

Shapeoko CNC A to Z

Introduction

v4 - May 2020

The intent of this guide is to help new users of the Shapeoko CNC learn enough about both the big picture and the underlying technical details, to feel at ease with the machine, the workflow, and CNC lingo in general.

As fun as it is to go ahead and cut a first project following a tutorial (and you should), having a good understanding of how the machine works and what underlying CNC concepts are involved goes a long way to avoid frustration later, when things do not work out the way you expected (and sooner or later they will).

There are many ways to get started. Most people want to be up and running and making things as quickly as possible, without the burden of learning all the nitty-gritties, and that's fine. At the other end of the spectrum, some users (including yours truly) enjoy CNC and using the Shapeoko as a hobby, and end up spending more time experimenting with the machine than actually cutting useful projects, and that's fine too. Hopefully there is a little something for everyone in this guide.

This is a personal initiative (call it a fan project), not affiliated with the Carbide 3D company in any way, yet positively biased (it's such a great machine, what can I do...)

Safety considerations

This is not (only) a legal disclaimer, this is plain old common sense but needs to be said: using a CNC as a hobbyist comes with a small number of specific risks for your safety, that are easily addressed.

Mechanical hazard

- keep your hands away from the moving parts during operation, cleaning up the mess on the belts/pulleys can probably wait until the end of the job.

- get yourself a way to do an emergency stop, within arm's reach. A killswitch removing ALL power (to the machine *and* to the router) is easy to install.
- turn off the machine before diving in to check/adjust/fix something. Sure, the machine is "not supposed" to move by itself, but user errors happen (ever clicked anywhere by mistake?) and software has bugs. Changing an endmill while the machine is powered is probably the only sensible exception (and unplugging the router while doing so is a best practice).
- wear protection goggles anytime the machine is operating without an enclosure, no exceptions. Endmills break from time to time, and shrapnel may fly out in random directions when they do.

Respiratory hazard

- cutting some materials (e.g. MDF or exotic wood) produces dust that can be very toxic, and wood dust is just bad for your health in general.
- a good dust collection and filtering system will take care of this risk. You'll find out soon enough if you don't use one (and/or an enclosure) that a thin layer of dust ends up covering everything around the machine, so imagine what it does to your lungs. If you need a good scare, go read about wood dust related health issues, I bet you will invest in dust collection and filtering promptly afterwards.

Fire hazard

- while this risk is very remote, it does exist.
 - don't leave your machine (completely) unattended while it is cutting. The stock material may become loose, and rubbing a piece of metal spinning at thousands of RPM against wood is a potential way to start a fire. Google "CNC" + "fire" if you need proof.
-

Overview

There are many ways to learn and use the Shapeoko, but the typical experience for a new user can go something like this:

- order the machine (after reading a lot of good things about it)

- while waiting for delivery, get familiar with CNC concepts:
 - this is the intent of the first sections ([CNC workflow](#), [Anatomy of a Shapeoko](#), [CAD, CAM, and G-code](#), [Cutters & collets](#), [Feeds & speeds](#), [Toolpaths](#), [Workholding](#)). They go in way more detail than is strictly necessary to get started, so if it feels overwhelming just skip the details and get cutting.
 - the [Cutters & collets](#) section includes a few indications as to what starter set of endmills and collets could be suitable for you to purchase, depending on the projects you have in mind.
- receive and assemble the machine, and run the Hello World example with a marker. Oh the joy of these first few hours! I chose *not* to have a section about machine assembly, because Carbide 3D's docs are quite detailed already, and there are many things that are either specific to the machine type (standard vs XL vs XXL) or bound to change frequently and that would render this information obsolete very quickly. Carbide 3D's email support and the community forum are the best way to sort out any specific machine assembly issue.
- run your first cuts in wood or MDF (this is covered in the [Running a job](#) section).
- realize that:
 - CNC is messy, and that now is a good time to look into dust collection and possibly an enclosure. This is covered in the [Shapeoko setup](#) section.
 - The geometry of cut pieces or finish quality is not quite right, and that it may be necessary to look into [Squaring, surfacing, tramping](#) the machine, and possibly [Dimensional accuracy](#).
- get confused about required feeds and speeds for that particular case you want to use but for which no recommendation exists: hopefully re-reading the [Feeds & speeds](#) section should help.
- try cutting plastics or metal (see [Usecases: cutting plastics](#) , [Usecases: cutting metal](#))
- at some point, witness the machine misbehave, blame the hardware or the software, start investigating, and (often) end up concluding that you messed up somewhere in the workflow. The [Troubleshooting & maintenance](#) section has a few pointers, after that the user community and Carbide 3D's support team are your best bet.
- somewhere along this path, outgrow Carbide Create and Carbide Motion and consider moving to more advanced [CAD, CAM, G-code](#) tools.
- after a while, begin to feel an itch for [HW upgrades](#).

Feedback & credits

Comments/corrections/contributions are most welcome, you can contact me on the Shapeoko community forum (<https://community.carbide3d.com/>) as @Julien.

Shoutout to the whole forum community, the collective knowledge and experience accumulated over there and on the Shapeoko wiki (https://wiki.shapeoko.com/index.php/Shapeoko_3) is incredible. I tried to bring out *some* of that know-how in this e-book, but it is also tainted by my own limited experience and habits, so any potential incorrect information/guidance is my bad.

Special thanks to :

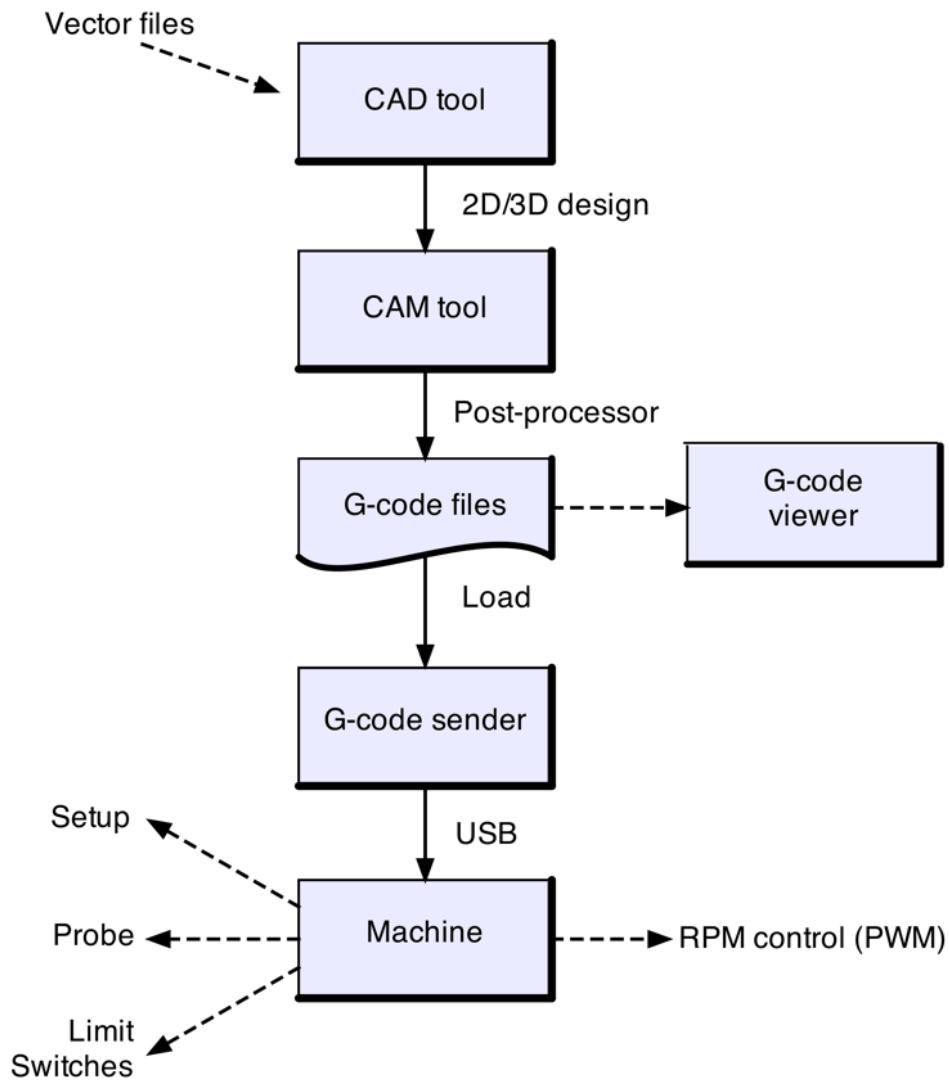
- **@patonclover** for the digital versions of all illustrations, which are a very welcome replacement for my initial (poor) hand-drawn doodles
- **@WillAdams** and **@luc.onthego** for proofreading and suggesting lots of good improvement ideas
- **@snaterst** (on the forum) for providing the vise setup picture
- **@Dusty.Tools** (find him on Instagram) for the T-tracks setup picture
- **@Griff** for the spindle cooling system pictures
- **@stutaylo** and **@PaulAlfaro** for the bullnose endmill pics
- **@bikerdan** for the Z-plus pic
- **@Todd** for the Sweepy pic
- **@i3oilermaker** for the v-carved inlay pics
- **@gmack** for his excellent feeds and speeds worksheet
- **@Hooby** for the nice Janka wood hardness database included in the worksheets
- **@neilferreri** for his awesome CNCjs macros, Fusion360 post processor, and other goodness.

This e-book is released under the Creative Commons CC BY-NC-SA license (in plain English: do whatever you want with it except using it for commercial purposes, and if you modify & redistribute, give credit and keep the same license).



CNC workflow

First things first, the **workflow** of a typical CNC job:



Everything starts in a **CAD** (Computer-Aided Design) program: this is where you will create the 2D or 3D objects to be machined. CAD software packages are usually able to import 2D and 3D features from a variety of file formats, and the most common/useful ones for CNC are "vector" formats.

- ⓘ **Carbide Create**, the CAD program provided by Carbide 3D for the Shapeoko, can import SVG or DXF vector files (and the Pro version has additional capabilities)

Once the object is designed, a **CAM** (Computer-Aided Manufacturing) module that is usually integrated in the CAD suite, is used to create **toolpaths** to cut the object out of a block of stock material (more on this later). Once all required toolpaths are created, the very last step in the CAM program is to generate one or several **G-code** files, containing instructions for the machine to move the cutter along these toolpaths.

- ⓘ G-code format is a standard (originally ISO 6983-1 back in the 80s) so one would expect that a G-code file can be run on any CNC. Well almost, but not quite. Different CNCs support different subsets of the G-code instructions, as well as implement their own custom instructions.

Since CAM programs are usually not bound to any specific CNC machine, they make use of a specific **post-processor** to generate the correct G-code for a given machine.

- ⓘ In **Carbide Create**, there is a single G-code post-processor that gets executed behind the scenes, and it knows what Shapeoko model you have since there is a dedicated "Machine" parameter in the job setup.

If needed, a G-code viewer can be used to double-check the generated toolpaths, if the CAM tool does not have a toolpath preview feature. CAMotics is a popular (and free) option.

Finally the instructions from the generated G-code files must be sent to the machine to produce the required movements of the router to cut through the material. This requires a **G-code sender**, that goes through the G-code file line by line and sends the instructions to the machine, or more precisely to the machine's **controller**, via a communication link (USB on the Shapeoko).

- ⓘ **Carbide Motion** is Carbide 3D's G-code sender for the Shapeoko, but alternative senders can also be used, they are covered in the [CAD, CAM, and G-code](#) section.

The controller executes a piece of software that interprets incoming instructions, and translates them into specific movements of the X, Y, and Z motors. On the Shapeoko, this software is "**GRBL**", (pronounced "Gerbil"), an open source motion control software (see <https://github.com/gnea/grbl>)

It handles the detection of limit switches, and can manage a touch probe to help define and store the coordinates of the reference/starting point for the toolpaths.

- (i) The G-code file also contains instructions to control the **rotation speed** (RPM) of the router, as defined in the CAM program. On a stock Shapeoko, the RPM needs to be set manually on the router anyway, but the controller still generates an output PWM signal modulated based on the RPM value requested in the G-code.

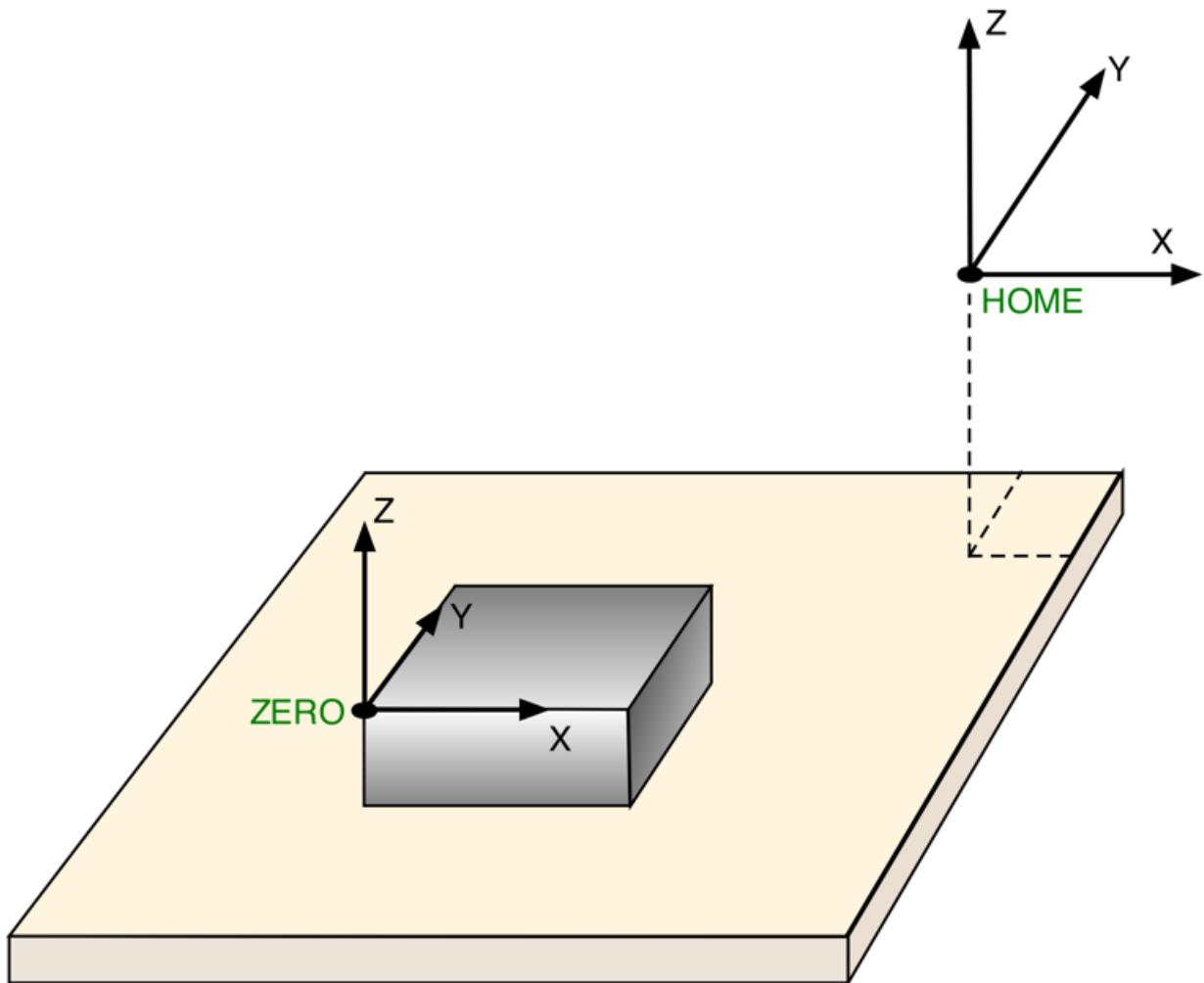
Coordinate system

The coordinate system is one of those things that can be a little confusing at first. The axis definitions themselves are straightforward:

- **X** is the left-right axis, with values increasing from left to right.
- **Y** is the front-back axis, with values increasing from front to back.
- **Z** is what you would expect, vertical axis pointing up, so the "altitude" if you will.

The next question is, where is the origin? On a CNC like the Shapeoko, there is no mechanical feedback telling the machine where it is positioned in space at any given time, so the only thing it can do is control X/Y/Z movements **relative** to a given starting point.

The **ZERO** point (X0,Y0,Z0) is the point in space against which all movements described in a G-code file will be referenced.



This point is usually referenced somewhere on the stock material (e.g., a corner or the center of the top face), but it could be set anywhere in the 3D workspace. The G-code for a given job will use this reference, and perform movements **relative** to this local origin.

However, the machine also has a **Home** position, which is where it can go to reset its location: the Home position corresponds to somewhere where the machine will get a physical feedback that it has reached the position, and on the Shapeoko that's above the back right corner, where **homing switches** on the X, Y, and Z axis happen to be triggered.

Homing consists in telling the machine to move in the direction of positive X, positive Y and positive Z until it detects that each associated limit switch has triggered, and stop movement on the corresponding axis then. Once all three limits switches have been triggered, the machine is guaranteed to be in a known position (mechanically), i.e. Home.

If a G-code file is executed from an arbitrary Zero point, why does it matter where Home is? The trick is that the **coordinates of the Zero point** itself, are defined with respect to the Home position, and happen to be stored in the permanent memory in the controller. So when the machine is in an arbitrary position and is turned off, the next time it will be turned on, Homing allows to go back to this known **absolute** Zero point coordinates.

- (i) Due to the way the X/Y/Z axis are oriented, the Zero point in absolute machine coordinates will have negative values. But since everything will happen relative to the zero point anyway, you can just ignore this fact.

The next section gets into some details about how the different parts of a Shapeoko interact to get the job done.

Anatomy of a Shapeoko

This section provides details on *how* the Shapeoko is working.

-  None of this info is required to be able to successfully use the machine, but you may find some of this information useful for troubleshooting, repairing, or later upgrading the machine.

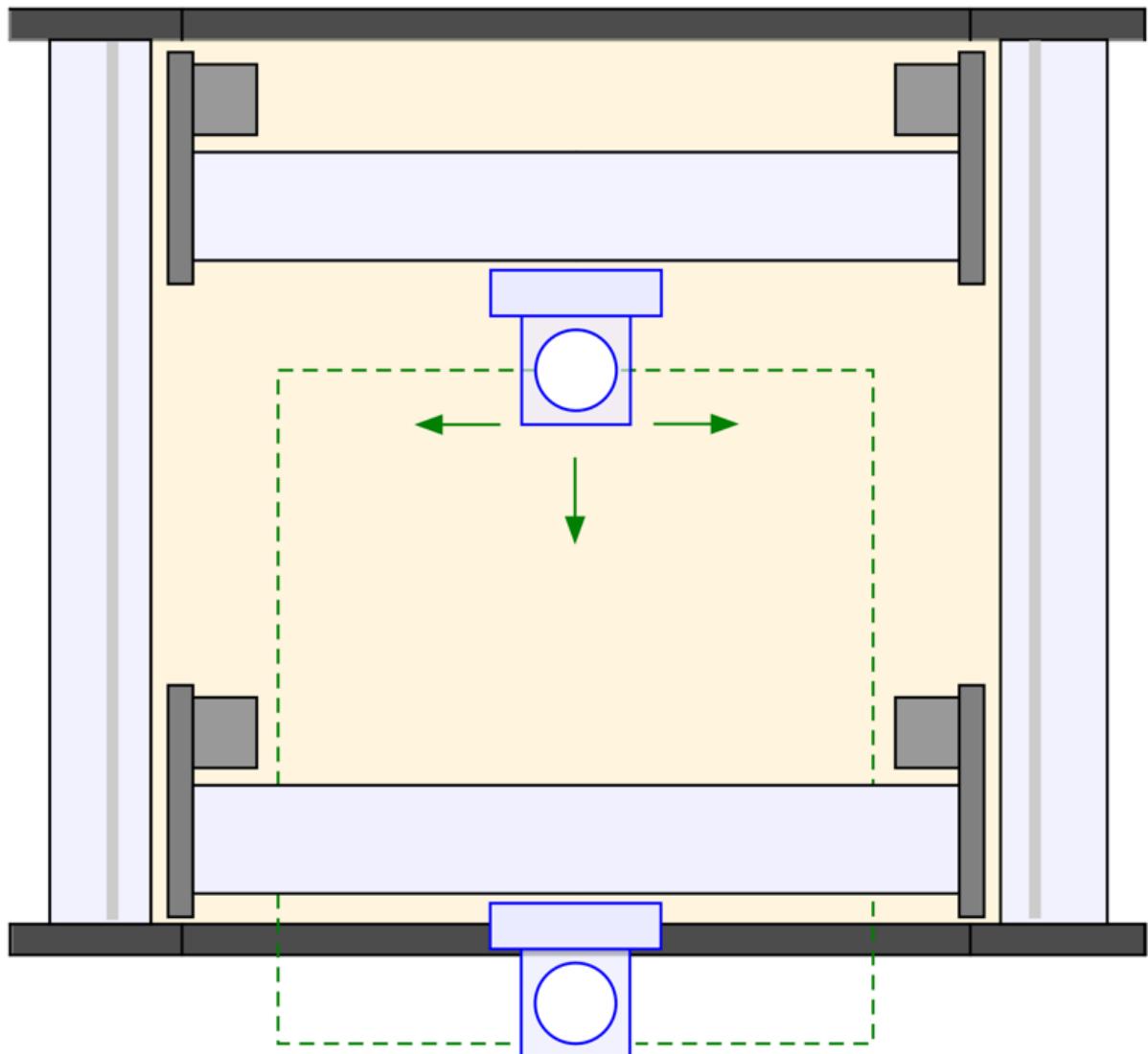
Mechanical structure

That part is self explanatory, since the Shapeoko is a kit everyone will get acquainted with the (very simple) mechanical structure during assembly anyway. The correct geometry of pieces cut on the Shapeoko will depend mainly on:

- making sure the mechanical structure is assembled such that all three axes of movement are *actually* orthogonal to each other (see [Squaring, surfacing, trammimg](#)).
- making sure that there is no slop or excessive friction anywhere in the moving parts (which mostly boils down to tightening the V-wheel eccentric nuts correctly).

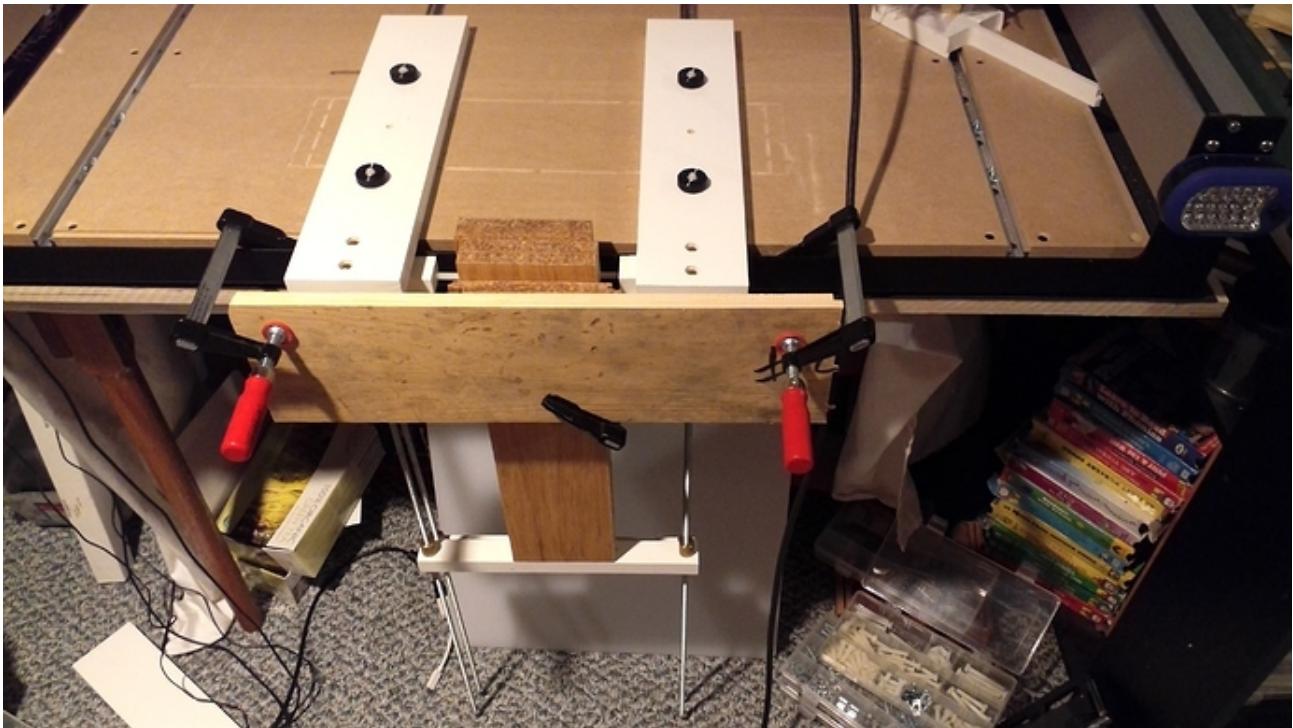
Work area

The Shapeoko3 has a cutting area of $16 \times 16"$, the XL has $16 \times 33"$ and the XXL has $33 \times 33"$. One thing to consider is that the cutting area is not centered, it extends beyond the front plates of the machine:



Some implications may not be immediately apparent:

- a (small) part of the cutting area is lost unless the stock material is overhanging the front of the machine.
- however this is also a great opportunity to use this front area to mill a workpiece clamped **vertically** to the front of the machine e.g., to cut finger joints:



(i) Plans to build this vertical fixture are available at
<https://cutrocket.com/p/5cb25f3380844/>

- when designing an enclosure you may want to take this into account: the router (and potentially the dust shoe) will protrude by a significant amount when the full area is used.

(i) Some folks have *swapped* the Y plates on their machine, this allows them to shift the usable cutting area towards the back *i.e.*, getting larger cutting area inside the machine at the expense of losing the overhang capability.

Stepper motors

As the name implies, stepper motors are designed to rotate by small angle increments, rather than rotating continuously as conventional motors do when they are powered.

Stepper motors are ubiquitous in hobby CNCs, for a good reason: they provide the capability to move by a precise amount (i.e. a given number of steps), without the need for a position feedback mechanism.

The stepper motors on the Shapeoko3 are **NEMA23** (just a fancy way to say that their front/back face size is 2.3" square), are of the "**bipolar**" type (which means that they are controlled by two pairs of wires, the "A" phase and the "B" phase, and that one step is made by sending current in A and turning off B, or the other way around), and are internally designed so that each step rotates the shaft by exactly 1.8° , which translates to $360^\circ / 1.8^\circ = 200$ steps to perform a full revolution.

It's the job of the **stepper driver** to generate current alternatively in the A and B phases, when instructed to move by one or more steps.

- ⓘ The stepper drivers use a trick to optimize the torque: instead of generating a constant voltage (and therefore current) in a phase, they generate a higher voltage and chop it at a high frequency so that it produces the desired value on average. It's just an implementation detail, but it explains the (normal) humming sound that the motors emit when the machine is idle.

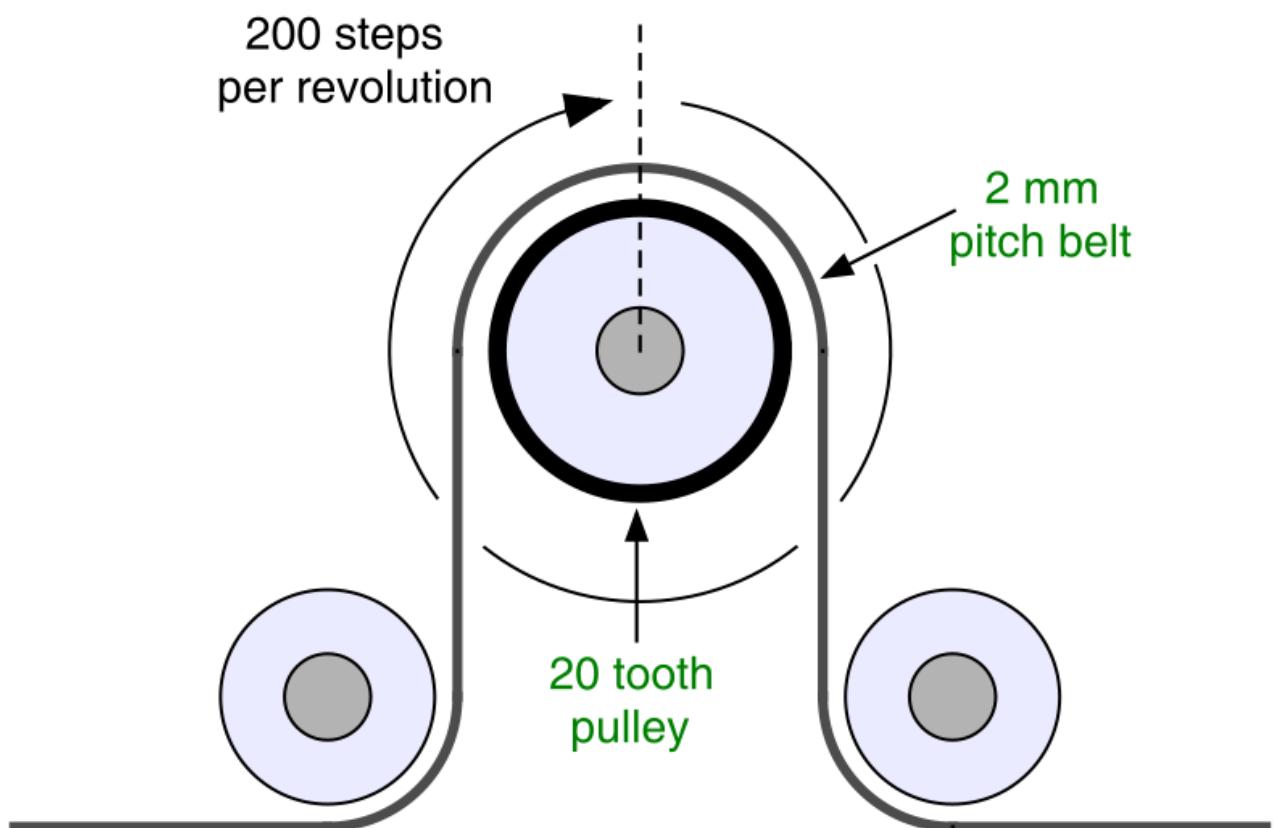
But it gets better: instead of just turning the current on and off completely alternatively on phases A and B to move one step at a time, if the stepper driver sends a carefully chosen variable amount of current in A and B simultaneously, then it is possible to reach (and hold) intermediate positions between two steps: these are called "**microsteps**", and the drivers used in the Shapeoko are capable of controlling **8 microsteps for each step**.

So overall, the motors can be controlled with a precision of $200 \times 8 = 1600$ microsteps per revolution.

- ⓘ Interestingly, stepper motors use more power when doing nothing (i.e., holding a position) than when moving. This is why they tend to get warm when the machine is on but idle. This is not a problem in itself, the motors are designed to support high temperatures but you might as well turn the machine off if it is going to stay idle for a long time, if only to save power.

Pulleys & belts

A pulley is installed on the motor shaft, and drives a "GT2" belt that has a 2mm pitch, *i.e.* distance in mm between two teeth).



The pulleys used on the Shapeoko 3 happen to have 20 teeth, with (obviously) the same 2mm spacing as the belt: so when the shaft does one full revolution, the belt moves by 20 teeth, *i.e.* 40mm.

This full revolution requires 1600 microsteps, which means that it requires $1600/40 = 40$ steps to move by 1mm. And this is where the "**40 steps/mm**" setting in the Shapeoko controller comes from (more on this in the [Dimensional accuracy](#) section)

Which means that the minimal movement that the Shapeoko can theoretically do in any axis is 1/40th of a mm, that's 0.025mm or 0.001". Quite precise, right?

And that when the machine needs to move by X mm along one axis, the stepper motor must be told $40 \times X$ times to do one (micro)step.



Everyone is too lazy to write "microstep" every time, so they just write "step". 99% of the time when discussing steps, you should understand "microstep", because this is what the stepper driver generates. The other 1% is when discussing stepper motor characteristics themselves, which are characterized by their number of "full" steps, because that's an intrinsic property of the motor design (while the number of microsteps is only driven by the selected stepper driver).

So in theory, telling the motor to do N steps will move the associated axis by $N/40$ mm. But that's only true if the effort put on the shaft is lower than the torque the motor is able to provide. When the forces on the shaft exceed the max torque of the motor, commanding one step of rotation will result in...the motor staying in the same position, so effectively "losing" one step, which then causes a discrepancy between where the machine actually is, and where it thinks it is (as it has no feedback loop to verify if it actually moved), and this is bad for accuracy.

Another potential reason for "losing" steps, is that the pulley may slip on the motor shaft if the **set screws** are not tight, so they should be secured/checked (using a 1.5mm Allen key):



Finally, the belts must be **tensioned** correctly, to avoid any slop that could lead to the belt jumping the pulley teeth when a large force is applied on that axis. This is a Goldilocks situation where the belt needs to be tight enough to avoid this problem, but not too tight to avoid bending the motor shaft.

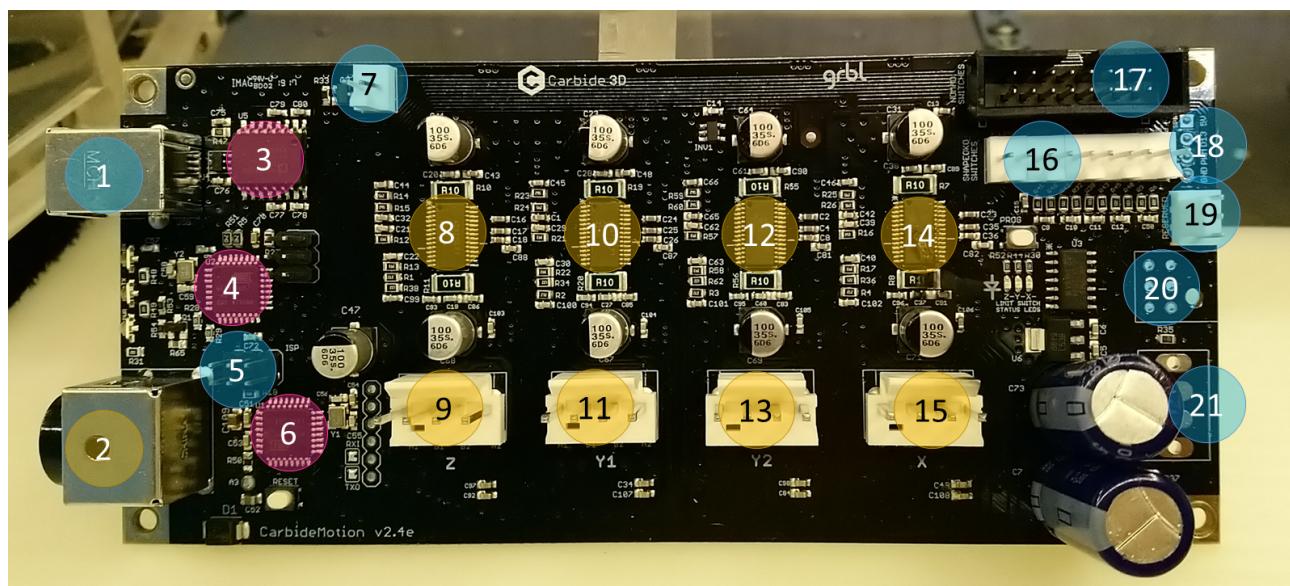
- i** The usual words to characterize an adequate belt tension are "guitar-string tight". For the Y belts, a good indication of proper tension is that when the gantry is at one end of the rails, it should be possible to lift the belt a bit, but it should not be possible to slide a full pinky finger under the middle of the belt.

Natural manufacturing variations across belts and pulleys, a variable level of tensioning, and probably multiple other factors, are such that the actual number of steps required to move by 1mm may not be *exactly* 40, but a tiny bit more or a tiny bit less: this is covered in the X/Y/Z calibration part of the [Dimensional accuracy](#) section.

It is the job of the **controller** to send commands to the stepper drivers (which in turn will generate the right current waveforms on the motor phases), to achieve the desired movements of the machine.

Controller board

The brain of the Shapeoko is the controller board. There have been several revisions over the years, the one presented below is from circa 2017 (v2.4e), the latest one may be slightly different, but fundamentals will likely be the same.



- #1 is the connector of the **USB link** from the host PC that sends the G-code commands
- #2 is the **main power** connector (24V from the external power supply)
- #3 is the **USB isolator**, it helps preventing potential EMI issues.
- #4 is the **Serial-to-USB** microcontroller, supporting the main Arduino controller for USB comms
- #5 is the "Arduino ISP" **header** and carries these signals:

pin column1	pin column2	pin column 3
5V	SPINDLE PWM (= D11 = MOSI)	GND
LIMIT Z (= D12 = MISO)	SPINDLE DIR (= D13 = SCK)	RESET

- **#6** is the Arduino microcontroller (ATMega328P) that runs the motion control software
 - Note: there is a small push button right under the Arduino, to RESET the board manually if needed...never the case in normal usage.
- **#7** is a header with GND and +5V available.
- **#8** is the stepper driver for Z axis
- **#9** is the stepper connector for Z axis
 - from left to right pins: A1/B1/B2/A2 motor signals
- **#10** is the stepper driver for Y axis motor #1
- **#11** is the stepper connector for Y axis motor #1
 - from left to right pins: A1/B1/B2/A2 motor signals
- **#12** is the stepper driver for Y axis motor #2
- **#13** is the stepper connector for Y axis motor #2
 - from left to right pins: A1/B1/B2/A2 motor signals
- **#14** is the stepper driver for X axis
- **#15** is the stepper connector for X axis
 - from left to right pins: A1/B1/B2/A2 motor signals
- **#16** is the "Shapeoko switches" header
 - From left to right, 2 pins per switch:
 - X-limit switch
 - Y-limit switch
 - Z-limit switch
 - E-stop
 - Probe
 - Feed Hold
- **#17** is the "Nomad switches" (the Nomad CNC uses the same controller board), this header is now used for an adapter board on machines with inductive homing switches.
- **#18** now has a connector for a Carbide3D accessory. On older boards it has plated holes for soldering headers to get the following signals (labelled on the silkscreen)
 - GND
 - PWM
 - 5V

- D13
- #19 is a "RESERVED" connector, that happens to be used for the Carbide 3D probe now.
- #20 has plated holes for these signals:

Left pin column	Right pin column
24V	RESERVED
PWM	RESET
RESERVED	GND

- #21 has plated holes for the following (unused) signals, from top to bottom
 - AUX 24V INPUT
 - AUX 24V INPUT
 - AUX ON/OFF
 - AUX ON/OFF

GRBL Motion control software

On the Shapeoko, the piece of embedded software that runs in the Arduino microcontroller is **GRBL** ("Gerbil"), an open-source CNC motion control software (available here: <https://github.com/gnea/grbl>)

Carbide 3D contributes to the development of GRBL, which is why it comes with a Shapeoko-specific set of settings already built-in the code (so if you're into software and ever need to figure out why GRBL behaves the way it does, you can go and check the source code, which is neat!)

For example, here's a copy of the Shapeoko3 settings in GRBL1.1:

```

1 // Description: Shapeoko CNC mill with three NEMA 23 stepper motors, dri
2 #define MICROSTEPS_XY 8
3 #define STEP_REVS_XY 200
4 #define MM_PER_REV_XY (2.0*20) // 2mm belt pitch, 20 pulley teeth

```

```

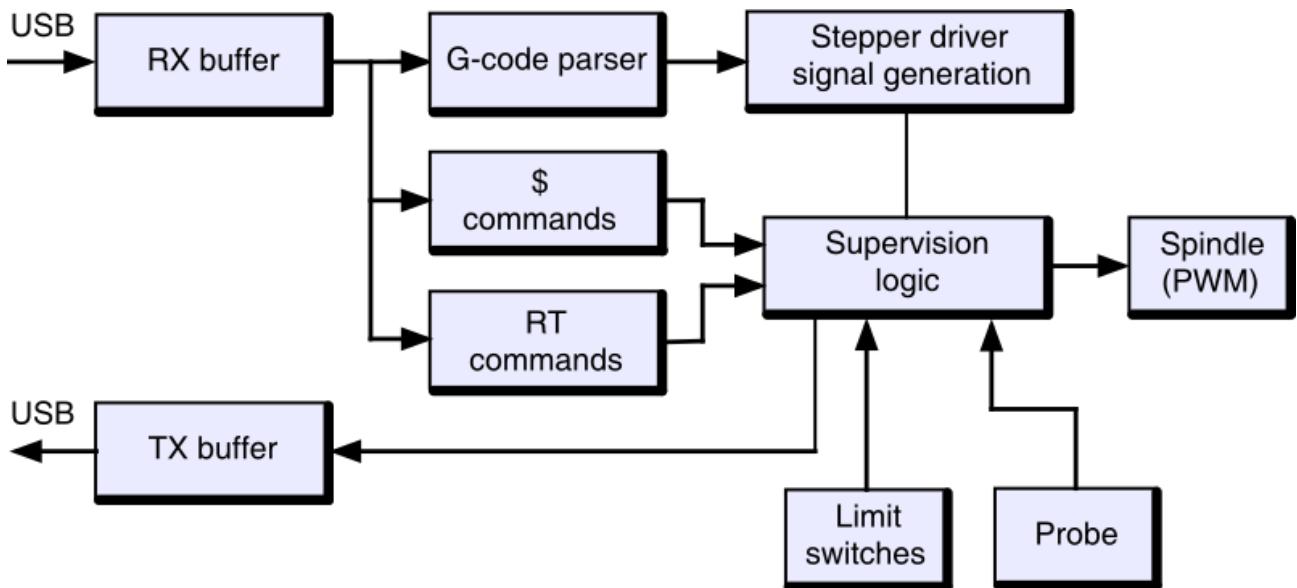
5   #define MICROSTEPS_Z 8
6   #define STEP_REVZ_Z 200
7   #define MM_PER_REV_Z (2.0*20) // 2mm belt pitch, 20 pulley teeth
8   #define DEFAULT_X_STEPS_PER_MM (MICROSTEPS_XY*STEP_REVZ_XY/MM_PER_REV_XY)
9   #define DEFAULT_Y_STEPS_PER_MM (MICROSTEPS_XY*STEP_REVZ_XY/MM_PER_REV_XY)
10  #define DEFAULT_Z_STEPS_PER_MM (MICROSTEPS_Z*STEP_REVZ_Z/MM_PER_REV_Z)
11  #define DEFAULT_X_MAX_RATE 5000.0 // mm/min
12  #define DEFAULT_Y_MAX_RATE 5000.0 // mm/min
13  #define DEFAULT_Z_MAX_RATE 5000.0 // mm/min
14  #define DEFAULT_X_ACCELERATION (400.0*60*60) // 400*60*60 mm/min^2 = 400
15  #define DEFAULT_Y_ACCELERATION (400.0*60*60) // 400*60*60 mm/min^2 = 400
16  #define DEFAULT_Z_ACCELERATION (400.0*60*60) // 400*60*60 mm/min^2 = 400
17  #define DEFAULT_X_MAX_TRAVEL 425.0 // mm NOTE: Must be a positive value.
18  #define DEFAULT_Y_MAX_TRAVEL 465.0 // mm NOTE: Must be a positive value.
19  #define DEFAULT_Z_MAX_TRAVEL 80.0 // mm NOTE: Must be a positive value.
20  #define DEFAULT_SPINDLE_RPM_MAX 10000.0 // rpm
21  #define DEFAULT_SPINDLE_RPM_MIN 0.0 // rpm
22  #define DEFAULT_STEP_PULSE_MICROSECONDS 10
23  #define DEFAULT_STEPPING_INVERT_MASK 0
24  #define DEFAULT_DIRECTION_INVERT_MASK ((1<<X_AXIS)|(1<<Z_AXIS))
25  #define DEFAULT_STEPPER_IDLE_LOCK_TIME 255 // msec (0-254, 255 keeps ste
26  #define DEFAULT_STATUS_REPORT_MASK 1 // MPos enabled
27  #define DEFAULT_JUNCTION_DEVIATION 0.02 // mm
28  #define DEFAULT_ARC_TOLERANCE 0.01 // mm
29  #define DEFAULT_REPORT_INCHES 0 // false
30  #define DEFAULT_INVERT_ST_ENABLE 0 // false
31  #define DEFAULT_INVERT_LIMIT_PINS 0 // false
32  #define DEFAULT_SOFT_LIMIT_ENABLE 0 // false
33  #define DEFAULT_HARD_LIMIT_ENABLE 0 // false
34  #define DEFAULT_INVERT_PROBE_PIN 0 // false
35  #define DEFAULT_LASER_MODE 0 // false
36  #define DEFAULT_HOMING_ENABLE 0 // false
37  #define DEFAULT_HOMING_DIR_MASK 0 // move positive dir
38  #define DEFAULT_HOMING_FEED_RATE 100.0 // mm/min
39  #define DEFAULT_HOMING_SEEK_RATE 1000.0 // mm/min
40  #define DEFAULT_HOMING_DEBOUNCE_DELAY 25 // msec (0-65k)
41  #define DEFAULT_HOMING_PULLOFF 5.0 // mm

```

- (i) Note the default maximum feedrate of 5000mm/min, i.e. a bit less than 200"/min. Carbide Motion 505 and later will override this setting during the initial machine configuration, to a maximum of 10000mm/min (394ipm), to allow for fast rapid movements (outside the material)

The first six settings should be clear from the whole steps/microsteps section above, the rest are default values for settings that can be reconfigured anyway.

What GRBL does is listen to incoming commands on the USB interface, and act upon receiving them. Here's a very rough description of what's going on inside GRBL:



- first there is a **reception buffer**, and that's a critical point because while the Arduino microcontroller can execute code with very deterministic timings, that's not the case of the host PC at the other end of the USB cable, which executes e.g. Carbide Motion on Windows or Mac OS X. And as everyone has experienced, from time to time Windows or MAC OS can decide to go and do something else for a little while, and this could break the flow of G-code commands into the controller. With this buffer, the control software can send a few additional G-code commands in advance of the current one, to ensure that the controller will never starve waiting for the next command from USB.
- the **G-code commands** are parsed and processed by a dedicated piece of code that generates the appropriate signals to drive the stepper driver to produce the desired motion
- the **'\$' commands** are interpreted to update various internal variables
- the **realtime commands** consist of a single character, that has an immediate effect as soon as it is received, having priority over anything else GRBL is currently doing. For example , sending '!' will trigger a Feed Hold.
- GRBL also uses hardware input signals from the **limit switches** and potentially from a **probe**, and commands an output "**PWM**" spindle signal to potentially drive the RPM of a spindle

-  The PWM/Spindle signal is unused on a stock Shapeoko, but installing a spindle is a popular mod, and then this signal comes in handy.

Below are the three most useful GRBL commands when using the Shapeoko:

- **\$\$** prints out the current value of the settings
- **\$<x>=<val>** writes value *va*/ in setting *x*
- **\$H** homes the machines

These commands can be typed from the G-code **console**, all G-code senders have one (the one in Carbide Motion is called "MDI" for "Manual Data Input").

Trim Router

All of the above of course serves a single purpose: moving a router and its cutter in three dimensions. The very concept of the Shapeoko is to be able to use an inexpensive consumer-grade compact **router**. The most popular ones (if only because they are supported out of the box by the Shapeoko mechanical kit) are:

- the DeWalt DWP611 in the US, or its European counterpart the D26200
- the Makita RT0701 in the US, or its European counterpart the RT0700
- Carbide 3D's "Carbide compact router" (CCR), for the US only, which is a close sibling of the Makita.

but in theory any other hand router could be fitted on the Shapeoko, with the right mount adapter.

The most significant difference between the DeWalt and the Makita/CCR is the RPM range: the Makita can be set from 10.000 RPM to 30.000RPM, the CCR from 11.000 RPM to 31.000 RPM, while the DeWalt is limited to 16.000 to 27.000RPM.

Dial position (Makita)

RPM



1	~10,000
2	~12,000
3	~17,000
4	~22,000
5	~27,000
6	~30,000

Dial position (DeWalt)	RPM
1	~16,500
2	~18,000
3	~19,800
4	~21,500
5	~24,200
6	~27,000

Dial position (CCR)	RPM
1	~11000
2	~13600
3	~18250
4	~25000
5	~29000
6	~31000

If you know you will need to be using lower RPMs, the Makita or CCR will be a better choice. Other than that, all three routers (and more) have been used successfully for all kinds of jobs on the Shapeoko.

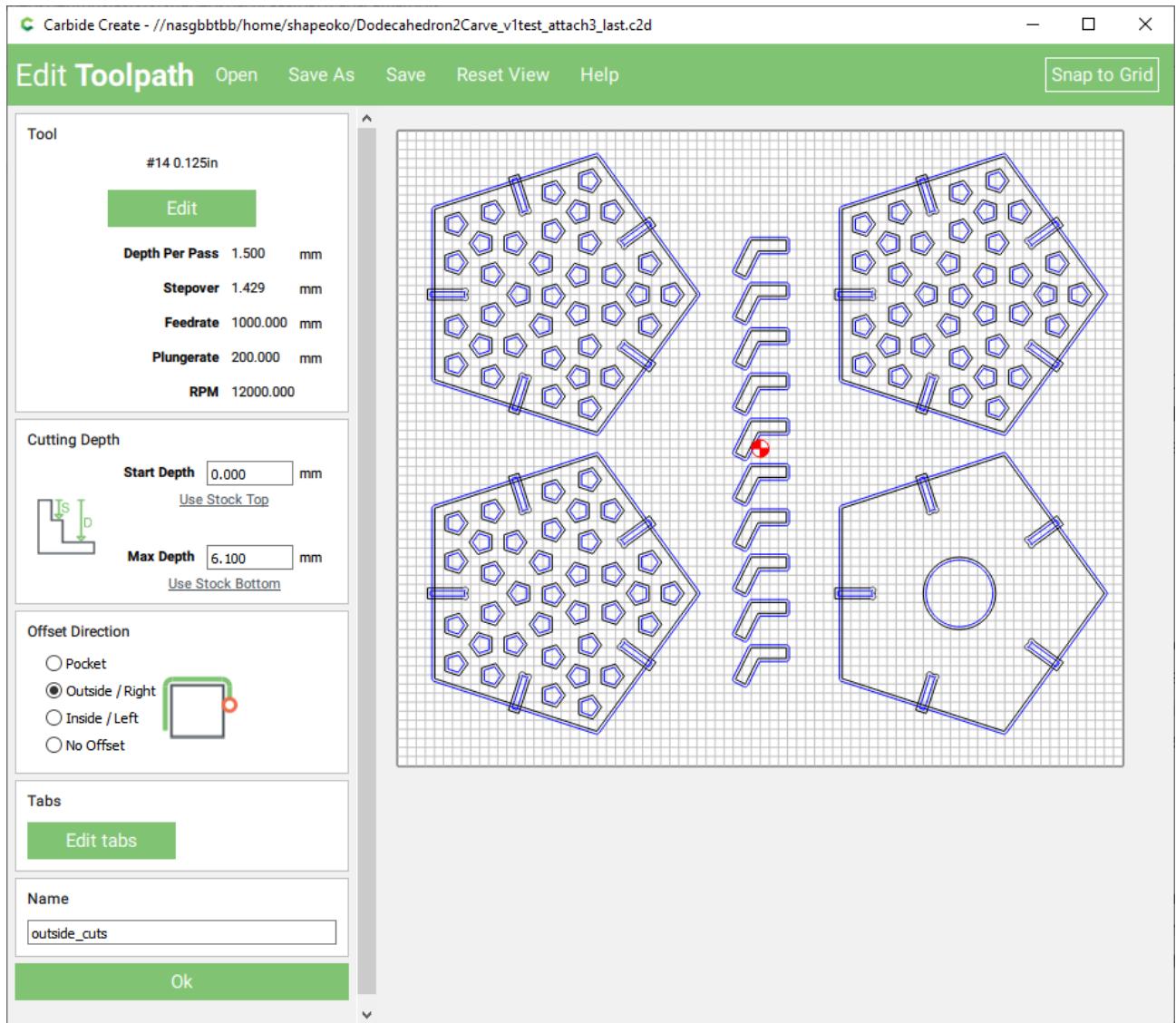
Pro CNCs usually have a **spindle**, not a router, and that is a possible (and popular) upgrade path for the Shapeoko, check out the [HW upgrades](#) section for more.

CAD, CAM, and G-code

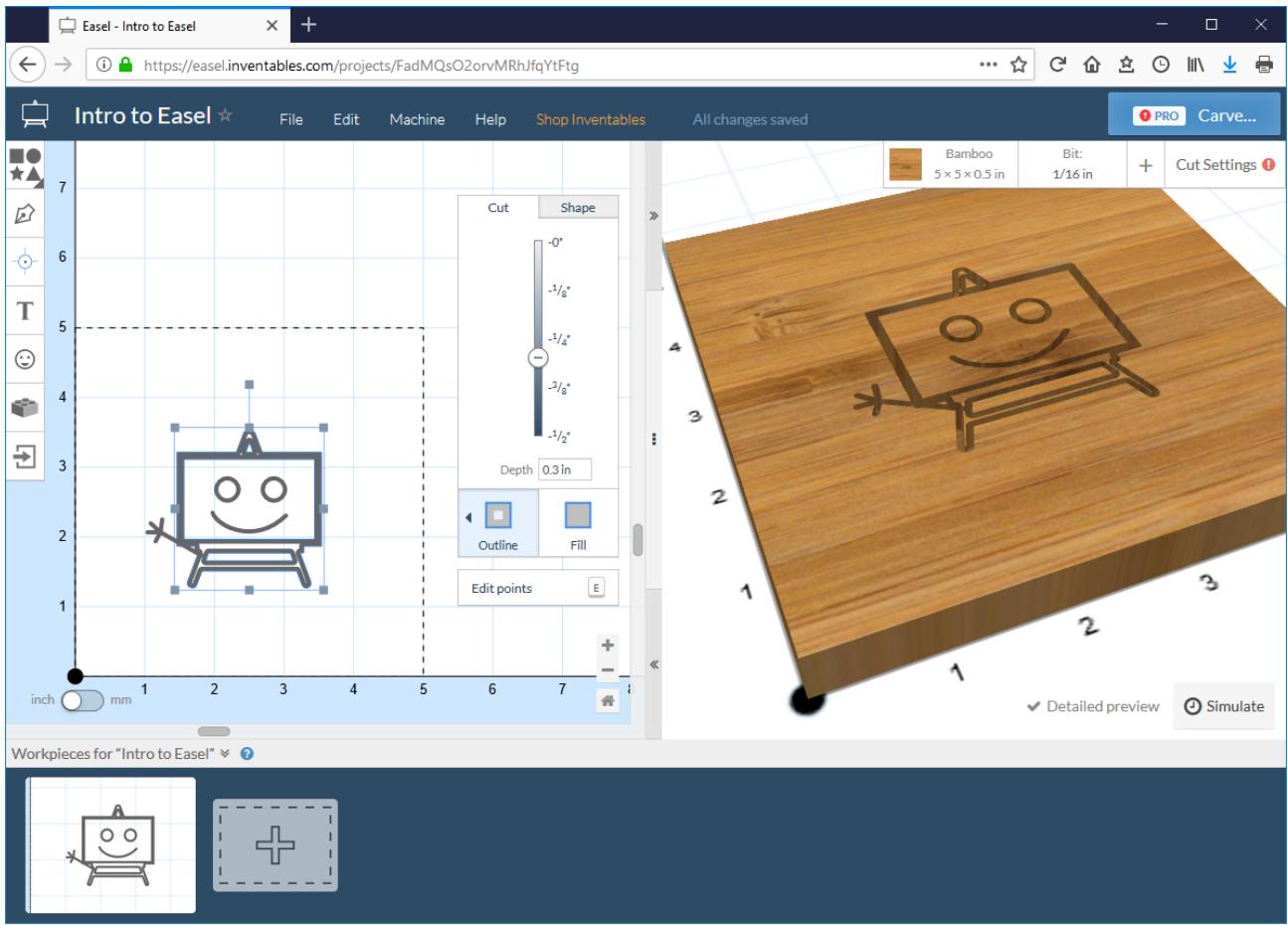
This section provides a brief overview of popular software for CAD, CAM, and sending G-code to the Shapeoko, at the time of writing. Most users start using their Shapeoko with Carbide Create and Carbide Motion, some stick to them while others move on to something else, for a variety of reasons.

CAD/CAM tools

Most users obviously start with **Carbide Create**, which really is a great solution to get started with CNC, because it has *just* the right amount of features to not overwhelm newbies with tons of parameters but still allow them to experience the full design workflow: stock setup, creating a 2D design, creating basic toolpaths based on these design elements, previsualizing these toolpaths and the final workpiece, and finally generating G-code to run on the machine.



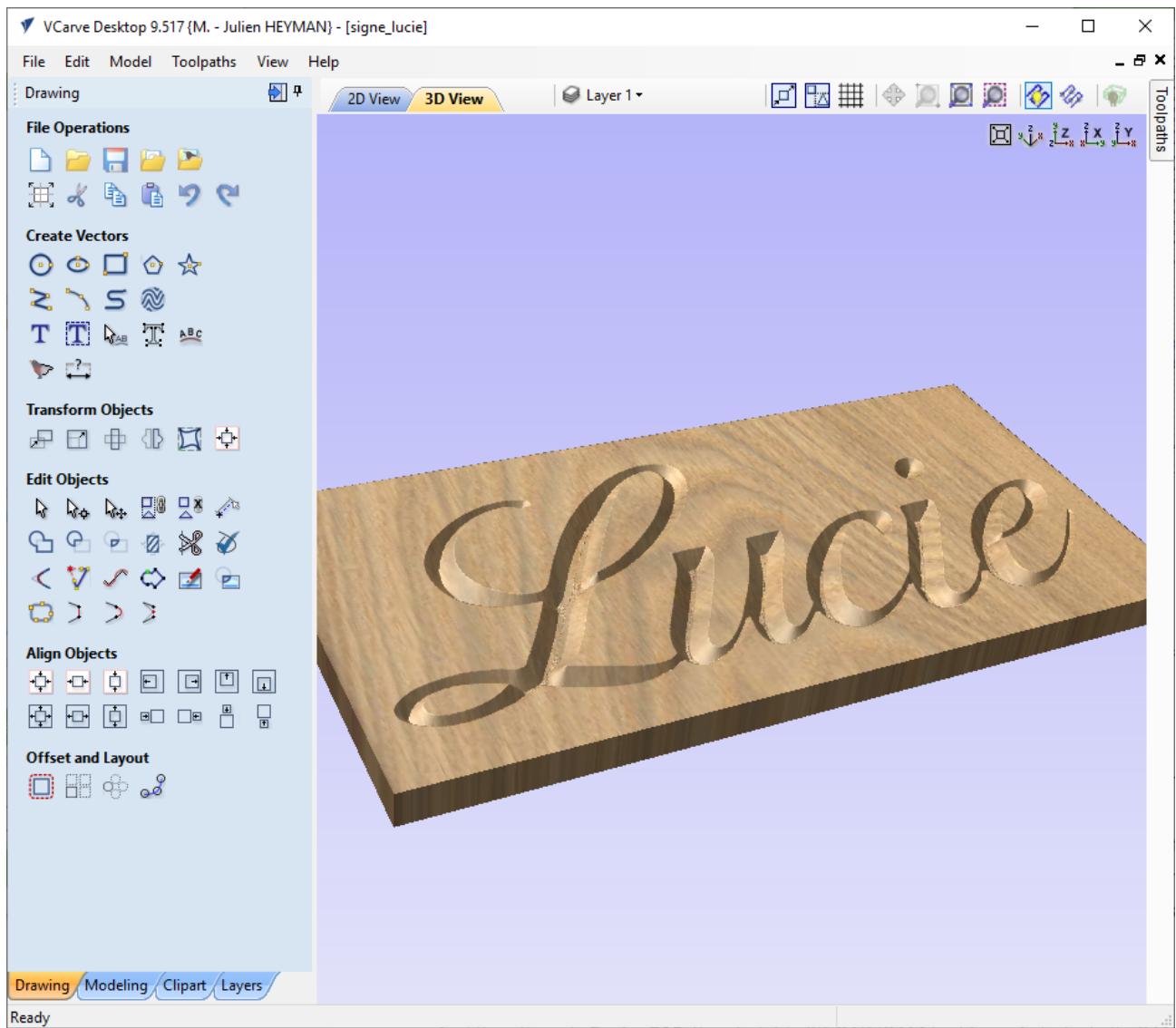
As a side note, the X-Carve CNC (competitor product sharing some DNA with the Shapeoko) comes with a similar entry level CAD/CAM program called **Easel**:



Even though it is mostly intended to be used with the X-Carve, some folks use the CAD part of Easel (an admittedly nice and simple albeit web-based UI) to generate G-code files that can then be used in any G-code sender for the Shapeoko.

If/once you outgrow Carbide Create, you can obviously look into upgrading to **Carbide Create Pro** and its advanced features.

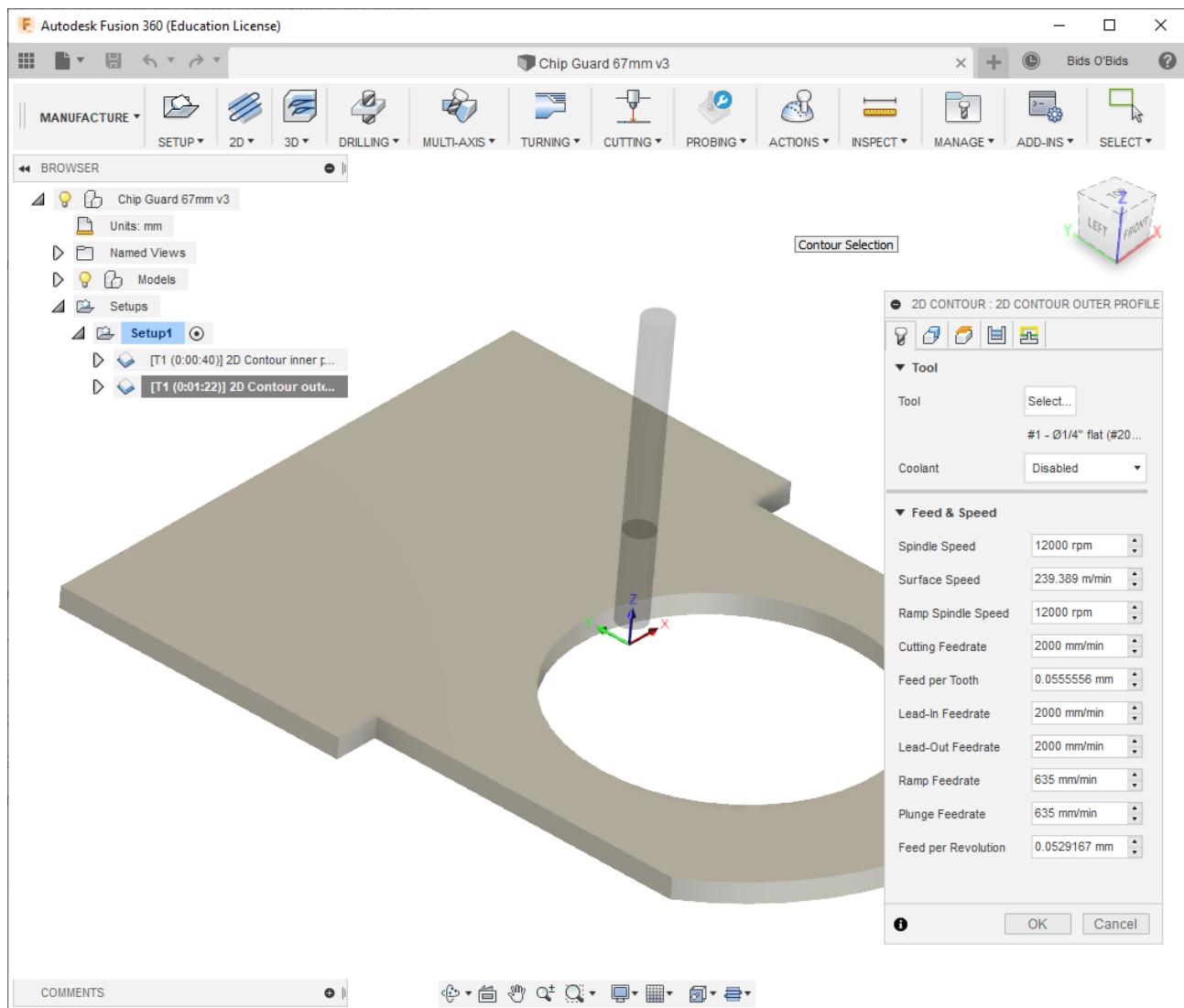
Vectric **VCarve** is another very popular (albeit somewhat pricey) upgrade path. The name is slightly confusing, as it is not only specialized in V-carving but is a complete generic CAD/CAM tool. The workflow is quite similar to Carbide Create's, which makes the transition easy. It has more CAD features (layers, support for 2-sided machining,...), intermediate-level CAM features (built-in support for roughing and finishing strategy is great), and it is just a very polished and robust software. You can also go crazy and buy the top of the line **Aspire** software from Vectric, if you need the advanced/pro features it offers.



And then there is **Fusion360**, the almighty 3D CAD/CAM tool from Autodesk. Its CAM module has all the bells and whistles and a truckload of settings, which is also why it has an admittedly steep learning curve that can repel many casual CNC users. But if you get past those first few weeks of figuring out its workflow and main settings, it opens up a fascinating range of possibilities, and not only for CNC.

- ⓘ If you are interested in Fusion360's "adaptive clearing" toolpaths covered in the [Toolpaths](#) section, a possible alternative is Estlcam software (<https://www.estlcam.de/>), it features "trochoidal milling" toolpaths which are essentially the same, and is arguably much simpler to learn.

There is a small catch though: while it has an offline mode, this is primarily an online/cloud-oriented tool, it's from Autodesk, and it's free for students and hobbyist...for now. Not everyone feels comfortable investing a lot of time into learning how to use a tool that might become unusable locally if the servers are shutdown, or could become costly. Also, it does not do (proper) Vcarving.



I use CC or VCarve or Fusion360 depending on the project at hand. I will use Carbide Create when I need a simple 2.5D piece done quickly. VCarve is my go-to solution for the 2.5D projects that are more complex (require more structuring) and/or involve Vcarving, 2-sided work, or require a roughing+finishing toolpath strategy. And then I will use Fusion360 for all things 3D, for metal work (mainly because of the adaptive clearing feature), and whenever I feel like I need to make the design parametric so as to be able to adjust dimensions without having to redesign everything.

Of course, there are many other CAD/CAM tools out there, those are the ones that are most popular at the time of writing in the Shapeoko community.

- (i) While the post-processor to generate G-code is built in to Carbide Create, for VCarve and Fusion360 you need to initially configure it (easy enough) but then you get to tune it to your liking, which is very convenient.

G-code primer

So the CAD/CAM tool generates G-code files. One can perfectly use the Shapeoko without any understanding of the G-code syntax, or ever opening a generated G-code file. Still, having at least a superficial understanding of what the most common G-code instructions do goes a long way for troubleshooting why the machine behaves the way it does.

G-code is a (somewhat) standard language to define instructions to be fed to multi-axis machines, CNC and 3D printers being prime examples.

A G-code file is usually a **plain text** file (that can be opened with any text editor), containing a sequence of G-code "blocks", i.e. lines of instructions. Each block (line) can contain several G-code instructions, but it often has a single command and its parameters, for better readability.

These instructions are being sent (by *e.g.* Carbide Motion) to the machine, that executes them **in order**.

- (i) G-code standard does define loop/jump instructions (GOTO), but they are not supported by GRBL anyway, so on a Shapeoko the G-code is guaranteed to execute from the top to the bottom of the file sequentially.

Here's the beginning of a random G-code example generated using Carbide Create (other CAM tool produce slightly different variations of the code) :

```

1 %
2 (TOOL/MILL,0.1,0.05,0.000,0)
3 (FILENAME: )
4 ()
5 G21
6 G90
7 G0X0.000Y0.000Z10.000
8 (TOOL/MILL,1.5875,0,1.0000,0.0)
9 M6 T112
10 M3 S10000
11 G0X0.000Y150.000
12 G0Z10.000
13 G1Z-0.250F300.0
14 G1X150.000F1200.0
15 Y0.000
16 X0.000
17 Y150.000
18
19 [...many other G-code instructions...]
20
21 G0Z10.000
22 M5
23 M30
24 (END)

```

Let's break this down:

- the % line just means "start of program". Well it actually means "Tape start", from the days when CNCs used tape readers. Anyway, this sign is optional, you will find some CAM tools add it, others don't.
- everything between parentheses is a **comment**, and is ignored by GRBL. This is just there for readability of the G-code file.
- **G21** is the command to set **units** to mm (G20 being the command to set units to inches)
- **G90** is the command to go to **absolute positioning** mode, i.e. all subsequent move commands will interpret X/Y/Z values as coordinates relative to the ZERO point currently defined. By contrast, G91 would activate relative positioning mode, and the X/Y/Z instructions would be interpreted as offsets from the current position.
- **G0** is the **Rapid Move** command, it takes X and/or Y and/or Z and/or Feedrate values as parameters, and is intended to reposition the tool to a different location while not cutting anything on the way.

- **M6 T112** corresponds to a **Tool Change** command, requesting tool number 112 in this example. On a Shapeoko, since there is no automatic tool changer, this is ignored by the machine (but used by Carbide Motion to trigger a user prompt)
- **M3 S10000** instructs the controller to turn the **Spindle on**, at 10.000RPM in this example.
- **G1** is the **Linear move** command, similar to G0 but intended to be used for actual cutting moves, that typically happen slower than the G0 rapid moves. The feedrate for these moves can be specified on the line, that's the "**Fxxx**" part and the xxx is the feedrate value in units (inch or mm) per minute.
- standalone coordinates such as **Y0.000, X0.000, Y150.000** are in fact implicit/short versions of G1 move commands: as long as the movement mode does not need to be changed, the latest G commands applies and in this example the "G1" can be omitted from these lines.
- **M5** is used to turn the **Spindle off**
- **M30** means "End of program" in GRBL.



- On a stock Shapeoko with a trim router, there is no automatic control of the router activation nor RPM, so the Spindle commands have no visible effect, but it does modify the output of the PWM signal on the controller board, which is used if you have a spindle
- The Shapeoko does not have an automatic tool changer, so M6 commands are ignored by the controller, but Carbide Motion uses the M6 tool change command to trigger a user prompt.
- Spaces inside a line are ignored

Beyond this basic example, a few more common commands are:

- **G2/G3** commands to do **Arc moves**.
- **G17** explicitly selects the XY plane as the plane in which these arc moves are done.
- **G94** explicitly sets the feedrate values to be "units (inch or mm) **per minute**".
- **M8/M9** is Coolant on/off. Not applicable/ignored on the Shapeoko, but may be present depending on the post-processor used.
- **G54** is the command to select "coordinate system #1", a.k.a. the coordinate system that is based on the Zero point you set. This command is optional since G54 is the default at GRBL startup anyway.

There are many, many more G-code commands, but basically the commands above will cover 99% of the needs on a Shapeoko.

G-code senders

Again, most users will initially use **Carbide Motion** to send G-code to the Shapeoko, and there is very good chance that many will never need to consider anything else. After all the workflow is quite simple (load G-code, zero, run), and Carbide Motion does the job.

But here are a few reasons why other senders can be considered too:

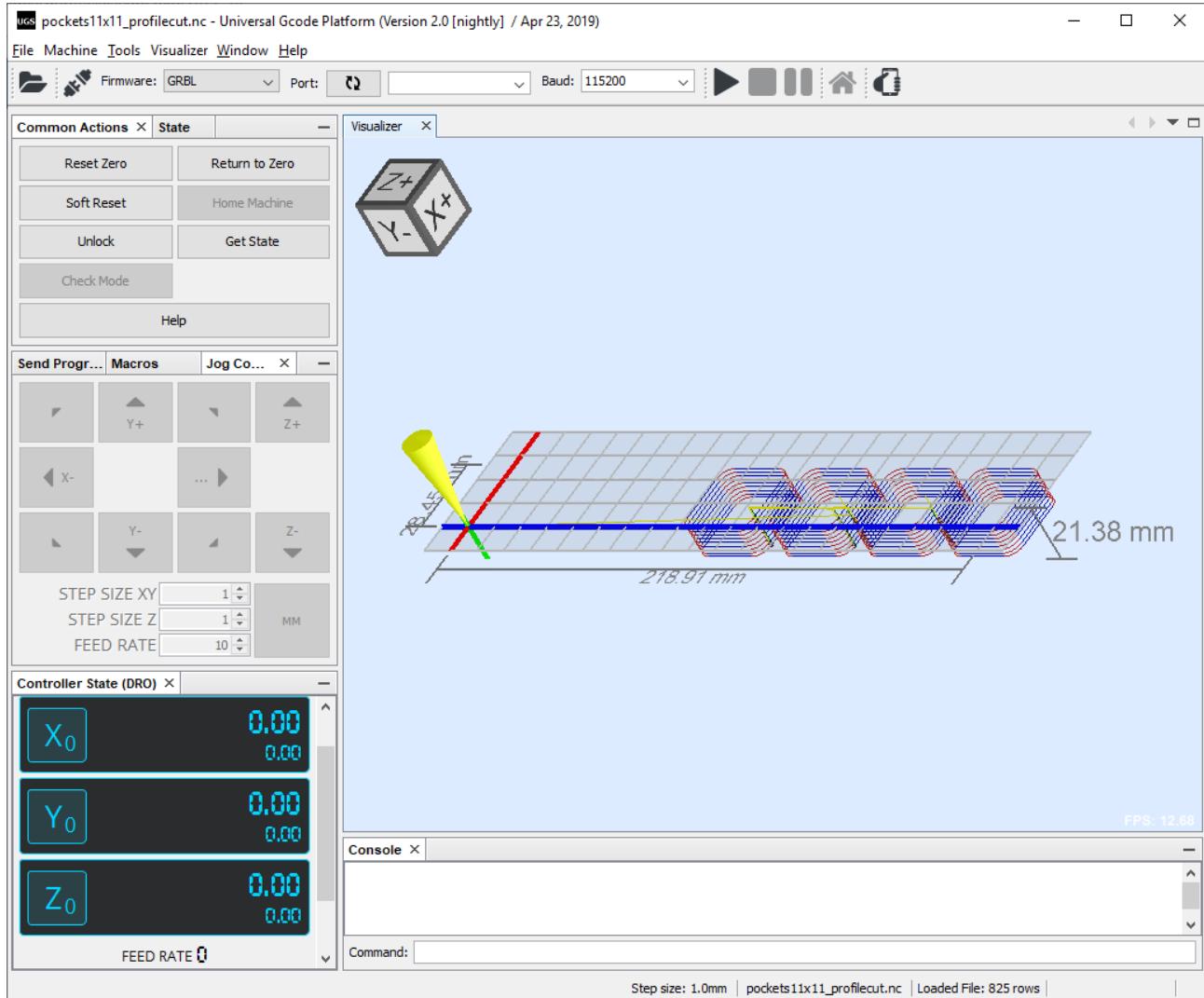
- **live toolpath simulation.** It is quite convenient to be able to visualize the toolpaths contained in the G-code file that was loaded, if only to double-check that it looks right before running it, and then to monitor the cut and be able see what move is coming next in the toolpath.
- **G-code Macros.** Small snippets of G-code with associated buttons/shortcuts in the UI can be very useful to streamline the workflow. It can be as simple as just going to X0/Y0, or be a complex custom automated probing routine.
- **machine limits customization.** While Carbide Motion is setup for a stock Shapeoko, and while it does provide access to modify the various GRBL parameters, some aspects are hardcoded (e.g. the probe dimensions, that match Carbide 3D's probe, or the Z limits that match a stock Z-axis). If you start modding your machine, you *may* come to a point where you need more customization than Carbide Motion allows.

There are many alternative G-code senders in various states of maturity/activity, I will just focus on the two that I have used, and which seem to be most popular on the Shapeoko forum:

Universal G-code Sender is a cross-platform desktop application based on Java (which in other words means you will be able to use it on any operating system). Its UI looks a bit dated but it is simple and efficient. Here's a few highlights of why I like it:

- the G-code preview pane is particularly helpful: not only does it display the toolpaths in the currently loaded file with the associated max dimensions (I check those every single time I load a file, as a double check), it also shows where the tool is in realtime (I like

this feature when I need to check how many more depth passes are left until a pocket is finished, or where the tool will go next):



- it has a very simple editor for GRBL parameters: just type in the value, hit save, and the new parameter is stored in the controller:

The screenshot shows a Windows-style dialog box titled "Settings" with a blue header bar and a red close button. The main area is a table with three columns: "Setting", "Value", and "Description". The table contains 32 rows of configuration parameters. At the bottom of the dialog are four buttons: "Close", "Export", "Import", and "Save".

Setting	Value	Description
\$0	10	Step pulse time
\$1	255	Step idle delay
\$2	4	Step pulse invert
\$3	2	Step direction invert
\$4	0	Invert step enable pin
\$5	0	Invert limit pins
\$6	0	Invert probe pin
\$10	255	Status report options
\$11	0.020	Junction deviation
\$12	0.010	Arc tolerance
\$13	0	Report in inches
\$20	1	Soft limits enable
\$21	1	Hard limits enable
\$22	1	Homing cycle enable
\$23	0	Homing direction invert
\$24	100.000	Homing locate feed rate
\$25	1000.000	Homing search seek rate
\$26	25	Homing switch debounce delay
\$27	5.000	Homing switch pull-off distance
\$30	10000	Maximum spindle speed
\$31	0	Minimum spindle speed
\$32	0	Laser-mode enable
\$100	39.955	X-axis travel resolution
\$101	40.035	Y-axis travel resolution
\$102	320.000	Z-axis travel resolution
\$110	5000.000	X-axis maximum rate
\$111	5000.000	Y-axis maximum rate
\$112	1000.000	Z-axis maximum rate
\$120	400.000	X-axis acceleration
\$121	400.000	Y-axis acceleration
\$122	200.000	Z-axis acceleration
\$130	419.000	X-axis maximum travel
\$131	396.000	Y-axis maximum travel
\$132	150.000	Z-axis maximum travel

- it has configurable keyboard shortcuts for most of the actions, which makes it convenient to use a custom keypad (see [Shapeoko setup](#)). Finally, I use the ability to define Macros, for simple but useful things like "goto X0Y0"

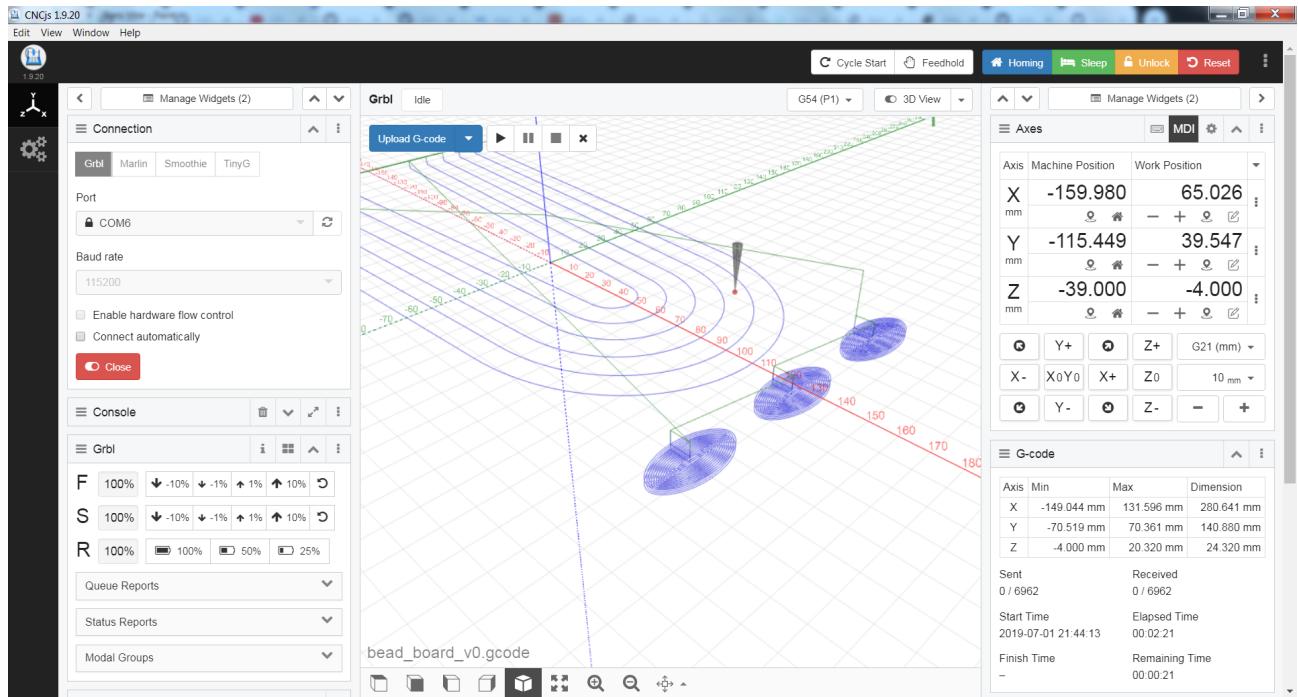


Carbide Motion also has convenient shortcuts for use with a keypad, see:

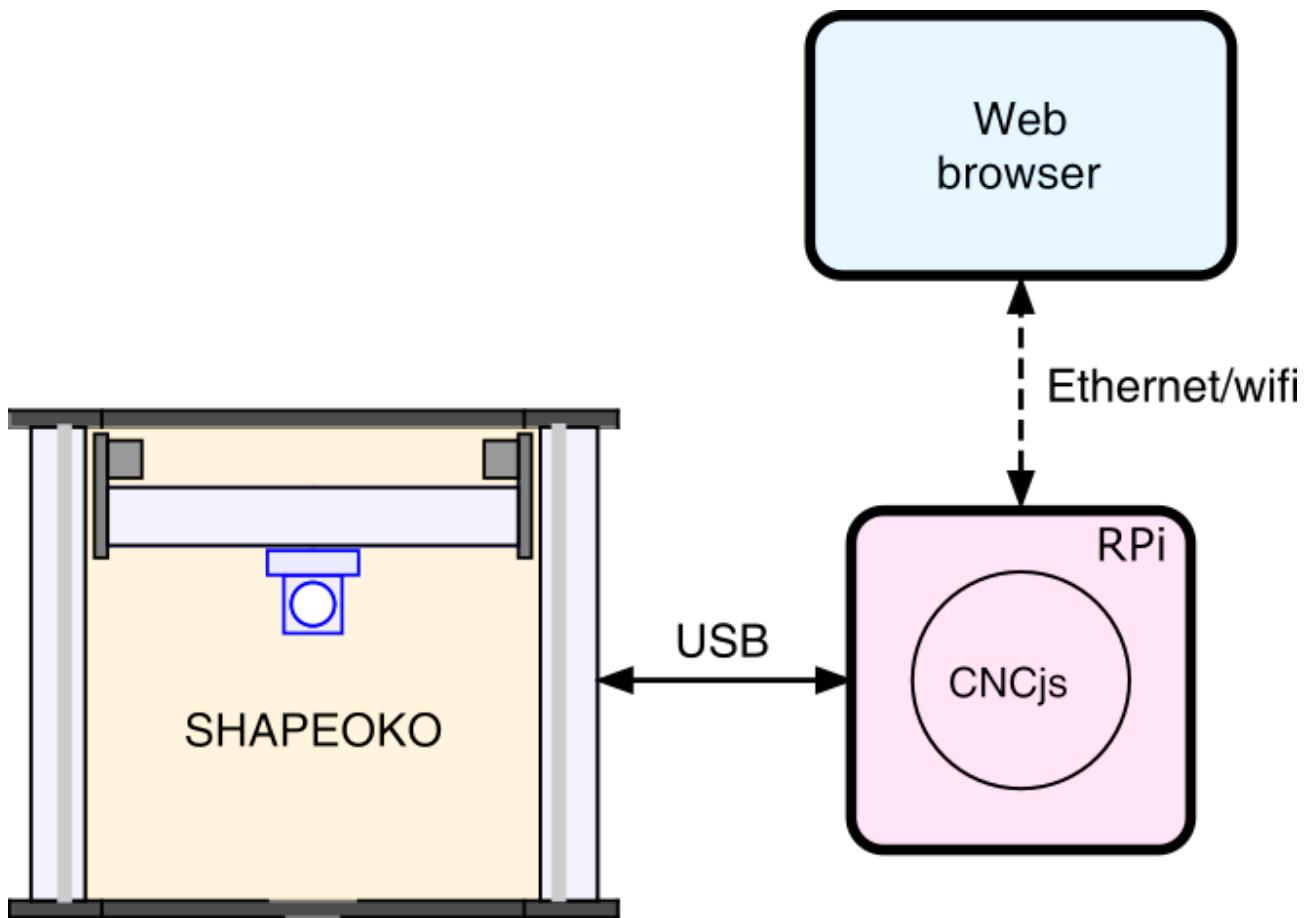
<https://community.carbide3d.com/t/keyboard-cheat-sheet-for-carbide-create-and-motion/7839>

- it supports G-code filters, which can turn out to be convenient to ignore e.g. generated tool change commands.

CNCjs is another popular sender, it has all the features of UGS and more, an arguably better-looking UI, and powerful macro capabilities. It comes either in a standalone desktop application:



or as a backend accessed via a web interface from any browser. This latter configuration makes it interesting for installation on a Raspberry pi (or any other cheap computer that can be dedicated to being a G-code sender):



In this configuration, a G-code file can be loaded directly from a web browser running on any device (PC, Mac, tablet, smartphone...), and CNCjs running on the Raspberry Pi takes care of feeding the G-code to the Shapeoko via the Raspberry's USB port. The device running the web browser can be turned off, or used for some other tasks, without impacting the ongoing job.

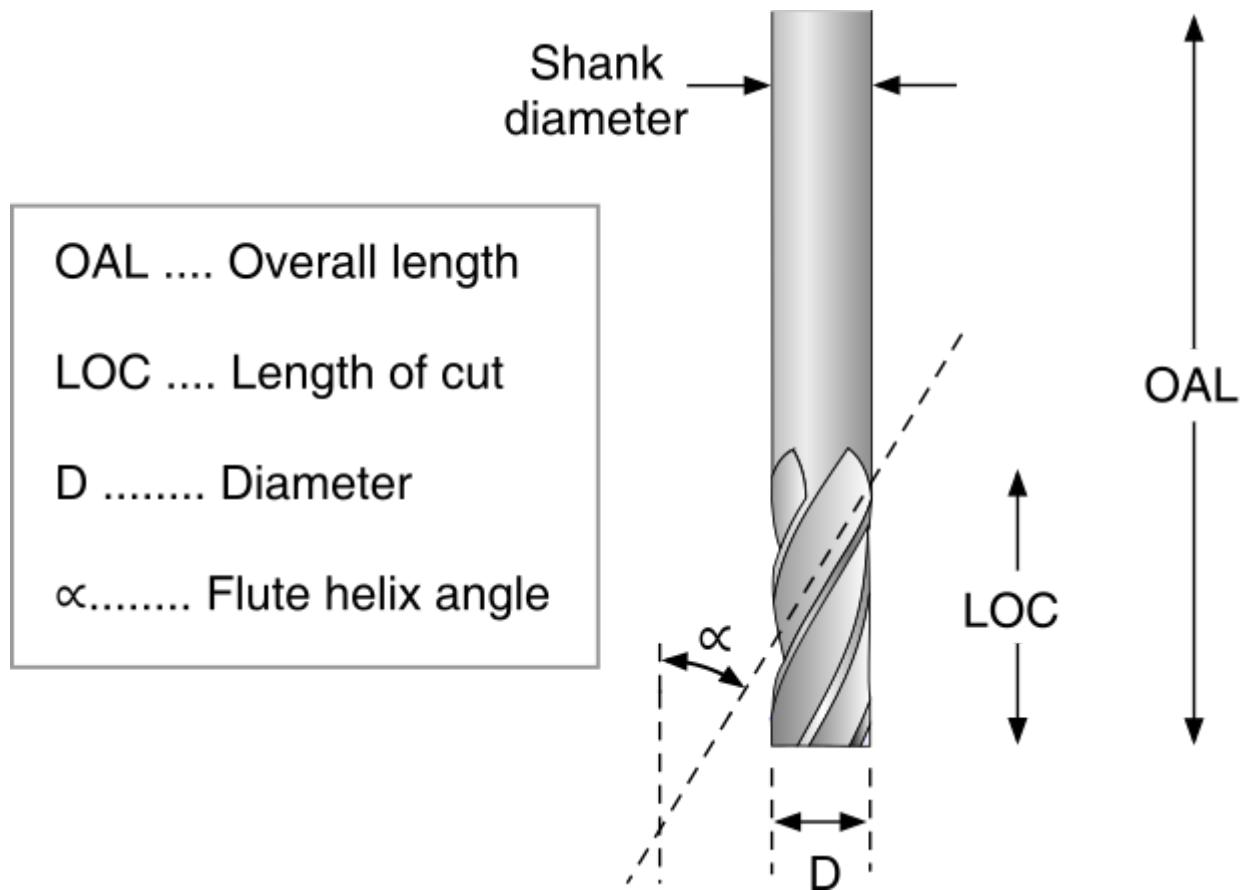
Cutters & collets

It's easy to be overwhelmed at first by the variety of cutters and their characteristics.

Roughly, an endmill is characterized by:

- its **type/geometry**, which relates to its intended use:
 - **square** endmills are used for cutting flat surfaces *e.g.*, pocketing and contouring, and for removing material quickly (roughing).
 - **ballnose** endmills are typically used for milling 3D surfaces.
 - **V-bits** are used to carve variable-depth grooves based on a 2D feature.
 - several other types are covered below but may require specific CAM features.
- the **diameter of its shank** (the part that goes in the collet/tool holder)
 - a large shank is better for reducing deflection.
 - the choice of shank diameter is constrained by the available collets.
- the **diameter of its cutting part**
 - the most common sizes used on the Shapeoko are 1/4" (6.35mm), 1/8" (3.175mm), 1/16" (1.5875mm), and 1/32" (~0.8mm), and their metric cousins (6, 3, 2, and 1mm).
 - the smallest feature size in a design determines the smallest endmill diameter needed.
 - smaller endmills are more fragile and more sensitive to runout (more on this below).
- the **length of its cutting part** (Length of cut / **LOC**), and its overall length (**OAL**).
 - a short LOC is better for stiffness, but obviously constrains the max depth of cut.
 - a long OAL provides better reach, at the expense of rigidity/deflection.
 - On the smallest diameter endmills, the cutting length is really short otherwise the tool would be extremely fragile and deflect too much
- the **number of flutes** (number of cutting teeth)
 - see [Feeds & speeds](#) for the impact of the number of flutes.
 - fewer flutes are better for chip evacuation.
 - more flutes are better for stiffness and finish.
- the **material** it is made of:
 - **carbide** is king these days for CNC milling, but it is brittle.
 - **high speed steel** (HSS) is cheaper and tougher, but more limited in speeds.
- its **coating**, if any:
 - Generally, no coating is needed for cutting wood and plastics.

- **ZrN** (Zirconium Nitride) coating is good for non-ferrous metals e.g., aluminium, brass, copper, titanium.
- **AlTiN** (Aluminium Titanium Nitride) coating is good for steel/ferrous-metals.
- whether it is **center cutting** or not:
 - most are, it means they have the ability to plunge into the material (vertically), like a drill bit does.
 - non-center cutting tools are more commonly used on manual power tools, think router bits. It is *possible* to use them on a Shapeoko, but it requires a very careful CAM design.
- the **helix angle**, if it's a spiral endmill:
 - depending on the helix angle, cutting forces will be oriented differently between the axial and radial directions: it's not really something you should worry about.
- the **direction** of rotation
 - in practice virtually all endmills are designed for a **clockwise** tool rotation (as seen from above the cut)



Square endmills

Square endmills are the workhorses of CNC milling. They come in several variants:

- **upcut**: the direction of the flute spiral is such that it pulls chips away from the cutting surface, thus is quite efficient at evacuating chips. It will produce a nice finish at the bottom of pockets, but can produce tear out on the top edges in some materials.
- **downcut**: it pushes chips downward when cutting, so is not efficient to evacuate them from the cutting area. It tends to do the opposite as an upcut, i.e. leaving produce a clean cut at the top edges of the cuts, but potential tear out at the bottom of pockets.
- **compression**: the flute geometry is such that it combines an upcut section at the bottom of the tool, and a downcut section at the other end of the cutting part, so for certain pocket depths it can give a nice finish both at the bottom and top of pockets.

Here's an example for a 3-flute, 1/4", upcut square endmill (codename "#201" on Carbide 3D store)



It also comes in a version coated with zirconium nitride (ZrN) which minimizes the risk of sticking when cutting non-ferrous metals:



Here's an example of a 1/4" endmill with only 2 flutes:



and here's a 1/4" one with a single flute (a.k.a. "O-flute"), that provides excellent chip evacuation:



Downcut endmills have their helical flute(s) oriented the other way, as in this 1/4" 1-flute downcut:



Here's another downcut, 2-flute, 1/8" endmill:



And then there are the funny looking **compression** endmills, that start with an upcut section at the tip, and have a downcut section higher up the shaft, here's a 1/4", 2-flute version:



and a smaller 1/8", 1-flute compression endmill:



Corner radius endmills

Square endmills have a weak point, and that's the sharp tip of each of their flutes. In demanding materials (like metal) this is likely to chip, so one can use **corner radius** (a.k.a. "bullnose") endmills instead: they behave very similarly to square endmills, but are much less prone to chip at the tip, hence can be used at more aggressive settings. Notice in the two pics below, the rounded corners (top: TiCN-coated, bottom: ceramic-coated)





Ballnose endmills

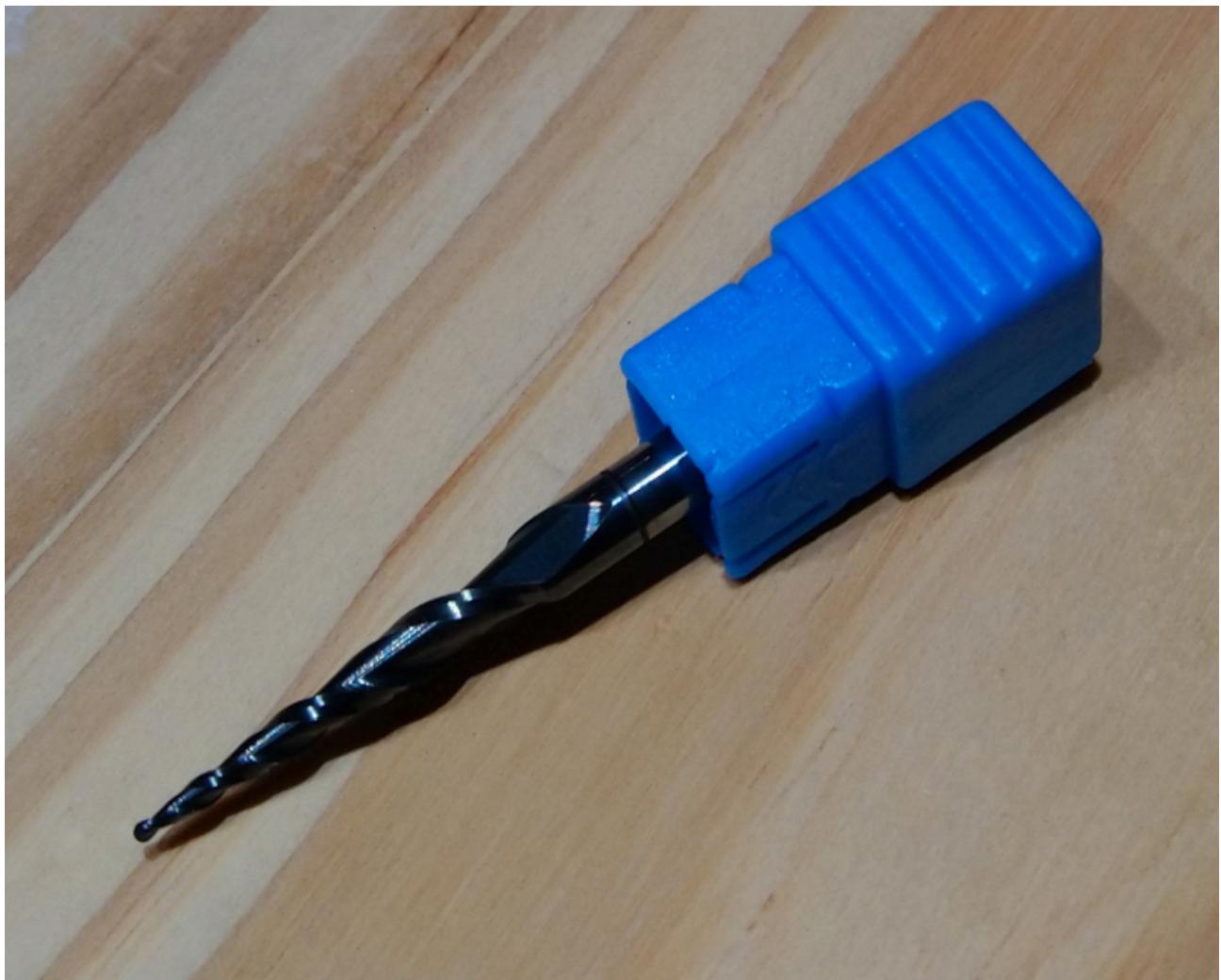
Ballnose endmills are better suited for machining smooth 3D curves (but are quite inefficient for machining flat pockets), here's a 2-flute 1/4" ballnose:



Here's a 0.032" ballnose:



For carving fine 3D details, a **tapered ballnose** endmill can be very useful: the tip can be very small, but the tapered flutes make it much more robust than a straight endmill of the same tip diameter, so it can support much more aggressive feeds and speeds. Here's a tapered endmill with a 0.5mm (0.02") ballnose tip:



V-bits

V-bits come with various angles, they are typically used to carve text or tiny grooves (more on this in the [Toolpaths](#) section):



Surfacing bit

They usually have a very large diameter, to be able to surface a wide area in one (very shallow) pass:



Engraving bit

Another specialized bit is used for engraving very small details, it's basically a very pointy one, and can be used for example to carve very narrow tracks on the top of a copper-clad PCB.



Diamond drag bit

The diamong drag bit has a tiny diamond on its tip, and is intended to be dragged across the surface of the material (with the router TURNED OFF!) to engrave a 2D pattern on the surface. The tip is usually spring-loaded, so that the tip is always in contact with the surface with a controlled pressure, while the bit moves around:



Thread milling bit

Special thread milling bits are used in conjunction with very specific spiral toolpaths, to thread the inside of a hole:



And there are many more, but these should cover a large part of the usecases/projects.

Collets

There is not much too say about collets, other than "they come in various sizes and quality". Here's a sample set of collets for the Makita router:

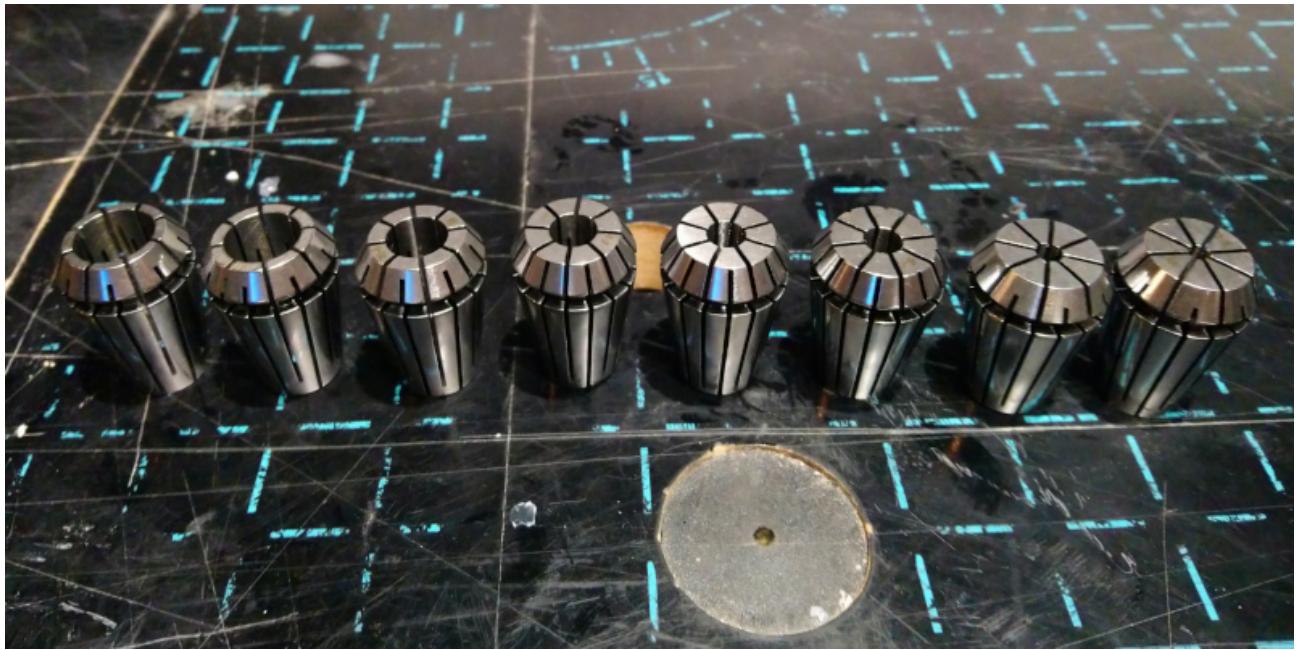


From left to right:

- 1/4" Makita collet (comes with the router in the US version)
- 6mm Makita collet (comes with the router in the European version) with a 6mm to 3.175mm adapter inserted
- 1/8" (3.175mm) cheap unbranded collet
- 1/4" (6.35mm) precision collet from Elaire Corp
- 1/8" (3.175mm) precision collet from Elaire Corp

i It is mandatory to use a collet size that matches the endmill shank diameter. Mistakenly using a 1/4" (6.35mm) collet for holding a 6mm endmill will likely end badly, with the endmill slipping in the collet during the job. The wiggle of a 6mm endmill in a 1/4" collet is hard to overlook, but beware.

Typical router collet sizes are 4 mm, 6 mm, 8 mm, 1/8", 3/16", 1/4", 3/8", so the range of available sizes is limited, especially toward the larger shank diameters. Spindle users have access to specialized collets such as the "ER"-style which has become an industry standard, that are available in many more sizes:

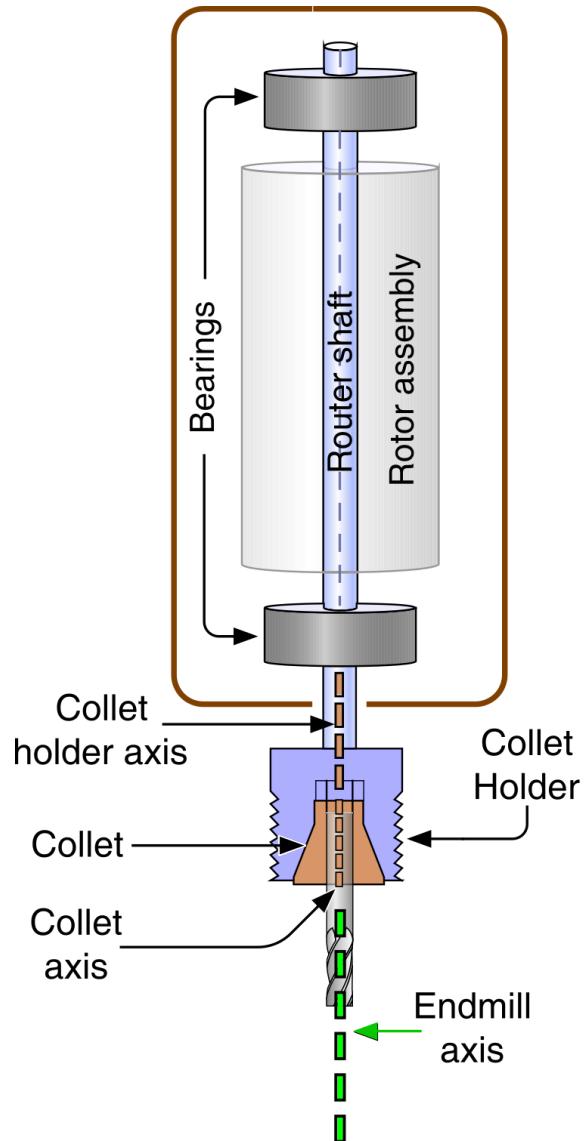


Using a **collet adapter/reducer** is generally not recommended as it tends to increase **runout**, but for most jobs it will still work fine.

Below is a short overview of what runout is, but overall this is not something new users have to worry about.

Runout

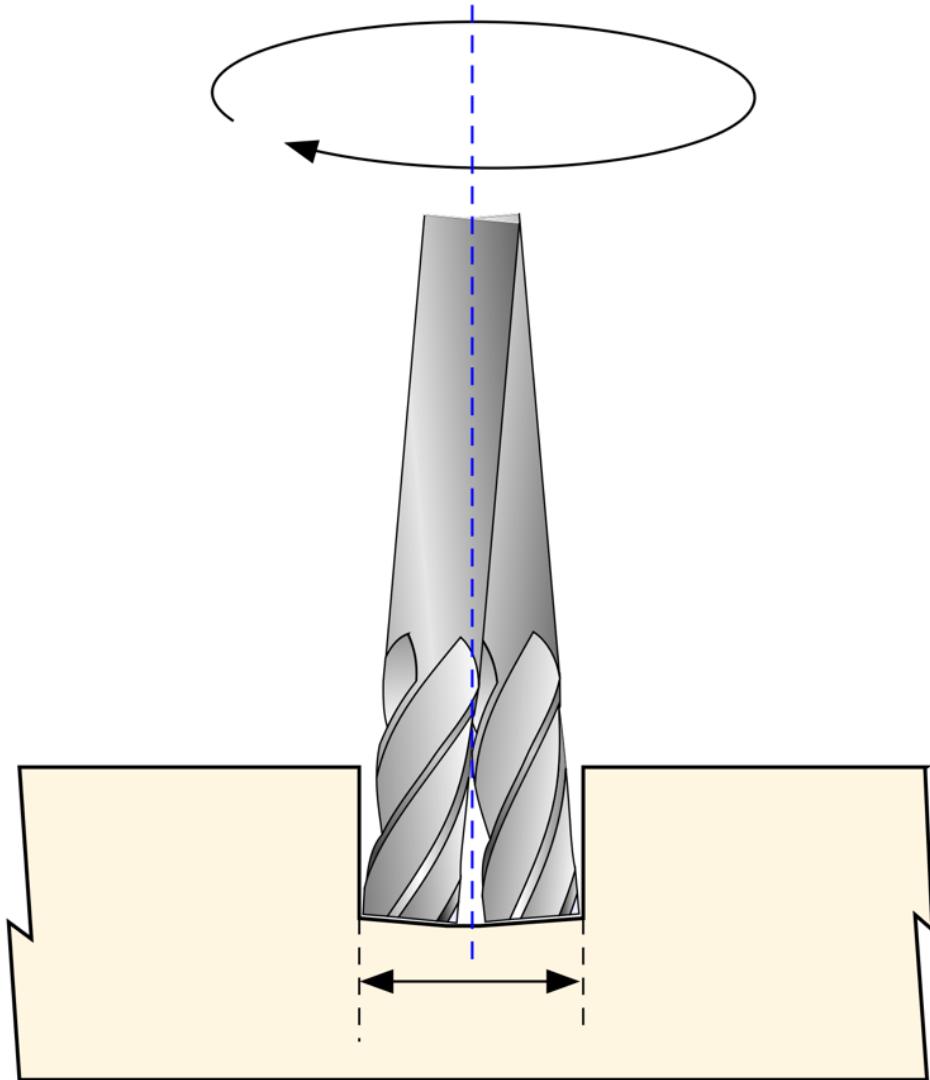
Ideally, the rotation axis of the router shaft (in black), the axis of the collet (in blue), and the axis of the endmill (in green) are perfectly aligned:



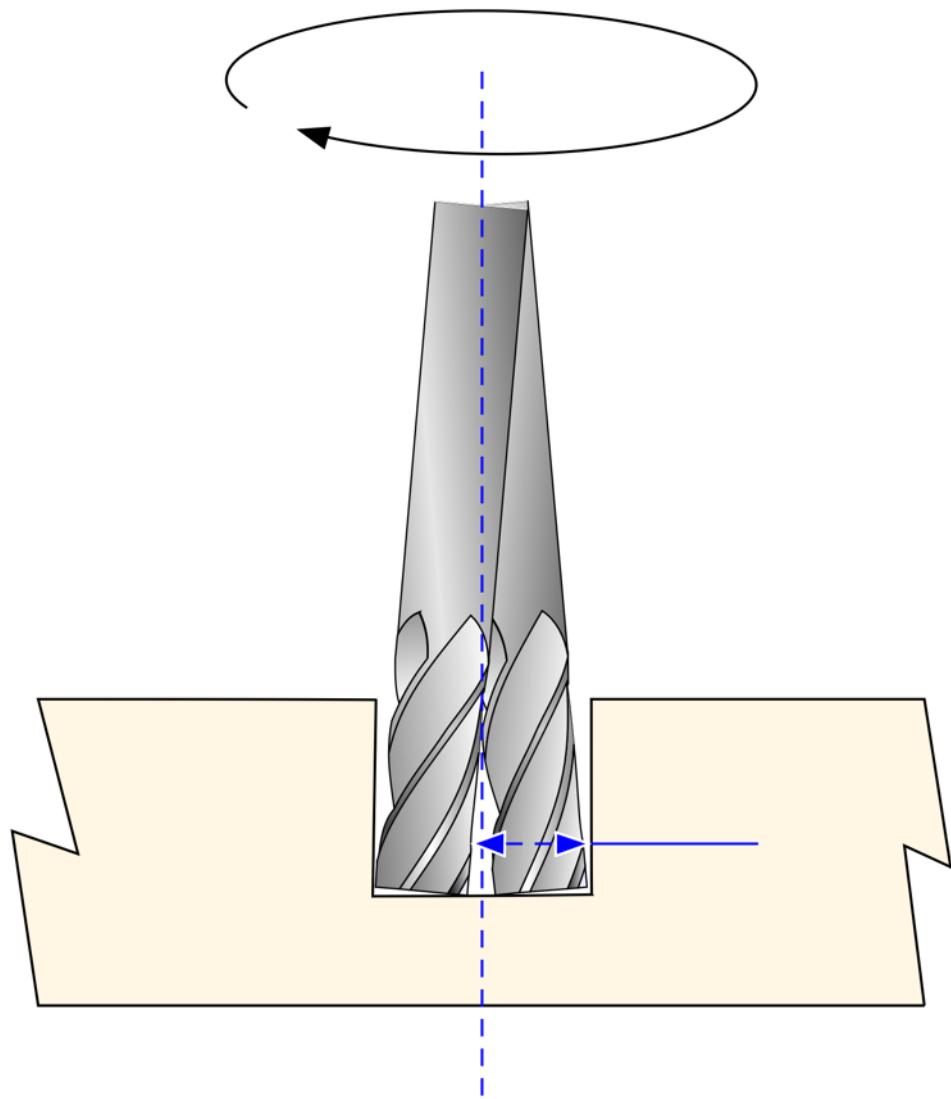
But in practice, manufacturing tolerances are such that there are small imperfections at all levels:

- the router shaft itself may not rotate perfectly on its axis
- the collet geometry may not be perfect, introducing a misalignment of the axis between the outer surface (attached to the router shaft) and the inner surface (holding the endmill)
- the endmill itself may not have a perfectly cylindrical shape

The end effect is that the movement of the endmill's tip in the material is the combination of the rotation along its own axis and other unintended deviations. Here's a very (very) exaggerated view of what happens when cutting a single slot with a lot of runout:



The expected width of the slot is the endmill diameter, but the actual width of the slot (effective cutting diameter) is the sum of the endmill diameter and the amount of deviation (runout). This deviation can be characterized by the maximal displacement measured at a given position on the surface of the endmill as it rotates (again, extremely exaggerated on the view below):



Refer to the runout section in [Dimensional accuracy](#) for details on how to measure runout, and correct it (or at least take it into account).

Endmills & collets starter set

A common question when buying the Shapeoko is "which endmills and collets should I get?"

- the answer of course is "it depends" (on the nature of your projects)
- the Shapeoko ships with a 1/4" square endmill (#201 from Carbide 3D store), and the router ships with a 1/4" collet: this is enough to get started and make a lot of beginner

projects actually.

- getting a couple of spare 1/4" square endmills is a good idea: sooner or later, the original #201 will wear out (or chip, or even break in the event of a really big mistake)
- the usual next step is to realize that 1/4" is too large to cut small features: getting a couple of 1/8" square endmills will not go to waste anyway.
 - this comes with the need to get a 1/8" collet, or at least a collet adapter.
- if you intend to use 3D toolpaths and curvy surfaces, get a ballnose endmill (1/4" or 1/8" or smaller depending on the size/precision of your target projects)
- get one V-bit: V-carving is quite easy and satisfying, you will probably want to try it, and it's a very common way to engrave text. They come in different shapes (angles), the most common ones are 60 degrees and 90 degrees. Make sure you invest in a good quality V-bit, it makes a big difference (while it is easier to get away with using cheap square endmills)
- if you intend to cut mostly plastics, do get an O-flute square endmill.
- if you intend to cut mostly aluminium, ZrN-coated endmills will help.
- a surfacing bit is useful to reduce the time for surfacing your wasteboard, but honestly not required in the starter set, a 1/4" endmill will do fine.
- the other types are very specific, so unless you know you will need them for sure, they can wait.

So my recommendation for a **starter pack** would be something like:

- 2 × 1/4" square endmills (2 or 3 flutes) to complement the one that ships with the machine
- 1 × 1/8" collet for your router (or at least a collet adapter to fit 1/8" in the 1/4" collet)
- 2 × 1/8" square endmills (2 flutes)
- 1 × 60° V-bit
- 1 × 90° V-bit
- (optionally for 3D work) 2 × 1/4" ballnose endmills
- (optionally for plastics) 1 × O-flute 1/4" square endmill + 1 × O-flute 1/8" square endmill
- (optionally for aluminium) a set of ZrN-coated 1/4" and 1/8" endmills

Feeds & speeds

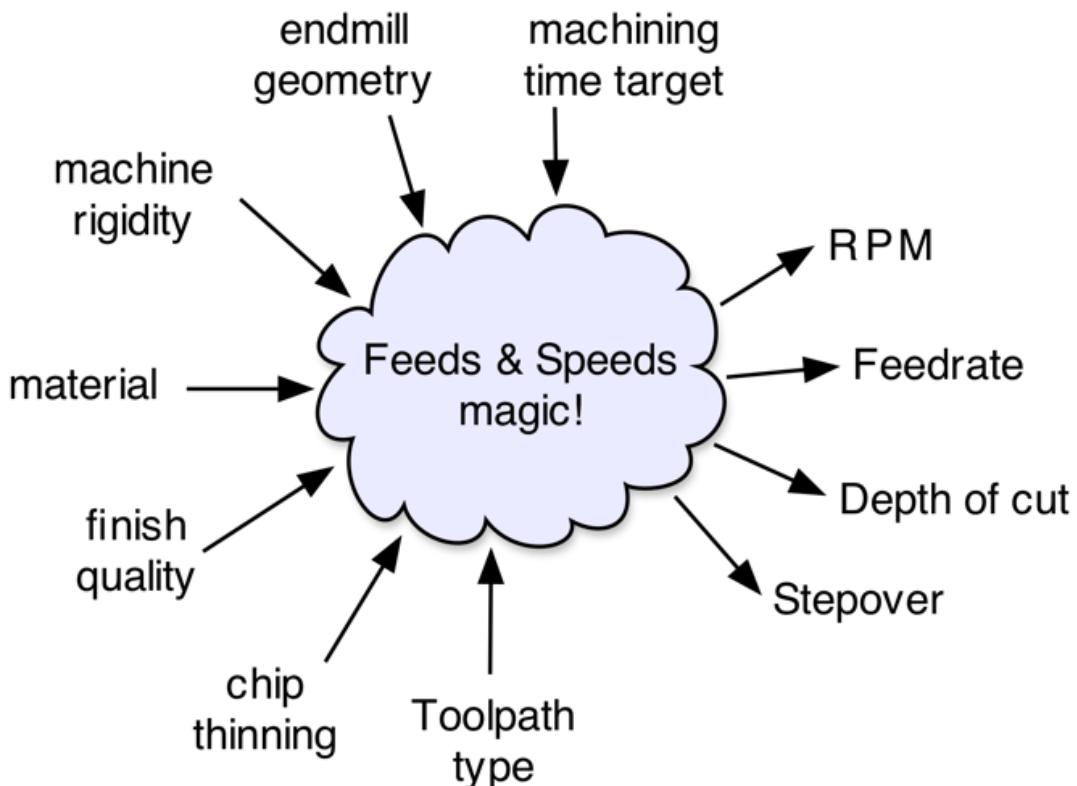
The whole "feeds & speeds" topic is arguably the most daunting part of learning CNC.

"**Feeds**" is feedrate, on some CNCs with a fixed tool and moving plate this is the speed at which the material is fed into the cutter, on a Shapeoko this is the speed of the gantry pushing the cutter into the material.

"**Speeds**" is the rotation speed of the endmill, i.e. RPM value.

"Feeds" and "Speeds" go hand in hand, what really matters is the *combination* of feedrate and RPM values for a given situation. Actually, they are also somewhat coupled with a number of other parameters (e.g. depth and width of cut), so "feeds and speeds" is often short for "all the cutting parameters".

When first starting CNC, selecting adequate cutting parameters feels a little bit like this:



Using proper feeds and speeds and depth/width of cut values is important to :

- get a good quality of the cut (e.g. surface finish, dimensional accuracy)
- increase tool life (i.e. keep tool wear to a minimum), or at least avoid tool breakage.
- avoid/minimize chatter (the horrendous sound heard when the endmill/machine vibrates while cutting through the material)
- optimize material removal rate (e.g. how long it takes to complete the cut)

While there is definitely a good amount of experience (and experimentation) involved in finding the perfect feeds and speeds for any given situation, there are a few underlying principles that are worth understanding for two reasons:

- to figure out reasonable values to **start** from, when a new situation shows up for which you cannot find any predefined recommended values.
- to be in a position to understand how to **tune** the cutting parameters to achieve the desired result.

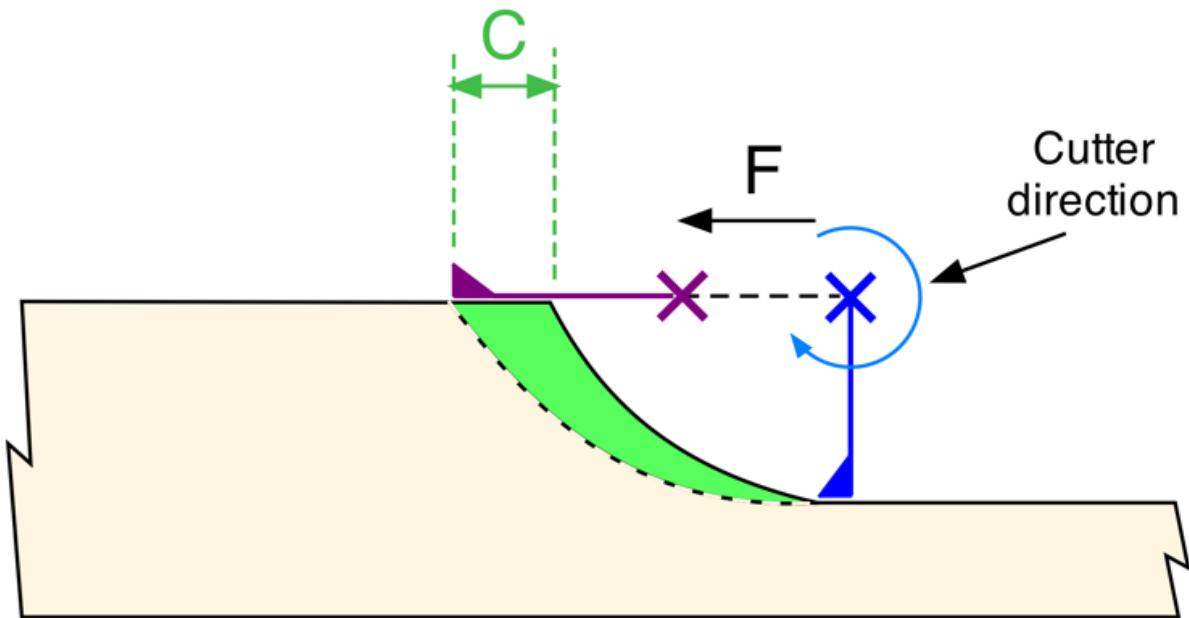
 This section includes a little math (nothing too fancy), but not to worry: while it is important to understand the *dependencies* between the cutting parameters, calculators will take care of all those computations for you.

Before diving into the relation between feedrate, RPM, and the other parameters, let's check how the tool cuts into the material.

Conventional milling

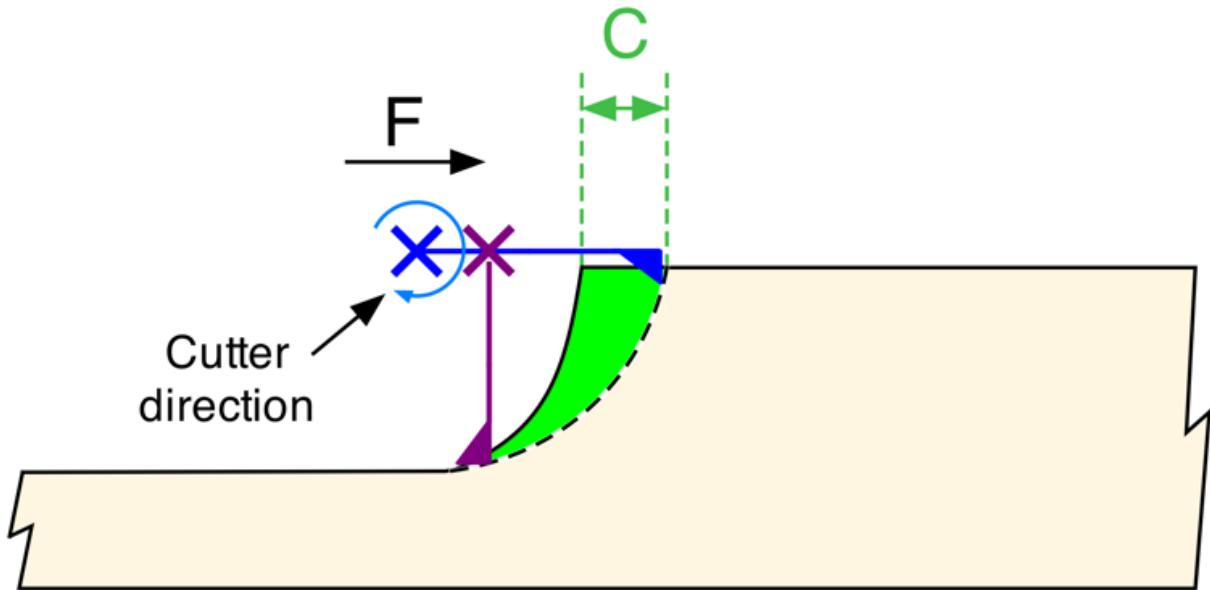
In so-called "conventional" milling, the direction of the endmill movement is such that the cutting edges bite from the inside to the outside of the material. In the sketch below, imagine the blue triangle represents one cutting edge of the endmill. If the blue cross is the position of the center of the endmill when this cutting edge starts biting into the material, and if the endmill is moving into the material at feedrate F , then a little bit later the endmill center is at the position of the purple cross, and the cutting tooth has rotated and gone out of the material. The resulting chip of material that was cut during that time is the green part.

It starts out very thin, and gradually increases in thickness. The maximum thickness (noted "C" below) happens when the cutting edge exits the material. It is typically called the "feed per tooth" or "chipload per tooth", or usually just "**chipload**", and this is the cornerstone of feeds and speeds.



Climb milling

"Climb" milling is when the direction of the endmill movement is such that the cutting edges bite from the outside to the inside of the material. In that situation, a cutting edge first bites a large chunk of material (blue position), and as the endmill rotates and moves to the right at feedrate F , the cut gets thinner, until the tooth has nothing left to cut (purple position). The resulting chip (in green) has a similar shape to that in conventional milling, and again the max thickness of the chip is the chipload.



Chiploads

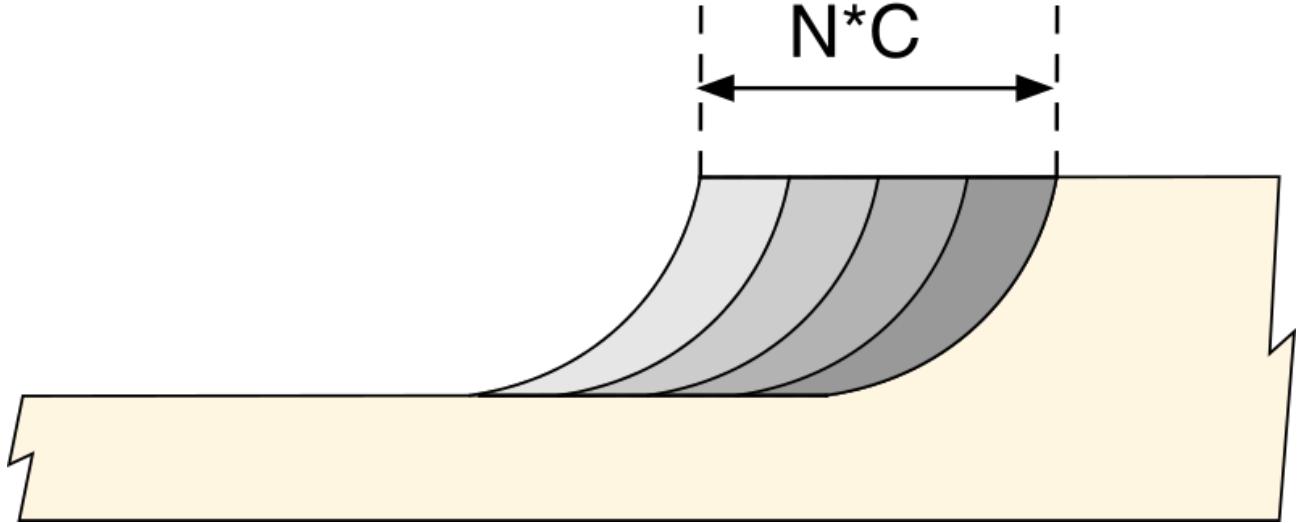
So why does chipload matter so much ?

A large chipload requires a lot of router torque and machine rigidity, and each endmill has recommended chipload limits from the manufacturer anyway (i.e. there's always a limit to the size of the bite you can take, whether you're a squirrel or a white shark).

A too small chipload is actually worse: since the cutting edges are not infinitely sharp, at some point instead of slicing into the material, the cutting edges will mostly rub against the surface, and then "heat happens" and this is very bad for the quality of the cut and for tool life. In CNC, rubbing is the enemy.

So this is a Goldilocks situation: the chipload must be high enough to avoid rubbing and overheating of the endmill, and small enough to be within the torque/rigidity limits of the machine and the endmill's rated maximums.

Each flute contributes in turn to removing material during one revolution of an endmill. If we sketched N successive bites that the endmill takes into the material, it would look something like this:



If the endmill has N flutes, one revolution will cut N chips, i.e. a length of $N \times \text{chipload}$ of material. Since the endmill revolves at RPM turns per minute, in one minute a length of $N \times \text{chipload} \times \text{RPM}$ will have been cut. And the distance being cut per minute is exactly the definition of feedrate, therefore $\text{Feedrate} = N \times \text{RPM} \times \text{Chipload}$, which also means:

$$\text{Chipload} = \frac{\text{Feedrate}}{Nb_Flutes \times \text{RPM}}$$

This relation is quite intuitive:

- for a given endmill and RPM, the faster the feedrate the larger the chipload.
- for a given feedrate and RPM, an endmill with more flutes will cut thinner chips.
- for a given feedrate and endmill, the faster the endmill rotates the thinner each chip will be.

The important takeaway here, is that there are many possible combinations of feedrate, endmill type, and RPM to reach a given chipload. Say you are using a feedrate of 1000mm/min (39ipm), and a 3-flute endmill at 10,000RPM. Given the formula, you may just as well use a 2-flute endmill at 15,000RPM, or keep the 3-flute endmill but spin it at 20,000RPM while increasing feedrate to 2000mm/min (79ipm), the chipload thickness would be the same:

$$\frac{1000}{3 \times 10,000} = \frac{1000}{2 \times 15,000} = \frac{2000}{3 \times 20,000} = 0.033\text{mm}$$

- (i) This also means that if your CAM tool comes up with feedrates or RPMs that are not in the range of your machine's abilities (e.g., recommended RPM lower than the minimum RPM of your router), you can just scale both RPM **and** feedrate values by any factor, and it will still provide the same chipload.

For a given chipload, some combinations are still better than other mathematically-equivalent ones though (more on this below). Since the feedrate/RPM combination is derived from the desired chipload value, let's first have a look at what the range of acceptable chipload values is for the Shapeoko.

Shapeoko chiploads guideline

For the **minimum** chipload value to avoid rubbing, there is a large consensus in the CNC community that a value of **0.001"** (0.0254mm) is a good absolute lower limit guideline, at least for 1/4" endmills and larger. It may need to be lowered to 0.0005" for 1/8" and smaller endmills.

- (i) This is assuming you are using a **sharp** cutter. You should never use a dull cutter anyway, if you do you may end up rubbing even at this 0.001" chipload.

The **maximum** reachable chipload depends on a lot of things, but mostly:

- the **hardness of the material** being cut
- the **type and diameter of the endmill** (smaller teeth need to take smaller bites: the maximum chipload for a given endmill scales linearly with its diameter)

- the **toolpath** used (how wide/deep the cutter is engaged) and the **rigidity of the machine**: it is quite easy to forget that the Shapeoko is not as rigid as industrial CNCs, so endmill manufacturers recommendations may not be directly suitable for the Shapeoko. Any mechanical mod of the machine also impacts the max chipload capability.

The Shapeoko's limits must also be accounted for: the absolute maximum theoretical chipload on a stock Shapeoko would be reached when using a single-flute endmill at the lowest RPM (10,000RPM on the Makita router) and at the fastest feedrate of 200 inch per minute, and that would be $200/(1 \times 10,000) = 0.02" = 0.5\text{mm}$

So all chiploads should be somewhere between 0.001" and 0.02".

(i) Technically, the feedrate can go beyond 200ipm, if the associated GRBL limits parameters are set to a higher limit. In practice, feedrates above 200ipm are used for rapids only, there are very little usecases where actually cutting at higher than 200ipm would be useful.

The following is an (arguable) table I am using as a personal reference, which I derived from analysis of a large number of feeds and speeds settings shared in the Shapeoko community, as well as my own experimentations. Do not take it for granted, start above 0.001" and increase it incrementally (by keeping RPM constant and increasing feedrate) to find the limits for your machine and for a given material. Higher chiploads are definitely possible (but may not be desirable)

	Soft plastics	Soft wood & hard plastics	Hard wood & metals
1/16"	0.002"-0.003"/ 0.05mm-0.075mm	0.001"-0.0015" / 0.025mm-0.04mm	0.0005"-0.001"/ 0.013mm-0.025mm
1/8"	0.002"-0.005" / 0.05mm-0.13mm	0.001"-0.0025" / 0.025mm-0.063mm	0.0005"-0.001"/ 0.013mm-0.025mm
1/4"	0.004"-0.01" / 0.1mm-0.254mm	0.001"-0.005" / 0.025mm-0.127 mm	0.001"-0.002"/ 0.025mm-0.05mm

 To keep this guideline table simple, I chose to only divide woods into "soft" and "hard" categories, and this labeling is not the correct definition either (which relates to whether the tree *seeds* have a hard or soft shell). I mean it in the "wood hardness" way, and there is a useful **Janka** scale that measures that. @Hooby on the forum consolidated a nice list of Janka hardness values for many types of wood, which I include here for reference. The Janka threshold for "hard" vs. "soft" is highly debatable, but a value of 1000 seems reasonable to steer the chipload selection.



[@Hooby's Janka wood hardness list](#)

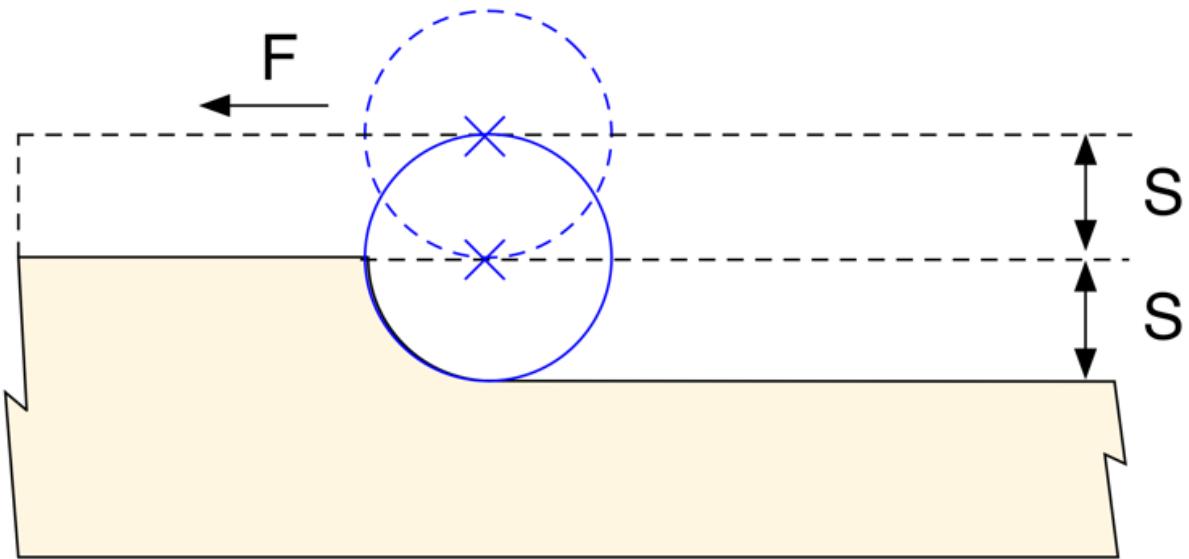
Wood_Hardness for CNC.xlsx - 19KB

Now we have to take a little detour and talk about stepover, because it impacts the *effective* chipload.

Stepover/WOC/RDOC

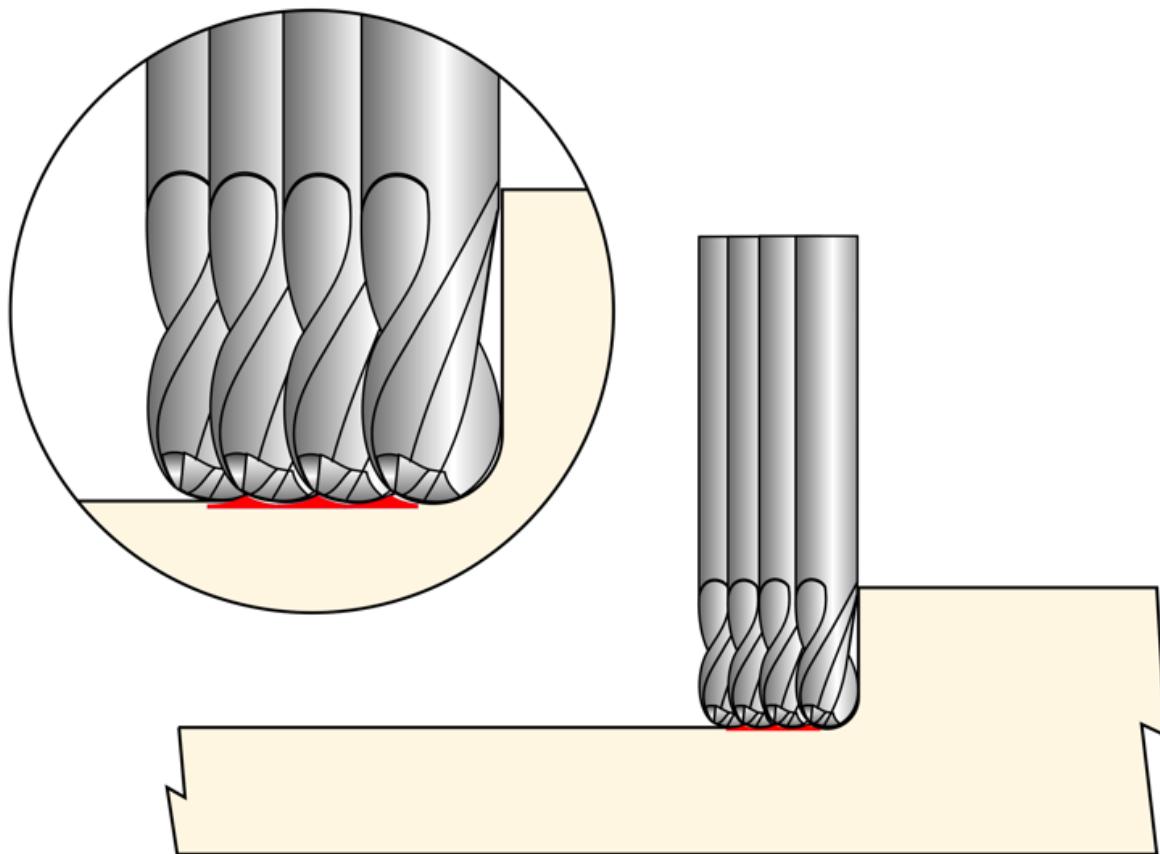
"**Stepover**" refers to the offset distance of the endmill axis between one cutting pass and the next one, which also translates into how much new material is being removed by the endmill, or how much radial engagement is put on the endmill. It's also called Width of Cut (**WOC**) or Radial Depth of Cut (**RDOC**)

In the example below, the stepover S is 50% of the endmill diameter:



The larger the stepover, the larger the force on the endmill. Cutting passes with a small stepover are better for surface finish quality, while passes with large stepover obviously reduce overall cutting time since fewer passes are required to cut a given amount of material. And then **depth** of cut will also come in the picture (more on this later).

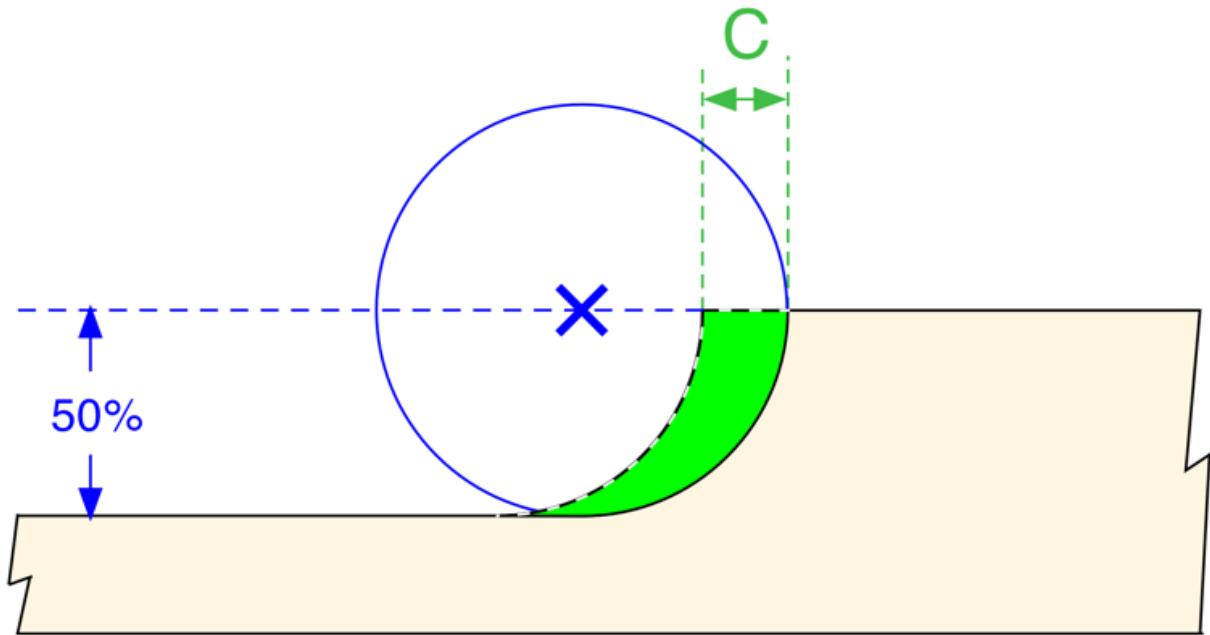
As a side note, for ball endmills, stepover value influences surface finish quite a lot. Consider the following sketch of a side view showing multiple passes:



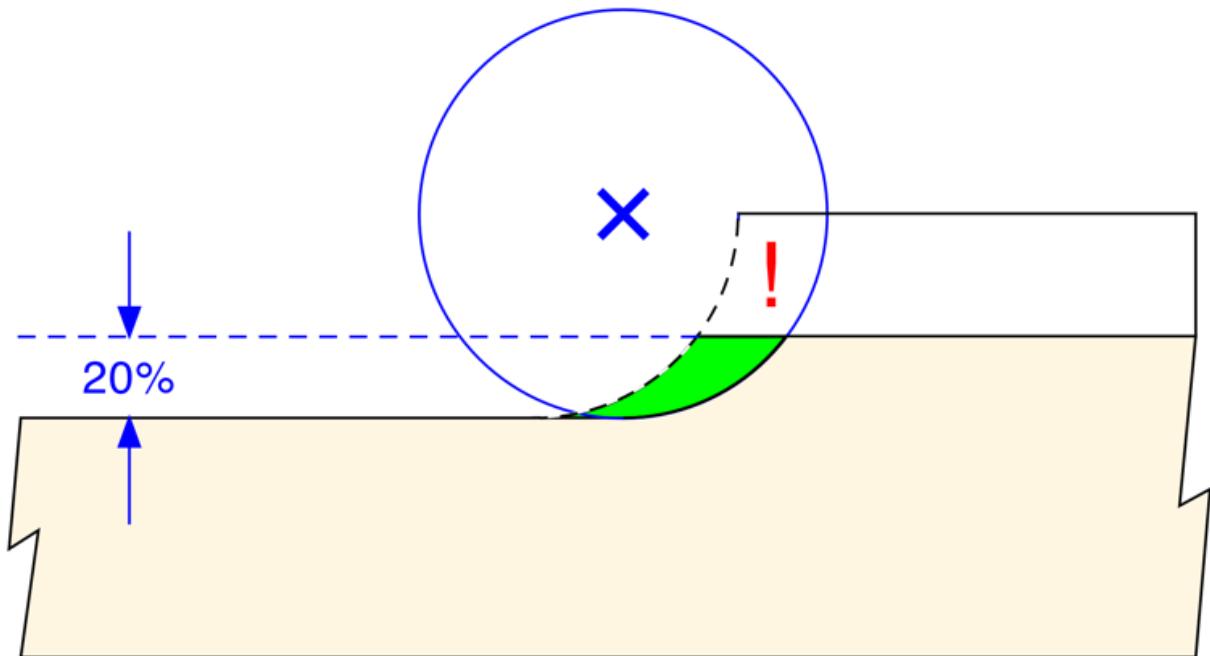
Due to the geometry of the endmill tip, scallops of residual material will be left at regular intervals on the bottom surface. These will be more or less visible depending on how well the material can hold small details (a 20% to 33% stepover should be small enough for wood, while it could need to be lowered down to 10% stepover for metal)

Chip thinning

The chipload values discussed earlier assumed that the stepover is at least 50% of the endmill diameter:

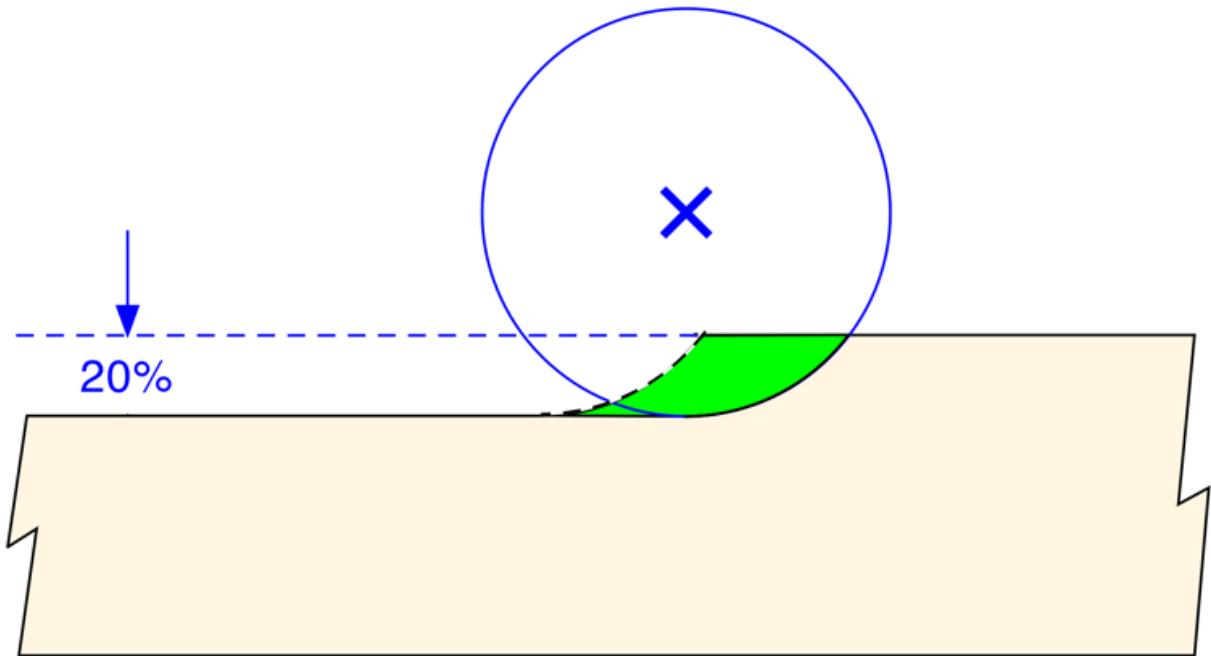


Now consider what happens if the stepover is lower than 50% of the diameter, say 20% only:



For the same RPM and feedrate, the *actual* chip is smaller, its maximum thickness is smaller than targeted, so there is again a risk of rubbing, or at least of sub-optimal heat removal.

The solution is to artificially target a higher chipload value (all other parameters staying the same), such that the actual size of the chip is increased to approximately what it *would* have been if the cutter were engaged at 50%:



The formula to determine this higher, adjusted target chipload is:

$$Chipload(\text{adjusted}) = \frac{\text{Diameter}}{2 \times \sqrt{(\text{Diameter} \times \text{Stepover}) - \text{Stepover}^2}} \times Chipload$$

- ⓘ For basic toolpaths, the stepover is often in the 40% to 50% range, and then you can just ignore chip thinning altogether. Where chip thinning really matters is for adaptive clearing toolpaths, that typically use small stepovers (more on this in the [Toolpaths](#) section)

- (i)* If we wanted to be pedantic, the term *chipload* should be used for the case where there is no chip thinning, while the term *chip thickness* should be used to name the adjusted/effective chipload after chip thinning is taken into account.

Choosing RPM and Feedrate

At this stage, the material is known, the endmill geometry is known, chip thinning is accounted for, which gave us an adjusted target minimal chipload. The remaining part is to chose a specific combination of RPM and feedrate values that together will produce this chipload, following the formula described earlier.

In theory, there are two options: selecting a feedrate value and solving for the associated required RPM value, or selecting an RPM value and solving for the associated feedrate.

In practice, the latter is done. The main reason is that the traditional way to determine feeds and speeds (especially when cutting metal) is to start from the required **SFM** (Surface Feet Per Minute): this is the linear speed of the edge of the cutter, and it should be within a certain range depending on the material and the endmill. And to achieve a given SFM for a given endmill diameter, only the RPM needs to be determined:

$$RPM = \frac{SFM(\text{in feet per min})}{0.262 \times \text{endmill_diameter}(\text{in inches})}$$

In practice, for most of the materials cut on a Shapeoko, there is a wide range of acceptable SFMs, so RPM could initially be chosen pretty much anywhere within the router's RPM limits (10k to 30k for the Makita/Carbide router, 16k to 27k for the Dewalt router, and typically a few hundred to several tens of kRPM for spindles)

- Low RPMs are quieter (significantly so with a router), but induce higher forces on the cutter (more on this later)

- High RPMs induce lower cutting forces and generally provide better finish quality, but will also require higher associated feedrates to maintain a correct chipload: feeding faster can be a little scary at first, and leaves less time to react should anything go wrong.

A rule of thumb is therefore to set RPM to "**the maximum value you can tolerate and feel comfortable using**", and then determine the associated feedrate to get the right chipload.

Example:

- Material is hard wood and endmill is a 3-flute 1/4" => the chipload table recommends up to 0.002"
- Say we use a 25% radial depth of cut / stepover, i.e. 25% of 50% of 1/4" = 0.03125", so adjusted chipload is:

$$\frac{0.25}{2 \times \sqrt{(0.25 \times 0.03125) - 0.03125^2}} \times 0.0014 \approx 1.5 \times 0.002 = 0.003''$$

- The ideal setting would be to max out the RPM, say 24,000 (to take an example that is reachable on the Makita, DeWalt, and common spindles). The required feedrate would then be :

$$Feedrate = Chipload \times Nb_Flutes \times RPM = 0.003 \times 3 \times 24000 = 216 \text{ ipm}$$

- That is above the default capability of the Shapeoko (200ipm), it would be scarily fast for cutting hard wood, and 24,000 RPM may sound too loud to your taste anyway. Let's say we decided to go for 16,000 RPM instead, the required feedrate would become:

$$0.003 \times 3 \times 16000 = 144 \text{ ipm}$$

- If going 144ipm still *feels* a little fast, it is possible to obtain the same chipload at lower RPM and lower feedrate, e.g. 12,000RPM and 108ipm, at the expense of higher cutting

- forces (which or may not be a problem, see power analysis section later below)
- Alternately it is also possible to lower the feedrate by targetting a smaller chipload while ensuring it is still at least at the minimum recommended value of 0.001", and assuming you are using a sharp enough cutter:
 - To get a 0.001" effective target chipload, the adjusted target chipload would become 0.0015"
 - the feedrate would then be $0.0015 \times 3 \times 16,000 = 72\text{ipm}$

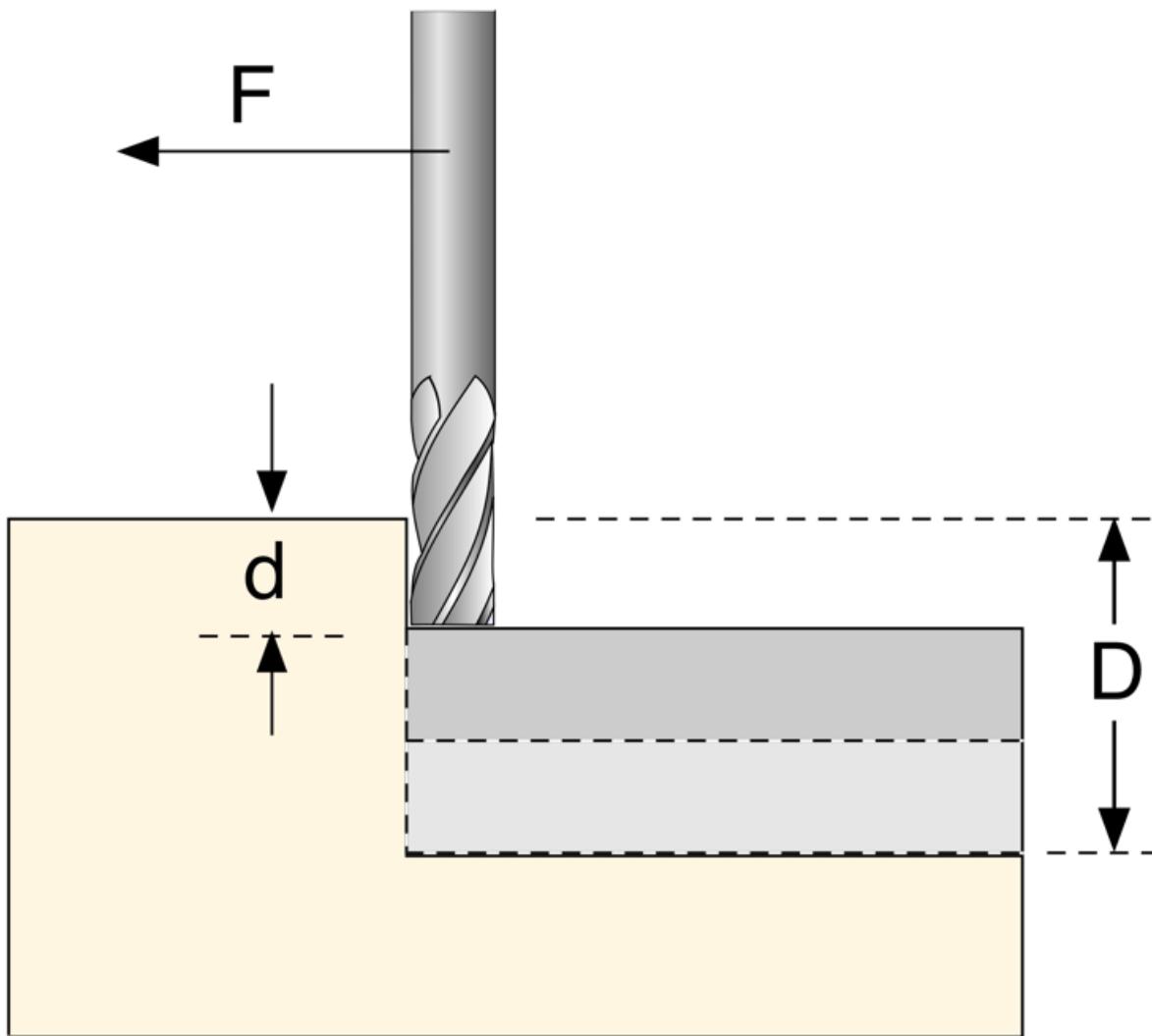


Some usecases call for the use of an O-flute endmills: this will probably mean reducing the feedrate and/or increasing the RPM to maintain a proper chipload.

Choosing DOC & WOC

Depth of Cut (DOC) a.k.a. Axial Depth of Cut (ADOC) a.k.a. Depth Per Pass, is how deep into the material the endmill will cut, along the Z axis. For a given feedrate and RPM, the deeper it is the larger the forces on the endmill.

Multiple cutting passes at depth of cut d will be required to cut down to a total pocket depth of D :



DOC is just as important as feeds & speeds to achieve a good cut, yet surprisingly there is much less information about how to determine its value, compared to the abundance of feeds and speeds charts.

The reason is probably that while there are mathematical recipes to choose feedrate and RPM for a given endmill geometry, the achievable DOC is much more tightly linked to the specific machine you are using, and specifically its rigidity and power.

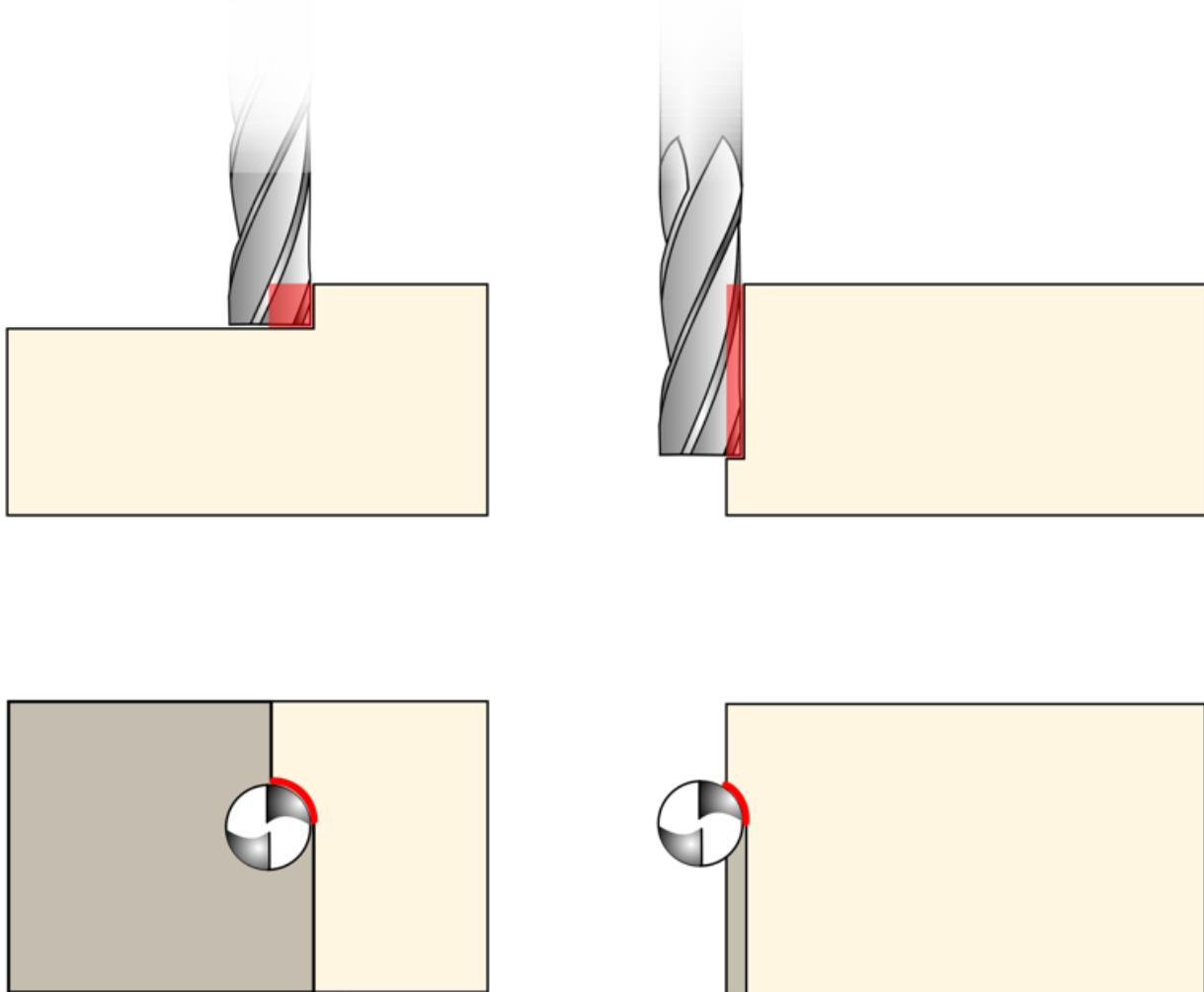
The Shapeoko is not as rigid nor as powerful as pro CNC machines, so DOC recommendations for these machines need to be dialed back, even when using perfect feeds and speeds.

There is a strong dependency between DOC and WOC: since cutting forces increase with both DOC and WOC, you cannot cut very deep while using a very large stepover, that would

put too much effort on the endmill. So the two choices are:

- large WOC but small DOC
- small WOC but high DOC

These two situations are illustrated below:



The **small WOC, high DOC** approach is much preferable, as it spreads the heat and tool wear much more evenly along the length of the endmill. However, it requires specific toolpath strategies (e.g. to initially clear material down to the required depth, to allow small WOC to be used for the rest of the cut), this is covered in the [Toolpaths](#) section. This is a very popular approach when cutting metals on the Shapeoko, but its benefits apply to other materials too.

The **large WOC, small DOC** approach only ever uses the tip of the endmill, so that part will wear out quickly while the rest of the endmill length of cut remains unused. But it is still a very common approach for pocketing and profile cuts on the Shapeoko, and it has simplicity going for it.

Unlike chiploads that NEED to be in a specific range to get good cuts, the situation is easier for DOC and WOC: you can just start with small, conservative values and then increase them to find the limit for your machine/endmill/material combination.

For the "wide and shallow" cut scenario (large WOC, small DOC), I like to start in this ballpark:

For DOC:

- **5% to 10%** of the endmill diameter for metals *e.g. aluminium*
- **10% to 50%** of the endmill diameter for **softer materials**

For WOC:

- **40% to 100%** of the diameter of the endmill for roughing
- **5% to 10%** for finishing passes



Don't go below 5% DOC, or you may get rubbing just like when chipload is too low

For the "narrow and deep" cut scenario (small WOC, large DOC), I like to use these guidelines:

For DOC:

- **100% to 300%** of the endmill diameter

For WOC:

- **10% to 25%** stepover for roughing

- 5% to 10% for finishing passes
- possibly even less for the hardest materials (e.g., 4% for steel)

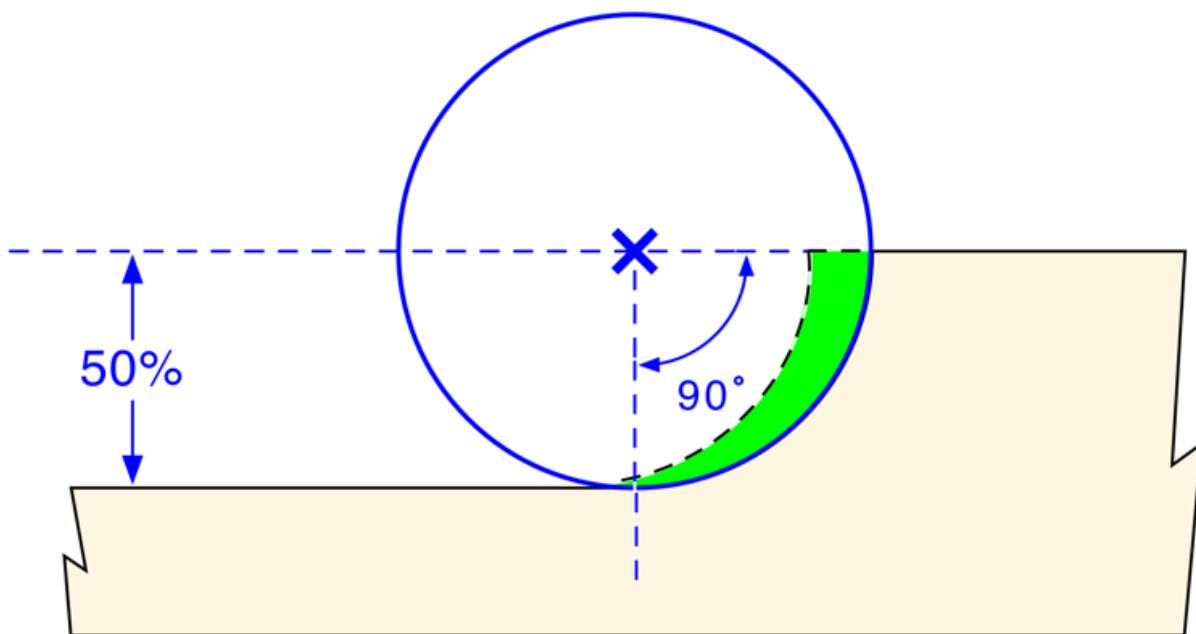
i If you go for narrow and deep (and you should!), given the small WOC values you will definitely need to take chip thinning into account. Also, check out **adaptive clearing** in the [Toolpaths](#) section, that goes hand in hand with high DOC and small WOC.

The figures above provide a ballpark for DOC and WOC, taking into account two specific cases: slotting, and corners.

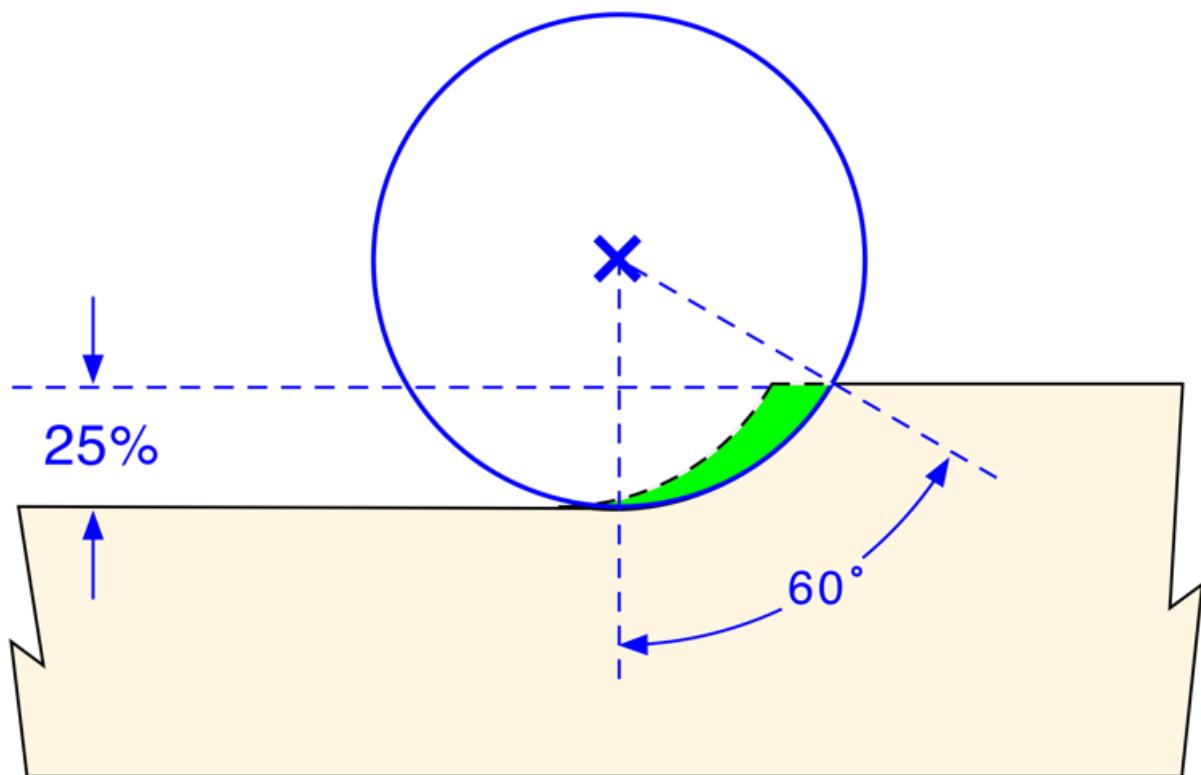
Slotting

Depending on the stepover, the portion of the endmill that will be engaged in the material, a.k.a. the Tool Engagement Angle (**TEA**), will be different:

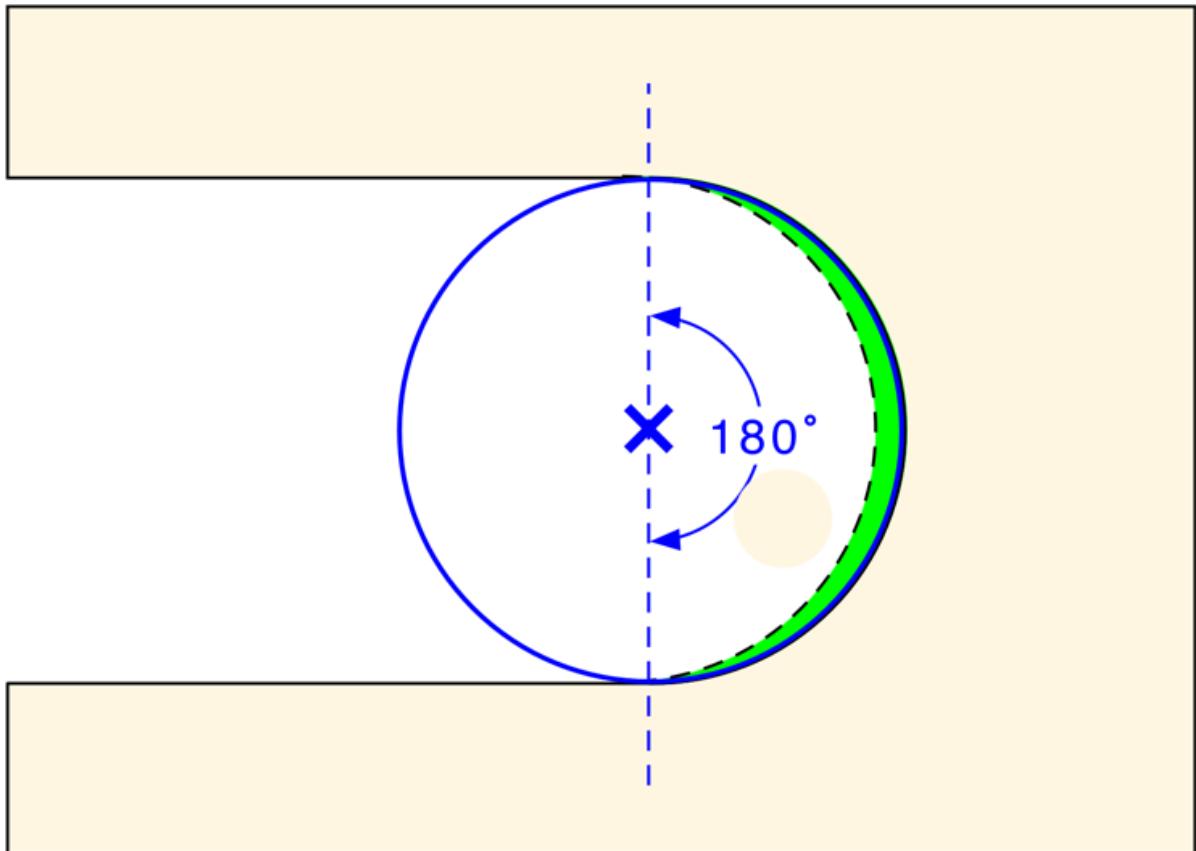
For a 50% stepover, the TEA will be 90°:



For a smaller stepover, say 25%, the TEA will be reduced (in this case to 60°):



Slotting is a different story: half of the endmill is engaged at all times, so the TEA is 180°:



The force on the endmill will be much higher than when cutting at 90° TEA, so the max achievable chipload/DOC combination for a given machine/endmill/material is lower. The recommended chipload/DOC values mentioned above include some margin to take this effect into account to some extent.

The other side effect of slotting is that chip evacuation is not as good: the flutes are in the air only 50% of the time, so the chips that form inside them have less time/fewer opportunities to fly away. Deep slotting is notorious for causing issues when chips cannot be evacuated quickly enough.

Bottomline: slotting is hard on the machine, so you may have to:

- limit DOC to the low end of the range of values
- limit feedrate (chipload...)
- optimize chip evacuation by using an endmill with a lower number of flutes, and/or a good dust shoe or blast of air

-  Or, you can take a different approach and *avoid* slotting altogether, by using smarter toolpaths. See adaptive clearing and pocketing in the [Toolpaths](#) section!

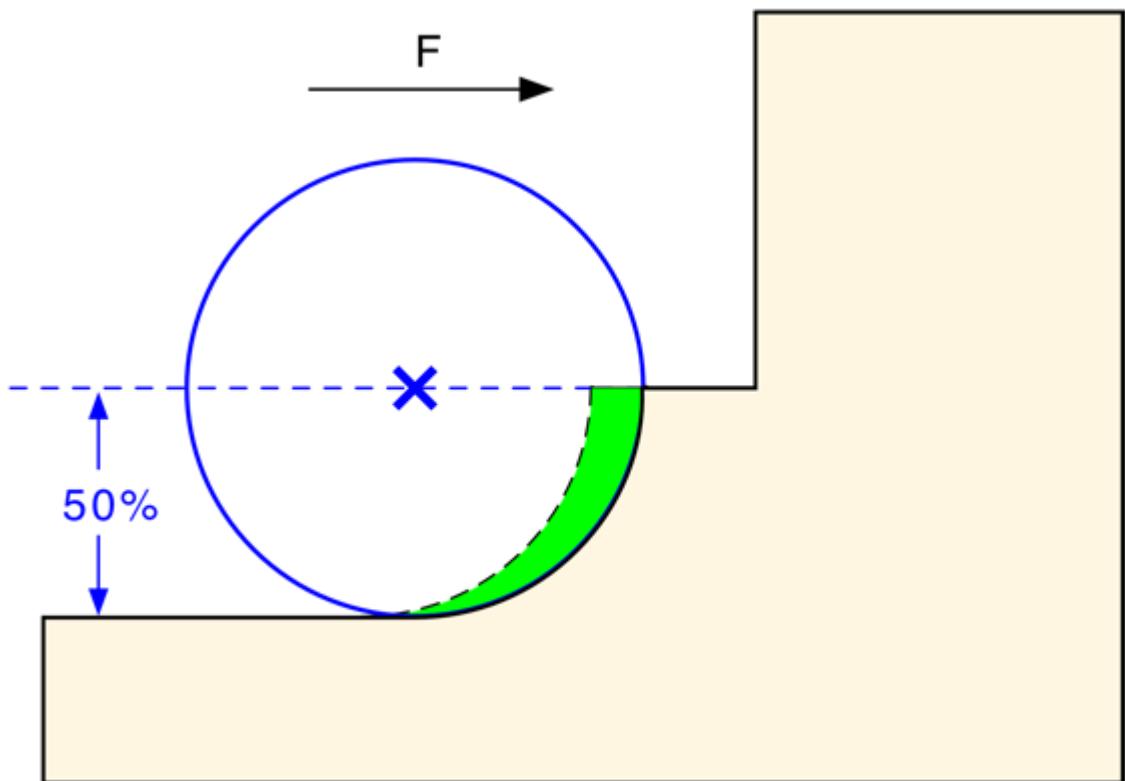
Not that you will ever need to use it, but for the math-inclined among you, here's the equation to compute TEA from stepover value:

$$TEA = \cos^{-1}\left(1 - \frac{\text{stepover}}{0.5 \times \text{endmill_diameter}}\right)$$

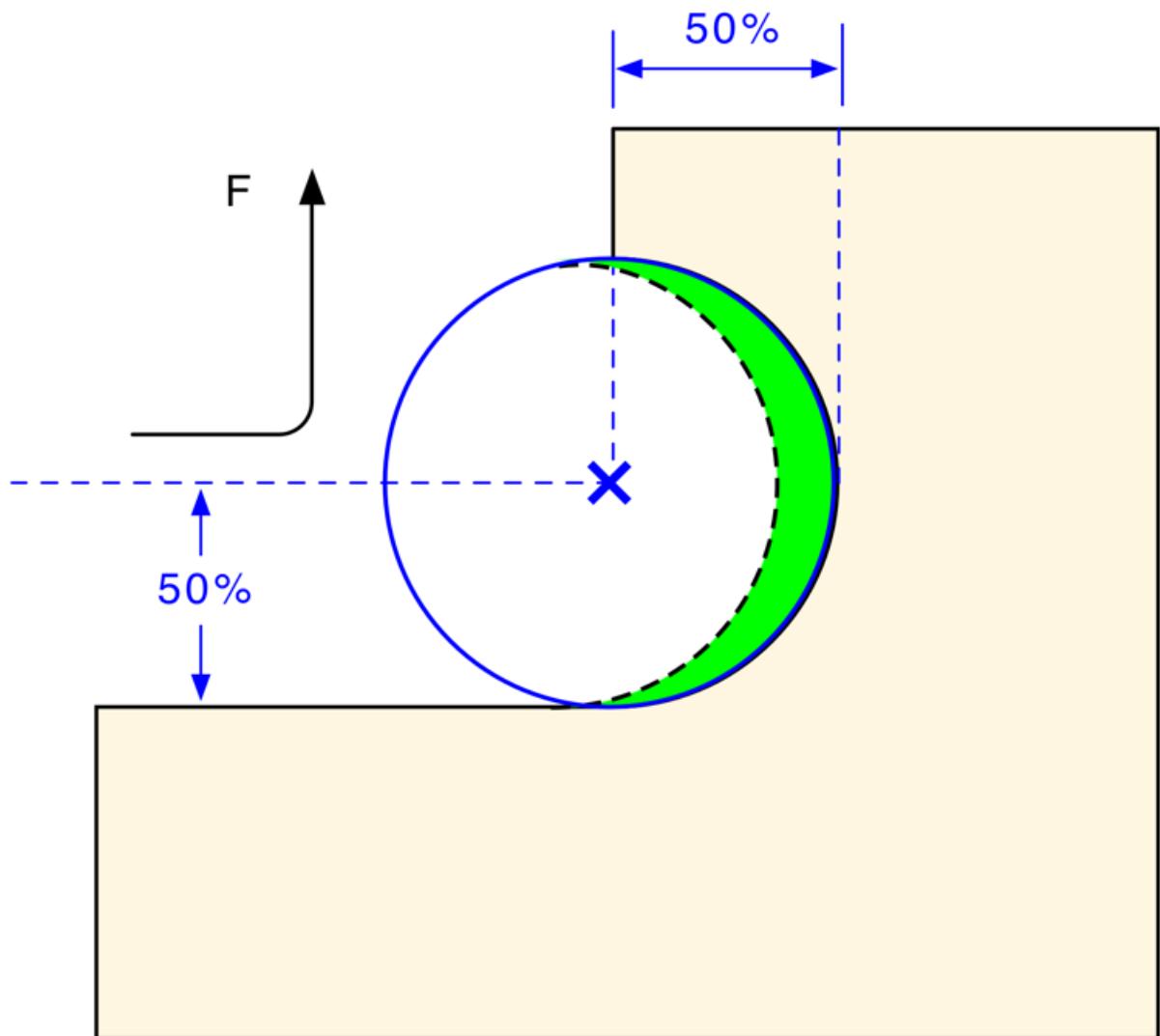
Corners

While we are talking about TEA, let's take a look at what happens when cutting a square pocket at 50% tool engagement (90° TEA) and reaching a corner:

Just before moving into the corner, the tool engagement angle is 90°:



But *while* cutting the corner, the TEA momentarily goes up to 180° :



before going down to 90° again. So the machine sees a "spike" in the cutting resistance at each corner. Just like for slotting, this means that the feedrate and DOC cannot be as high as one would like, since they need to be dialed back a bit to manage corners.

- ⓘ This boils down to optimizing the cut parameters used throughout the job specifically for these very short times when the corners are being cut, which is not very efficient. The alternatives include avoiding straight corners in the design if possible (e.g. round the corners...) or use an **adaptive clearing toolpath** that will take a lot of very shallow bites at the corners instead of a deep one.

Plunge rate

All of the info above only focused on the feeds and speeds for the radial part of the cut, but when the endmill is plunging (straight down/vertically), things are quite different:

- (obviously) the cutting edges on the circumference of the endmill are not cutting anything anymore, the cutting happens at the tip of the endmill only, like a drillbit.
- BUT endmills are really not optimized for drilling, so their ability to plunge efficiently through material is quite limited.

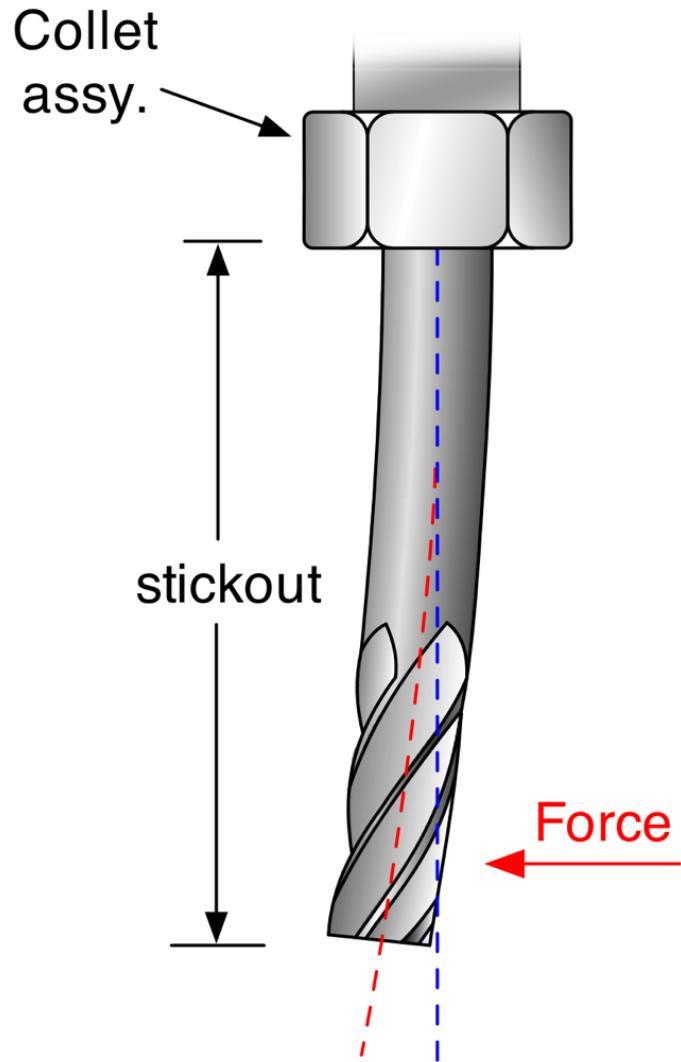
One could compute specific plunge rate and RPM based on the specific geometry of the tip of the endmill, but in practice it's easier to just:

- use the same RPM as for radial cuts. This is a given when using a router where there is no dynamic control on the RPM anyway, so the same value is used throughout the cut.
- use a plungerate that is experimentally chosen, following the rule of thumb
 - **10% to 30%** of the feedrate for metals
 - **30% to 40%** of the feedrate for woods
 - **40% to 50%** of the feedrate for plastics (plunging fast is required to avoid melting)

 These numbers are for plunging straight down. If the toolpath uses some ramping at an angle into the material, they can be increased quite a bit.

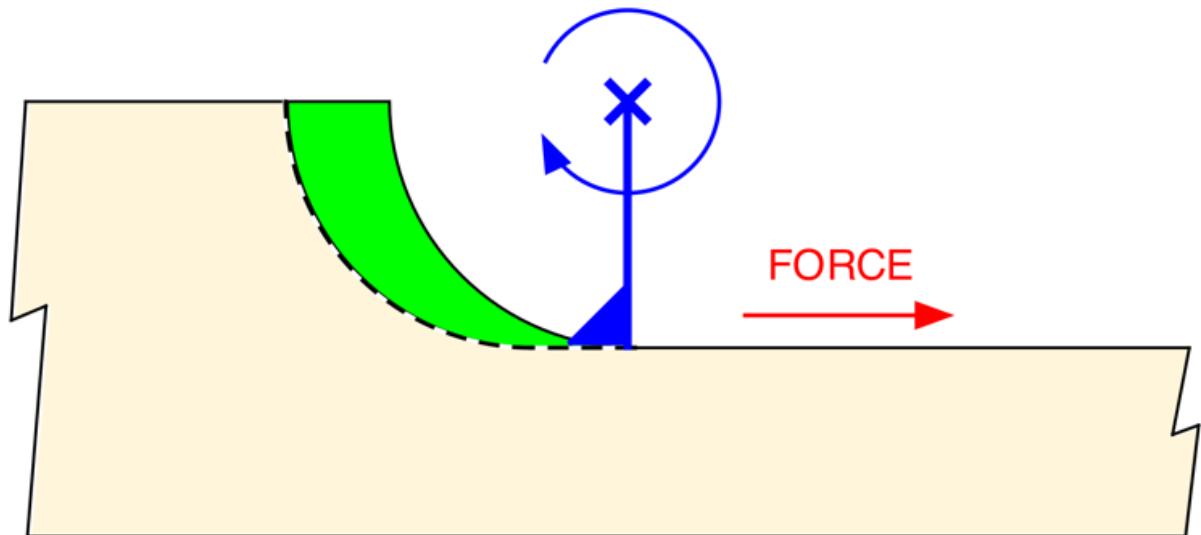
Tool deflection

Endmills are not infinitely rigid, they tend to bend (deflect) when submitted to the cutting forces, and that deflection needs to be taken into account in the feeds and speeds. Here is a grossly exaggerated sketch of an endmill being subject to the cutting force:

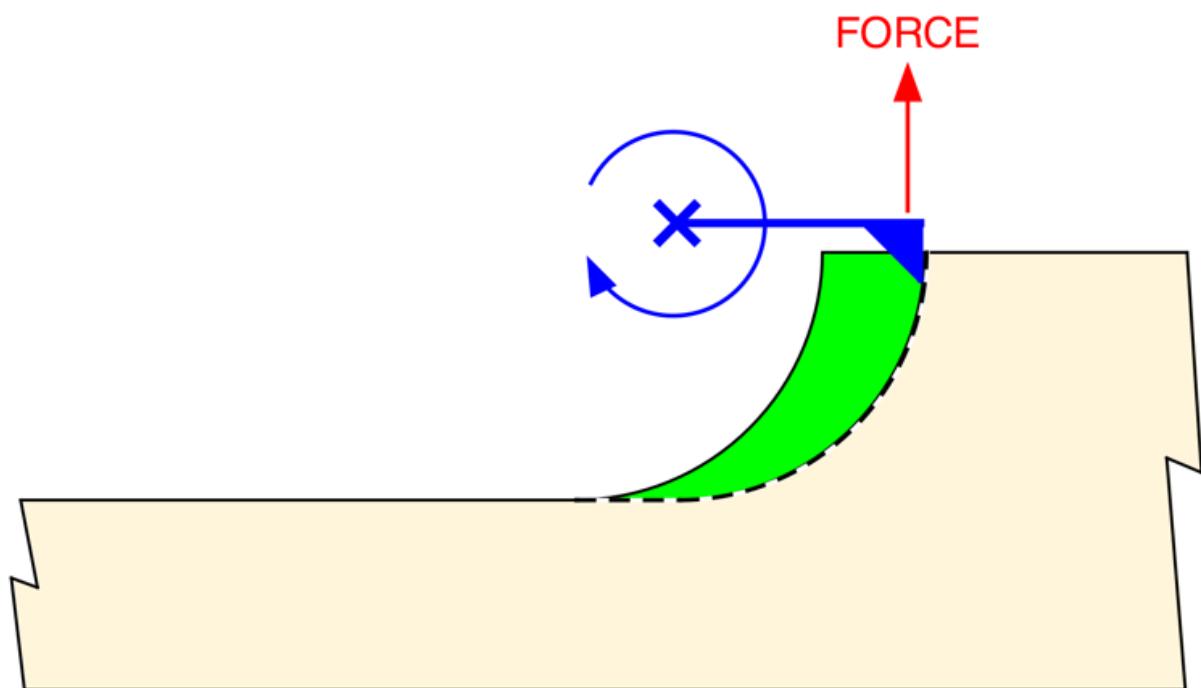


The amount of deflection depends on the endmill material (carbide is more rigid than HSS), diameter (larger is stiffer), stickout length, and of course the cutting forces that the endmill is subjected to, that depend on the chipload, DOC, WOC, and material.

When using conventional milling, the force tends to be parallel to the stock:



When using climb milling, the force tends to be perpendicular, i.e. push the endmill away from the material:



Either way, too much deflection is bad:

- moderate deflection will affect accuracy (pieces will cut slightly larger or smaller than expected)

- excessive deflection will cause tool wear or even tool breakage

So this is yet another parameter to watch out for when selecting feeds and speeds and DOC/WOC values, especially when using small diameter endmills.

Climb or conventional ?

The direction of the cut (climb versus conventional milling) pertains to the toolpath's generation options and not directly to the feeds and speeds, but while we are on this topic: since tool deflection is mainly perpendicular to the cut when using climb milling, it would seem like it is better to use conventional milling, to keep deflection parallel to the cut and therefore minimize dimensional errors on the final piece. However there are other factors at play:

- in conventional milling, the chip is cut from thin-to-thick, so by definition when the flute first comes in contact with the material, it is rubbing the surface a little before it starts actually cutting into the material. This temporary rubbing amounts to heat, so in the long run a conventional cut produces more heat, leading to faster tool wear. Climb milling, since it cuts chips from thick-to-thin, does not have this problem.
- for the same "thick-to-thin" reason, climb milling is a little more tolerant of less-than-perfectly-sharp endmills.
- in conventional milling, the cutter flutes move against the direction of the feedrate, so chips are more likely to be pushed to the front of the cut, leading to chip recutting which is bad for finish quality. In climb milling, the chips tend to be pushed to the back of the endmill / behind the cut, so they are much less prone to recutting.
- in climb milling, the router torque pushes in the same direction as the feedrate, while in conventional it fights against the feedrate, so the forces on the stepper motors are higher.
- climb milling used to have a bad reputation for being dangerous to use on machines with a lot of backlash. While this was perfectly true on older manual mills, the point is moot on CNCs in general and the Shapeoko in particular.

So when all is said and done, climb milling wins on almost every aspect except deflection. The [Toolpaths](#) section will cover the notion of "roughing" versus "finishing" toolpaths, and

that will then open the way for the best approach: using **climb for roughing, then conventional for finishing**.

This way, climb and its many advantages is used for most of the cut, and the possible deflections happening during this roughing pass will be taken care of by the light conventional finishing pass (where the drawbacks of conventional will be irrelevant, since this finishing pass puts such low efforts on the machine anyway, and chip evacuation is not a problem either)

- (i) This being said, your CAM tool may or may not give you the option to select the milling direction (climb or conventional). At the time of writing, Carbide Create did not have this feature, so it generated all toolpaths using conventional milling.

MRR, Power, Torque, Force

If you need to optimize **cutting time** for a given piece, you will also need to take a look at the **material removal rate** (MRR):

$$MRR = WOC \times DOC \times Feedrate$$

This yields a value in cubic inches (or cubic millimeters) of material removed per minute, and therefore relates to how fast you can complete a job. There is always a compromise to be found between going faster but with a lower tool engagement (low DOC and/or low WOC), or going slower but with a higher tool engagement (higher DOC or high WOC), while staying within the bounds of what the machine can do. The interesting thing about the MRR figure is that it allows one to **compare** different combinations, and figure out which one is the most efficient (time-wise).

Now if you want to figure out how close you are to the absolute/physical **limits** of the Shapeoko, (yet) another formula comes in the picture, to characterize the required power at

the endmill level to achieve this MRR:

$$\text{Cutting power (in HP unit)} = \frac{MRR}{K_factor} = MRR \times \text{Unit Power}$$

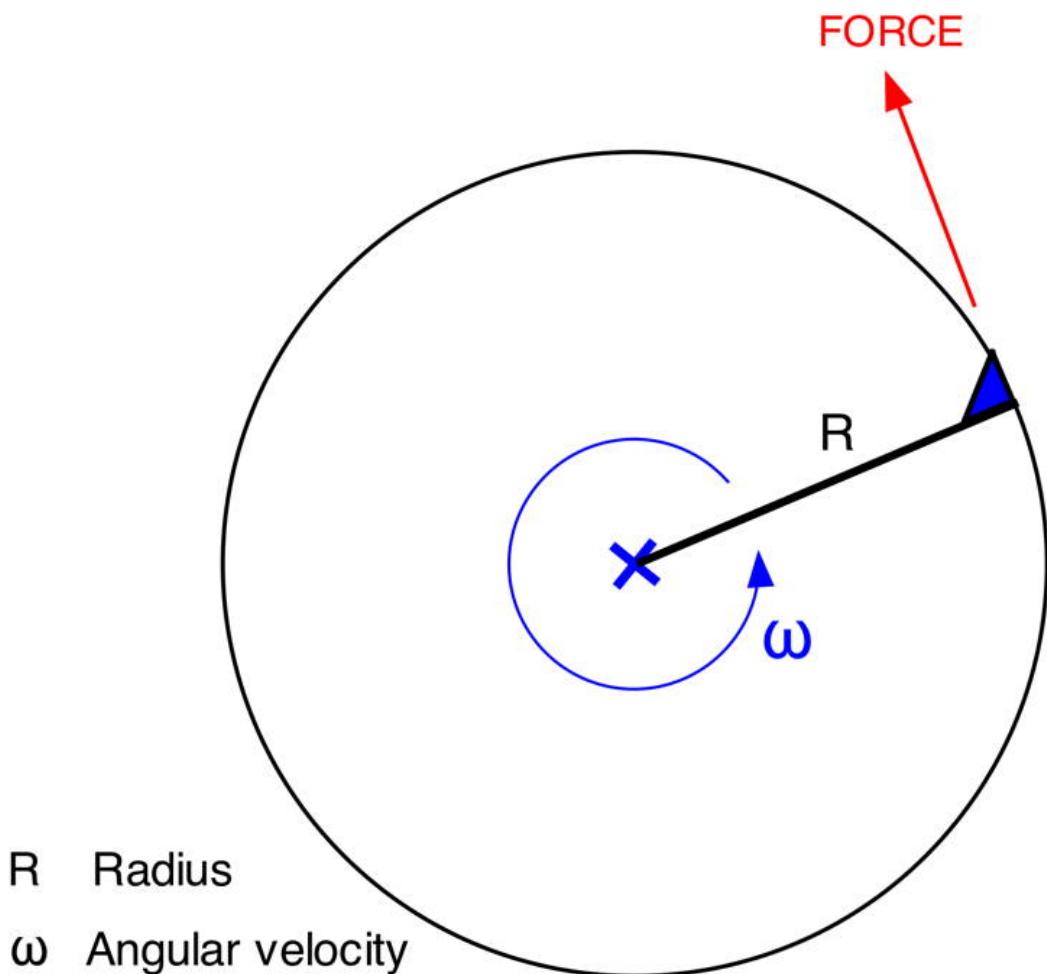
the "K" factor (or its inverse value the Unit Power) is a constant that depends on the material's hardness, and corresponds to how many cubic inches per minute (or cubic millimeters per minute) of material can be removed using 1 horsepower.

For example,

- 6061 T6 aluminium has a K of 3.34 cubic inches per minute
- it's about 10 in³/min for hard woods and hard plastics
- and up to 30 in³/min for soft woods, MDF, ...

Once you get this power value, you can compare it to your router's maximum output power. The Makita and DeWalt routers are rated at a max of 1.25HP (932Watts), but that is *input* power, and the power efficiency of a router is not very good (~50%), so the max actual power at the cutter is more likely around 450W.

And finally, even if the cutting power is within the range of your router, there is still the matter of the **cutting force** that the Shapeoko has to put on the endmill to move it through the material:



In metric units, the torque is the force (in Newton) times the distance in meters (in this case the radius of the endmill), and power in Watts is torque times the angular velocity ω , in radians per second. Since the cutter does RPM revolutions per minute and each of them is 2π radians:

$$\text{Cutting torque_N} \cdot \text{m} = \frac{\text{CutterPower(Watts)}}{2\pi \times \text{RPM} / 60}$$

or in the Imperial units converting N·m to lbf·in ($\times 8.85$ factor), since $(60 \times 8.85) / (2 \times 3.14159) = 84.5$:

$$\text{Cutting torque}_\text{lbf} \cdot \text{in} = \frac{\text{CutterPower(Watts)} \times 84.5}{\text{RPM}}$$

and finally cutting force is:

$$\text{Cutting force} = \frac{\text{Cutting torque}}{\text{endmill radius}}$$

So all of this can be derived from the feedrate, WOC, DOC, endmill size, and material. And this cutting force can then be compared with the Shapeoko's limit estimated (experimentally) to be around **20 lbf** (9Kg)

Wrapping up: suggested process

A proposed workflow to determine a *reasonable starting point* for feeds and speeds and DOCs on the Shapeoko for a given project that uses a specific **material** and **endmill**, is:

- select a **target chipload** in the recommended chipload range for this material+endmill combination
 - refer to my proposed guideline table, or roll your own. Aim for the low end of the range, to reduce cutting force.
- determine **depth of cut** and **width of cut** (stepover) based on the machining style you want (large WOC and small DOC, or large DOC and small WOC).
- if stepover is less than 50%, **adjust target chipload for chip thinning**.
- select **target RPM** as the maximum value you can tolerate and feel comfortable using.
- determine the **required feedrate** for this RPM to achieve the adjusted target chipload
 - if computed feedrate exceeds the Shapeoko limit, choose a lower RPM value and recompute feedrate.
- determine **plungerate** depending on the material.
- check **deflection** value to make sure there is no risk of breaking the tool, and to optimize dimensional accuracy and finish quality.
- check that cutting **power** is within the router's limits.
- check that cutting **force** is within the machine's limits.

- check **MRR** to compare the efficiency of various cutting parameters.

Then...experiment. The realtime **feedrate override** available in most G-code senders is a great way to tune the chipload value and find the sweet spot for a particular job.

Calculators

This section should have highlighted that MANY factors influence the selection of adequate feeds & speeds & DOC & WOC settings.

While predefined recommendations for common endmills and materials are very useful, at some point it becomes impossible to produce feeds & speeds charts for every possible combination of factors, and also very tedious to compute everything manually. A number of calculators have been implemented to address this, ranging from free Excel spreadsheets that basically implement the equations mentioned above, to full-fledged commercial software that embed material/tool databases, the most famous one probably being **G-Wizard**.

Whether or not you *need* a feeds & speeds calculator is debatable: most people use a limited number of combinations of material/endmill sizes anyway, in which case relying on a few good recipes for your machine is enough. The real value of calculators is in **optimizing** the feeds & speeds for a particular situation, and to see the effects of any parameter change on the rest of them.

A pretty neat feeds and speeds worksheet has been put together by @gmack on the Shapeoko forum (which he derived from an original worksheet from the NYCCNC website). I have attached a version here for convenience, but you may want to check if a more recent version is available on the forum.

Cutter Manufacturer, Type, and Part Number: Kennametal UGDE025015B**			Job type, owner, date, etc. 1018 Steel with Kennametal HARVI II 0.25" WOC X 1.25" LOC 5 Flute Endmill		
Imperial/Inch Cutter Information:			Spindle Manufacturer, Type*, and Part Number: Makita 0701		
Cutter Material (Carbide, HSS, or Cobalt)	Carbide	HF Spindle: Speed Max=Rated for Constant Torque	1	Rated Input Voltage (Volts)	
Cutter Diameter (D)	0,25	Rated Power (Watts) @ Rated Speed	2200	Rated Input Current (Amps)	6,5
Cutter Length	1,25	Rated Speed (RPM)	24000	Rated Input Power (Watts)	
# of Cutter Flutes/Teeth	5	Maximum Speed (RPM)	24000	Efficiency	50%
Shank Diameter	0,25	Current at Rated Power (Amps)	10	Rated? Output Power (HP)	
Overall Stickout	1,375	Available Power (HP) @ RPM	1,88	Rated? Speed (RPM)	27000
Required Inches Per Tooth - IPT@WOC	0,00060	Torque Constant (in-lbf / Amp)	0,7747	Available Power? (HP) @ RPM	
IPT for Feed Rate Calculation (IPT@WOC = 1/2D)	0,0015	manufacturer's recommended maximum RPM/SFM or IPT		CNC Machine Rigidity (Minimize spindle/router stickout), Available Feed Force, and/or Workpiece Holddown Strength Limit Maximum Available/Usable Machine Force.	
Young's Modulus	8,70E+07				
Maximum Acceptable Deflection (in):	0,0010	for work-piece!	SFM (>990 is Al HSM)	1000	Maximum Machine Force (Lbf): 18
Workpiece Material:	1018	Material K Factor from Spreadsheet:	1,88	K Factor for Power & Force Calculations	3 1,88
Cutter Speed (RPM):	From RPM: 4 OR From SFM: 1000	Chosen Spindle/Router and Cutter Speed (RPM) (From Either RPM entry OR SFM): 15279			
her speed (RPM) reduces force and cutter deflection, but requires higher feed rates (IPMs) to maintain the Required Inches Per Tooth (IPT) AKA "chipload". Using cutters with fewer flutes (teeth) increases chipload.					
Cut Depth and Width	Minimum	Maximum	Fusion Adaptive	As Percentage of Diameter	Minimum Maximum Fusion Adaptive In Inches Values to Use (in):
Depth of Cut (DOC):	0%	300%	100%		0 0,75 0,25 0,250
Width of Cut (WOC):	5%	100%	20%		0,0125 0,25 0,05 0,010
CNC Feed Rate (IPM):	From IPT@WOC: 117	OR Enter IPM: 46	OR Enter Force (lbf): 2,0	No Entry will select From "IPT@WOC"	"Use climb milling to minimize rubbing - Think thick to thin chips."
Estimated Results from Speeds and Feeds Calculator vs. 7 Components of Spindle/Router Input Current/Power					
Calculated and Measured Results	Chipload (IPT)	MRR (in³/min)	Cutter	Machine Force	% Spindle/Router Available Power (HP) HF Spindle Cutting Current Increase (Amps)
Calculated Estimates	0,00023	0,11	Power (HPe) 0,061	Torque (in-lbf) 0,25	% Max Deflection 10,39% % of Maximum 2,00 11% 3% 0,3
Measured					
HF Spindle Current Input (A) When Not Cutting			HF Spindle Current Input (A) When Cutting		
Deflection can be reduced without decreasing MRR by increasing RPM, reducing cutter stickout, using a carbide cutter, and/or by increasing the diameter of the cutter or its shank. Deflection, Force, and Power can be reduced at the expense of MRR by decreasing WOC, DOC, and/or IPM (at the expense of IPT).					

 gmack's advanced feeds and spe...

2019-08-11 Speeds and Feeds Workbook.xlsx - 161KB

- fill-in the **specs of your router** or spindle (once).
- fill-in the **specs of the selected endmill**, and the **target chipload value** you chose (chip thinning will be taken into account automatically depending on WOC value)
- if you care about power/force analysis, look-up the **K-factor** for the material being cut (there's a list in a separate tab of the worksheet) and update it here.
- select **target RPM** value (or alternatively SFM, then RPM will be derived from it).
- select **WOC and DOC** (depending on your machining style)
- The **required feedrate** to reach the target chipload will be computed. You can alternatively choose to override it with a given feedrate value (and see what this does to chipload displayed below), or to forget about chipload and use a given cutting force as the ultimate target. Either way, the feedrate to be used will be displayed at the right end of this line.
- You can then check the analysis of **deflection, cutting force, and power** in the lower part of the worksheet.

Then play with the input values to compare various cutting scenarios while staying within the machine's hard limits (max RPM, max feedrate, max power, and max cutting force)

- i** The latest version of @gmack's worksheet is available in the forum here:
<https://community.carbide3d.com/t/speeds-feeds-power-and-force-spf-calculator/16237>

- i** Once you determine good feeds and speeds and confirm that it is cutting correctly, it is useful to capture a snapshot of the worksheet for that particular usecase for future reference (just duplicate the tab in the worksheet)

If you still feel overwhelmed or don't care about optimizing power, force and deflection, I derived a more basic version:

INPUTS		Endmill guidelines			Predefined constants																												
1 Endmill flutes	N=2	Consider using an endmill with fewer flutes if computed feedrate is above machine limit			Inputs																												
Endmill diameter	0.250"				outputs																												
					computed																												
		Chipload guideline																															
2 Target chipload	0.0010"	<table border="1"> <thead> <tr> <th>Endmill</th> <th>Soft plastics</th> <th>Soft wood & hard plastics</th> <th>Hard wood & aluminium</th> </tr> </thead> <tbody> <tr> <td>1/16"</td> <td>0.002" to 0.003"</td> <td>0.001" to 0.0015"</td> <td>0.0005"</td> </tr> <tr> <td>1/8"</td> <td>0.002" to 0.005"</td> <td>0.001" to 0.0025"</td> <td>0.0005" to 0.001"</td> </tr> <tr> <td>1/4"</td> <td>0.004" to 0.01"</td> <td>0.001" to 0.005"</td> <td>0.001" to 0.002"</td> </tr> </tbody> </table>			Endmill	Soft plastics	Soft wood & hard plastics	Hard wood & aluminium	1/16"	0.002" to 0.003"	0.001" to 0.0015"	0.0005"	1/8"	0.002" to 0.005"	0.001" to 0.0025"	0.0005" to 0.001"	1/4"	0.004" to 0.01"	0.001" to 0.005"	0.001" to 0.002"													
Endmill	Soft plastics	Soft wood & hard plastics	Hard wood & aluminium																														
1/16"	0.002" to 0.003"	0.001" to 0.0015"	0.0005"																														
1/8"	0.002" to 0.005"	0.001" to 0.0025"	0.0005" to 0.001"																														
1/4"	0.004" to 0.01"	0.001" to 0.005"	0.001" to 0.002"																														
		RPM guideline			Min RPM																												
3 RPM	12000	<table border="1"> <thead> <tr> <th colspan="2">Set as high as you can tolerate (noise) and feel safe using</th> </tr> <tr> <th colspan="2">Decrease if computed feedrate is above machine limit</th> </tr> </thead> </table>			Set as high as you can tolerate (noise) and feel safe using		Decrease if computed feedrate is above machine limit		10000 (Makita/Carbide3D: 10000)																								
Set as high as you can tolerate (noise) and feel safe using																																	
Decrease if computed feedrate is above machine limit																																	
					Max RPM																												
					30000 (Makita/Carbide3D: 30000)																												
		WOC guideline			DOC guideline																												
4 WOC	0.010"	<table border="1"> <thead> <tr> <th>Milling style</th> <th>Toolpath type</th> <th>recommended range</th> </tr> </thead> <tbody> <tr> <td>wide and shallow for...</td> <td>roughing</td> <td>0.1" to 0.25"</td> </tr> <tr> <td>wide and shallow for...</td> <td>finishing</td> <td>0.0125" to 0.025"</td> </tr> <tr> <td>narrow and deep for...</td> <td>roughing</td> <td>0.025" to 0.0625"</td> </tr> <tr> <td>narrow and deep for...</td> <td>finishing</td> <td>0.0125" to 0.025"</td> </tr> </tbody> </table>			Milling style	Toolpath type	recommended range	wide and shallow for...	roughing	0.1" to 0.25"	wide and shallow for...	finishing	0.0125" to 0.025"	narrow and deep for...	roughing	0.025" to 0.0625"	narrow and deep for...	finishing	0.0125" to 0.025"	<table border="1"> <thead> <tr> <th>Milling style</th> <th>Material</th> <th>recommended range</th> </tr> </thead> <tbody> <tr> <td>wide and shallow in...</td> <td>aluminium, copper, bronze</td> <td>0.0125" to 0.025"</td> </tr> <tr> <td>wide and shallow in...</td> <td>softer materials</td> <td>0.025" to 0.125"</td> </tr> <tr> <td>narrow and deep in...</td> <td>all materials</td> <td>0.25" to 0.75"</td> </tr> </tbody> </table>		Milling style	Material	recommended range	wide and shallow in...	aluminium, copper, bronze	0.0125" to 0.025"	wide and shallow in...	softer materials	0.025" to 0.125"	narrow and deep in...	all materials	0.25" to 0.75"
Milling style	Toolpath type	recommended range																															
wide and shallow for...	roughing	0.1" to 0.25"																															
wide and shallow for...	finishing	0.0125" to 0.025"																															
narrow and deep for...	roughing	0.025" to 0.0625"																															
narrow and deep for...	finishing	0.0125" to 0.025"																															
Milling style	Material	recommended range																															
wide and shallow in...	aluminium, copper, bronze	0.0125" to 0.025"																															
wide and shallow in...	softer materials	0.025" to 0.125"																															
narrow and deep in...	all materials	0.25" to 0.75"																															
		OUTPUT			Shapeoko max feedrate																												
Required feedrate	61 ipm (includes compensation for chip thinning)				196.85 ipm																												
		(straight) plunge rate guideline																															
Plunge rate	<your call>	<table border="1"> <thead> <tr> <th>Material</th> <th>plunge rate</th> </tr> </thead> <tbody> <tr> <td>aluminium, copper, bronze</td> <td>6 to 18 ipm</td> </tr> <tr> <td>hard wood</td> <td>18 ipm</td> </tr> <tr> <td>soft woods</td> <td>24 ipm</td> </tr> <tr> <td>plastics</td> <td>24 to 31 ipm</td> </tr> </tbody> </table>			Material	plunge rate	aluminium, copper, bronze	6 to 18 ipm	hard wood	18 ipm	soft woods	24 ipm	plastics	24 to 31 ipm																			
Material	plunge rate																																
aluminium, copper, bronze	6 to 18 ipm																																
hard wood	18 ipm																																
soft woods	24 ipm																																
plastics	24 to 31 ipm																																

 [FS_worksheet_basics.xls](#)

FS_worksheet_basics.xls - 70KB

1. fill-in the **number of flutes** and **diameter** of your endmill
2. pick a **target chipload** value from the guideline table on the right
3. select an **RPM** value
4. select **WOC** and **DOC** based on the recommended values on the right (derived from the selected endmill diameter)

This basic worksheet will just compute the required feedrate to get the desired chipload (taking chip thinning into account). If the computed feedrate turns red, it is beyond the limit

of the Shapeoko, and you should select a lower RPM and/or use an endmill with a lower flute count.

Telltale signs of wrong F&S

The most common signs of inadequate feeds and speeds are:

- sound, and specifically **chatter**: when feeds and speeds are not right for a given material/endmill/DOC/WOC, the tool tends to vibrate, and this vibration can get worse if there is resonance with another source of periodic variation elsewhere in the system (most often: the router and its RPM). This results in an ugly sound, a poor finish with marks/dents/ripples on the surface, and a reduced tool life.
 - **finish quality**: even without chatter, a poor surface finish can indicate that the final cutting pass was too aggressive (too much chipload or too much deflection). Increasing RPMs may help, but the best approach is to use a finish pass with very low WOC.
 - **melted material**: especially in plastics and soft metals like aluminium, if the feedrate is too low for the selected RPM, the friction will cause the material to melt rather than shear, the tool flutes will start filling with melting material, and this usually ends up with tool breakage. You will need to feed faster, and/or use an endmill with a lower flute count.
 - **endmill temperature**: the endmill should not be more than slightly warm at the end of a cut: if it gets hot to the touch (careful!), the feeds and speeds are likely incorrect (too low or too high chipload), or the tool is dull and is rubbing rather than cutting. In extreme cases, the endmill color itself may change to a dark shade.
 - making **dust**, instead of clearly formed chips is an indication that chipload is probably too low (MDF is an exception, you just cannot get chips anyway with this material)
-

What about V-bits?

Notice how I carefully avoided the case of V-bits throughout this section ? That's because V-bits are special, due to their geometry and the nature of their associated toolpaths:

- The cutting speed varies along the edge of a V-bit, from its largest section ("top" of the V-bit) to its point (the surface speed is zero at the tip).
- The effective cutting diameter varies depending on how deep the V-bit goes.
- The V-carving toolpaths tend to generate sloped trajectories and a lot of plunges and retracts, so the cutter engagement is constantly changing.

So it does not quite make sense to be using a target chipload value for a V-bit.

Experimentation is king in V-carving, but a common starting point for using V-bits in wood is as follows:

- RPM in the 16k–20k range
- Feedrate in the 30–60 ipm range (lower for hard wood, faster for soft wood)
- Depth per pass in the 0.1–0.2" range
- Plunge rate in the 10–20 ipm range

 If your CAM software supports it, you may want to use a roughing pass and a finishing pass (with more aggressive settings for the roughing pass to spare time, and more conservative settings for the finishing pass)

 Sometimes when using V-bits, running the G-code twice can lead to a cleaner finish.

Toolpaths

A **toolpath** is the intended trajectory that the tip of the endmill will follow, to remove material and produce the desired geometry of the workpiece.

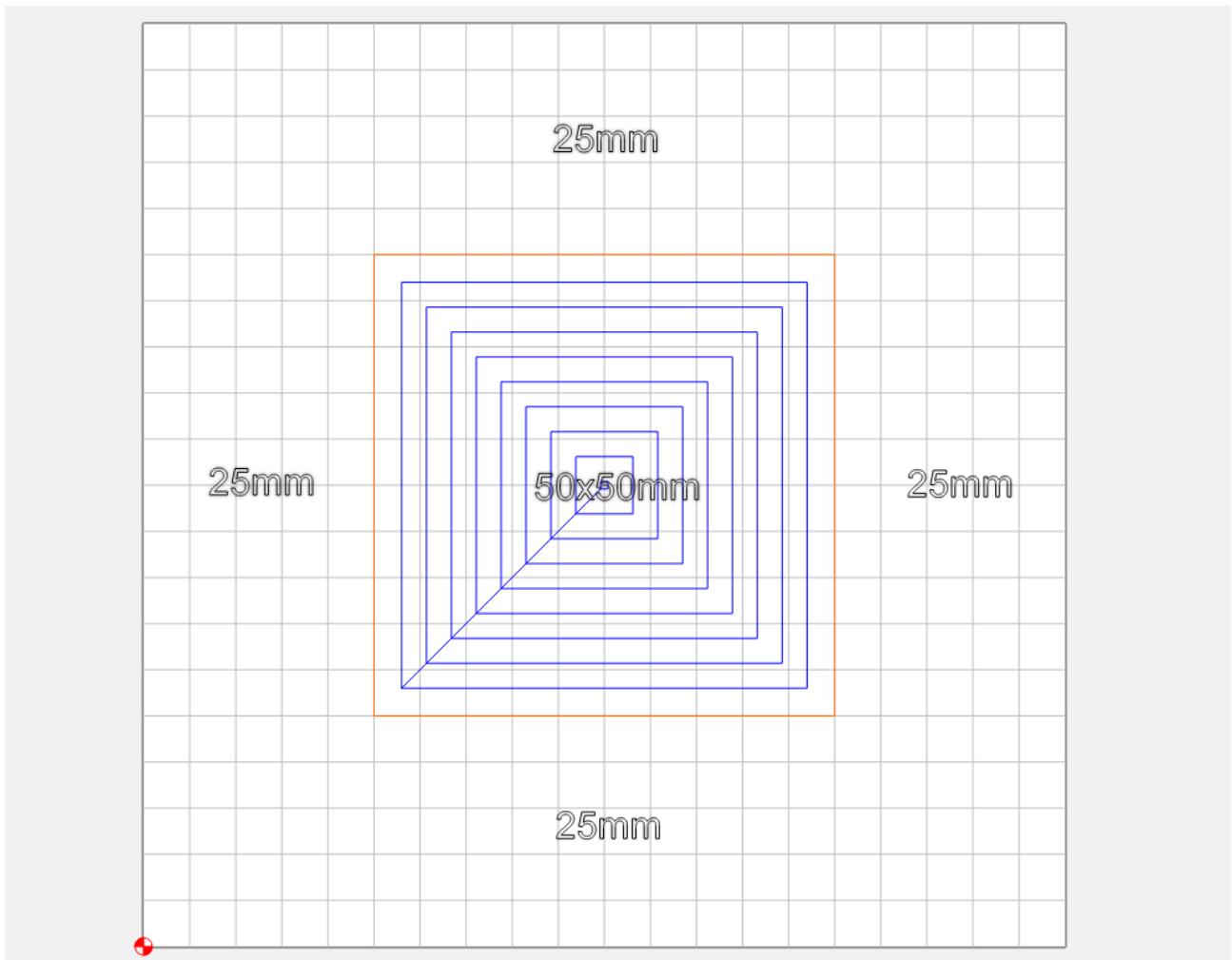
For a specific object geometry/feature defined in the CAD tool, the CAM tool will generate toolpaths as sets of lines and curves defined in X/Y/Z space, and the post-processor will then generate the corresponding G-code instructions for the selected machine.

"**2D**" toolpaths correspond to cutting a 2D feature in the design, moving only two axes of the machine at a time (typically, stepping down along Z only, then moving only in the XY plane, before proceeding with a deeper Z). Since X, Y, and Z are moved (albeit not simultaneously), this is sometimes called "**2.5D**".

"**3D**" toolpaths correspond to cutting a 3D feature, and potentially moving the three axes of the machine simultaneously.

- (!) Not all toolpaths presented below are available in the standard version of Carbide Create: 3D toolpaths, adaptive toolpaths, REST machining, and the roughing vs. finishing toolpaths feature require the use of other CAM programs. If you are just starting you can ignore those, Carbide Create is perfect to learn simple 2D toolpaths before moving on to more complex projects. The intent here is to show what is available in various CAM programs, and then everyone can decide whether to invest money (e.g. VCarve, Carbide Create Pro) or time (e.g. Fusion 360) to access those features.

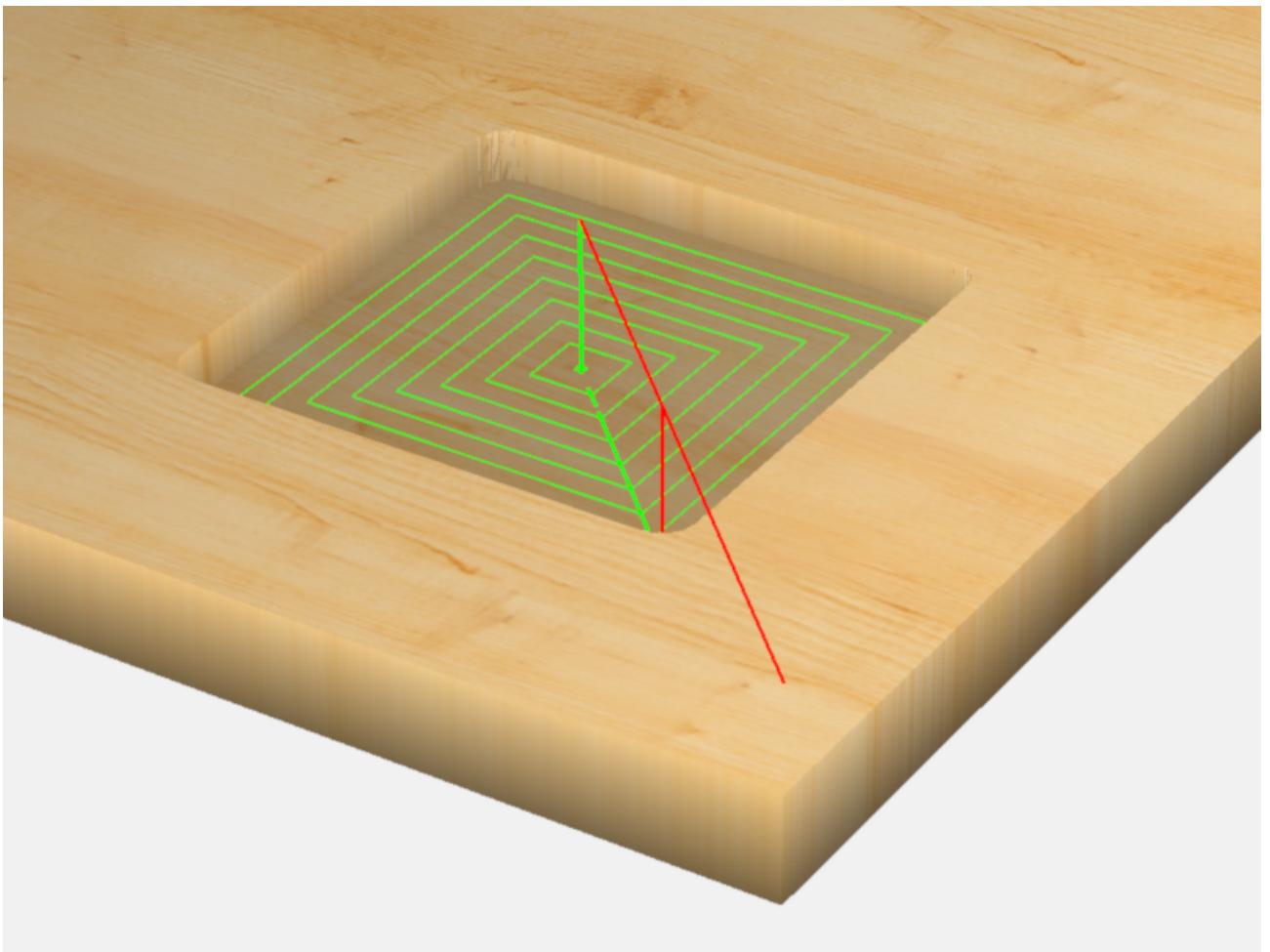
Let's take the simplest example of a 50×50mm square shape in a CAD tool, placed in the center of a 100×100×10mm stock material, with the zero point defined in the lower-left corner, and using a 6mm endmill (~1/4") :



Various types of toolpaths can be created based on this reference shape.

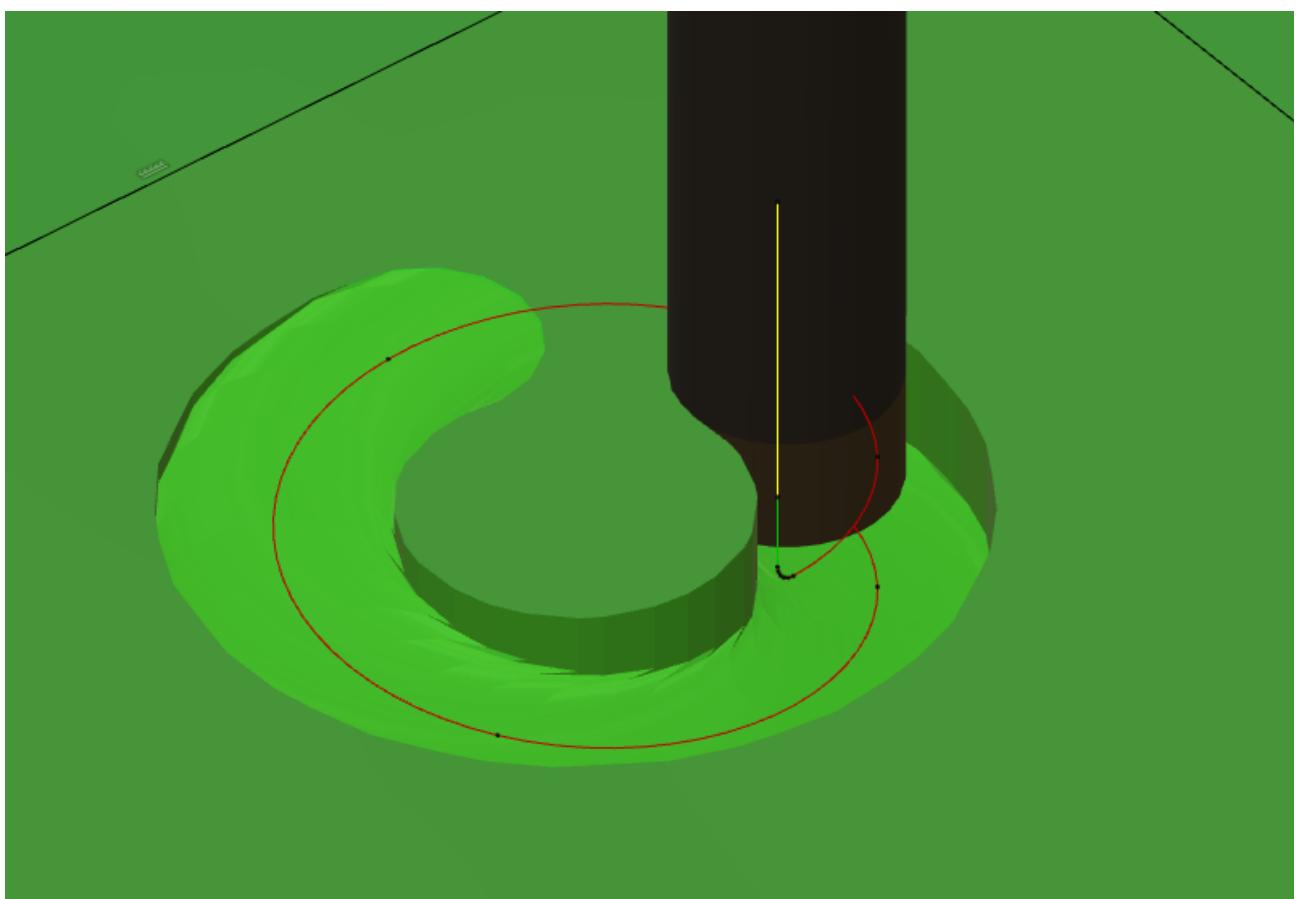
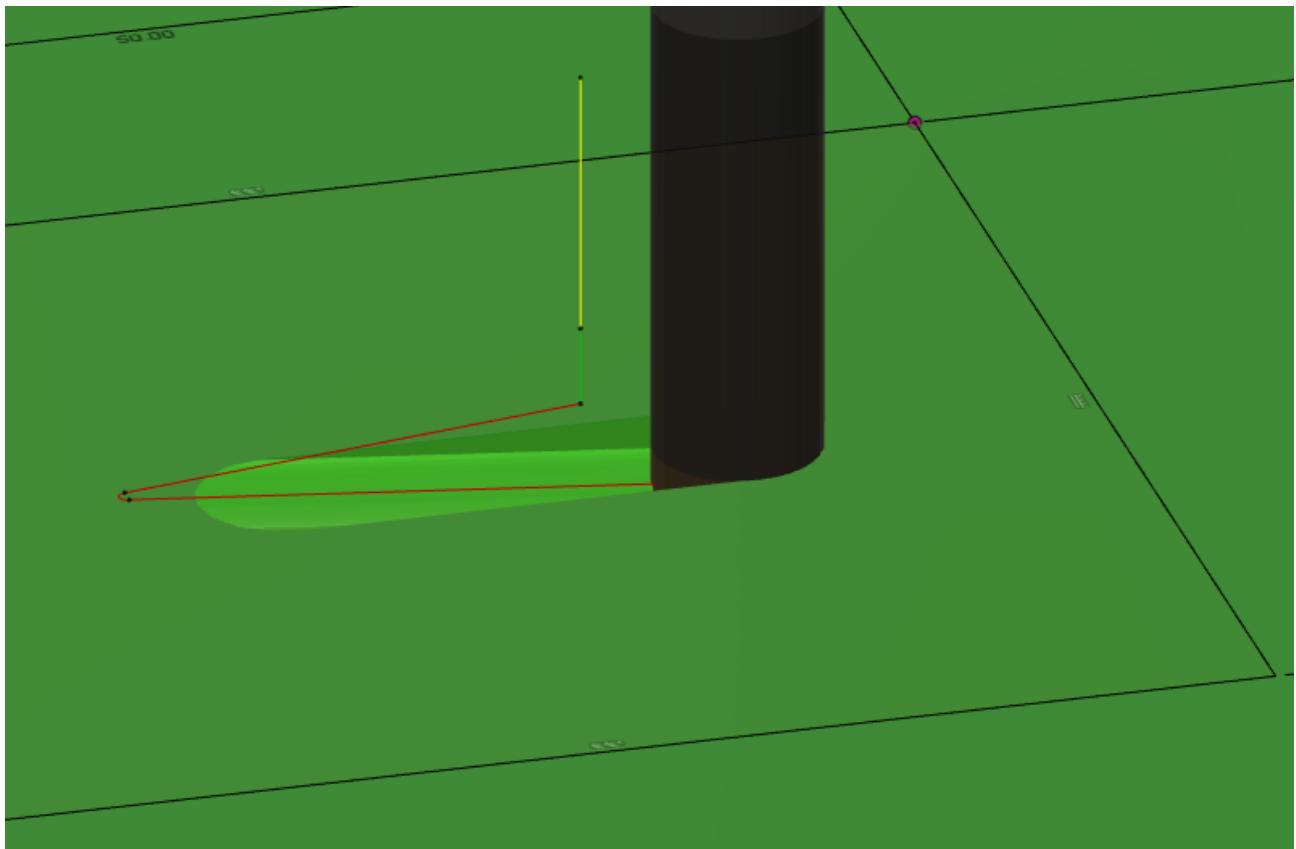
Pocket toolpaths

In a pocket toolpath the endmill is moved within the boundaries of the shape, to remove all material from the top surface down to a given depth:

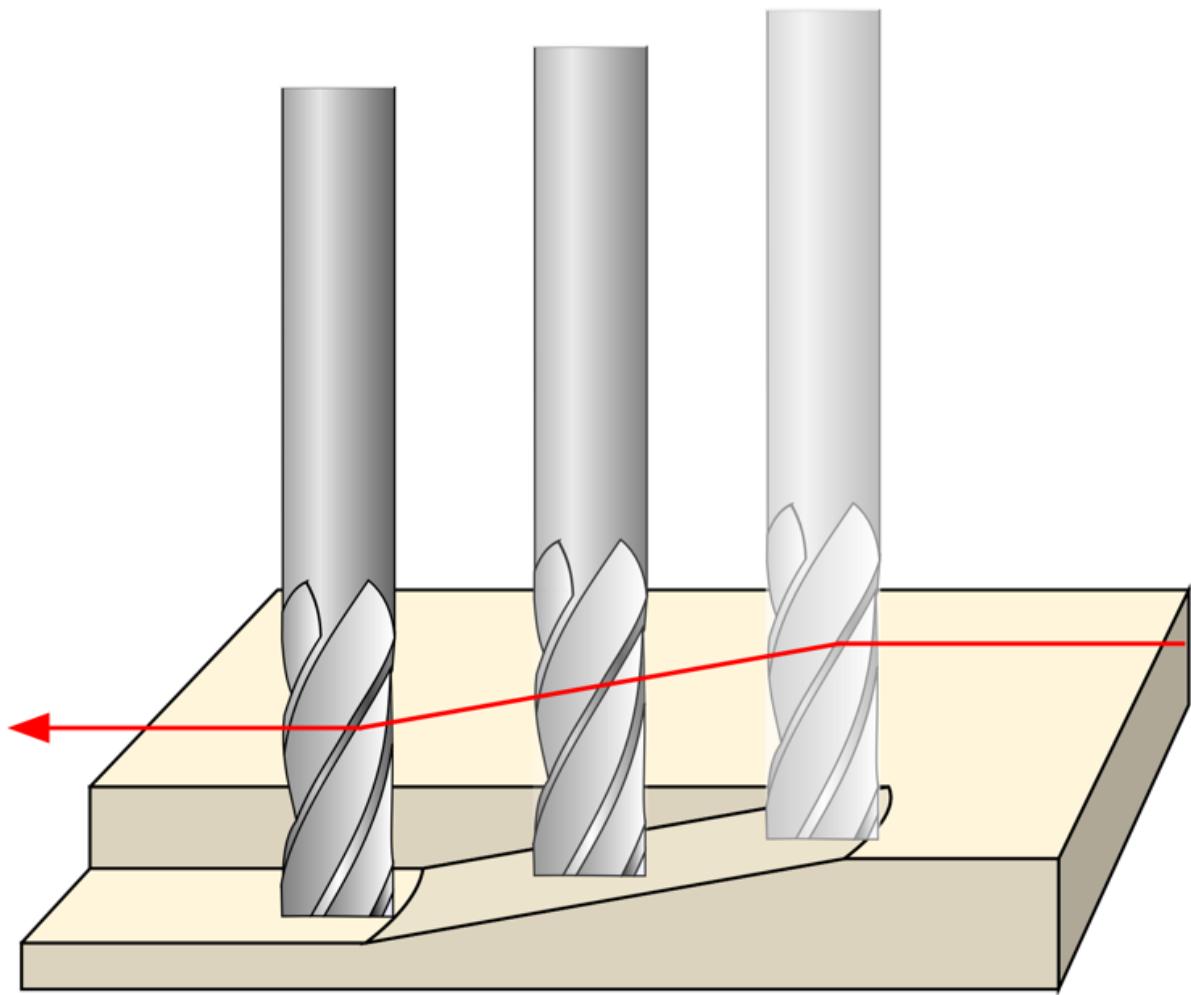


Extremely simple, but it already illustrates a few interesting general parameters that will apply to any toolpath:

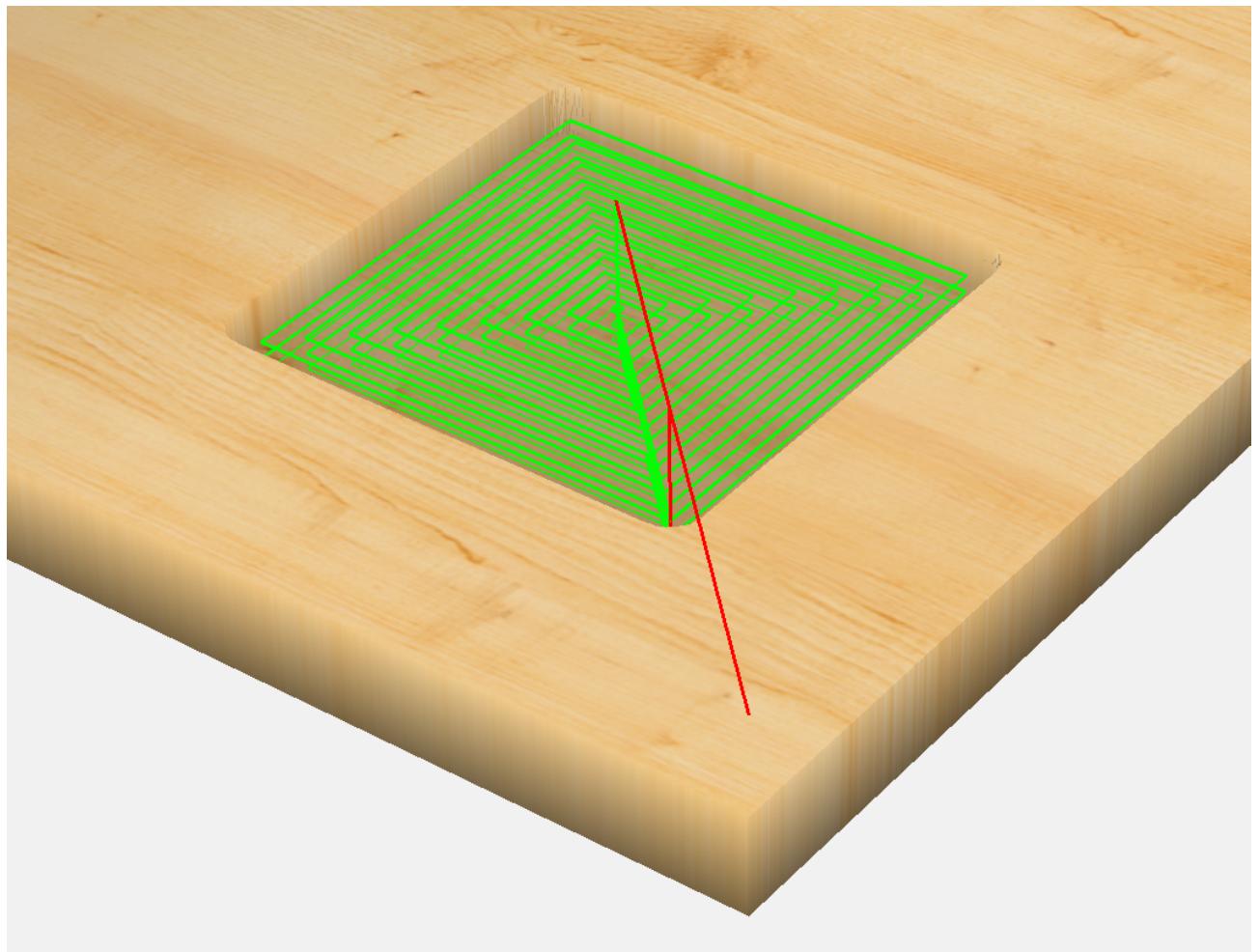
- the **zero (X0/Y0/Z0) point**: in this example, it is positioned in the lower-left corner of the top of the stock.
- since the toolpath defines the trajectory of the *tip* of the endmill, an endmill positioned at the zero point touches the stock surface, so the very first action is to raise the endmill up to the **retract height** (Z offset above Z0), to avoid scraping the surface when moving to the cutting area.
- the red lines illustrate rapid moves ("rapids"). The first move is from the zero point (+ retract height), to the point where the endmill will start **plunging** into the material (vertical green line)
- The example above illustrates a straight plunge, since this is what Carbide Create generates, but plunging vertically is a bit hard on the endmill, so there are other approaches to ensure a smoother entry into the material. Below are examples of **linear ramping** and **helix ramping** (using Fusion360 CAM preview) :



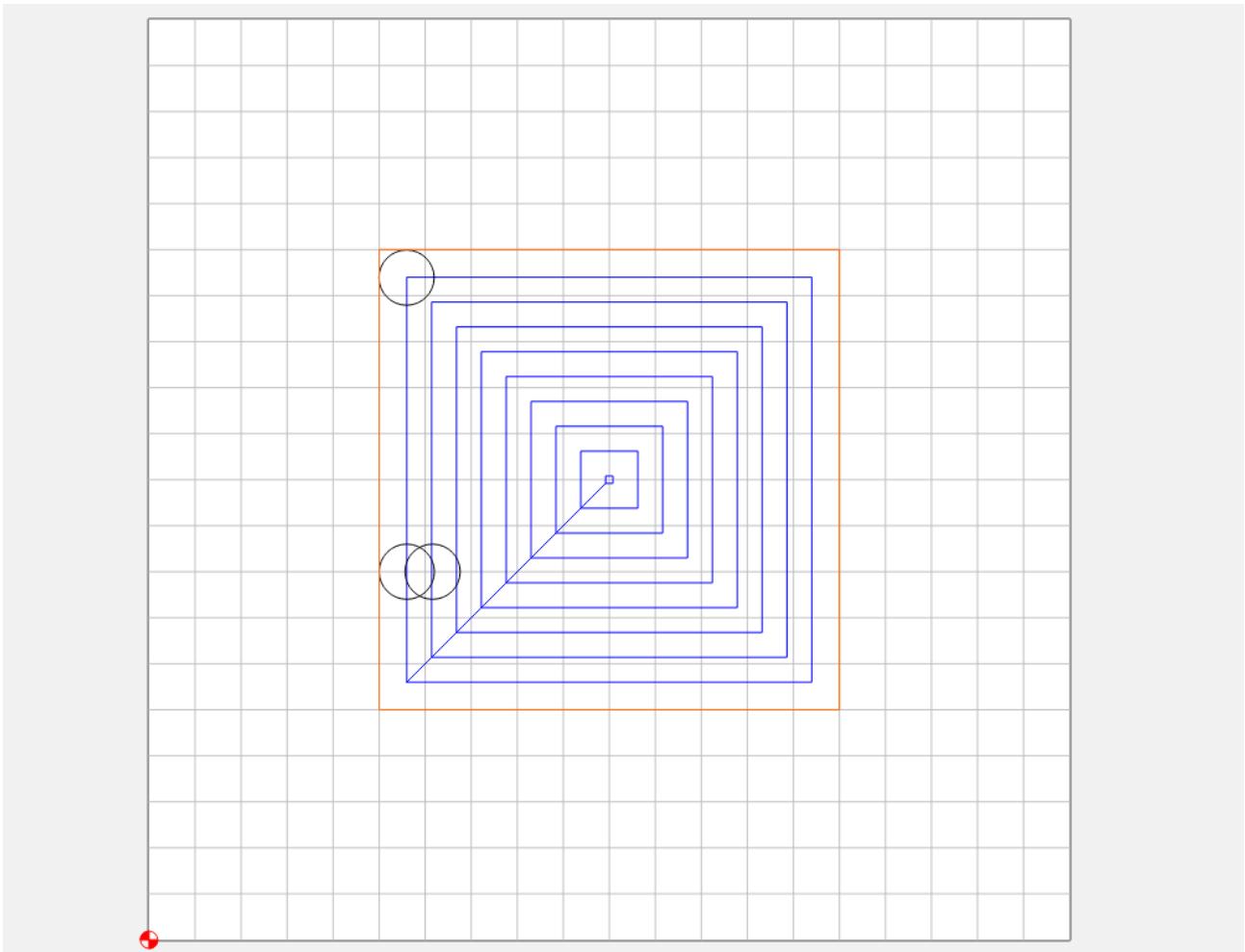
Here's an alternate view of linear ramping into the material:



Since the pocket depth may exceed what can be cut in a single pass by the endmill, it is often necessary to take several passes, removing a given depth of cut (DOC) a.k.a. **depth per pass**, until the full pocket depth is reached:

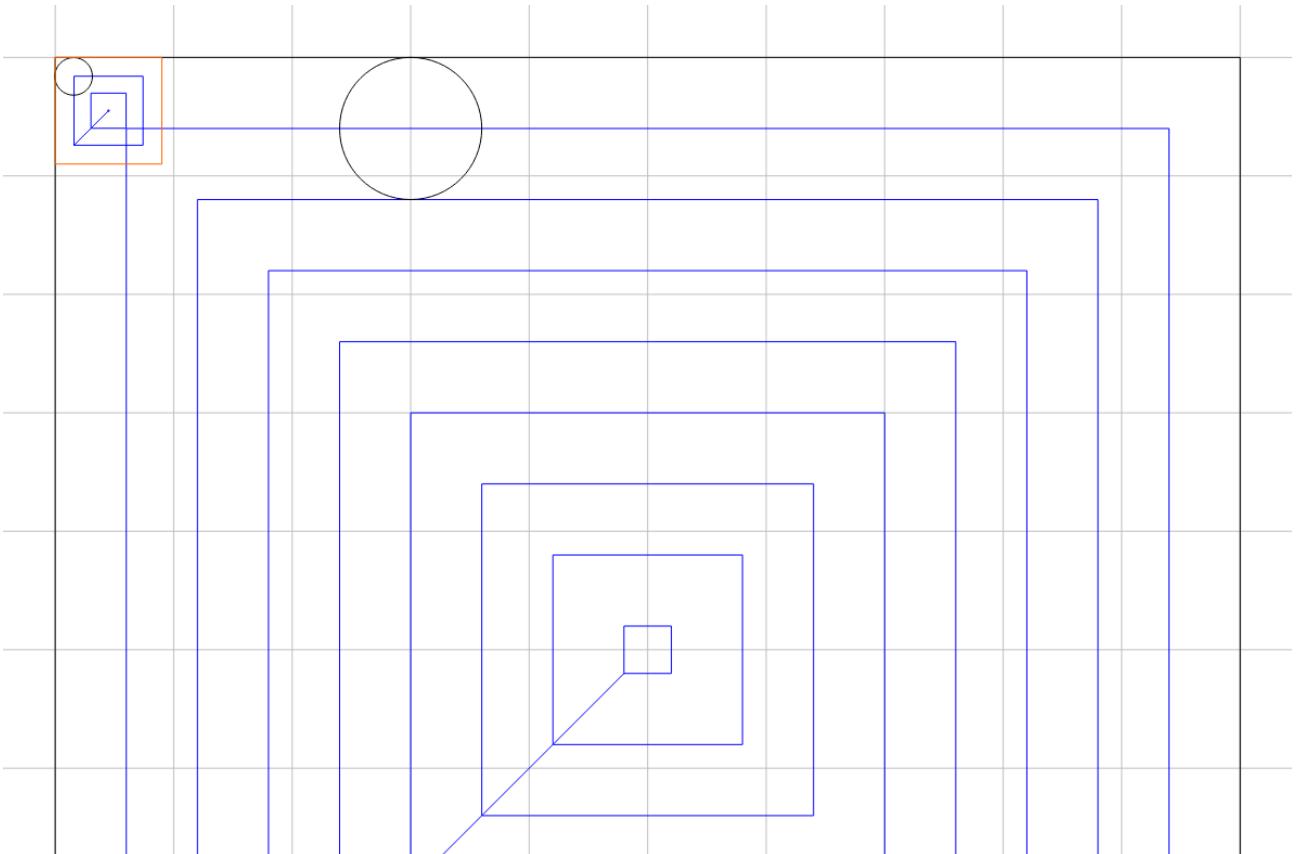


On the following snapshot, a few circles of the same diameter as the endmill were added for the purpose of illustrating two things:

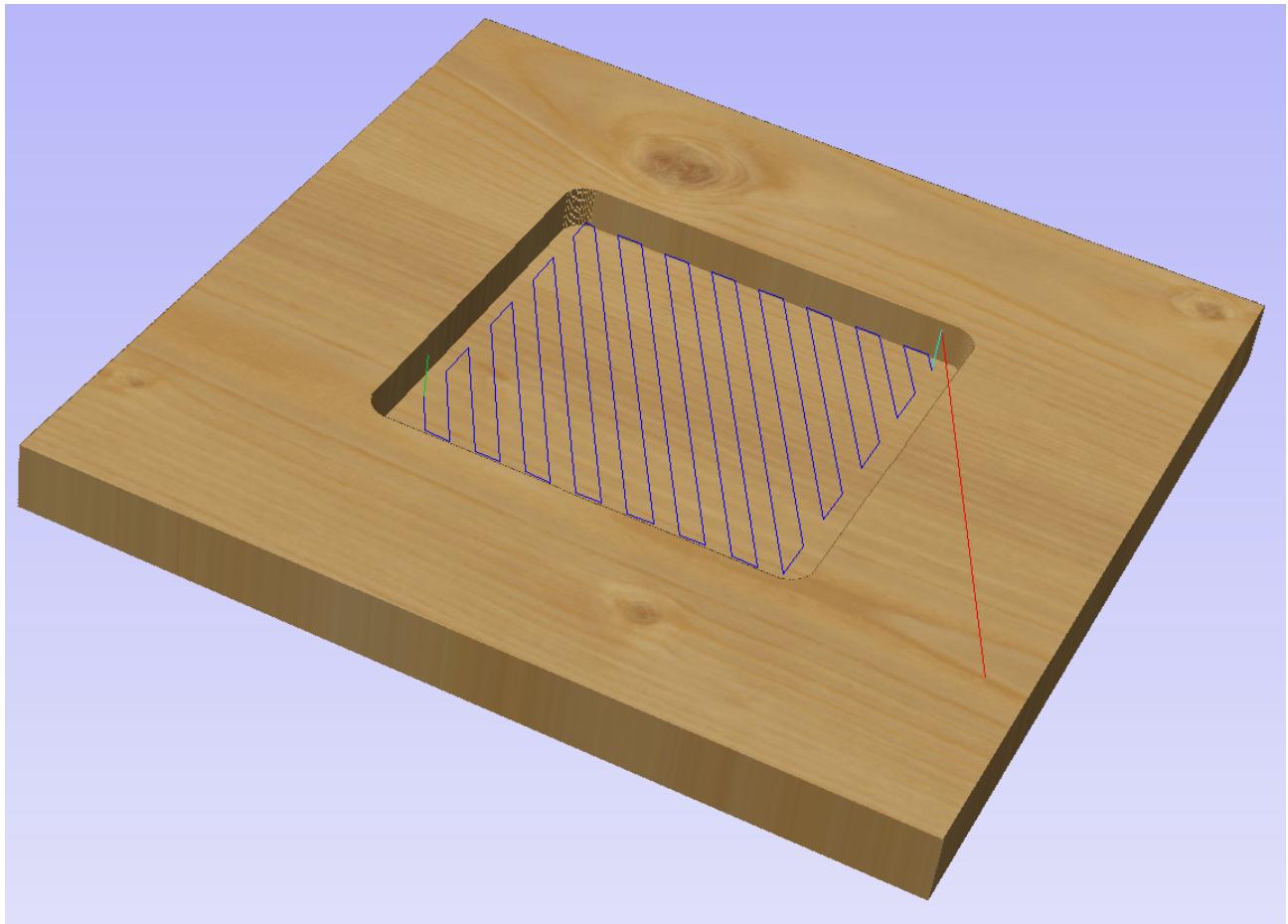


- the **stepover** (a.k.a. width of cut, a.k.a. radial width of cut) parameter of the toolpath controls how close to each other successive loops of the toolpaths are, in this case the stepover was chosen to be 50% of the endmill diameter for simplicity. Check out the [Feeds & speeds](#) section for recommended stepover values, and how this affects chip thinning and ultimately the optimal feeds and speeds.
- in **corners** two things happen:
 - the cutter engagement temporarily increases (see tool engagement in the [Feeds & speeds](#) section). Nothing to be concerned about in many cases, but this limits the feeds and speeds to being more conservative than they could be. This is where advanced toolpaths help, *e.g.* adaptive clearing, more on this later.
 - obviously, the round endmill cannot reach all the way into the corners, so some material is left and all corners end up rounded to the diameter of the endmill. One way to mitigate this is to use a smaller endmill, but cutting a large pocket using a very small endmill would take forever, so a better alternative is first cut the pocket normally with a large endmill, then run a second toolpath with a smaller endmill, that will only work locally in the corners. In advanced CAM tools, this is easy using the "rest machining" option described later below, where the CAM is smart enough

to figure out how much material is left and where and to produce a second toolpath with a smaller tool that will only cut there. At the time of writing Carbide Create does not support rest machining, but you could fake it by manually creating additional geometry. In the example below, a 4.5×4.5 mm square was added in one corner, with an associated pocket toolpath using a 1/16" endmill. The corner will still not be perfectly square, but its radius will be 4 times smaller, so it will look much closer to square.

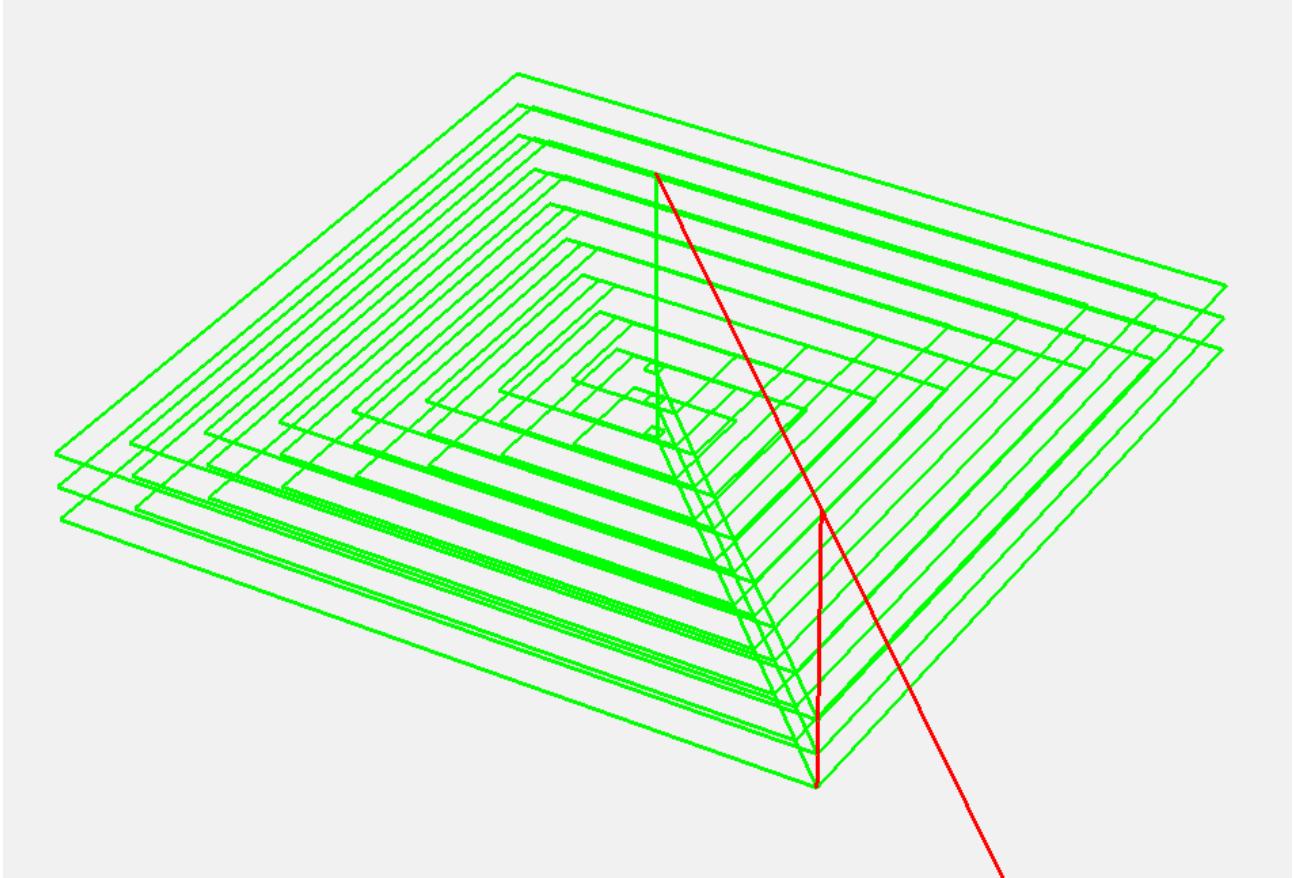


Even though this "concentric squares" toolpath is by far the most common, other patterns exist. Below is an example of the same pocket using a "zigzag" or "raster" pattern toolpath in Vectric VCarve :



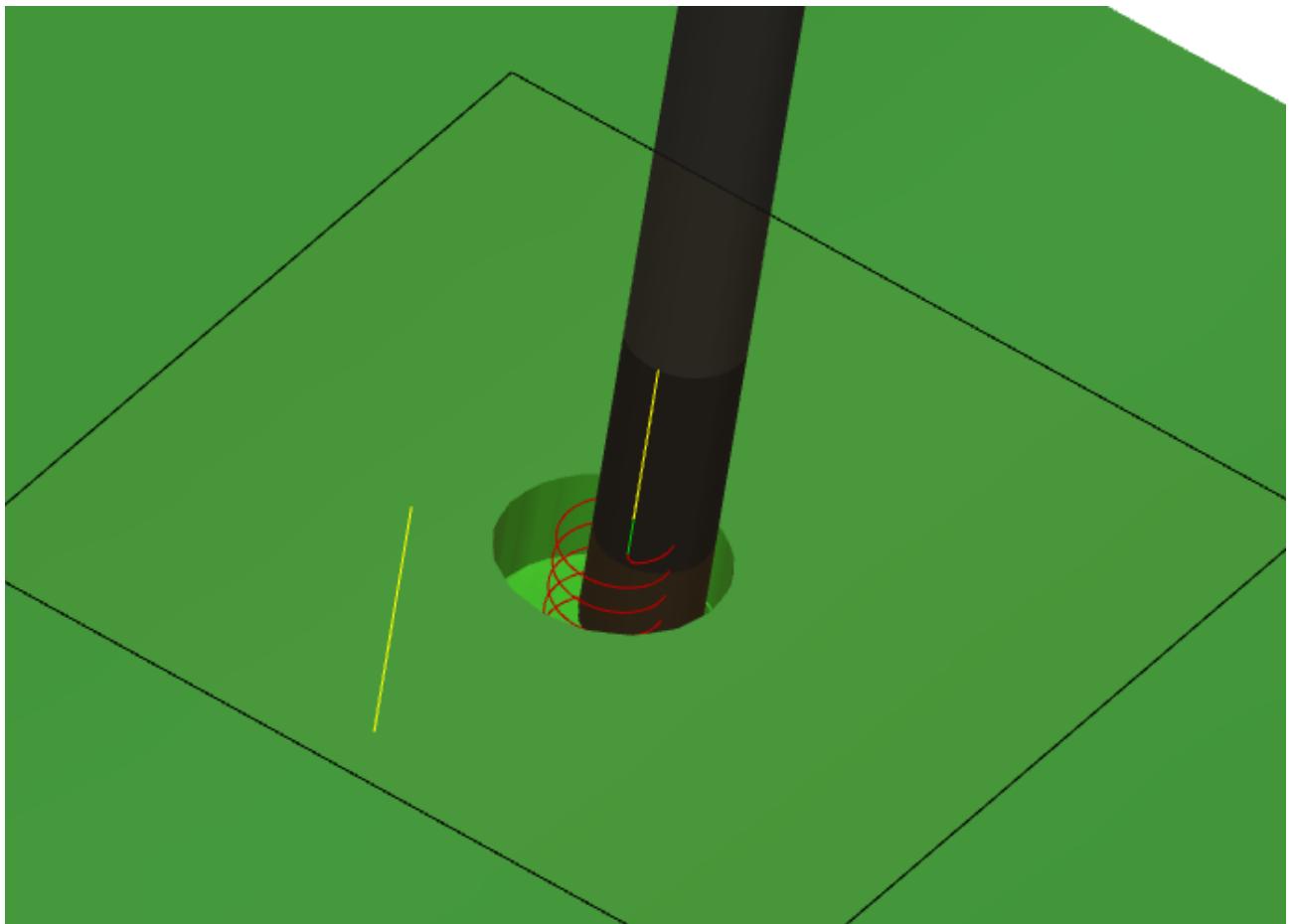
And then there is the matter of the DOC and WOC. As covered in the [Feeds & speeds](#) section, you can either go for a low DOC and high WOC, or high DOC and low WOC.

To illustrate the former, considering the 6mm endmill used in this example and assuming we want to cut a 6mm (0.25") deep pocket, we may want to choose a DOC of 2mm (33% of endmill diameter) and WOC of 3mm (50% stepover), which would result in three passes:

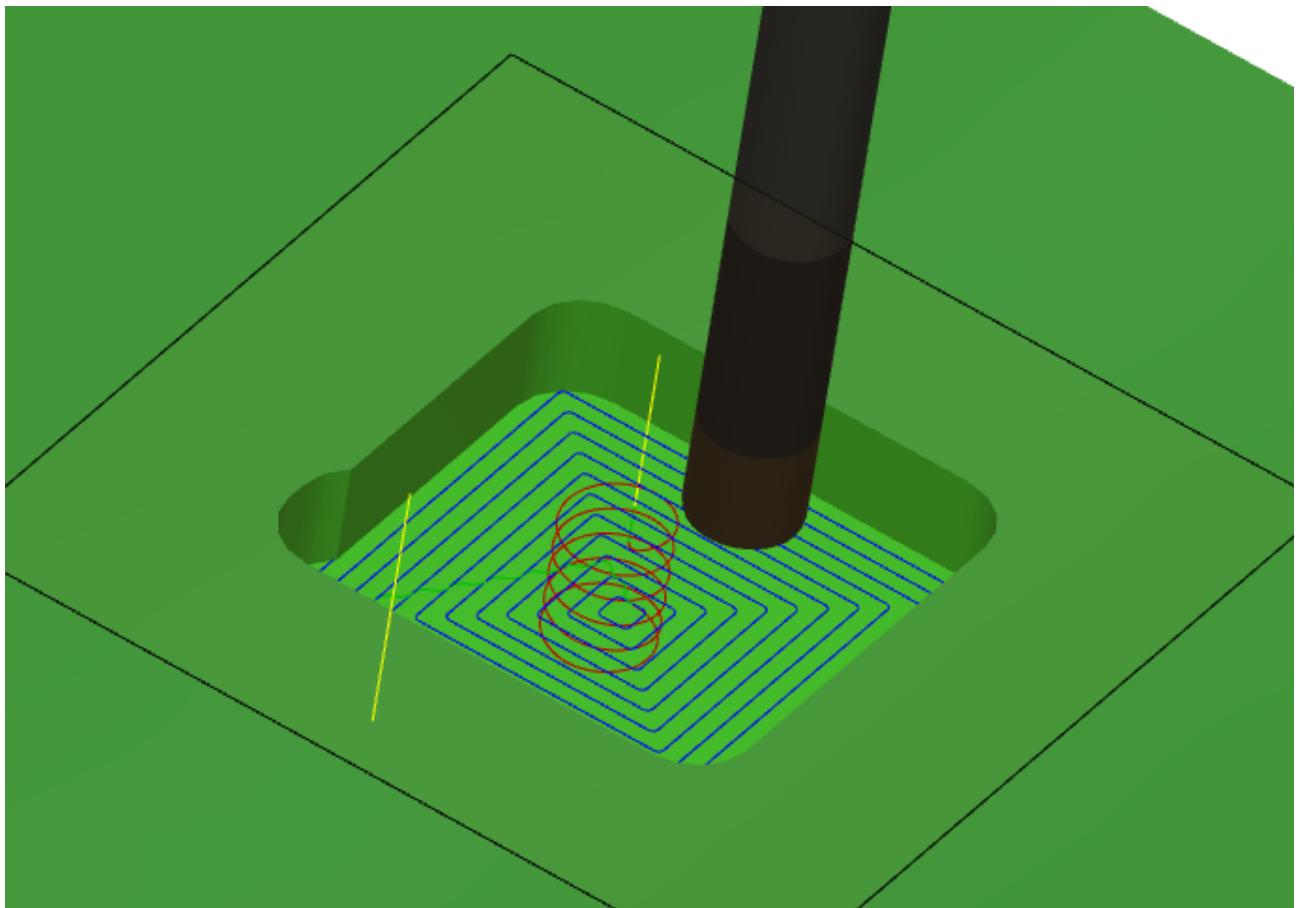


If we wanted instead to go for a high DOC (say full pocket depth *i.e.*, 100% of endmill diameter in this case) and a low WOC (say 20%), we would also need to take care of the initial clearing down to full depth, because just plunging to full depth and starting to cut the pocket would initially involve a slotting cut (100% stepover) at full depth, which for some materials may be too much to take for the Shapeoko.

If your CAM tool allows, you can just use the option of helical ramping down to the full depth of the pocket, as in this Fusion360 example:

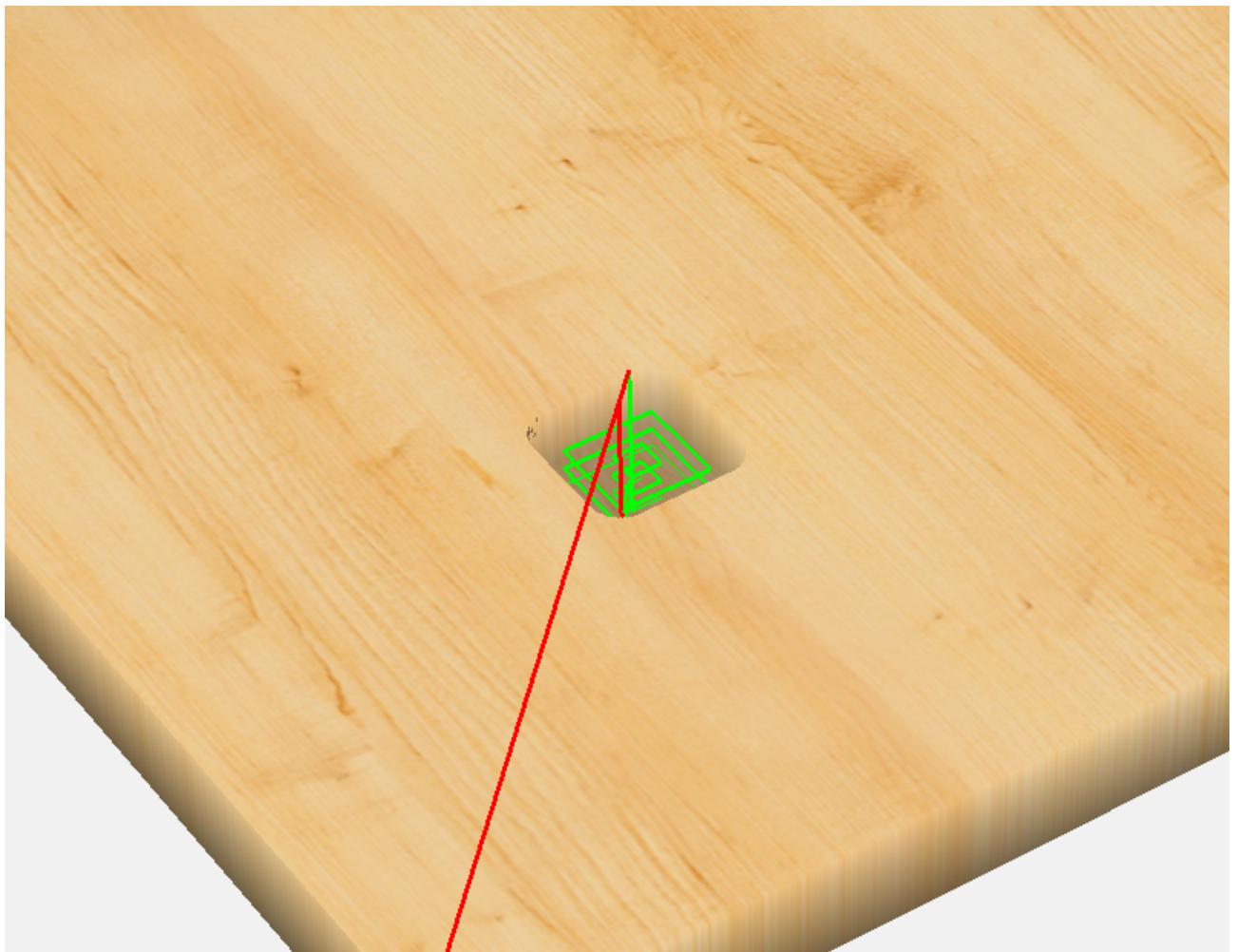


This allows reaching full depth without ever engaging the tool completely, and then the pocket toolpath at full DOC and low WOC (20% of tool diameter in the example below) can proceed:

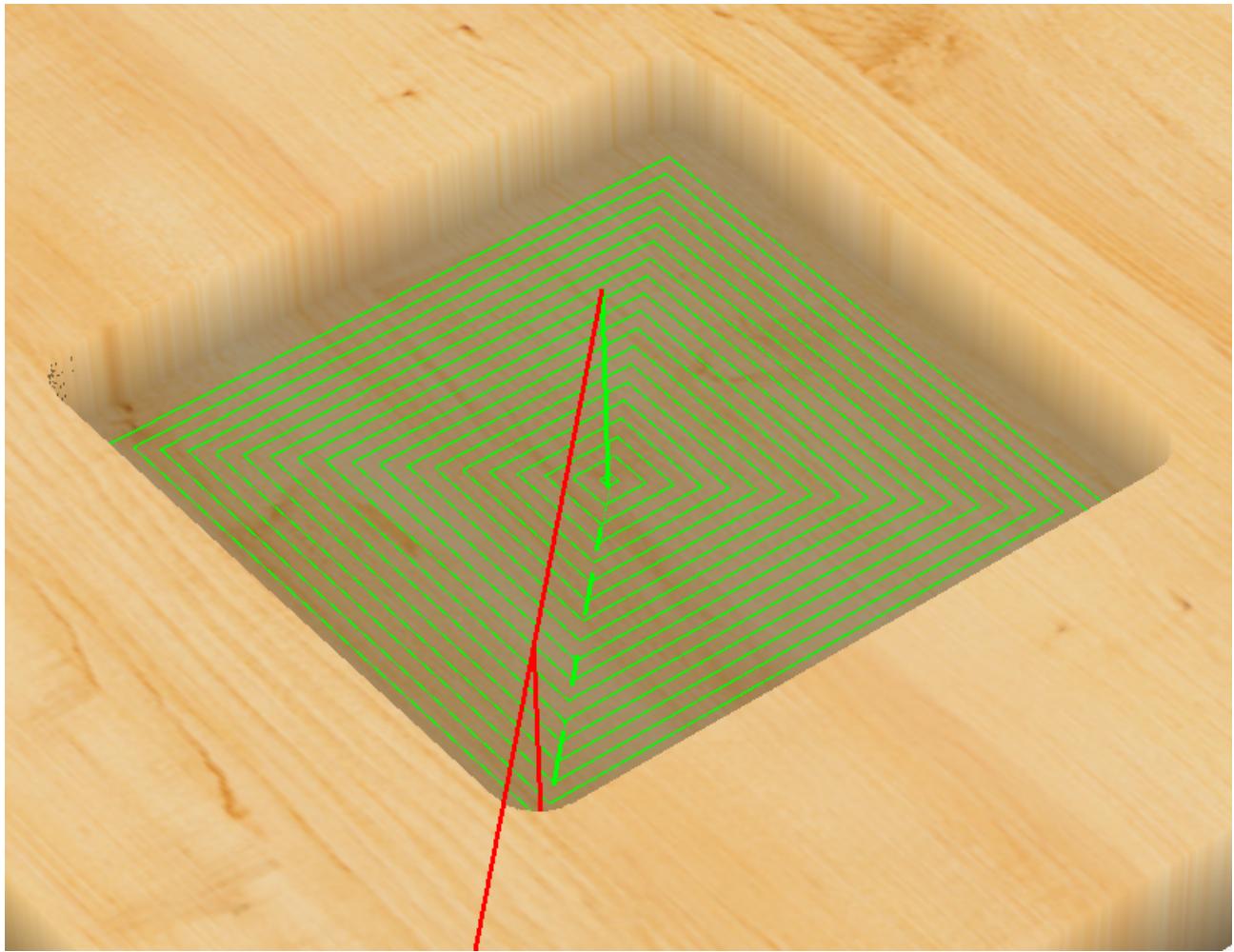


In Carbide Create, you could emulate this by creating two toolpaths:

- a first one using regular low DOC/50%-stepover pocketing, down to full depth, on a small area:



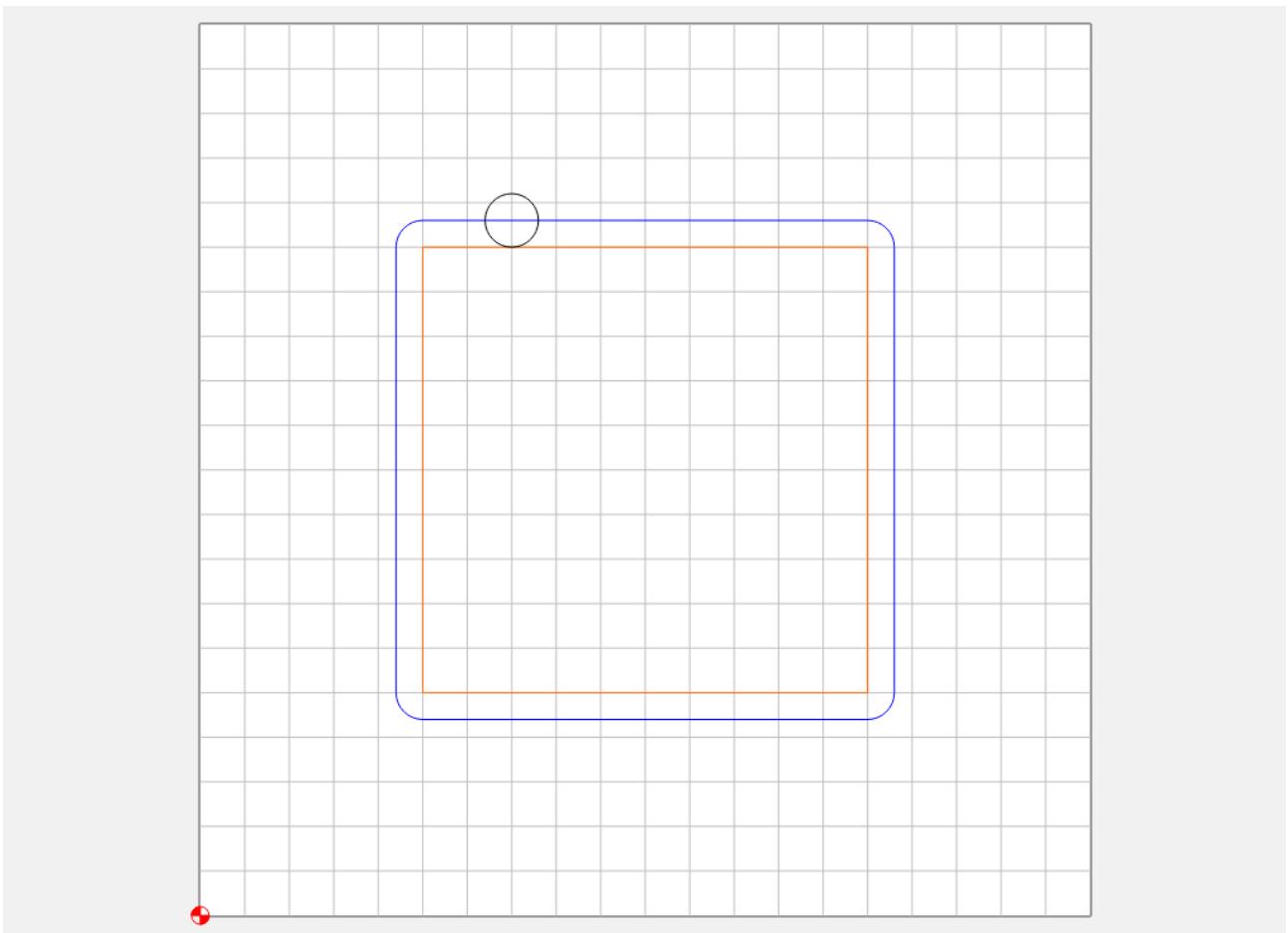
- this would clear the way for a second toolpath doing a single pass at full depth, and 20% stepover:



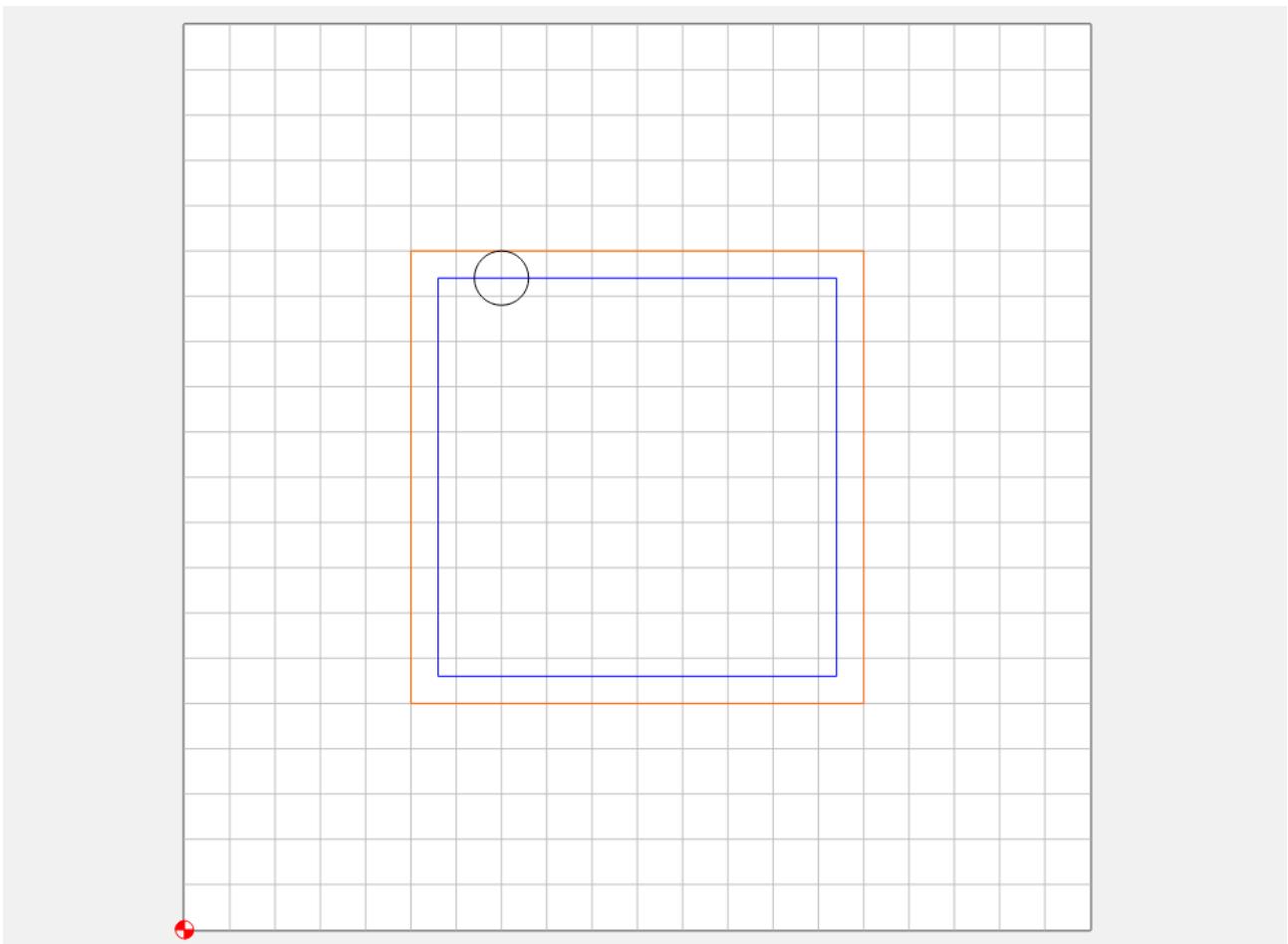
Contour/Profile toolpaths

Contour (a.k.a. Profile) toolpaths just follow the contour of a shape, with the option to have the endmill positioned either on the outside or on the inside of the shape, or right on it:

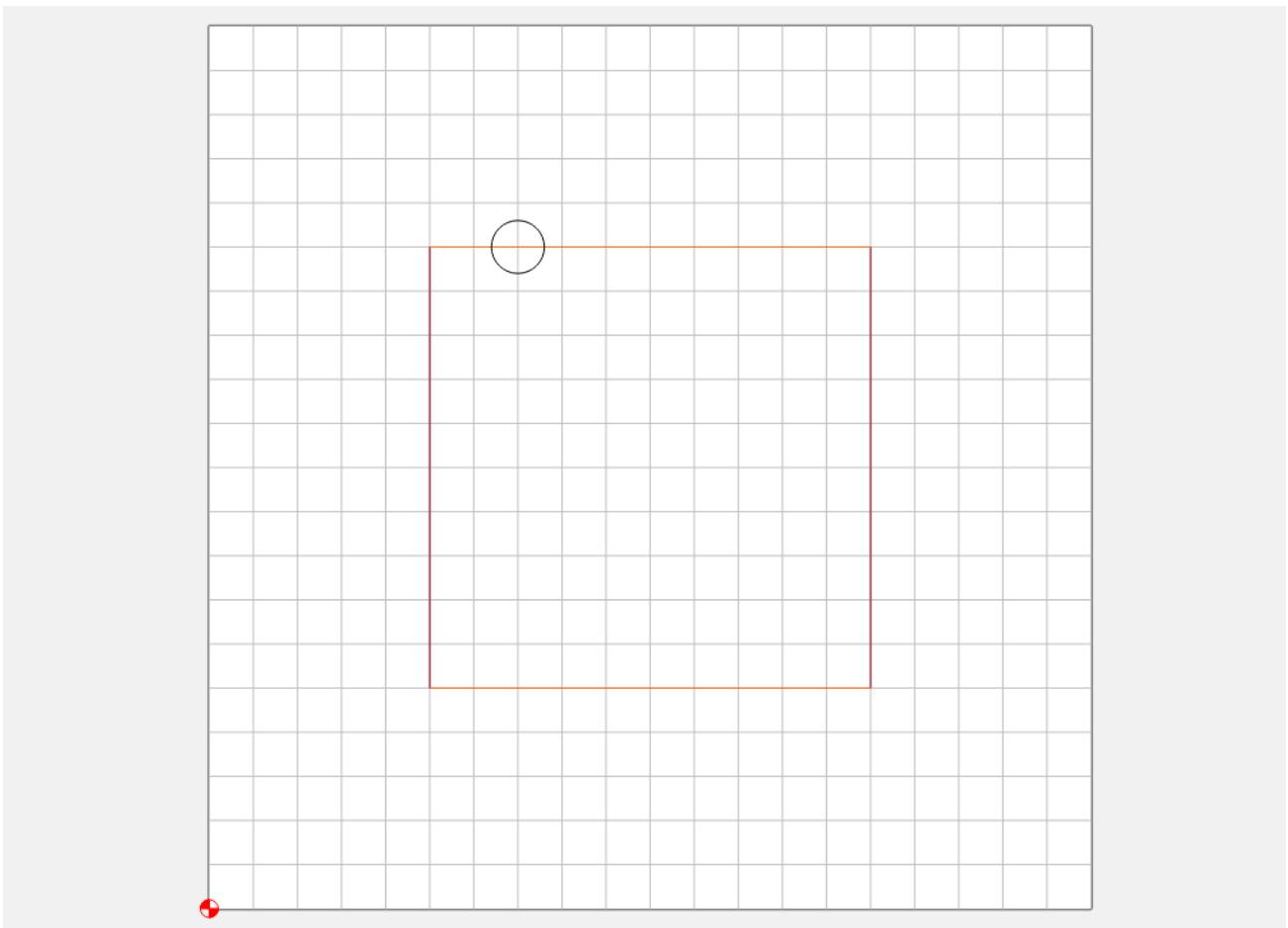
- **outside contours** are usually associated with cutting the shape out of the stock material:



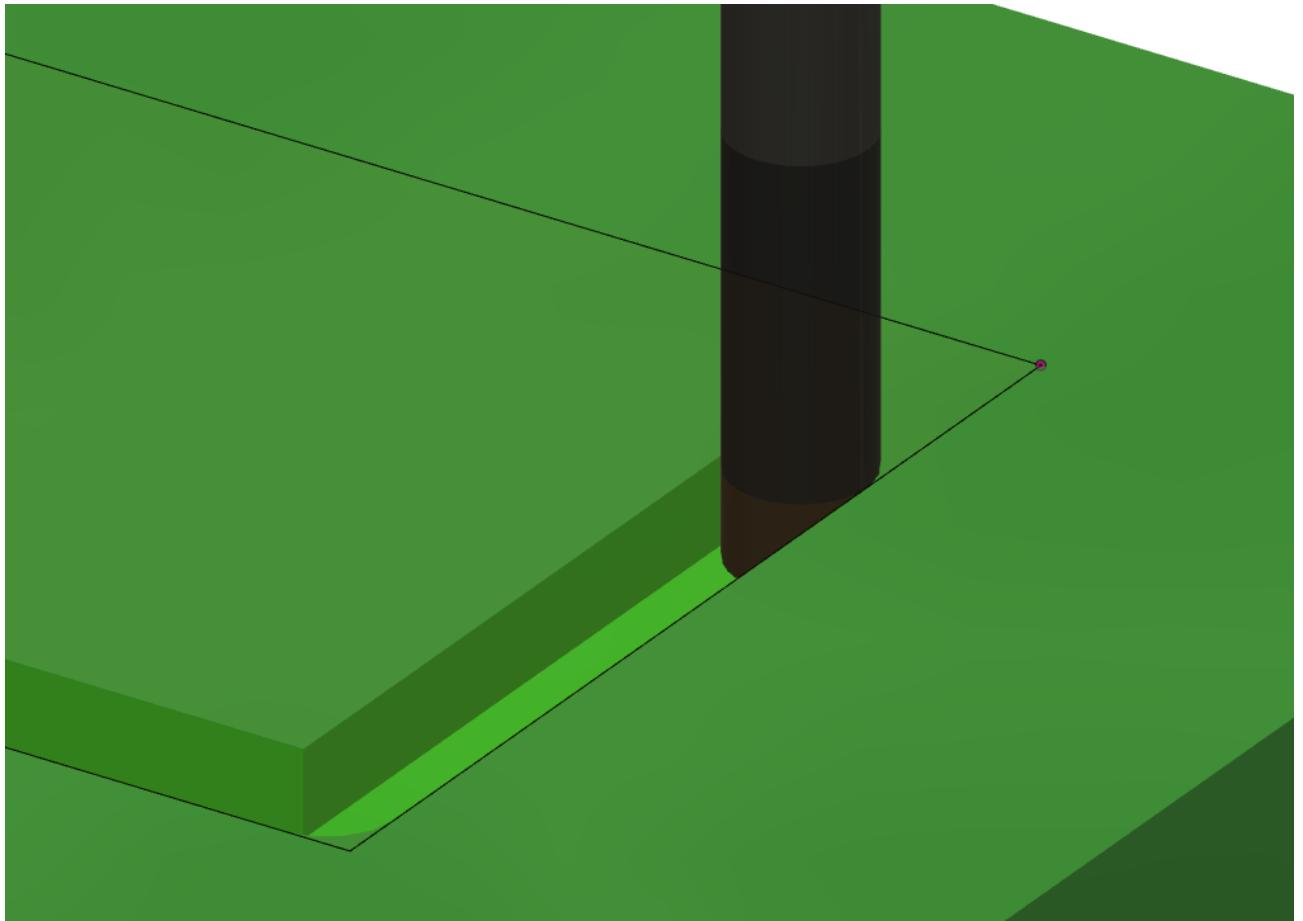
- **inside contours** are usually associated to creating a hole in a piece, the size of the shape:



- **no-offset contours** can be used for example for engraving the shape on the surface of the piece with a pointy bit.

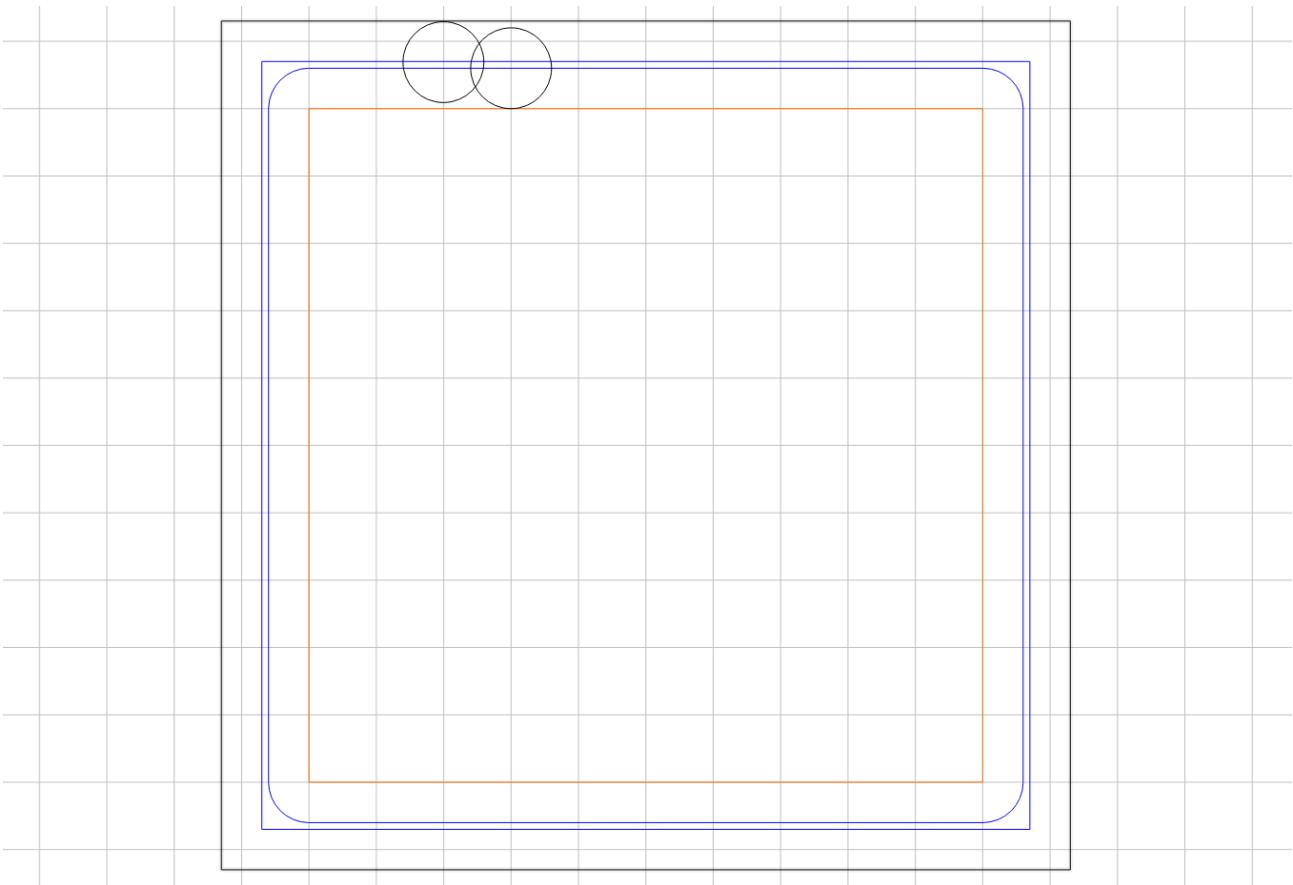


So, contour toolpaths are extremely simple, but there is a small catch: they produce **slotting** cuts:



In this situation, half of the endmill circumference is engaged in the material at all times, refer to the slotting paragraph in [Feeds & speeds](#) section for details. The bottom line is that since this is a worst case scenario for the cutter, the feeds & speeds and DOC need to be dialed back quite a bit to avoid chatter or excessive tool deflection.

Sometimes this is no big deal and you can just proceed with the reduced DOC and be fine. In other situations (*e.g.* cutting plastics or metal), deep slotting can be difficult and a solution can be to avoid it altogether when possible, for example by cutting the profile as a pocket instead of a slot. This usually requires creating additional geometry around the piece, and creating a pocket toolpath to cut material in-between the original shape and the added geometry:



- The orange square is the original shape
- The black square is the extra geometry, created to be larger than the original square by a value of the endmill diameter plus a small margin.
- Selecting the two squares, Carbide Create produces the pocket toolpath shown in blue
- The tool first goes around the inside of the outer square: no benefit there, this results in slotting.
- But then the tool proceeds to follow the second/inner blue path, around the outside of the original square, and this is where pocketing helps: the tool now just has to remove the thin remaining layer of material, and has some clearance against the opposite wall, so this is like a cutting pass with a very small stepover: this is perfect to minimize forces, and act as a finishing pass.

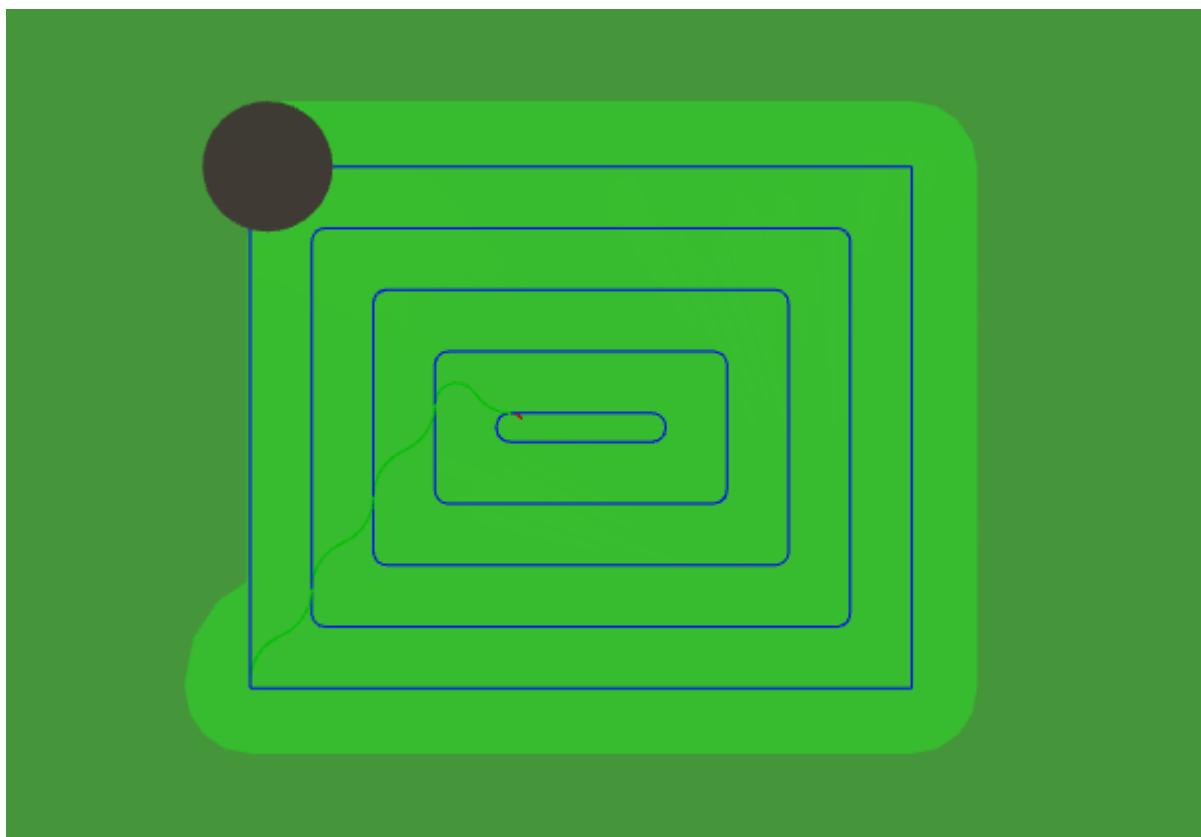
Adaptive clearing toolpaths

Regular pocket and profile toolpaths share a common problem that is not immediately apparent: they involve large tool engagement angles in the material (constantly when

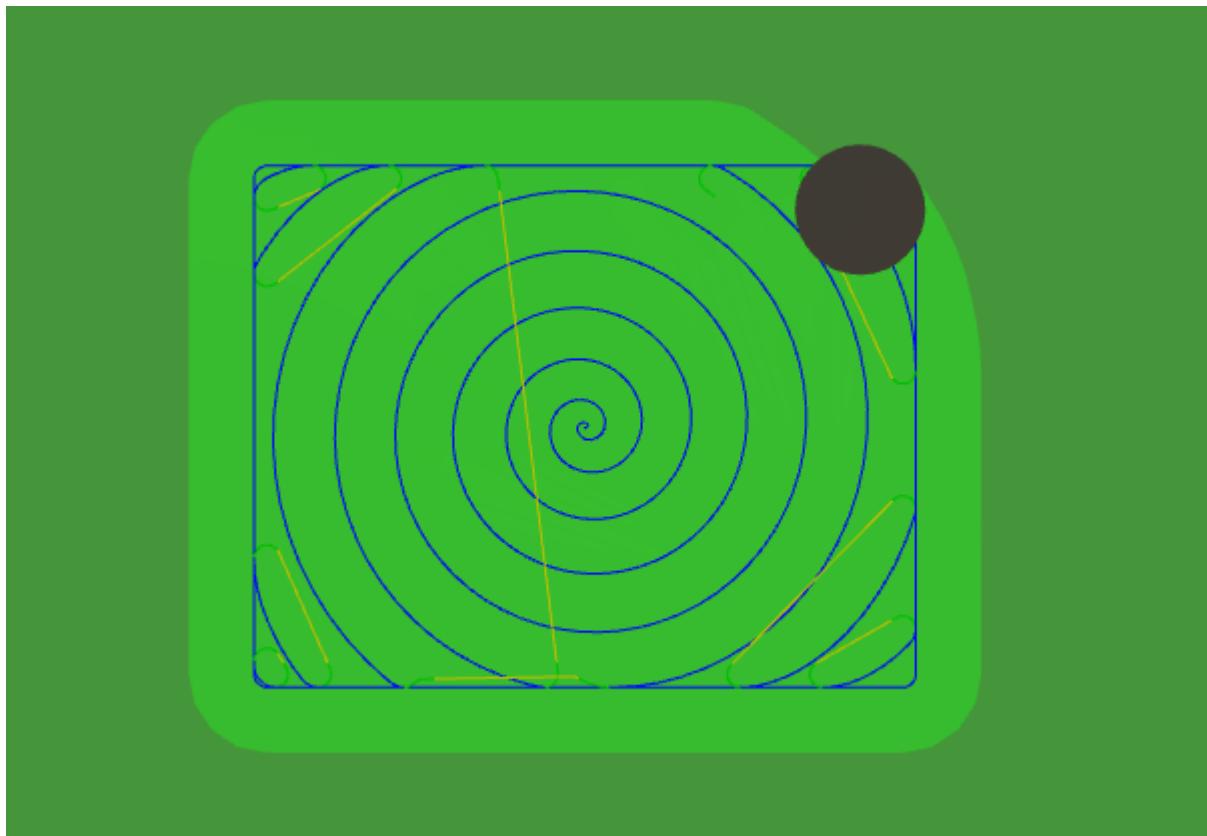
slotting, and temporarily in the corners when pocketing, see [Feeds & speeds](#) section), which in turn limits how deep/fast one can cut.

Adaptive clearing (which is Fusion360's name for their trochoïdal milling family of toolpaths) is another approach that generates toolpaths that have a constant (and relatively low) tool engagement value throughout the cut. When going straight, this is the same as using a very small stepover. When cutting corners, this means taking many small scoops of material, instead of going straight and taking a sharp 90° turn.

While a pocketing operation using a stepover of 60% would look like this (notice the ~180° tool engagement in the corner):

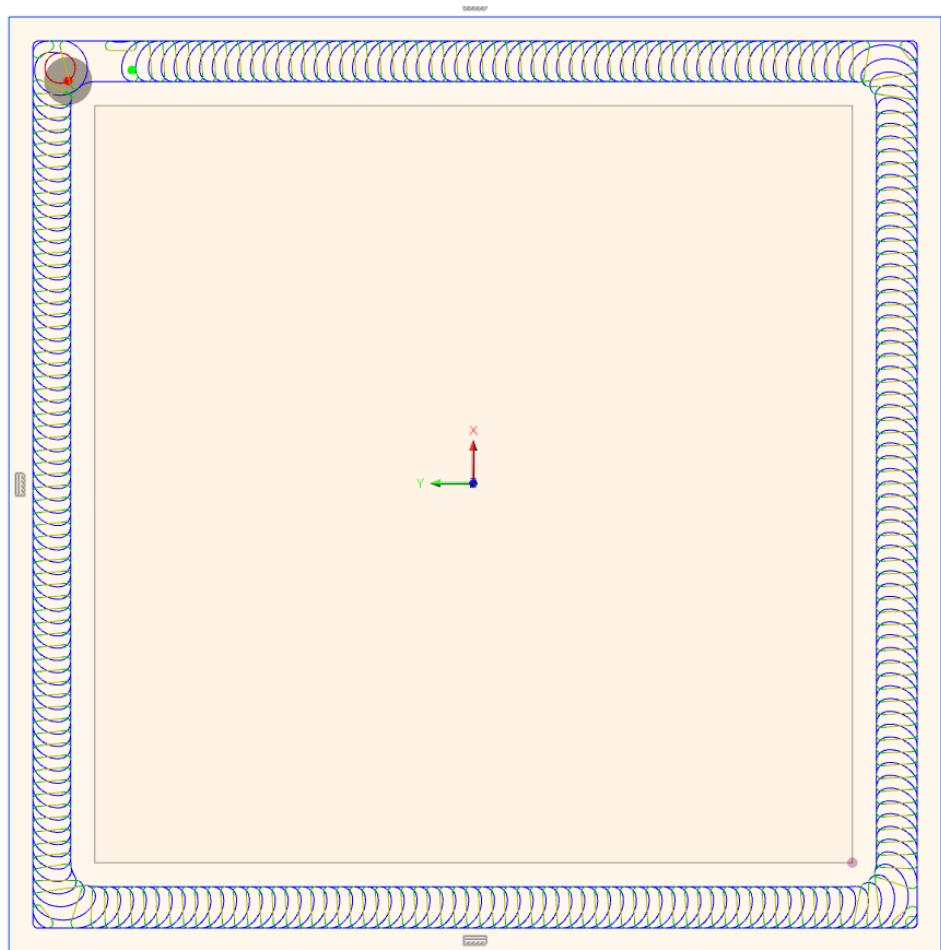


an adaptive clearing toolpath on the same geometry with radial width of cut of 60% would look like this (notice how the tool is turning before the corner, keeping tool engagement to ~90°):

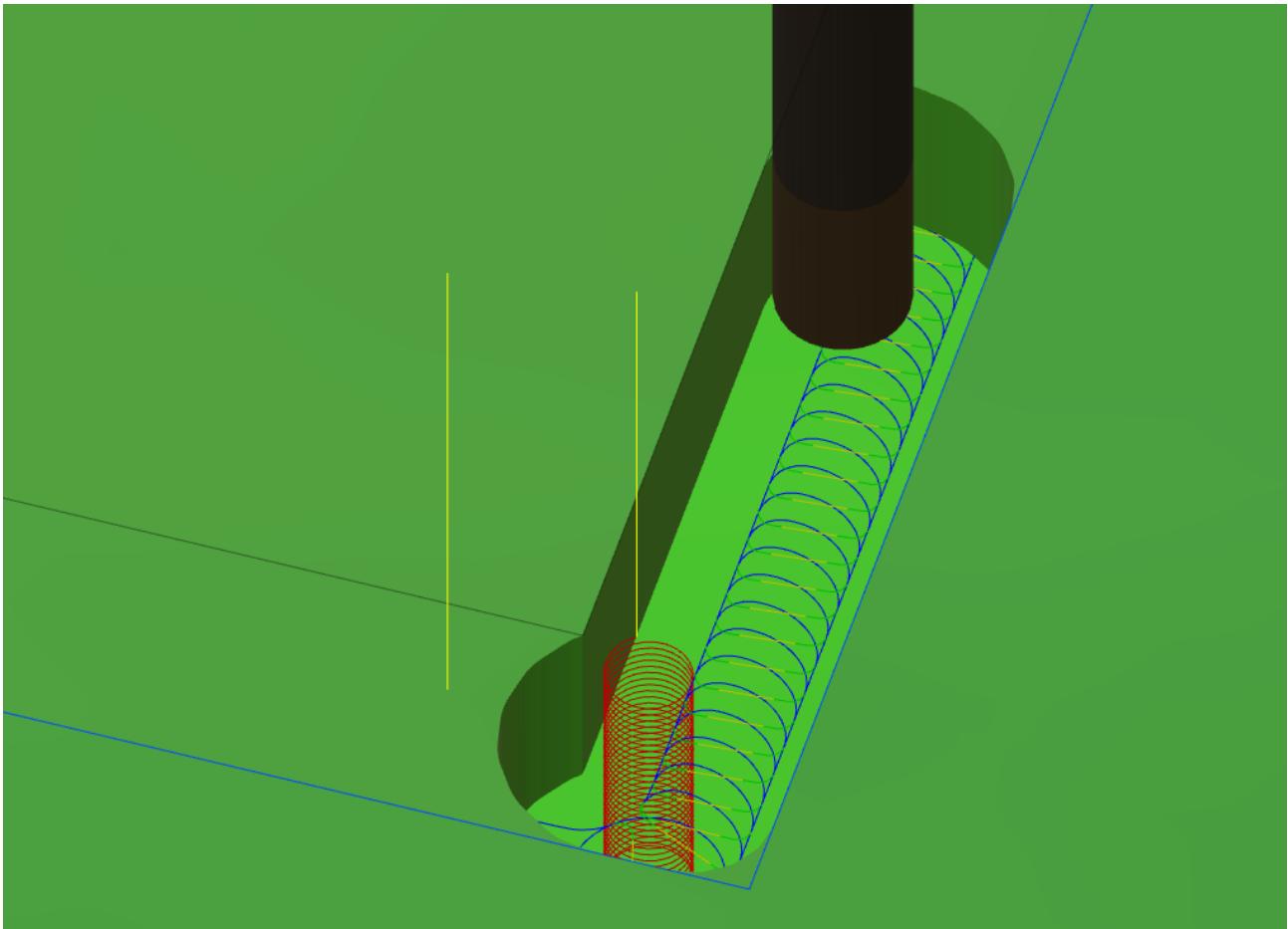


(i) In Fusion360, the radial width of cut (stepover) parameter is called **optimal load**

Coming back to the profile cut example discussed earlier, adaptive clearing can be leveraged to avoid slotting. Using extra geometry around the original shape and creating a 2D adaptive clearing toolpath results in something like this:



After it plunges to the depth of cut (possibly using helical ramping), the endmill is moved in small spiral movements between the inner geometry and the outer geometry: at any given time, the endmill engagement into the material is constant (and small):



Since the cutter engagement is low, one can use much more aggressive feeds and speeds (but of course, it takes longer to do all these spiral movements rather than going in a straight vertical line, so whether or not this results in a longer or shorter job time depends on the situation)

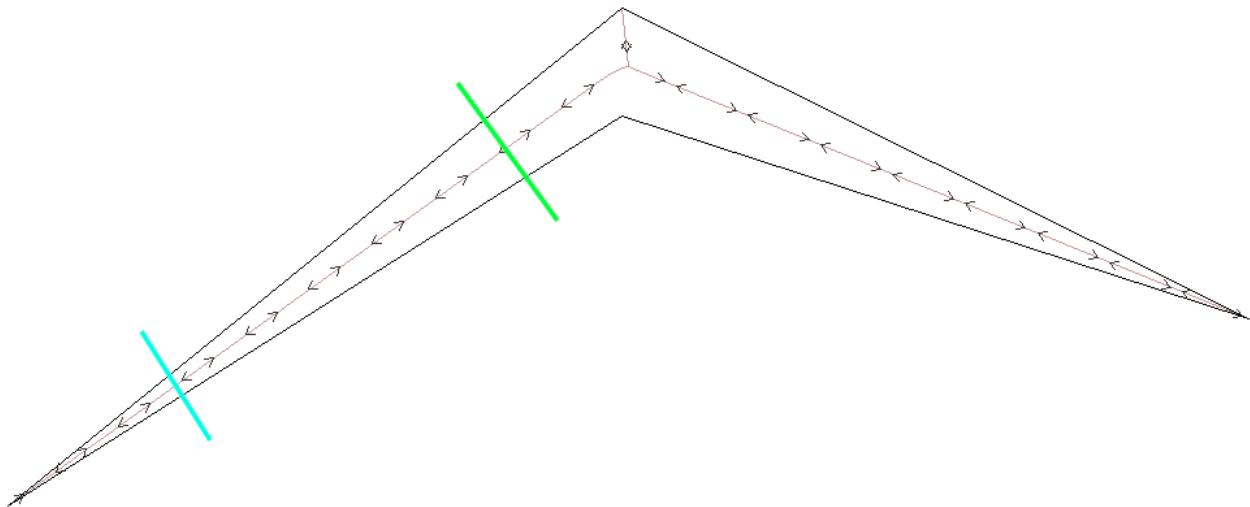
For the same reason, the DOC can also be increased very significantly *i.e.*, adopting the "high DOC, low WOC" approach.

- (i) Since adaptive clearing is typically used with a small WOC (much lower than 50% of the endmill diameter), chip thinning must be taken into account in the feeds and speeds, as described in the [Feeds & speeds](#) section

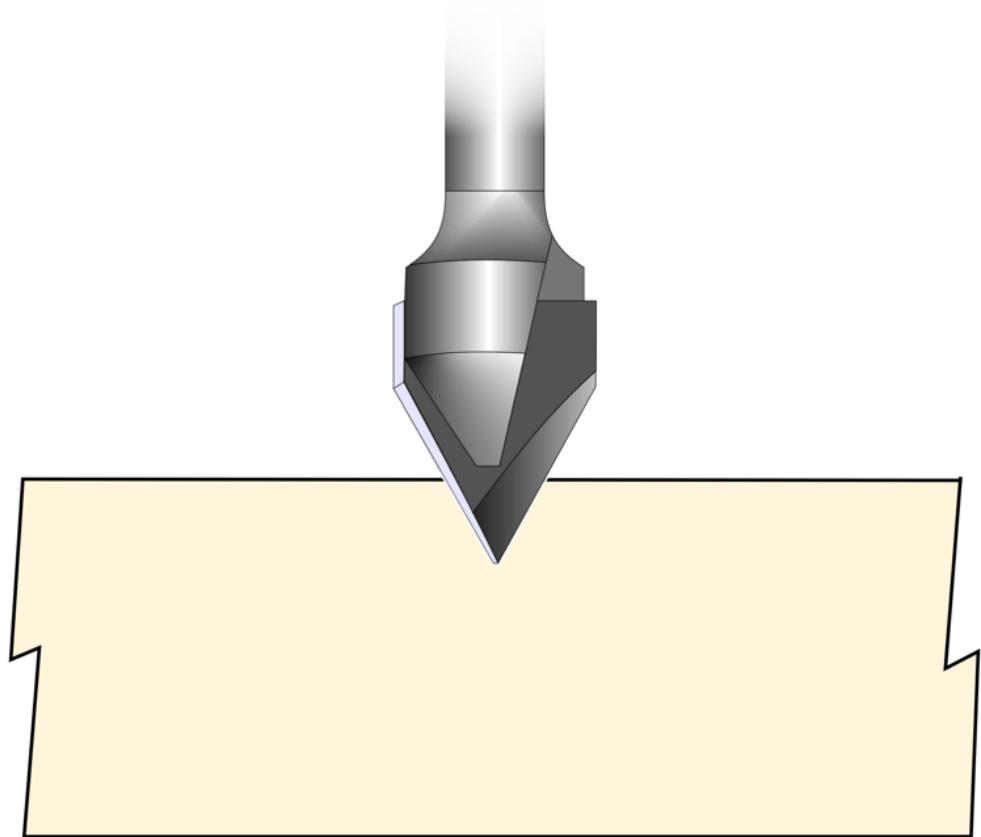
V-Carving toolpaths

V-carving is a specific toolpath strategy dedicated to using V-bits to cut the inside of shapes, resulting in a cut depth that depends on how far apart the sides of the shape are.

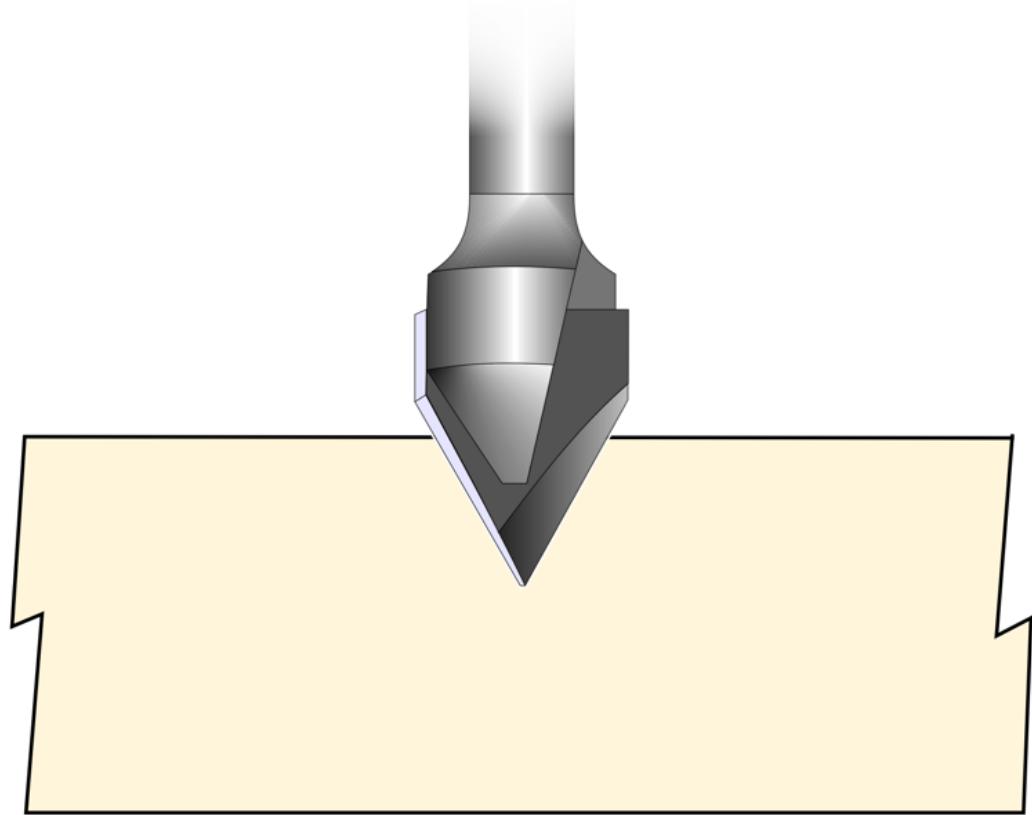
Let's take the example of a simple chevron shape, to be cut with a 60 degree V-bit:



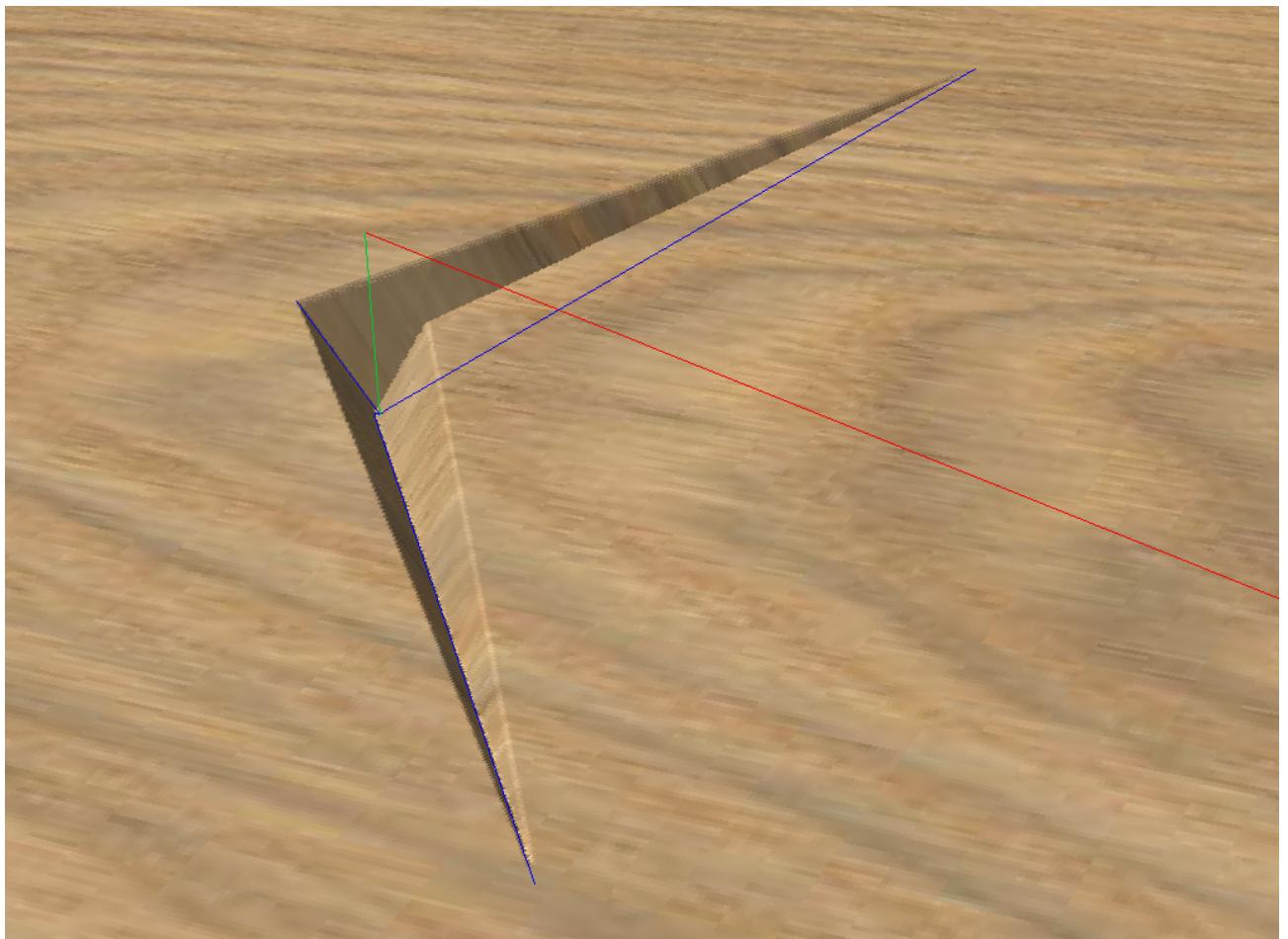
- the toolpath (pink) is where the tip of the V-bit will be
- if we look at a vertical cross-section where the blue line is, the V-bit will be like this:



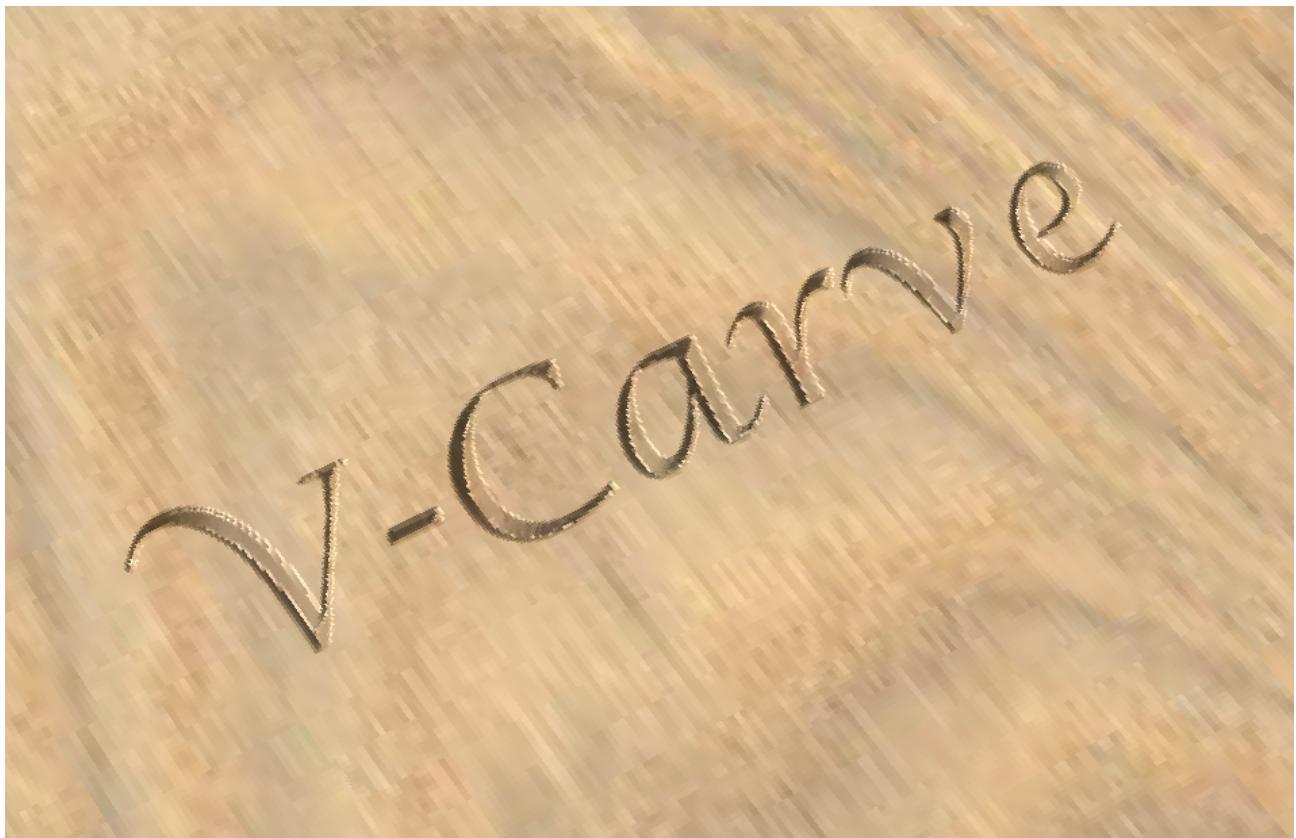
- while at the level of the green line, the cross-section would look something like this:



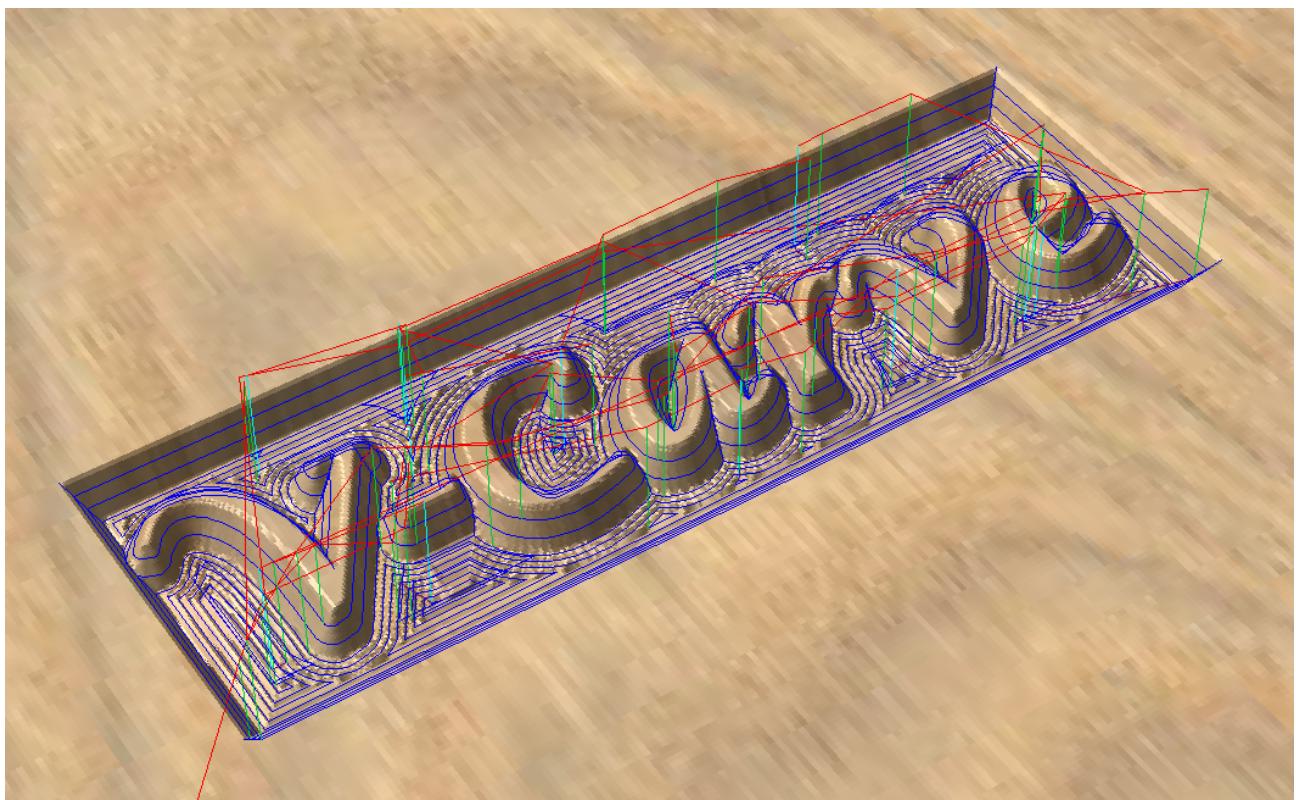
Here's a preview of this toolpath, the width of the cut is controlled by how deep the V-bit is driven:



A typical use is to carve the inside of text characters:



Carving the outside of characters within a predefined boundary is another popular option, and when combined with the ability of some CAM tools to limit the v-carve depth to a predefined max value, it can produce interesting 3D-like results:



- i** a V-bit is very inefficient at removing large amounts of material, and even worse at carving flat-bottom pockets, so the scenario above would better make use of a first clearing toolpath using a regular endmill in flat areas, and a second toolpath taking care of V-carving around the characters.

Roughing vs. finishing toolpaths

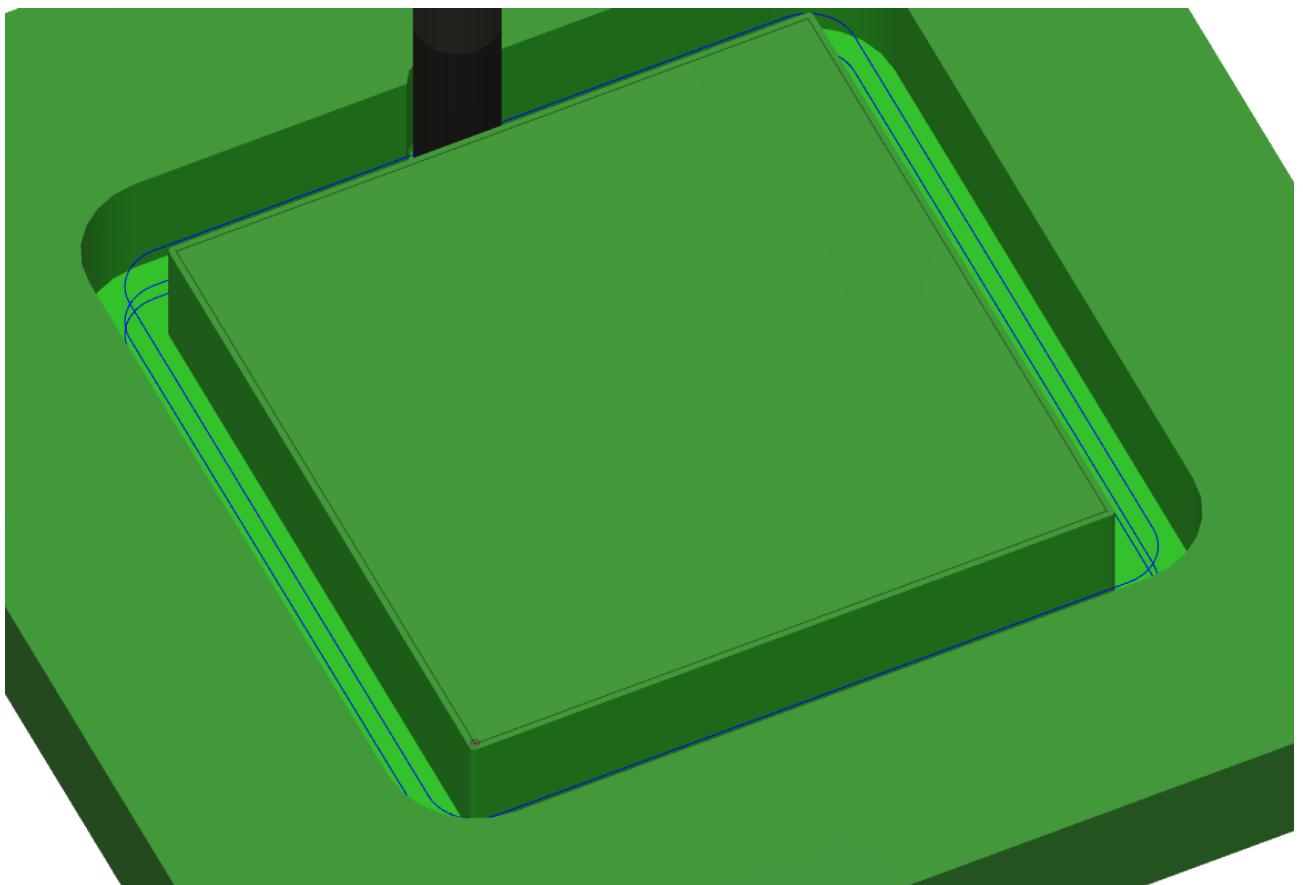
For every toolpath there are two conflicting needs: minimizing the total runtime, and getting the best finish quality and dimensional accuracy of the workpiece. Instead of settling for a middle ground that accommodates both constraints, a very common approach is to create two different toolpaths:

- the first (roughing) toolpath will be optimized to go fast, maximising material removal rate, but will be programmed to leave a little bit of material around the selected geometry. Even if the tool deflects, vibrates, or more generally produces a poor surface finish, this will be taken care of by the next toolpath.
- the second (finishing) toolpath will be optimized for getting a good surface finish and accuracy: it can be set to go slower, but even if it isn't, the simple fact that it will have very little material left to cut will reduce the efforts on the tool, and produce a cleaner result.

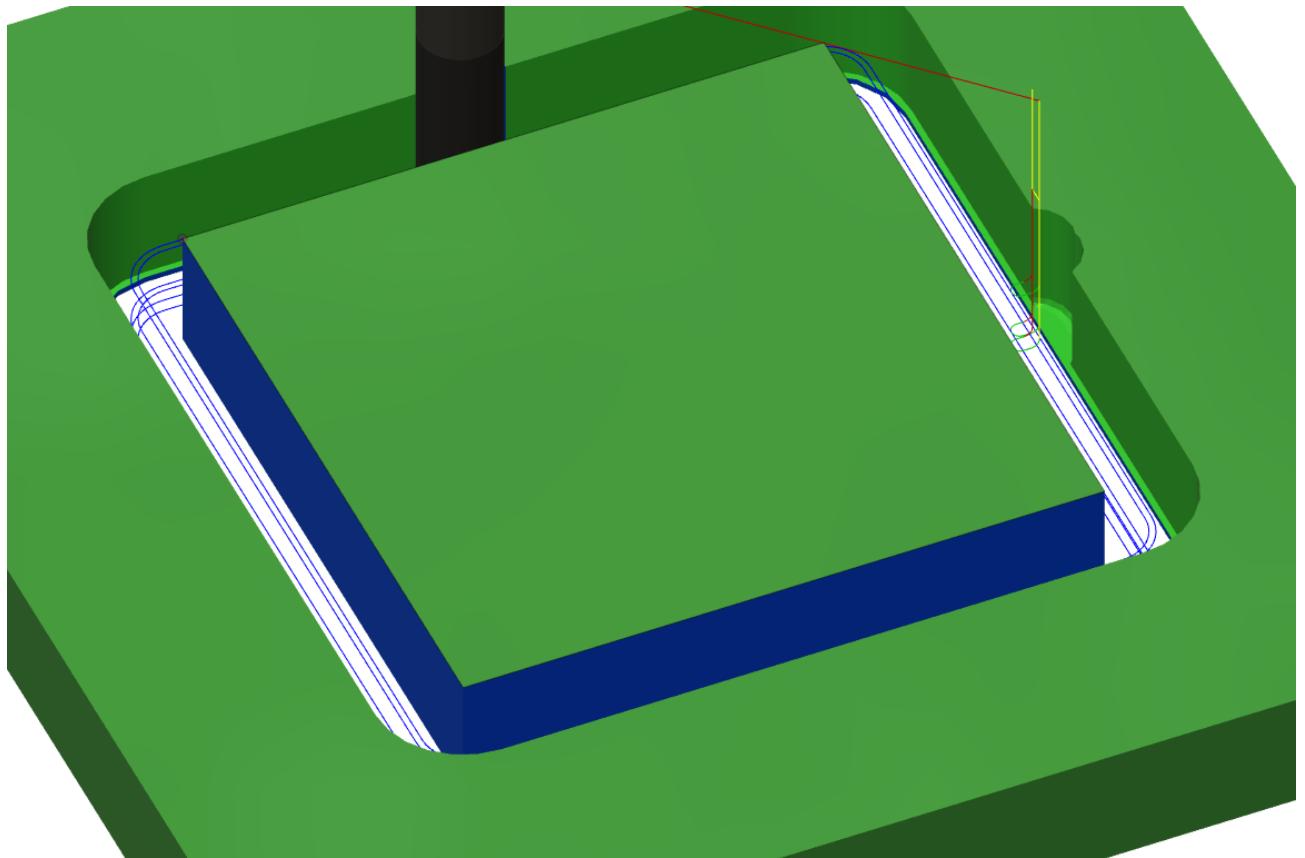
How roughing/finishing is set up depends on the CAD/CAM tool being used:

- Carbide Create does not support (at the time of writing) any roughing/finishing option explicitly, but:
 - one can manually create additional geometry in the design to do it, as explained above in the alternatives to slotting.
 - or one can define a "fake" roughing tool and declare it to be slightly larger than the tool actually is, and use that tool in the toolpath: this will generate a toolpath that when cut with the real (slightly smaller) tool, will leave a thin layer of material around the shapes. Then, generate toolpaths based on the same geometry, but this time using a tool that is declared to have the true size, and run that: it will act as a finishing toolpath and shave off the excess material from the "roughing" pass.

- Vectric VCarve has explicit options in its toolpath parameters to create an "allowance offset", basically a margin that will be kept when cutting. One can therefore select a geometry and:
 - create a first toolpath with an allowance offset (of say 0.01")
 - create a second toolpath with an allowance offset of 0.
 - for profile toolpaths, this is even simpler: there is a "**Do Separate Last Pass**" option with an allowance offset value, to tell VCarve to use the allowance value for successive passes down the material, but for the last (deepest) pass, discard the allowance: this will result in the endmill taking a single full-depth pass all around the geometry at the end, shaving off the material to get to the final dimensions and finish quality.
- Fusion360 has a "**Stock to leave**" option in the toolpaths, which works similarly:
 - create a first toolpath with *stock to leave* at a small value. Both the radial ("vertical") and axial ("horizontal") stock to leave values can be specified. Here is an example using 0.5mm radial and axial stock to leave in an outside profile toolpath: notice how at the end of this toolpath, there is a small amount of extra stock remaining around the center back square:



- create a second toolpath, identical to the first one except setting radial and axial stock to leave to 0: this will shave off the extra material, to reveal the final walls of the workpiece (in blue) as well as remove the remaining 0.5mm at the bottom:



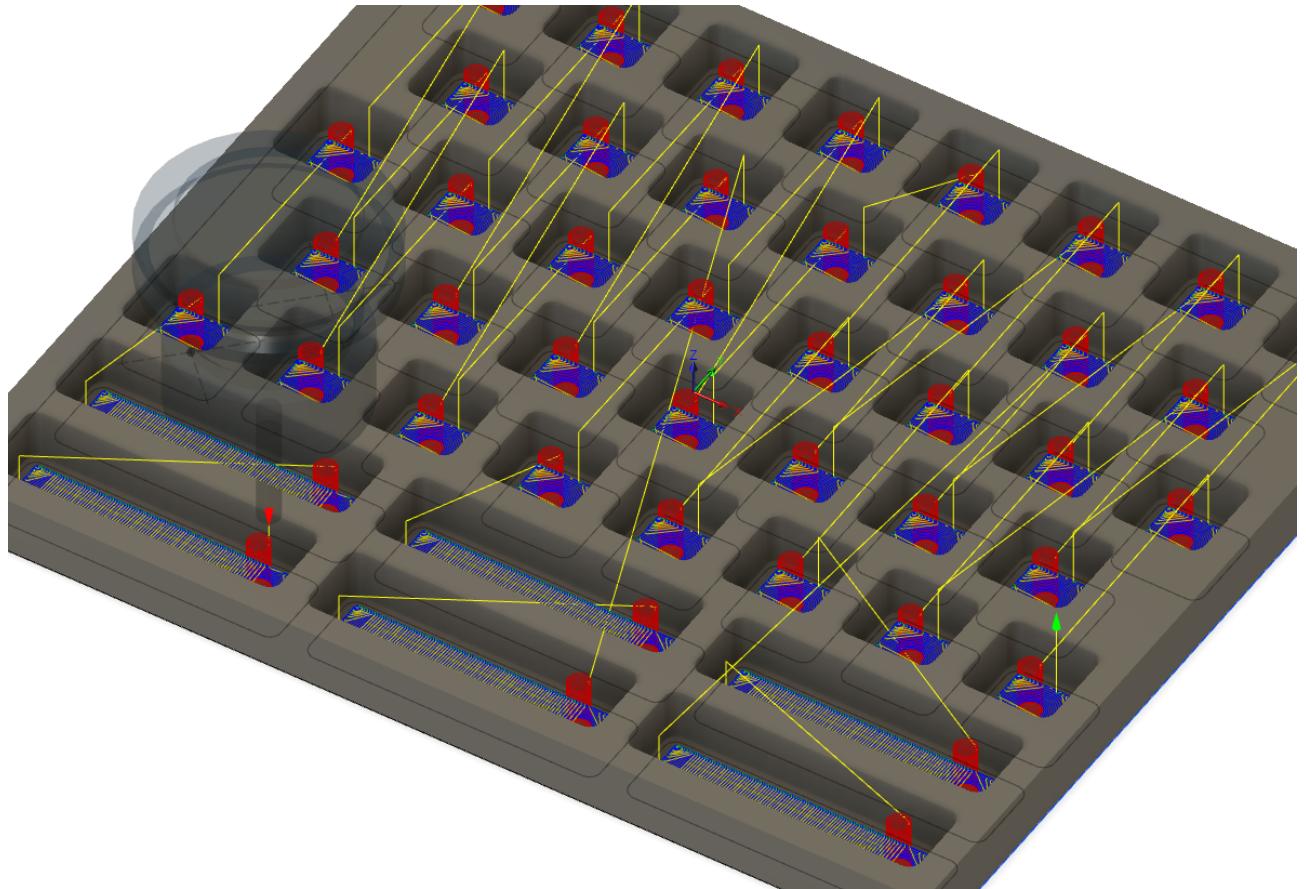
REST machining

A very common scenario is to first use a large tool to remove a lot of material quickly, and then switch to a smaller tool to cut finer details. CAM tools that support REST machining can optimize toolpaths such that the tool only works in the areas where material was not already removed by the previous toolpaths.

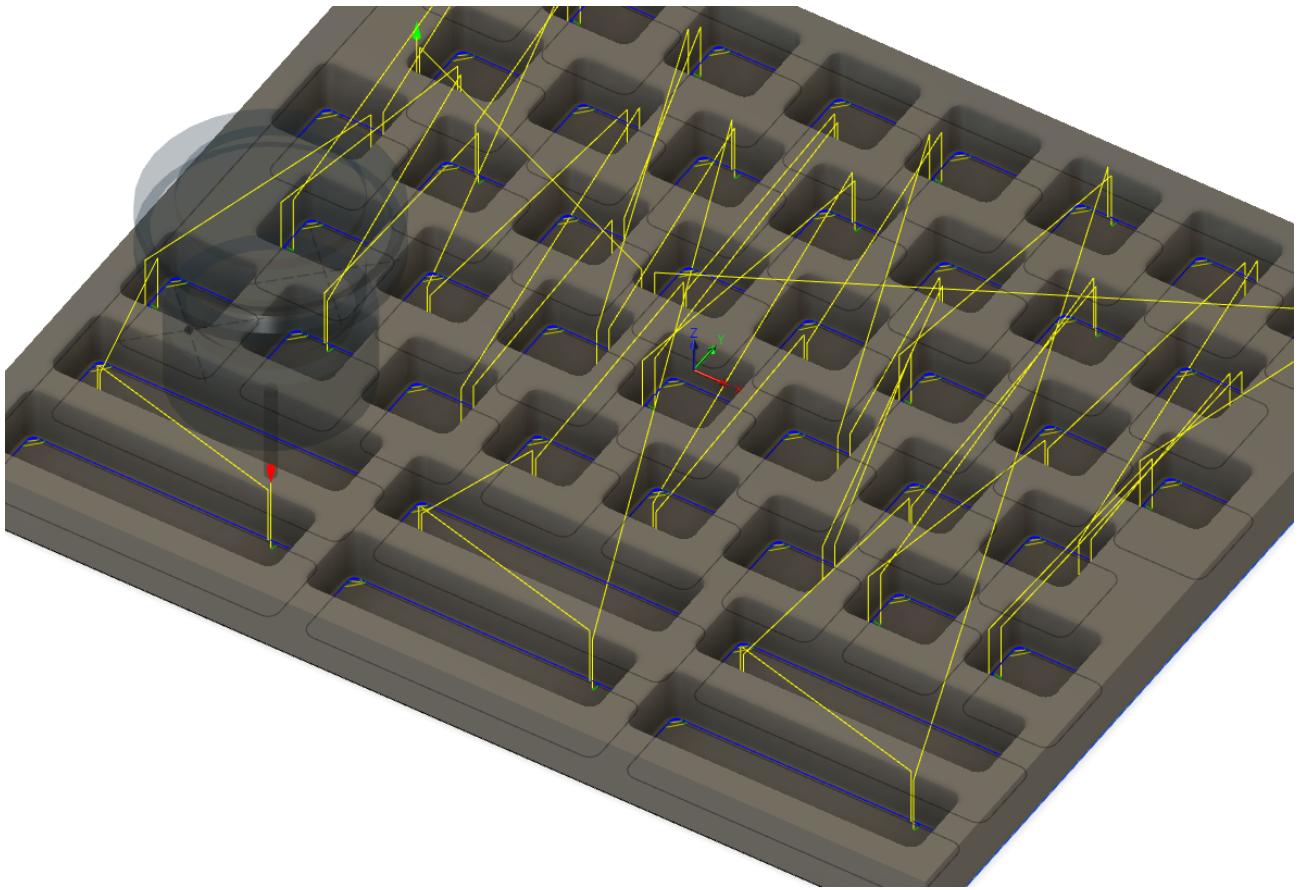
Consider the case where a large pocket must be cut, but the pocket has tight corners radius:

- a large endmill will be more efficient at removing material quickly, but will not be able to reach into the tight corner
- a small endmill will be able to reach the corner, but would be very inefficient at cutting the whole pocket

In the example below a 6mm (0.24") square endmill is first used, with aggressive feeds and speeds to clear out a lot of pockets quickly, and with some stock to leave:



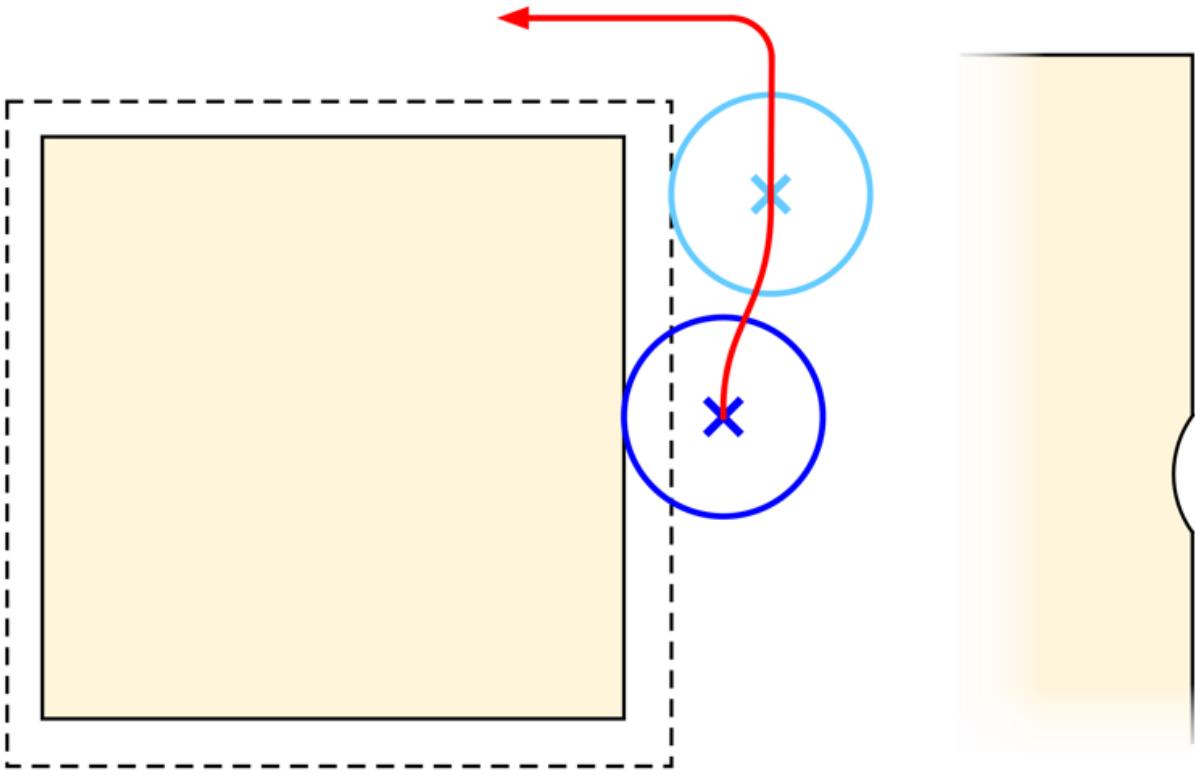
A second toolpath using an 1/8" endmill and REST machining option, with no stock to leave, takes care of cutting (only) the tight corners as well as removing the remaining stock on the pocket walls (i.e. finishing)



The power of REST machining lies in the fact that both toolpaths refer to the same geometry (the pocket outlines), there is no need to manually create any additional geometry or constraints to restrict the second toolpath to working on the walls and corners only.

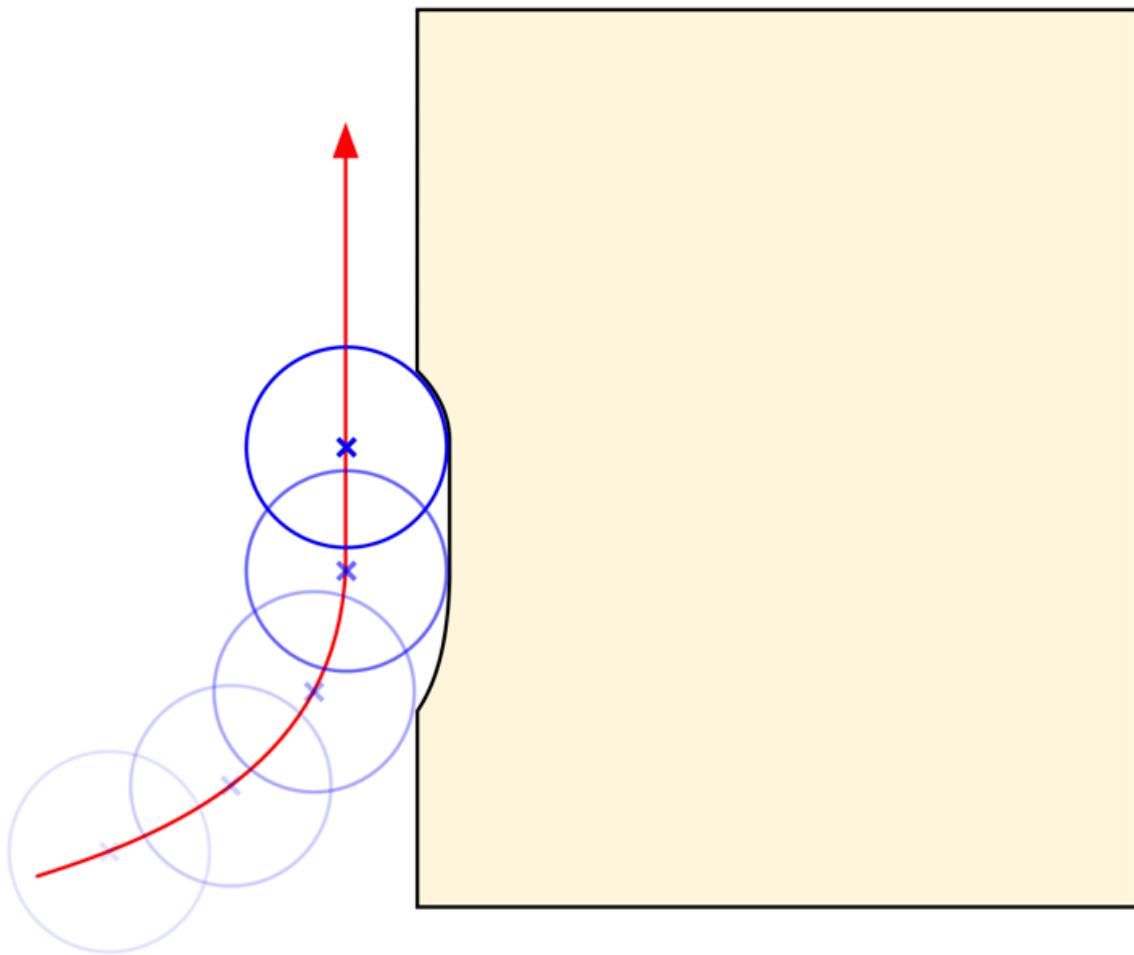
Lead-in/Lead-out

Consider the case of a profile cut, where the tool plunges straight down at a point somewhere along the profile. During the plunge, the forces on the endmill are mostly vertical, the tool will not deflect. But once the endmill has reached the DOC and starts moving around the profile, lateral forces on the endmill will cause deflection, and the actual cutting path will deviate from the intended path by a tiny amount (greatly exaggerated on the sketch below). The resulting profile cut may then end up having a (small but) visible notch at the point where the endmill plunged into the material:



One way to avoid this is to use a roughing pass with stock to leave: the deflection effect will still happen, but it will happen away from the contour, and a finishing pass (with very little deflection) will then come and shave off this small variation.

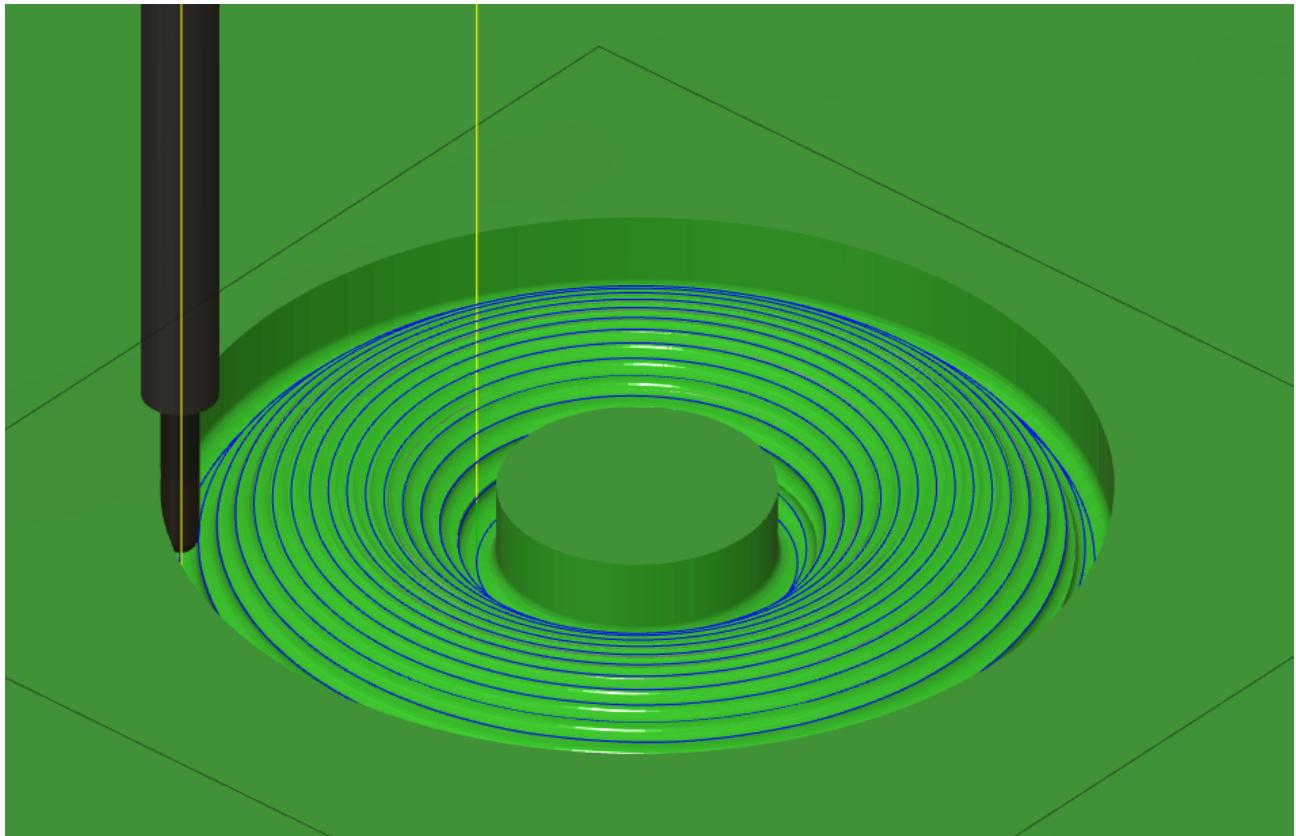
Another way to deal with such problems is to use **lead-in** (respectively lead-out) options if the CAM tool supports it: the toolpath will make the endmill plunge away from the profile, and then lead into (respectively out of) the profile edge:



At the time of writing, this feature is not supported in Carbide Create, but one can fallback to manually adding geometry around the piece to achieve similar results.

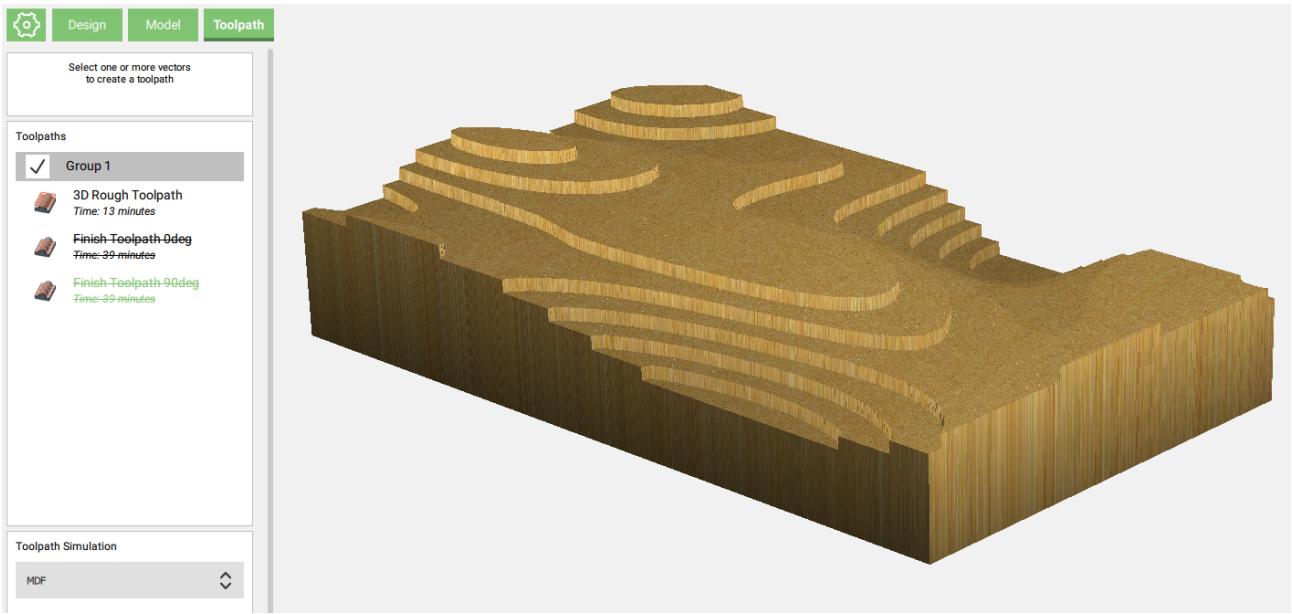
3D toolpaths

The standard version of Carbide Create does not support them, and the topic is too wide and too specific to be covered here properly, but here is a simple example of milling a donut shape in Fusion360:

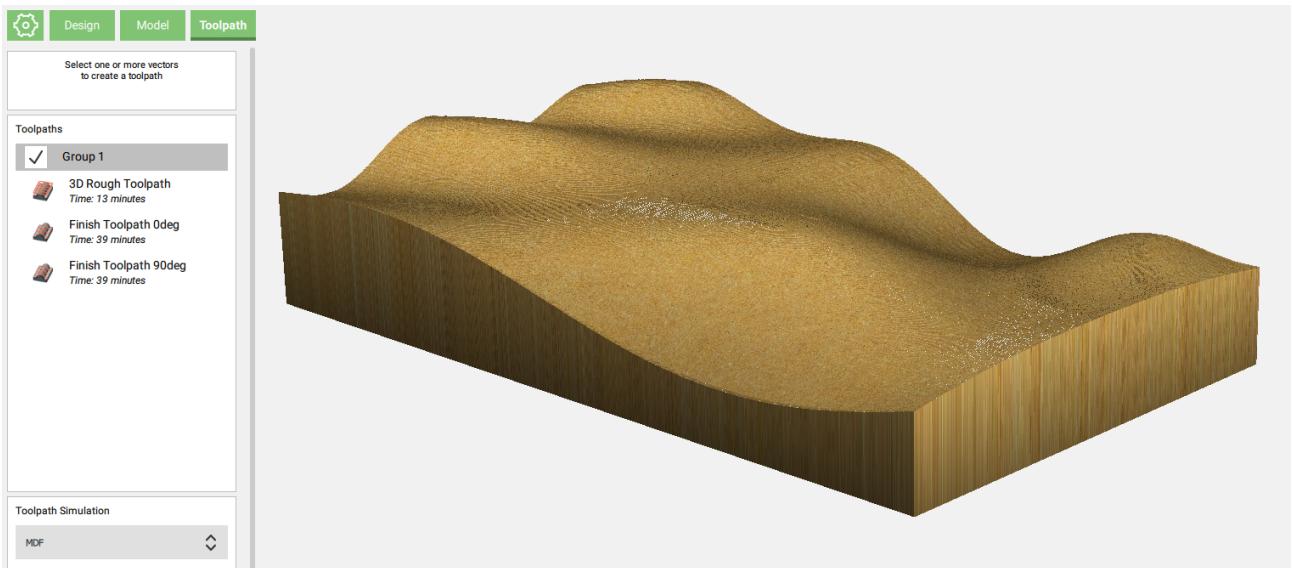


Many of the concepts of 2D toolpaths apply, but the notion of roughing + finishing will be paramount for 3D, to get both a reasonable job time and a smooth finish. Typically, a large diameter square endmill will be used for roughing, and a small diameter ballnose will be used for finishing.

Carbide Create Pro includes support for creating 3D toolpaths from 2D features or from grayscale heightmaps. Since many 3D models lend themselves well to being projected to a heightmap, this opens up the possibility to do very intricate 3D carvings. Here is a simple example of milling a 3D surface in CC Pro, starting with the 3D roughing pass:



And then adding 3D finishing passes:



Drilling toolpaths

For drilling a hole, a first option is to use an endmill smaller than the hole, and use a circular pocket toolpath. It works fine, but turns out to be inconvenient:

- if a job that could otherwise be done completely with e.g. a 1/4" endmill needs 1/4" holes, a 1/8" endmill will be required just to mill the hole pockets.

- small endmills usually have a short length of cut and shank, so cutting a deep 1/8" hole with a 1/16" endmill could turn out to be impossible.

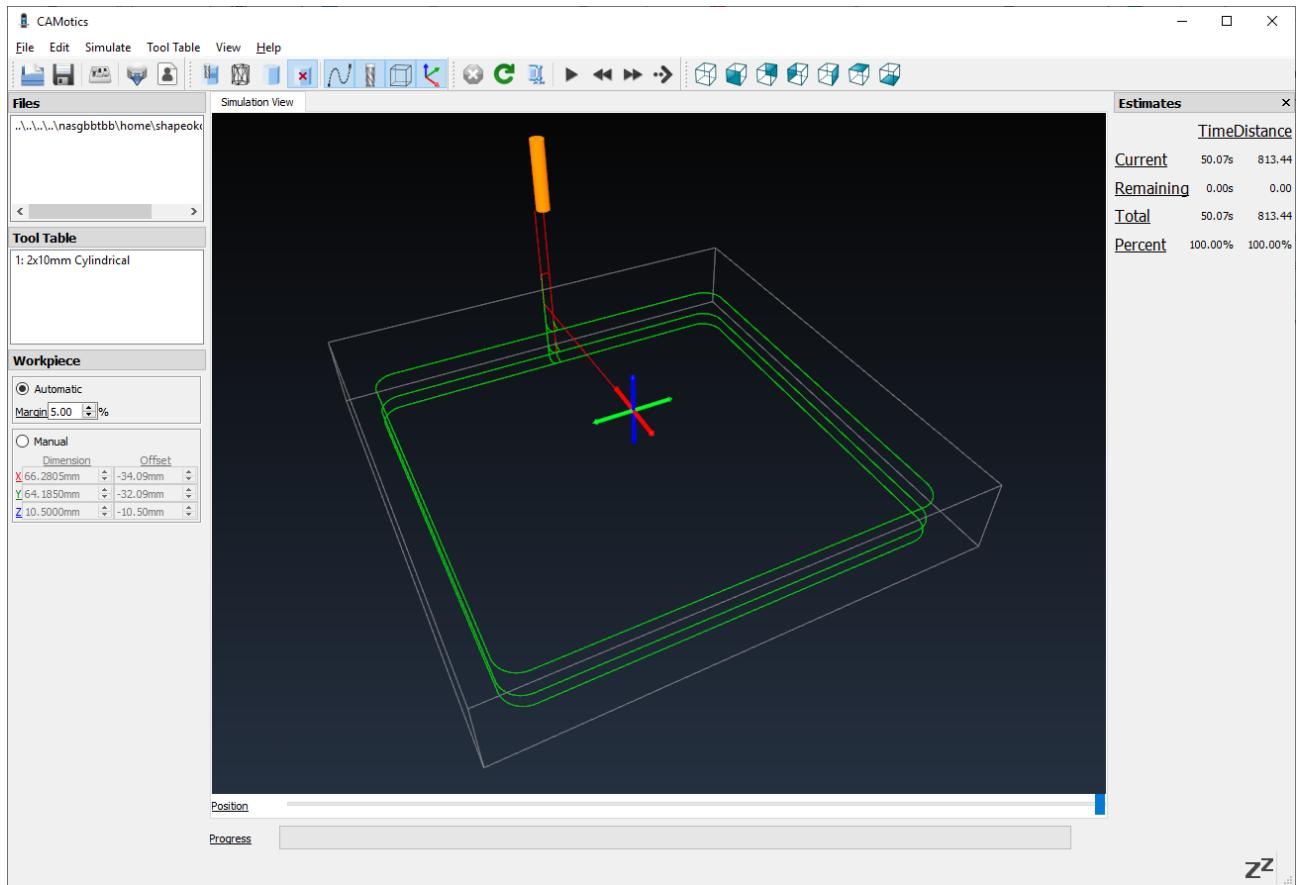
A useful alternative is to use specific drilling toolpaths, that just plunge the endmill vertically, so it becomes possible to do a 1/4" hole with a 1/4" endmill. However, an endmill is very bad at drilling, it is just not designed for this, so the plunge rate should be limited, and the "**peck-drilling**" option used: the tool will cut a small DOC, retract to clear out the chips, and then plunge again, repeatedly until the full depth has been cut.

A more efficient alternative is of course to use an actual drill bit instead of an endmill, but it implies an additional tool change. Also, it's important (for safety) to check that the drill bits are rated for the speed the spindle will spin them at.

Previewing toolpaths from G-code

Pretty much all CAM tools embed a toolpath visualization feature, however what they display is the toolpath defined in the design, and this is not *guaranteed* to be 100% what the machine will execute, because of the post-processor step: the generated G-code *might* be subtly different depending on the selected options.

One way to double-check is to visualize the toolpath from the generated G-code file itself. There are a number of offline and online G-code viewers (CAMotics is a popular open source desktop app, and there are many online G-code viewers too):



Toolpath ordering matters

It's easy to understand that the order in which the toolpaths are run (usually) matters, but also quite easy to overlook a wrong ordering when a project involves many toolpaths. On the Shapeoko, toolpaths using different tools will be in different G-code files (since there is no automatic tool changer), so the likelihood of manually executing the files in the wrong order is small. But multiple toolpaths using the same tool will usually be included into a single file, and the ordering will be as declared in the CAM tool, so a user error is more likely!

- Toolpath preview, and better yet realtime toolpath simulation (when available), is the best mitigation against this risk

Workholding

Workholding can be confusing at first because there are many (many) ways to do it, and everyone seems to have their own tricks. The goals are:

- obviously, that the stock material will not come off during the cut, which would both ruin the piece *and* be dangerous (flying objects, machine damage, fire...)
- to provide good rigidity of the machine and stock setup. If the stock or workholding bends/shifts slightly under cutting forces, cut quality will be poor, and dimensions will be off.
- to maximise clearance/access to the faces of the stock to be machined.
- that it is easy/convenient/quick enough to install and remove the material.

This section does not intend to cover them all, but to present the most popular ones for the Shapeoko. But first, a small detour to talk about wasteboards.

Wasteboard

Many projects require cutting all the way through the stock material, *e.g.* to cut the outer profile of the workpiece. Having a wasteboard placed under the stock material is very common for at least the following reasons:

- even on a properly calibrated machine, it is not easy to cut *exactly* down to stock bottom.
- it is often not desirable anyway: the quality of the cut on the bottom of the piece can be poor if not overcutting (*e.g.* small variations in depth/flatness would leave material here and there).
- mistakes in the CAM design/G-code, or mechanical issues, might cause the endmill to cut deeper than expected, it would be too bad to damage the machine baseboard.
- the stock must lay on a surface that is flat and square to the machine's Z axis, and the simplest way to achieve this is to use the machine itself to surface its own base (see [Squaring, surfacing, trammimg](#)), so it should be a replaceable part.

It is *possible* to use the Shapeoko's MDF baseboard itself as a wasteboard, but who wants to be replacing the baseboard of their machine? (considering it would mean disassembling it, reassembling with a fresh baseboard, and then squaring everything again). It is much more convenient to use a **supplementary** wasteboard bolted onto the baseboard.

The (only?) downside to using a supplementary wasteboard is that it reduces the maximum Z travel (i.e. the maximum possible height for a workpiece)

Wasteboard material is very often MDF (cheap & easy to procure), HDPE or PVC is another option (much more expensive, but it does not wear off / tear out as easily, and is immune to humidity variations).

The wasteboard needs to be held onto the machine's baseboard, in such a way that it can easily be removed, the usual way is to bolt it onto the baseboard one way or other.

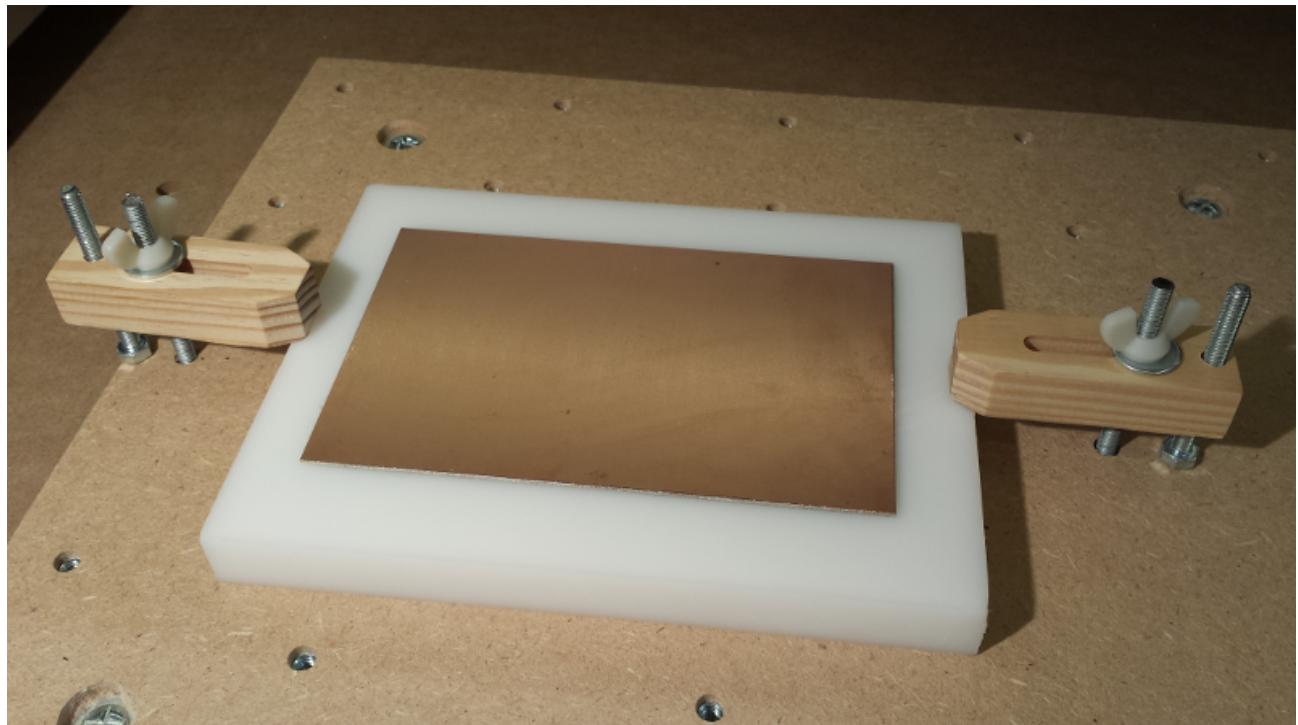
- ⓘ Make sure the wasteboard X/Y dimensions are such that the router will be able to reach the whole area, with some margin: this will be important for surfacing the wasteboard. The max dimensions vary depending on the machine (and accessories) setup, so actually moving the router in the four corner positions, measuring where the endmill ends up being, and making the wasteboard area slightly smaller than that, is an easy way to get this right.

Note that the wasteboard dimensions cannot match the advertised cutting area of your Shapeoko model, since some of that cutting area is the overhang at the front of the machine.

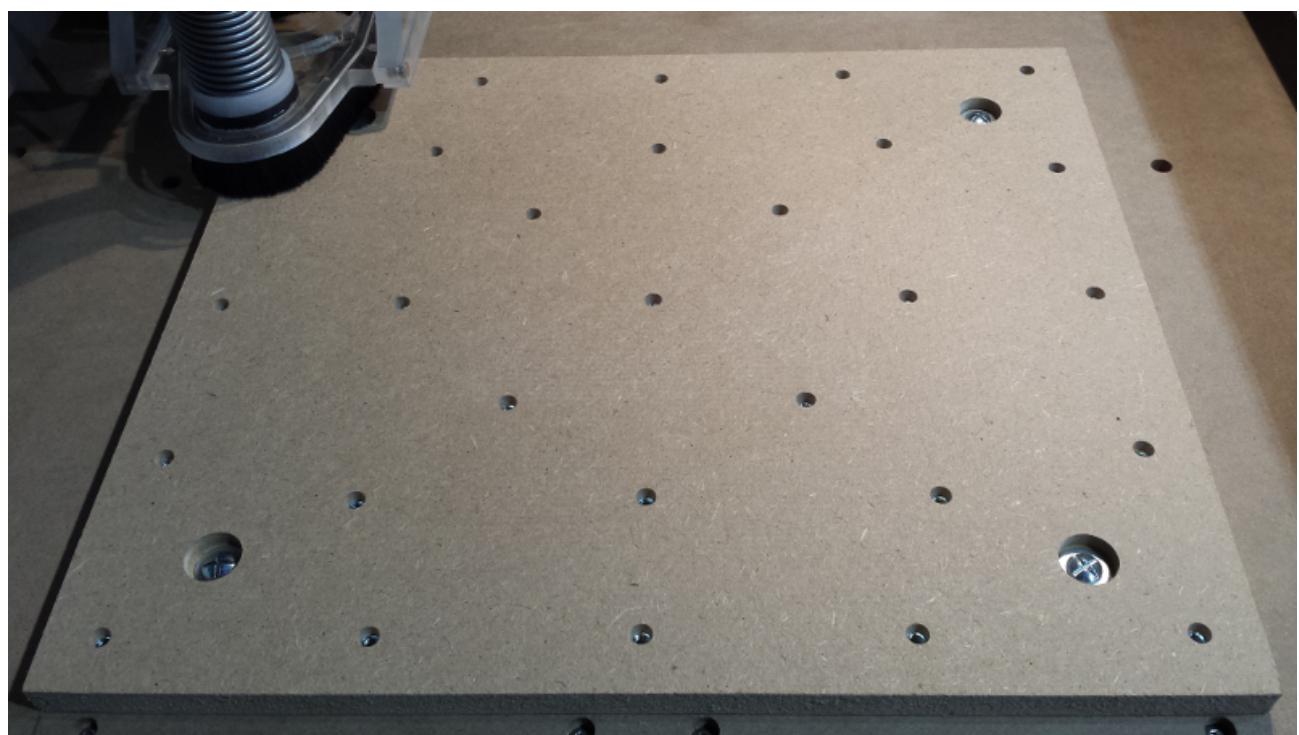
For the thickness of the wasteboard, make sure to include margin for the recessed bolt heads and enough material on top of that to allow for a few future resurfacing operations.

Threaded table, T-tracks, clamps

A very common way to hold the stock onto the wasteboard is to use top **clamps**, secured to the wasteboard using bolts:

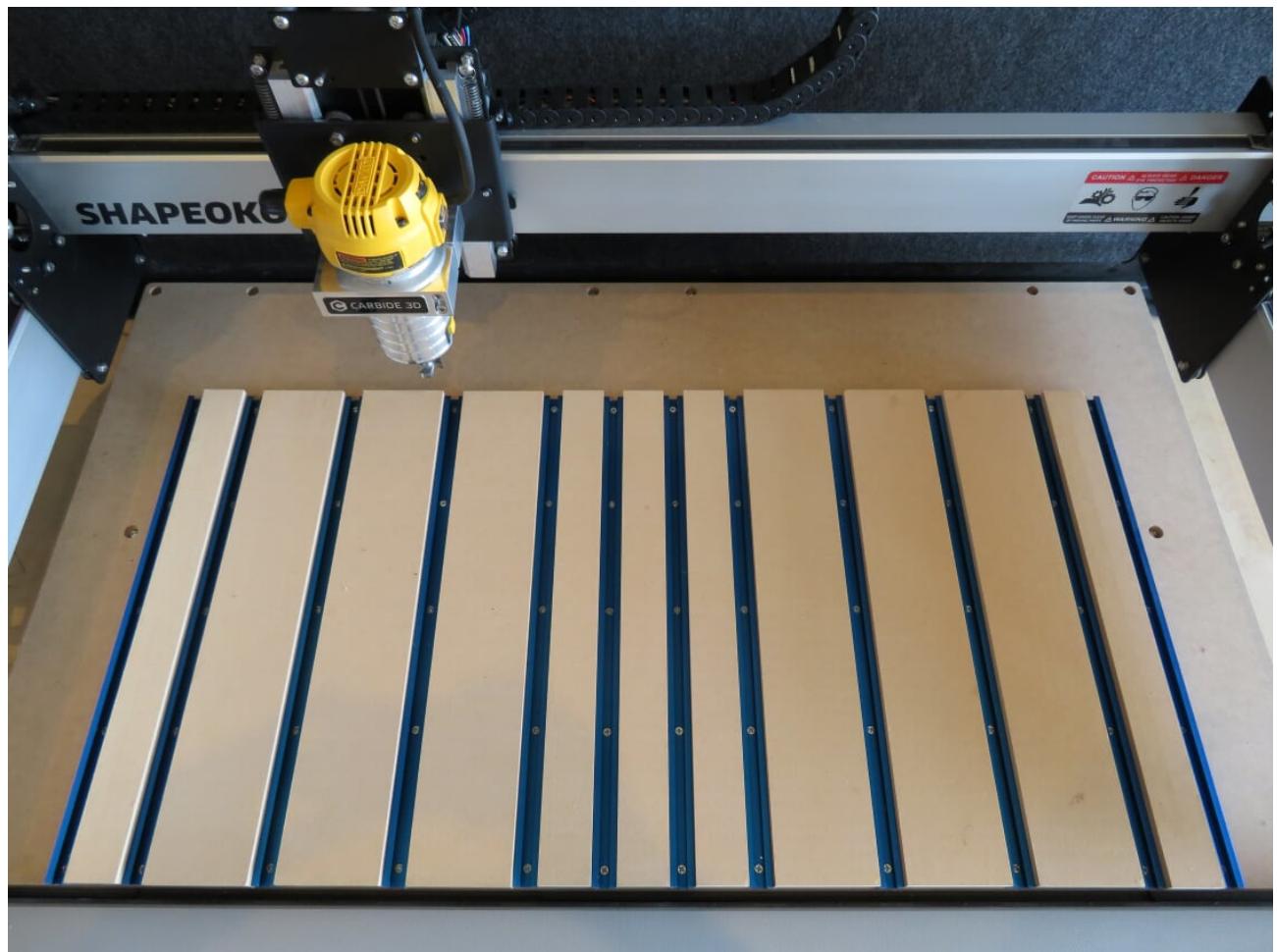


which requires to have threaded holes available in various places across the work area:



There are many different ways to make a "sea of holes" wasteboard. A popular option is to use the Shapeoko itself to drill the holes into either its own baseboard or the supplementary wasteboard, and then manually install **threaded inserts** in the holes.

Another very popular option to attach clamps is to use **T-tracks** inserted at regular intervals between "strips" of wasteboard area:

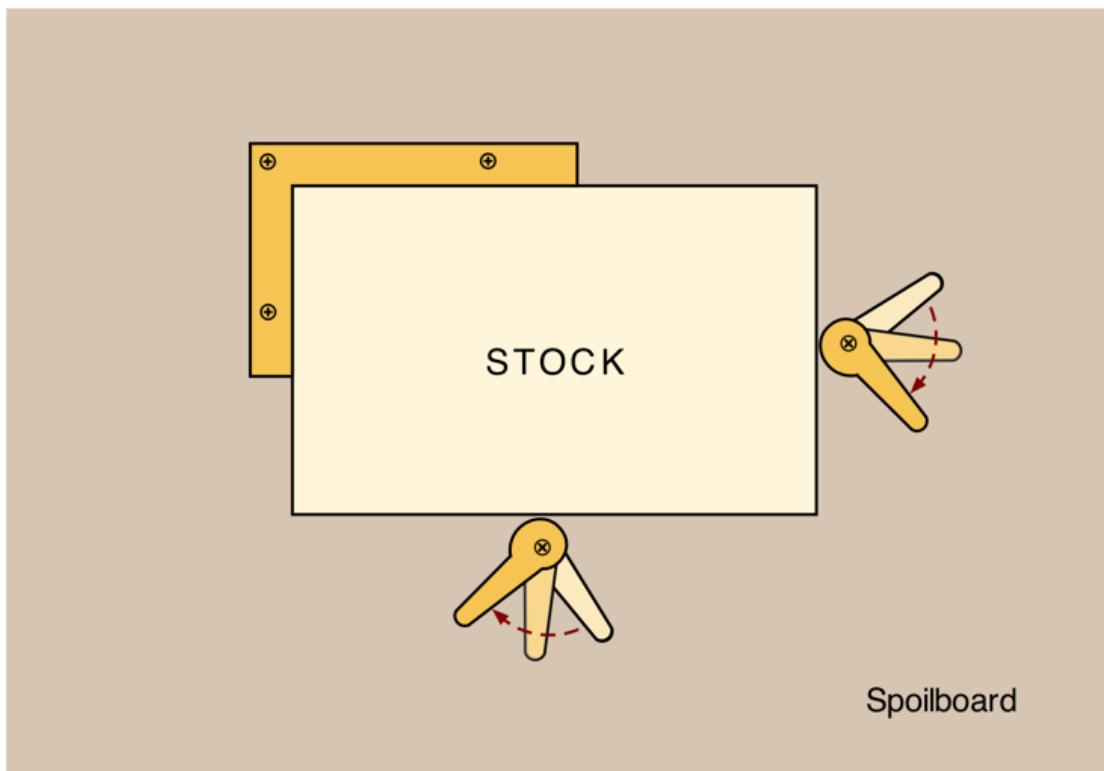
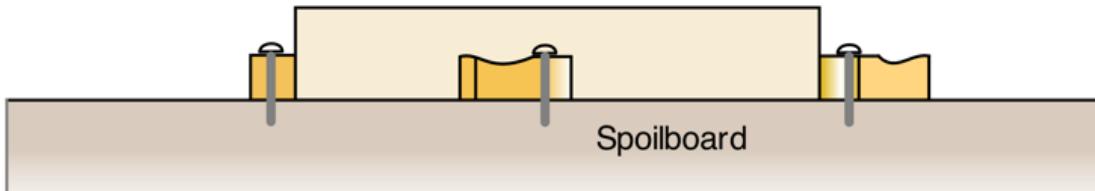


The main drawback in both cases is that the area where the clamps are holding the stock is not accessible to the cutter, and one should carefully design toolpaths such that there is no risk of collision between the cutter (or the dust shoe around it) and the clamps. The usual mitigations are:

- using a stock larger than strictly necessary for the workpiece, and machine only the center area of the stock. Easy, but not very efficient in terms of how much material is needed/wasted.
- using **low-profile clamps**: this addresses the issue of collision with the dust shoe itself, as the clamps can slide under the bristles of the dust shoe. The risk of collision with

the tool itself is still to be managed though.

- using side clamps/**eccentric clamps** as illustrated below: they push the stock from the side into a corner block, and free up the top surface of the stock completely.



Tabs, Onion skin

Since clamps only hold the stock by its sides, contour/profile cuts leave a middle piece that gets separated from the main stock after the last pass, and this needs to be managed too: at best it can damage the piece (it may move and bounce right into the endmill), and at worse it can be dangerous if the piece flies away in a random direction.

One solution is to use **tabs** in the design, most CAM tools support this:



The tabs will hold the piece during the last passes of the cut, however they will have to be removed/cleaned-up manually afterwards, which can turn out to be time consuming and may leave clean-up marks on the workpiece.

In some materials (especially soft plastics like HDPE), an alternative option is to leave a thin **onion skin** at the bottom of the profile cut, by limiting the cutting depth to something slightly less than the stock thickness. The onion skin should be thick enough to keep the piece from moving, but thin enough to be easily cut manually afterwards with e.g. a X-acto blade. The bottoms edges of the cut still need to be cleaned-up manually, but in soft plastics this can be very easily done with a deburring tool.

Double-sided tape

Since clamps get in the way of the cut, and tabs require clean-up work, the other approach is to use double-sided tape under the stock to secure it onto the wasteboard.

The main issue with this method is finding the right balance between putting enough to hold the stock firmly, and not putting so much as to still be able to remove the piece easily after the cut.

While it can certainly be a good solution for some usecases, the tape & glue method described below has all the same advantages, but provides better rigidity and requires much less force to remove the piece, so read on !

- (i) Double-sided tape can be useful in combination with other methods: for example, using clamps to hold the stock but putting a piece of double-sided tape underneath the areas that will be cut free, can be a good way to avoid using tabs.

Tape & glue

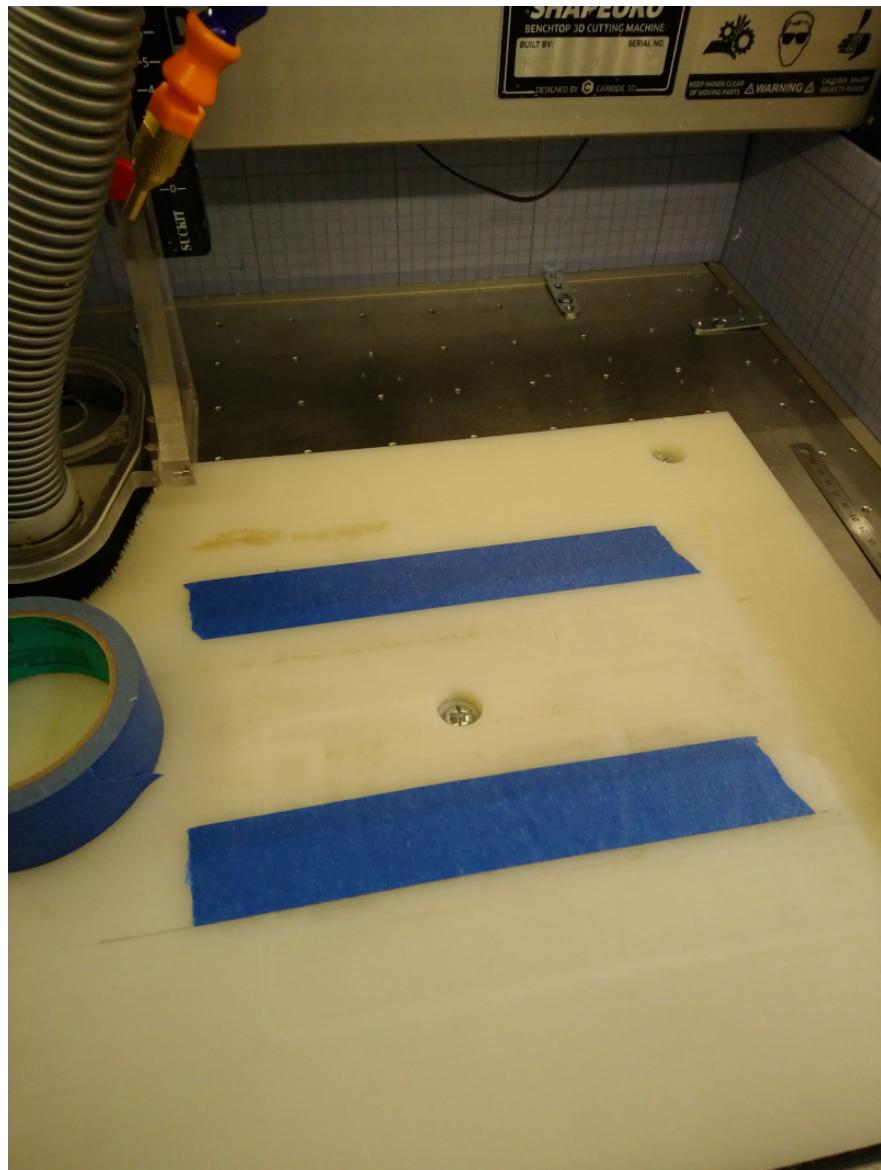
Another very popular workholding method is surprisingly simple, and surprisingly efficient. It uses painter's masking tape, and cyanoacrylate (CA) glue:



Apply masking tape on the back side of the stock, using several stripes if needed to have a large area covered. I use the roller shown in the picture above to make sure the tape is pushed firmly against the stock and everything is nice and flat (make sure there are no wrinkles/bubbles):



Apply masking tape on the top of the wasteboard where the stock will go, to match the tape under the stock. Again, the roller is useful to make things stick.



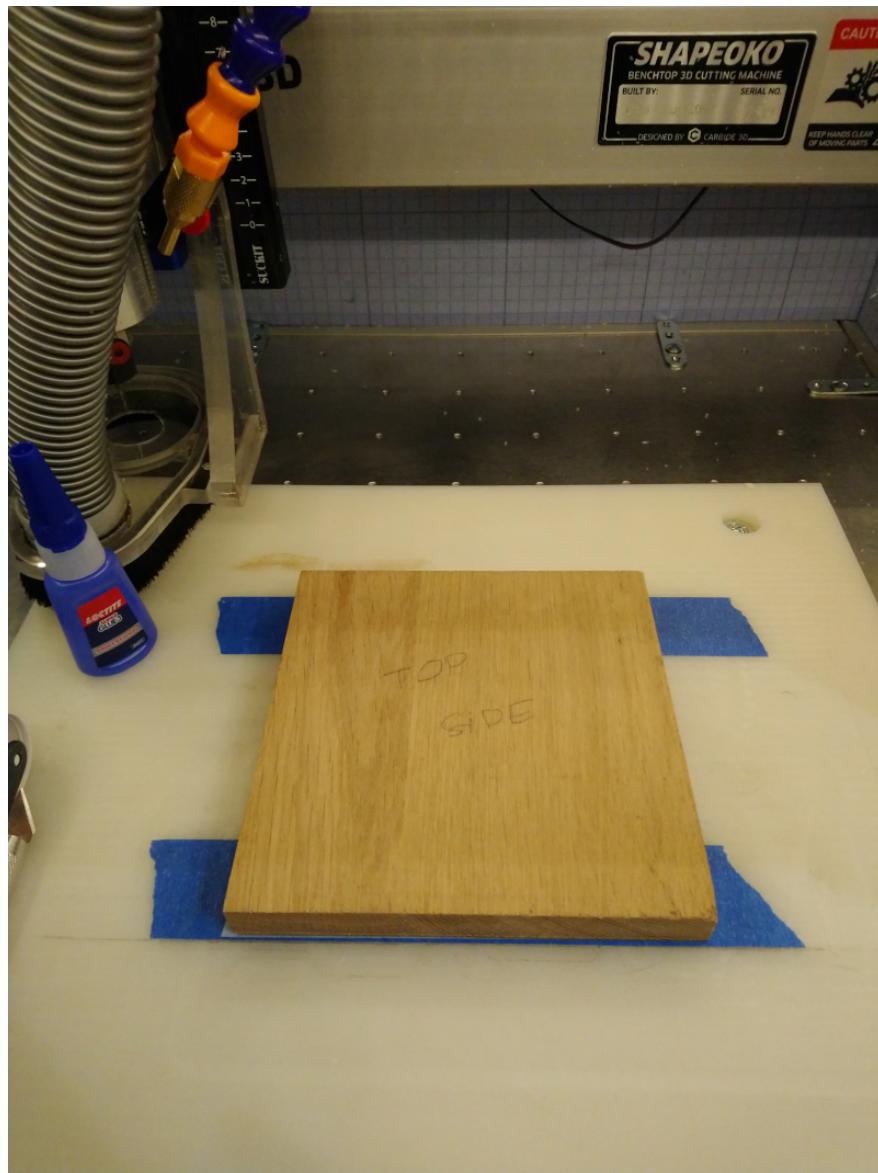
Now apply a zig-zag of CA glue on the tape under the stock,



Flip it, quickly position it, then push down firmly across the whole surface for a few seconds (some use glue accelerator, but this is typically not necessary):



You may want to cut the extra pieces of tape, just to prevent any possible mishap where they would get in the way of the cutter (think about the case of doing a final contour at full depth around this stock...)



At this point, the stock should be held very tightly: apply lateral force and make sure it does not budge. Push harder if you need to convince yourself that this will hold. If done correctly, the amount of *lateral* force needed to break the hold is **HUGE**, way beyond what the machine can produce. Enjoy cutting the piece without any obstruction of the top and sides of the stock ! After getting a taste of this method, you might not want to go back to using clamps ever again.

Once the cut is done, insert a flat tool under the piece to exert an **upward** force:



It may take a few (gentle) tries working on each corner/edge in turn to pry the piece loose, but it usually comes off very easily.

The main drawback of this method is that for all cuts that go through the full depth of the stock, during the last passes the tool will cut into the tape and glue, which is bound to leave gummy residue on the tool. This is very easily removed with a little bit of alcohol/acetone.

- (!) However, this may turn out to be a real problem for jobs that do multiple profile cuts: if the first profile cut leaves some tape & glue residue on the tip of the endmill, it could impact the quality of the subsequent cuts. There is no real solution for this, the only partial mitigation is to cut exactly down to stock

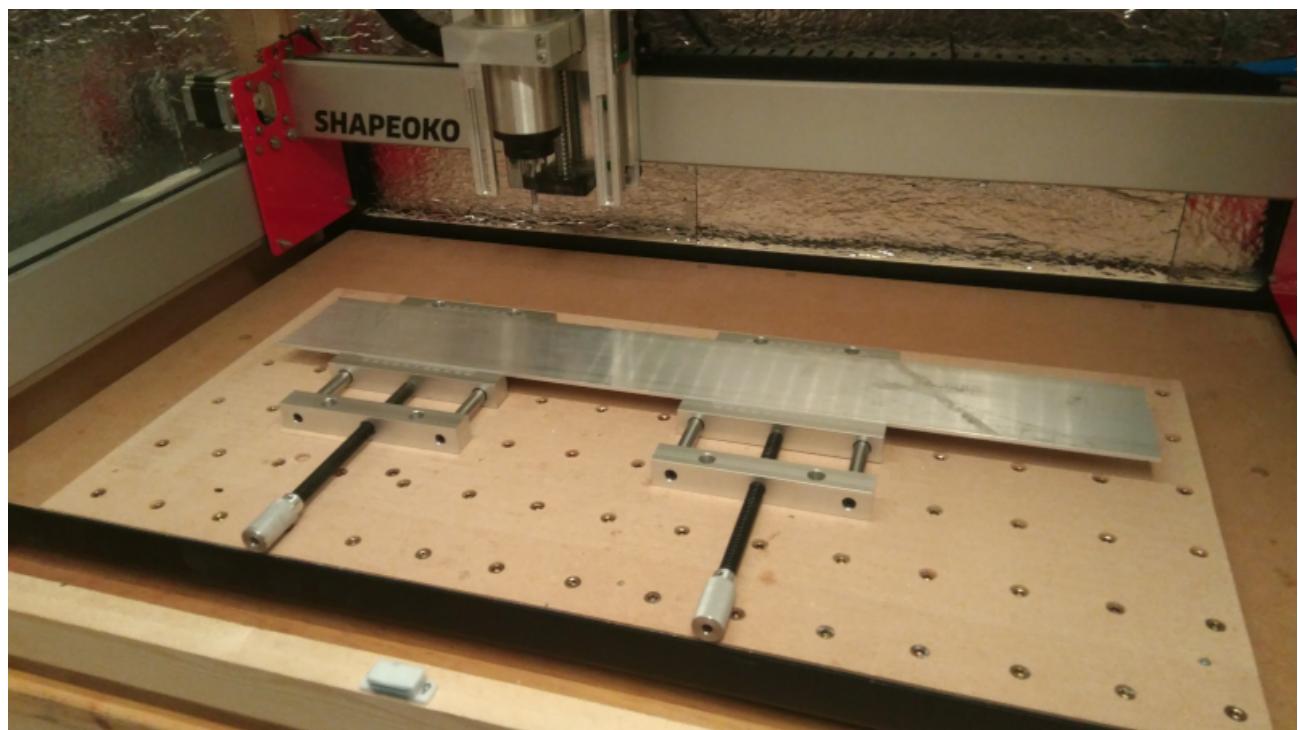
bottom to avoid cutting into the tape, but that brings back other issues if the stock bottom is not perfectly flat or the machine not perfectly trammed.

If cutting thin sheets of soft material (e.g brass, copper) the force required to pry the piece free of the wasteboard would likely bend/damage it, so it may be necessary to apply heat to soften the adhesives.

Vise

A vise is typically used when cutting high-precision metal parts, since it provides excellent rigidity. It's also interesting when productivity and/or repeatability over multiple runs of the same job matter: the vise provides a good reference point to align the stock consistently, and installation/removal is very quick. Or more simply, whenever there is a need to hold a piece that cannot sit flat on the wasteboard (e.g. anything round)

The max width of the vise jaw is the main limitation, and its height may be a concern, especially on a machine like the Shapeoko that has a somewhat limited Z travel. A low-profile vise, that can be installed and removed from the machine depending on the job at hand, is a good option to have:



Shapeoko setup

This section is a brief overview of the usual elements of a Shapeoko setup in the workshop, and focuses on the "why" and "what's important" in each element, rather than describing all possible options (which would be impossible, user setups vary wildly!)

Location

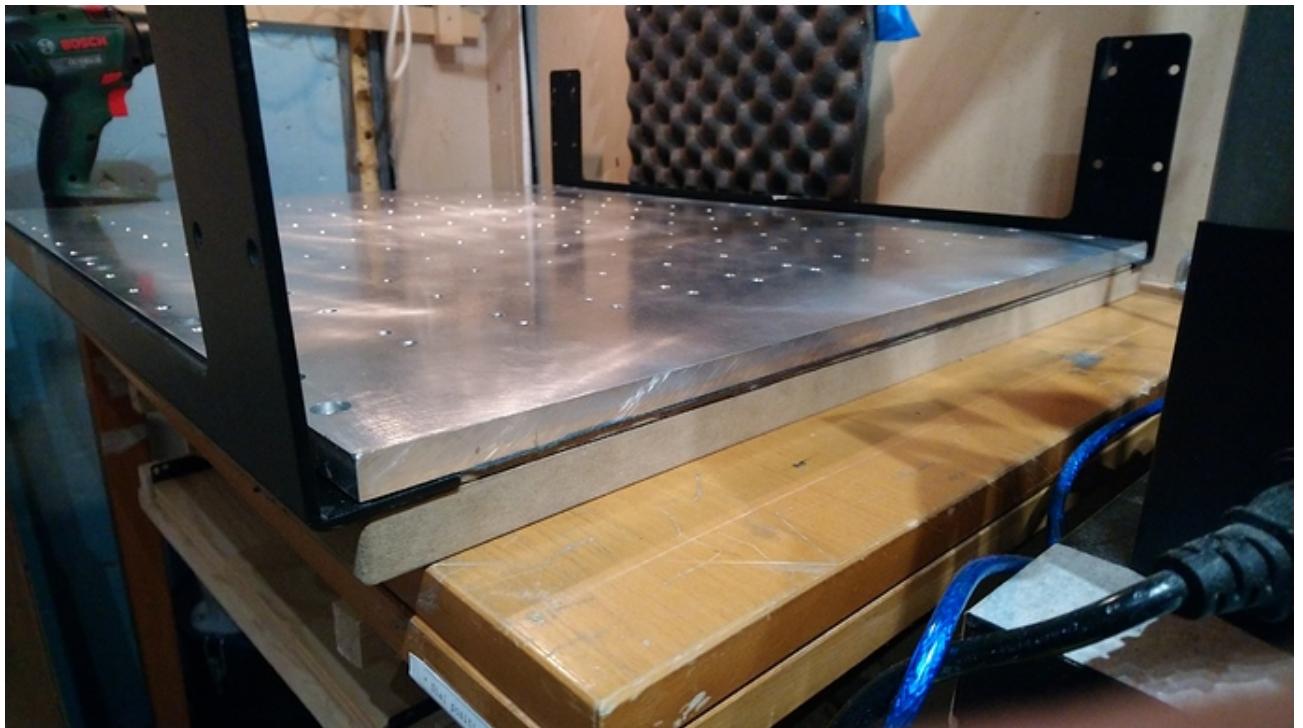
Many users do not have the luxury of having multiple options for where to install their Shapeoko anyway, but when possible:

- a heated/dry area is preferable. The structure and electronics are basically immune to low/high temps and humidity, but the MDF bed is prone to absorb moisture.
- spare space around the machine (especially above the machine) comes in handy
 - for accessibility during maintenance
 - to accomodate an enclosure, itself larger than the machine by a good margin
 - as a provision for upgrades (e.g. higher Z-axis upgrade)
 - to have room to fit a dust collection system underneath/next to the machine
 - to support tiling (feeding long pieces from the front or back of the machine, through to the other side).
- height: having the machine installed at arm/desk level is best. Kneeling to change the endmills on a machine installed on the floor or in a low cabinet will get old very quickly.
- keep in mind that CNC is noisy (router and sound of the cut itself and dust collection system), though an enclosure will help a lot.
- the Shapeoko should rest on a rigid and level surface. A custom-made bench with a torsion box is a popular choice, but mine is just installed on a sturdy Ikea kitchen table bolted to the wall.

Feet or no feet?

The regular setup with the four leveling feet works fine, but is susceptible to a couple of issues:

- the MDF bed can sag in the middle, especially on larger models (XL and XXL). Usually not by much, but enough to be a nuisance. Just adding a custom support point in the middle underneath the bed/rail addresses this weakness, easy enough.
- the other point that made me consider removing the feet is that I am using the tape & glue workholding method a lot, it involves pushing the stock firmly onto the wasteboard for a few seconds, and it did not feel right to be pushing in the middle of the bed, exactly where the natural sag already happens. As many others have done, I removed the four feet. My solution was to install the Shapeoko on top of a thick piece of MDF (with holes to accommodate the protruding nuts under the steel plates), with a narrower and thinner piece of MDF inserted between the plates on top of that, plus a thin sheet of roofing felt to provide a little damping. As long as the bench underneath is reasonably level, getting rid of the feet is not a problem, and this will provide a lot of additional rigidity: I can now push on the bed as hard as I need to.



(the picture shows an aluminium bed, which also helps the rigidity, but that is a story for the [HW upgrades](#) section)

Dust collection

CNC is just messy. While it is quite possible to operate the Shapeoko without a dust collection system and just clean-up manually once the job is finished, here are a few reasons why it is much better to have one:

- cutting MDF: unlike in wood, the cutter will not produce nice thick/heavy chips, but a very fine dust that will float and soon cover everything around the machine, and is dangerous if inhaled repeatedly.
- visibility: vacuuming chips during the cut allows seeing what the tool is doing, and how the cut looks like so far, which is important to detect if something is not right. And manual vacuuming during the cut gets old very quickly.
- clearing the path for the tool: if chips accumulate *e.g.* in a deep slot, they will end up being in the way between the tool and the material, they will accumulate and reduce chip evacuation inside the flutes, all of which are not good for the quality of the cut.
- some folks have caused cuts to fail by dust induced static when vacuuming by hand.

Dust shoe

Using a dust shoe attached to the X/Z carriage is the common solution. They come in two main types:

- the dust shoe can be attached to the router (or elsewhere on the moving part of the Z-plate). In this case, since it will move up and down with Z movements during the cut, one needs to take that into account, by leaving enough clearance under the dust shoe and ideally having long and flexible bristles, so that when the router will be cutting at maximum depth, the dust shoe won't collide with the surface of the stock. Here's a picture of Carbide3D's "Sweepy" dust shoe:



i The main benefit of a router-mounted dust shoe is that it can be quite compact, and it will not reduce the X travel. The main drawback is that it does not work well when cutting deep jobs, as the required clearance under the bristles will make the suction efficiency drop significantly

- or the dust shoe can be attached to the fixed part of the carriage, and then adjusted and locked with the bristles against the stock surface (you will still have to watch out for possible collision with clamps, or use low-profile clamps that will slide right under the bristles). They are called "Z-independent" dust shoes, here's a picture of such a dust shoe installed on my machine:



- ① The main benefit of a Z-independent dust shoe is that suction is optimal (since the bristles can be adjusted to be flush against the stock surface), and there is no need to worry about cutting depths. The main drawback is that the side arms holding it usually reduce the X travel slightly, and the dust shoe is generally a bit bulky, with risks of collision with clamps.

- ① If you buy or make a dust shoe, make sure it has a slot in the back so that it can be inserted/removed without having to raise the tool, unlike mine shown in the picture above. The natural thing to do is to zero without the dust shoe to see

what you are doing (even more so if you are using a touch probe), and then slide the dust shoe in place just before running the job.

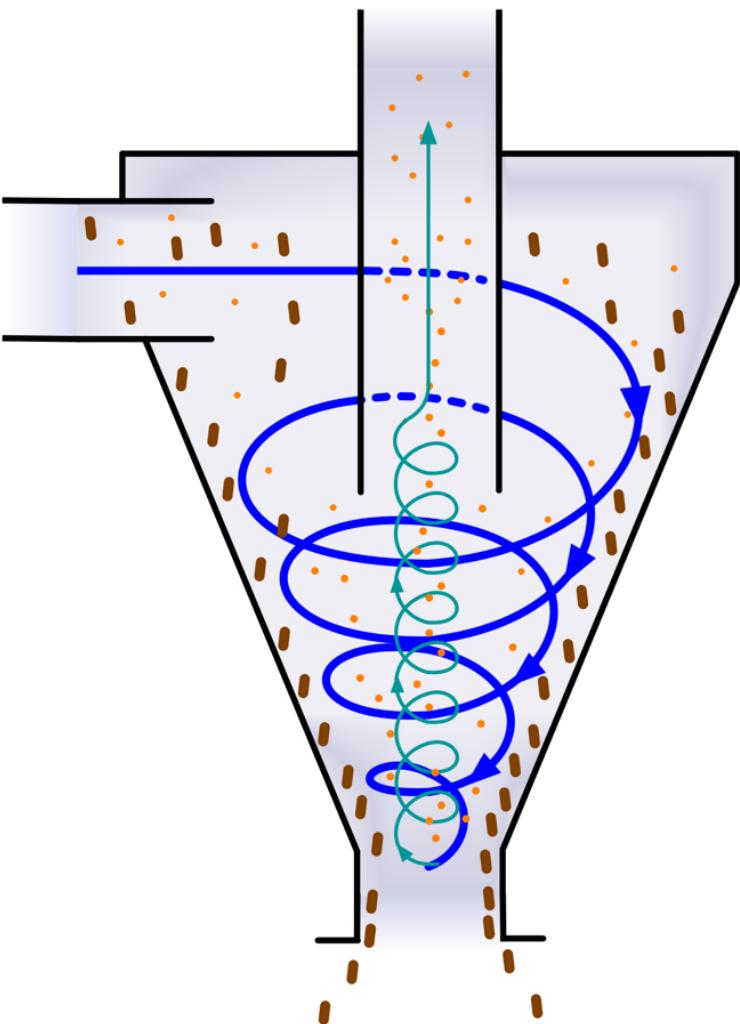
- ⓘ When the toolpaths are such that the dust shoe will move past the edges of the stock surface, it is useful to add extra material (of the same thickness) around the stock, to ensure that suction power remains optimal throughout the job.

Dust separator & shop vacuum

While it is possible to connect a shop vacuum directly to the dust shoe, it turns out to be inconvenient, and potentially unsafe:

- inconvenient because the amount of chips produced when cutting with a CNC can be significant, and you would end up replacing the shop vacuum paper bag very often.
- potentially unsafe because cutting some materials (*e.g.* MDF) produce fine dust instead of chips. Even with perfect feeds and speeds, the shop vacuum may not have an adequate filter to cope with this, and even if it does it would require cleaning very often.

A common solution is to buy or build a "cyclone" dust separator:



- the shop vacuum is plugged on top, while the hose from the dust shoe is connected on the left.
- the suction from the shop vacuum and the shape of the cyclone dust separator are such that an outer vortex (blue) and inner vortex (teal) are created, so the chips/dust contained in the incoming air flow are pushed against the walls of the cyclone by centrifugal force, and then slide down under the effect of gravity to the bottom where they can be collected using any kind of tank.

I chose to buy a cheap ~4 gallons/15L ash collector tank at my local hardware store, and bolted the cyclone onto that (with a little glue around the base for sealing):



Surprisingly little dust/debris gets past this and into the shop vac, so much so that I hardly ever change the shop vacuum filter itself now

- ⚠ BUT if you are cutting a lot of MDF or other nasty material that produce very fine dust, you should still have proper (HEPA) filtering in place, the cyclone alone will not be good enough.

Enclosures

Putting the Shapeoko inside an enclosure has many benefits:

- noise reduction
 - the enclosure walls alone will reduce noise, or at least filter out the most irritating high frequencies of the router.
 - the inside walls can be padded with a sound-damping material.
- dust/chips containment
 - while most of the job will be done by the dust collection system, *some* chips/dust will still find their way around the machine, and the enclosure helps in keeping that from spreading to the rest of the workshop.
- additional safety
 - the front window of an enclosure is an excellent protection against flying debris, or the occasional broken endmill.

Most people design their own custom enclosure. Here are a few things to consider:

- **height:**
 - it should be high enough to accommodate the Shapeoko itself *and* still have a comfortable margin on top on that. The dust collection hose should be able to move freely when the gantry moves to any position on the work area.
- **accessibility**
 - a front window that can be removed or opened is standard. For maintenance reasons the ability to remove/raise the top of the enclosure out of the way is handy.
- **visibility**
 - the front side should be a window, you will want to be able to check what is going on at all times.
- **integration**
 - with the dust collection/shop vacuum, which is commonly placed underneath the enclosure.
- **clearance** at the front of the machine:
 - the router can reach beyond the limits of the bed/wasteboard on the front side, and this can turn out to be very useful (stock overhanging on the front), so the enclosure should not prevent this.
- **sound-proofing**
 - covering the side walls with a sound-absorbing material helps dampening the noise.

- **lighting**

- you will probably want to install (LED) lighting inside the enclosure, at 360° if possible to avoid shadows on the work area.

- **future-proofing:**

- provision areas for installing various controls (see control panel below), a VFD controller when you upgrade to a spindle, etc...
-

Control panel / E-stop

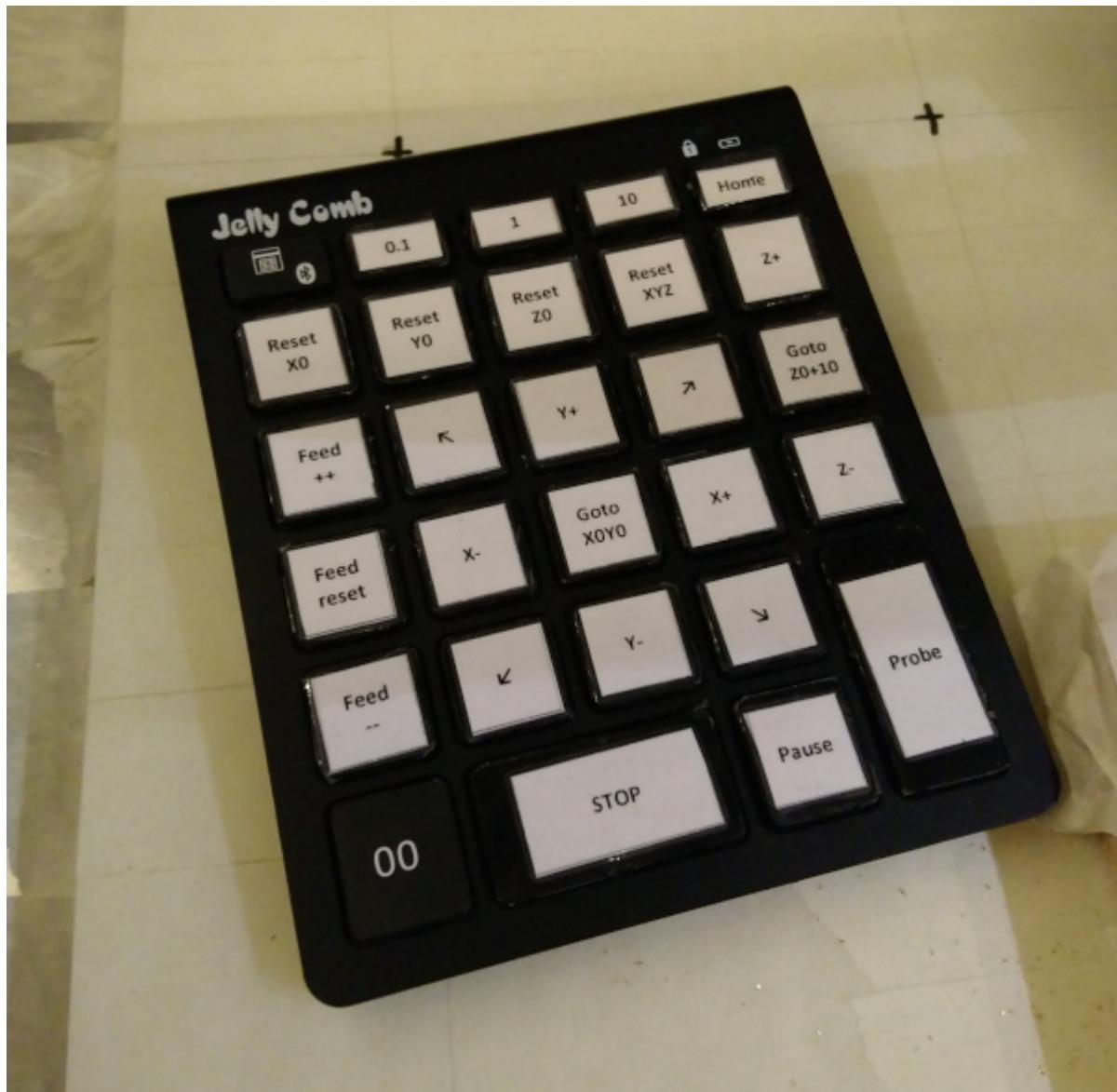
While you are designing your enclosure, why not plan to add a custom power control panel at the same time? It makes for a simple and useful project, especially if you include a kill switch/emergency stop button. When things go wrong, you do not want to have to reach for the various power switches (router and machine, at least), so having a big red button located right within arm's reach could save you (or the machine) someday.

Mine is a crude version made from MDF parts, it does not look fancy but does the job perfectly. Beyond implementing the emergency stop button, having all power switches in a single place and with a visual cue as to what is currently turned on is very convenient:



Control pad

If you are using a G-code sender that supports keyboard shortcuts, it can be convenient to use a remote keypad (wireless or wired) for those shortcuts, so that the jogging/probing commands can be used without looking at the computer screen. Mine is wireless and costs about 20\$, I printed custom labels for the actions I use most, and then mapped the underlying keys to the associated actions in my G-code sender.



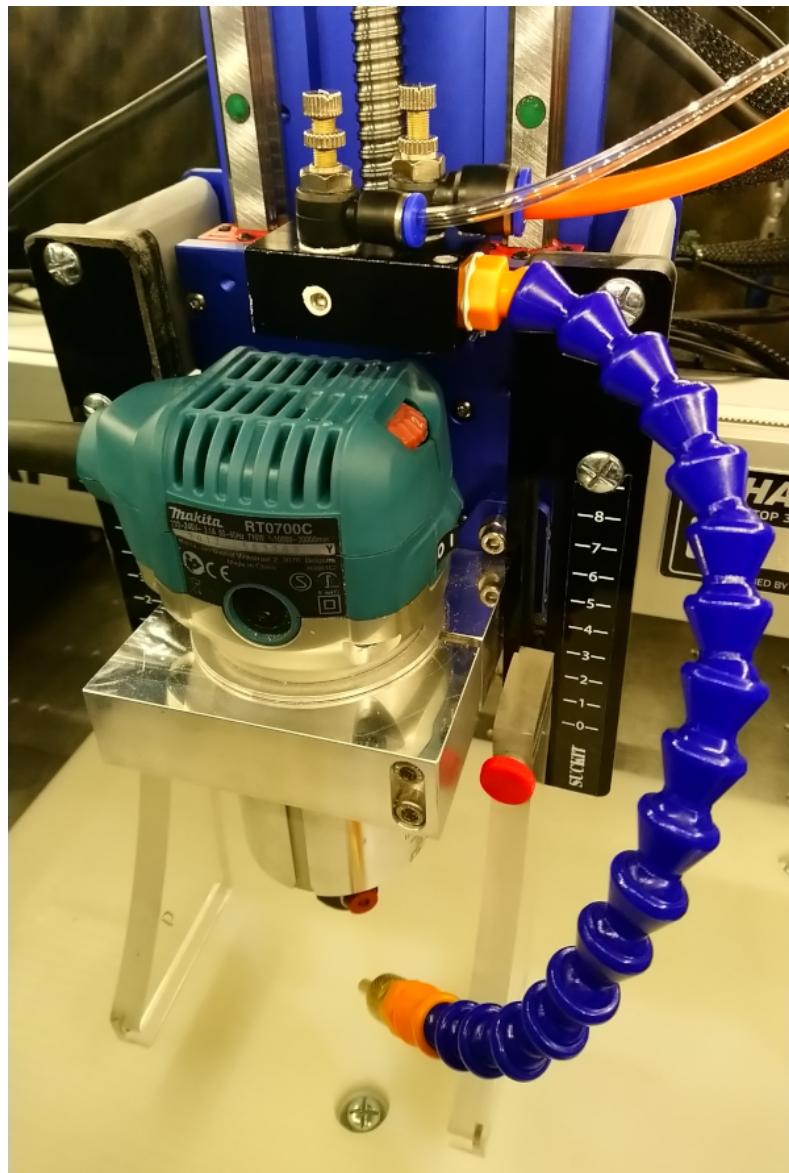
You can also go crazy and buy a fancy keypad with programmable OLED displays on each key, and dynamically switch between different keymap configurations.

Air jet & lubrication

While a dust collection system works great for cutting wood & plastics, cutting **metal** is different: the chips may be too heavy to be efficiently sucked out of the cut, and if any lubrication is required it is incompatible with the use of a dust shoe.

The usual solution is to use an air jet to push the chips away from the cut. A mist coolant spray system is a useful addition to the shapeoko setup when cutting metal. In the picture

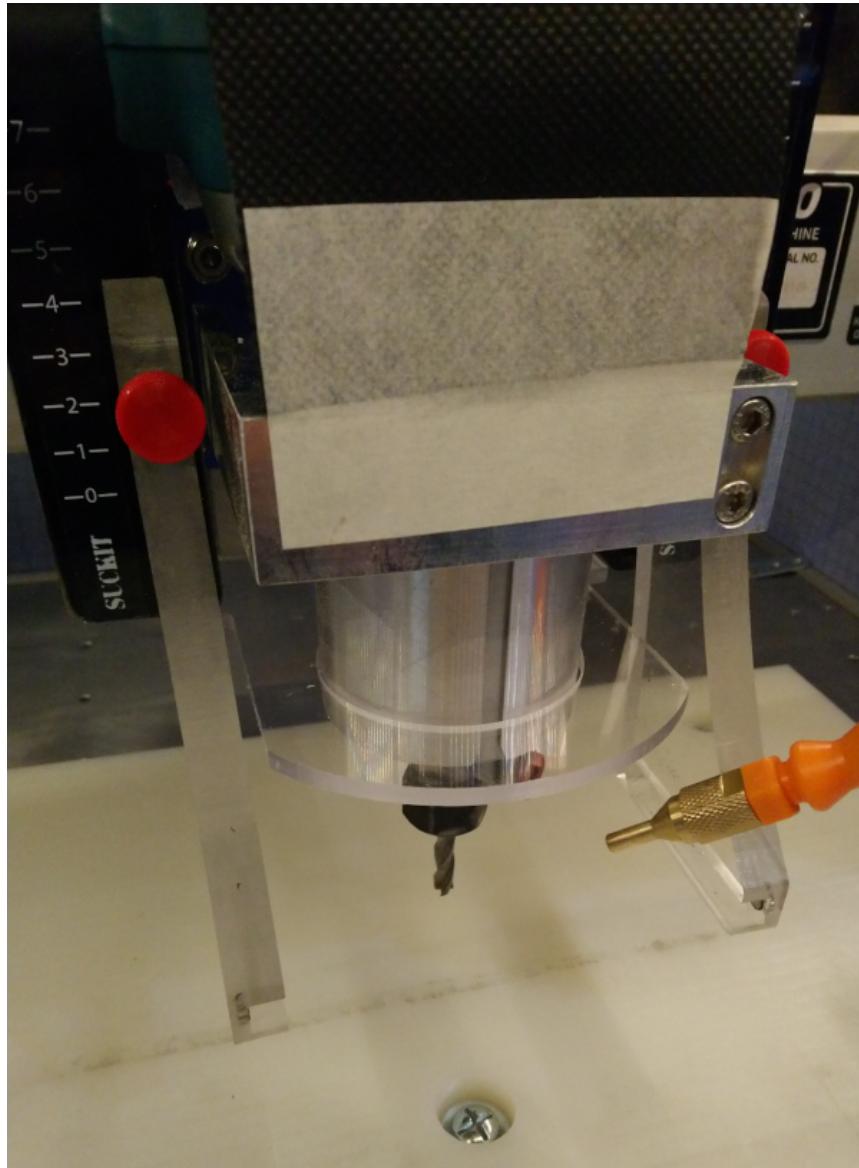
below, the orange tube goes to a (heavy duty) air compressor, and the transparent tube goes into a bottle of lubrication liquid. The lubricant gets mixed with the compressed air, and the spray is aimed at the cutting point:



However this means that metal chips will be flying all over the work area, so it can be useful to add specific protections.

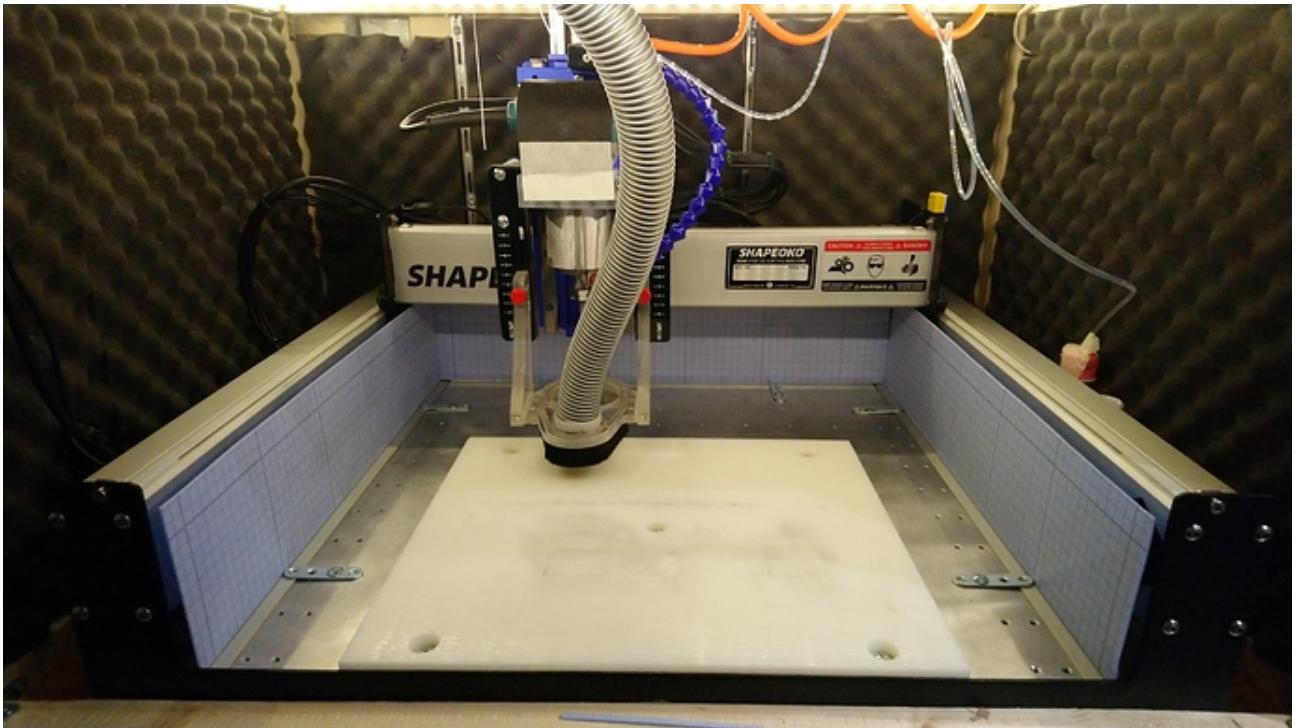
Some kind of **filter** on top of the router: the picture below shows a quick and dirty hack that protects the router air inlets from flying chips, while still letting the air enter freely. Some just put a sock on the top part of the router. Whatever works for you, but do protect it somehow, metal chips going into the router's air intake will not end well.

A **chip guard** can be installed to prevent chips from flying upwards and getting stuck in the Z-axis or router, while still allowing the air jet to be aimed at the cut:



Side walls

Protection walls can be installed on the sides/back of the machine, to at least keep the chips contained within the work area, when not using a dust shoe:



Mine are just made from 0.1" thick hard foam strips glued to aluminium corner guards, bolted onto the bed.

- (i) With an MDF bed, another idea is to cut slots on the sides, making it very easy to insert/remove the walls.

They do a good job of containing the chips (but if you look closely enough at the picture below, you will see that some chips ended up on the left rail anyway) :



GRBL settings

The Shapeoko's default configuration procedure in Carbide Motion programs a set of specific GRBL parameters in the controller. Depending on the version of CM used, these settings were meant to be conservative, they are ideal while learning how to use the machine, but they can then be tuned to more aggressive values to improve jogging and homing speed.

I tuned mine based on a great [suggestion](#) from **@wmoy** on the forum, and captured them here for reference:

- 1 \$25=2000.000 (Homing search seek rate, mm/min)
- 2 \$27=1.000 (Homing switch pull-off distance, millimeters)
- 3 \$110=10000.000 (X-axis maximum rate, mm/min)
- 4 \$111=10000.000 (Y-axis maximum rate, mm/min)

```
5 $112=1400.000 (Z-axis maximum rate for HDZ, mm/min, 1000 for stock Z axis)
6 $120=500.000 (X-axis acceleration, mm/sec^2)
7 $121=500.000 (Y-axis acceleration, mm/sec^2)
8 $122=200.000 (Z-axis acceleration, mm/sec^2)
```

These values can be tuned from the MDI menu in Carbide Motion, or from the G-code command shell in any other G-code sender.

 Those GRBL settings are now the standard values in Carbide Motion, starting from version 505

 Note that those increased MAX speeds are fine for rapid moves outside the material, but it will still be your responsibility to program adequate cutting feedrates in the toolpaths, which are usually much lower than those limits. That 10,000 mm/min rate on X and Y corresponds to 393ipm, while cutting feedrates are typically always (well) below 200ipm.

Running a job

The basic workflow for running a job is covered pretty well in Carbide 3D's docs/tutorials, the intent of this section is to provide background information and various tips about the different steps.

-  The ordering of these steps can vary based on personal preference and software being used. For example, I like to have the machine turned off while I change the cutter, but many find it more convenient to have it turned on to be able to jog the router automatically to the front.

Installing the cutter

A few things to watch out for when installing a tool in the collet :

- make sure the collet matches the endmill diameter. This may sound silly, but once you start having a collection of various imperial and metric tools and collets, inadvertently using a 6mm endmill in a 1/4" (6.35mm) collet is a (remote) possibility, and not paying attention to this will result in serious trouble during the cut.
- the collet and collet taper must be **clean** of any debris/dust. Those might introduce runout.
- **stickout:** it should be
 - as low as possible to minimize tool deflection
 - but still be compatible with the max cutting depth for this job
 - short enough that the shank is engaged over the full height of the collet
 - but not pushed so far back that the flutes are inside the collet
- **runout:** if it matters for the job at hand (high-precision and/or micro-machining), now is the time to check it and tune it.
- the saying goes that the collet should be "**monkey tight, but not gorilla tight**". It should not be a baby monkey either, or the endmill might slip in the collet.

Moving manually

Whenever the machine is turned off, it is possible to move it manually, with just one constraint: do it slowly. When the machine is turned off and moved manually, the stepper motors behave as alternators, they generate current, enough current in fact to back-power the electronics (you might see the LED on the controller board come to life...). Moving slowly ensures that the generated current stays low enough to not damage the stepper electronics.

Homing

- (i) That "clonk" sound when the machine is turned on is just the stepper motors locking in place, nothing to be concerned about.

The reason for homing was introduced in the [CNC workflow](#) section: the only way the Shapeoko can tell for sure where it is, is when it contacts the three limit switches. Without homing, each time the machine is power cycled it would be unable to go back to any specific coordinates with precision.

You could argue that homing is useless since you are going to manually jog to the Zero point and set it to be the reference anyway. For a job that can be done in a single run / with a single tool, that could work. But the power of homing is that it will allow returning with great precision to the Zero point defined last (which happens to be stored in the non-volatile memory of the controller by the GRBL software).

If you turn off the machine to change tools, this will be needed. But even if you don't, you may still want to home between runs: if for some reason the steppers or belts skipped a few steps during the previous run, returning to Zero without homing first *may* bring the machine to a subtly different point, a few steps away.

-  The notable exception is 2-sided jobs: depending on the precision of your limit switches, you may find that it is better to NOT re-home between the two sides to have minimal offset error.

Setting & checking RPM

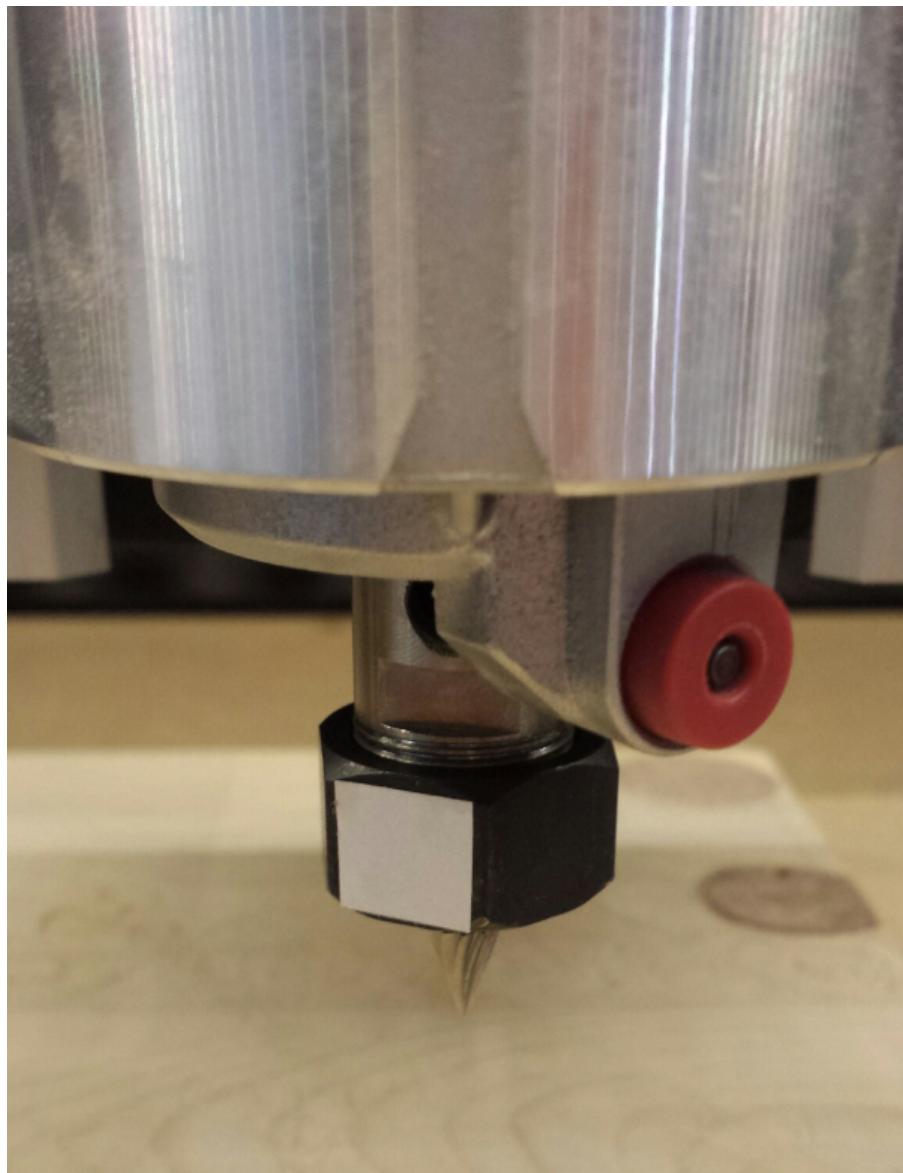
Unless you have installed a spindle or a PID controller (see [HW upgrades](#)), you need to adjust RPM manually using the knob on the trim router. The [Anatomy of a Shapeoko](#) section has a reminder about the mapping between knob values and RPMs, but the actual RPM can be slightly different than advertised, and on my router the knob did not even have a reference point on the casing, so I added a visual cue with a marker to at least have a repeatable setting:



But it's not easy to interpolate between knob settings to use intermediate RPM values, so I found it much easier to buy a laser tachometer (about 20\$), turn the router on, and adjust the knob to get the desired value:



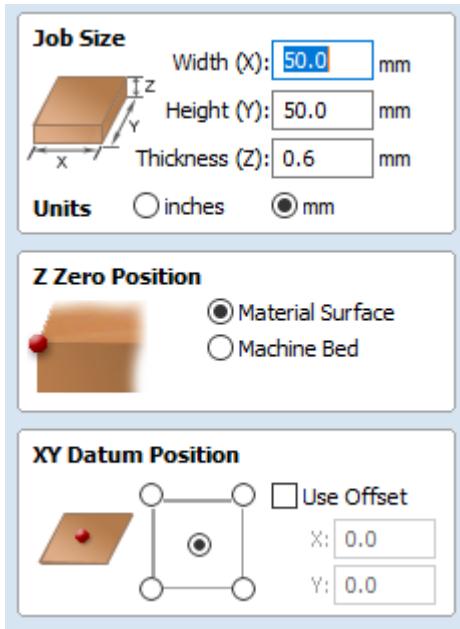
All it takes is a small patch of white/reflective tape on the collet nut:



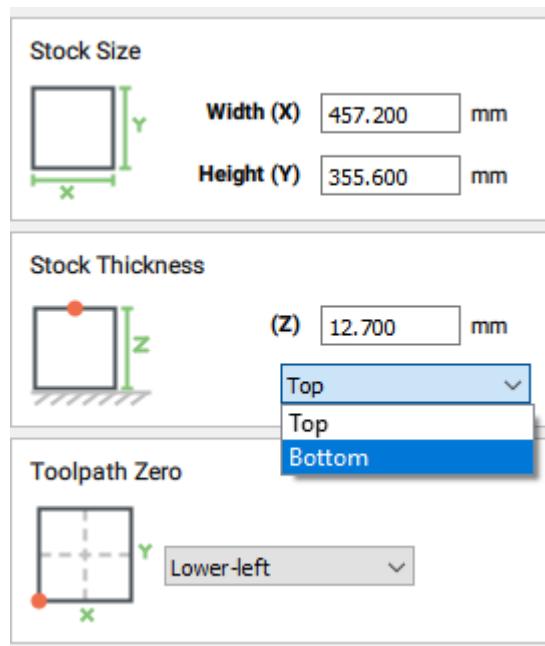
Zeroing strategies

The next step before cutting is to set the zeros. You will probably have noticed that two options are typically offered in CAM software : defining zero somewhere on the **top** of the stock, or on the **bottom**.

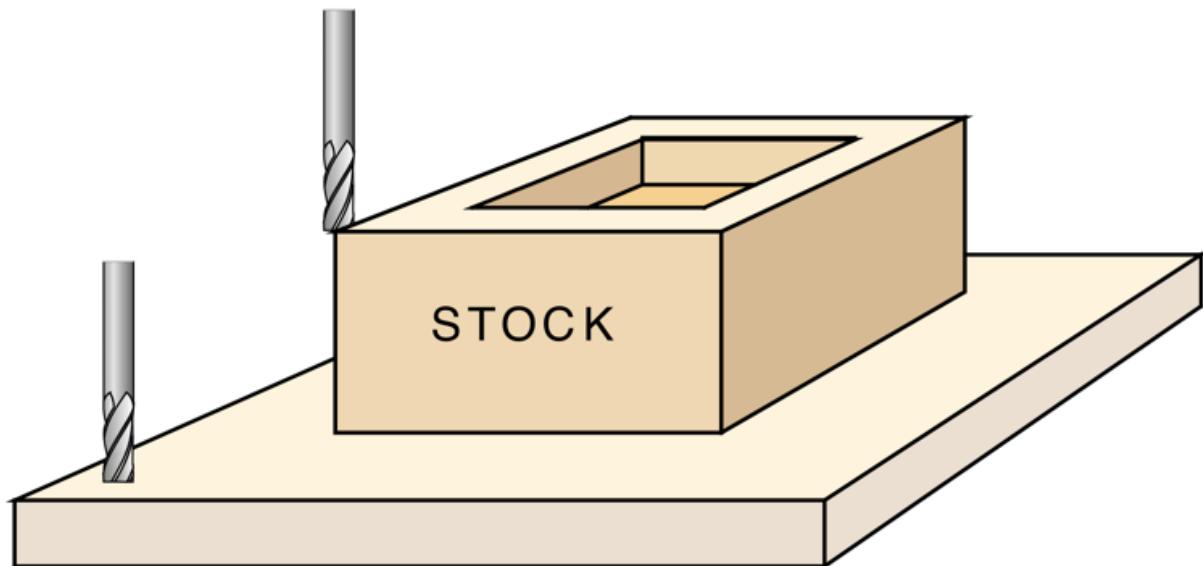
In VCarve this is called setting the "Z Zero position" to "Material Surface" or "Machine Bed":



In Carbide Create, this is selected using the "Top/Bottom" drop-down list

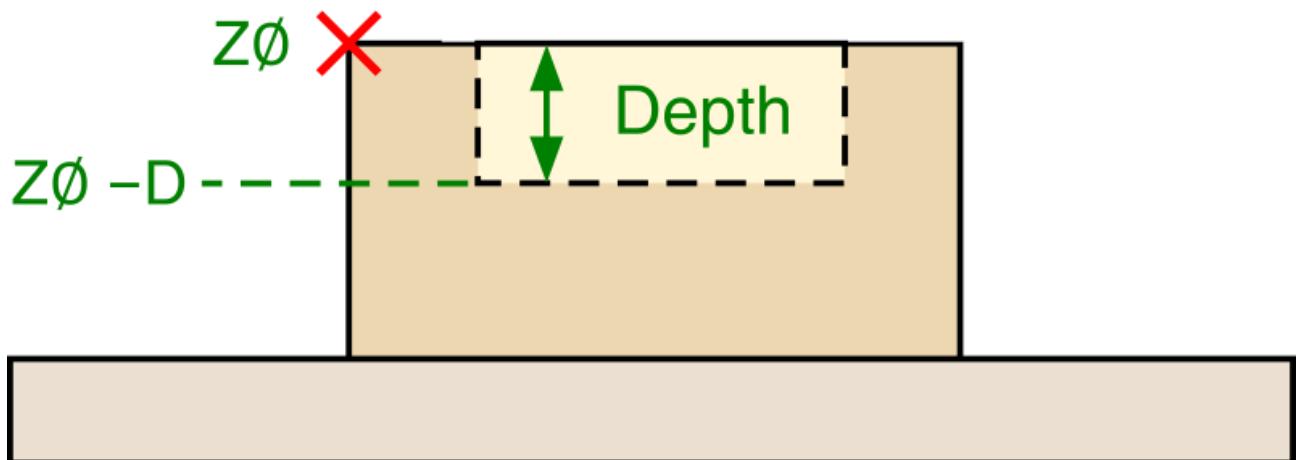


Consider this example where one would want to cut a rectangular pocket on the top of a square piece of stock:

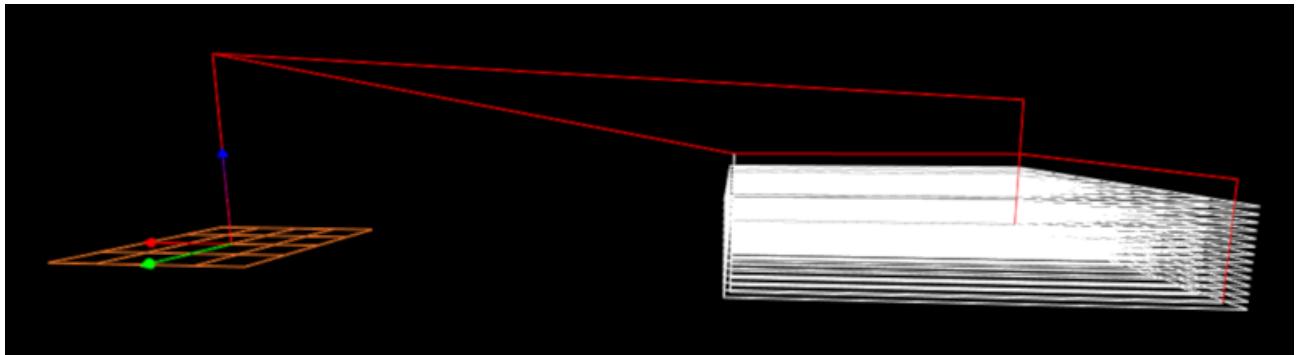


In many cases, one will want to zero on the **top** of the stock, just because:

- it feels more natural to set that plane as a reference ($Z0$), and then picture the machine milling the pocket down to a predefined DEPTH under that reference.
- as long as the pocket is not going all the way through the stock, there is no need to know or care about the stock thickness.
- it is often convenient to use the top left corner as a reference point for not only $Z0$, but also $X0$ and $Y0$, and using a corner probe makes that very easy (more on this below).

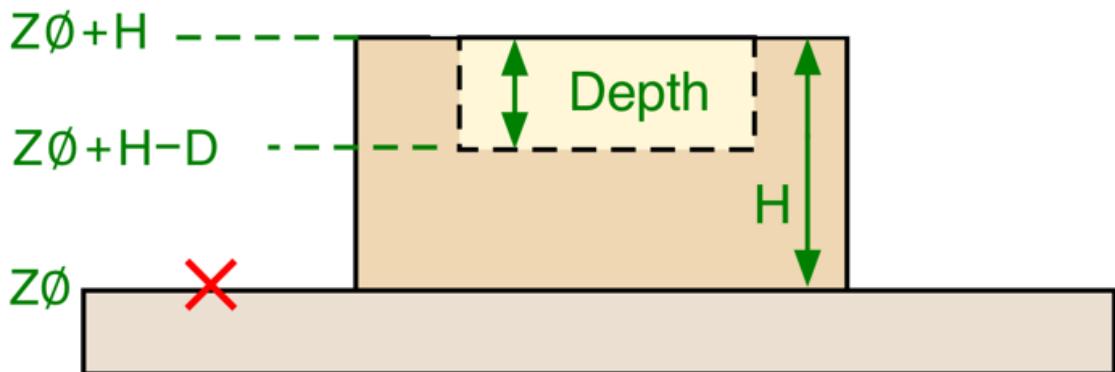


Previewing the g-code for that pocket toolpath would look something like below, with the tool starting from the upper left corner of the stock, retracting, moving to the pocketing area, and cutting to lower and lower depths:



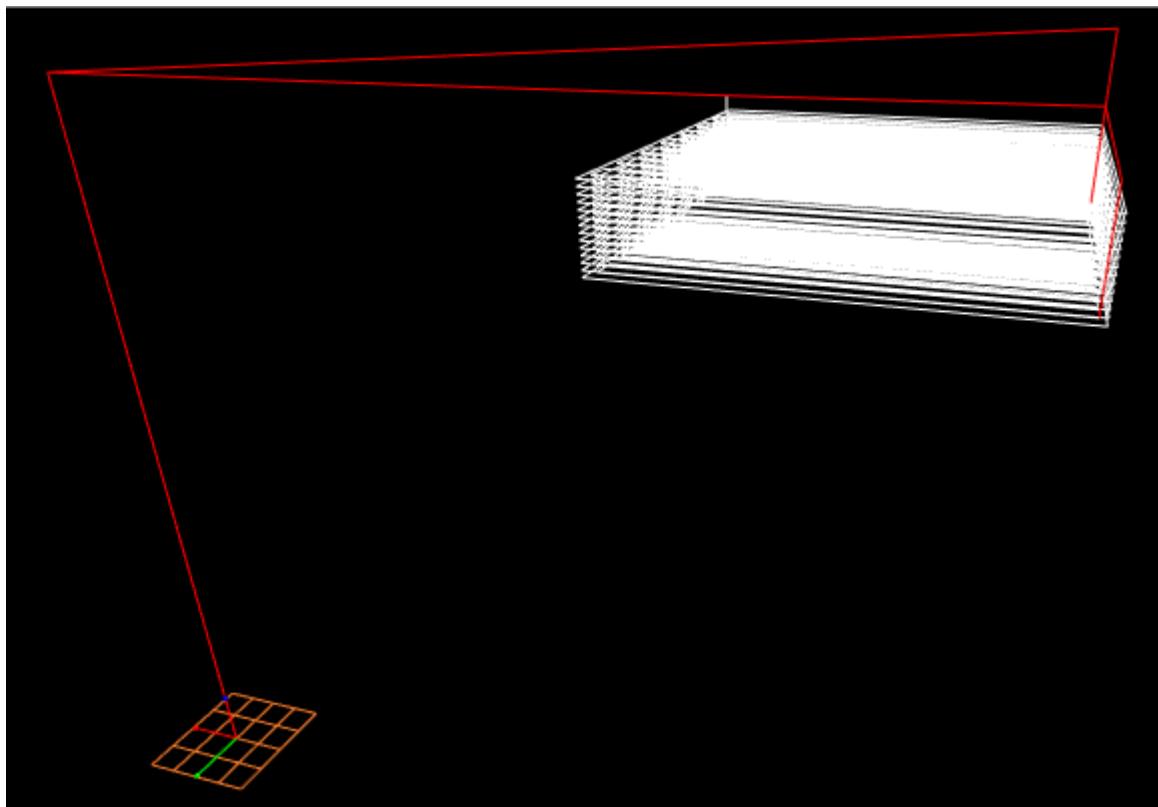
But sometimes, it is convenient to declare Z0 on the **bottom** of the stock / surface of the wasteboard:

- when the stock is not quite the desired thickness, and ones wants to surface it, down to a precise final height.
- when milling a piece that requires multiple tools, and there may not be an flat surface *left* after the first toolpath has been run (think of a roughing pass that would mill the top surface away and leave only a curved surface)



Now there is a catch with zeroing on stock bottom: the CAM software NEEDS to know the stock thickness (H in the picture above), such that it can offset Z0 by that, and end up in the same situation as is setting Z0 on top.

The G-code for the SAME pocket toolpath as above, regenerated after declaring Z0 being on stock bottom in the CAM, would look like this:



The tool would start from stock bottom, then rise all the way to the top surface (plus the retract height) and start cutting as it would if Z0 had been declared to be on top of the stock.

In summary,

- you can choose to set zero on stock top or stock bottom in CAM, but then on the machine you MUST zero the tool where you said you would!
- when setting zero on stock bottom, make sure to set the stock thickness value correctly in the CAM

Zeroing (manually)

After jogging to the vertical of the intended X/Y zero point, lower Z gradually then use fine steps to touch off on *e.g.* a piece of paper placed between the tool and the stock, stopping as soon as the paper cannot be moved freely under the tool anymore, and then tell the G-code sender to reset X0/Y0/Z0. Yes, it does mean that you will zero a tiny bit above the real stock surface, but the average thickness of a piece of paper is around 0.004" / 0.1mm, and

chances are that you are already compressing the paper since it cannot move anymore, so you will actually be very, very close to the stock surface.

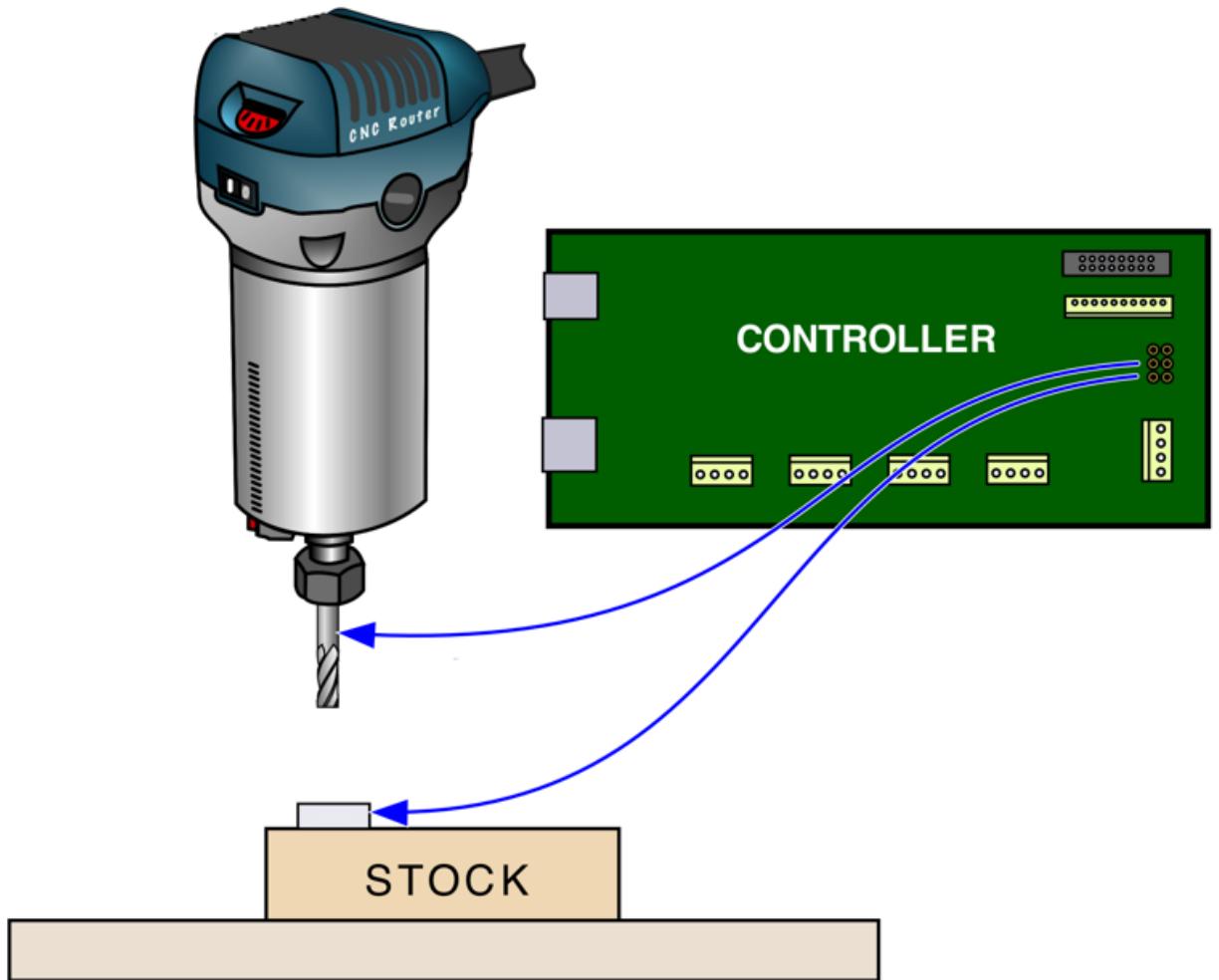
Miscellaneous tips:

- double-check your Z jog step before doing the final steps to touch off. It is very easy to hit the "down" arrow one time too many by mistake, and if the step is still say 0.1", this might bury your tool into the stock, or more likely break it if it is a small endmill.
- touching off with a very pointy V-bit can be tricky, it's easy to go too far down without noticing, so you should use even finer jog steps and/or extra caution.
- for jobs involving multiple tools, consider the fact that you will have to re-zero after tool change: it is more convenient to zero on a part of the surface that will...still be there after the first tool has done its job.

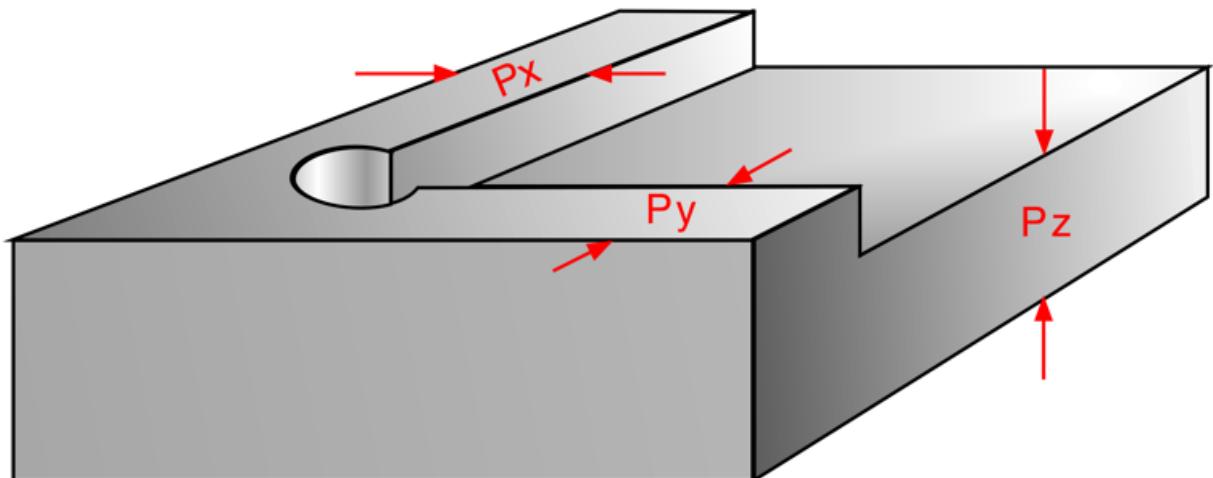
The main limitation of manual zeroing is that you need to eyeball the X/Y location, which for some jobs is not precise enough. And of course, the manual careful jogging to touch off is somewhat time consuming.

Zeroing (with a Probe)

Enter the touch probe, to automate the zeroing process. The Shapeoko controller has a dedicated "Probe" input, that works like the other limit switches. It detects whether there is continuity between the two pins:

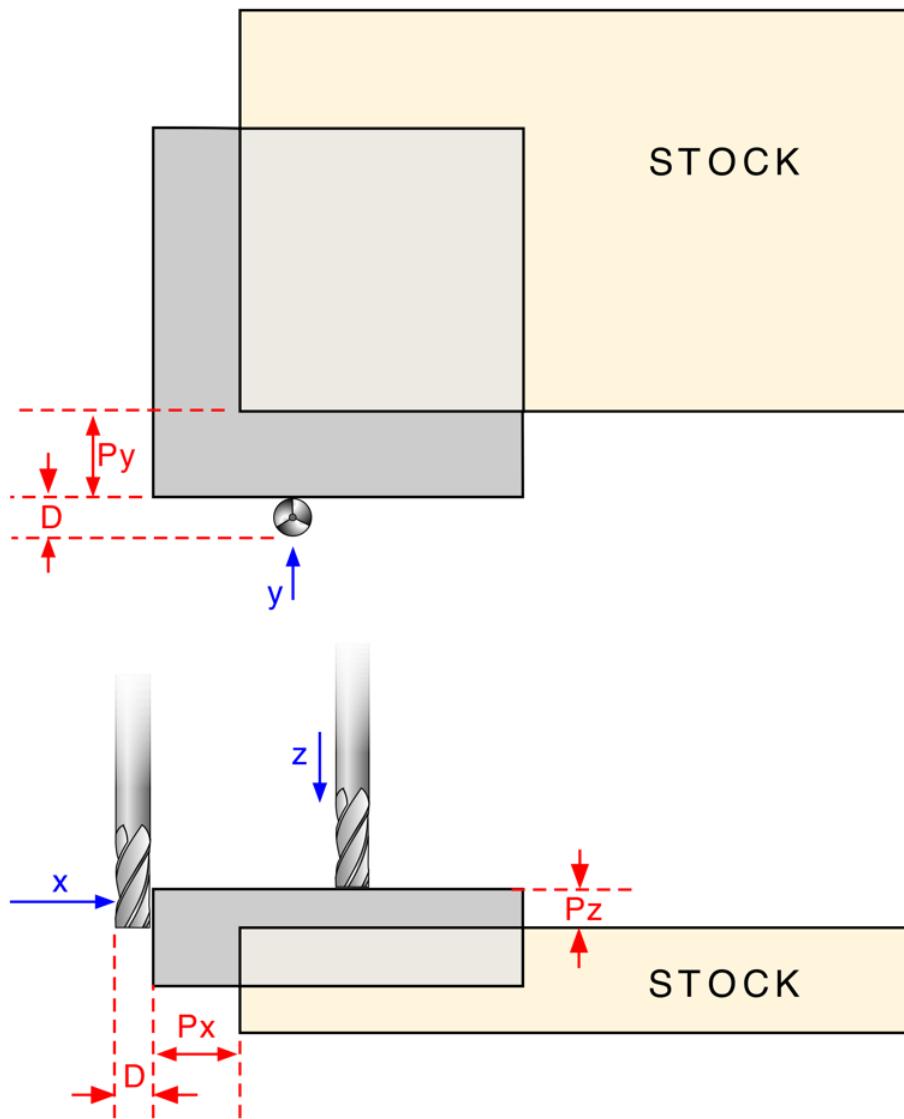


For zeroing Z only, a probe can boil down to a piece of (conductive) metal sheet (of known thickness), for zeroing X, Y and Z, it needs to be 3-dimensional but the principle is the same. X/Y/Z probes typically have a recessed face on the bottom side, so that they can be placed on a corner of the stock top surface:

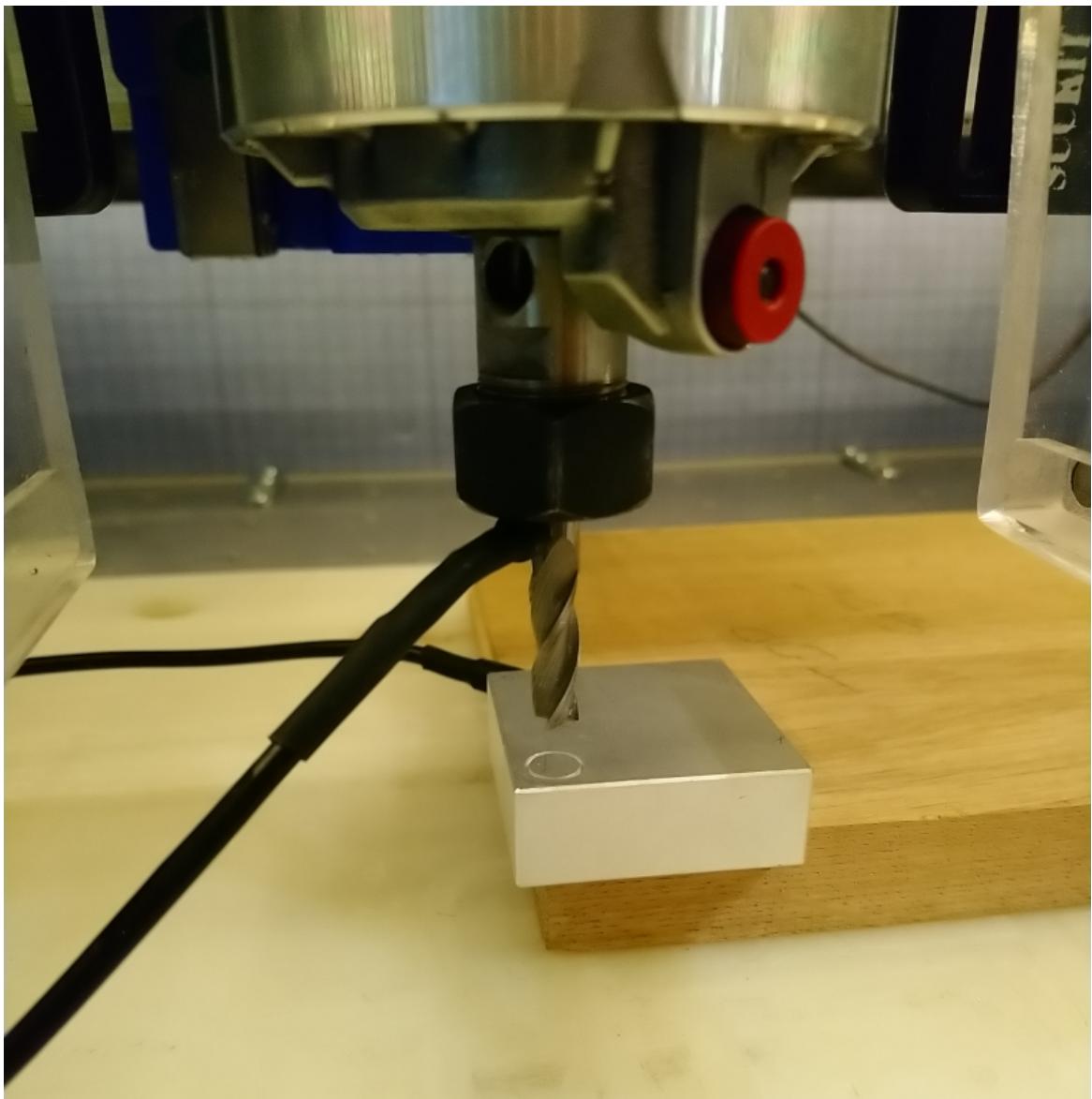


- ⓘ Probes come in two types: passive and active. Passive is just what was described above, i.e. a glorified metal cube. Active probes have internal electronics to support features such as an embedded test LED that lights up when contact is made, which is useful for checking that the probe chain is working fine before initiating the probing cycle

The probing goes like this:



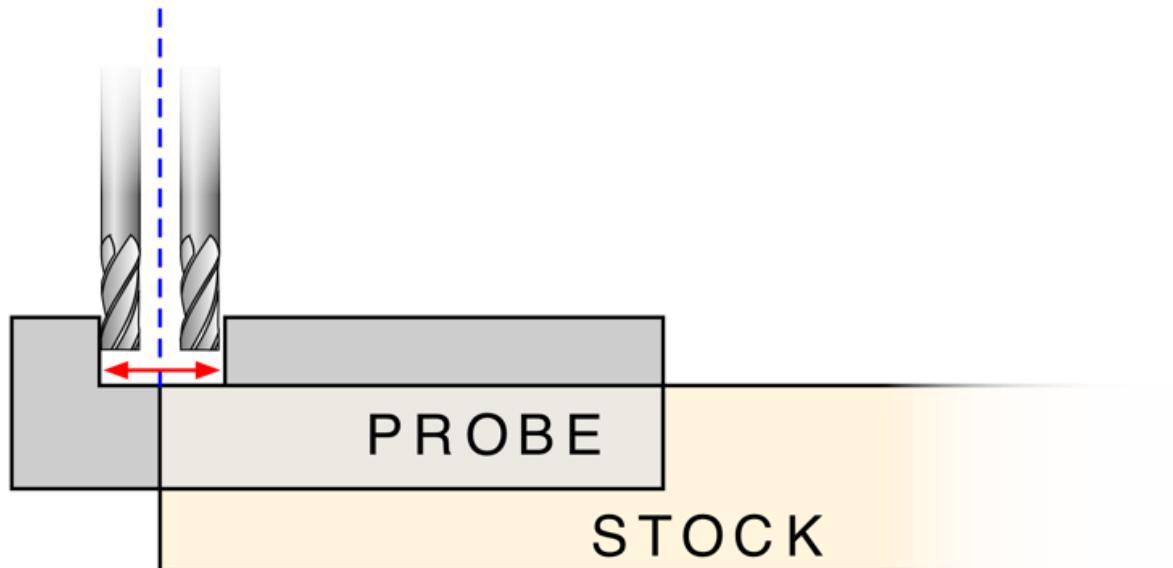
- the probing cycle starts with the tool raised above the probe. It could be anywhere above, but there is usually a target area above which to (coarsely) position the tool, to facilitate the rest of the sequence.
- the tool is lowered along Z, until it makes contact. When it does, the software can just read the current Z and subtract the thickness of the probe (P_z in the sketch above) to get Z_0 . This Z touch off sequence can be repeated to average out values and improve precision
- if the probing cycle is configured to also probe X and Y, it retracts the tool and proceeds to move away from probe, lowers the tool and then comes back towards it until it detects contact, then repeats that operation for the other side. It can then determine X_0 and Y_0 by reading the X or Y values at contact, add P_x or P_y , and add half the (preconfigured) diameter D of the tool itself.



If the stock shape does not have a straight corner, the probe can also sit on top of the stock surface and be used for Z probing only (the software will know that when Z probing only is selected, it should compensate for the total height of the probe, not just the PZ step).

One problem remains: the X0/Y0 computations depend on (half the) diameter of the tool, so the geometry of the tool must have been configured beforehand, and this manual operation is error prone. Also, the *actual* precise diameter of the tool is often not quite the advertised value, so this will introduce a *slight* error in X0/Y0, which will result in a shift between runs with different tools. One more probing trick can be useful: if the probe has a hole, one can lower the tool into the hole and then probe its sides: probe left and memorize current X value, then probe right and memory current X value: the average (middle) of these two values is at the X center of the hole. Repeating this operation by probing on the front/back side of the hole will locate the Y center of the hole. Since the location of the

center of the hole is at a known distance from the inner corner of the probe, this gives X0 and Y0. Z-probing can then be done normally elsewhere on the top surface. The beauty of this method is that it is independent of the tool diameter!



@neilferreri on the forum came up with a wonderful probing macro for CNCjs, that does just this, go check it out: <https://github.com/cncjs/CNCjs-Macros>

Running a single-tool job

In case the G-code file was generated from a design that only used a single tool for all toolpaths, this is straightforward. Nothing special to be said here so I'll just share my own habits:

- if your G-code sender allows, raise the tool after zeroing. It is normally not necessary, but depending on your CAM post-processor's settings, the G-code may or may not contain an opening statement to do it, so this will give you a (little bit) of time to react if something unexpected happens. It is also required anyway in case your dust shoe model cannot be installed if the tool is lowered (and since zeroing with the dust shoe in place is not fun)

- If using a fixed-height dust shoe, double-check that it is lowered at stock surface level and won't crash into anything during the job.
- turn on the dust collection (and air jet/lubricant mister if applicable)
- put on your safety goggles and/or close the enclosure window.
- turn on the router.
- hit start...with your hand/mouse over the pause/stop button.

 At least the first time I run a new project, I like to watch & listen throughout the cut, looking for hints of incorrect cutting parameters (chatter, bad looking chips), debris build-up, or anything I might have done wrong in the CAD/CAM, or could optimize.

Running a multi-tool job

Now, when multiple endmills are involved in a project, one must deal with **tool changes**.

This involves two things:

- a moment where the machine is paused and the router is accessible, to go and change the tool.
- readjusting the Z0 after installing the new tool. Indeed, the new tool is probably not the same length as the previous tool, and is probably not seated at the same depth inside the collet anyway. Since Z0 was set using the tip of the previous tool, it needs to be reset.

There are basically two ways to manage this:

1) The manual way :

- split your project into multiple G-code files, **one per tool**. A given file may still contain multiple toolpaths using the same tool.
- zero using the first tool, run the G-code file corresponding to that tool.
- turn off the router and install the second tool.
- redo the zeroing procedure, for Z0 only.

- this is mandatory, and it's easy to forget...don't ask me how I know.
- The "return to Z0 + xxx mm" button in Carbide Motion is very convenient to go back to X0/Y0, but beware: one day the second tool may stick out by more than xxx mm compared to the previous one...and that command may result in crashing the new tool into the stock
- load and run the second G-code
- proceed similarly each time there is a new tool change needed.

2) The semi-automated way

- this approach requires one to have a **tool length offset probe** installed on the machine.
- generate a **single G-code file** containing all toolpaths, for all tools.
- Zero using the first tool, then run the G-code file: the machine will go and probe the length of that first tool, and will start cutting.
- The G-code sender will know when a tool change is required (by detecting the "M6" commands inserted in the G-code file by the CAD tool), and it will pause the job.
- install the second tool.
- proceed to the automatic Z0 adjustment:
 - In Carbide Motion when using a BitSetter, resuming after the tool change prompt will trigger a new tool length offset probing, moving at the predefined BitSetter location to do so automatically.
 - In other G-code senders, it may be needed to launch a macro to trigger a new tool length offset probing, or even go and adjust Z0 manually if one does not have a tool length probe installed.
- The cut then proceeds, until the next tool change (if any), where this tool length offset adjustment is repeated.

Regardless, a few points to be considered during tool change:

- some people like to jog the router to the front using a predefined position in the G-code sender, to have easier access. Carbide Motion does this automatically when the BitSetter option is enabled.
- **safety:** if you are using a router or spindle that is externally controlled, I would recommend actually cutting its power source. Do you really trust your PID/VFD that much? If you are manually turning the router on and off, this is less of a risk but I choose to be extra cautious (paranoid?), and also kill the router power source (in my case, flipping a switch on a control panel)

- remove the tool and collet and make sure the collet taper is free from any debris/dust that could create runout for the next tool.

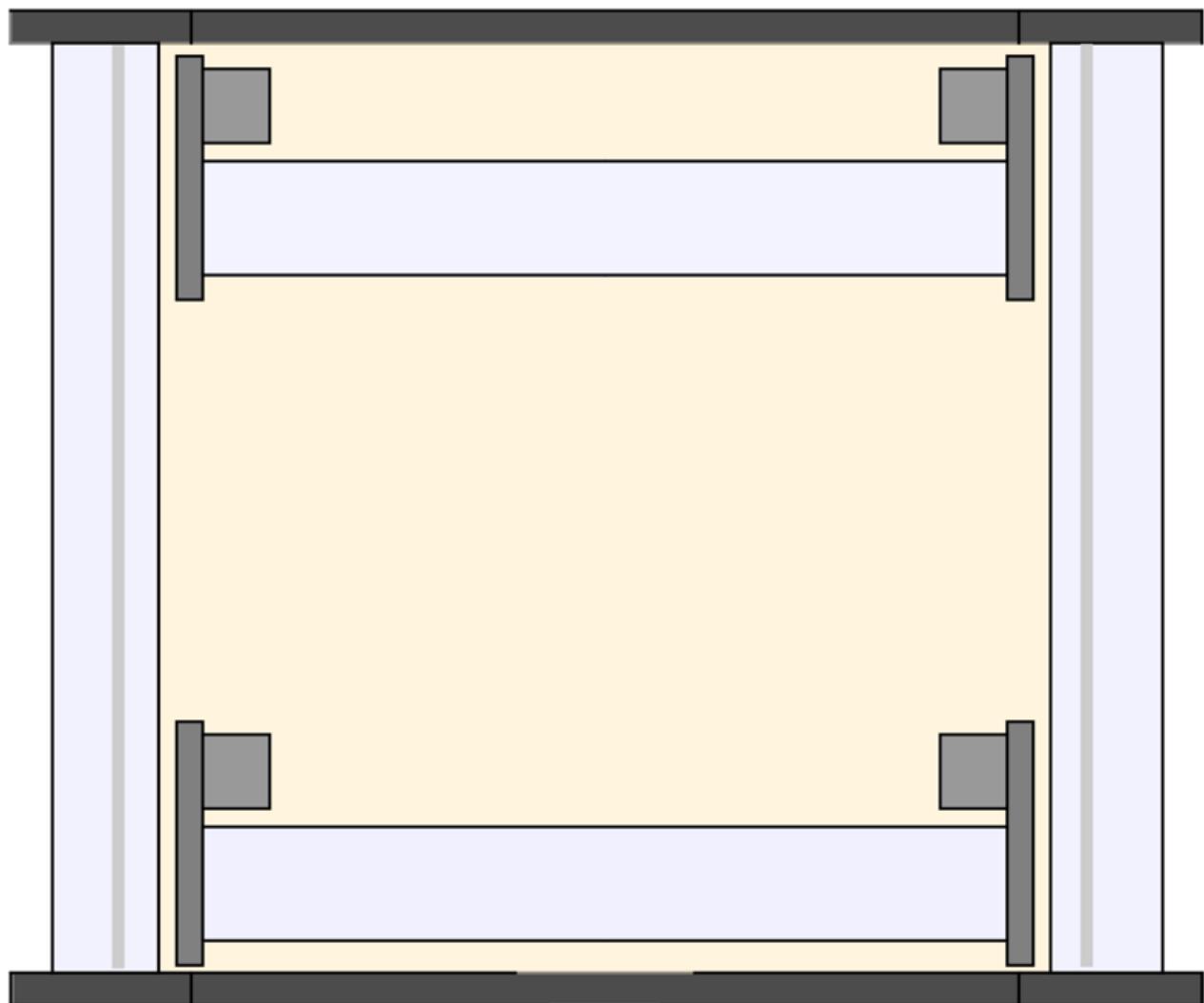
 If the collet is stuck in the taper, you may try moving the endmill gently left and right in the collet to get it out.

Squaring, surfacing, tramping

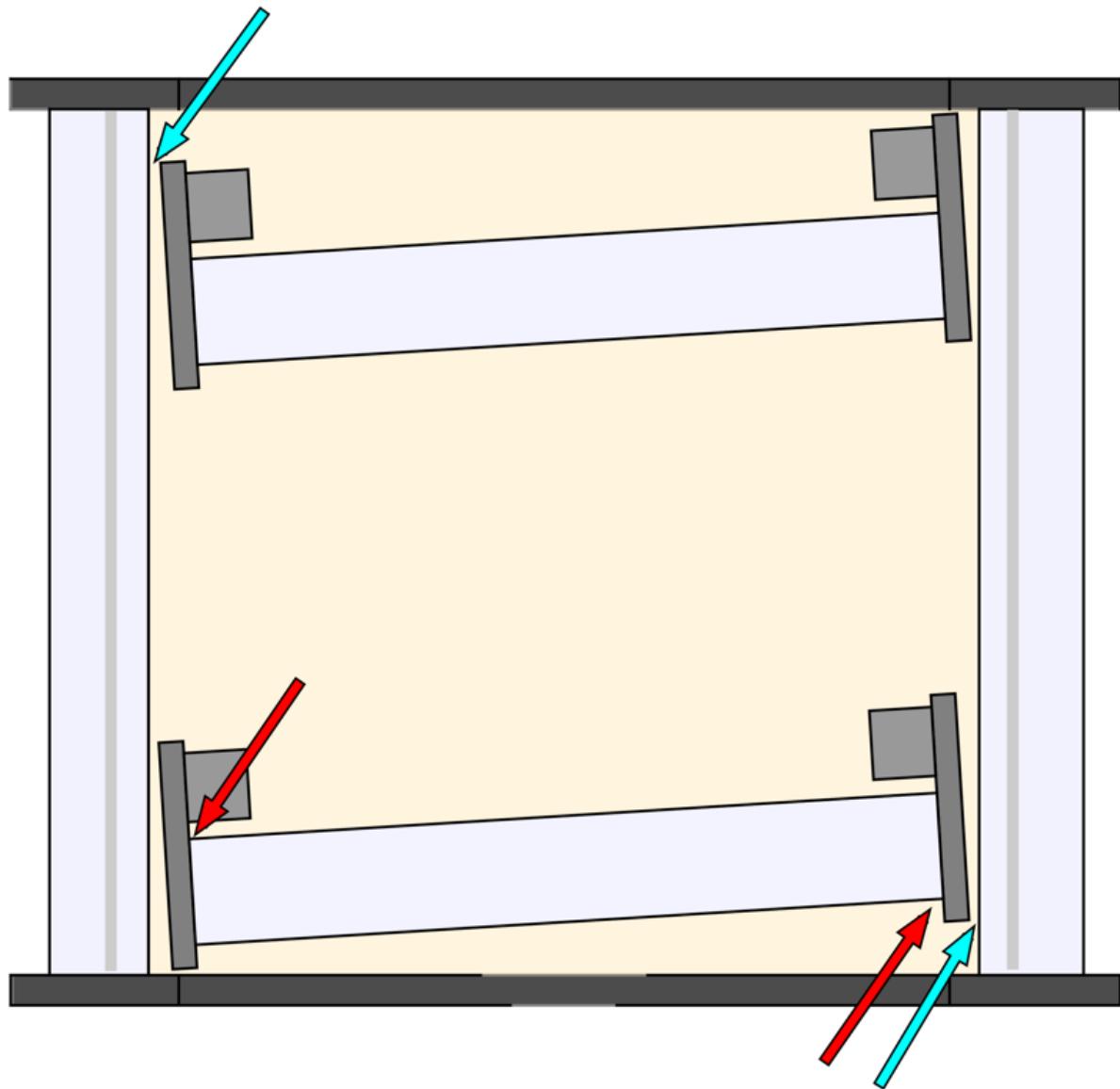
Squaring the machine

Squaring the machine is covered in the Shapeoko assembly instructions, this section just adds a few notes on why this matters, and a few tips.

In the best case scenario, bringing the gantry to the front side, tightening the front screws, sliding the gantry to the back, and tightening the back screws is enough to make the gantry square to the Y rails. Here's a view from the top with the gantry pushed all the way back or all the way to the front steel plates, ideally the side plates should make contact simultaneously, leaving no gap on either side:



But you may and likely will get a small gap on one side, and a similar gap on the opposite side when pushing the gantry to the other end of the Y rails (teal arrows on the exaggerated sketch below):



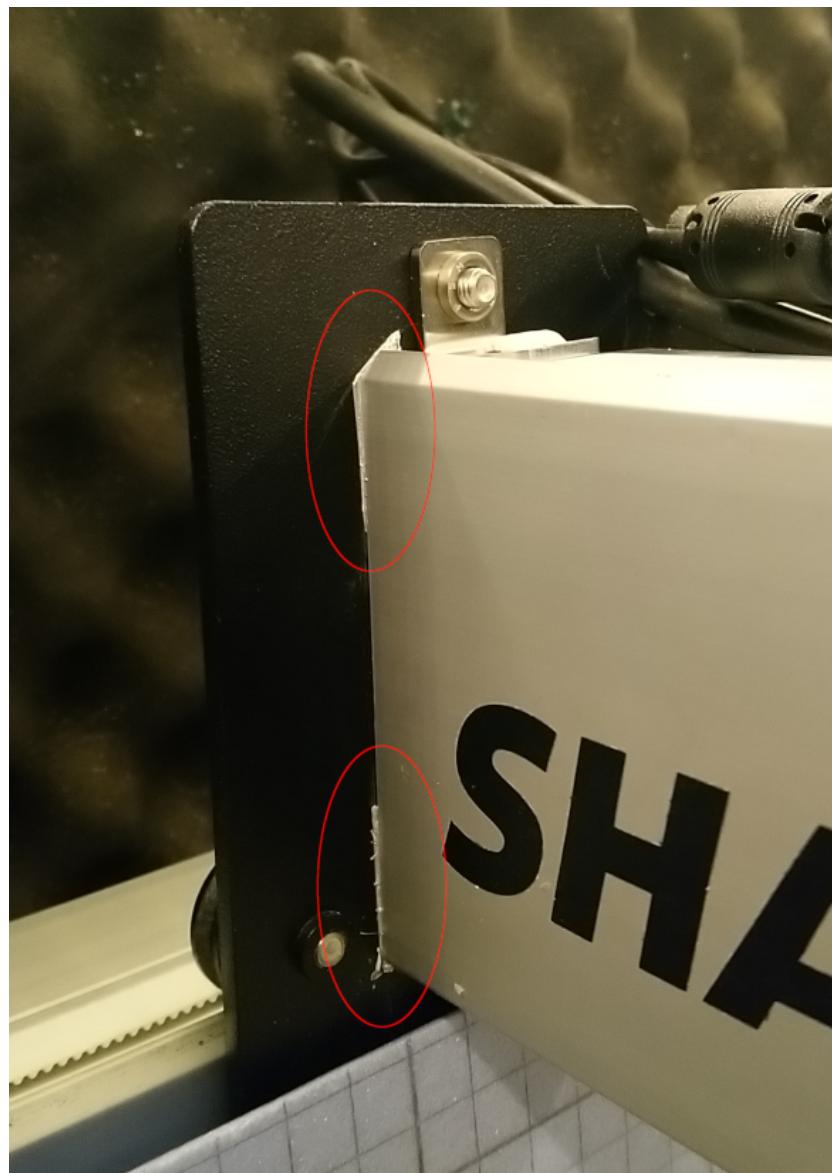
This can be due to a number of reasons (X extrusion sides not being perfectly square, tolerances in the side plate / V-wheel /washers assembly, etc...), and common solutions to fix this are:

- to insert **shims** between the X extrusion and the side plates on both ends (but on opposite sides of the rail, e.g. where the red arrows are on the sketch). Of course, should the skew be in the other direction, the shim location should be reversed. One shimming technique is to just fold a piece of aluminium foil a few times over, and insert it there. Or use a feeler gauge.

- to use **washers** carefully selected to have slightly different thicknesses, and use them on the front and back V-wheels on each side plate, to achieve the same effect (in the sketch above, one would install slightly thicker washers on the back wheels on the left side, and on the front wheels on the right side)

i You *could* also try and file the end of the extrusion flat to correct for squareness, but honestly this is not easy to do correctly for the casual hobbyist

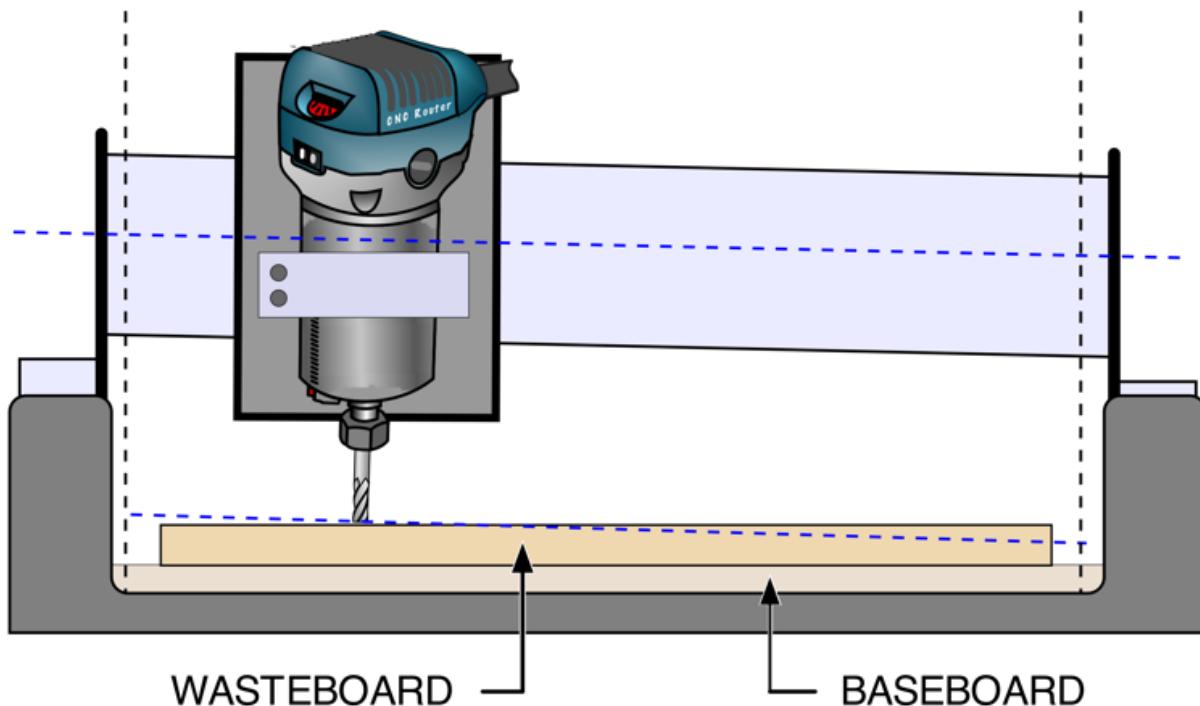
Here's a (poorly executed) example of aluminium foil shimming on my machine, that was enough to fix the gap:



Once X and Y axis are square, the next step is to make the wasteboard as flat and parallel to the mechanical X/Y plane of the machine as possible, and this is what surfacing the wasteboard does.

Surfacing the wasteboard

Either the X/Z rail is not perfectly parallel to the bed (as shown exaggerated in the sketch below), or the wasteboard is not perfectly flat to begin with, either way the end result is that after zeroing somewhere on stock surface and moving the endmill elsewhere along the X/Y axis, the tip of the endmill can end up being above or below stock surface:



Running a surfacing toolpath on the top of the wasteboard will make it parallel to the gantry, therefore providing a known-flat reference to mount the stock onto, which turns out to be important for projects where depth accuracy matters:

- V-carving is a typical example: a small error in the depth being cut shows up as a difference in the width of features on the surface, which can be quite visible. In severe cases or when doing very shallow V-carving, the tip of the V-bit might not even make contact with the stock anymore on one side of the piece.

- PCB engraving is the extreme case, since by definition it cuts a very, very shallow trace onto the (supposedly) flat surface of a copper clad board: any minute difference in depth will ruin the quality of the PCB traces.

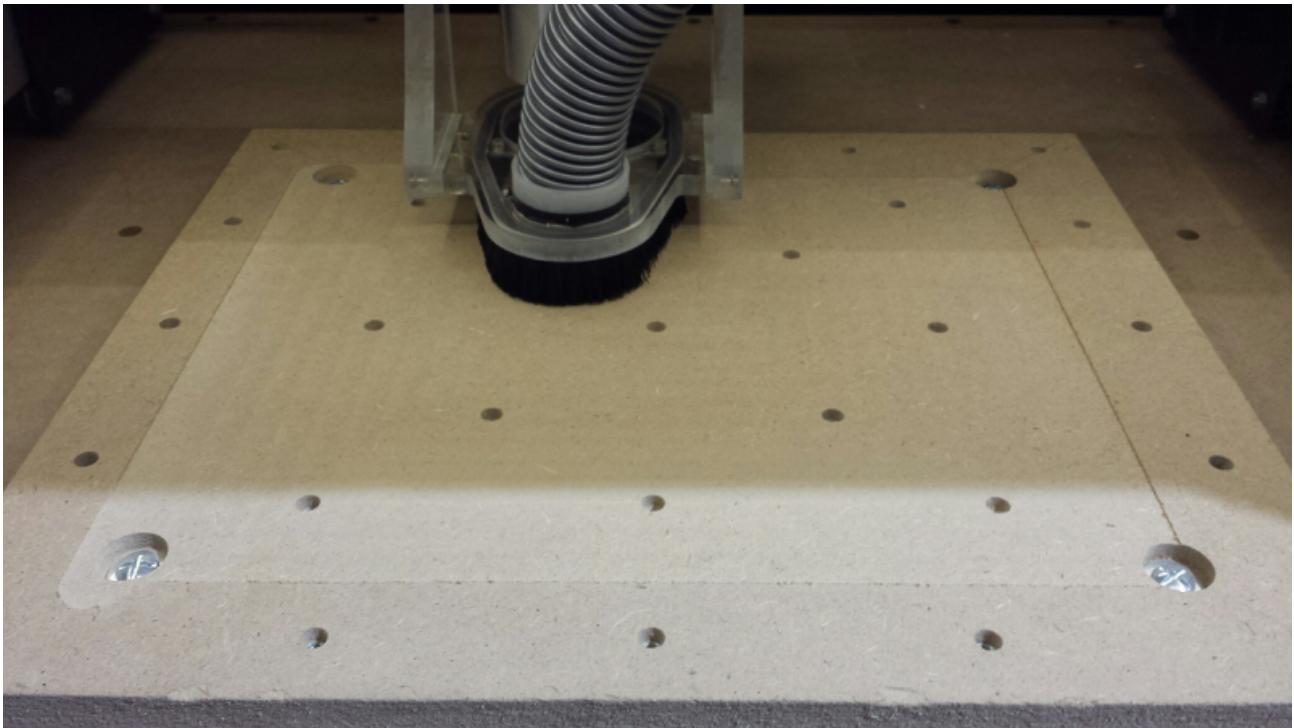
 For PCB engraving, another technique to compensate for depth variations is to use bed leveling, i.e. use a probe to map the height of the PCB surface in various points, and compensate for it at the G-code instruction level

- but even regular jobs are impacted: in the example illustrated above, a pocket cut on the left side of the stock would end up being shallower than expected, and the same pocket cut on the right side would be deeper than expected. And both would have a sloped bottom.

Since the surfacing toolpath has to cover a lot of real estate, a fly cutter with a large diameter is useful (but any large endmill will do, it just takes longer).

 Tip: draw squiggles with a pencil all over the wasteboard before starting: after the surfacing operation you will be able to tell if you missed a low spot (i.e., did not surface deep enough to make it perfectly flat)

Here's an MDF wasteboard surfacing operation in progress:

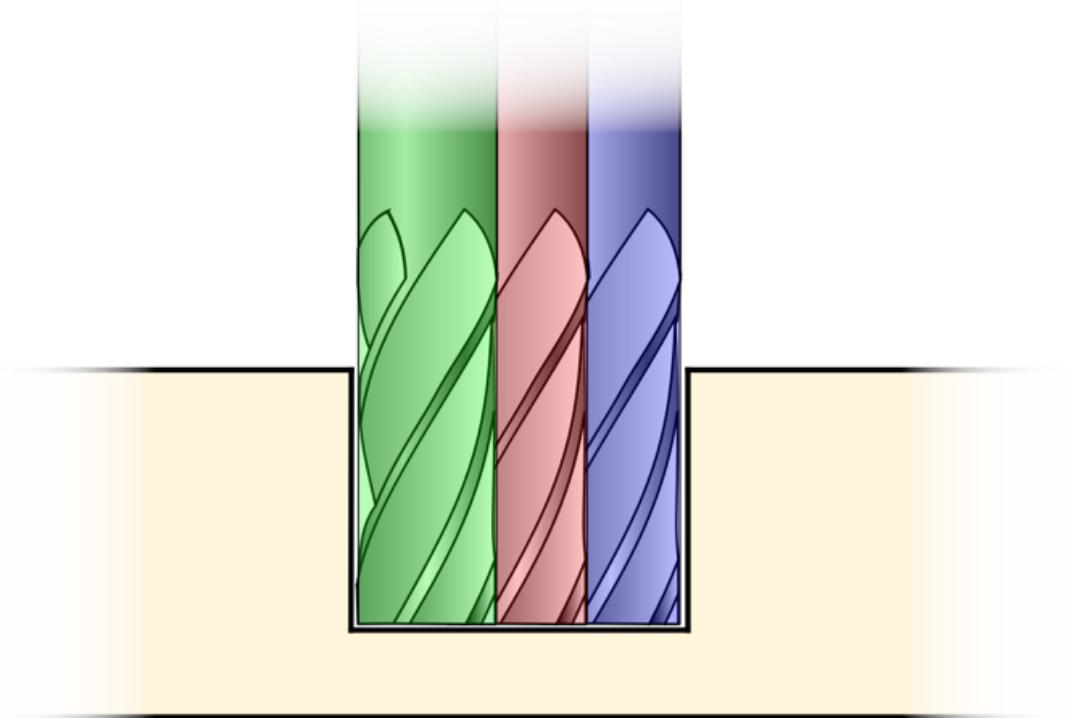


- (i) For MDF wasteboards, it is recommended to seal the surface after surfacing with a few coats of whatever you have on hand (shellac, polyurethane, varnish, lacquer...), this will reduce the likelihood of the MDF absorbing a lot of humidity, as well as make the surface harder/less likely to tear off, especially when using the tape & glue workholding method.

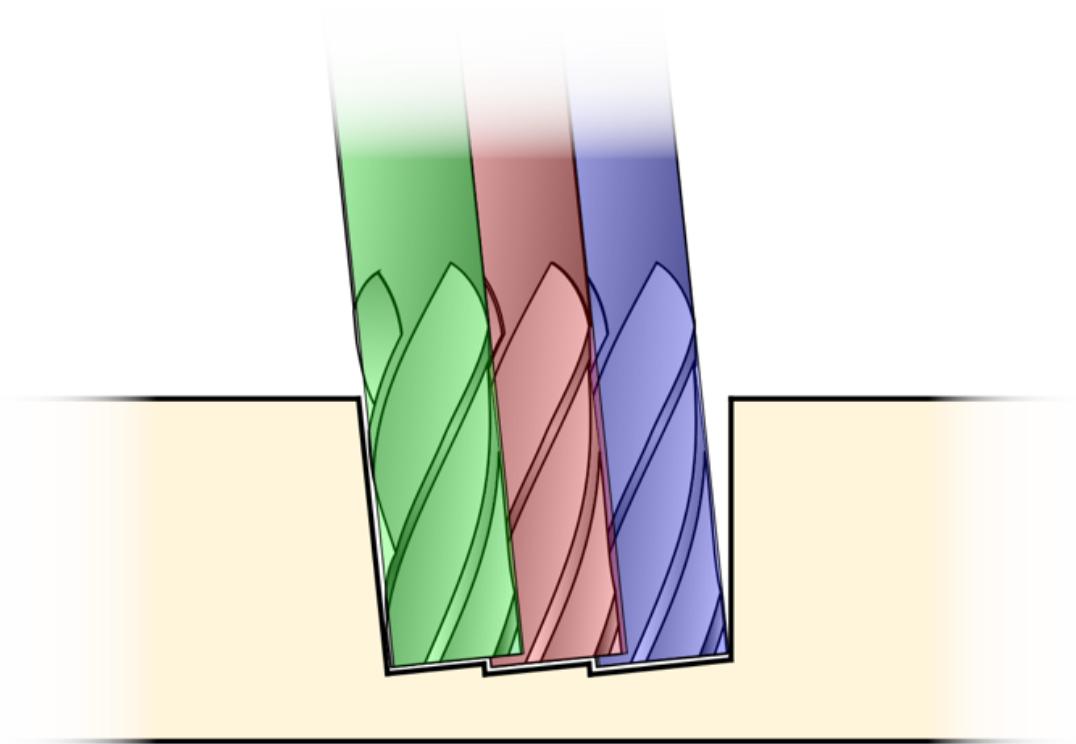
While surfacing ensures that the surface of the wasteboard is true to the X/Y axis of the rails, it does not do anything about any potential **tilt** of the router itself around the X or Y axis: hence the need for trammimg the router.

Trammimg the router/spindle

In a perfect world, the router axis would be perfectly square to the surface of the wasteboard (and of the stock material). For a pocket operation, a cross-section of three successive passes of an endmill could look like the sketch below, the bottom of the pocket would be flat, and its width exactly as intended:



But in reality, the mechanical assembly and tolerances on the router mount are such that the angle between the endmill axis and the stock surface can be slightly different from 90°. Here's a very exaggerated example:



Now the pocket ends up being wider than it was programmed to be, AND the bottom of the cut is not perfectly flat, there are small ridges at the overlap between passes. Here's a real life example:

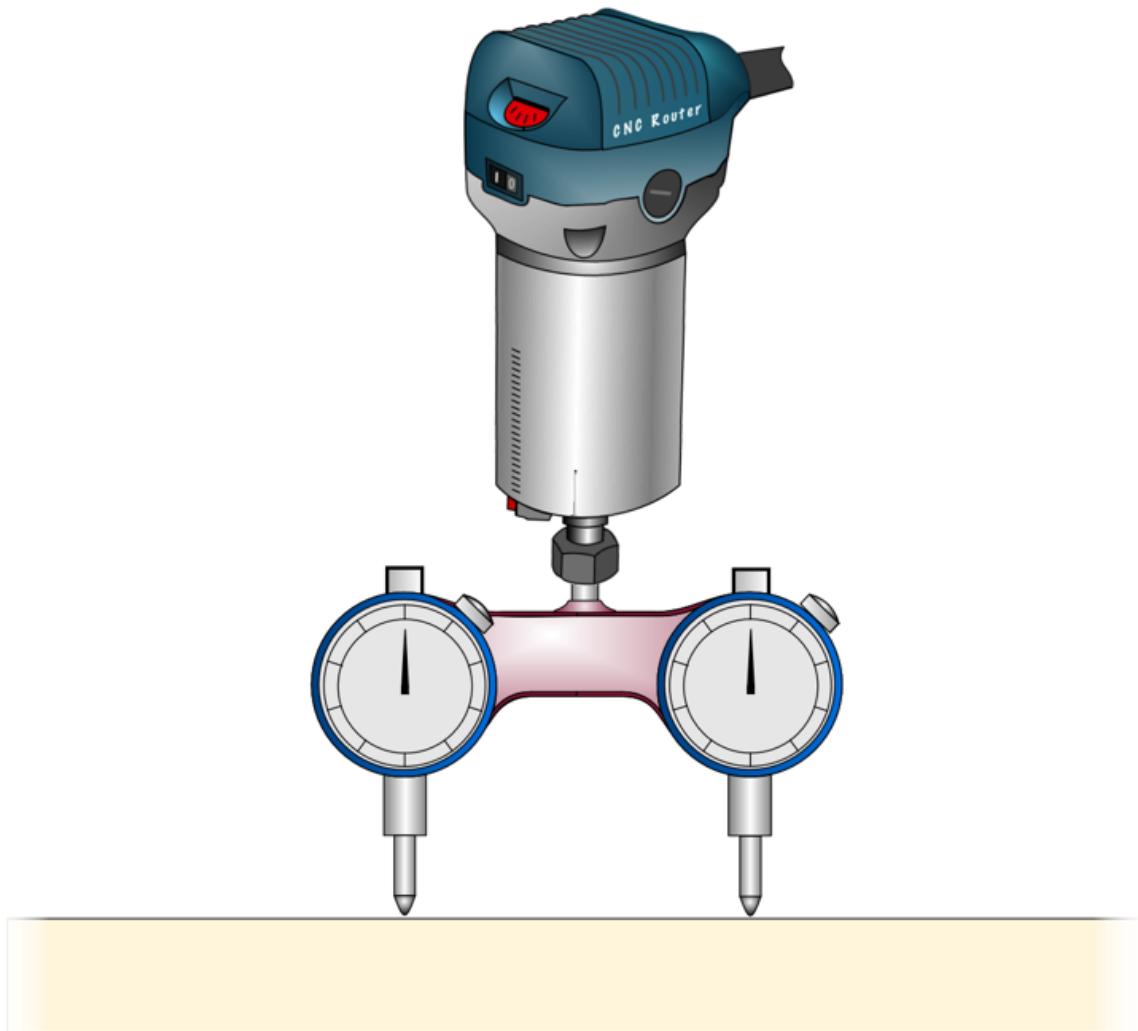


If the router is slightly tilted around both the Y axis and the X axis, there can be ridges visible in both X and Y directions. To get rid of those, the router should be **trammed**, i.e. its Z axis made perfectly orthogonal to the wasteboard.



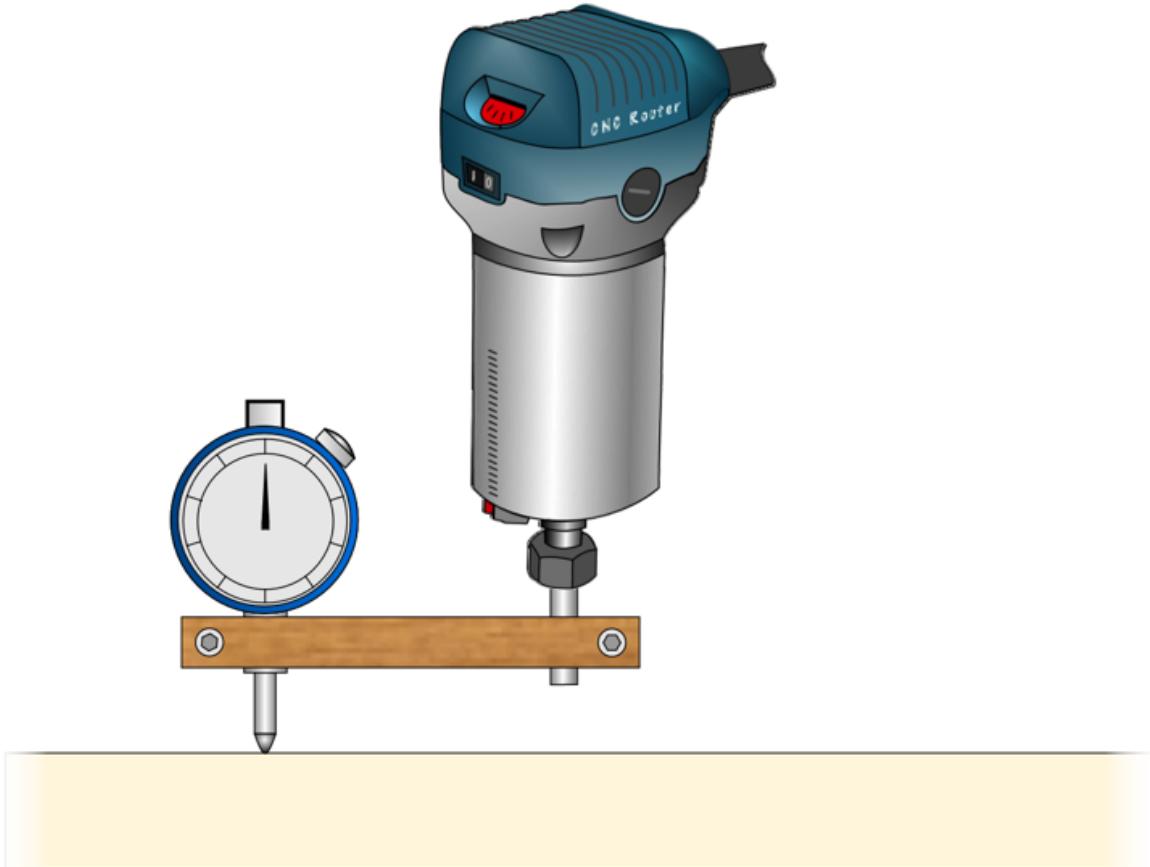
You should surface the wasteboard **before** tramping. If you don't, you may end up including any tilt or thickness variation of the wasteboard in the adjustment

One option is to use a dedicated tramping device mounted in the router and integrating two dial indicators. Any difference between the readout on the indicators is a sign of a tram angle. The arm can be rotated manually to check the angle in any direction:



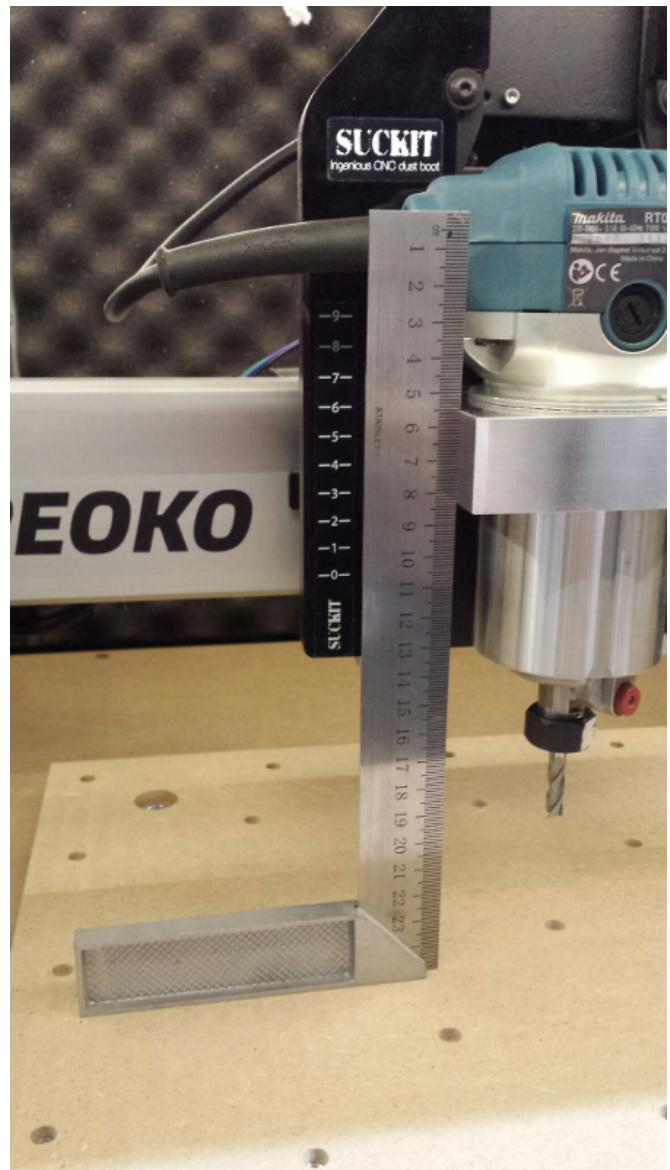
Very convenient, but pricey (unless you make your own, but getting enough precision in the parts/assembly is not easy).

The next best thing if you just have a dial indicator, is to mount a stub/dowel in the router, and attach the indicator at the end of a small arm. Then rotate the arm manually to detect variation, and determine which way the router is tilted from there:

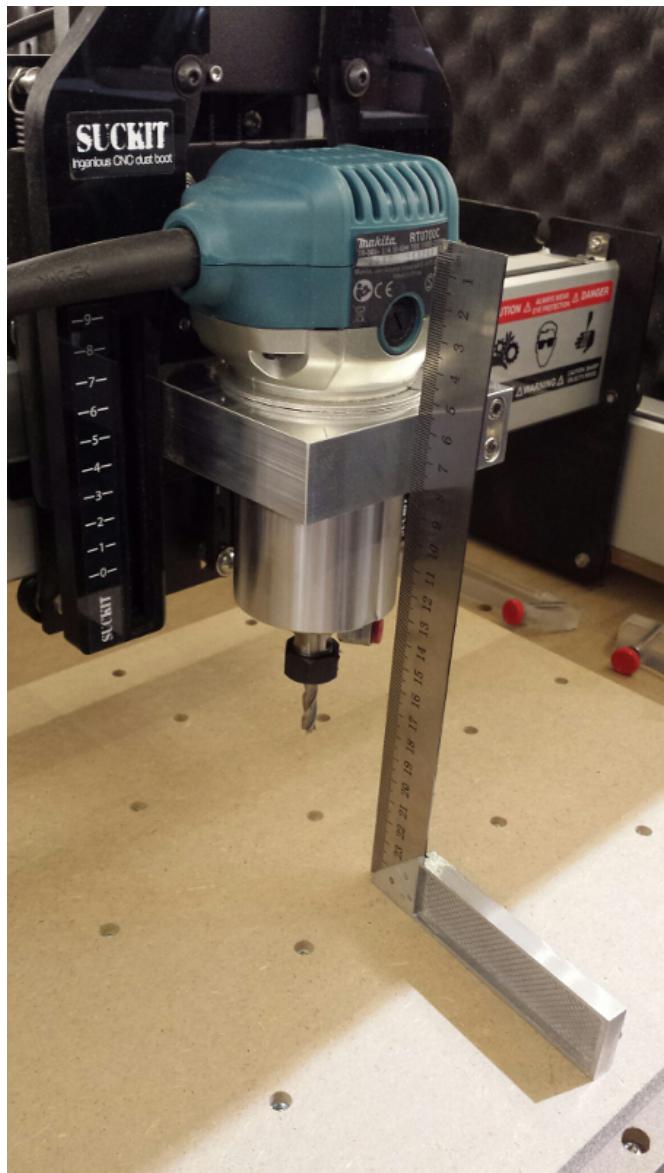


- ⓘ An even cheaper version of this, is replacing the dial indicator with a fixed dowel, and rotate the arm: if the dowel catches the wasteboard on one side but not the other, adjust router tilt to compensate, rinse and repeat.

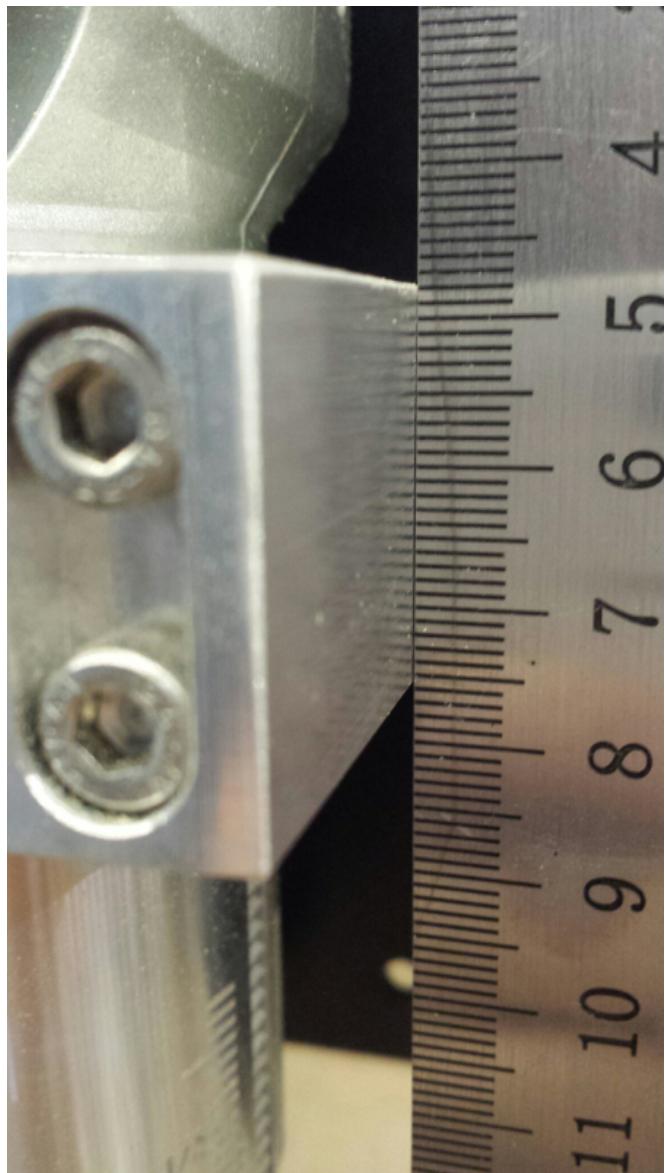
And finally, if you don't have a dial indicator and don't even want to bother with the dowel & arm gadget described above, you can still do a "good enough" trammimg using a machinist square placed against the router mount. Check the left/right tilt:



and then the front/back tilt:

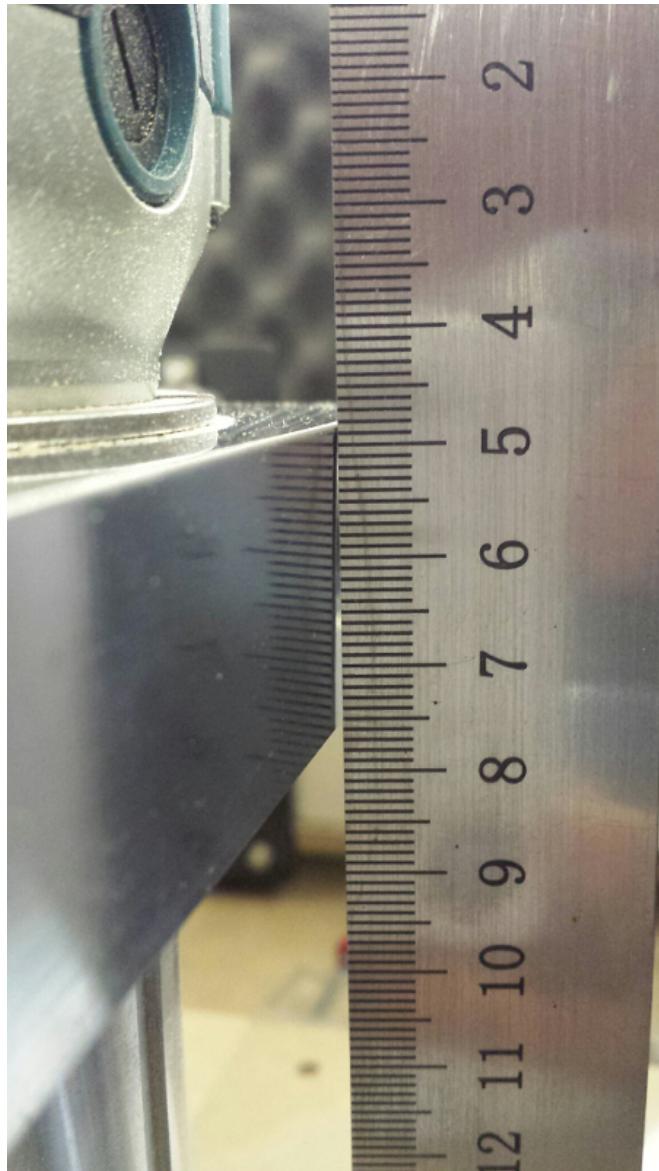


On this machine, the left/right tilt happened to be almost zero:



If adjustment is necessary, loosen the screws holding the mount, tilt it ever so slightly in the opposite direction, tighten everything and re-check. This can be an iterative (and sometimes frustrating) process, but well worth the time spent. To make this easier, you could buy or make an adapter plate that sits between the router mount and the Z-plate and has eccentric nuts to tilt it easily left and right.

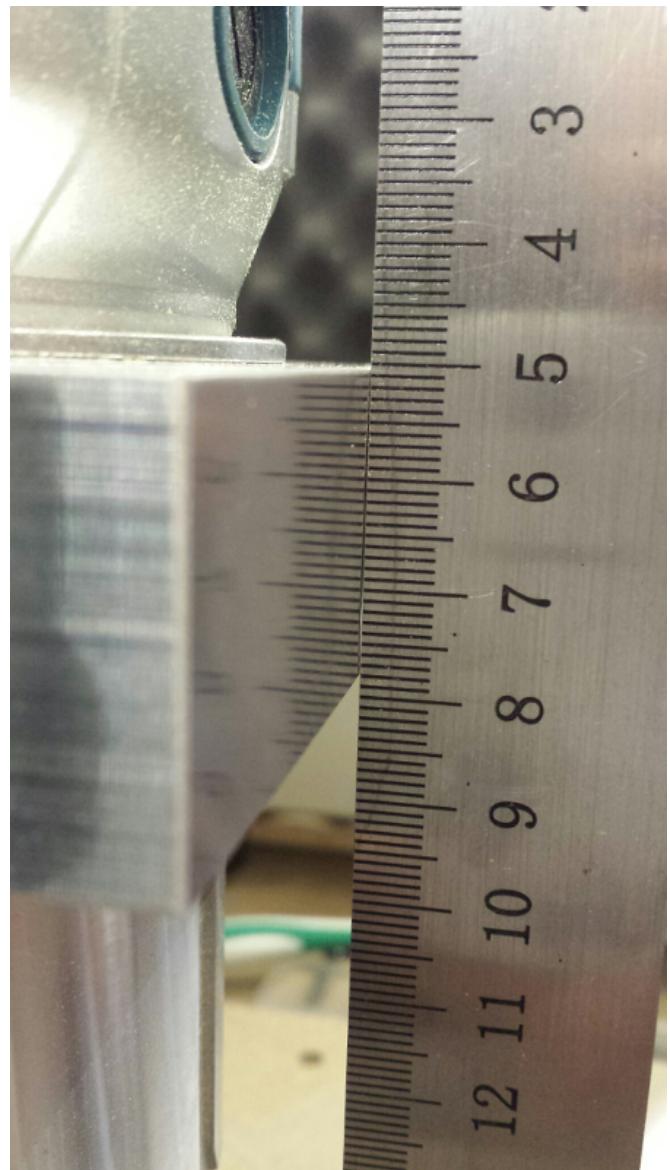
Now on the front side of my machine, there was a non-negligible tilt angle:



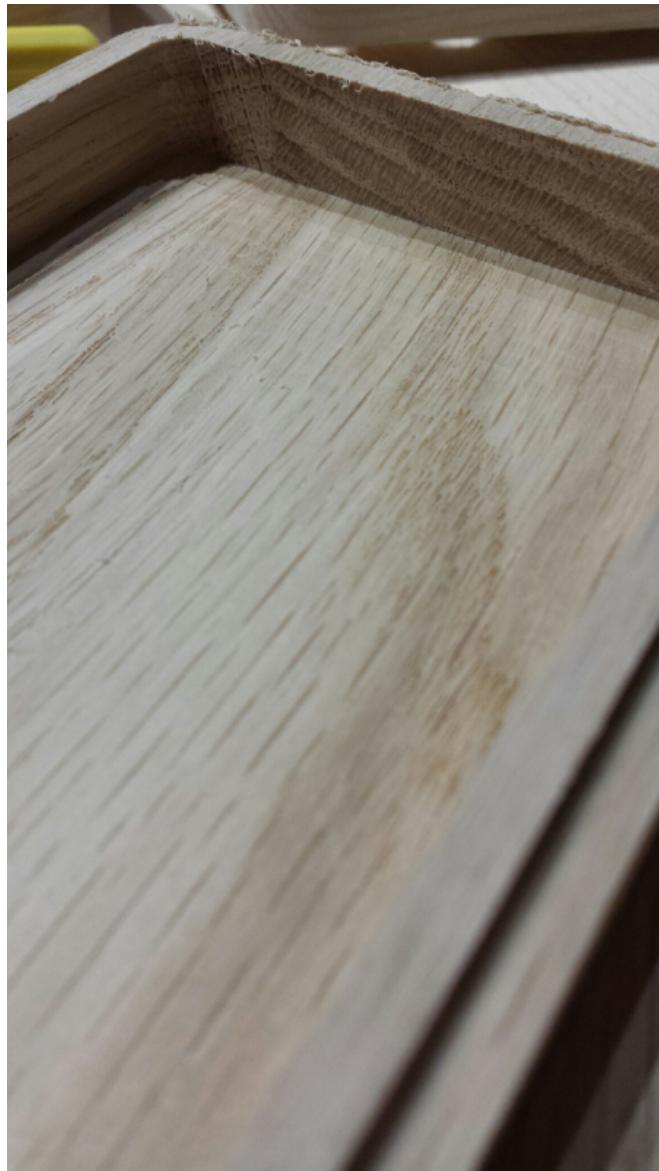
There are (at least) two popular ways to fix the front/back tilt:

- loosen the 8 screws on the side plates of the X extrusion, then **rotate the whole gantry** around its axis, towards the back or front, and re-tighten the screws. Again, an iterative process, because tightening the screws may make the gantry rotate very slightly. You may want to remove the powder coat inside the screw holes, to get a little wiggle room.
- **shim the router mount.** A thin strip of aluminium folder over a few times, inserted between the bottom (or top) of the mount and the Z-plate, can correct the fractions of a degree of front/back tilt.

Here is the same machine after front/back tramping:



Not perfect, but much better. Here is a re-run of the same pocket operation as earlier, after trammimg. The ridges are not completely gone, but are much less visible:



- (i)** Since tramming modifies the Z axis angle slightly, it is advised to re-surface the wasteboard afterwards, to get it as flat as possible.

Dimensional accuracy

For "decorative" projects, some inaccuracy in the dimensions of pieces cut on the Shapeoko does not matter, but many projects require having a finer control of the final dimensions:

- cutting geometrical patterns where a discrepancy between the X and Y dimensions could be visible to the naked eye, *e.g.* round shapes coming out like ellipses and squares like rectangles. Out of the box, on a properly assembled/squared/tensioned Shapeoko, this X/Y dimensional discrepancy should be small enough to not be noticeable.
- cutting pieces that are intended to fit into each other (*e.g.* hole & peg, box & lid, ...). This is where one typically starts noticing that dimensions are not *quite* what they were supposed to be, and are inaccurate enough that pieces either do not fit, or require manual rework (*e.g.* sanding) to be assembled.
- cutting pieces that are supposed to fit, from a hard material (hard plastics and metals). This is where the accuracy problem becomes obvious, as there is no elasticity in the material to compensate for small inaccuracies.
- and then obviously, projects where the whole point is to cut pieces of specific dimensions within a controlled tolerance (*e.g.* mechanical parts)

Many factors are involved in getting good dimensional accuracy:

- intrinsic mechanical accuracy of the machine (that may drift over time, too)
- *actual* dimensions of the tool being used
- rigidity of the workholding solution
- dynamic effects during the cut, mainly tool deflection, depending on the tool, material, toolpaths, feeds & speeds.

If the Shapeoko forum is any indication, many people (including myself) have gone through these phases:

- 1) starting to care (then obsess) about tweaking/calibrating/tuning the machine to improve dimensional accuracy
- 2) chasing their own tail trying to squeeze these last few % of additional precision

3) realizing that there are so many factors involved, that a much more efficient approach is to just cut a piece, measure it, adjust the CAD and/or CAM to compensate, rinse and repeat until the result is perfect.

Yet, I still think it is important (or at least interesting) to know about sources of inaccuracy to decide what to do (or not to do) about each of them. This section will only scratch the surface on this topic, but hopefully provide useful insights on the matter.

- (i) The accuracy that is possible to achieve on a Shapeoko with the proper mix of calibration and CAD & CAM iterations, seems to be typically around 0.002" / 0.05mm, with some achieving down to 0.0002" / 0.005mm

X/Y/Z calibration

The Shapeoko belts and pulleys are such that it takes (very close to) 40 motor steps to move by 1 mm. But the tensioning of the belts induces some stretch, such that in reality when the stepper motor does 40 steps, the actual resulting movement on the associated axis will be slightly more or slightly less than 1mm.

The GRBL software in the Shapeoko controller stores a "steps per mm" adjustable parameter for each axis, and uses it to convert a G-code command that requests a displacement expressed in distance units (inches or mm), to a number of steps to be generated on the stepper motor.

These parameters are stored as \$100, \$101, and \$102 for X, Y and Z axis respectively.

By measuring by how much a given machine *actually* moves for a given commanded distance, one can adjust these parameters, i.e. calibrate/compensate for belt stretch.

The basic principle is to:

- make a note of the current value of \$100, \$101, and \$102 parameters.
- design a piece of given dimensions in your CAD tool.

- cut the piece on the machine, and measure its actual X and Y and Z dimensions with a caliper.
- adjust the \$100, \$101, and \$102 values accordingly:

$$\text{new parameter value} = \frac{\text{expected dimension}}{\text{actual dimension}} \times \text{old parameter value}$$

- cut the piece again: it should now be very close to the expected dimensions

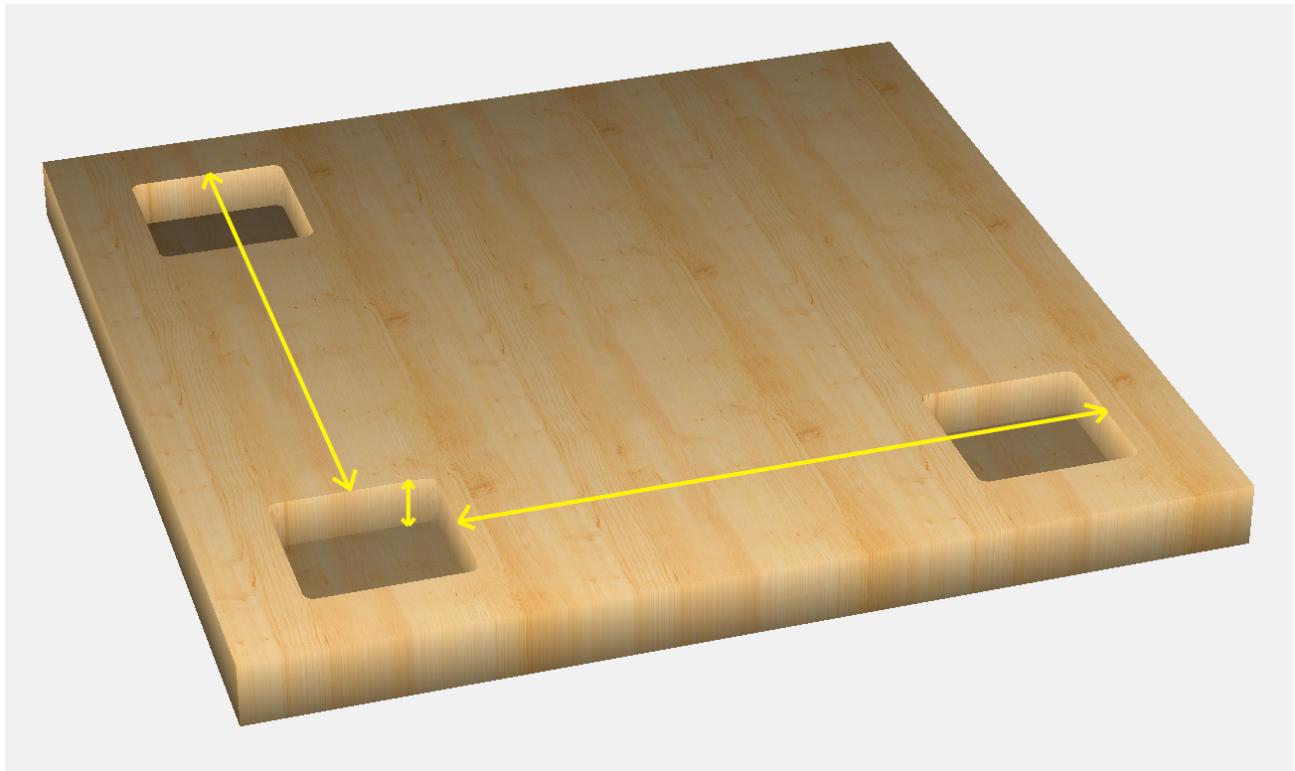
In theory, cutting a cube would be ideal to support the XYZ measurements. In practice, the Z measurement is much easier to do inside a pocket, while for the X and Y dimensions there is a catch: they include at least two sources of errors that are actually not coming from the machine:

- **tool deflection** during the cut: this is influenced by the tool diameter and stickout, the material, the feeds & speeds, the DOC and WOC. If tool deflection is significant, the piece will be larger or smaller than expected due to this, not due to belt stretch.
- **tool diameter uncertainty and tool runout**: the endmill used to cut the test piece is never *exactly* the diameter it is supposed to be, and there is probably a little runout too, but the CAM tool generated a toolpath based on the theoretical value of the tool diameter and zero runout, so the piece dimension will be slightly off due to this too.

These errors should not be included/compensated for in the X/Y calibration, since they will change when a different set of conditions are used (different endmill, toolpaths,...)

A simple way to address these constraints is to:

- cut three deep pockets (two aligned along X, two aligned along Y)
- measure the border to border distances, and the pocket depth:



The border-to-border measurement for X and Y will cancel out the endmill diameter/runout/deflection errors from the measurement, and the pocket depth can also easily be measured with a caliper to adjust \$102.

The downside of this method is that measuring border-to-border distance with a ruler will not be extremely accurate, but if the distance between pockets is large enough, it should still grant good results.

Another popular method is to rather measure the gantry movements themselves, instead of measuring the dimensions of the resulting cuts. One can use:

- **a dial indicator**
 - attached to the bed of the machine and oriented along X then Y axis, to measure how much the gantry moves versus what was commanded.
 - attached *e.g.* to the router and oriented along the Z axis, to measure how much the Z plate moves versus what was commanded.



- a (long) **caliper**, oriented along X then Y axis, with the fixed part attached to the bed, then pushing the moving part with the shank of the endmill
- a **DRO** (digital readout scale) can be used to do the same, over a longer distance to minimize the error.

i The axis of the measurement tool should be aligned carefully to match the axis of the machine, any angle between the two would reduce the accuracy of the measurement.

Limitations of belt calibration

The pitfall of belt calibration is expecting to get excellent accuracy out of it alone. Unfortunately, belt stretch is not fully linear, and many users found out that the calibration values varies depending on the calibration distance, and where on the work area the measurement is made.

It could even be considered to be counter-productive, because of the finite precision of GRBL computations that will round numbers with lots of decimal places, and the rounding effect may be worse with calibrated values than with the default 40steps/mm value.

As mentioned earlier, another approach to dimensional accuracy is simply to do test cuts, measure the result, adjust the design and toolpaths, and iterate. In this case, leaving the steps per mm at their default value is arguably a good option.

I chose a middle ground: calibrate X/Y/Z to be in the right ballpark on the first try (which is enough for many projects), and then do a second round after compensating the error in the design/toolpaths when this is required for a given project.

- (i) The "stock to leave" option in Fusion360 is a great way to deal with this: you can initially use a positive stock to leave value to have margin, run the job and measure, then adjust stock to leave accordingly, relaunch the cut, and measure again. You can even specify negative stock to leave values, to cut beyond the profiles defined in the design (e.g. to cut a hole slightly larger than it was designed to be). All of this assuming of course that you will not cutout/remove the piece from the stock before you are done with these iterative cuts.

Managing tool deflection

Beyond being bad for tool life (and quality of the cut in general), tool deflection is bad for dimensional accuracy. To minimize it, and depending on the situation, one can:

- use a larger (=stiffer) endmill.
- use an endmill with large shank diameter (stiffer, again).
- use an endmill with a long shank and short cutting part, if depth of cut allows.
- minimize tool stickout (when possible)
- use conventional milling cut direction
- minimize lateral forces on the tool, which usually means taking a lighter cut (e.g. small DOC and or WOC).

This last point is best addressed by the "**roughing + finishing**" toolpath strategy: the roughing toolpath can take aggressive cuts, leaving enough margin (stock to leave) that deflection effects will be included in this margin. And then the finishing toolpath will only have to

shave off a thin layer of material, which takes very little force and therefore causes very little deflection.

For smaller endmills, deflection becomes a critical issue that can easily lead to breaking tools, a deflection-aware feeds & speeds calculator is a good way to check before cutting.



HSS endmills are more flexible than carbide endmills, so they will withstand a greater amount of deflection before breaking.

Managing runout

Why bother?

Runout was defined in the [Cutters & collets](#) section. Since the toolpaths in the CAM tool are based on the endmill diameter, and since runout artificially increases the effective cutting diameter, the parts will have a dimensional error of (at least) the runout value.

Runout matters a lot for projects using small endmills, i.e. anything smaller than 1/8". The reason is that the endmill will see +/- 1 runout of variation on the chipload. Since runout can be of the same order of magnitude as the target chipload, a small endmill seeing a chipload twice as high as the optimal target value can easily break.

Runout is bad anyway for tool life, chatter, and surface finish.

Measuring runout

The simplest way to measure runout is to do a test cut of a slot, measure the actual width of the resulting slot, then subtract the actual endmill diameter: the difference is the runout value. Unfortunately, it is not straightforward to determine precisely the actual diameter of a given endmill: they are manufactured within given tolerances, and are never quite the diameter they are sold to be. A 1/4" endmill could typically have an actual diameter anywhere between 0.245" and 0.255". Measuring precisely the actual diameter is not easy,

as the measurement must be made on the flutes, and without proper tooling that can be difficult especially for endmill with an odd number of flutes. There is also a real risk of damaging (chipping) the flutes during this measurement.

Another way to measure runout is to use a **test indicator**, capturing the minimum and maximum values read over a 360° revolution of the endmill. Ideally, the runout should be measured at the tip of the endmill, where the cutting action is, but that means measuring on the flutes themselves. Alternatively, the dial indicator can be positioned on the shaft, as close as possible to the flutes, this will give a continuous readout of the runout over a 360° revolution, but could under-estimate the actual runout a bit.

Runout is the difference between the maximal and minimal values read, at a given measurement point.

TIR (Total Indicated Runout) is the range of runout values if we were to measure across the full length of the endmill, so it boils down to the max runout.

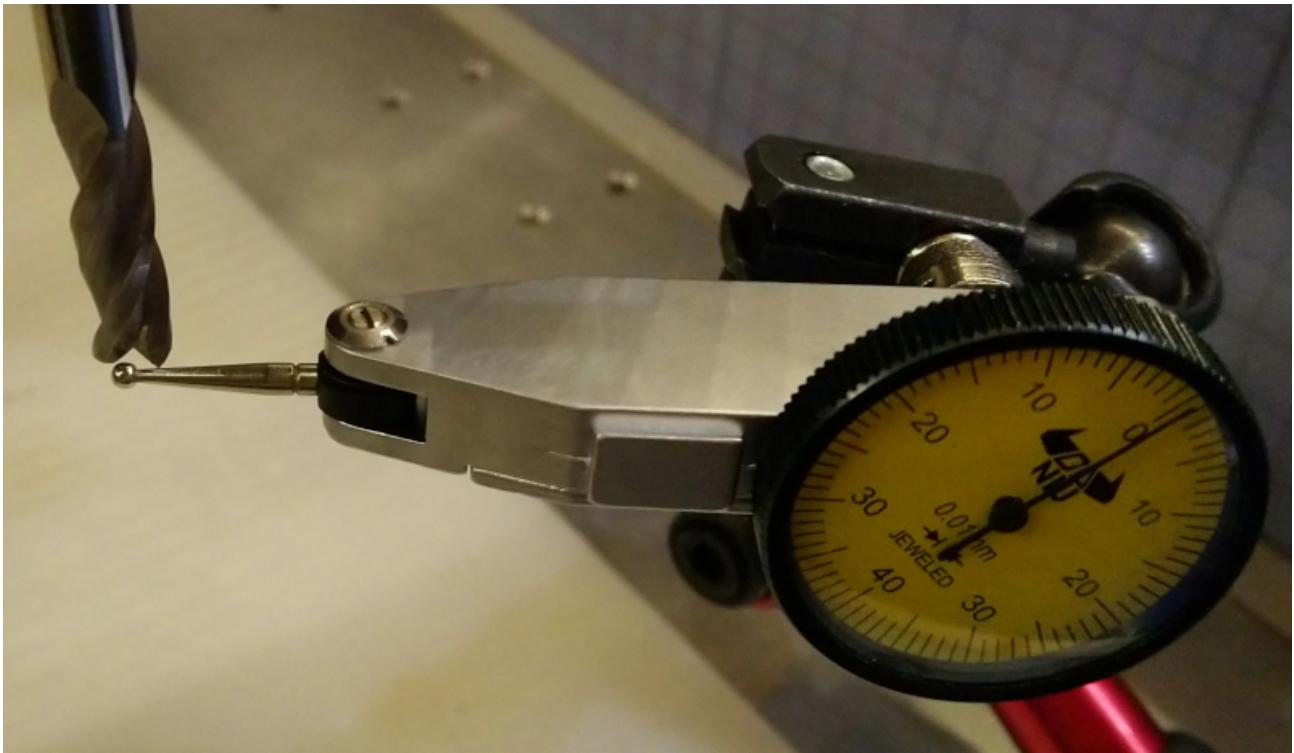
Runout measured on the endmill is the result of the combination of the router runout, collet runout, and endmill runout.

Router runout itself can be measured by positioning the test indicator inside the collet taper:

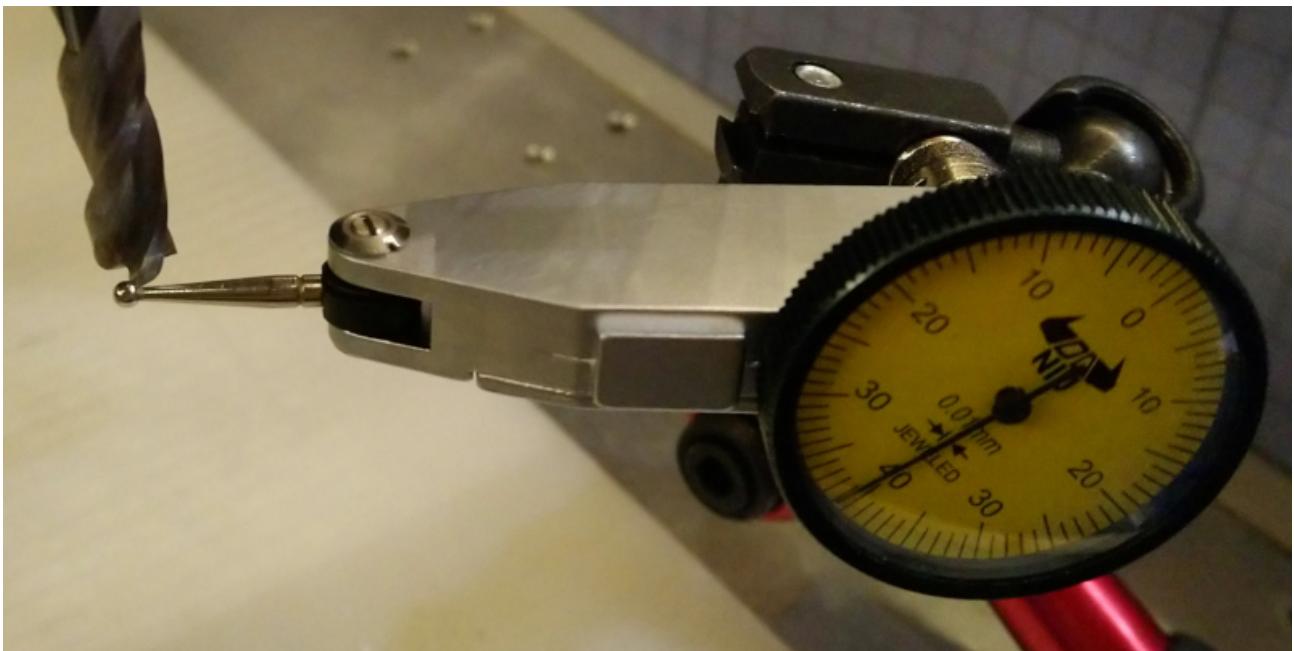


In this case, the measured taper runout was approximately 0.02mm (0.0008").

But what really matters is total runout at the tip of the endmill. Below is an illustration of measuring runout at the tip of a 3-flute 1/4" square endmill. First, the test indicator is positioned at a point where no flutes touch it, and zero is set there:



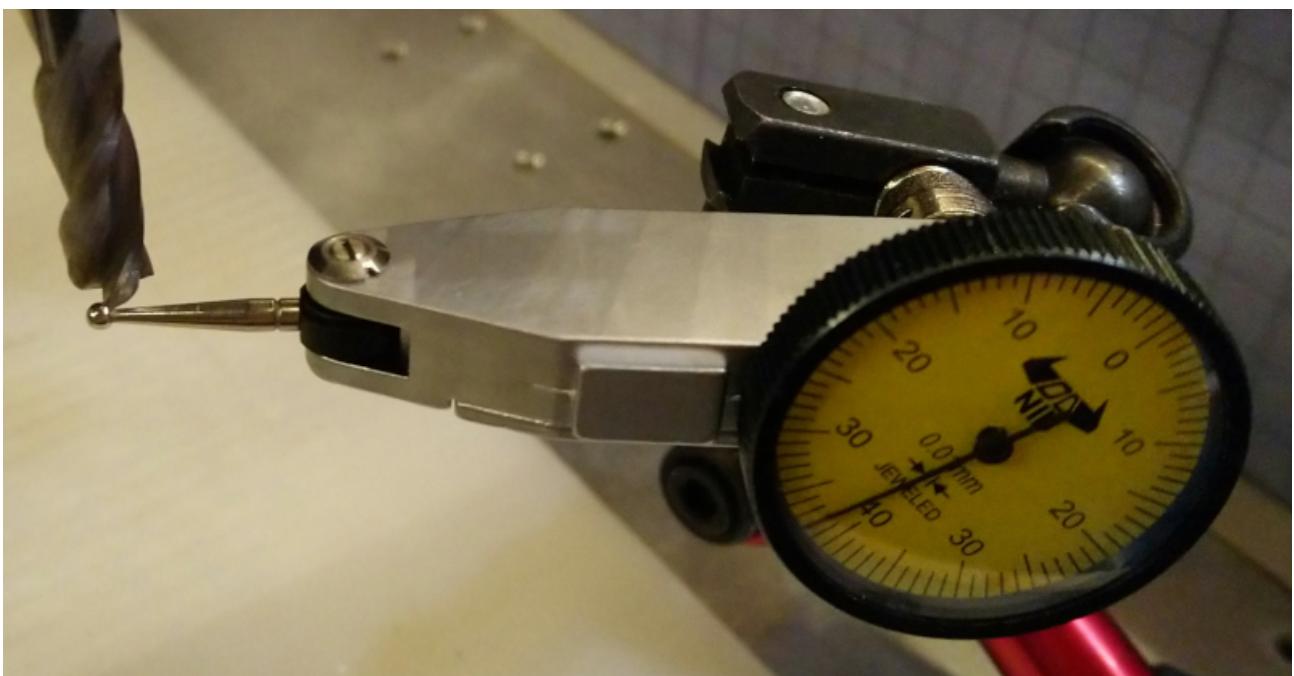
then the endmill is rotated manually until the test indicator readout maxes out at the tip of the first flute. In this case it read 0.41mm:



the endmill is then rotated further and the deviation at the tip of the next flute is measured, in this case 0.37mm:

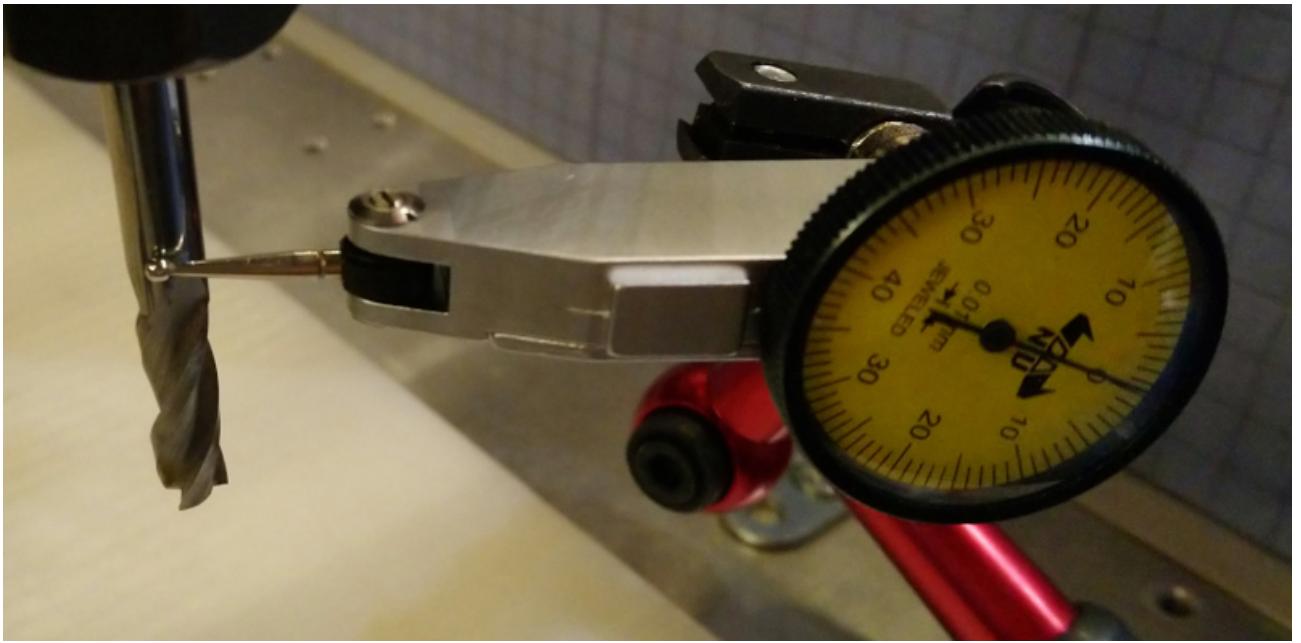


ditto for the last flute, which read 0.42mm in this setup:



So the max difference is $0.42 - 0.37 = \mathbf{0.05\text{mm runout}}$

As a comparison, measuring runout on the shaft after setting the indicator zero to the point giving the minimal readout,



gave a max value of about 0.045mm:



Not too far from the 0.05mm measured on the flutes, and easier to measure there.

This example value of 0.05mm (0.002") runout on a Makita router with the stock Makita collet and a 1/4" endmill is not great, but not really a concern in day to day use. It will vary significantly depending on which collet and endmill are used and how they were fastened. Another combination of collet/endmill on this same router granted a runout of only 0.02mm (~0.0008").

Reducing runout

- there is nothing one can do about the intrinsic runout of a specific router (short of returning the router to get a new one, that *could* have a lower runout, but that is random)
 - **Spindles** usually have lower runout than routers, so upgrading to a spindle can be another approach (runout alone may not be a good enough reason to upgrade though)
- using **precision collets** is a simple way to help minimizing runout, and is one of the cheapest "upgrades" to the stock Shapeoko setup.
- using quality endmills helps, but the endmill is probably not the main source of runout anyway.
- anything which comes between the bore and the collet will prevent correct alignment and cause/increase runout: always make sure the taper and collet are **clean** when inserting a new tool.
- minimize **tool stickout**: this will minimize the effect of the axial runout.
- considering that the final runout results from the relative positioning of the router, collet, and endmill, one can try to "**clock**" these three elements, so that the different runout contributions cancel each other to some extent. There is no easy way to determine runout of each element separately, so this can be a trial & error process, but worth trying: slightly turn the collet inside the router taper, and/or turn the endmill inside the collet, and this *might* yield a lower overall runout.
- finally, a method that looks scary but gives surprisingly good results is **tapping** (gently!) on the endmill shaft using a screwdriver (or ideally a brass drift punch) and a mallet:
 - first find the orientation of the endmill that gives the maximum readout on a dial indicator placed on the front side.
 - if using a screwdriver put a small piece of electrical tape on its tip, to avoid metal to metal contact.
 - then...*lightly* tap the back of the screwdriver with a mallet a couple of times.
 - check runout and repeat if necessary. It takes a few tries to find the right amount of force. In this example, the original runout was around 0.002", and tapping the endmill reduced it to around 0.0006". This adjustment should hold, at least until the end of the current job with this endmill.



Managing residual runout

Even if it can be minimized to some extent, some runout will always exist. A simple way to deal with it is to make the CAM tool believe that you are using a slightly larger endmill, corresponding to the actual endmill diameter + runout, measure this **effective cutting diameter** on a scrap piece of material, and then use that effective diameter in the CAM tool instead of the theoretical diameter.

This won't do anything about the negative aspects of runout such as possible vibrations/chatter/poor surface finish, but at least it should give more accurate dimensions.

Multi-tool jobs

When running jobs involving tool changes, an additional constraint impacting dimensional accuracy appears: position errors from readjusting the zero reference each time a tool change is required (usually, resetting Z0 to compensate for the tool length difference).

There are at least two things you can do to minimize those errors:

- do not turn-off/re-home the machine between tool changes if you can avoid it. While homing establishes a known absolute position, it comes with a finite repeatability performance; upon returning to zero after homing, you may notice that it is not PERFECTLY in the same location as before.

 You might argue that homing provides the benefit of clearing the effect of lost/skipped motor steps. But if you lose steps in the first place, there is no chance that dimensional accuracy will be good, so you should address that first (using less aggressive settings)

- use a [tool length probe](#) to automate the adjustment of the Z0 between runs. Sure, a tool length probe comes with its own accuracy/repeatability limitations, but chances are that it will still provide a more repeatable result than manual re-zeroing. And once things are repeatable, it's easy to adjust the CAM design to compensate for residual dimensional errors.

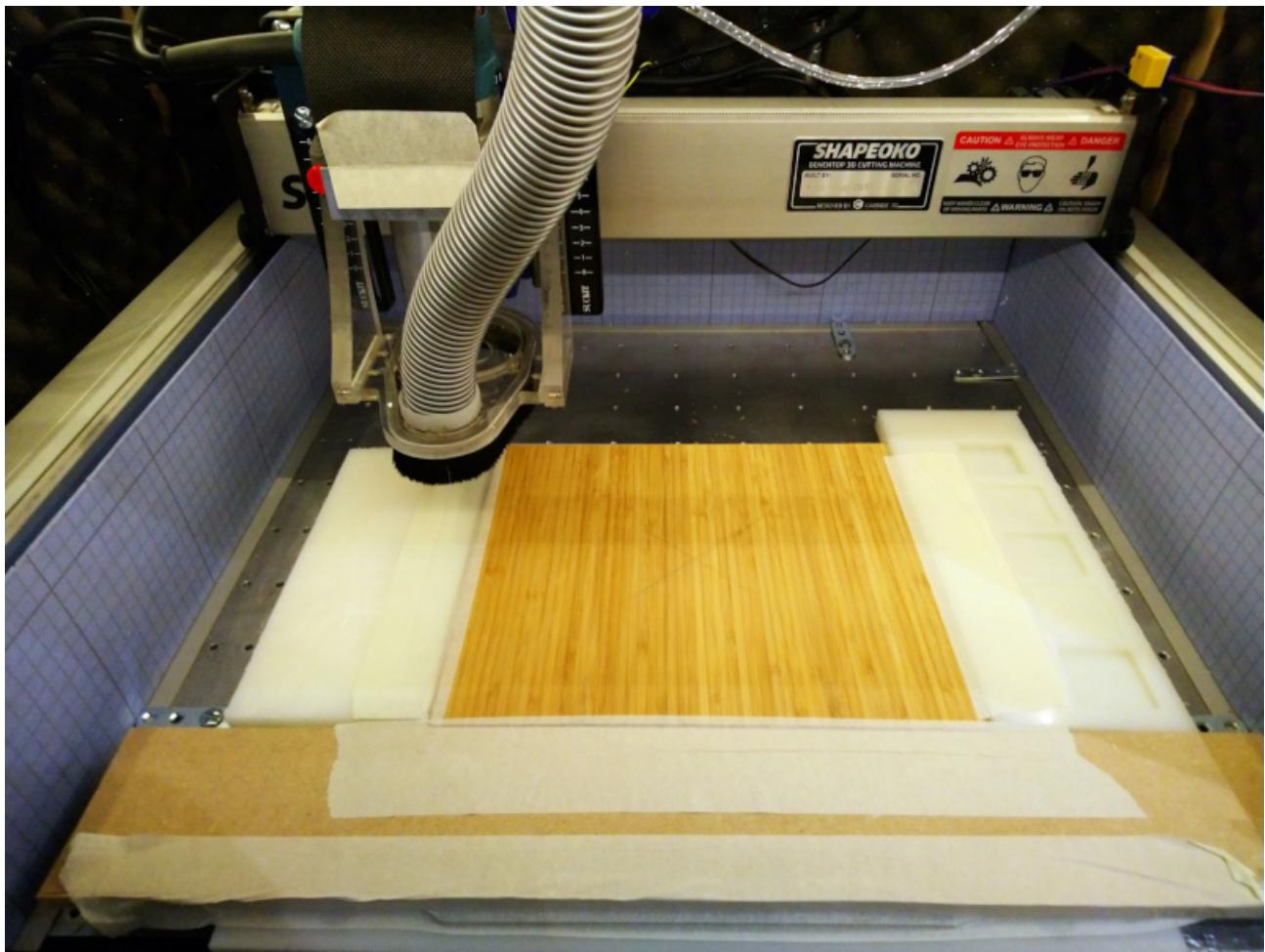
Usecases: cutting wood

These usecases are intended to illustrate how the information from the [Feeds & speeds](#) section can be used to determine workable cutting parameters, for a project that uses wood as the stock material. Wood is quite forgiving, in the sense that the range of acceptable feeds and speeds is much larger than with e.g. plastics or metal. So this is by no means a recommendation about specific values to use for wood, but rather an example of how to make informed decisions about the types of endmills, feeds and speeds, and toolpaths for cutting wood.

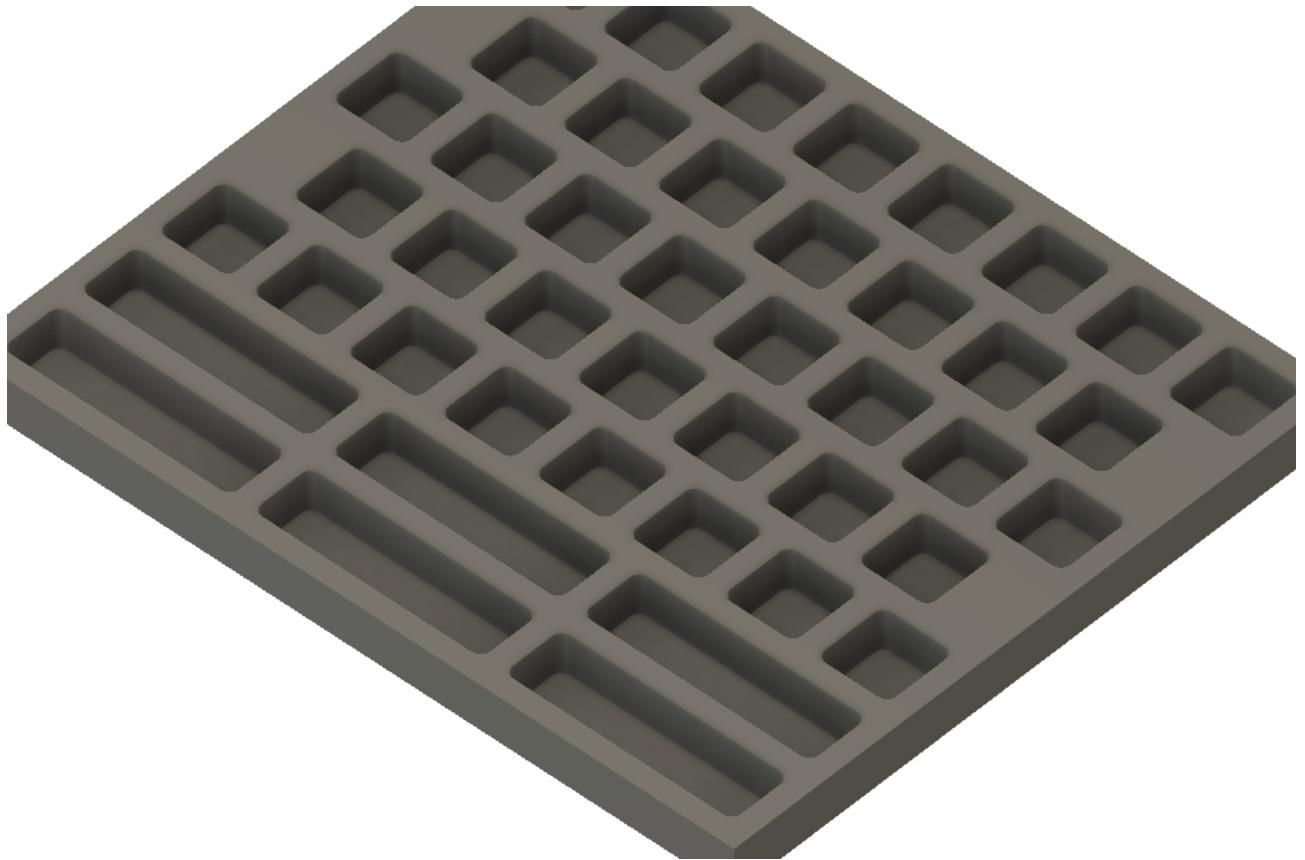
Bamboo tool holder

Ok, technically bamboo is not wood, but close enough. In this example, I wanted to make a holder for my endmills from a cheap (Ikea) 11" × 18" bamboo cutting board. I used the tape & glue method to secure it to the wasteboard.

-  I added extra pieces of similar thickness on the sides and front, this allows the dust shoe to not lose suction when it moves past the edge of the stock during the toolpath.



The design is very straightforward but has a lot of deep pockets, with tight corners that a 1/4" endmill could not cut. And cutting of all this with a 1/8" endmill would take forever:



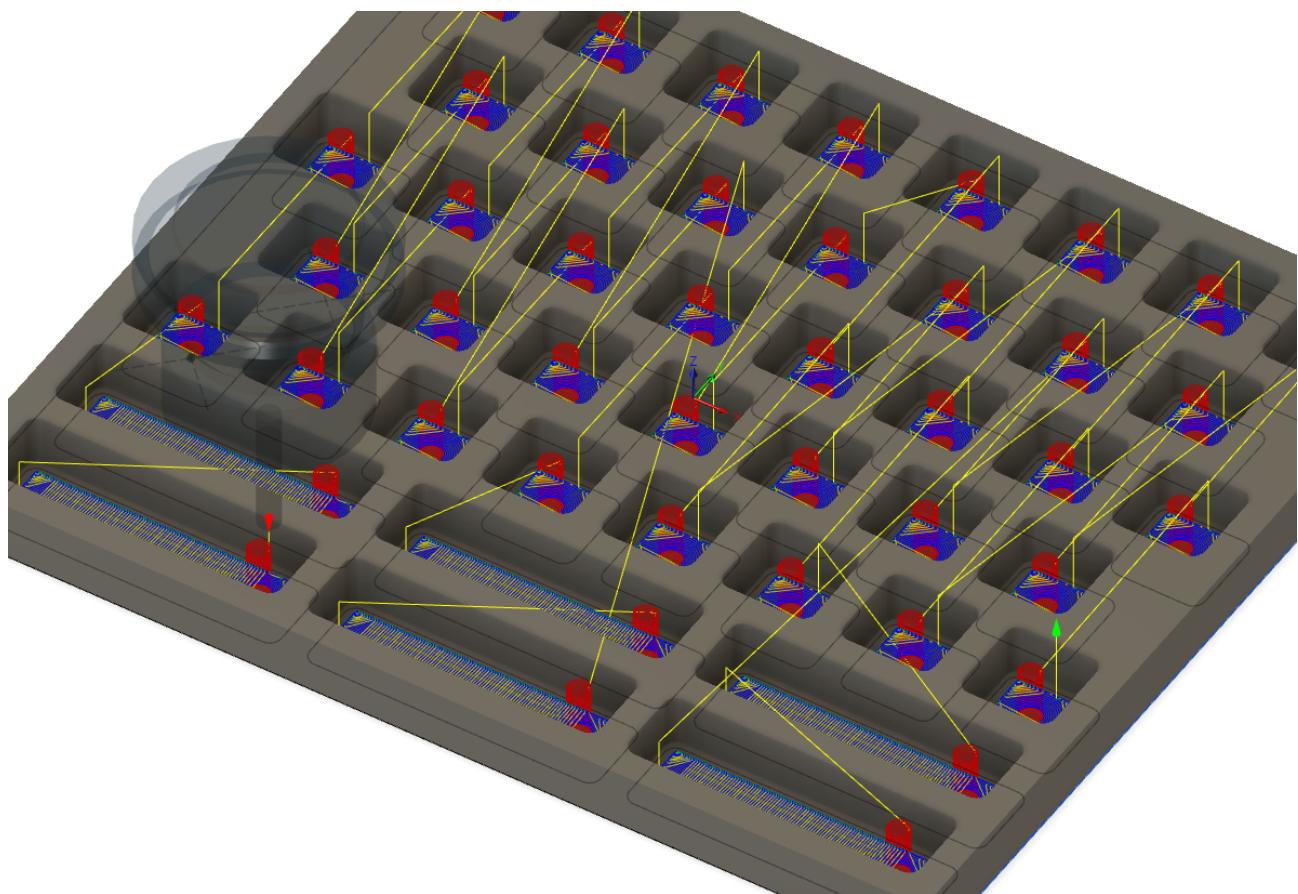
So I first created a roughing toolpath using a large (6mm) endmill, to do most of the clearing.

- an **upcut** endmill is used, to have good chip evacuation and since there will be a finishing pass afterwards anyway to clean the rough top edges of the pockets.
- Using the **target chipload guideline** (see [Feeds & speeds](#)), and considering bamboo is somewhat easier to cut than "hard wood", I picked a target chipload value of 0.002"/0.055mm in the high-end of the range.
- While regular pocketing toolpaths would have worked fine, I chose to use an **adaptive clearing** toolpath, and with a stepover/radial width of cut/optimal load of 0.0315" / 0.8mm (13% of endmill diameter, a very conservative value for wood)
- After taking **chip thinning** into account for a 13% stepover, the corrected target chipload is 0.0327" / 0.083mm

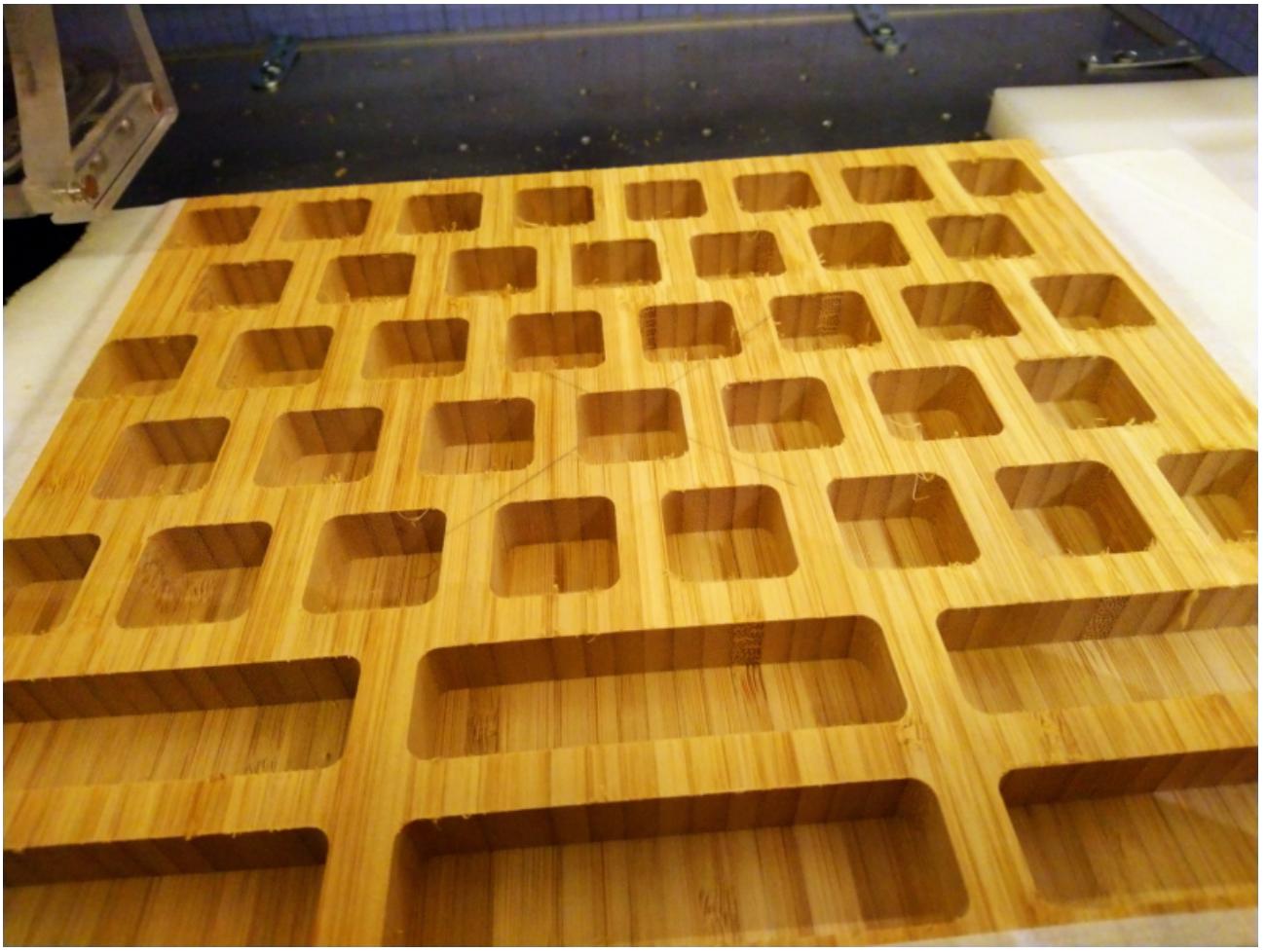
$$Chipload(adjusted) = \frac{6}{2 \times \sqrt{(6 \times 0.8) - (0.8)^2}} \times 0.055 = 0.081mm = 0.00319"$$

- Since my endmill has 2 flutes, and selecting 12,000 RPM as my speed, the required **feedrate** to reach that chipload value is $0.00319 \times 2 \times 12000 = 76.6\text{ipm} = 1945 \text{ mm/min}$
- Since the stepover is very low and the adaptive toolpath enabled it, I went for cutting the **full pocket depth** ($0.5" = 12.7\text{mm} = 200\%$ endmill diameter) in one pass.
- I chose 400 mm/min for **plunge rate**, which is very conservative for wood especially considering I also used helical ramping into the material.
- I left 0.5mm radial **stock to leave**, that the finishing pass would take care of.

A preview of the toolpath gave me this:



and the cut proceeded uneventfully, taking about 40 minutes. Notice the fuzzies on the pocket edges though, due to the use of the upcut endmill:

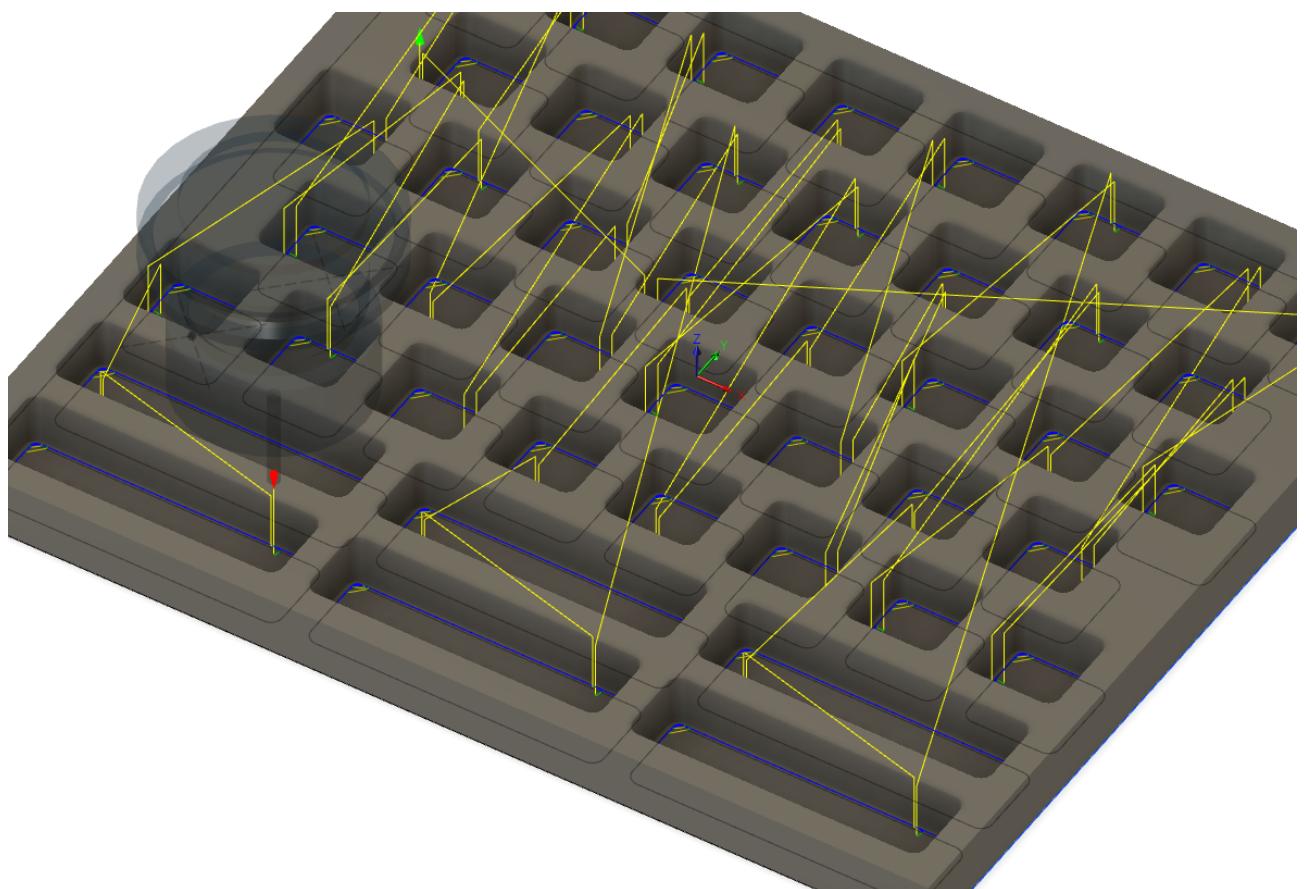


I then created a finishing pass that would do two things: take care of cutting the corners to their final radius using a smaller (1/8") endmill, and remove the 0.5mm remaining stock from the pocket walls.

- I used a **downcut** endmill, to get clean top edges. The poor chip evacuation is not an issue here since there is very little material to remove and it happens inside a large pocket with lots of space around the endmill.
- I picked a radial width of cut of 0.02" / 0.55mm, i.e. just a bit more than the stock to leave from the previous pass, so that it will only take one pass on the walls (and just a few passes in the corners). That is only 17% of the endmill diameter anyway.
- Then to both cut the corners and remove the stock left from the previous pass, I selected the same geometry but used the **rest machining** option: the CAM tool is aware of the material already cut during the first pass, and will generate a toolpath that cuts the rest.
- The target chipload from the guideline for an 1/8" endmill in hard wood is up to 0.001"/0.025mm, and since bamboo is somewhat softer and that I would only be cutting a thin layer of material, I aimed at a value a little higher, 0.0014" / 0.035mm.

- Adjusted for chip thinning with 0.02"/0.55mm stepover, that's a 0.0019"/0.046mm chipload/
- Considering the 2 flutes on this endmill and 12,000RPM, that requires a feedrate of 43ipm = 1100mm/min
- Still using a full pocket depth cut.
- Still using a plunge rate of 400mm/min.

Thanks to the magic of rest machining, this finishing pass is quick (7 minutes):

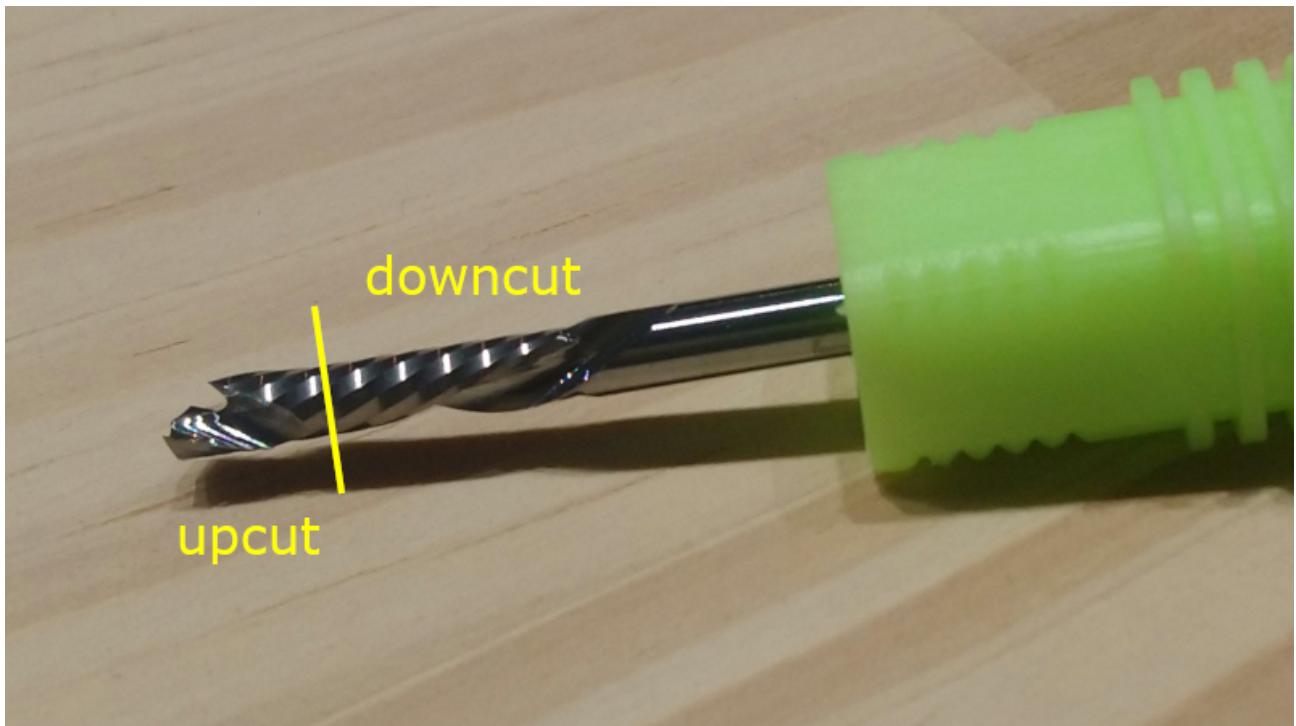


Which gave me this result, right out of the machine, with perfect edges and no need for any clean-up/sanding:

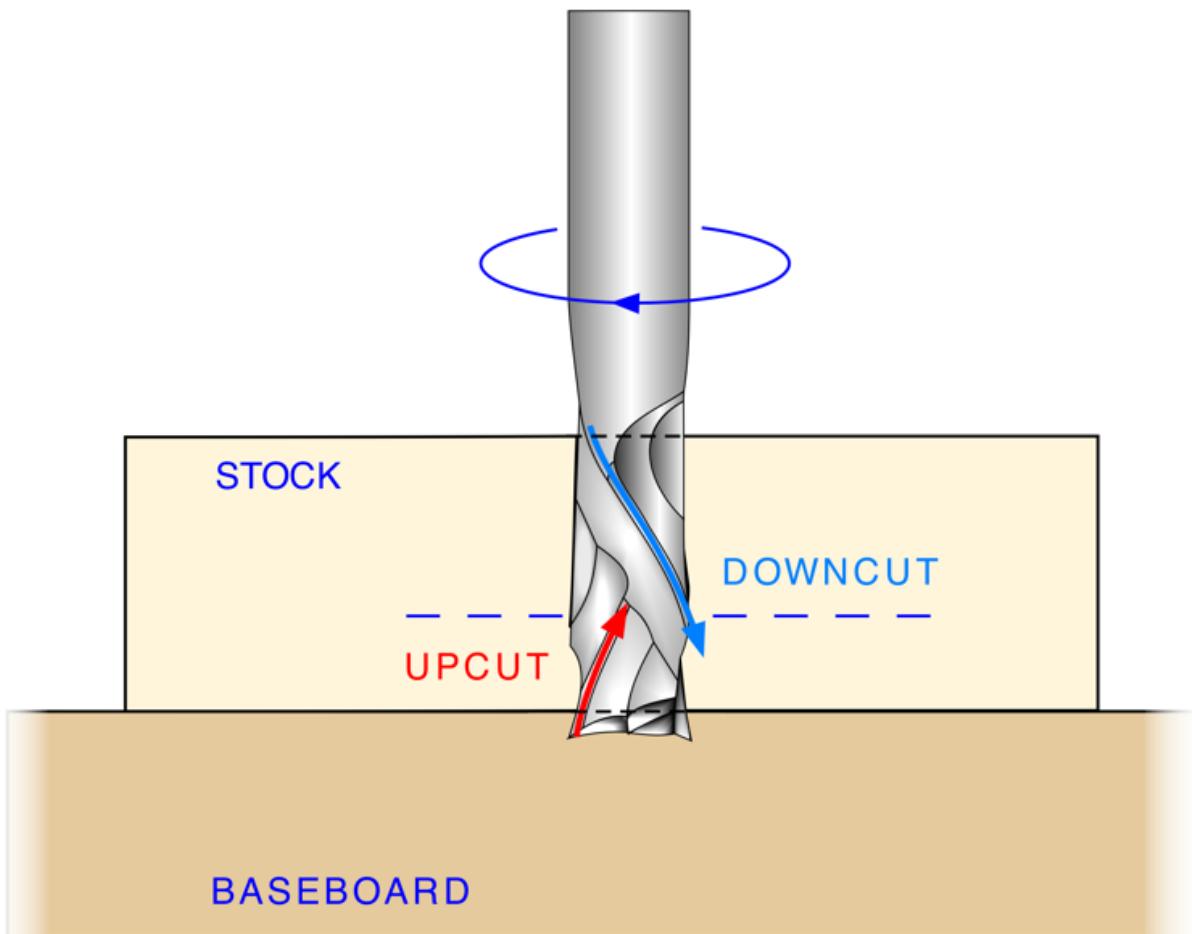


Plywood parts

A much simpler example now: doing profile cuts in plywood. Plywood is soft and easy to cut, but the top and bottom veneers are susceptible to tearout. A common solution to avoid tearout is to use a **downcut** endmill, which will leave clean edges on the top surface. However, the risk of tearout is then moved to the underside of the part. That's where **compression endmills** come in handy: they have both an upcut section at the tip, and a downcut section higher up the flutes:

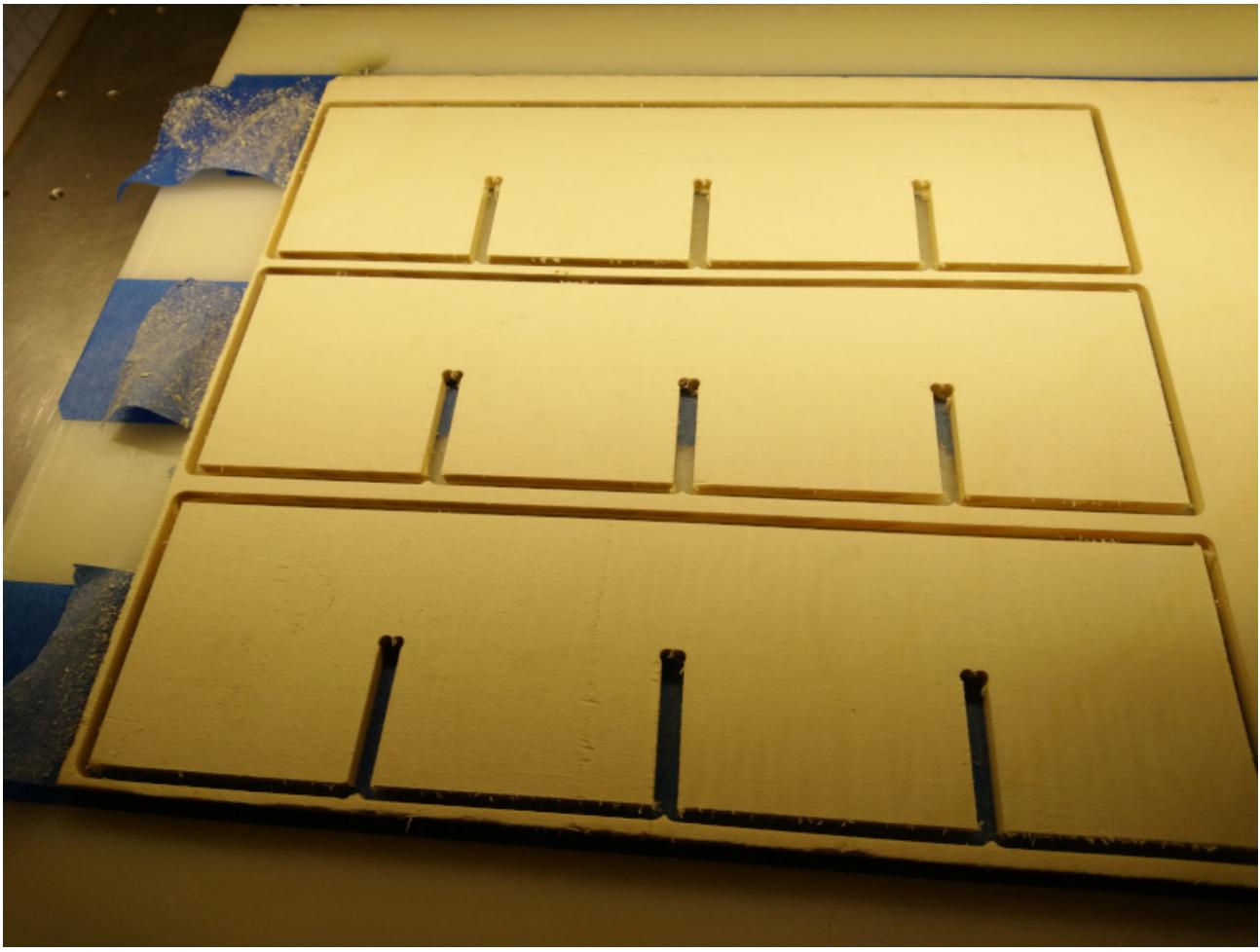


To leverage this peculiar geometry, the endmill is intended to be used for cutting through a stock material which thickness matches the distance between the upcut and downcut parts of the endmill:



The downcut section of the flute will tend to push down on the stock material top surface, while the upcut section will tend to push up on the bottom surface (hence the name "compression"). The expected result is that both the top and the bottom surfaces should have no tearout.

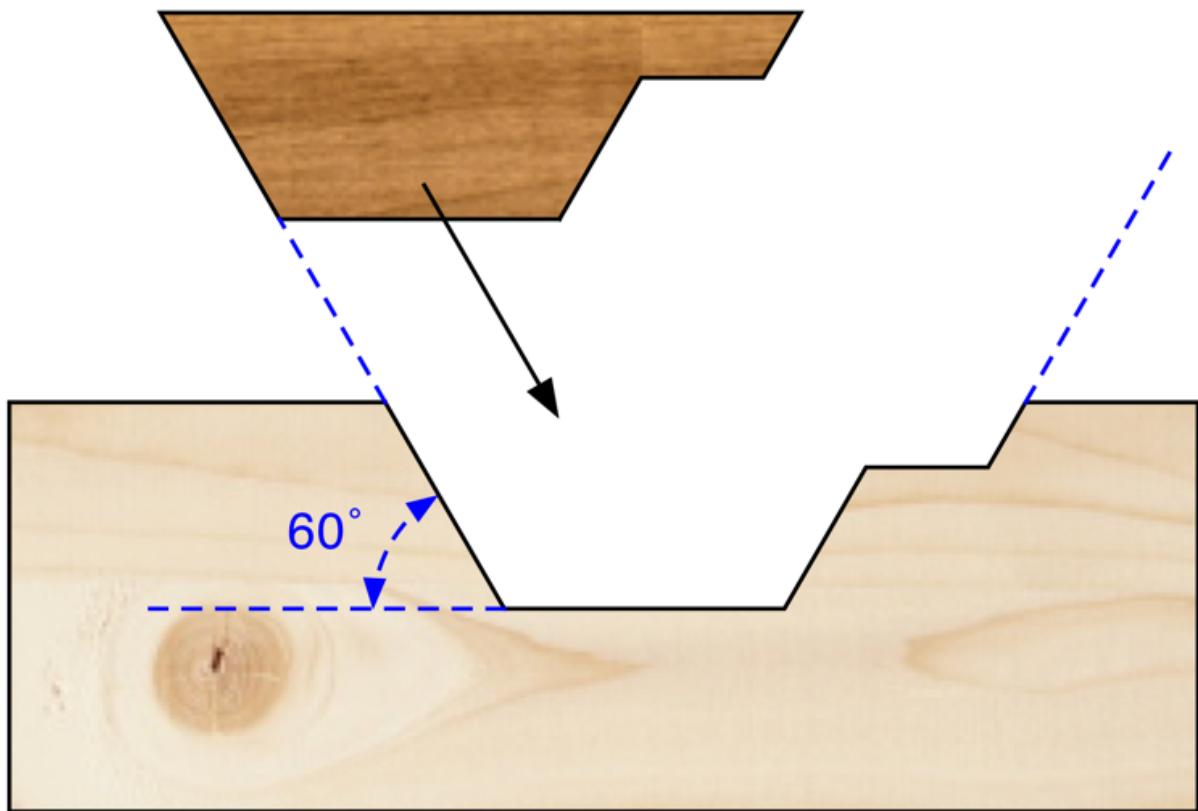
Here is the result of trying the 1/8" O-flute compression endmill shown above, for profile cuts in 0.2" plywood. Feeds and speeds were set to 20,000RPM, 40ipm feedrate (20ipm plunge rate), 0.2" depth of cut:



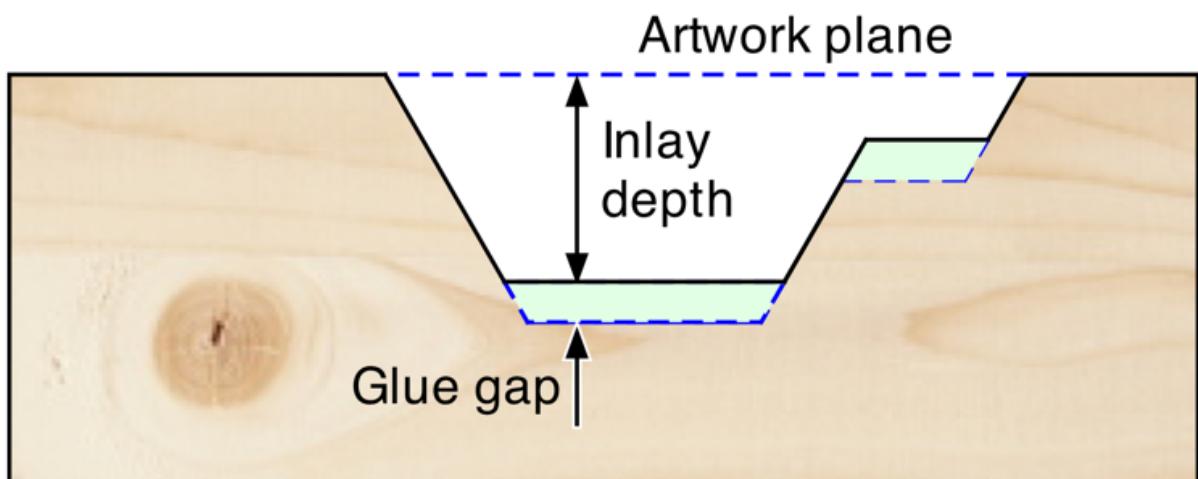
V-carved inlays

(i) This is basically my take on explaining v-carved inlays, based on this great video: <https://www.youtube.com/watch?v=l4VMo9DCzO8>

Making wood inlays is a very popular technique, that is surprisingly easy once you understand the underlying principles. Like regular V-carving, it allows one to mill features that would otherwise be impossible to do with straight endmills, like stars or other pointy objects. It starts from the basic observation that if one uses a V-bit of a given taper angle, it is possible to cut a cavity AND the associated part filling that cavity, with the same tool, for a perfect match. Here's a basic example with a 60° V-bit:



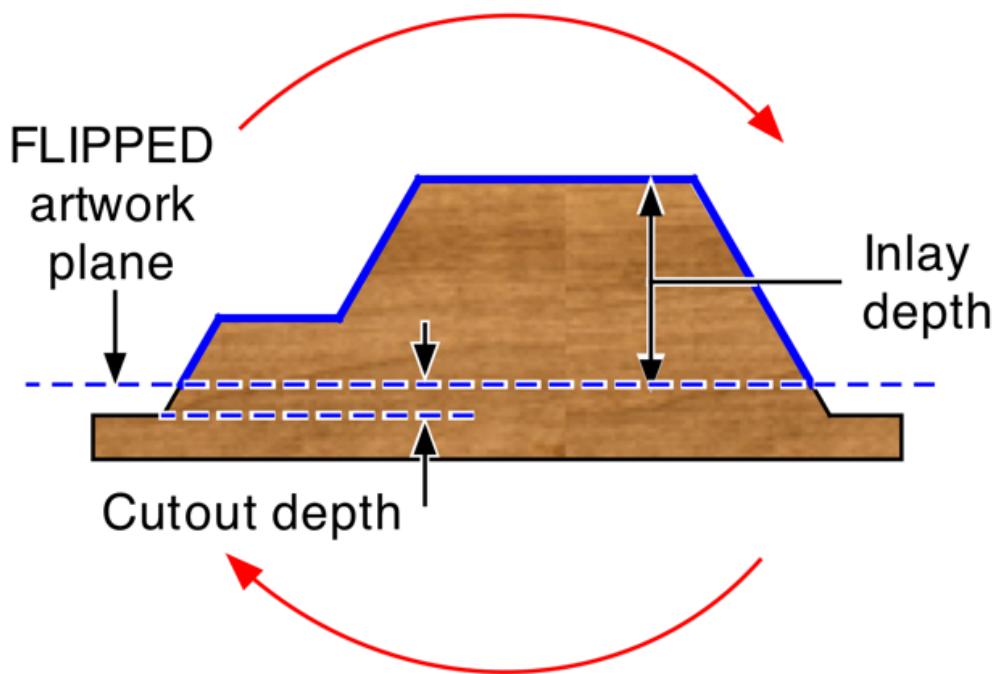
To cut the **base**, one uses a regular v-carve toolpath, based on any 2D shapes (as a convention these reference shapes constitute the "artwork plane"). The toolpath only needs to cut down to a depth corresponding to the desired inlay thickness, however it is more practical to carve a bit deeper, to have a little bit of space for the glue that will hold the base and the inlay, and to cope for small depths errors:



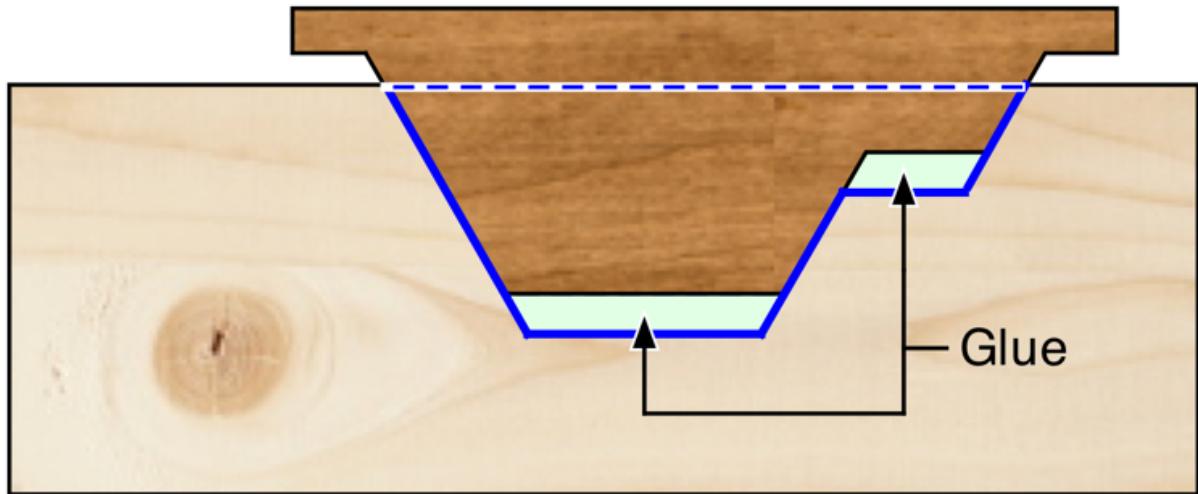
i This is the beauty of v-carved inlays: the slope introduced by the use of a V-bit, the extra depth margin (glue gap), and making the inlay a bit thicker than required, will ensure that even if the inlay depth is not a perfect match for the base, it will still fit perfectly, and one will only have to remove the excess material after glueing.

Now, cutting the **inlay** part itself requires a bit more caution:

- the very first thing to notice is that the artwork needs to be **flipped** in the left/right direction (i.e. to apply a 180° rotation along the Y axis, and have the inlay part facing "up" for the v-carve toolpath). Unless the v-carved feature is perfectly symmetrical, this is essential or the inlay will not fit.
- the second thing is that one needs to cut the inlay to be a bit thicker than strictly required, to allow for thickness errors and to leave some space between the base and the top of the inlay for cutting the excess material after assembly. This corresponds to the extra "**cutout depth**" on the figure below.
- finally, it is often convenient to add a solid **base underneath the inlay** part(s), especially when the artwork is such that many individual inlay pieces will be cut: the base will hold them all together during assembly.



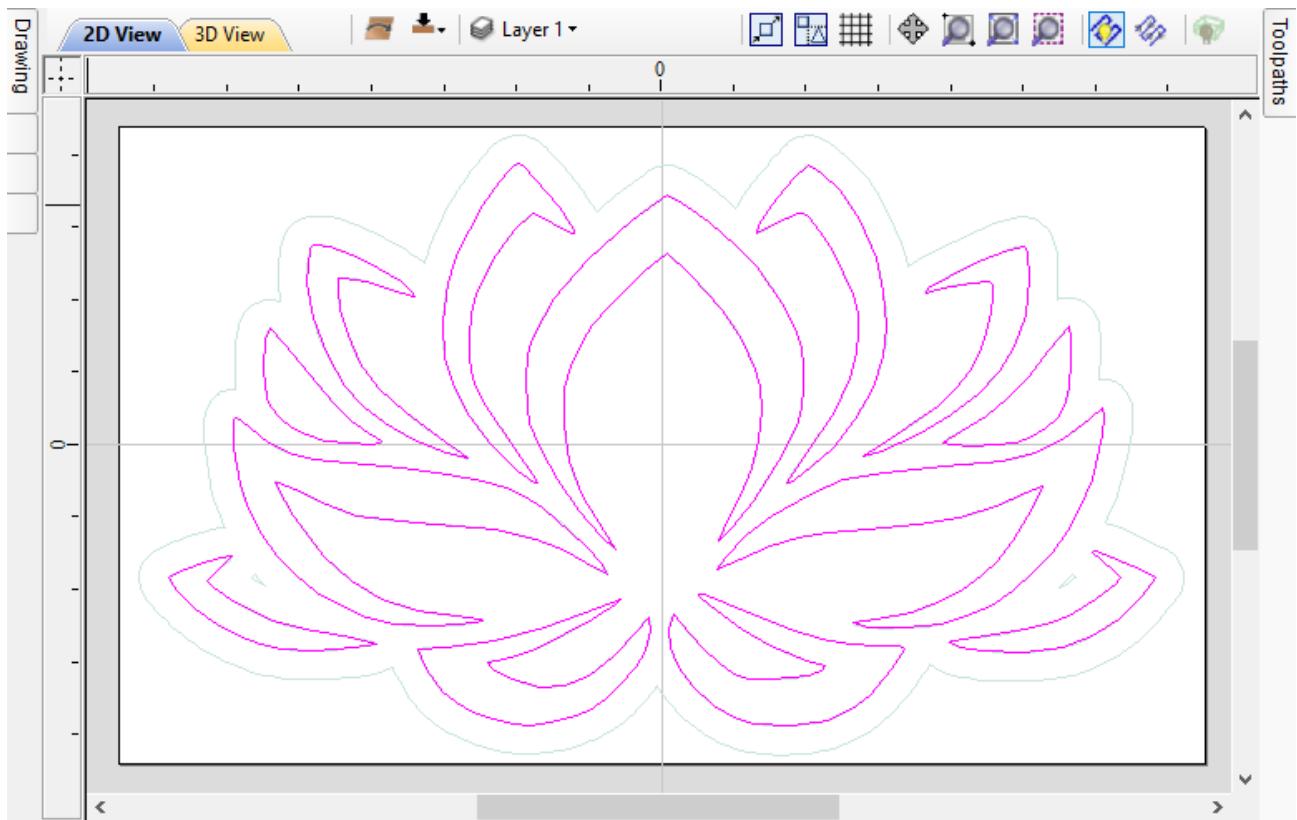
You should then end up with two pieces that match perfectly (the V-bit angle does that for you), and with the inlay sticking out from the final surface by the cutout depth plus the inlay base depth:



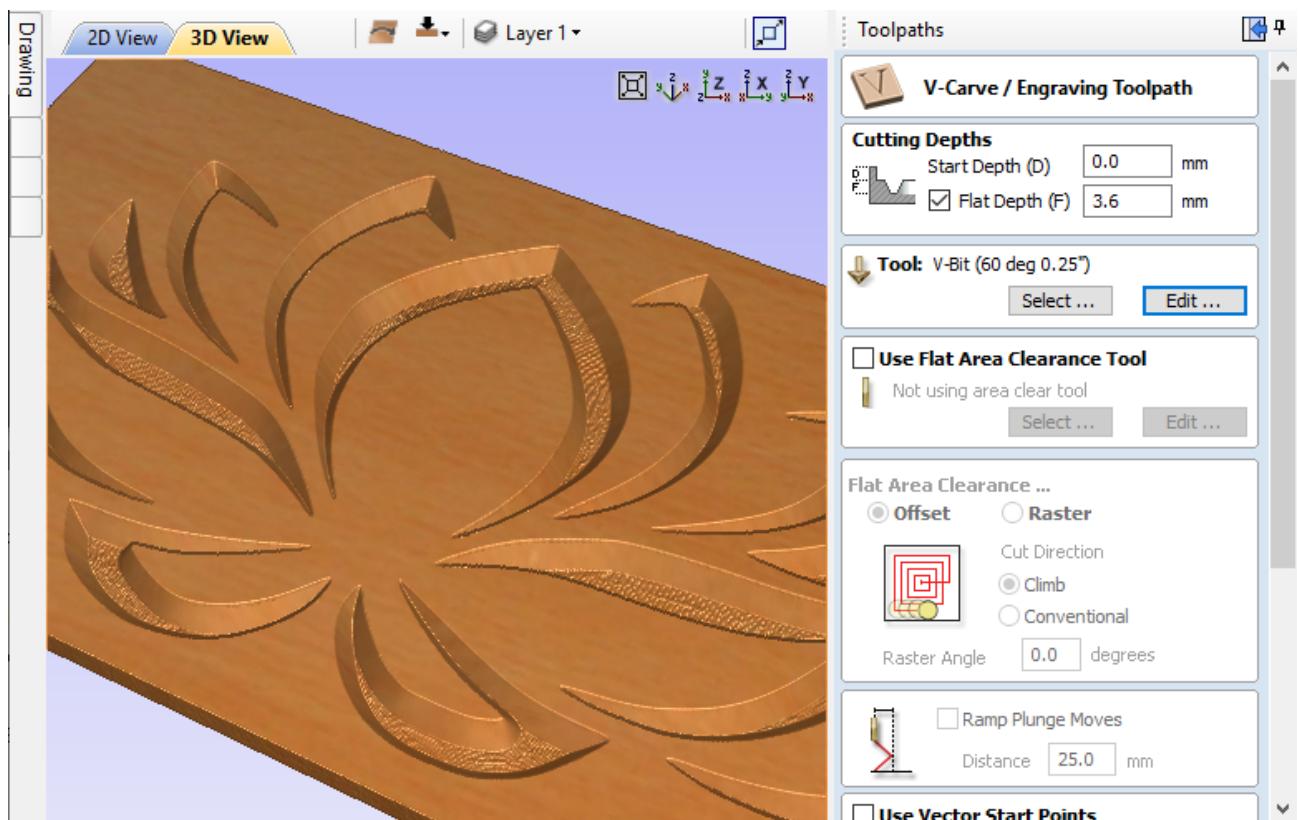
Below is an example project of carving a lotus flower inlay using a 60° V-bit.

- (i) It was designed in Vectric's VCarve: its support for double-sided projects makes managing the base and the inlay in the same design file easier, but the same can be done in any CAD software supporting V-carving toolpaths.

First, I imported a vector design, and selected all parts of the artwork to carve the base:



I chose the inlay to be 3mm thick, and added a 0.6mm glue gap, so the v-carve toolpath has a start depth of 0, and a flat depth of 3.6mm:



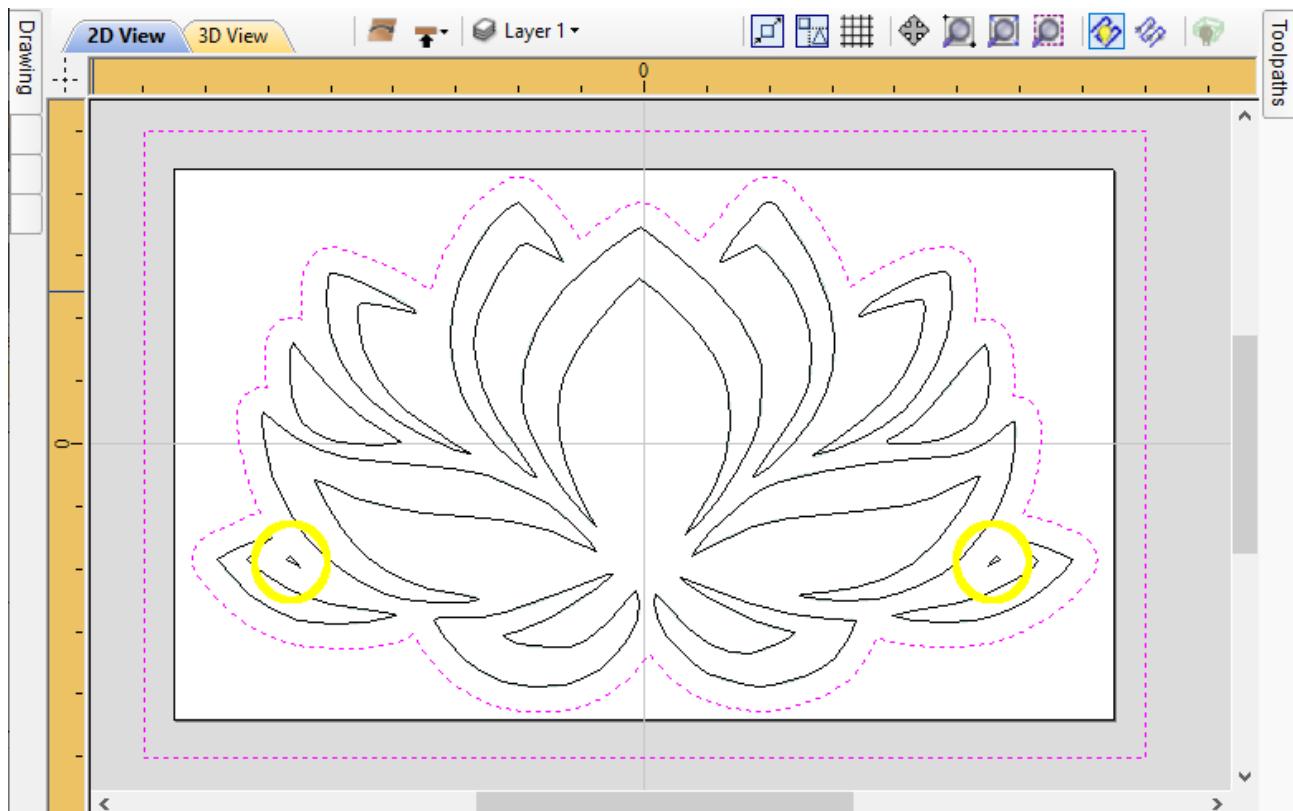
I ran that toolpath (using a 60° V-bit at 24,000RPM, 60ipm feedrate, 14ipm plunge rate, 0.1" pass depth, 0.06" stepover) and got this:



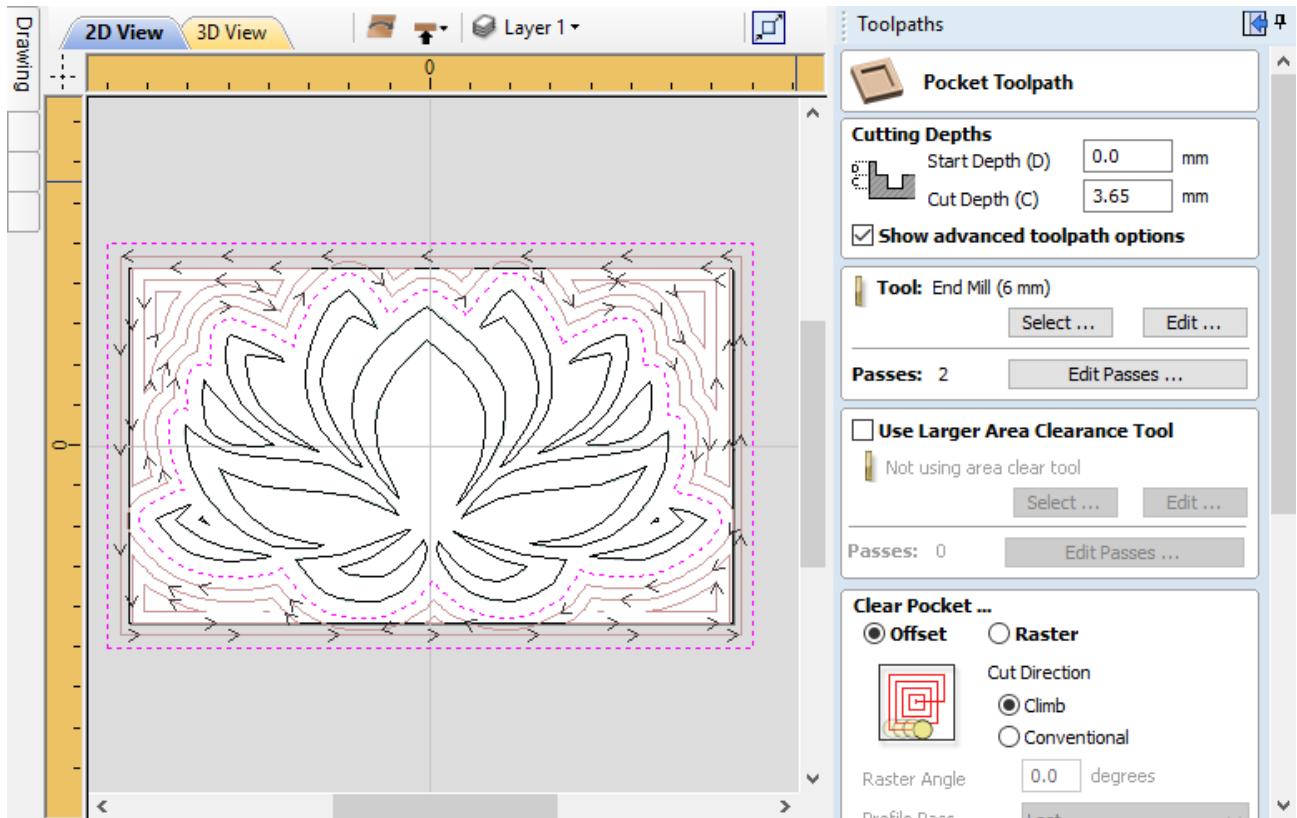
Next, I switched to the other layer to design the inlay part:

- I imported the same vector but **FLIPPED IT** left/right. This is important ! That lotus flower design is *almost* symmetrical, but not quite.
- I selected all shapes and created an **offset** vector around them: this is used to create a boundary for the v-carve operation near the area of interest, and another toolpath is used to clear the rest of the material in the "boring" part:
 - you could decide to skip this step and use the V-bit to clear material all around the selected shapes, but V-bits are very inefficient for clearing large areas, so this would take forever.
 - you may need to clean-up artefacts generated by the offset operation, see yellow circles in the example below

- I also created a **perimeter** around everything, corresponding to the stock size (outer rectangle in the example below), to define the area where the flat pocketing operation will run.



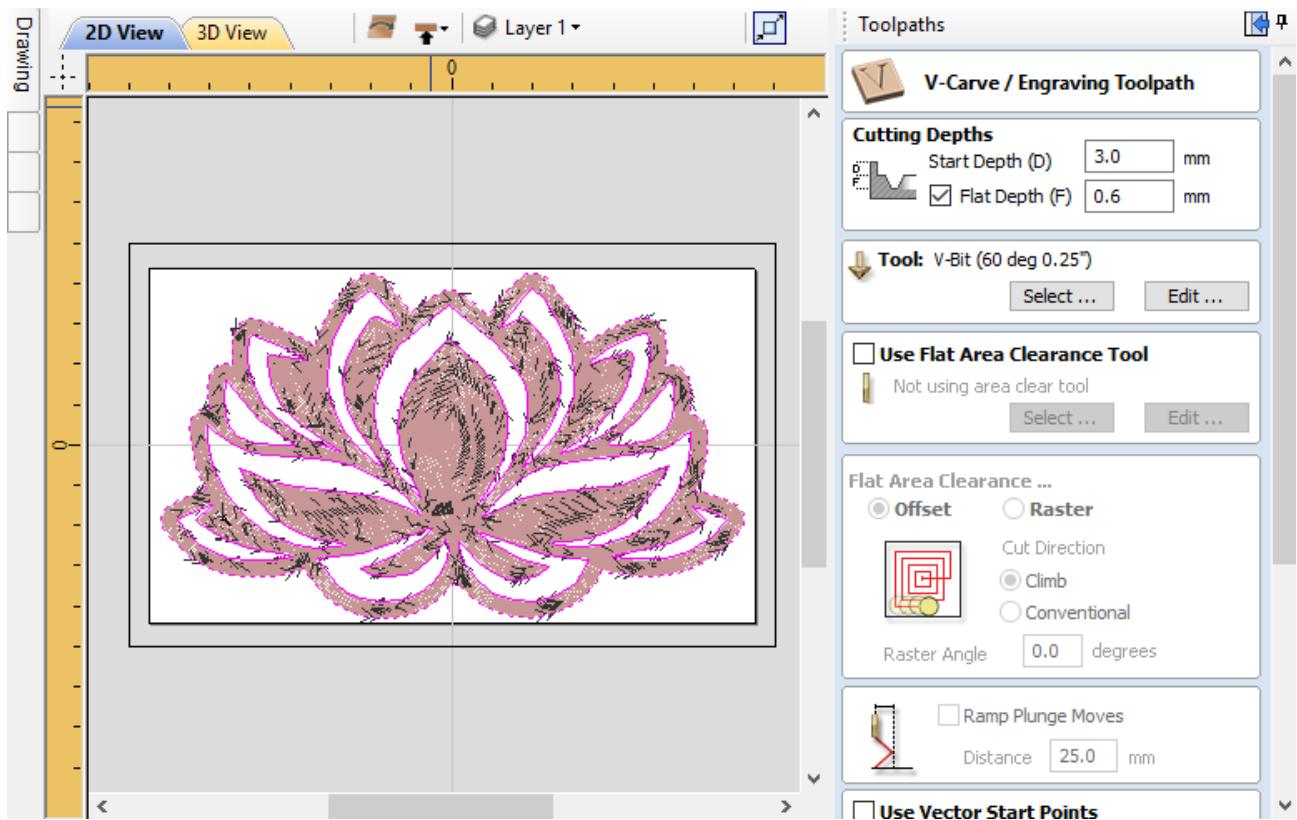
The bulk of the material, between the outer perimeter and inner offset shape, is removed using a square endmill, down to a (flat) depth of 3.65mm (3mm inlay height, 0.6mm cutout depth, and an extra 0.05mm margin just because):



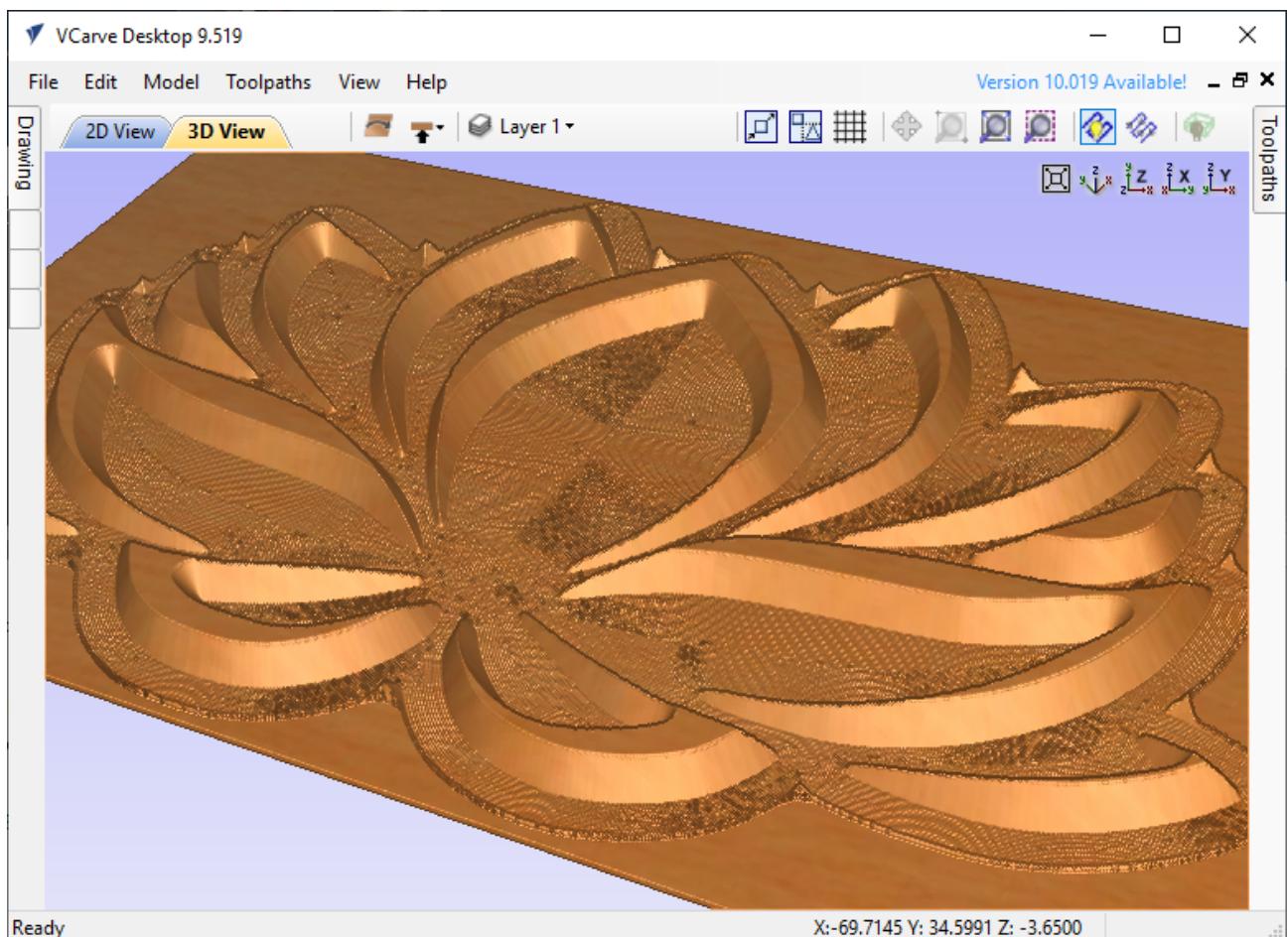
i Some CAM software products will support this clearance pass with a straight endmill natively, as part of the v-carve toolpath options. The idea remains the same, it just saves you the trouble of manually creating the offset shapes.

Now the most important step: I created the v-carve toolpath for the inlay itself. The trick is to select the (flipped) artwork, and then:

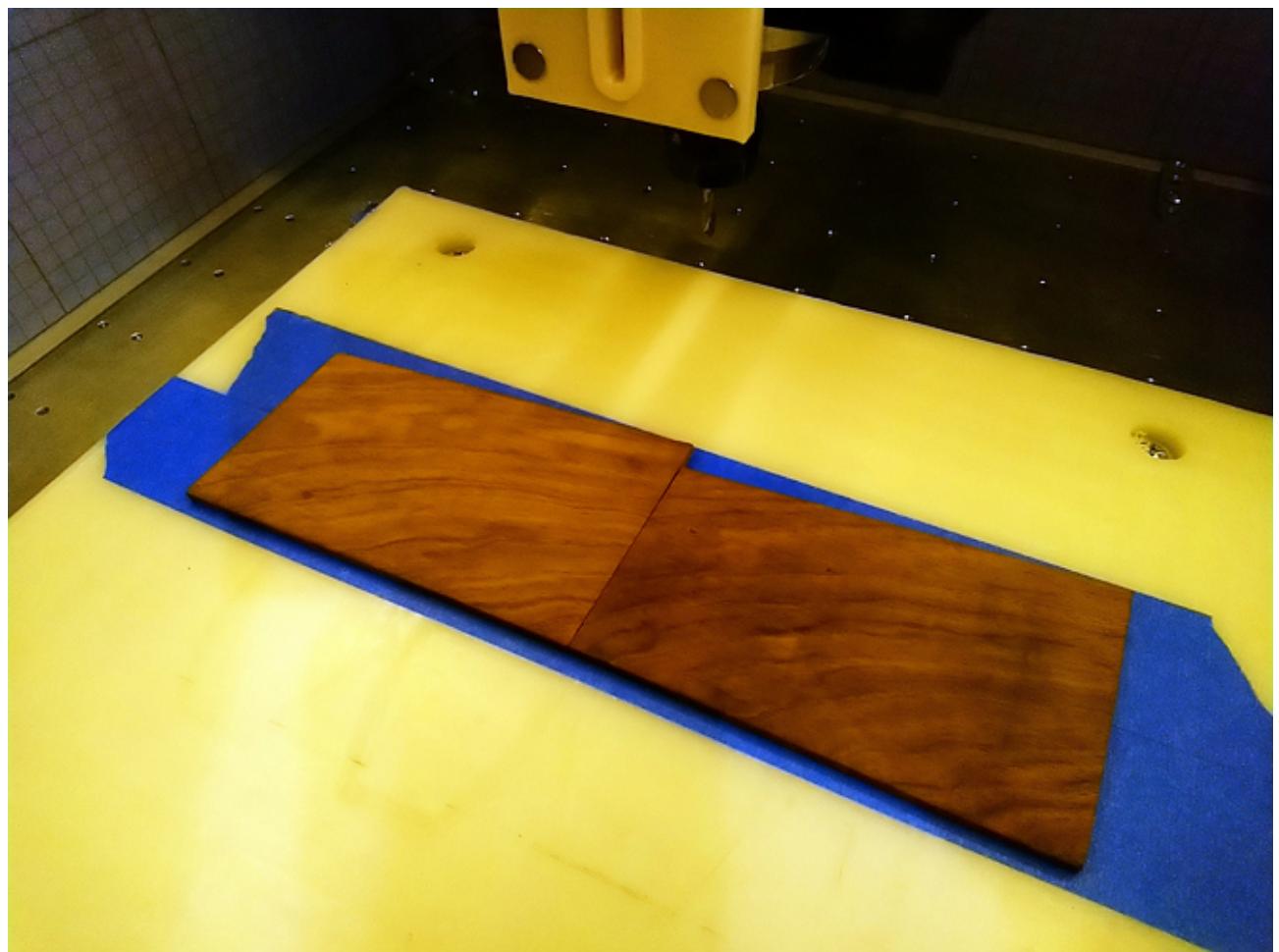
- use a **start depth** corresponding to the inlay thickness (in my case 3mm). This corresponds to the height of the inlay above the artwork plane.
- use a **flat depth** corresponding to the extra/cutout margin.
 - here, I chose 0.6mm, but this is a coincidence, it has nothing to do with the glue gap depth in the base part. Since I did not intend to use a bandsaw for cutting the top of the inlay, 0.6mm provided enough margin for depth errors.



The preview after simulating both toolpaths shows the intended result:



After creating all this, I realized my walnut stock was...not large enough to accomodate the design in one go. No worries, I split the toolpaths between two smaller halves, and proceeded to prepare the stock:



Which gave me this, after running both the pocketing (clearing) toolpath, and the v-carve toolpath:



Then came the rewarding moment of inserting the inlay in the base, for a perfect fit. I used a generous amount of glue in the base, positioned the inlay(s), added weight on top, let it cure, and got this:



To remove the excess material, one way is to leave a large cutout depth, then use a bandsaw to cut in-between the base and the inlay base. I chose instead to use the Shapeoko to do a simple surfacing operation with a square endmill:



After a little sanding and oiling, the resulting inlay looks like this:

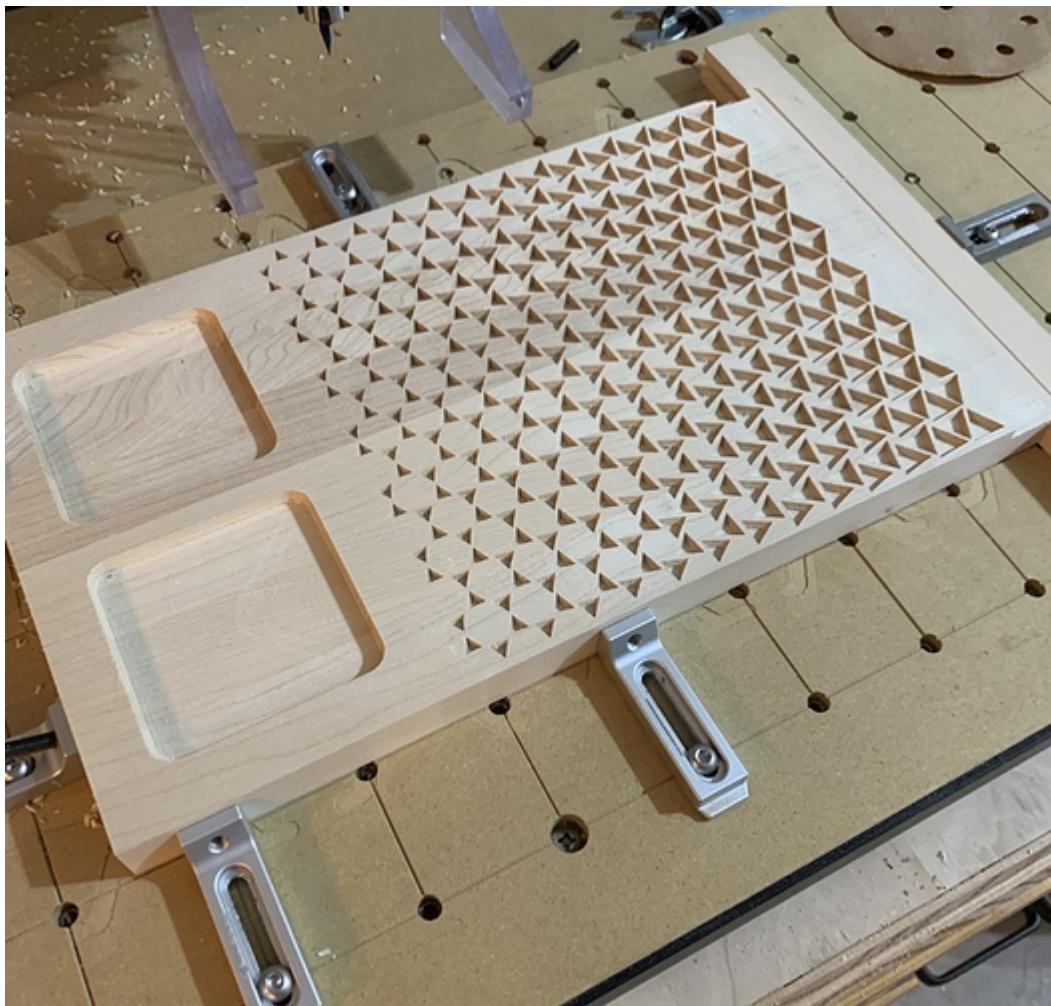


But inlays look much better when there is a lot of contrast between the base and the inlay, so I had another go at it:

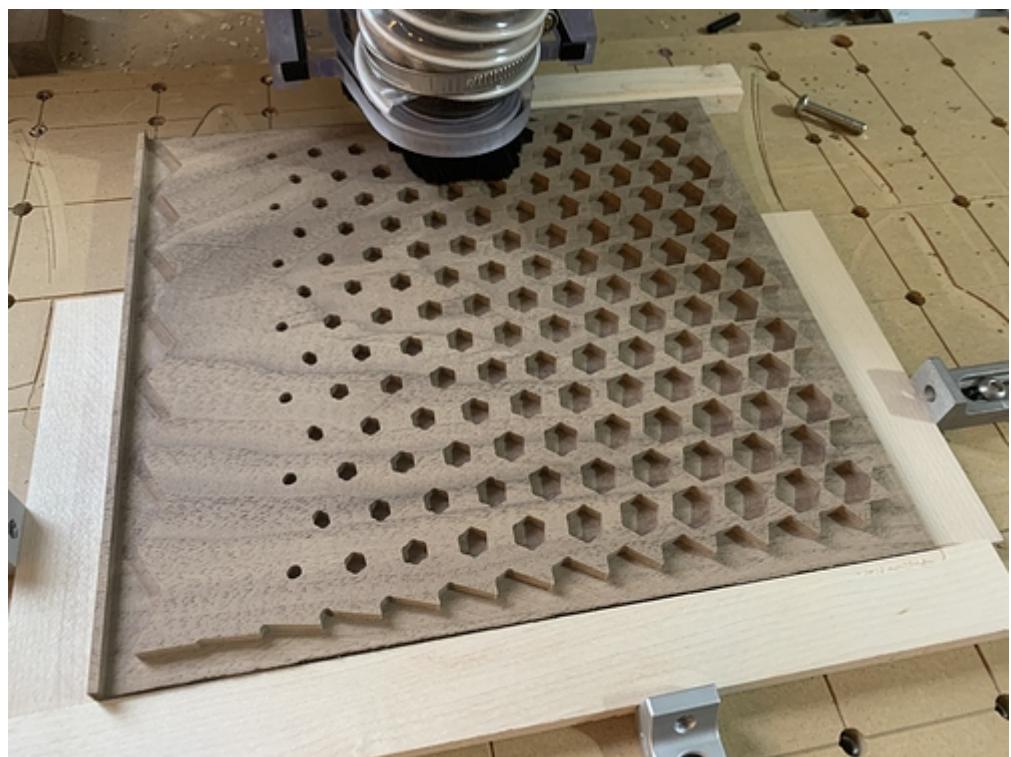


Community member **@i3oilermaker** used the same concept to make a beautiful serving tray that I wanted to share here, as a great example of what one can achieve with v-carved inlays:

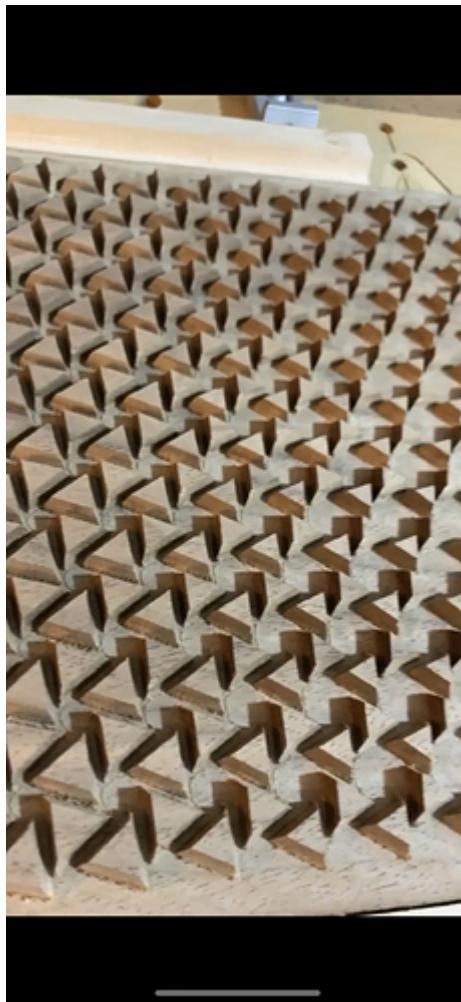
- first, V-carving a maple base with a 30deg V-bit:



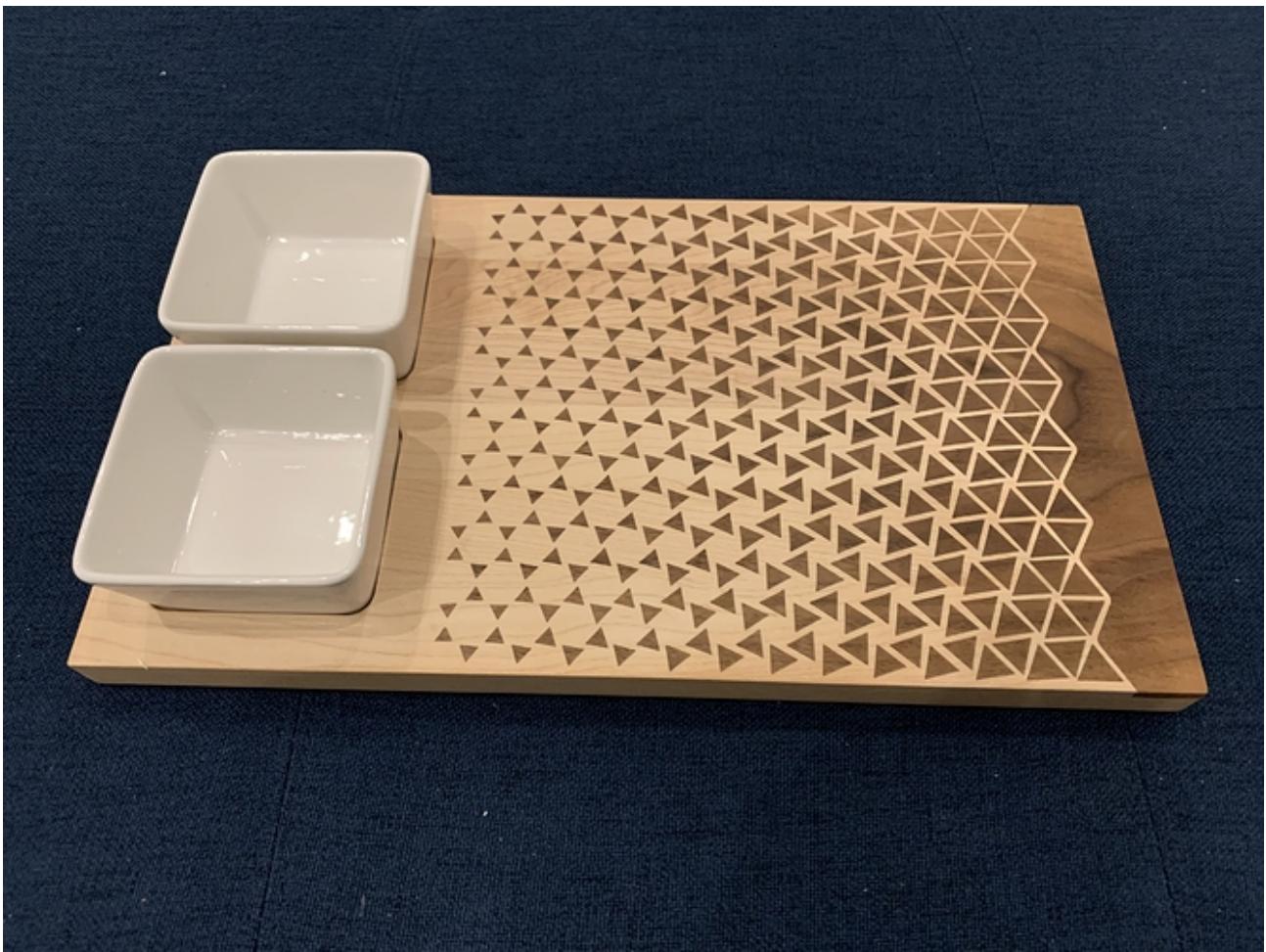
- Then running a clearance pass for the walnut inlay part, using a 1/8" square endmill:



- Then running the V-carving pass itself, to get this:



- And finally glueing the inlay onto the base, letting it dry, cutting the excess material with a surfacing bit, and finishing the part to get this beauty:



Usecases: cutting plastics

Plastics are relatively easy to mill, but they are less forgiving than wood if feeds and speeds are not set to appropriate values. The reason is that plastic melts easily when heated, and since cutting forces produce heat...the name of the game is to evacuate heat as efficiently as possible.

This boils down to two things:

1) Watch **the chipload**:

To ensure heat removal, you should either cut lots of thin chips very quickly (i.e. use high RPM, and associated high feedrate maintaining chipload above 0.001"), or cut thicker chips at lower RPM/feedrate.

If you are going for the high RPM option, a sharp cutter and a chipload of 0.002" should do the trick for most of the situations.

If you are going for low RPMs, then you should aim for the high-end of the recommended chipload range. As discussed in the [Feeds & speeds](#) section, for a 1/4" endmill in plastics, this is between 0.005"/ 0.13mm and 0.01" / 0.25mm. Such chiploads are hard to reach with 3-flute endmills. For example on the Dewalt router at its minimum RPM of 16,000, and at the highest possible feedrate (196.85"/ 5000mm per min), the chipload maxes out at $\sim 197 / (3 \times 16,000) = 0.004"$.

A usual solution is to use a single-flute (**O-flute**) endmill, this will allow a higher chipload at a comfortably slow feed rate. Recommended chiploads for smaller tools (1/8" and below) are easier to reach, even with two flute endmills. Another solution is to use even lower RPMs (below 10k), but that is only possible for spindle owners, not for Makita/Dewalt routers.

2) Make sure **chip evacuation** will not be a problem:

Even at the correct chipload, if chips get in the way of the cut, melting may happen. It is important to ensure that chips are evacuated efficiently, with one or several of the following solutions:

- a good dust shoe (or manual vacuuming).

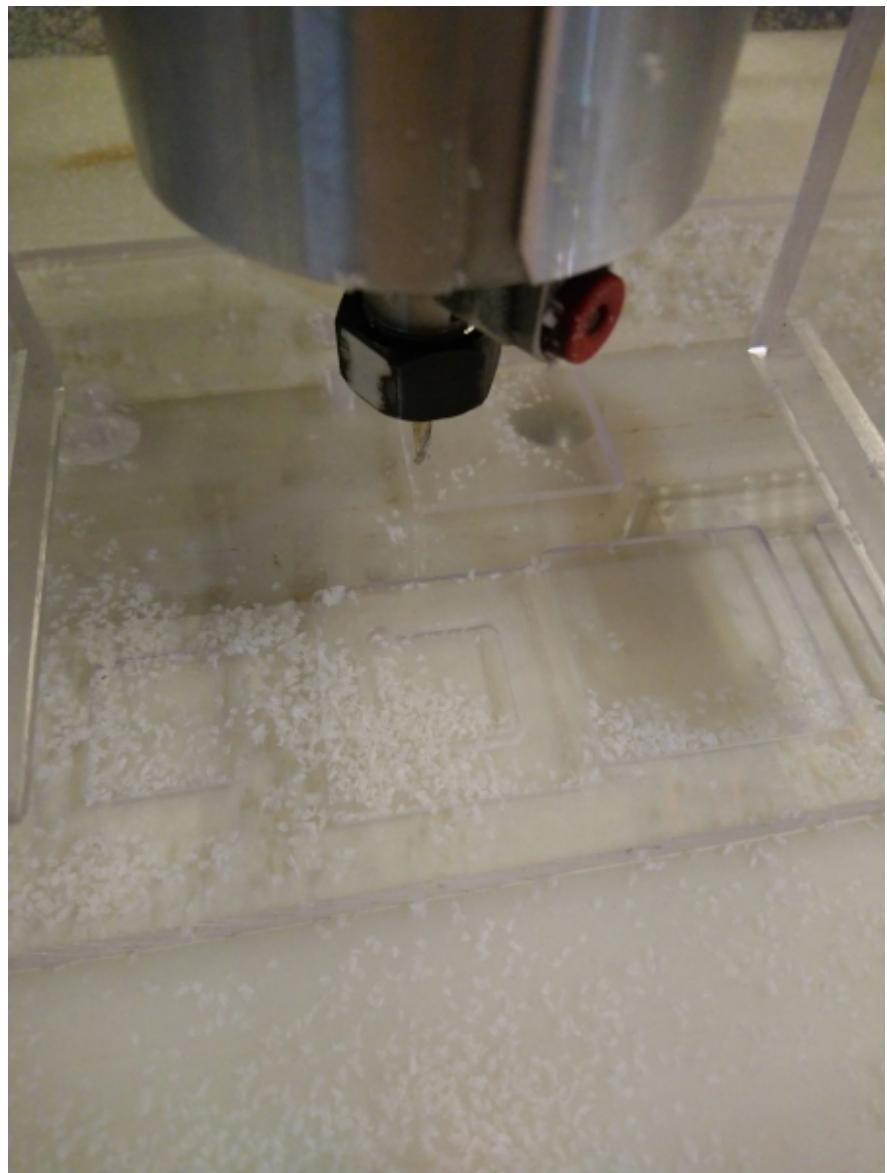
- an air blast directed at the cut.
- avoiding deep & narrow pockets/slots when possible.
- using an O-flute endmill (beyond the chipload vs. RPM/feedrate advantage, it also leaves much more room for chips to get away from the cut, compared to 2 or 3 flute endmills).

Two very common types of plastic are hard plastics like acrylic (cast or extruded) and soft plastics like HDPE (High Density PolyEthylene), example cuts for both are provided below.

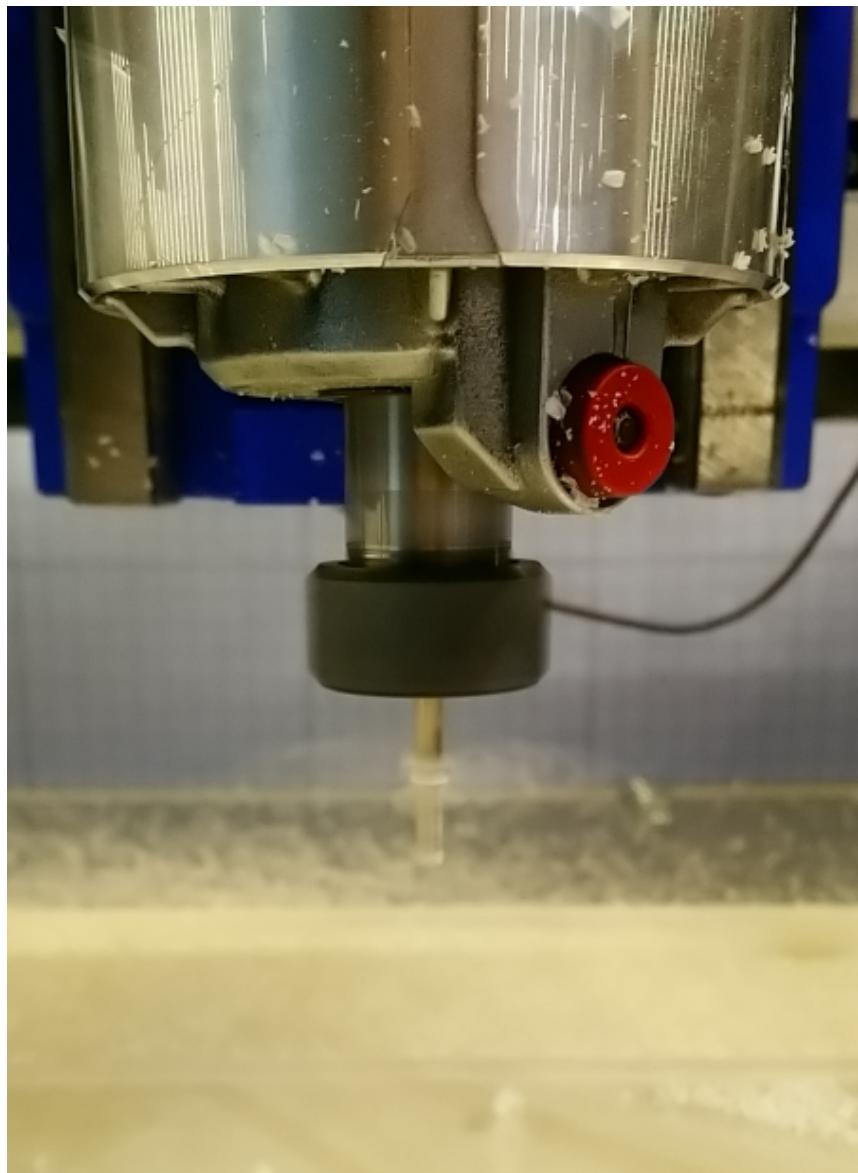
Acrylic

For the first example below I went with the "low RPMs" approach and used a 2-flute 1/8" endmill, 10,000 RPM, and a feedrate of 1200mm/min (47"/min), to get a chipload of $1200 / (2 \times 10,000) = 0.06\text{mm} = 0.0023"$, which is the high end of the range recommended in the [Feeds & speeds](#) section for acrylic for this endmill size. The depth of cut was 50% of the endmill diameter i.e. 0.0625"/1.5875mm, and stepover was 0.056"/1.4mm.

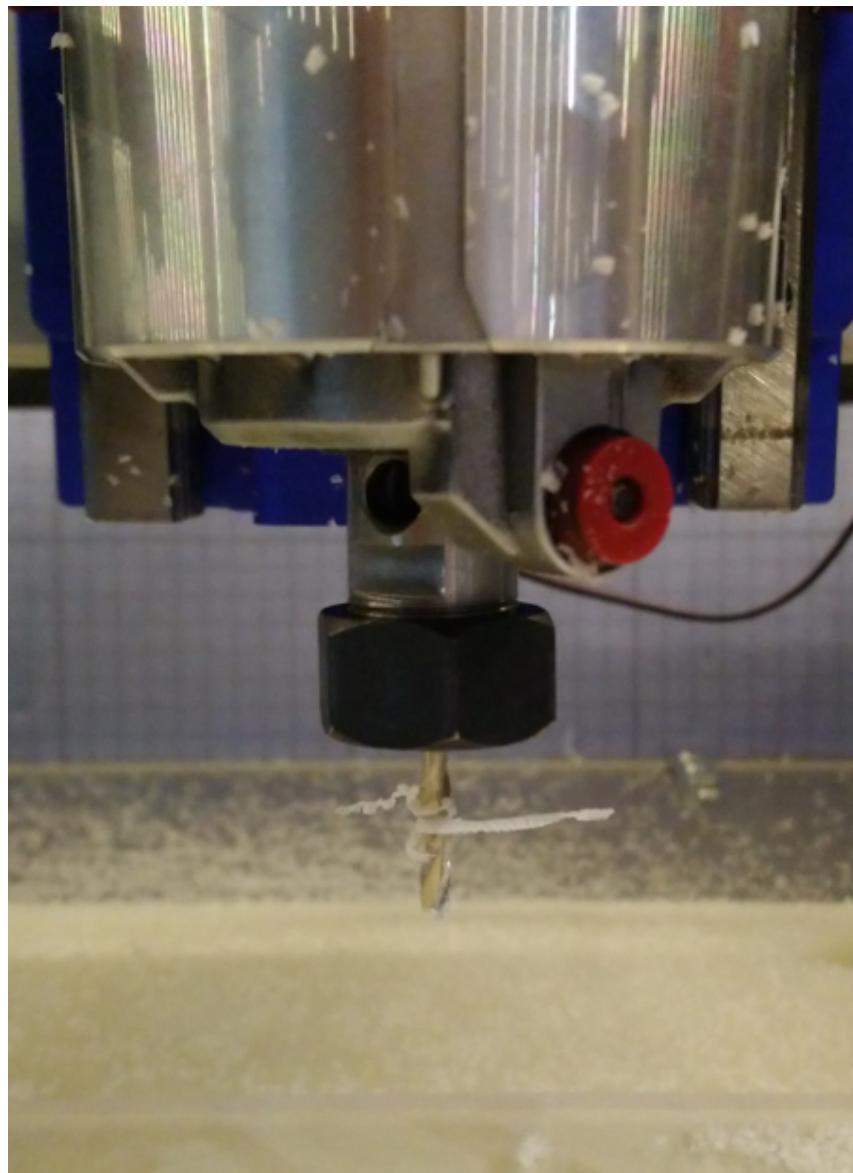
You should be making well-formed snowy chips :



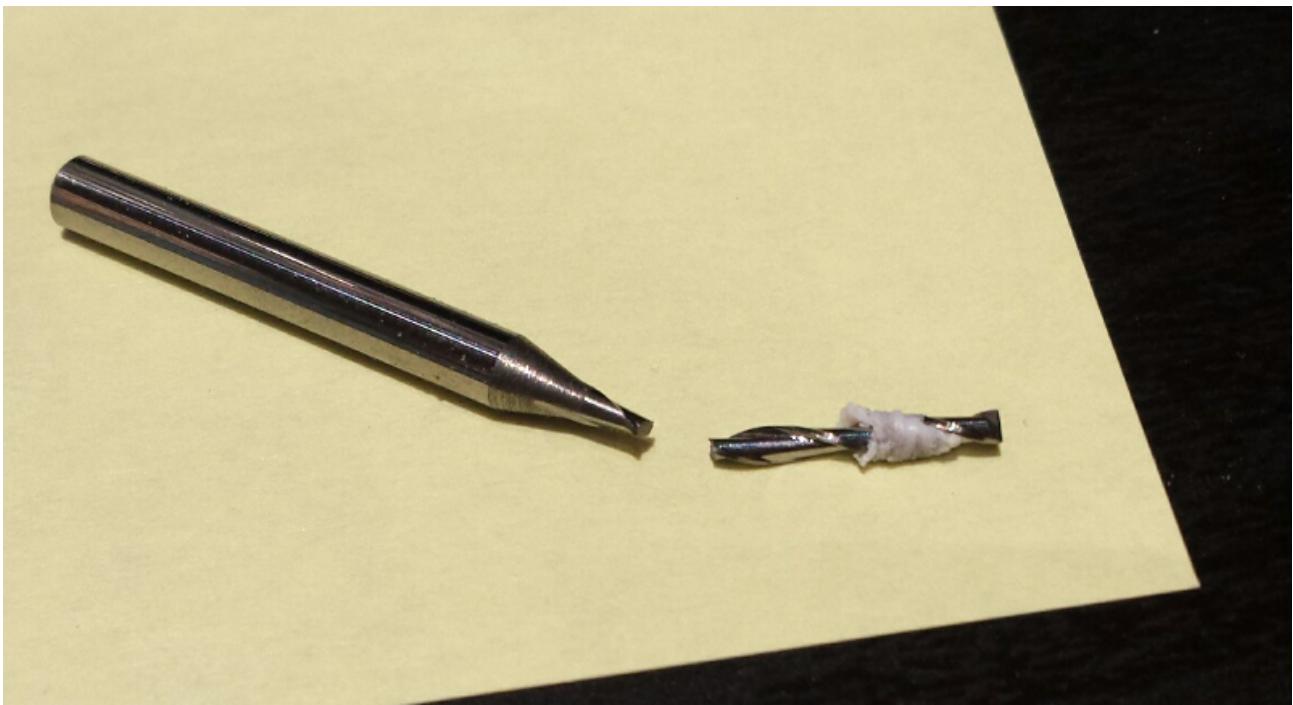
If you get gummy edges on the cut, or see strings of plastic accumulating on the endmill, then the feedrate is too low and/or the RPM is too high. Here is what it may look like while the router is running:



and with the router turned off:



This string of plastic is an indication that rather than cutting, the endmill is melting & pushing plastic out of the way. You should stop the cut, because things will probably go bad quickly, with melted plastic accumulating in the flutes, leading to more rubbing and more melting, until the endmill is covered in plastic and is not cutting anything anymore, and eventually breaks as the machine continues to push it through the material:



(i) In the [Feeds & speeds](#) section, the guideline for **plunge rate** is "*40% to 50% of the feedrate for plastics*". You want to be on the high end of this range: plunging too slowly will result in a little melted plastic string on the endmill at the beginning of the cut, just like in the pictures above. And then even if the feedrate is correct for the rest of the cut, if you accumulate these strings of plastic on the upper part of the endmill each time the tool plunges, you may end up in a situation where this is enough to get in the way of the cut, and then bad things happen.

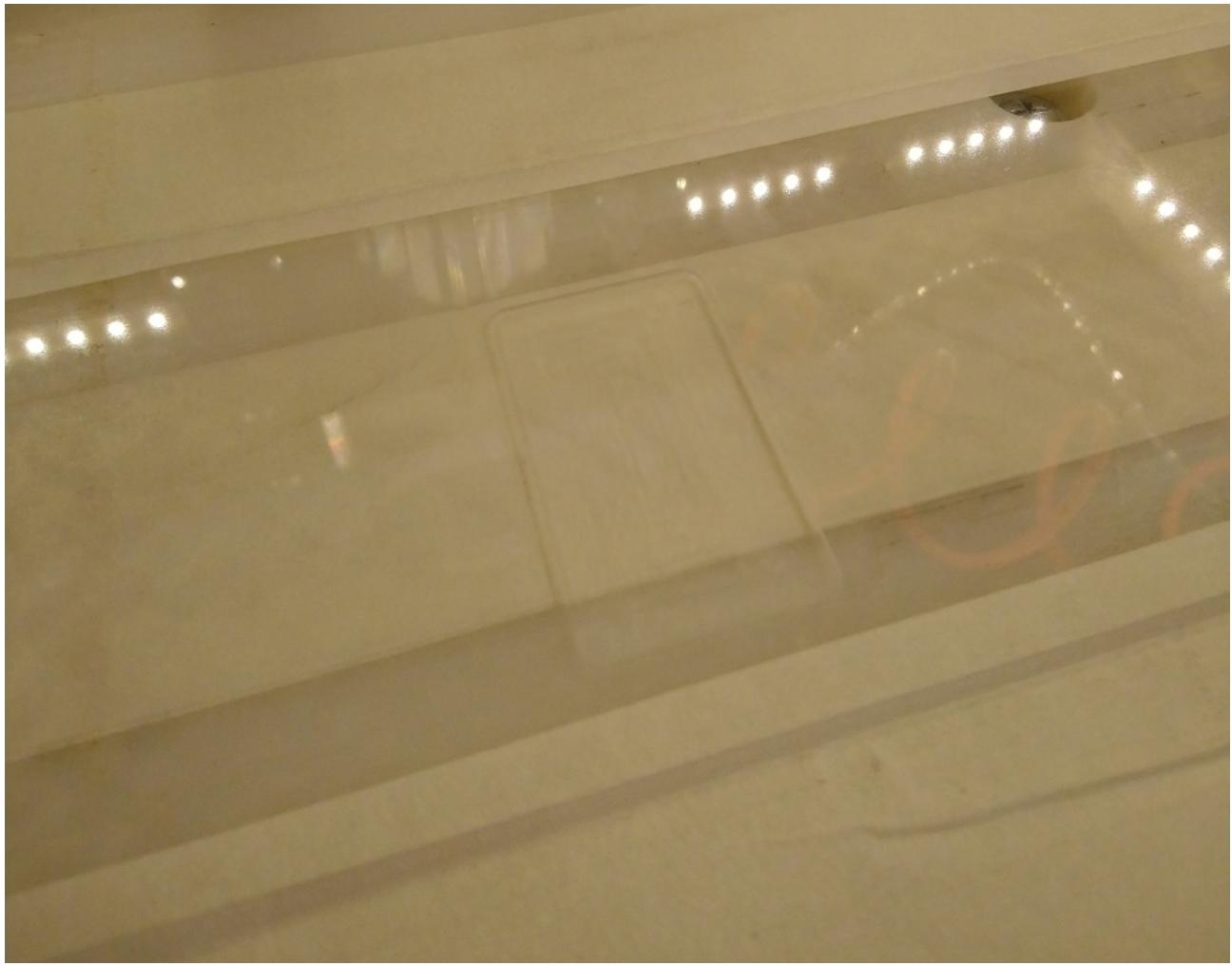
Using the "high RPM" approach works fine too as long as chipload is in the right range. In the second test illustrated below, I used a 2-flute 1/4" endmill for a pocketing operation in acrylic:

- At 25,000 RPM (near the max of the Makita router RPM range)
- for a 1/4" endmill in acrylic, the recommended chipload range is 0.001"-0.005", I went for 0.002" to have a little margin above the 0.001" minimum. The associated required feedrate was therefore $0.002" \times 2 \text{ flutes} \times 25,000 \text{ RPM} = 100\text{ipm}$

The cut produced equally good chips,



and a clean cut:



HDPE

For HDPE and if cutting at low RPM/feedrate, the chipload value needs to be pushed even further as this is a soft plastic that melts easily.

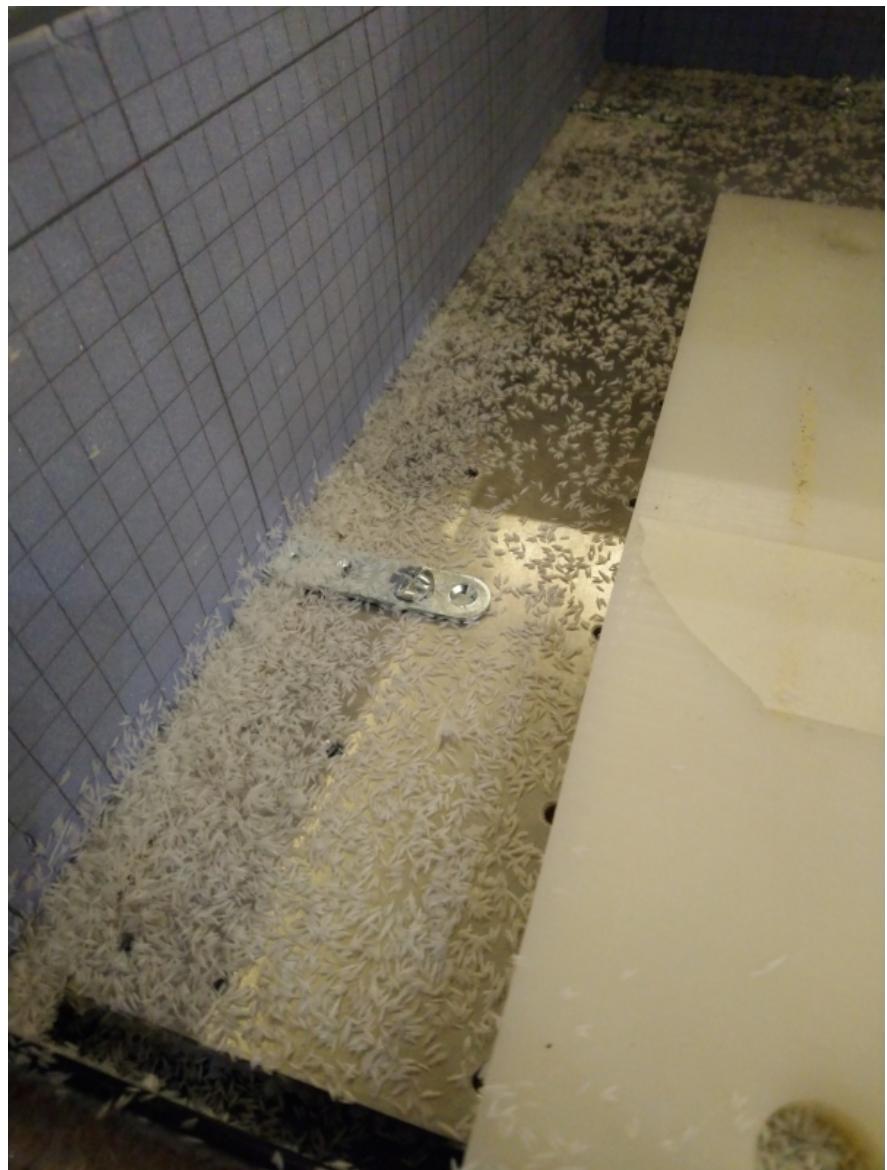
To get the nice clean chips below, I used a 1/8" O-flute at 10,000RPM and 1200mm/min (47ipm), granting a chipload of 0.12mm (0.005"), which is the high end of the recommended starting range. For this pocketing operation the depth of cut was 50% of the endmill diameter, i.e. 0.0625"/1.5875mm, and stepover was 0.056"/1.4mm (44%)



To test the other approach to cut thinner chips but at much higher RPM/feedrate, I used a 2-flute 1/4" endmill:

- I went for 25,000 RPM again (nearly maxed out)
- I chose to try a chipload of 0.002" (the min recommended value for 1/4" in soft plastics)
- the associated required feedrate is therefore $0.002" \times 2 \text{ flutes} \times 25,000 \text{ RPM} = 100 \text{ ipm}$
- DOC happened to be 0.1" in this test.

This got me an equally good cut, with nice chips and no sign of melting:



Usecases: cutting metal

Profile cuts in copper

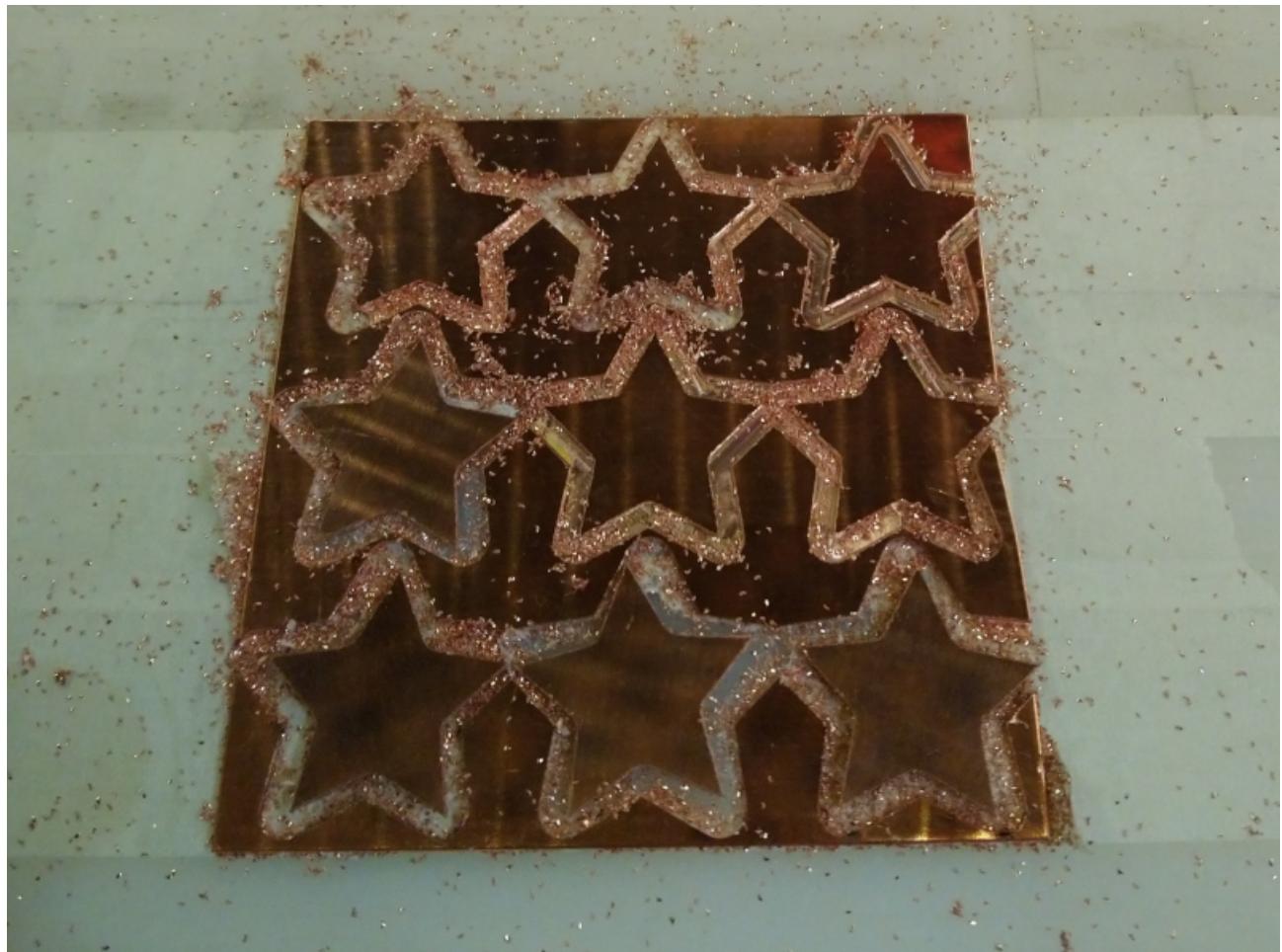
The following was just a little experiment in doing profile cuts in copper with a 2-flute 1/8" square endmill (#102Z from Carbide 3D store), to figure out which feeds and speeds would work. I used a 100mmx100mm (~4"×4") piece of 0.9mm (0.035") thick copper, used tape and glue to hold it onto the wasteboard, and cut simple profiles at increasing chiploads, to find the sweet spot.

The toolpath is a single **contour/profile** cut, which is not the best thing to do if you need a perfect finish (you should rather do a roughing cut followed by a light finishing cut to have perfect edges), but for the purpose of this test, it represented a worst case scenario i.e. slotting throughout the cut.

- the first step was to determine an adequate target **chipload** for a 1/8" endmill in copper. The guideline in the [Feeds & speeds](#) section for aluminium is from 0.0005" / 0.0127mm to 0.001" / 0.0254mm. Aluminium and copper are different, and their hardness varies with the specific type/temper used, but overall they both are in the 75–150BHN hardness range, so I assumed I could use the same target chipload (since my copper sheet was of unknown origin, I could not make any better informed decision anyway).
- **stepover** did not apply this since was a slotting cut. Therefore **chip thinning** did not apply either.
- I chose **12,000 RPM**, to keep things quiet and since cutting force was not going to be a problem for such a shallow cut.
- to achieve the 0.0005" chipload at 12,000RPM with this 2-flute endmill, the **feedrate** needed to be $0.0005 \times 2 \times 12000 = 12\text{ipm} = 300\text{mm/min}$
- **plunge rate** should be low since this is metal *and* I would not be using any ramping into the material, I picked 4ipm (100mm/min)
- for **depth of cut** the guideline is 5 to 10% of endmill diameter for large WOC in metals, 5% of 1/8" is 0.00625" and 10% is 0.0125", I selected a middle value of 0.008" / 0.2mm
- **deflection** would not likely to be a problem for this shallow cut, the calculator told me the deflection for those settings was actually 0.002mm (0.00008") for a 20mm (0.8") stickout.

I actually tested 9 feedrates, starting from below the recommended min chipload, to above the max recommended chipload: 200, 300, 400, 500, 600, 700, 800, 900, and 1000 mm/min, which correspond to chipload values from 0.008mm (0.0003") to 0.042mm (0.0016")

-  I did not use any coolant or blast or air for this cut to be in a worst case scenario, for a real cut it would be better to do so.



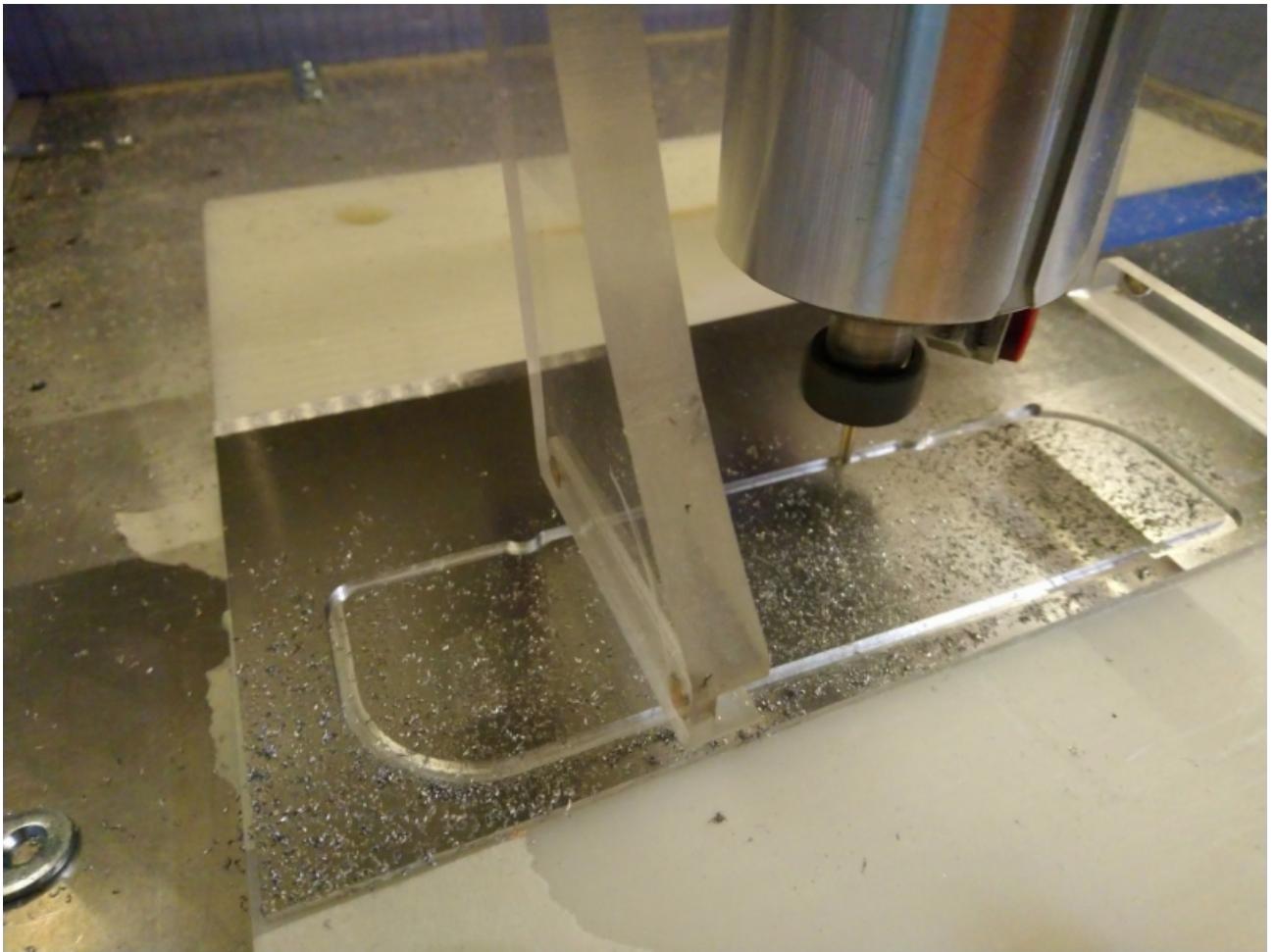
All the cuts completed without issue, but:

- the 200, 300, and 400mm/min cuts were a bit noisy with hints of chatter during some passes
- it got better at 500mm/min
- 600mm/min was the best cut (clean noise and clean cut)
- 700 to 1000mm/min cut got increasingly noisy and rough.

For this specific test, 600mm/min i.e. a chipload of 0.001" / 0.0254mm seemed to be the sweet spot, but it also showed that there is a good range of usable feedrates around this optimal value.

Profile cut in aluminium

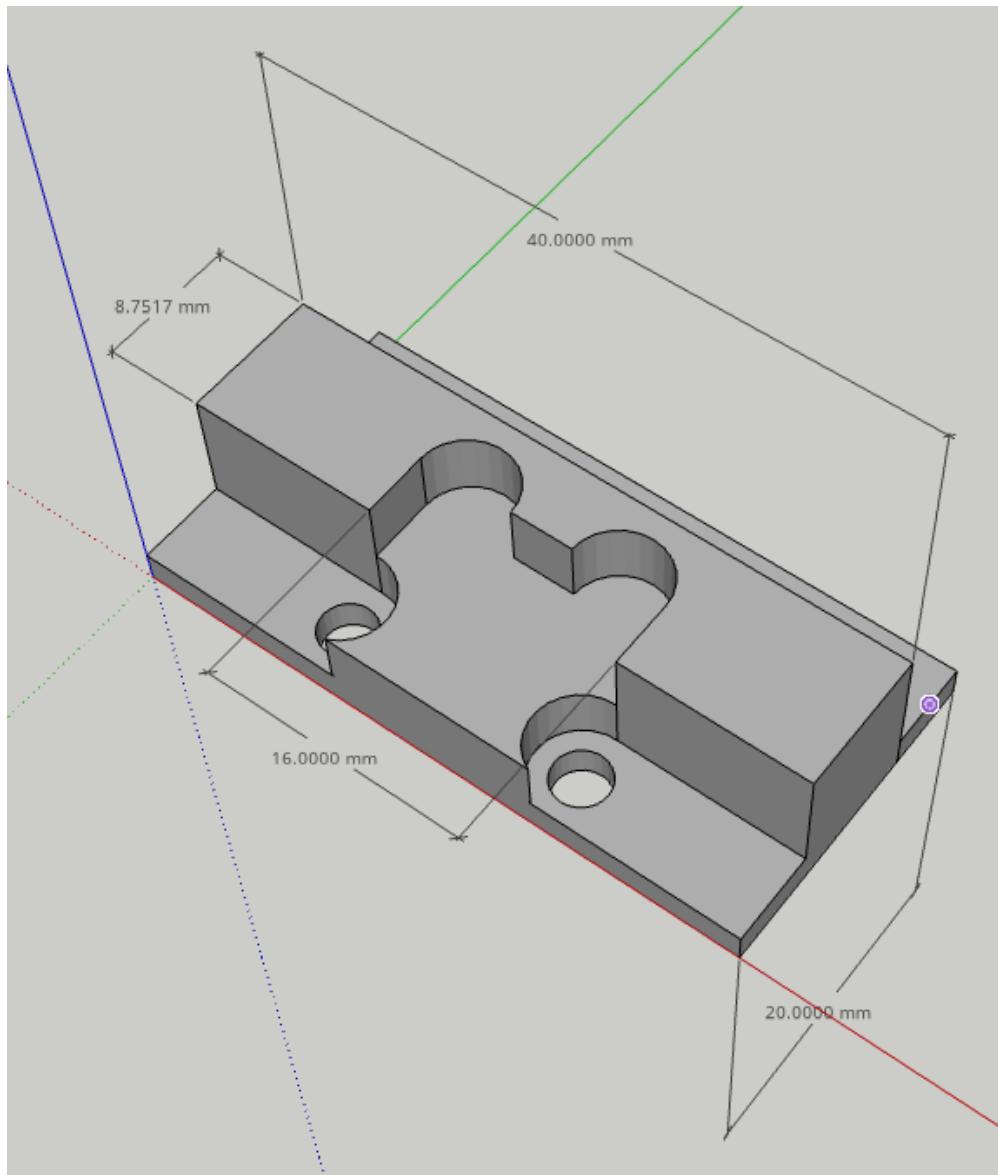
- same 2-flute ZrN-coated 1/8" square endmill, #102Z
- material for this test was 2017 T6 aluminium (a remote European sibling of 6061)
- again target **chipload** for an 1/8" endmill in aluminium is 0.0005" / 0.0127mm minimum, I started from that.
- same **12,000 RPM** speed
- to achieve the 0.0005" chipload at 12,000RPM with this 2-flute endmill, the **feedrate** needs to be $0.0005" \times 2 \text{ flutes} \times 12000 \text{ RPM} = 12\text{ipm} = 300\text{mm/min}$
- **plunge rate** of 100mm/min (4ipm) again.
- for **depth of cut** I tried the high end of the recommended range for metals, i.e. 10% of the endmill diameter, so 0.012" / 0.3mm
- predicted **deflection** is still negligible at 0.001mm (0.00004") for a 20mm (0.8") stickout
- still using a simple slotting toolpath, but this time I used linear ramping and lead-in/lead-out options in VCarve.



Once the cut started, I could feel that the 300mm/min feedrate was a bit too low, so I gradually increased using **feedrate override** it until it sounded right, and ended up at +50%, i.e. 450mm/min (18ipm), i.e. a chipload of 0.00075", which happens to be right in the middle of the recommended range for this endmill size.

Adaptive clearing in aluminium

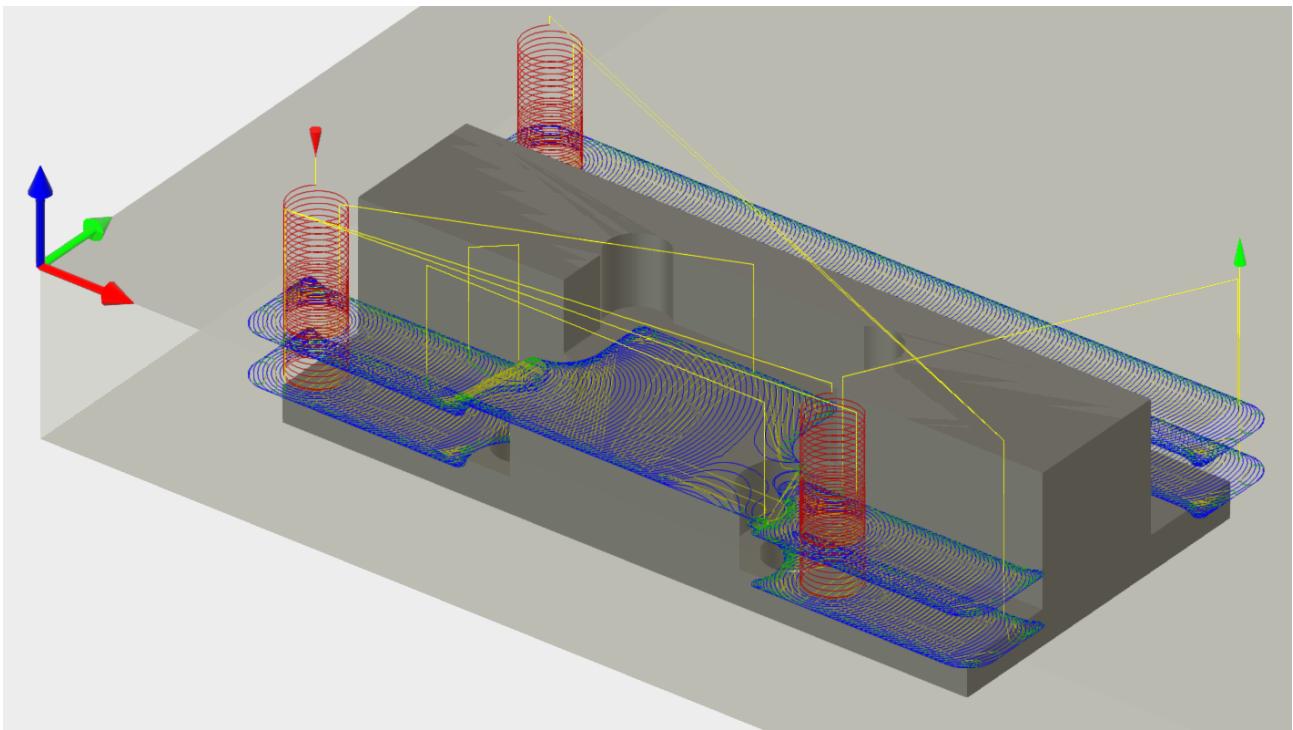
I needed to cut the following piece from 2017 T6 aluminium:



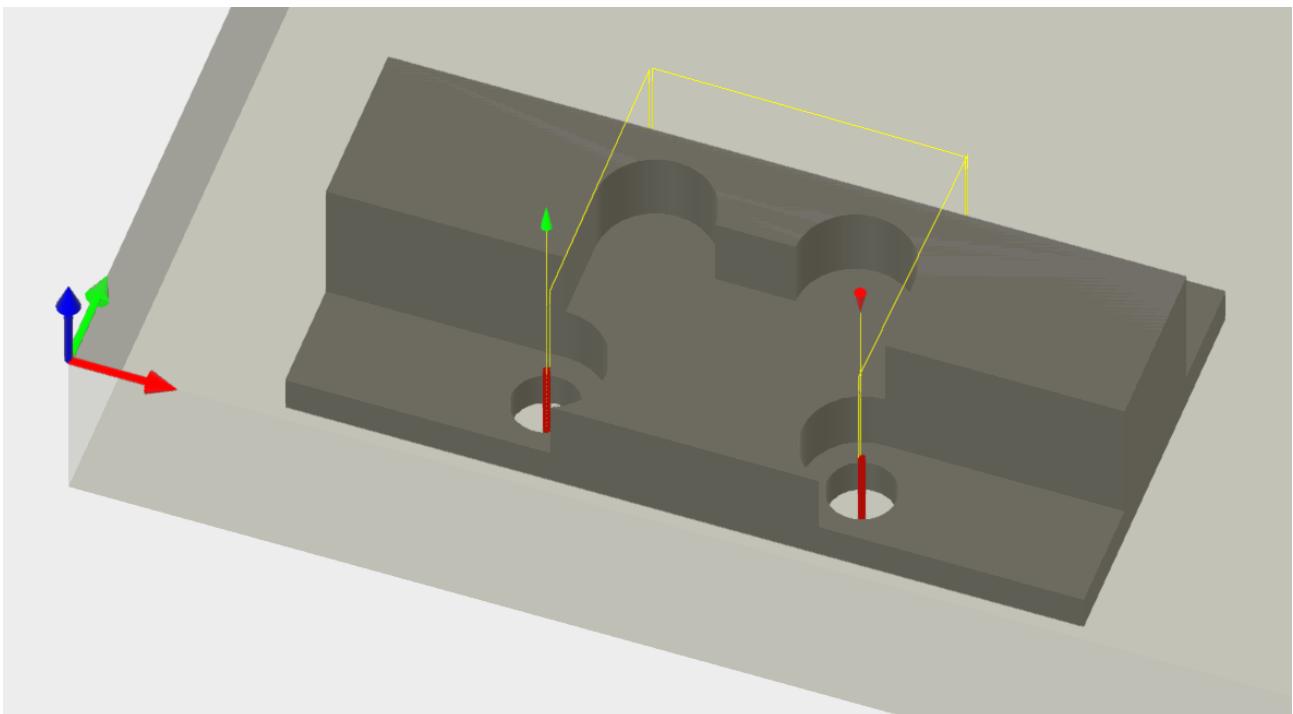
It is relatively small (about $0.8'' \times 1.5'' \times 0.4''$), and holes are about $0.15''$, so I went for using a $1/8''$ endmill (same as before: 2-flute ZrN-coated $1/8''$ square endmill, Carbide 3D's #102Z)

The majority of the cut was done with one **3D adaptive clearing toolpath** for roughing:

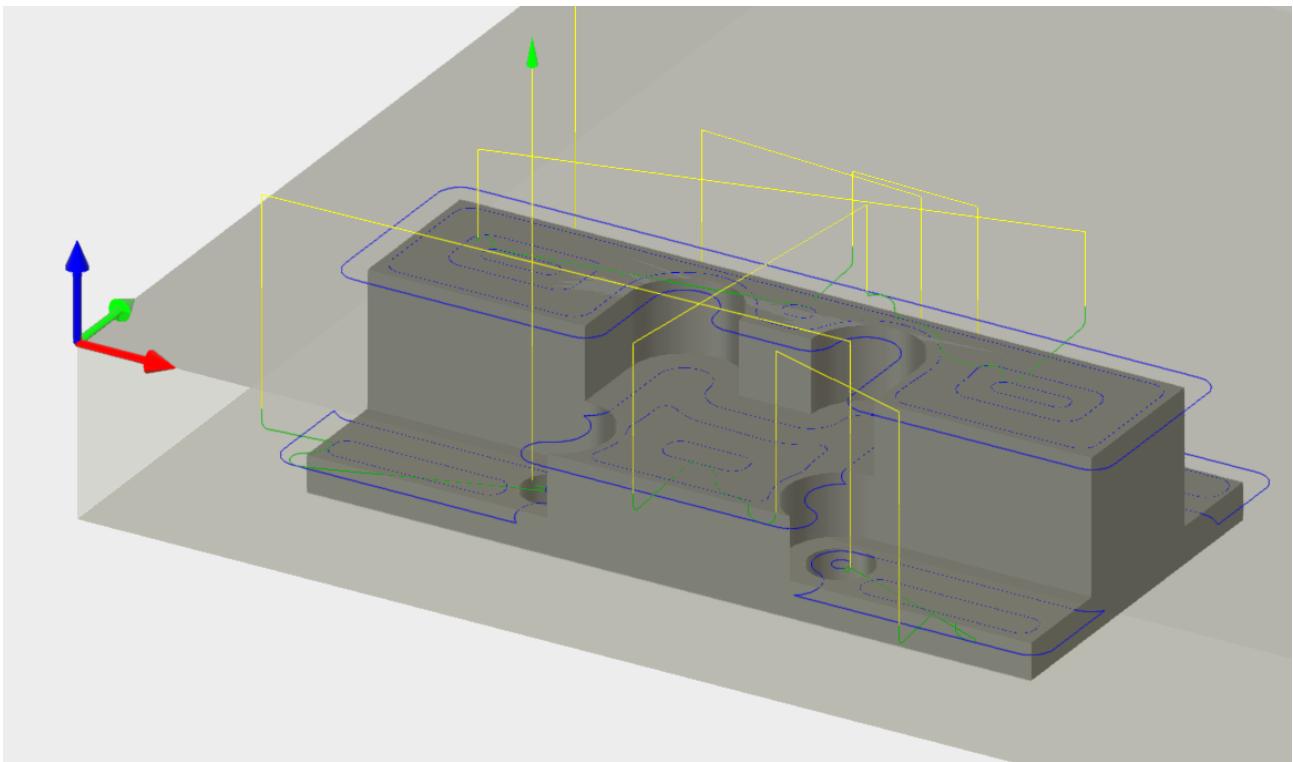
- 10,000 RPM
- target chipload of 0.001"
- stepover ("optimal load" in Fusion360) of 0.012" ($\sim 10\%$ of endmill diameter)
- target chipload adjusted for chip thinning for this stepover is 0.0017"
- feedrate was therefore 34ipm ($= 0.0017'' \times 2 \text{ flutes} \times 10,000\text{RPM}$)
- I used 13ipm plunge rate since helical ramping (shown in red below) allows it.
- I kept 0.02" axial stock to leave, and 0.02" radial stock to leave.
- 0.16" DOC ($\sim 130\%$ of endmill diameter), which resulted in two passes (in blue):



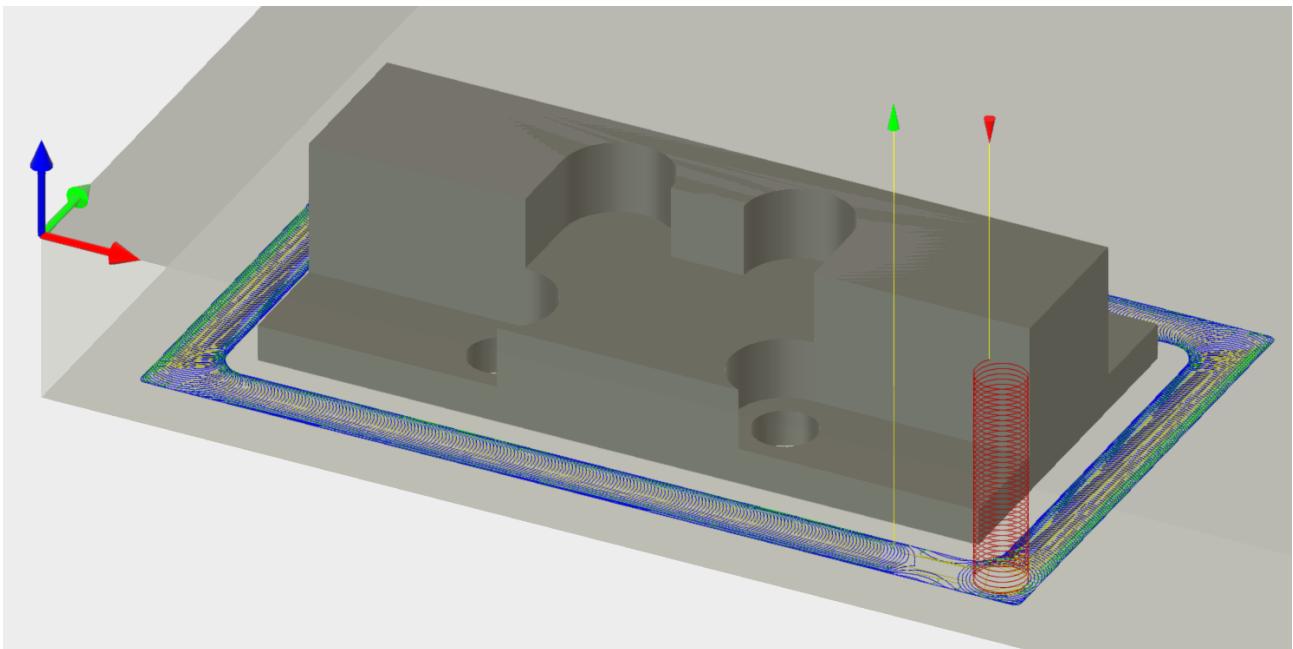
I used a regular pocketing toolpath to cut the four holes, using the same feeds and speeds:



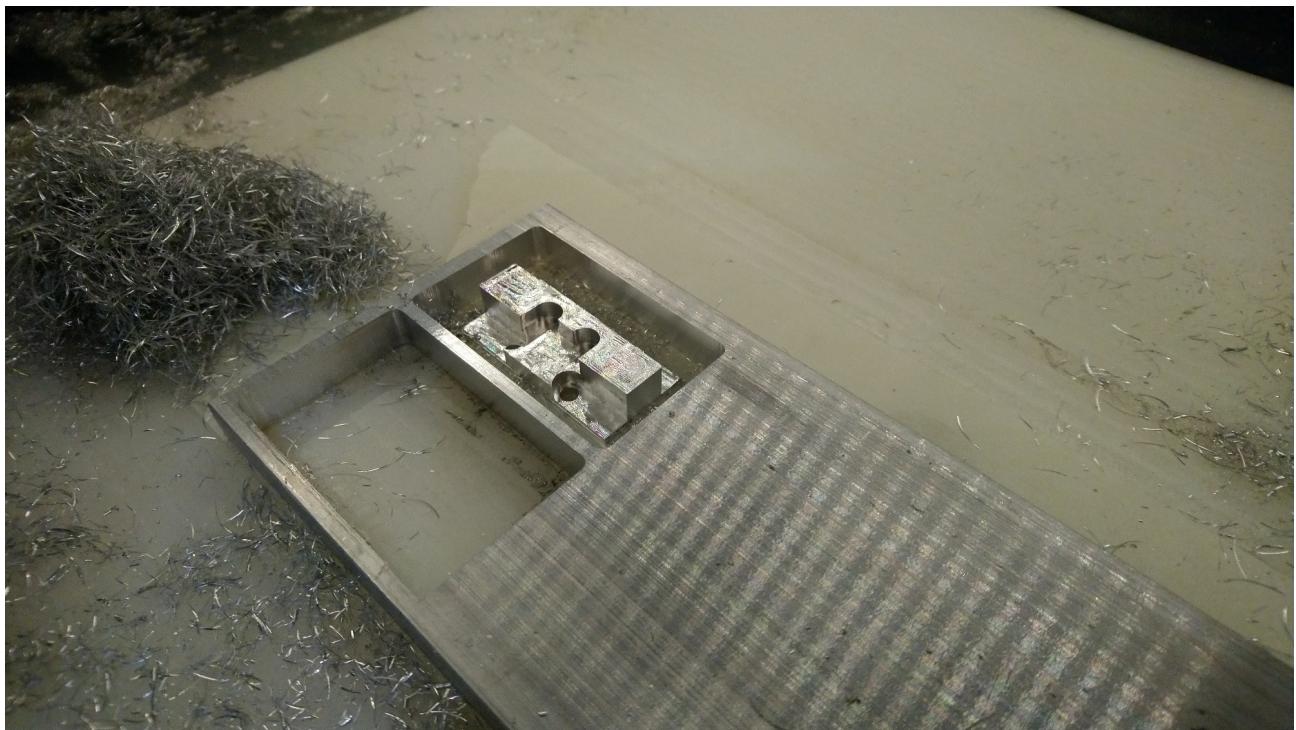
I added a horizontal finishing pass to shave off the 0.02" (axial and radial) excess material left during the adaptive clearing, and get to the final piece dimensions:



Finally, I used an adaptive toolpath to cutout the piece. I could have used a regular profile cut (slotting), but considering the small endmill (1/8") and relatively thick stock (0.4"), this seemed less risky albeit longer to execute:



I used an air jet throughout the cut to help chip evacuation, no coolant. This got me nice long chips, and a decent finish quality:



Retrospectively, I should have used much higher RPMs (while maintaining the same chiploads) to optimize this cut.

Troubleshooting & maintenance

This section touches on a few of the most common mishaps that can happen when working with the Shapeoko, as well as basic maintenance tips.

Basic checks

First, it's always advisable to go back and check the machine operating checklist, it's available at: <https://docs.carbide3d.com/general-faq/machine-operating-checklist>

Hitting the limits

On one end of each axis, the homing switches will interrupt the toolpath if triggered *and if hard limits* are enabled in GRBL, which by default they are not.

On the other end of each axis, since there is no way for the machine to know if it went too far, there are two cases:

- either the **soft limits** in GRBL have been activated and configured to the correct value for your machine, and the job will stop.
- or the soft limits are turned off, and you can get a mechanical crash.

Carbide Motion uses its own hard-coded soft limits during jogging, so if you are using this sender you are better off leaving soft limits turned off in GRBL (first that would be redundant, and second the values set in GRBL may not be consistent with the ones in CM).

If you are using another sender, then by all means do turn on soft limits in GRBL, it's easy:

- \$20 parameter controls whether soft limits are enabled (1) or not (0). Start with soft limits disabled, to be able to jog freely.
- \$130, \$131, and \$132 parameters respectively control X, Y and Z maximum travel : (carefully) jog to the limit of each axis, leaving a small margin you are comfortable with,

write the current axis value in millimeters, drop the minus sign, and set it in the corresponding parameter (e.g. type in \$130=<positive value in mm> in the GRBL console).

- once you are done, activate soft limits, and check that when trying to jog past them, the sender produces an error and stops the movement.

 You should actually TEST how far you can go on YOUR machine, not rely on theoretical X/Y/Z travel values for the Shapeoko. There are a variety of factors that can make these values specific to a given machine. For example, on mine the X travel is limited by the arms of the dust shoe.

Cut depth issues

When the cut depth is off, assuming the toolpath itself is correct, chances are something is slipping in the Z direction (that is arguably the weakest axis on a stock Shapeoko). A few potential reasons are listed below, more or less by decreasing likelihood:

- under load, the pulley may be slipping on the motor shaft if the **set screws** are not tight, so that should be checked/secured first (using a 1.5mm hex key):



(i) The original set screws work just fine for most users, but some use beefier/larger set screws, or replace them with cap screws

- the belts must be **tensioned** correctly, to avoid any slop that could lead to the belt jumping the pulley teeth when a large load is put on that axis. This is a goldilocks situation where the belt needs to be tight enough to avoid this problem, but not too tight to avoid bending the motor shaft.

(i) The usual words to characterize an adequate belt tension are "guitar-string tight". For the Y belts, a good indication of proper tension is that when the gantry

is at one end of the rail, it should be possible to lift the belt a bit, but it should not be possible to slide a pinky finger under the middle of the belt.

- the endmill could be slipping in the collet, if it is not tight enough ("monkey tight, not gorilla tight"), or worse if the wrong collet size was used (e.g 6mm endmill in a 6.35mm collet)
- also, and this applies to all axis, in theory commanding the motor to do N steps will move the associated axis by (close to) $N/40$ mm. But that's only true if the effort put on the shaft is lower than the torque the motor is able to provide. When the forces exceed the max torque of the motor, commanding one step of rotation will result in...the motor staying in the same position, effectively "losing" one step, which then causes a discrepancy between where the machine actually is, and where it thinks it is (as it has no feedback loop to verify if it actually moved), and subsequent cuts will be off by the number of skipped steps.
- finally, if you are using a touch probe to zero Z axis only, make sure it does not sit on its recessed part, which is used for XYZ probing : installing it on a corner and probing Z would result in a Z0 that is off by the height of the probe step.

 If nothing is slipping and the depth error is small, it *might* just be that the Z steps per mm value (\$102 parameter in GRBL) is not right, refer to the [Dimensional accuracy](#) section for more.

Crashing the machine

This will happen sooner or later, and it will very likely be a user error somewhere in the workflow. First, in most cases it's not that big a deal: the steppers have limited power and will stall, or the belts will skip. You should still rush to the emergency stop button but chances are that the machine will not be permanently damaged. Reports from people who actually broke the machine beyond trivial repairs are (very) rare.

Here's a checklist of things to double-check after a crash, while the machine is turned off:

- move each axis manually, and check for slop or any unusual noise/friction.
- check V-wheels are still tight. They are probably the parts that are most likely to have been damaged. It's a good idea to have a spare set available.
- check the belts: if the pulleys skipped hard enough, belt teeth *may* have been damaged (but they are more likely to just break : having a spare roll of GT2 belt is cheap and can come in handy). Pay specific attention to the Z-belt tensioning: a Z-related crash may have loosened the tensioning screw.
- optionally check whether the machine is still square and trammed.

Once everything is checked (and repaired if need be), turn on the machine and check whether homing and jogging still work fine.

Disconnects

If a job stops suddenly for no apparent reason, and commands from the G-code sender have no effect anymore, it is very likely an EMI-related loss of communication with the controller. Most users never experience this failure, but there are a number of user-specific variables that *may* lead to this problem. The main contributors are static discharges (that are more likely to be a problem when the air is very dry, or when making a lot of dust while cutting) and electrical noise (which has many sources, be it mains power or poor grounding).

A variety of solutions apply, the most commonly cited are:

- try different wall sockets if possible, for the machine, the router, and the shop vac.
- grounding the router body/mount (*e.g.* wrapping a copper wire around the router mount and connecting it to a known-good grounding point, on the machine itself or nearby).
- earthing the machine frame.
- grounding the dust collection hose, and/or use an anti-static hose.
- if the air is very dry, spray a little water on the stock, or use a humidifier in the room.
- using a high-quality shielded USB cable, and/or a USB isolator or powered USB hub, or just using a different USB port on the PC running the G-code sender.
- make sure your computer does not put the USB ports to sleep mode after a while. On a laptop, this can happen if running on battery: make sure it runs on power instead, or adjust power management options.

- using a voltage regulator/surge protector between mains and the machine.
- checking if router brushes are not worn out, and replacing them if needed.

 To isolate the source of disconnects, it can be useful to do "air cuts" with the router turned off, then try with the shop vac turned off : if the disconnects disappear, look at grounding the router or the dust hose/shop vac.

Double-checking the toolpaths

If the cut is not going as expected and there do not seem to be any mechanical issues, chances are the mistake lies in the toolpaths themselves. Going back over the toolpath parameters is a first step, then visualizing the toolpath details in the CAM tool is often helpful. This is one area where Fusion360 shines, the toolpath simulation feature is excellent, one can play it in slow motion, and it detects tool collision automatically (assuming the selected tool and tool holder's geometry are modeled correctly)

Typical mistakes include:

- typing the feeds and speeds values incorrectly (or just using incorrect feeds and speeds)
- inconsistency between where the zero point is declared in the CAM setup, and where you zeroed physically on the stock.
- wrong depth of cut setting (which may not be immediately visible on the toolpath preview).
- setting retract/safety height to an excessively large value resulting in the carriage lifting, bottoming out on the stops, then thinking it is much higher than it actually is, plunging deeply into the stock.
- wrong ordering of toolpaths.
- inconsistency between the toolpath and the selected tool capabilities, e.g. cutting deeper than the endmill flute length allows.

Double-checking the G-code

While it is very unlikely that the generated G-code is incorrect if the toolpath is OK, you can still double-check it using a G-code viewer :

- CAMotics is a good option for a standalone G-code viewer, there are many others, some of them online.
 - I often use <https://nraynaud.github.io/webgcode/> and just copy/paste some G-code in there as a quick check.
 - many G-code senders (*e.g.* CNCjs, Universal G-code Sender, etc...) include a G-code preview pane, and systematically checking that everything looks as it should there (*e.g.* dimensions/depth, location of the toolpath versus zero point, ...) before running a job will probably prevent a few silly mistakes, like not running the right file.
-

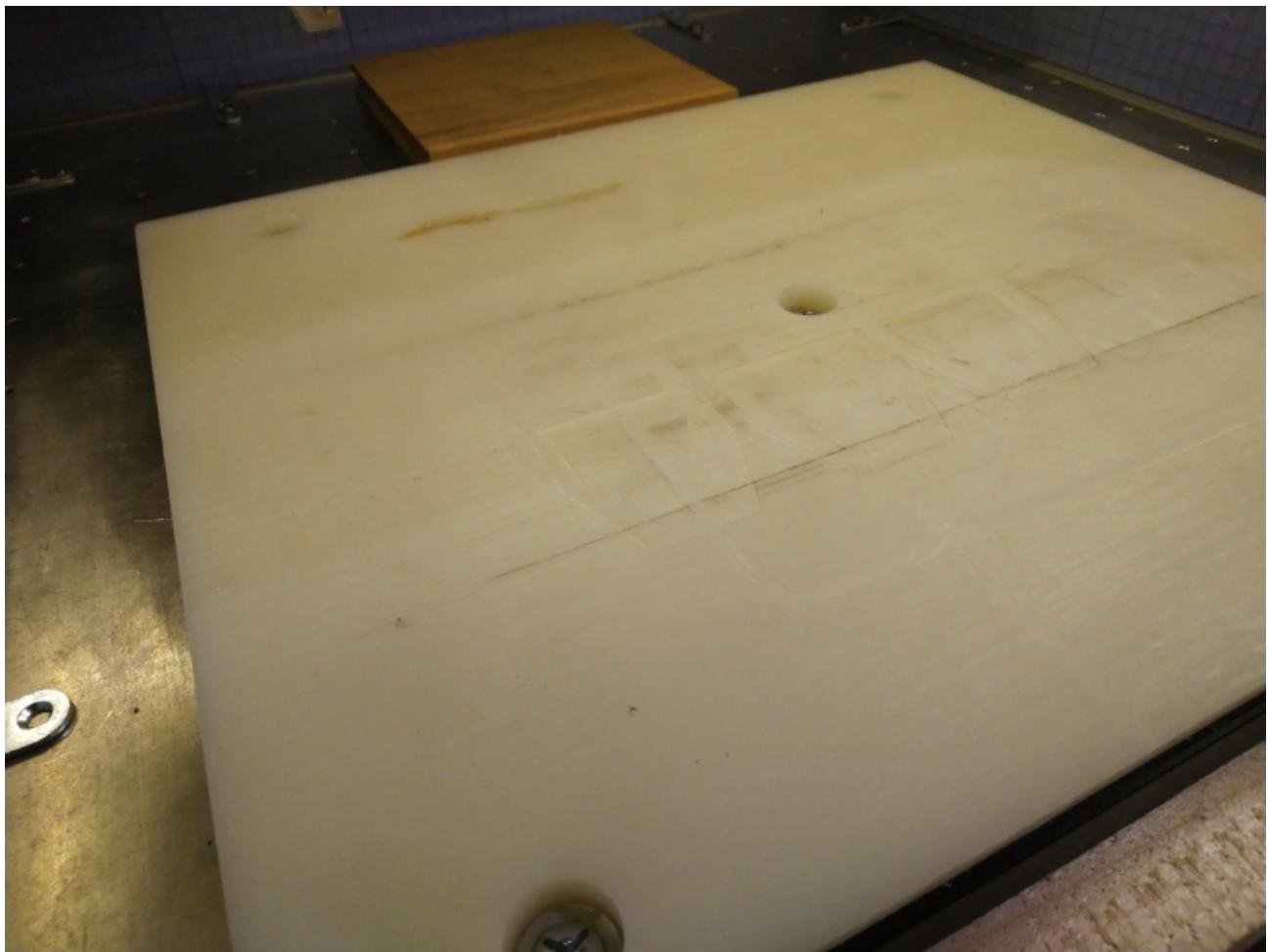
Resurfacing the wasteboard

Some people use the wasteboard as a truly disposable part, i.e. have their toolpaths overcut into it on purpose, while others like to make it last by carefully setting the toolpaths to not cut deeper than the stock. Regardless, sooner or later the surface of the wasteboard will be scarred enough to require a fresh surfacing.

Here's my first MDF wasteboard after a few months of use:

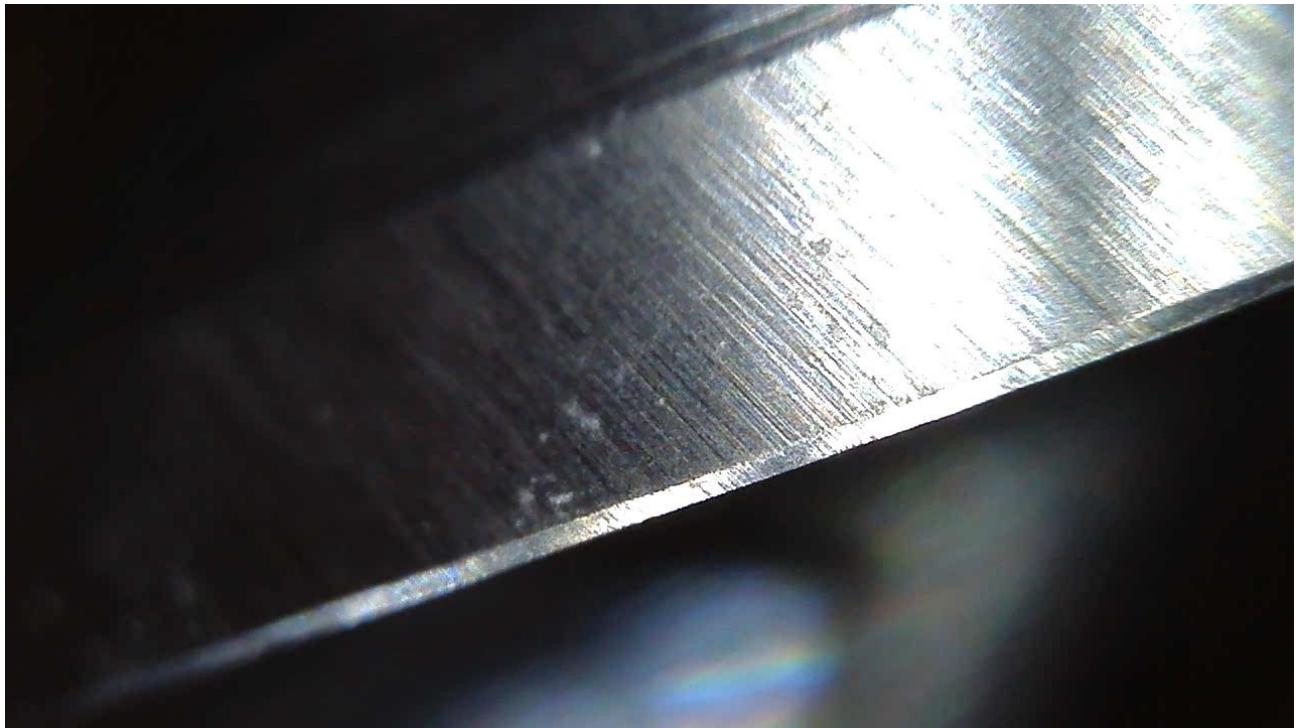


Tape & glue workholding is great, but removing it tends to tear out the MDF surface over time. I later switched to an HDPE wasteboard that is immune to this, but still needs to be resurfaced (much less frequently though):

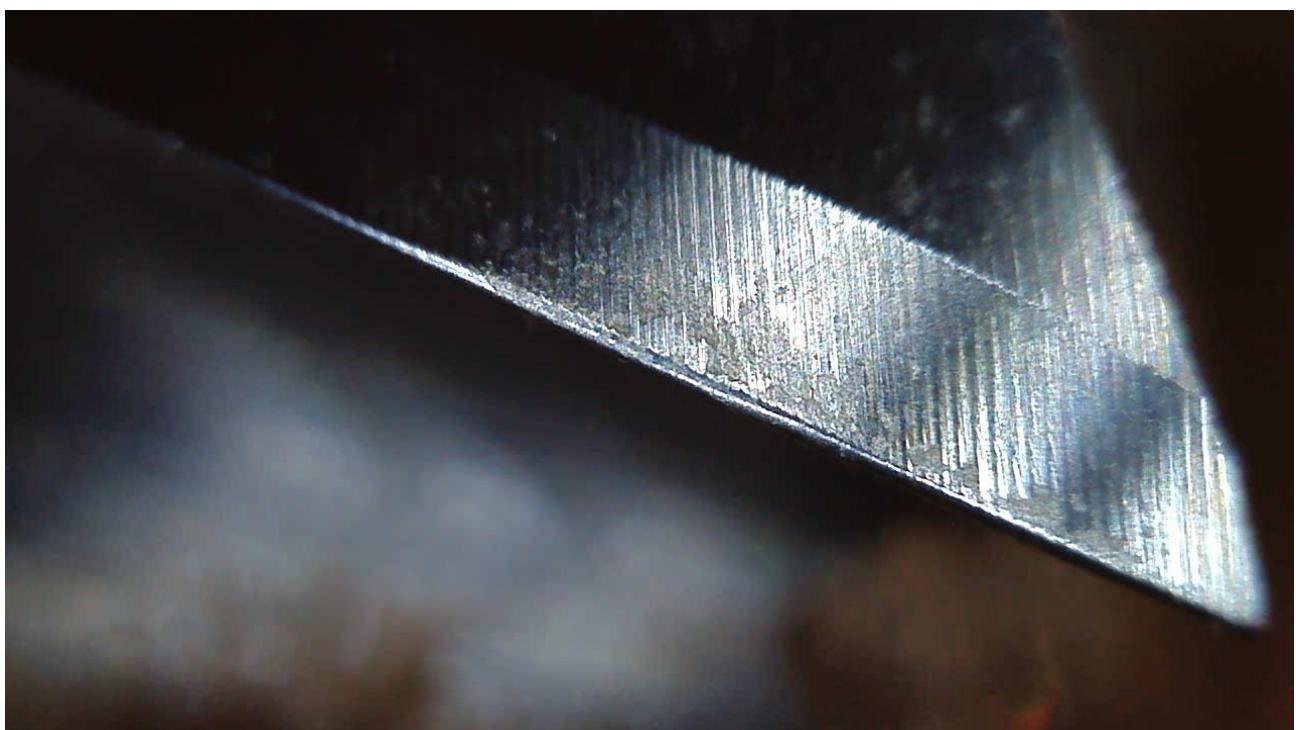


Tool wear

It takes a while to admit that endmills really are consumables (no matter how expensive they can be) and must be replaced when they are worn out, and the difficulty is that this is not often visible to the naked eye. Here's a close-up picture of the cutting edge of a brand new endmill:



Notice the narrow part that runs along the edge (a.k.a. "primary radial relief"), and the much larger surface ("secondary radial relief"). By comparison, here is how my ancient #201 endmill looks after being abused for months:



The primary radial relief is so blunt it is almost gone. At this point, the cutter is rubbing/forcing its way into material as much as it is cutting it.

The big question is how to tell when a cutter has had enough, and everyone will have their own tips but here are a few pointers:

- visual inspection with a USB microscope like the one used for the pictures above is a cheap option (20\$), an even cheaper option is just to use a loupe.
- the sound of the cut and finish quality can be telltale signs (assuming you remember how it sounded/looked with a fresh endmill for a similar cut).
- the look of the chips can be an indication too: a sharp endmill used with proper chipload will make clean chips of uniform sizes, while a dull one will tend to make irregular, torn-out ones.
- if the endmill color changes to a darker shade, it has probably been rubbing for some time, either due to incorrect feeds and speeds or wear. Either way, it's done.

And then again, the level of wear that is acceptable really depends on the material being cut and the required finish quality. I still use the battered #201 pictured above for quick & dirty tests in wood, and get away with it.

i If your endmills wear out very quickly, it is likely that you are not using proper feeds and speeds, and/or you have too much runout.

Cleaning the belts, wheels, V-rails

- a quick vacuuming along the belts length will remove the chips that may have landed there during the cut. It takes 10 seconds, and will prevent the V-wheels rolling over debris or chips getting into the pulleys/idlers.
- it is good practice to wipe off the V rails if crud accumulates.
- cleaning the V-wheels (ALL of them) once in a while helps keeping a clean contact with the rail too. I use a cotton swab placed between the wheel and the rail, and slide the machine manually:



i You could also use custom **wheels covers**: check out David Johnson's DIY engineering site to buy some, he also made them available on Thingiverse so if you have a 3D printer you could print them yourself.

Tightening belts & wheels

From time to time, it is worth checking whether the belts are still guitar-string tight, and the eccentric nuts still keep the V-wheels right against the rails.

- belts can stretch/loosen a bit, and this will not be readily apparent. Do the pinky finger test.
- with the machine turned off,
 - slide the gantry and X/Z carriage manually and check that all wheels are turning: if one of them is not moving, the eccentric nut is not tight enough, or it is worn out.

- Grab the X/Z carriage and (gently) try to twist it front/back and left/right: there should not be any slop.

 A little bit of blue Loctite on the eccentric nuts helps keeping them tight for a very long time

HW upgrades

The Shapeoko is a great machine out of the box, and there is really no *need* to modify it to get good results. No, really, check out Winston Moy's videos if you need convincing. Still, once you are reasonably comfortable with the machine, and how to set optimal CAD/CAM parameters, chances are you may be tempted to optimize the machine itself.

There are many possible hardware upgrades. Some are cheap, some are definitely expensive: this section provides a brief overview of the popular ones. Whether they add enough value for money is highly debatable, and completely depends on your use cases.

Tool length offset probe

This is a device to simplify [multi-tool jobs](#). It will automate the process of re-measuring tool length and adjusting Z0 value accordingly, upon each tool change.

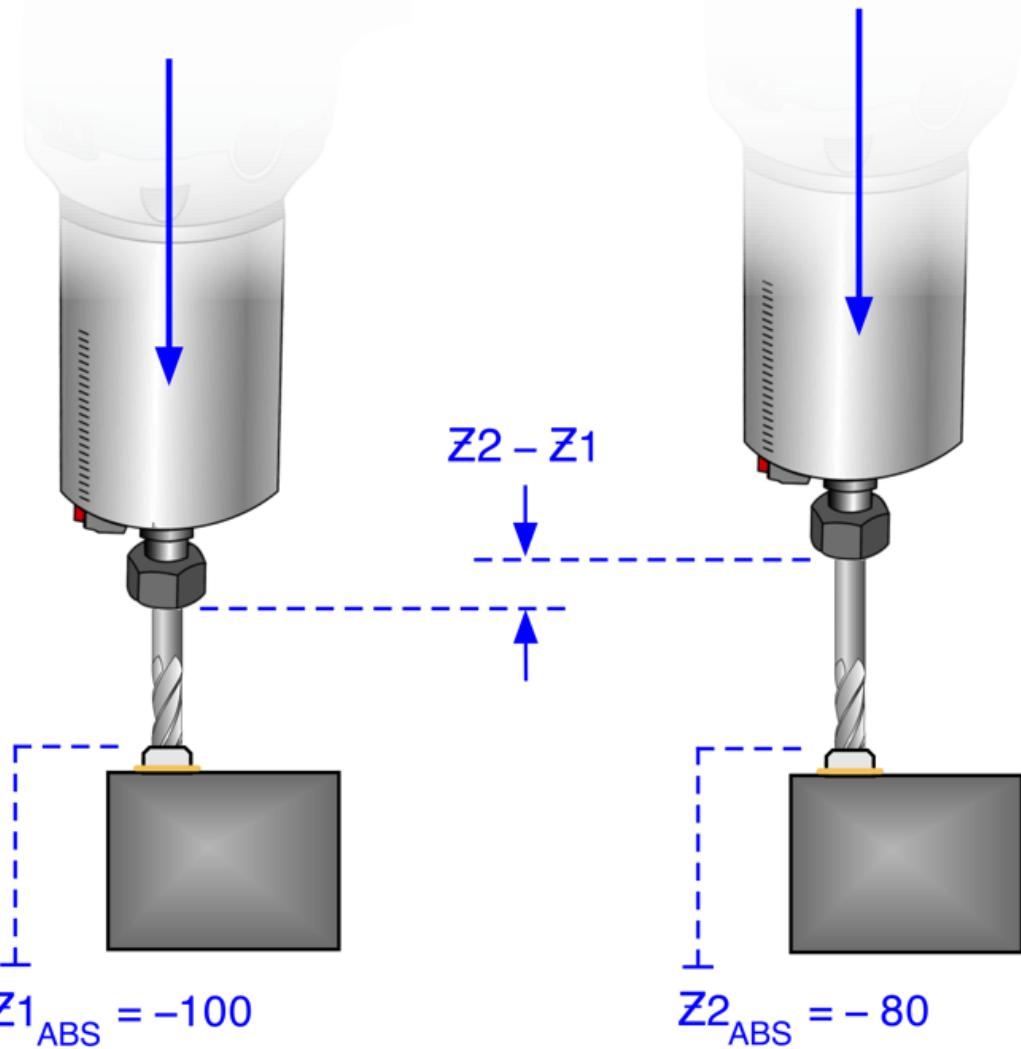
Carbide3D's Nomad machine comes with a built-in tool offset length probe. On the Shapeoko, it's not part of the default configuration, but Carbide3D sells the "**BitSetter**" device as an optional add-on that attaches to the front plate:



The device must be used in conjunction with an associated workflow in the G-code sender, that will launch a probing of tool length right after each tool change.

- (i) This device is connected to the "Probe" input of the controller (yes, the same input using for X/Y/Z probing), so it's basically a spring-loaded push button that activates the probe input upon contact.

The probing works something like this:



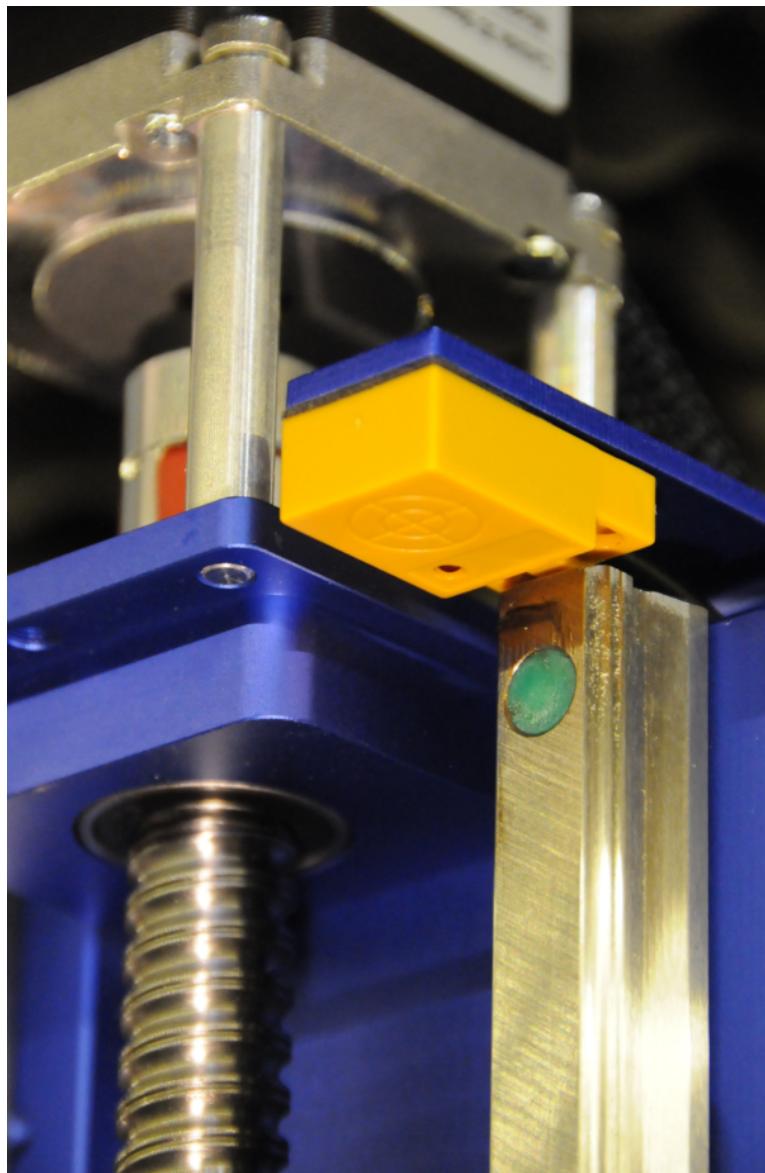
- the user initially sets the zeros normally, which sets Z0.
- the initial probing sequence moves the router over the predefined X/Y position of the tool length probe, then slowly moves the router down until the tip of the endmill pushes the button enough to trigger the probe: at that point, the machine makes a note of the current absolute Z value ($Z1_{ABS}$, arbitrarily shown as -100mm in the example above)
- when the time comes for a tool change, and after the user has installed the new tool, a new probing sequence is launched. In the example above, the second tool sticks out from the collet by 20mm more than the first tool. When the tip of the endmill contacts the probe, the absolute Z value will read a different value than before ($Z2_{ABS} = -80$ mm in this example)
- The machine therefore knows that it must adjust the Z0 by an amount of $Z2 - Z1$ mm.
- The job can then proceed to cut the toolpaths programmed to use the second tool.

- i** In reality, the tool length probing sequence will likely include several activations of the probe in a row, approaching it at different speeds, and then average the results, for optimal precision.

Proximity switches

The standard limit switches are fine, but they have mechanical parts that can wear out over time and eventually fail, which would result in a machine crash when homing. But the main reason to upgrade is to get an (even) better precision/repeatability when homing.

Contactless/proximity switches trigger when a metal object comes within a certain distance of their surface. Since there is no mechanical contact with the moving part, there is no mechanical wearout, and they also happen to be more precise. Here's an example of a proximity switch for the Z-axis:



They can be used as an (almost) drop-in replacement for the original switches, the only difference is that (depending on their technology) they may need an additional lead for power supply, connected to one power pin of the controller board (typically, the 5V pin on the Arduino ISP header, see [Anatomy of a Shapeoko](#) for details).

X/Z axis upgrade

Probably the most popular upgrade is the replacing the stock X/Z carriage with a sturdier one. This is by nature a costly upgrade since it involves multiple large metal parts, linear rails, and a likely ball screw. A few reasons to consider this upgrade path are:

- to eliminate the chances of the Z-belt slipping on the pulley when improperly tensioned.
 - to eliminate the intrinsic (small) front/back slop of the original design, between the main Z-plate and the sliding plate.
 - to support the weight of a spindle, that is typically much heavier than a router.
- Additional springs can be added as an alternative.

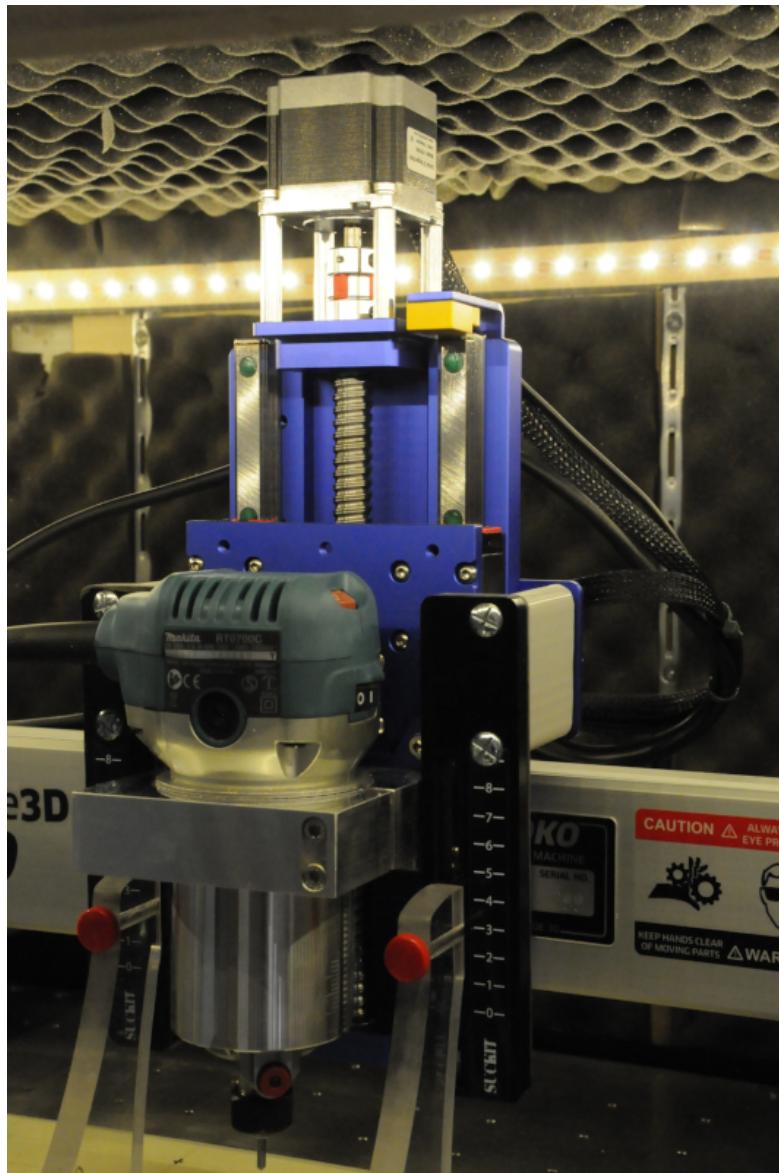
i One noticeable difference between a ball screw driven X/Z axis and a belt & springs one, is that when power is removed from the stepper the router/spindle will stay at its current Z (whereas with it drops naturally with belt & springs). This is usually a good thing (no unintended drop) but sometimes a minor nuisance (Z cannot easily be moved manually while machine is turned off).

The natural upgrade path is to go for Carbide3D's **Z-plus** option, shown here :



The linear rails and ballscrew design make for a robust solution that will be able to handle aggressive feeds and speeds (within the limits of what the V-wheels will allow)

Or for the most demanding jobs you could upgrade to the HD (Heavy Duty) Z axis. Here's an example of an early version of the HDZ installed on my machine:



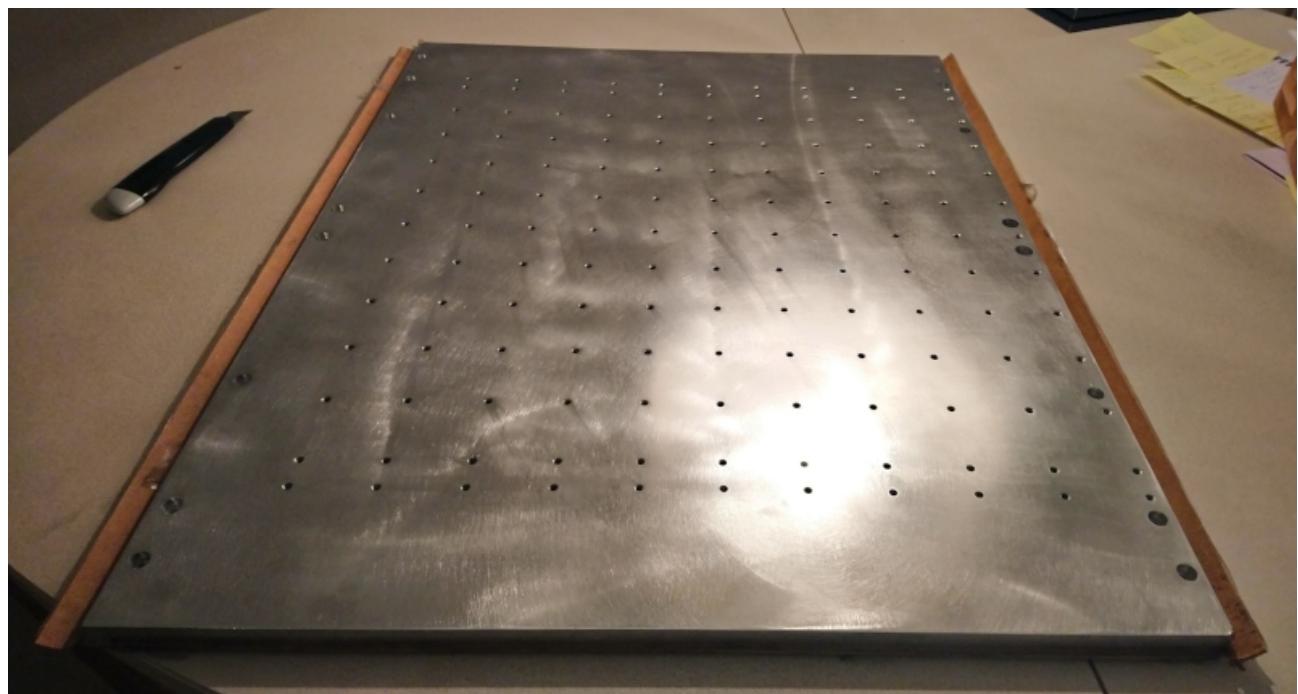
- ⚠ Note that different Z axes have different travel and width, so they require tuning the GRBL parameters accordingly. If using Carbide Motion, this is done through the setup menu. If using other senders, parameters \$132 (max Z travel) and \$130 (max X travel) should be tuned manually to match the setup.

Bed upgrade

There are two main reasons to replace the original MDF bed:

- to match a specific workholding solution (e.g. T-tracks)
- to make the Shapeoko more rigid.

While the T-tracks bed upgrade (that comes as an alternative to the "sea of holes" wasteboard) typically still uses MDF strips, upgrading to an aluminium bed is a good albeit expensive way to improve machine rigidity as well as have a more "weather-insensitive" machine (MDF tends to absorb humidity, which *may* lead to some warping/swelling over time)



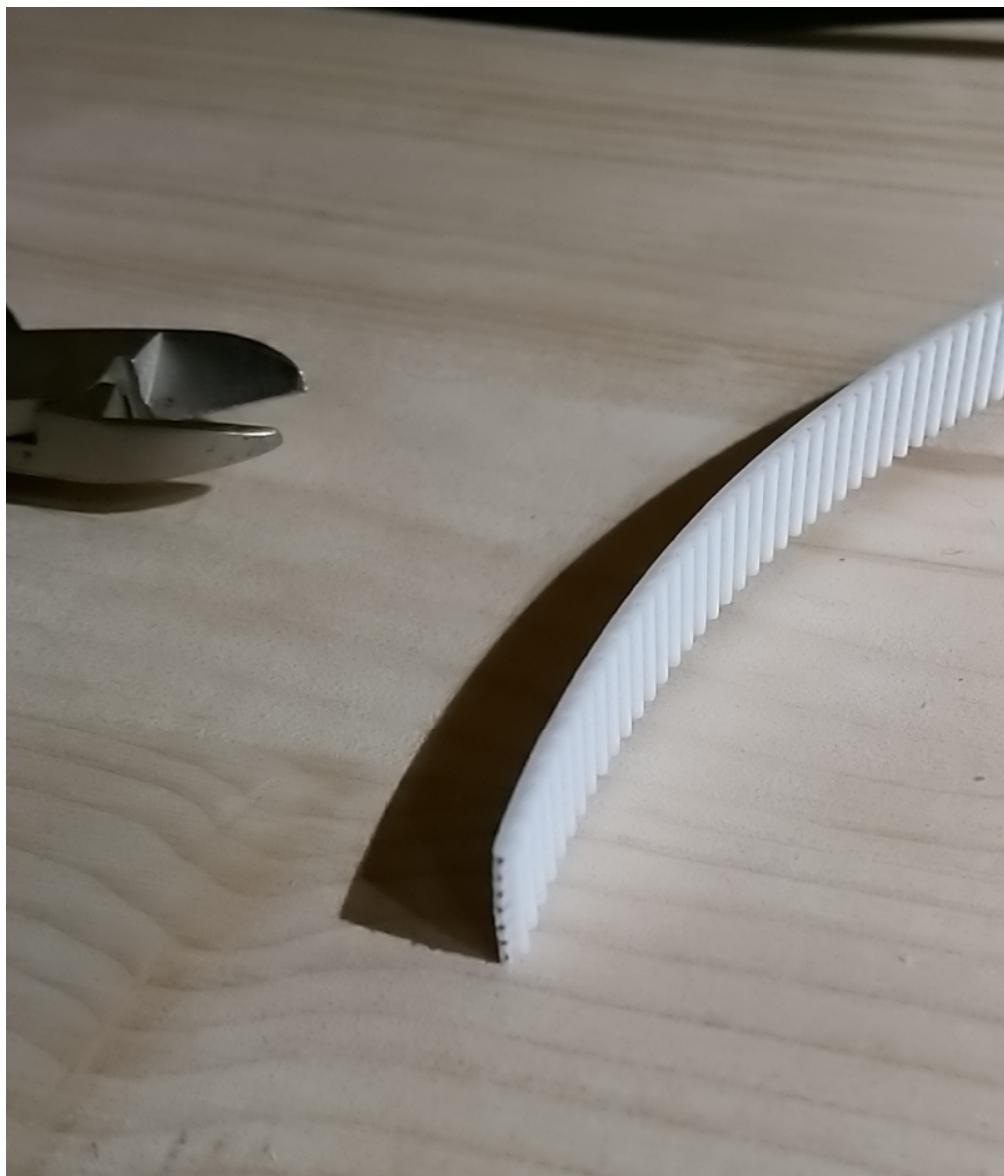
This one is 12mm / ~0.5", the one on Carbide 3D's store is 0.5" thick too.

i This is still only a "nice to have" and expensive upgrade: the large majority of Shapeoko users have an MDF bed and are happy with it

Belts upgrade

This is probably the cheapest upgrade: replacing the original GT2 belts with reinforced ones. The two most popular are **steel-core** belts, and **kevlar** belts.

Here's a section view of steel-core belts showing the embedded steel wires:



Buying several meters of steel-core belts is cheap, will serve as a provision in case the original belts snap, and replacing the belts is very easy (since it does not involve disassembling anything else other than the belt tensioners) so that upgrade is a no brainer if you feel you can benefit from the increased robustness, and possibly from the ability to push the max speed and acceleration beyond the default settings:



It has been reported that the belt stretch factor for those steel-core belts is about half as much as with regular belts, so they may also bring some benefits for precision work.

- i The tight bends around the pulleys is more than what steel belts are rated for, but no one in the community has reported any actual issue so far.

Automatic (router) RPM control

Having to manually set RPMs on the router knob at the beginning of each job can get old, and is also error prone. Luckily the Shapeoko controller board happens to have a "PWM" output that GRBL modulates as a function of the RPM values found in the G-code. So it is possible to feed this signal into a dedicated power controller, that will adjust the voltage applied on the router power leads accordingly, resulting in a specific RPM value.

This requires buying such a power control module, the most popular one is the **SuperPID**. You need to know what you are doing since installing it requires wiring mains.

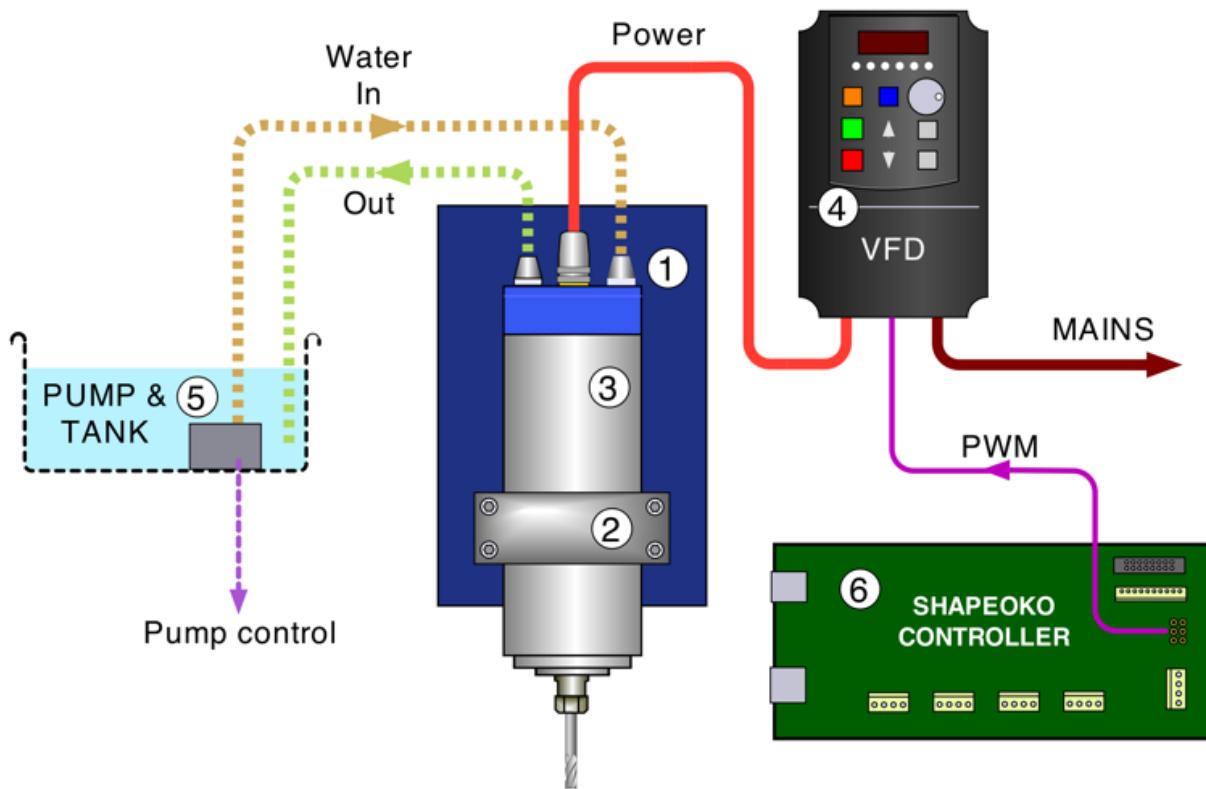
Beyond the automatic RPM control, the added benefit is that this allows the router to operate at a lower RPM value than the minimal knob setting, *e.g.* to run at 5000 RPM on a Makita that normally mins out at 10,000RPM.

Spindle upgrade

The Shapeoko uses a trim router by default for cost/convenience reasons, but most higher-end CNCs use a spindle instead. Some of the main benefits are:

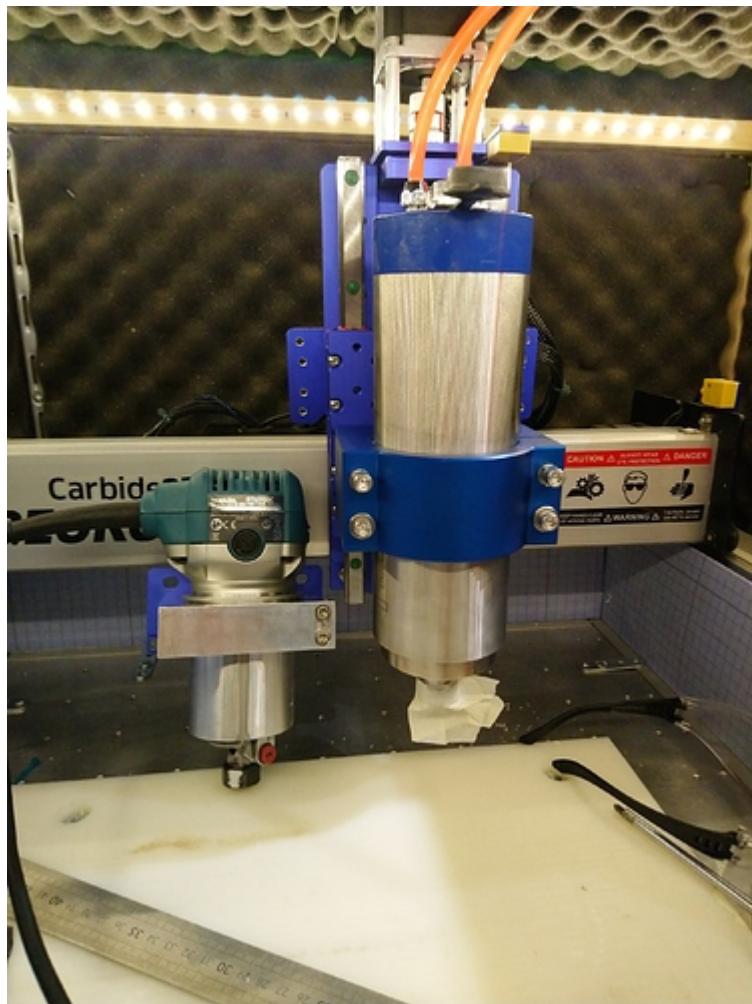
- a spindle can run at lower RPMs than a router (some models can run at much *higher* max RPMs too), and usually has higher torque. Since a spindle needs a dedicated controller to run anyway, the automatic RPM control described above is a given.
- the runout is smaller.
- it is WAY quieter than a router, especially when using a water-cooled version.
- there is no need to change bushing, so basically no maintenance.
- it usually supports "ER" collets, which come in much more varied sizes than trim router collets.

Here's an overview of the key points to consider for the upgrade:

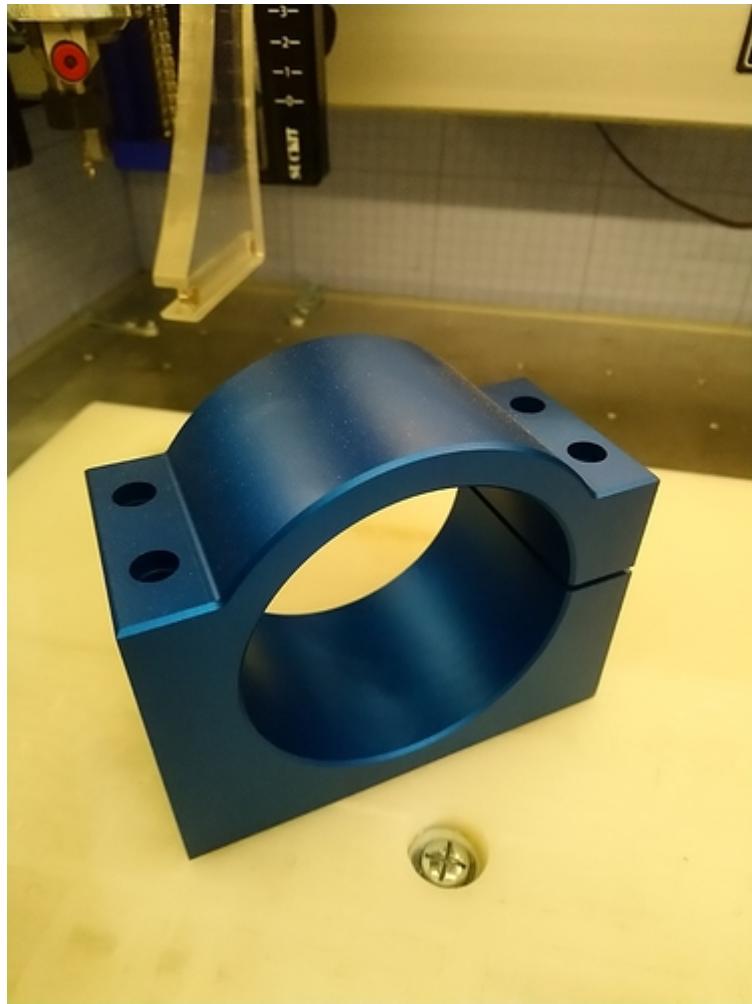


1) The Z axis capability. Spindles can be HEAVY, especially 2.2kW ones. The stock Z axis will not be able to cope with the heavier models, but should be fine for the lighter (800W) ones. The HDZ and Z-plus axis, with their ballscrew design, will handle all models.

For reference, here's the relative size of a 2.2kW spindle, next to the Makita trim router:

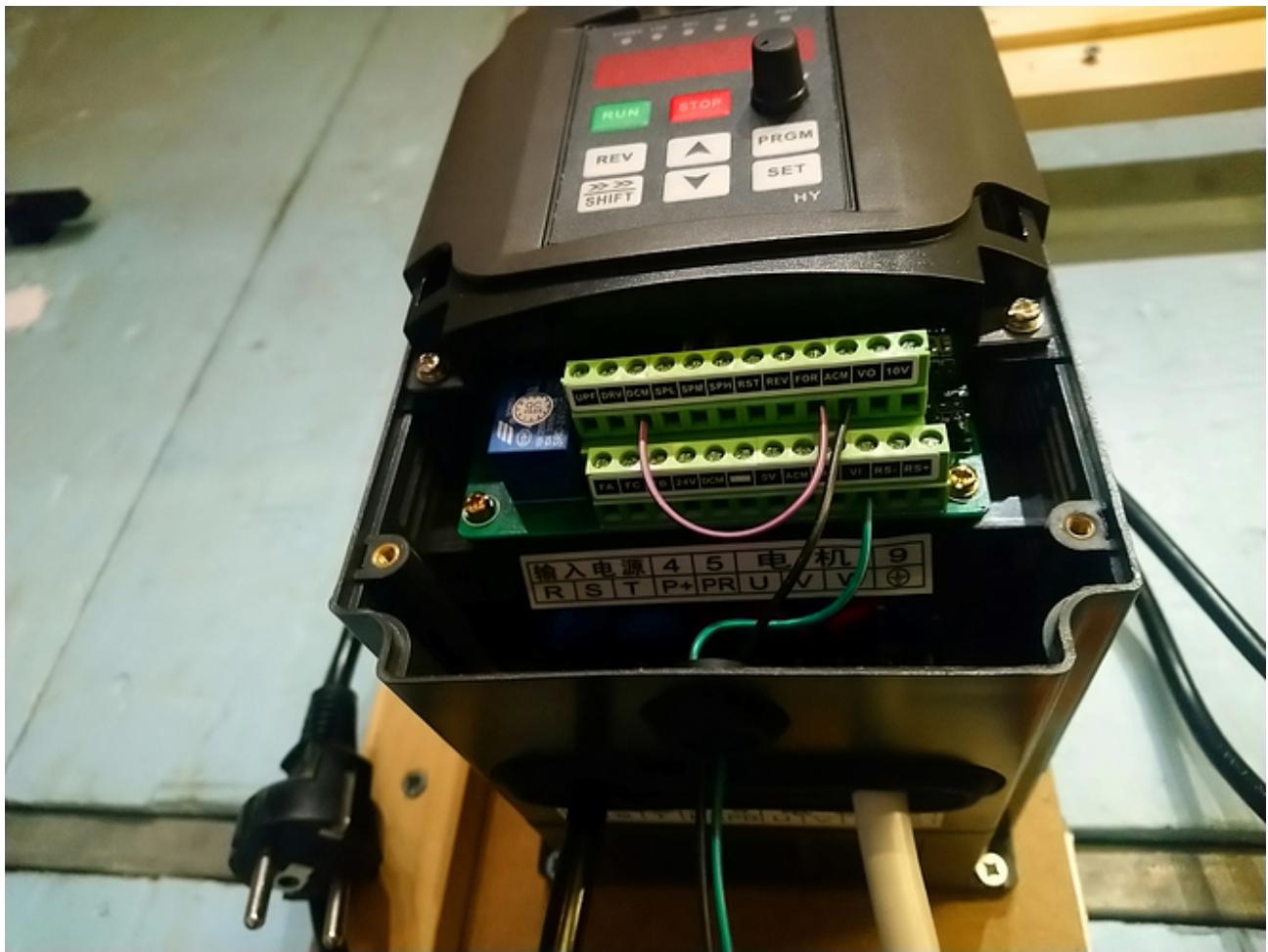


2) The router mount diameter: chances are, the stock router mount diameter will not match the diameter of the selected spindle. Spindle kits often include a mount, but it's usually very bad quality and not easily adaptable to the Shapeoko's Z axis. You are probably better off buying a spindle mount from Carbide3D's store. Here's a view of a 80mm one for reference:

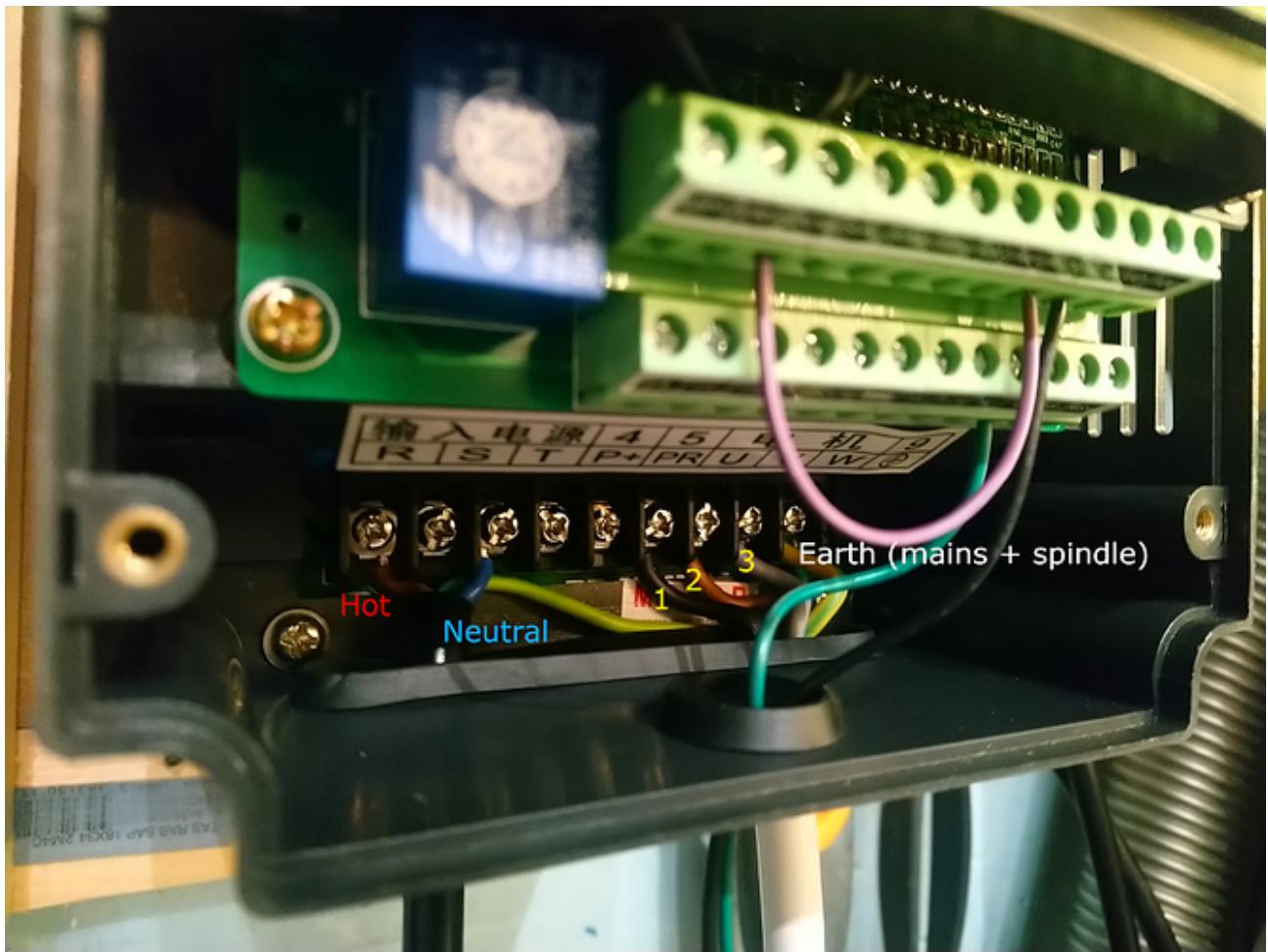


3) Then there is the choice of the spindle itself. They come in two types: **air-cooled** and **water-cooled**. The former is easier to install, but noisier. The latter is extremely quiet, but requires one to install an external water-cooling system.

4) A "Variable Frequency Drive" (**VFD**) is required to power the spindle and control its speed. The wiring of the VFD depends on the model (spoiler alert: chances are you will buy your spindle kit from China, documentation will be sub-par, but the community is here to sort it out). In the example pic below, the green wire is the PWM signal from the Shapeoko, the black wire is the Ground signal, and the purple wire is just telling to VFD to force the rotation direction:



Beyond that it's just a matter of connecting mains, and the four wires to the spindle (1,2,3 and Earth, also called U,V,W and Earth):



Make sure the spindle's **ground/earth pin** is actually connected inside the spindle to the body:



Sometimes it is left unconnected and then you are bound to get static build-up and possibly EMI issues

Finally one must configure the (many) parameters on the VFD, they are usually called "PDxxx" parameters, and depend on the VFD type/brand, so you will have to refer to your documentation.

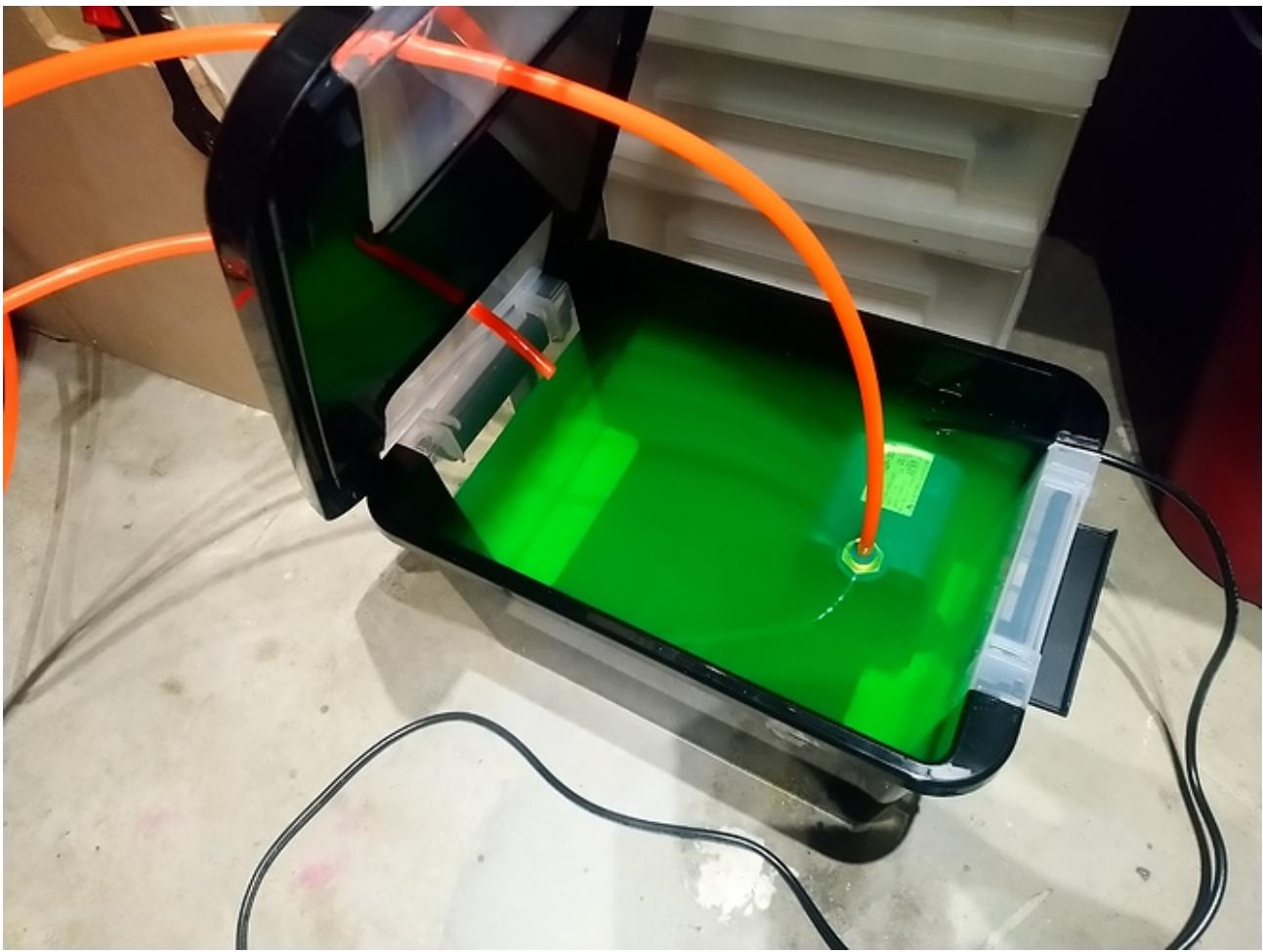


Huanyang VFD owners may find this discussion helpful :

<https://community.carbide3d.com/t/vfd-parameters-huanyang-model/15459/7>

5) For water-cooled spindles, one then needs to decide what type of cooling system to use. There are basically two popular approaches:

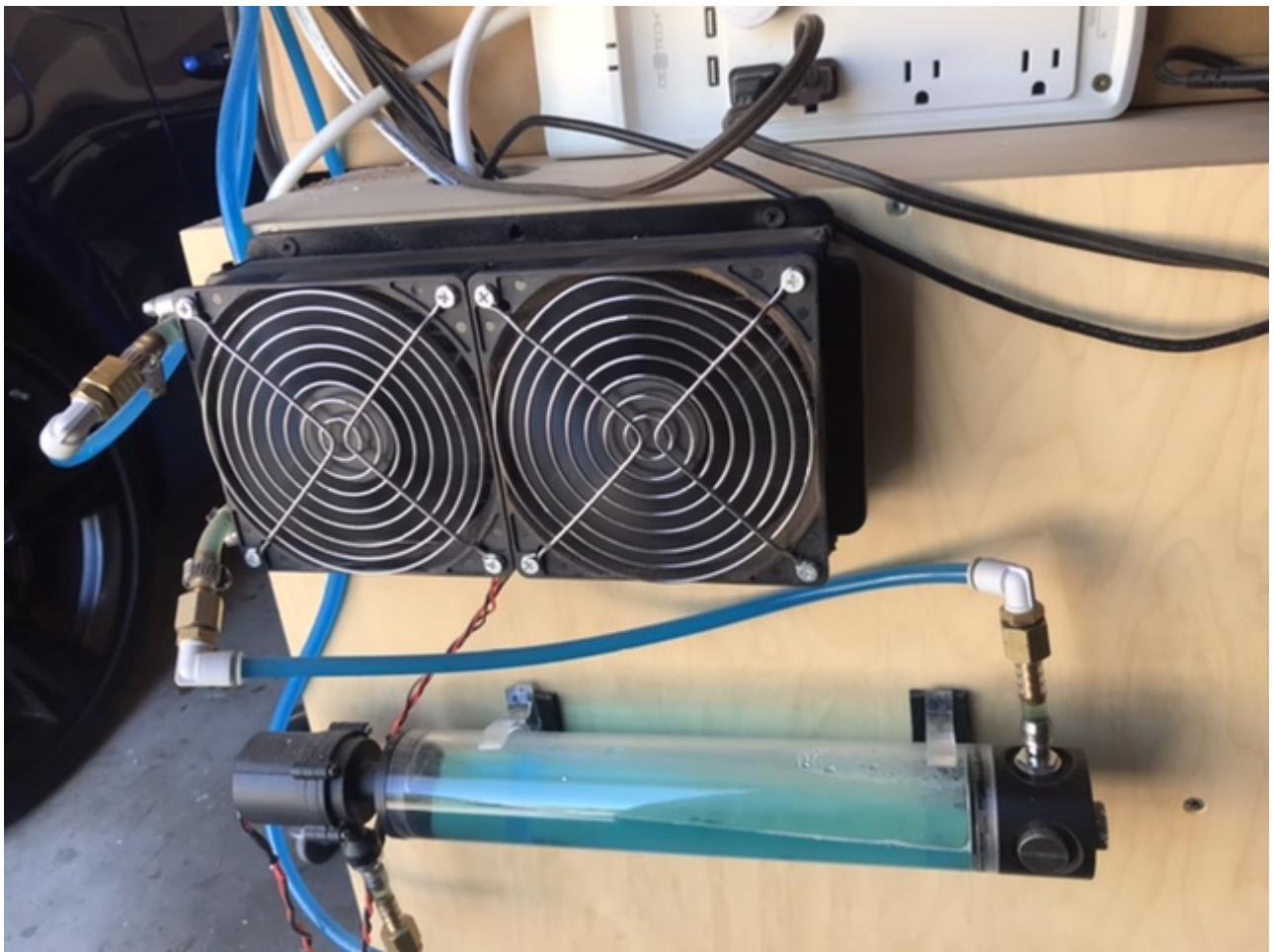
- one can use a large container as a tank, and install a **submersible pump** in it:



- ① The container should be closed/sealed when operating, or dust/chips could find their way through the cooling system.
 - ① With a large enough tank, the volume of coolant alone is enough to keep the temperature low enough.
 - ① One should not use plain water as the coolant, the tubes are bound to become covered in fungus after a while. A common trick is to use automotive coolant, or other chemical products, to keep the cooling system clean
- or one can purchase a **closed-loop** cooling system:



and install it on the machine:



i A section of the cooling loop should have fans, to keep the coolant temperature down.

6) While the spindle speed can be set manually at the VFD, it is very convenient to connect the PWM signal from the Shapeoko controller board to the VFD, and configure it such that it will control spindle speed automatically based on that signal. That way, there is no need for any manual action anymore: the Shapeoko controller will adjust the PWM signal based on the RPM value it finds inside the G-code files.

i Final tip: to optimize the life and precision of your spindle, it's useful to run a "spindle warmup" routine at the beginning of a session in the shop. The warmup routine consists in initially running the spindle at very low RPM for a little while, then gradually ramp up the speed, to the maximal value, over about 10 minutes. Below is a simple G-code macro to do just that.

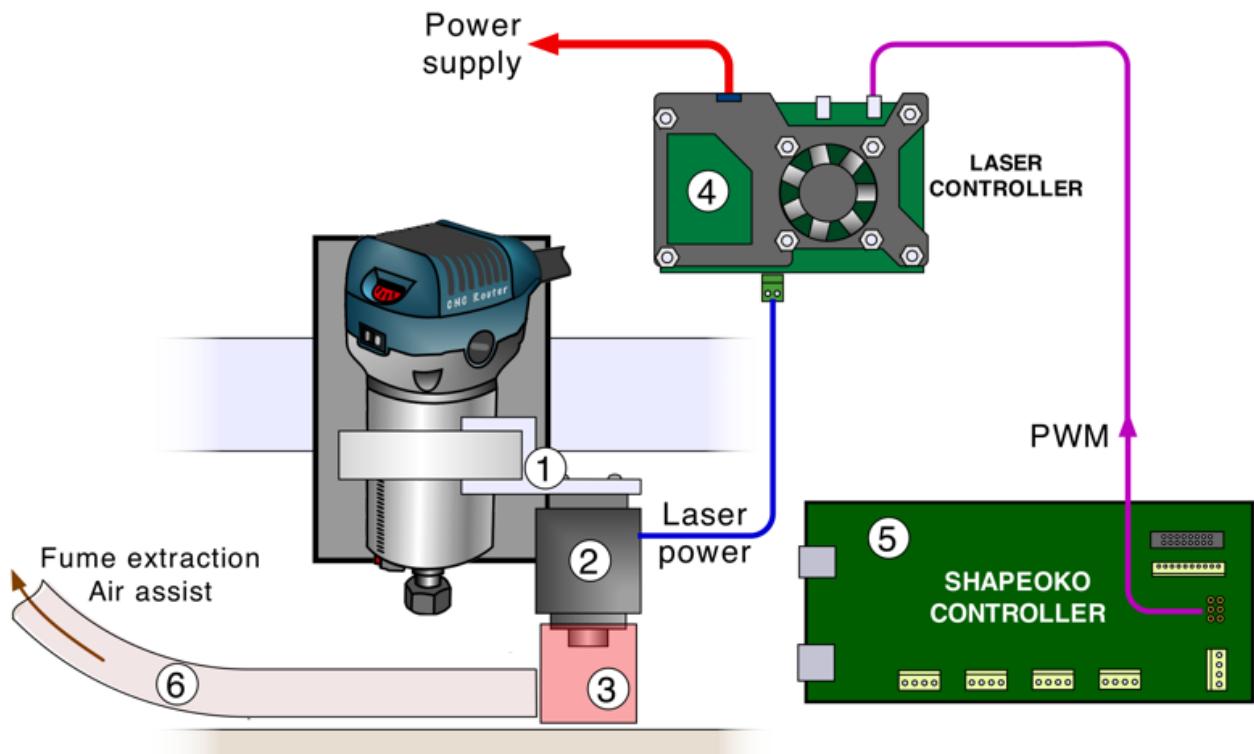
```
1 M3S2000
2 G4 P60.0
3 M3S4000
4 G4 P60.0
5 M3S6000
6 G4 P60.0
7 M3S8000
8 G4 P60.0
9 M3S10000
10 G4 P60.0
11 M3S12000
12 G4 P60.0
13 M3S14000
14 G4 P60.0
15 M3S16000
16 G4 P60.0
17 M3S18000
18 G4 P60.0
19 M3S20000
20 G4 P60.0
21 M3S22000
22 G4 P60.0
23 M3S24000
24 G4 P60.0
25 M5
```

Laser upgrade



If you are considering adding a laser to your Shapeoko setup, whatever you do put safety considerations at the very top of your priority list. Eye safety is priceless, so if you are going to do this, you need to have SEVERAL layers of safety measures in place. And the fire hazard is obvious.

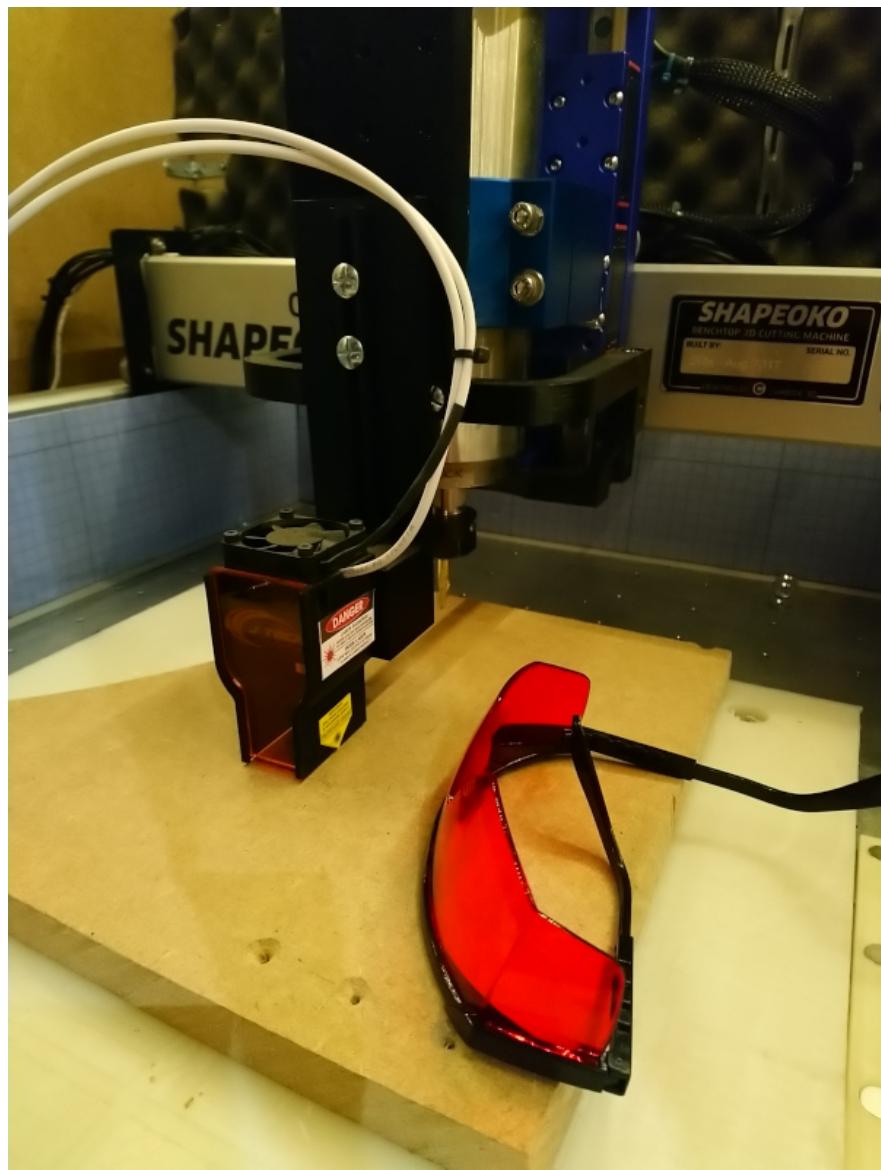
It's tempting to leverage the X/Y capability of the Shapeoko, to move a laser module around and engrave/cut the surface of parts. A typical setup will look something like this:



- 1) The laser module is usually attached somewhere on the router mount, in such a way that it can be installed/removed easily, when switching back and forth between CNC operation and laser operation.
- 2) One must select a laser module which matches the intended use. Modules mounted on Shapeokos are typically in the range of [2-10W] power, which grants the ability for engraving, and cutting thin and soft material (think plywood). If your main goal is to do a lot of laser cutting through thick/hard material, you should probably consider a standalone laser cutter instead.
- 3) The laser beam area should be covered by a **shroud**, filtering the harmful laser light. This is your first line of defence, so think twice before you decide to not have one. If not at the module level, it could be done on your enclosure's front window. The second line of defence is obviously to **use laser goggles at all times when operating the laser**.

! One might think that since the laser is always pointing down towards the material, this is inherently safe. But dangerous reflections are a thing, especially when engraving metals.

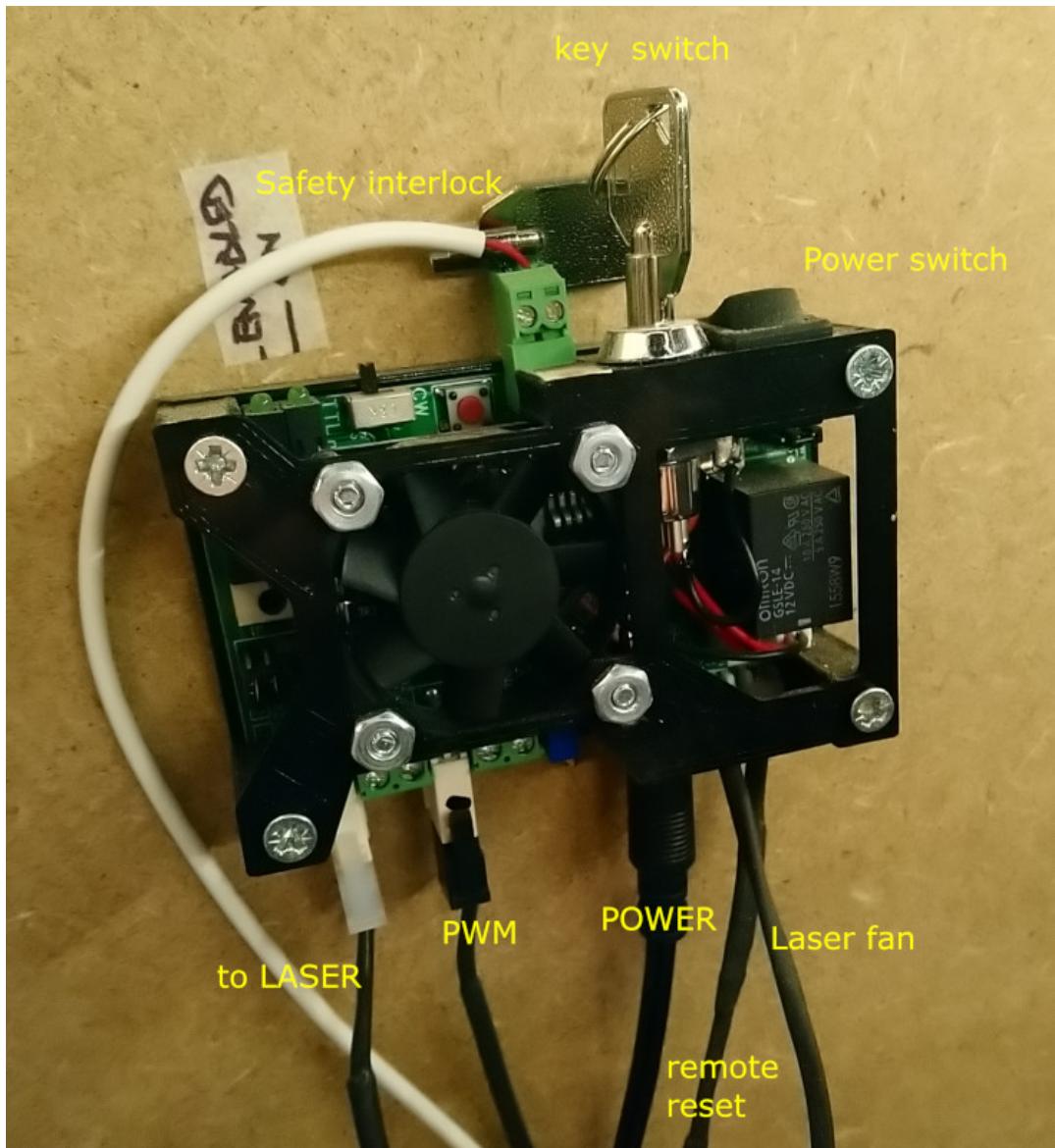
Here's what my 4.8W laser setup looks like:



4) The laser is connected to a controller, that will usually:

- provide and modulate its input power based on an input analog signal (PWM)
- power the laser fan
- implement a variety of safety features (switches/interlocks)

Here's a picture of my controller (from the J-Tech laser kit):



5) The PWM signal driven by the Shapeoko controller will be used to modulate laser using G-code instructions. GRBL generates the PWM signal based on the requested spindle speed value, from 0V for 0 RPM, to 5V for the max RPM value configured in GRBL parameter \$30. When using the Shapeoko in laser mode, the software generating the G-code will make use of that capability to control the laser power.

6) While the laser burns the material, it will produce fumes, which is bad for a couple of reasons:

- fumes are toxic, more so on some materials.
- fumes get in the way of the laser, reducing its efficiency and precision

So it's a good idea to add a ventilation system to extract fumes.

A final word on safety. It seems to me that the minimum one should do to use a laser module safely is:

- have the **laser power interlocked** with the presence of whatever physical protection/cover you are using: it must be impossible for power to get to the laser (voluntarily or by accident) if the protection is not present.
- wear proper **laser goggles** at all times as soon as the thing is POTENTIALLY turned on (that includes moments when the power is off, but under the control of the machine i.e. the G-code/controller). Never, ever trust a single element of software or hardware alone to keep you safe.
- **stay by the machine** while it is running. Seriously, don't burn down your house for a laser cut gone wrong.