54  string END_PROGRAM    = "M02" + EOL;
55  string OPERATOR_PAUSE = "M06 ";
56
57  //
58  // Coordinates
59  //
60  string MOVE_X   = "X" + FORMAT;
61  string MOVE_Y   = "Y" + FORMAT;
62  string MOVE_XY  = "X" + FORMAT + "Y" + FORMAT;
63  string MOVE_Z   = "Z" + FORMAT;
64  string MOVE_XYZ = MOVE_XY + MOVE_Z;
65
66  //
67  // Rapids
68  //
69  string RAPID_MOVE_X       = RAPID + MOVE_X;
70  string RAPID_MOVE_Y       = RAPID + MOVE_Y;
71  string RAPID_MOVE_XY      = RAPID + MOVE_XY;
72  string RAPID_MOVE_XY_HOME = RAPID + "X0 Y0";
73  string RAPID_MOVE_Z       = RAPID + MOVE_Z;
74  string RAPID_MOVE_XYZ     = RAPID + MOVE_XYZ;
75
76  //
77  // Feeds
78  //
79  string FEED_MOVE_X            = FEED + MOVE_X;
80  string FEED_MOVE_Y            = FEED + MOVE_Y;
81  string FEED_MOVE_XY           = FEED + MOVE_XY;
82  string FEED_MOVE_XY_WITH_RATE = FEED + MOVE_XY + FR_FORMAT;
83  string FEED_MOVE_Z            = FEED + MOVE_Z;
84  string FEED_MOVE_Z_WITH_RATE  = FEED + MOVE_Z   + FR_FORMAT;
85  string FEED_MOVE_XYZ          = FEED + MOVE_XYZ;
86
87  //
88  // Drilling holes
89  //
90  // G82 Xx.xxx Yy.yyy Z.zzz Fff.f Rr.rrr #dwell
91  //
92  string DRILL_CODE       = "G82 ";
93  string RELEASE_PLANE    = "R" + FORMAT;
94  string DWELL_TIME       = PARAM + "%f";
95  string DRILL_FIRST_HOLE = DRILL_CODE + MOVE_XYZ + FR_FORMAT + RELEASE_PLANE + DWELL_TIME + EOL;
96  string DRILL_HOLE       = DRILL_CODE + MOVE_XY + EOL;
97
98  //
99  // Tool change
100 //
101 string TOOL_CODE        = "T%02d ";
102 string TOOL_MM_FORMAT   = "%1.3fmm";
103 string TOOL_INCH_FORMAT = "%1.4fin";
104 string TOOL_CHANGE      = OPERATOR_PAUSE + TOOL_CODE + " ; " + FORMAT + EOL;
105
106 string TOOL_CHANGE_TABLE_HEADER = COMMENT_BEGIN +
107   " Tool|       Size       |  Min Sub |  Max Sub |   Count " + COMMENT_END + EOL;
108
109 string TOOL_CHANGE_TABLE_FORMAT(int tool_number, real size_mm, real size_inch, real min_drill, real max_drill,
        int count)
110 {
111   string formatted;
112
113   sprintf(formatted, COMMENT_BEGIN + " " +
114     TOOL_CODE + "| " + TOOL_MM_FORMAT + " " +
115     TOOL_INCH_FORMAT + " | " + TOOL_INCH_FORMAT + " | " +
116     TOOL_INCH_FORMAT + " | " +
117     "   %4d" + " " +
118     COMMENT_END + EOL,
119     tool_number, size_mm, size_inch, min_drill, max_drill, count);
120   return(formatted);
121 }
122
123 //
124 // Circles / Arcs
125 //
126 string ARC_CLOCK     = ARC_CW + MOVE_XY + R_FORMAT + EOL;
127 string ARC_CCLOCK  = ARC_CCW + MOVE_XY + R_FORMAT + EOL;
```

Index

download site, 5
Drill Depth, 8
Drill Dwell, 8
drill files, 29

eagle
    compatibility, 5
    DRC, 16

Feed Rates, 9

g-code
    customizing, 25
g-code style, 6
gcode-defaults.h
    definitions of commands, 26

installation, 5

Java, 8
javaw.exe, 8

keys
    previewer, 18

pcb-gcode
    running, 15
previewer, 18
    keys, 18
profiles, 29
    definitions of commands, 26

rack files, 29

settings
    loading, 24
    overview, 22
    path to, 24
    saving, 22
Spin Up Time, 8

Tool Change, 8

unit of measure, 8
user gcode, 31

Windows File Protection, 5

Z Down, 8
Z High, 8
Z Up, 8