CNC

COOKBOOK'S

G-CODE

COURSE

VOLUME 1: G-CODE BASICS

TABLE OF CONTENTS

Here are the articles available for maximizing your g-code proficiency and providing you with a little CNC programmer training. Included in nearly every article are examples using our CNC Programming Software, G-Wizard Editor.

INTRODUCTION

Looking to learn CNC G-Code? Need a quick and easy G-Code Tutorial or G-Code Course? Want some easy G-Code Training? Maybe you just want to learn more about a specific G-Code related topic or see particular G-Code examples. If so, you're in the right place with the CNCCookbook CNC G-Code Course. It's free, it's easy, and it's chock full of good information. This page is the syllabus. The articles are all listed below. There's no need to register, just get started learning at your own pace.

WHAT IS G-CODE?

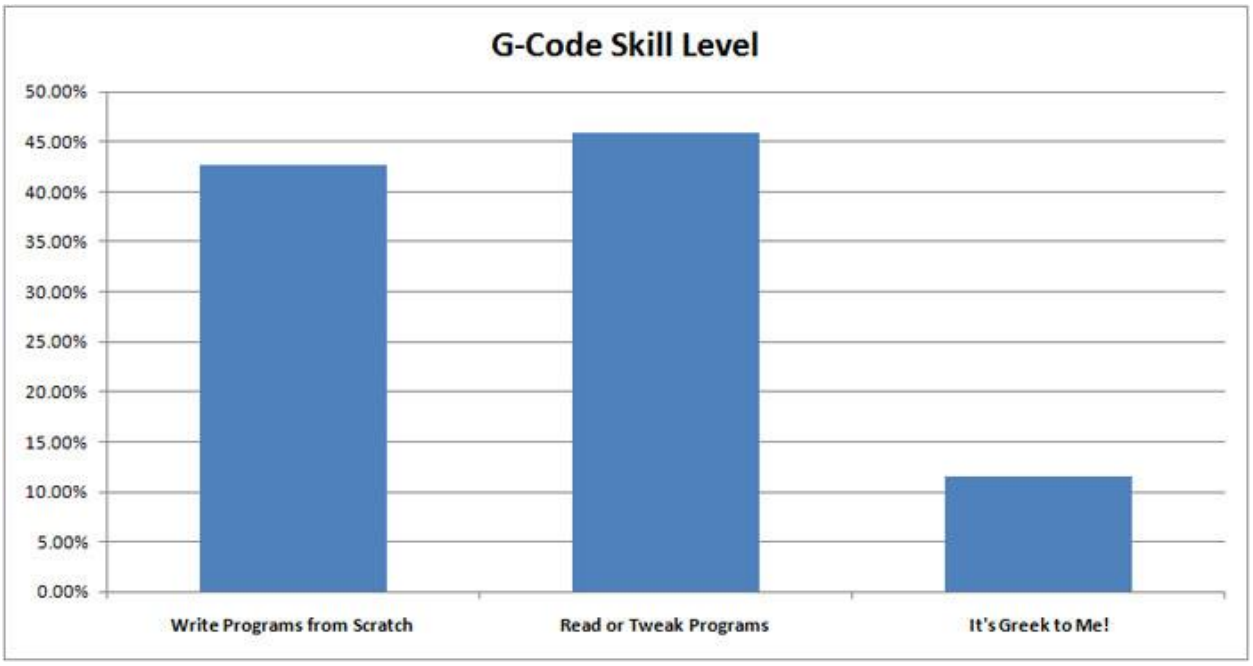
G-Code is the language used to control CNC machines. It's one type of CNC programming that CNC programmers use, the other type being CAM programming. Your machine's CNC controller probably executes g-code, although there are other possibilities--Heidenhain, Mazak, and others have proprietary formats. Some machines with proprietary formats can also run g-code. It is the Lingua Franca (working language) of CNC.

In order to make a part on a CNC machine, you tell it how to make the part using a G-Code Program.

WHY LEARN G-CODE?

Every CNC machinist should know g-code. If you're interested in CNC and machining, you should too.

We recently did a survey to assess the g-code skills of our readership. You should not be suprised to learn that many are quite proficient with G-Code:



We were impressed at how many readers can write g-code programs from scratch. In fact the overwhelming majority read, write, or tweak programs on a regular basis. If you're not yet able to do that, you need to learn. These articles are CNCCookbook's free course in g-code. No matter what stage in g-code learning you are at, you will find the tools to advance to the next stage in these articles. Check them out--it's easy to improve your g-code proficiency and well worth the productivity gains.

WHAT'S THE BEST WAY TO LEARN G-CODE?

A little bit at a time, trying out the examples, and in a continuous stream. In other words, find yourself a complete course like this one, start knocking out the lessons, work the exercises, and keep at it. Work at your own pace, and don't move on to the next lesson until you've worked the exercises and understand the material.

Working the exercises is a whole lot easier if you've got some software to help you play with g-code. That's what our G-Code Editor software is all about. It simulates g-code as well as decoding it for you. You can try out different g-codes and see visually what they do. Experimenting is one of the best ways to get a good grasp of g-code. At the end of each section is a Quiz to test your skills. Take the quiz and use the links on the

questions to go back and review anything you missed so you'll be solid before continuing to the next section.

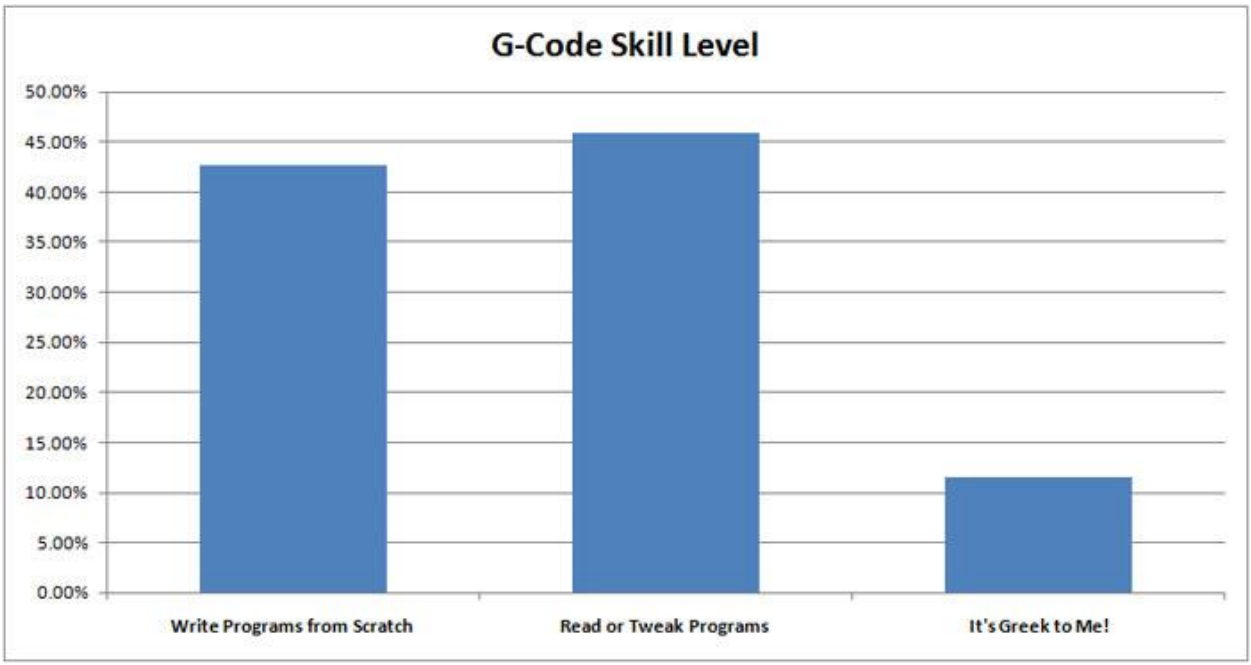
We encourage you to sign up for the free 30 day trial--that's plenty long enough to work through all the lessons free of charge. You'll find G-Wizard Editor not only makes it easy to work with g-code and try the exercises in the course, but it has many features designed to make understanding the g-code easier. For example, it offers "Hints", where it takes each line of g-code and explains in plain English what that code does. There's nothing else like it available anywhere. For more information, **visit the GW Editor Home page**.

The second thing you should do aside from following a course and using a simulator like G-Wizard Editor is to start following some articles about CNC. Getting some random inputs about a variety of topics is another way to help the juices flow. You'll see things that raise questions and get you thinking about the basic concepts in new ways. This helps all of the ideas to connect better. To get a good source of such inspiration you could hardly do better than to subscribe to our own CNCCookbook Blog. We're by far the largest CNC blog on the Internet in terms of readership. We post articles for all levels of knowledge. Signing up is easy--you'll automatically be added to the list if you start the 30-day trial of G-Wizard Editor. Or you can join 50,000 other CNC Enthusiasts by visiting our signup page.

INTRODUCTION FOR BEGINNERS

WHY LEARN G-CODE?

Every CNC machinist should know g-code. We recently did a survey to assess the g-code skills of our readership. You should not be suprised to learn that many are quite proficient with G-Code:



We were impressed at how many readers can write g-code programs from scratch. In fact the overwhelming majority read, write, or tweak programs on a regular basis. If you're not yet able to do that, you should learn. If you're at the "Read or Tweak" stage, you should advance to being able to write programs from scratch. Armed with a good understanding of g-code, you can:

- Avoid having to run back to your CAM program when simple changes to the g-code would do the job.

- Learn how to improve the g-code the CAM program puts out for better results.

- Understand better how to tweak your CAM software's post processor so it produces better code from the start.

- Get a second opinion on the CAM's g-code before you find out something's wrong at the machine.

- Make it faster and easier to fix the g-code when you run into a problem due to a bug in the CAM or post processor.

- Create quick and dirty g-code programs that allow you to get on with machining faster without having to sit down with your CAD/CAM.

- Develop a greater facility for working at the console of the machine directly.

These are all valuable skills that increase your productivity as a CNC machinist. According to the survey, many of you have already figured that out!

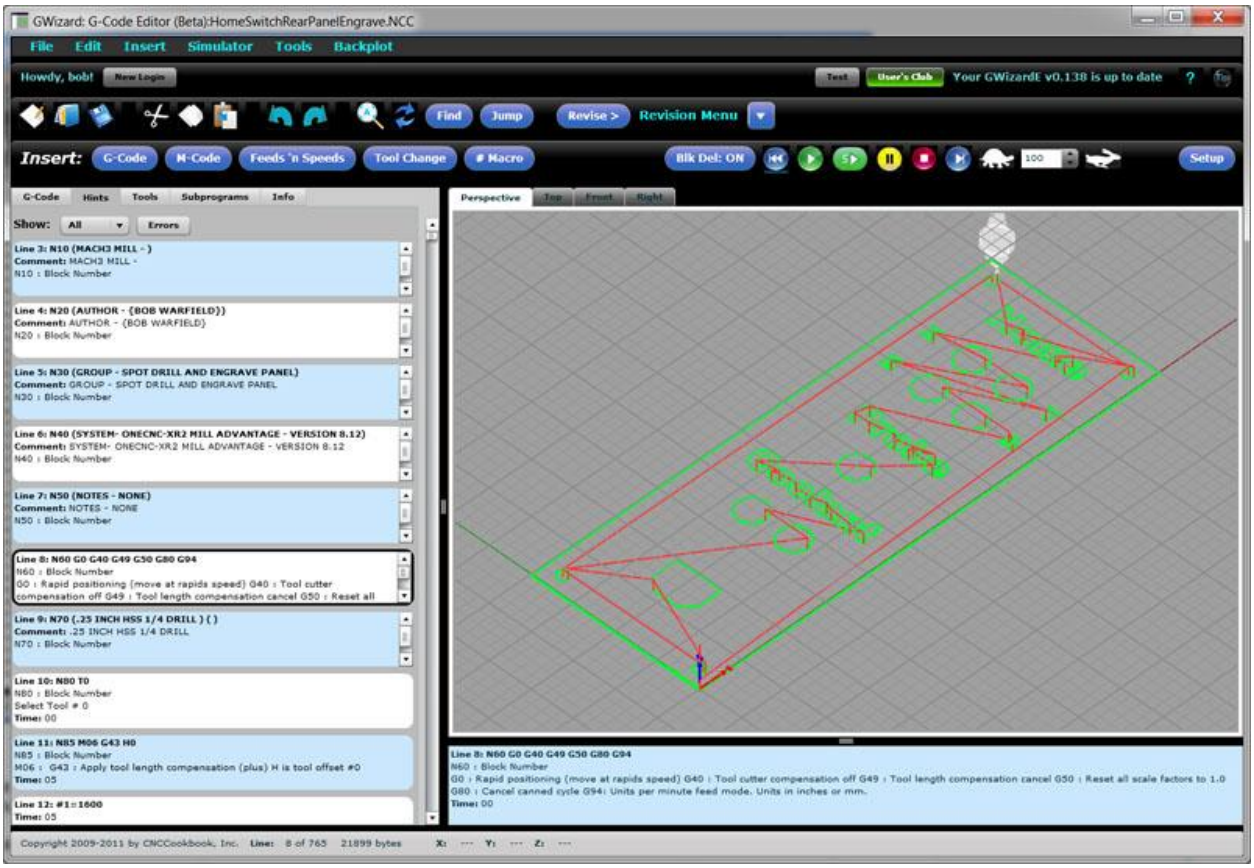
G-CODE FROM THE MACHINE'S PERSPECTIVE

There are a handful of basic concepts you should know before we dive into what individual g-codes do and how to use them. Most importantly, you must understand what a computer language is and how to think about it. G-Code is a computer language aimed at telling your machine what to do. The trouble with computer languages like g-code is that unlike people, machines are very literal. They assume you know exactly what you want, they don't question you about it, and they immediately try to comply, even if that means hurting themselves!

For that reason, you need to adopt an approach of being protective of your machine, perhaps even over protective bordering on paranoid. It will do exactly what you tell it to even if that means rapiding a spinning cutter with 20HP behind it directly into your expensive fourth axis and destroying it. There's no need to be afraid of g-code, but it is important to understand it and to respect it in order to be successful. Getting to that stage is another excellent reason to spend time learning the g-code. It will give you insights into your machine that enable you to extract more performance and to protect it better from accidents.

G-WIZARD G-CODE EDITOR

Speaking of understanding g-code and protecting the machine, throughout this course we will be setting up to do exercises using the G-Wizard G-Code Editor. Think of it as a convenient etch-a-sketch on which you enter g-codes and can immediately see the toolpath that results. This makes learning a lot faster and easier when you have that immediate response versus trying to plot things on graph paper or work with your machine controller. In addition, you can make your mistakes on a simulator rather than with the actual machine until you get a lot more comfortable in your ability to command the machine successfully.



GWE has a "Hints" window that shows you what each code does in plain English...

With GWE, you can study the g-code from the comfort of your armchair and let it sink in. More importantly, GWE gives you lots of additional information that's hard to get at with most controllers, and it has a host of features to help you through the process of learning g-code. For example, it's "Hints" feature tells you what each code does in plain English.

We'll be including some exercises with each section that involve working with GWE. From time to time we'll also include videos that help illustrate how to go about using GWE for the exercises.

EXERCISES

1. If you don't already have GWE, take a moment now to sign up. We'll be using it for many of the exercises on each section of this course.

2. Watch the G-Wizard Editor Getting Started Tour video, it's a quick and easy intro:

https://www.youtube.com/watch?feature=player_embedded&v=IWVVrH1_588

3. If you'd like to learn more, we have provided an optional chapter about G-Code Editors.

THE COORDINATE SYSTEM

RIGHT AND LEFT HANDED COORDINATE SYSTEMS

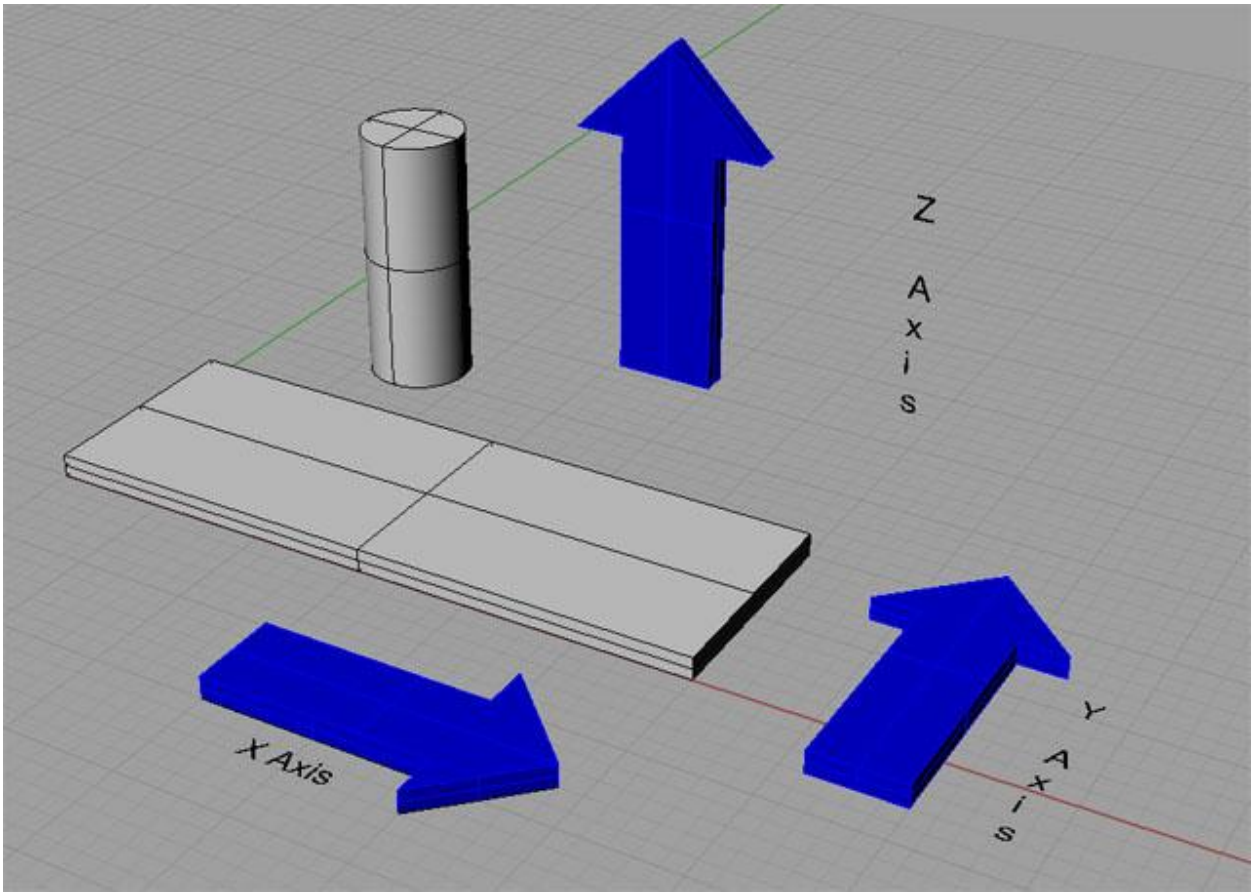
Hold up your right hand with the index finger extended and the thumb up, like you're simulating a gun. Now extend your second finger at right angles to the index finger. Those three fingers are now pointing in the directions of positive X (second finger), Y (index finger) and Z (thumb). In other words, the Z coordinate will get smaller as your spindle moves down towards the table of a mill. Now here is the tricky part:

Even if the table moves instead of the spindle, the handedness is based on the assumption it is the spindle moving!

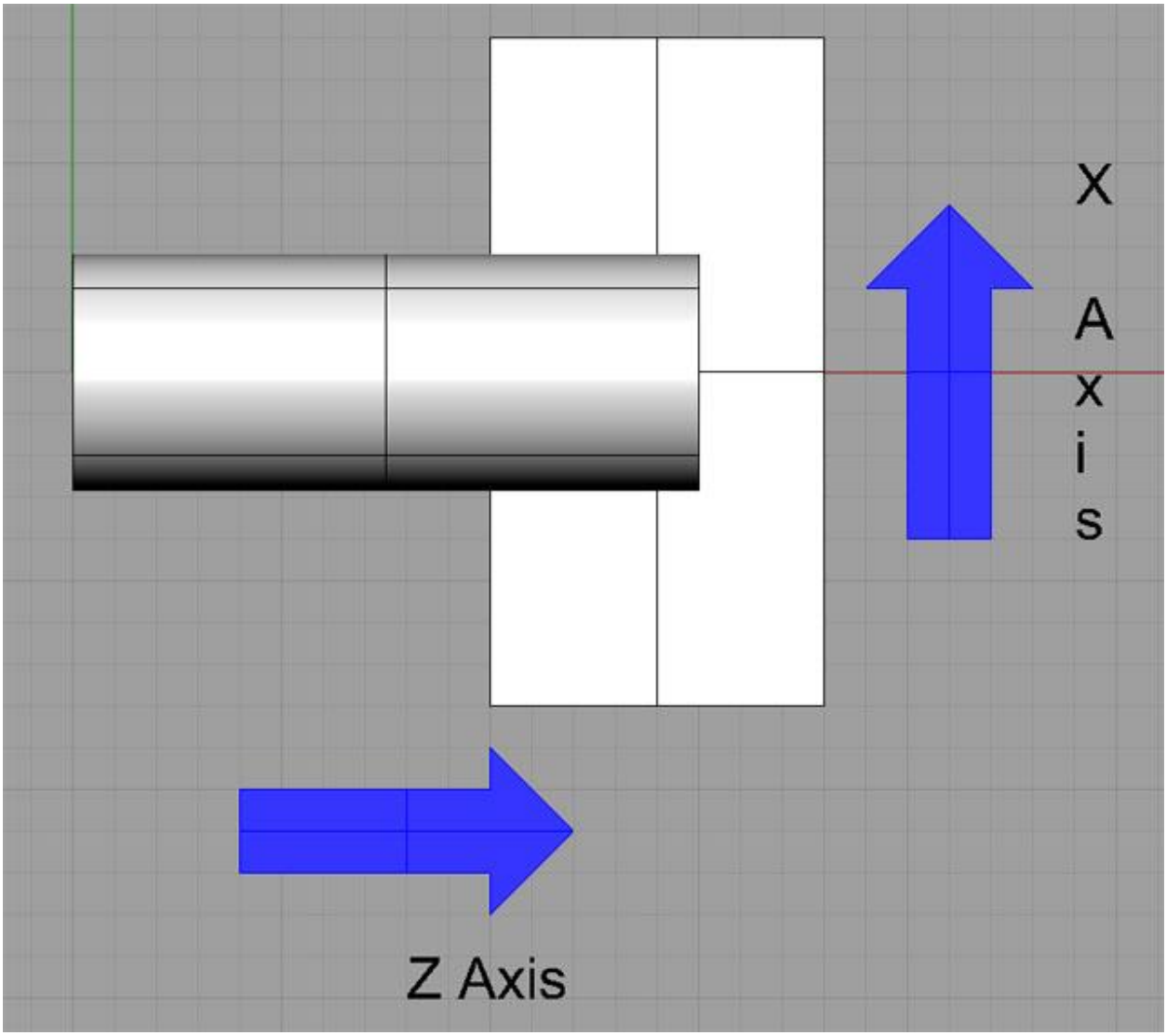
What that means is that for motions of the table, we reverse directions. Hence even though the diagram shows what looks like a left-handed system, if we consider how to get the same motion if the spindle moved instead of the table, we'd see it is really right handed.**That's why we say CNC uses a Right Handed Coordinate System, and the whole hand thing is just an easy way of remembering.** Even though most machines are right handed, it is not a requirement, and you need to check out how your machine is actually set up.

This whole handedness thing is just a way to remind yourself which way the coordinates go. It's not a big deal otherwise and you'll get used to what your machine uses pretty fast.

Each machine will have its own specific axis orientation which you should become familiar with. Here are some common types:



Mill Axes for a Typical Vertical Machining Center. Note: arrows show table motion in positive g-code direction. Handedness is spindle motion and reversed!

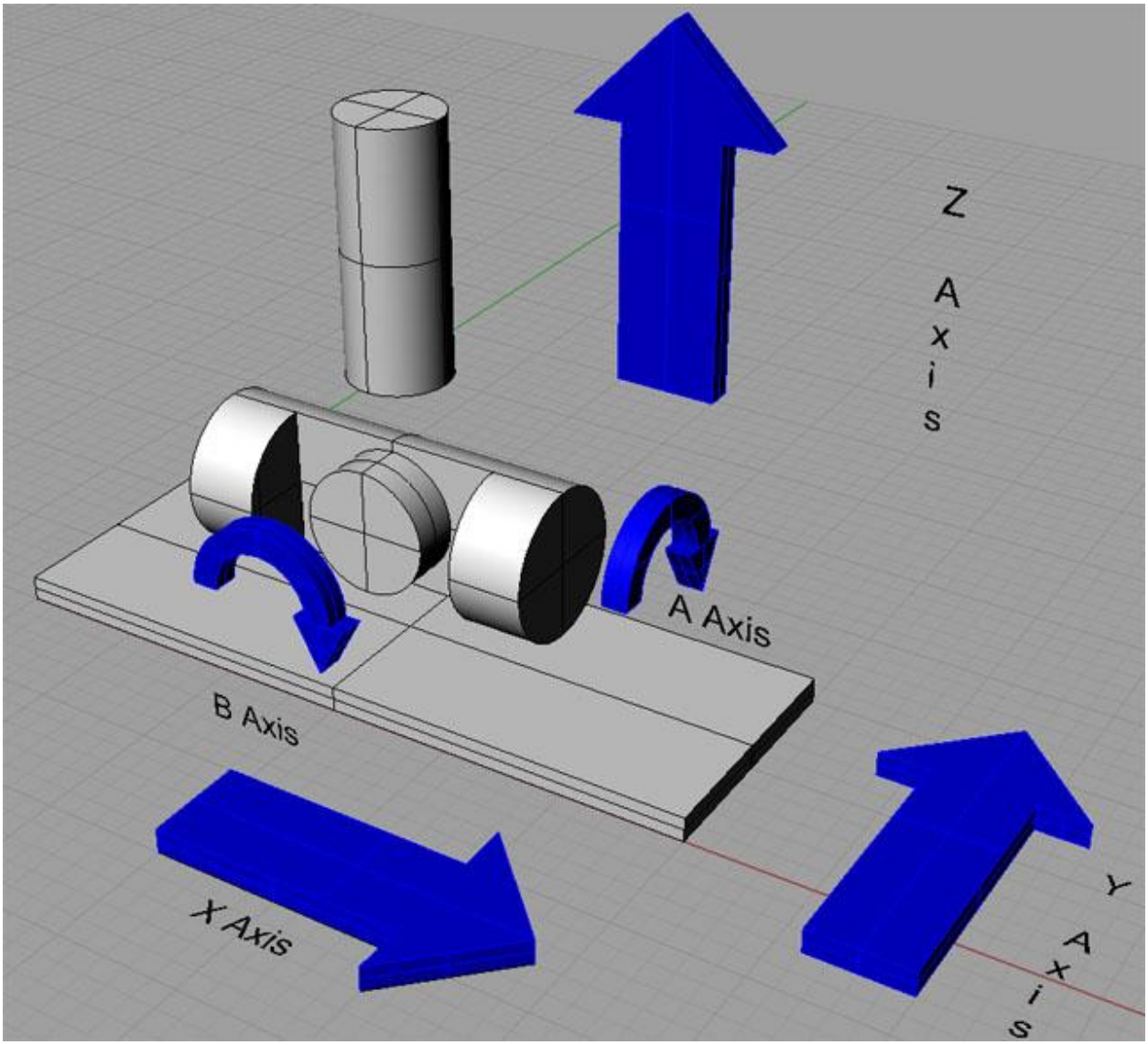


Lathe Axes for a Typical 2-Axis Lathe...

The cylinders in each drawing represent the spindle of the machine. Be sure to have a look at exactly how the axes are laid out on your machine. For example, horizontal mills are turned around considerably from the drawing I've shown. Lathes can get a lot more complicated than the simple 2-axis version I've shown.

4-AXIS, 5-AXIS, AND MORE

Much more complex configurations are possible when you have more axes. For example, here is a 5-axis setup:



5-Axis Mill With Trunion Table...

Note that we have added two *rotational axes* to the basic mill diagram to provide an A-Axis and a B-Axis. In general, A, B, and C are rotational axes that rotate around axes formed by the X, Y, and Z respectively.

EXPRESSING COORDINATES IN G-CODE

Now that we know what the coordinate systems are, how do we express coordinates in G-Code?

It's pretty simple: just take the axis letter and add the value. Spaces between the letter and its value are optional.

For example, a position that is 1 inch from 0 along X, 2 inches along Y, and 3 inches along Z is written as:

X1Y2Z3

You get used to reading them all run together like that quickly, but you can format them with spaces to make them more readable:

X1 Y2 Z3

or

X 1 Y 2 Z 3

Again, you get used to keeping the letters with the numbers, so I wouldn't add more spaces than just between the axes:

X1 Y2 Z3

That's actually the easiest to read once you get used to it.

WHAT ABOUT UNITS—METRIC OR IMPERIAL?

The example I just gave used inches, but in actuality the controller can be set to use either metric or Imperial. It's up to you to know which default the system comes up in and to change the units as needed. Try not to change units in the middle of a program, do so at the very beginning and then stay with the same units. It's too confusing otherwise. The G-Codes to change units only affect how the machine interprets the numbers. They don't change your program. We'll talk more about changing units in a future article, but for now, just be aware.

For rotational axes (which you'll only be using on a 4 or 5 axis machine), we don't use dimensions for the units, we use angles, typically in degrees. Rotating the 4th axis to the 90 degree position might be done as A90, for example.

RELATIVE VERSUS ABSOLUTE COORDINATES

Sometimes, it is very convenient to refer to *Relative* instead of *Absolute* coordinates. Let's assume the tool tip on my mill is at X0 Y0 Z0 and I want to move it to X1, Y2, Y3 (I dropped the commas, which are not used in G-Code, because I'm just trying to get you used to the switch from how you learned coordinates in school, e.g. (0, 0, 0), to how it's done in G-Code X0 Y0 Z0). I can make the move absolute or relative and it doesn't matter. "X1 Y2 Z3" does the trick since in either case we started from X0 Y0 Z0.

But, suppose your cutter is positioned at some point and you need to cut a 1" square with the corner aligned to that point. Perhaps you've used your edgefinder to locate the cutter precisely on some feature of the part. This is easily done with relative moves:

X1

Y1

X-1

Y-1

In essence, move 1" right, 1" up in Y, 1" left, and then 1" down in Y. Now we have a 1" square whose lower left corner is the initial point.

There are lots of cases where relative moves are handy so the ability to switch back and forth comes up a lot. We'll show you how to make that switch when we talk about how to move with G-Code, but for now, just be aware that there are both Relative and Absolute Coordinates.

Sometimes, we refer to relative coordinates with special axis letters. For example, IJK may be relative XYZ when defining arc centers. On some controllers, UVW may be used alongside XYZ to refer to relative coordinates without needing to change back and forth between relative and absolute modes. In other words, XYZ is used always as absolute and UVW is always relative.

For now, it is enough to be aware that relative coordinates exist. A little later, in the Intermediate Course, we have an entire chapter just on the subject of relative versus absolute coordinates.

OFFSETS

Another important Coordinate System concept I want to cover is that of Offsets. Offsets are another fancy way to think about relative motions. Let's suppose you want to machine 2 identical parts. Each is held in a vise on your table at the same time. How do you make one program that can do both parts without having to change the program for the position of each part?

The answer is that we use a Work Offset. More detail on those later, but for now, imagine that Work Offsets let us position the X0 Y0 Z0 origin in more than one place. We can put one on the first vise and another on the second vise. Now just by changing the work offset the same program can work to make the part on either vise.

There are lots of different kinds of offsets in CNC, and the skilled CNC operator/machinist finds that offsets are an extremely handy way to nudge the behavior of a G-Code Program without having to change that program. Most CNC controllers have an offsets screen where you do that. I mention this because any time you get a chance to learn about offsets, take the time to do so. They're digital power tools for the CNC machinist and are very handy. We'll cover them in greater detail later.

PLANES

It's convenient to refer to planes for various purposes. A plane is a flat 2 dimensional space defined by two axes. For example, the default plane on most mills is XY. If you draw an arc without specifying a change in the plane, it will be drawn on the XY plane. There is a plane for each combination of the linear axes XYZ:

- XY
- YZ
- XZ

The G17, G18, and G19 G-Codes select which plane is active. More on G17-G19 when we talk more about arcs.

CONCLUSION

You've now got the basics:

- You know how to visualize the coordinate systems relative to your machine using the left handed rule.

- You know how to express coordinates in G-Code.

- You know what units are used to measure the coordiantes.

- You know there is the possibility of both relative and absolute coordinates.

- You know that offsets let you shift the coordinate system around for various handy purposes.

- You know about planes.

We'll shortly introduce the notion of MDI, which is a simple way to use G-Code as though you are still a manual machinist. It's a good introduction to the basics of moving your CNC's axes. But first, we need to get you set up on G-Wizard Editor so you'll have a CNC Simulator to use for practice during these tutorial lessons.

EXERCISES

1. Form the left-handed and right-handed coordinate systems with your own hands and visualize which way the axes run on your machine. Which direction is the positive (increasing positive coordinates) for each axis? Which direction do increasingly positive coordinates in the g-code move the table and spindle?

2. Get out the manual for your machine and find the diagram that shows how its coordinate system works. Make sure to leave the manual handy, whether it is paper or online. We'll refer back to it a number of times as we go through the various exercises.

3. Bring up G-Wizard G-Code Editor. By default, you're in Mill Mode. There are views for Perspective, Top, Front, and Right. Download the sample engraving file from our download page. You want the file called HomeSwitchRearPanelEngrave. Start up GWE and do a File Open to load the downloaded file. Have a look at it in each view.

- Top is a view from the XY plane

- Front is a view from the XZ plane

- Right is a view from the YZ plane

G-CODE DIALECTS, POST PROCESSORS, AND SETTING UP G-WIZARD EDITOR

THERE ARE MANY DIALECTS OF G-CODE

Now that you have some basics under your belt, it's time to talk about some of the complications that arise from different dialects of G-Code.

Some wag once joked that the great thing about standards is there are so many to choose from. So it is from G-Code. While much of it remains the same from controller to controller (setting aside alternatives to G-Code from things like Mazatrol, Heidenhain's Conversational CNC language, and others), there are important details and defaults you need to be aware of to understand the particular dialect of g-code your controller needs to be happy.

In terms of sheer numbers of users, the Fanuc dialects of G-Code are probably the most common among professionals and Mach3 among hobbyists. This is not to say they are better than other G-Code dialects, just that they are more common and so if you're going to talk to other machinists or move around from job to job and machine to machine, it may be helpful if you're familiar with those dialects and how they differ if your machine doesn't use one of these two controllers.

G-Code has an extremely long history. The first attempts at standardizing it came out of the Electronics Industry Association's RS-274 standard which has evolved to NIST's RS-274NGC standard. The original EIA standards work was begun in the 1960's but the first standard wasn't released until 1980. Even though there are now standards (ISO has one too that is nearly the same as RS-274), it isn't clear how many controllers out there are purely standards based. Indeed, many controls will claim to be some standard or other, but when you look closely at the details they're pretty non-standard.

HOW ARE THE DIALECTS DIFFERENT?

G-Code dialects differ in a variety of ways. Most manufacturers have added their own little bells and whistles to make their dialect better for competitive and marketing reasons. For example, Haas has a series of special g-codes for pocket milling, as well as some special parameters and capabilities on some standard G-Codes. It pays to understand the special capabilities of your machine because they were probably put there to save time based on feedback the manufacturer got from its customers.

In general, we see the following categories of differences between G-Code dialects:

- Which G-Codes are Supported. Not all controllers support all G-Codes. For example, many early lathe controls do not support the G71 and similar roughing cycles.

- G-Code mappings. Sometimes the same function will be supported by different g-code numbers on different controls.

- Parameters and Macro Programming. Parametric programming with macros is something that emerged after the basic standards were in place. Fanuc Macro B is probably the most common standard for it. Many controls are very limited in their capabilities around Macro Programming and there are a lot of detail differences around exactly how Macros work.

- Parameters. Many G-Codes need additional information to do their job, so they use other words (letters) to collect that information. Exactly which words collect which information can vary from one control to the next.

- Formatting. Some controls allow G0 or G00. Some insist on G00. Some allow numbers with no decimal, others insist on a decimal or even a trailing zero. "1", "1.", and "1.0" are all variations that may be accepted, rejected, or required when specifying the number 1.

We'll talk more shortly about what all of this means, but for now, be aware that these differences exist. For simple programs and MDI use, obviously a lot of this won't matter. But, for writing complex hand-written G-Code or trying to understand why the G-Code your CAM program emits isn't quite right, you'll need to be aware of the dialect issues.

POST PROCESSORS

If you're using a CAM package to generate G-Code, it has a Post Processor. The job of the Post Processor (or "Post" as most machinists call it), is to convert generic geometry moves in a dialect independent (and often G-Code independent) language to the particular dialect of G-Code your machine uses. The Post also gives you a lot of flexibility to tailor your CAM output to the practices your shop likes to use, so it is a handy way to customize the G-Code output of your CAM.

If you frequently are making the same hand edits over and over again to your G-Code, it's worth investigating whether a change to your Post would eliminate that hassle. You can even modify the Post to save your operators some time. Maybe you want the table on your mill to move as close to the enclosure door as possible at the end of a job to facilitate loading and unloading. That's a logical thing to have the Post do for you. Depending on the CAM package and its Post capabilities, sometimes very powerful flexibility is an option. For example, the Post may be able to popup a custom menu allowing you to choose some options you've specifically designed for your shop's needs on every Post.

Posts are typically languages unto themselves that are often unique to the CAM package. In some cases, the CAM vendor wants to sell services customizing Posts so may not make it easier for you to obtain the knowledge and tools to modify your Post. In others, they're perfectly happy to help. Be aware that CAM without the right Post for your situation is painful to work with, so factor getting a good Post into any CAM acquisition plans.

There are also generic tools out there that specialize in making it easy to do your own custom Posts. Posthaste is one example of such a tool.

Setting Up G-Wizard Editor for Your G-Code Dialect

Before we go too much further, let's talk about how to set up GWE for your machine's dialect. That will get you started down the road of using the right dialect as you use GWE as a cnc simulator (also called a g-code simulator, or a cnc or g-code verifier) for your machine. Presumably by now you have registered for GWE and downloaded and installed it. Refer to the GWE Setup page for details, and follow along with this video to get GWE set up for your g-code dialect:

https://www.youtube.com/watch?feature=player_embedded&v=7dGWkmRAr2M

Very Important:

For purposes of this tutorial, unless we specifically say something different, we're going to assume you're running a Fanuc controller. When you follow along with our exercises, you should run a Machine Profile for the Fanuc Controller, preferably by downloading our canned profile. When you go to program for your machine, you'll need a profile properly set up for your machine!

Hopefully that makes sense to you.

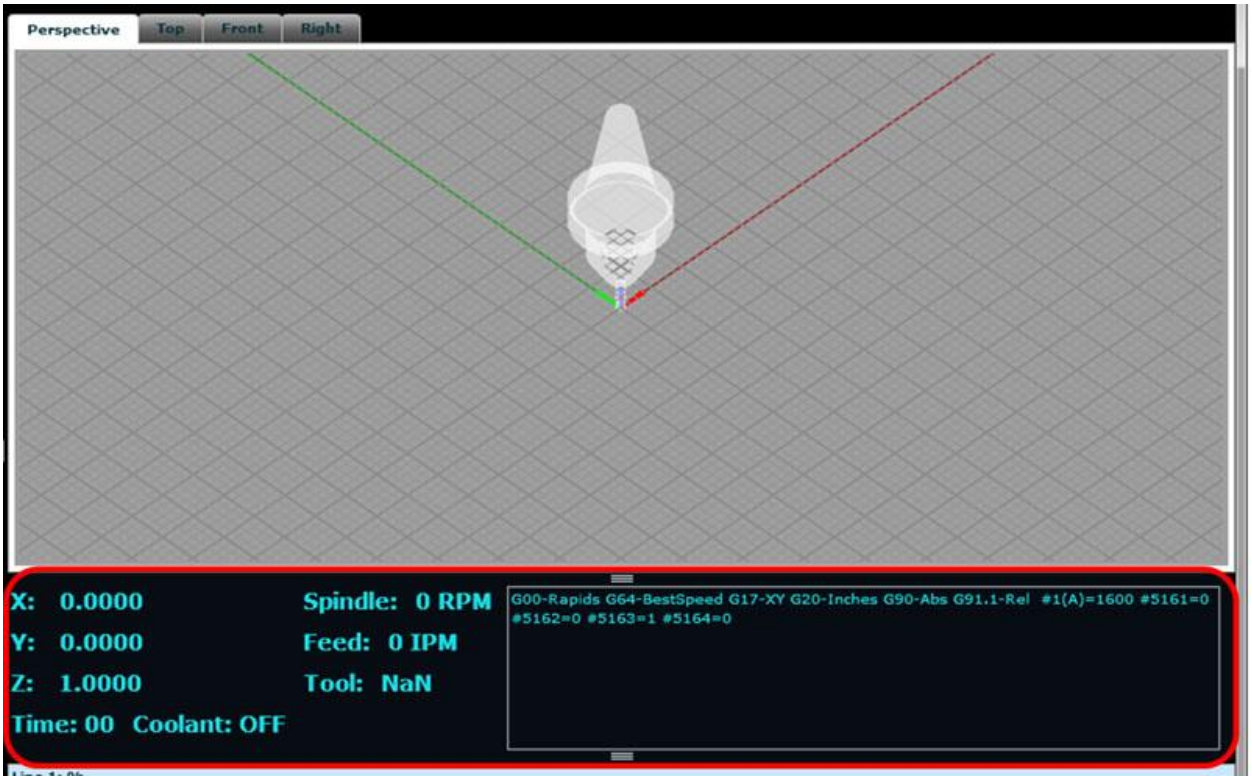
It's inconvenient that all CNC machines don't use the same G-Code dialect, but it is a fact of life you may as well get used to sooner rather than later. Knowing that there are differences and having some idea what they are will make it easier for you to exchange information and g-code with other machinists. You'll want to make sure all of your tools are on the same page as your CNC controller in terms of having the same Post settings, otherwise you'll have all sorts of problems getting your G-Code working right.

As we go through each G-Code area, we'll try to go over some of the Dialect differences so you can make sure all your software is doing the right thing!

MODAL BEHAVIOR FOR CNC CONTROLLERS

Note the use of the word "Modal" with respect to IJK and R centers. There are lots of "Modes" in G-Codes. A "Mode" simply means the CNC Controller is remembering some behavior from the last time you told it what to do. People often think modally, meaning if you don't specify every detail, they fall back to assuming you mean to do things the way you did the last time you specified.

When you run a G-Code program in the GWE simulator, it shows you which modes are in effect in the little box under the backplot:



The controller's state and G-Code Modes are shown Below the GWE Backplot...

We'll talk more about how to run the G-Code Simulator in GWE soon, but for now, just be aware of where the currently active modes are shown below the backplot and in the little box to the right. The red outline shows the area that describes the current state of the CNC controller, which includes any modes. The screen shot shows the modes that are active at the very beginning of the Engrave program. You can see the following modes are in effect:

- G00: This mode means all motion is in straight lines at rapids speeds--the fastest your machine can move.

- G64: Best speed mode means the controller will make best efforts to move at the desired speed. An alternate mode is called "exact stop". More on that later.

- G20: Motions are in Inches rather than MM.

- G90: Motions are absolute rather than relative.

A lot of G-Code programming involves making sure the right modes are in effect before telling the machine what to do next.

EXERCISES

1. Set up and save a machine profile using the Fanuc Mill profile so you can follow along with the examples and exercises for this tutorial.

2. Set up another machine profile for your machine's controller, assuming it is not a Fanuc. Go through each option on GWE's Post page and set the options up as your controller calls for. This will give you a GWE machine profile to use when programming your controller.

3. Go through each post option in the GWE menus and get some idea what they mean and how they can differ from one controller to the next. It's good to have a notion in the back of your mind that these differences exist. You can use the GWE post options to look up what some of the possibilities are.

4. If you have a CAM package, select the proper Post for your controller and create some G-Code with the CAM package. Load the result into GWE and see that the toolpath backplot looks like what you'd expect in GWE. If it doesn't, there are either errors in the CAM package's Post, or you don't yet have GWE's Post set up quite right.

5. Get familiar with how your controller shows its current state, including the modes, axis coordinates, spindle speed, feedrate, and currently selected tool.

MDI: CNC FOR MANUAL MACHINISTS

CNC CAN BE QUICK AND DIRTY TOO!

Many manual machinists are under the impression that CNC is only good for manufacturing multiple identical parts, and that one offs are a lot faster to do on manual machines if the part is simple. They have visions of spending hours making CAD drawings of even the simplest parts and then feeding those CAD drawings through a CAM program to finally produce G-Code, at which point they are finally ready to make some chips.

It just ain't so!

Good CNC machinists can do most anything a manual machinist can do and then some. This article is all about how.

You can do anything on your CNC that the manual machinist can and often faster once you know how...

DRO'S AND POWER FEEDS

The first trick is to forget about G-Code, CAD, and CAM. Think of that fancy CNC machine as nothing more than a manual machine that has DRO's (digital read outs) and Power Feeds on all of the axes. Most manual machinists would see that as a tremendous improvement over

handwheels and no DRO's, and wouldn't hesitate to use such a machine. Once you get used to the idea of using your CNC in that capacity, you'll be surprised at how easy it is.

A typical Fanuc panel you might use...

But how does it work? The CNC certainly doesn't look like a manual machine with DRO's and Power Feeds. There's way more knobs and buttons and no handwheels!

Let's deal with the DRO's first. By now you will have seen that the front panel of your CNC displays the X, Y, and Z coordinates at all times. See all those X, Y, Z, and A displays on the Fanuc panel above? Those are DRO's.

What about those Power Feeds?

There are two approaches. First of all, CNC's have the ability to jog. Jogging is where you're spinning a simulated electronic handwheel, or holding down a button or joystick to make the axis move. You generally select an axis, select a speed or multiplier (x1, x5, x100 or whatever), and then operate the control to move the axis. It's pretty easy. Here is a typical pendant with an electronic handwheel (called an "MPG" or "Manual Pulse Generator") for jogging:

A pendant for jogging a CNC. Select axis and speed then spin the wheel...

But, there is a better approach if you're actually trying to do some machining as opposed to touching off a setup, and it's called "MDI".

MDI quite simply stands for "Manual Data Input", and that is what it is. You put your machine into MDI mode, and it lets you directly type in G-Codes which are executed immediately, instead of having to create and run a program. MDI is extremely useful to know your way around. Check out this Mach3 Standard Mill home screen:

Mach3 Standard Mill Screen Set...

You can see the XYZ DRO's on the lower right. Just to the left of them is the MDI type-in field. When not running a program, you can type G-Code commands into the MDI field for immediate execution. Look up in your operator's manual how to go about accessing MDI for your controller.

USING MDI TO MOVE THE AXES

To use MDI to move the Axes of your machine, you need to know a couple of G-Codes.

G00 (some machines will accept G0 too) tells the machine to expect rapids motion. It is a mode, as we discussed earlier. Most CNC's come up with G00 active by default, meaning if you enter coordinates, they'll cause rapid moves. "Rapids" are when your machine moves absolutely as fast as it can to the place you've commanded it to move.

I want you to ignore G00 and rapids when you MDI for the time being if you're just starting out. You don't really need the machine to move at its fastest possible speed as a beginner--it just makes it easier for the machine to get away from you.

Instead, I'd like you to try to remember to perform a G01 command at the beginning of any session. That tells the CNC to switch from rapids to linear interpolation. It amounts to the same thing but you have a fine ability to control the speed the machine moves at using the "Feedrate" command, which is just an F. So for example, you might start up the machine and first thing you do is type in something like "G01 F20" in the MDI. What

that will do is limit your machine's speed to 20 units/minute. If you're in inches, its 20 IPM. For metric it'll be mm/minute.

Now the axis motions will be a lot less scary and easier to manage. If you know the feedrate you want to do your cutting at, use the "F" word to establish that feedrate. If you don't, try something like G-Wizard Calculator to determine a good feedrate.

Practice moving the axes around with MDI. If you're not already in G01 mode, enter "G01" to the MDI. Enter your feedrate using the "F" word--F20 for 20 IPM, for example. To make a move, simply enter the appropriate axis coordinates on the line. The machine will move using what's called "linear interpolation". This means multiple axes move at rates so that they all get to the end coordinates you've specified at the same time. If you don't need an axis to move from its current coordinate, just don't specify it.

For example, suppose you want to make a cut in the X axis. You've issued the "G01" and "F" words (G-Code calls those letters "Words") and you're ready for the coordinates. Let's say your cutter is all lined up for a face milling pass, and is sitting at the correct Y and Z coordinates. You've got a workpiece 5" long in X, the X coordinate on the console says 7.125 and the cutter is spinning about 1/2" off the edge. What do you do?

Well, you want to move the 5" and leave 1/2" on either side. Let's allow even a little more than that to be sure the cutter clears the workpiece--after all, we're just eyeballing quick and dirty. So, type "X1.0" and the machine will feed the spindle right to left moving across the workpiece in the X direction.

Pretty easy, huh?

FIRING UP SPINDLE AND COOLANT WITH M-CODES

Consider that a hypothetical case. Had we been doing it for real, we would've needed to fire up the spindle and the coolant. So let's take time to learn a couple more codes.

While the whole language is commonly referred to as "G-Code", and we have certainly used G-Codes G00 and G01 thus far, we're going to need some "M-Codes" to control the spindle and coolant. Think of M-Codes as "Miscellaneous" codes. Fanuc refers to M-Codes as "Auxilliary" functions.

To start your spindle going, let's first select a speed with the "S" word (S for Speed, it's easy to remember most of these codes). Let's go for 4000 rpm by entering S4000 in the MDI. The spindle won't actually start to rotate until we ask it to because it doesn't know whether to go clockwise or counterclockwise. Enter "M03" for clockwise rotation and your spindle should spin up to 4000 rpm in the clockwise direction. Coolant is switched on using M08 for flood and M07 for mist or just air, depending on how the machine is set up.

You can stop the spindle by typing "M05", and you can shut off all coolant with "M09".

M13 and M14: Start Spindle and Coolant

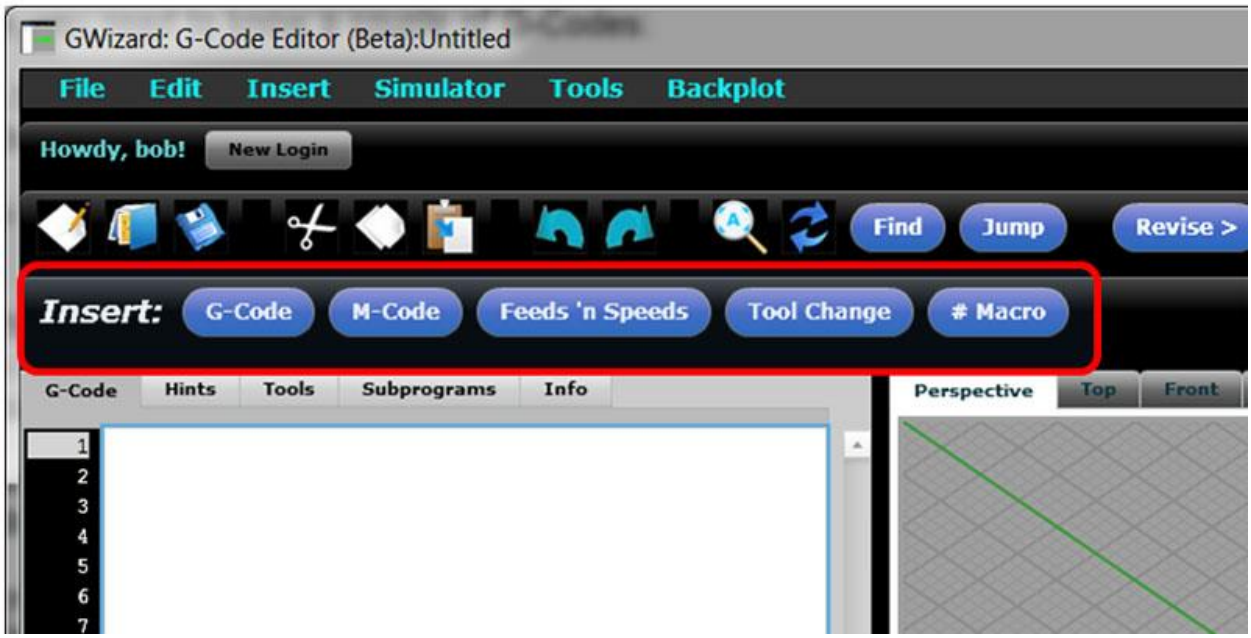
On some machines, typically lathes, you may find M13 and M14. These m-codes start the spindle and the coolant at the same time. M13 starts the spindle running clockwise and M14 runs counterclockwise. Stop the spindle with M05 and coolant with M09.

Now you've got the gist of using MDI. There are many more g- and m-codes you can use with MDI, but those are all the ones you need to do everything a manual mill with power feeds and DRO's can do.

USING G-WIZARD EDITOR'S WIZARDS TO HELP KEEP TRACK OF THE CODES

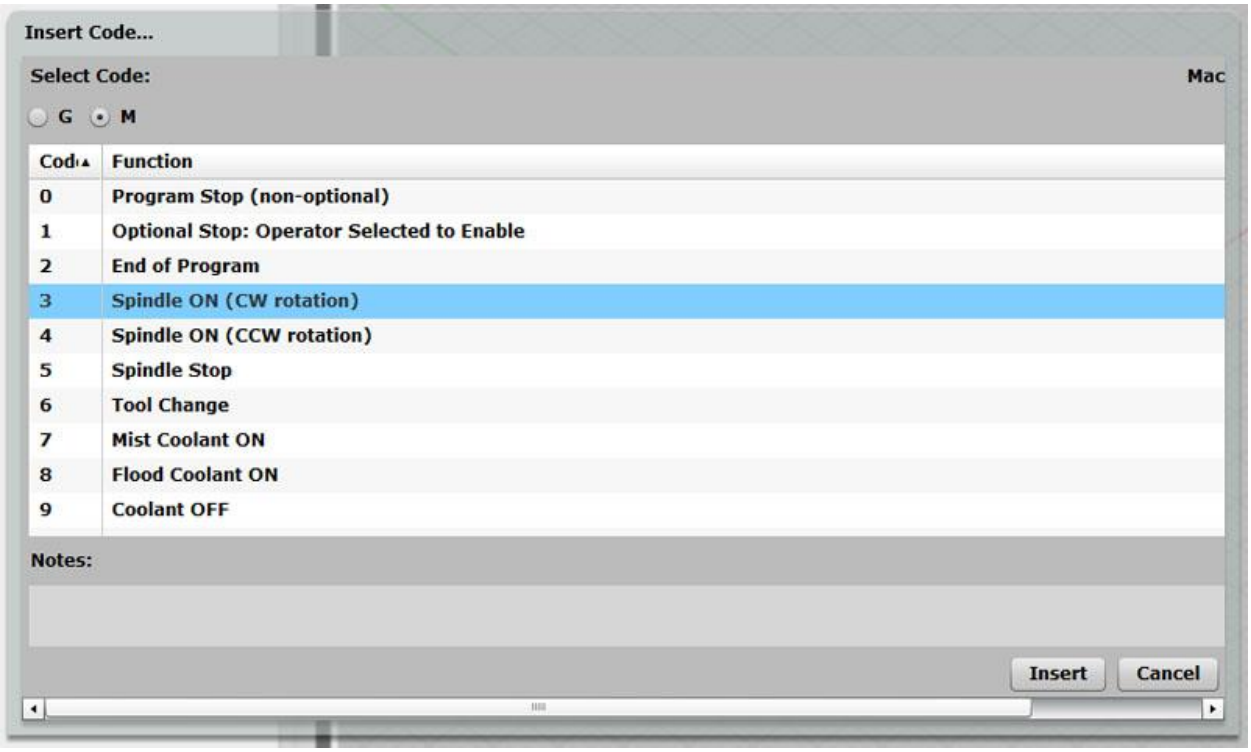
You're probably starting to wonder how you're going to memorize all these codes. After all G00 and G01 don't exactly mean anything by themselves--they're pretty arbitrary. Don't worry, there aren't all that many g-codes you'll be using on a regular basis, certainly not on MDI. You'll pick them up quickly with practice, and for the rest, there are quick reference tables and other tools.

One handy tool is G-Wizard Editor's G-Code Wizards. They remember all this for you so you don't have to remember the exact codes when you're writing a program. For example, the Toolbar groups different G-Code functions you can select for entry:



The buttons to the right of "Insert" trigger the various code insertion Wizards...

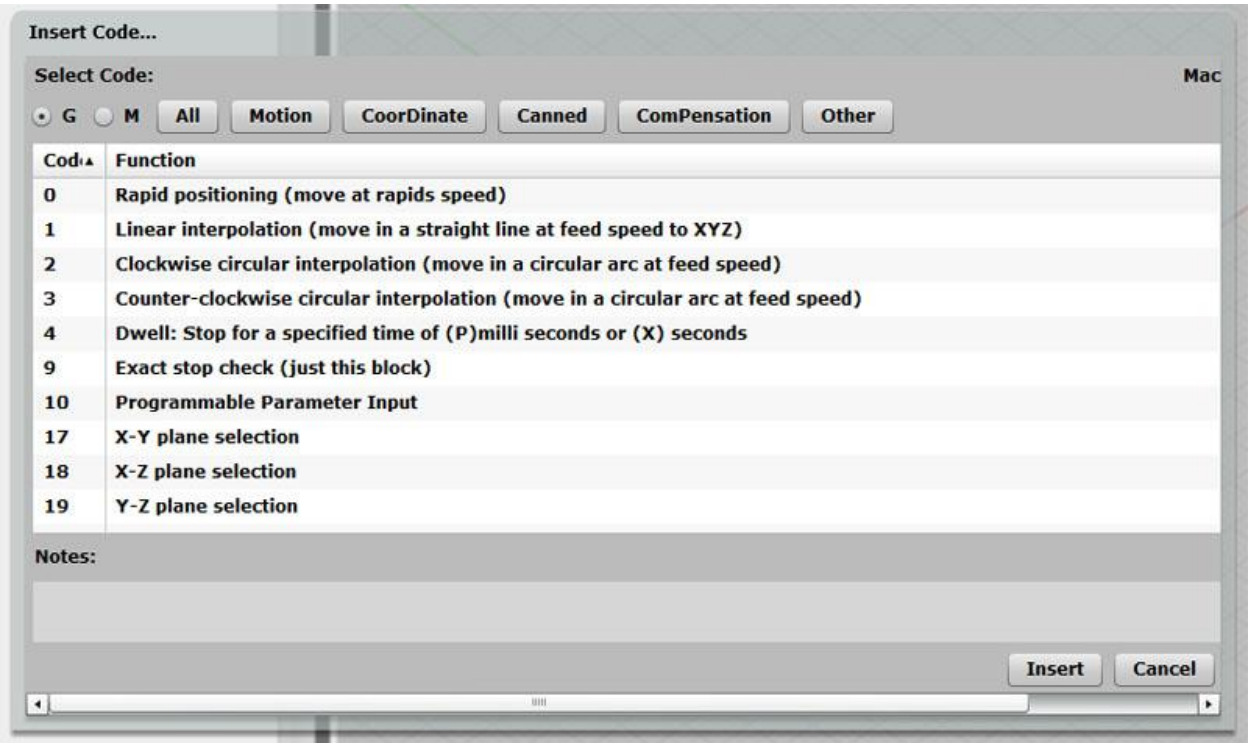
If you select "M-Code" (or type "Ctrl+M" for M-Code on the keyboard), you'll get the Wizard for entering M-Codes:



Each code is called out with a description of what it does...

Each code is called out in the list with a description of what it does. Just select the code you want (or type its number, the Wizards are designed so you need not use a mouse) and click "Insert" (or press Enter) and it goes into your program.

Don't see the code you want on the list? You can close and pop up a different Wizard or switch directly to G-Codes using the radio button at the top of the panel. The G-Code Wizard is a little fancier than the M-Code Wizard:



G-Codes are grouped in categories...

The G-Codes are grouped into categories by the buttons at the top to help you narrow your search. So far we have only considered G-Codes from the Motion category, G00 and G01, but there are many more and we'll be going through all of them.

Another thing to note: the Wizards only show you codes that are enabled by the Post for your control so you can't inadvertantly enter a code that won't be recognized and will cause an alarm (CNC jargon for an error). There are a lot more handy capabilities in the Wizards that we'll be covering during the rest of this course.

QUICK REFERENCE

Here is a little quick references of the commands we've discussed that are useful for MDI. You can print it out to keep near the machine until you memorize them. These codes and the ability to use MDI on your machine and read the console DRO display is all that's needed to run your CNC like a manual mill with DRO's and Power Feeds.

MDI G-Code Quick Reference	
G00	Rapid motion mode. Remember: We're going to use G01 instead, so type G01 in to cancel G00 as soon as you bring up MDI!
G01 F<ipm>	Linear Interpolation mode. Motion in a straight line at the feedrate established by "F". You can change feedrate without issuing a G01 each time.
XYZ	Move to the specified coordinates in a straight line.
ABC	Move the rotational axis to the specified angle.
S<rpm> M03	Set spindle to rpm and start it turning clockwise. You can change spindle rpm without issuing an M03 each time.
M05	Stop spindle
M08	Flood coolant on
M07	Mist or Air coolant on
M09	All coolant off

FINAL WORD: WATCH OUT BELOW!

MDI is almost too easy, in fact. I broke more tools when I was learning CNC by fat fingering MDI than running programs. If you get confused, it's very easy to start the cutter moving at high speed in totally the wrong direction. If you're lucky, it'll strike the edge of the workpiece moving too quickly and just snap off the cutter clean. If you're not so lucky, you can run it into the vise or worse, your table.

Here are some tips to try to cut down on the incidence of accidents with MDI:

- Always think carefully about what's about to happen before hitting the Enter key to initiate the MDI command.

- The bigger the tool, the worse the mishap if it crashes. Start out with small cheap endmills and leave the big expensive facemills and indexable tooling until you're sure you have the hang of it.

- Beware dropped digits and signs. This is an easy way to go somewhere you didn't expect.

- Use your FRO (Feedrate Override) to slow everything down until you're sure. In fact, crank it way down before you start the command and inch it up as things look to be going okay.

- Remember, avoid G00 and stick to G01 with a slow feedrate. First because you'll move more slowly and second because you won't get confused about whether the machine is in G00 or G01 when you're first starting out.

- Use your pendant and jog. It's easier to let up if things start to get pear shaped on you. Save MDI for feeding cuts.

- Entering the workpiece or moving close to the workpiece requires the most vigilance. Once you're in the neighborhood, small moves at feedrates are less likely to be a problem.

- Before you issue an MDI, make sure you know where the big red E-Stop button is. Visualize pressing it a couple of times. Don't hesitate to reach for it if you don't understand what's going on.

- A lot of CNC controllers have a distance to go DRO readout. Use it to make a quick check as the cutter starts getting close to cutting--if distance to go is wildly greater than the amount you expect the cutter to have to move to finish your command, you have a problem.

- Don't allow distractions while you're running the machine. Give it your full attention, and don't walk away. When a pilot is landing a plane, he uses his full concentration--all conversation in the cockpit not specific to landing the plane is stopped.

EXERCISES

1. Take out your manual and figure out how to jog your CNC machine and how to issue MDI commands.

2. Use GWE as a G-Code simulator. Start with an empty file and issue your faux-MDI commands by just typing them into the text pane. Watch what happens. Get good at predicting what will happen. Learn to make the cutter go where you want it to go.

3. Once you've mastered getting G01 to go where you want it to in GWE, get comfortable practicing MDI commands on the machine without turning on the spindle. Start out with the spindle high above anything it might touch, and don't issue any "Z" axis moves until you've got the hang of X and Y.

4. Load a piece of scrap material and a cutter and try some practice passes under MDI control.

5. Learn to use touch offs and an edge finder to precisely locate the tool just as you would with your manual machine. If you don't know how to do this, stay tuned as we're covering it next.

6. A great next step after learning MDI is CNC Conversational Programming. Check out G-Wizard Conversational to give it a try.

CONVERSATIONAL CNC

INTRODUCTION

Before there was CNC, we had Manual Machining (actually, we had hydraulic tracers and other limited automation, but bear with me) and a lot of very good parts got made fairly quickly. Even today the subject of whether CNC'ers need to learn Manual Machining will often provoke a strong debate back and forth. Many believe that CNC is only good for making many parts or complex parts. They maintain that manual machining will be faster any time you're just making one relatively simple part.

If you drill down on the argument and ask why manual machining would be faster, the two most common answers I get are:

#1 Because our CNC isn't tooled like a manual machine, it's tooled to make lots of identical parts quickly.

This, of course, is a matter of choice and I don't see it as a particularly good argument in favor of the manual machines. There are plenty of excellent toolroom CNC machines such as the Haas TL series lathes. You don't have to tool your CNC in that way.

#2 Because you have to stop and make a CAD drawing and then do CAM work to get the g-code for the CNC. With Manual Machining, you can often work directly from the print.

This one is much closer to the mark. It can be a lot of trouble to produce a CAD drawing and run a CAM program for some simple part, particularly on lathes where simple spacers and bushings are so easy to run off quickly on a good manual lathe.

We've seen in our last chapter that with MDI, you can treat your CNC like a Manual Machine. For many, that's the end of the argument--with the same tooling a CNC is just as fast or maybe even faster than the Manual Machine. But, we can go even a step further and be substantially ahead of the Manual Machine if we have Conversational CNC.

WHAT IS CONVERSATIONAL CNC?

Conversational CNC is a quick way to do simple jobs without resorting to creating a CAD drawing or running a CAM program. For all those times when you would turn to a manual machine instead of a CNC because the overhead of the CADCAM cycle was going to mean you could do it faster manually, this is a better way. Now you can stick with the CNC and do the job even faster than the manual, even for one-off parts. The key is simplicity. Lathes are particularly well suited to Conversational CNC, because many round parts are fairly simple. But there are many applications for Conversational CNC on Milling Machines too.

Think of Conversational CNC whenever you have a relatively simple part that doesn't have a lot of features or features that involve complex curves and shapes. Brackets, bushings, spacers, and the like are common examples. Another great application for Conversational CNC is secondary operations. Rather than go all the way back through

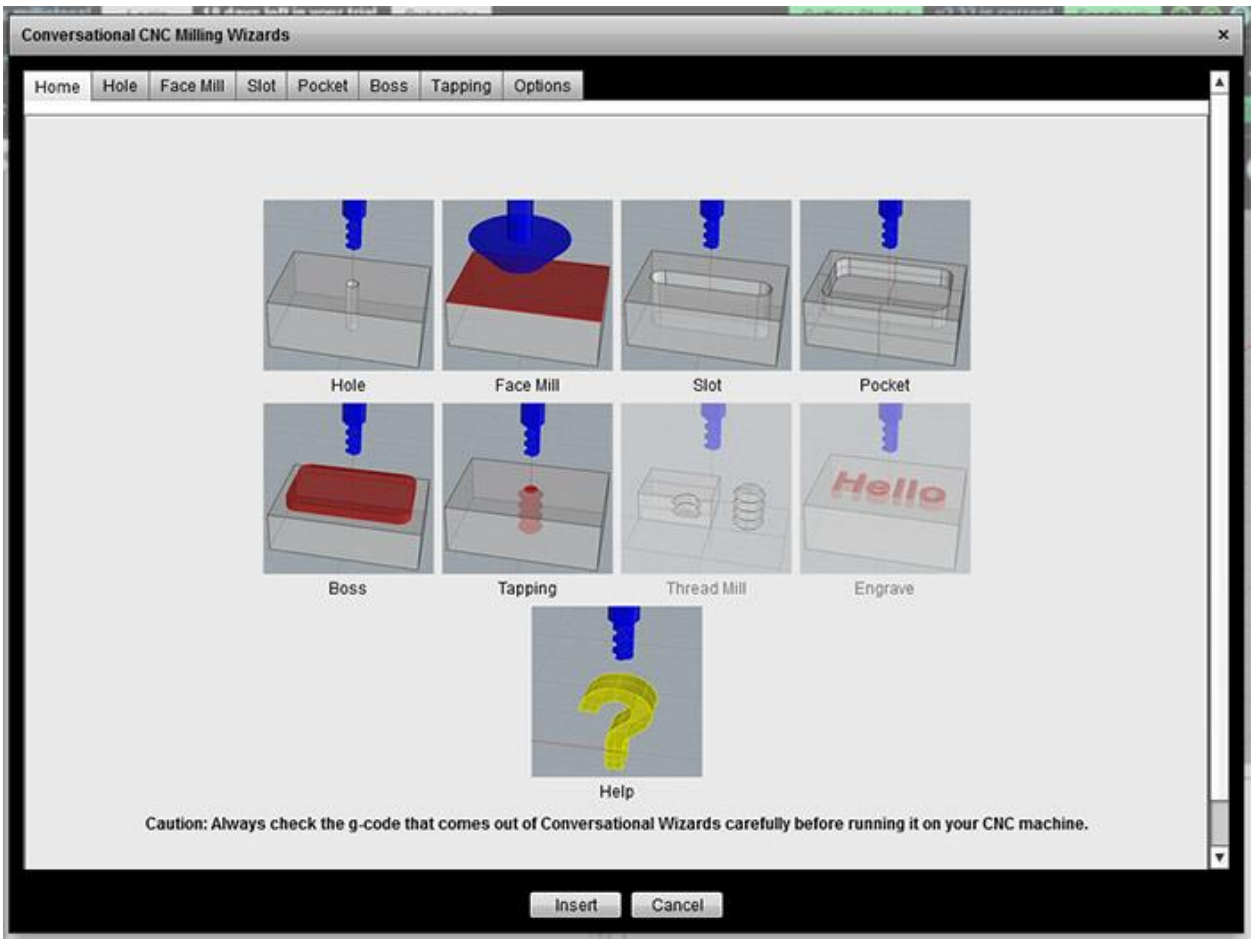
the CADCAM cycle to pick up a hole that was missed or to move some holes, try using Conversational CNC. It's quick and easy.

If you want to see some examples of how to program certain features in g-code, Conversational CNC is a quick way to turn them out.

Let's run through some Conversational CNC examples for both Turning or Milling.

CONVERSATIONAL CNC FOR MILLING

First, here is the graphical menu showing the various Conversational CNC operations available in G-Wizard Editor:



Conversational CNC Operations in G-Wizard Editor...

What you have to do is break your part down into these basic operations:

- Hole: Drill or Interpolate a hole.

- Face Mill: Surface the top of the workpiece

- Slot: Cut a slot at any angle.

- Pocket: Cut a rectangular or round pocket.

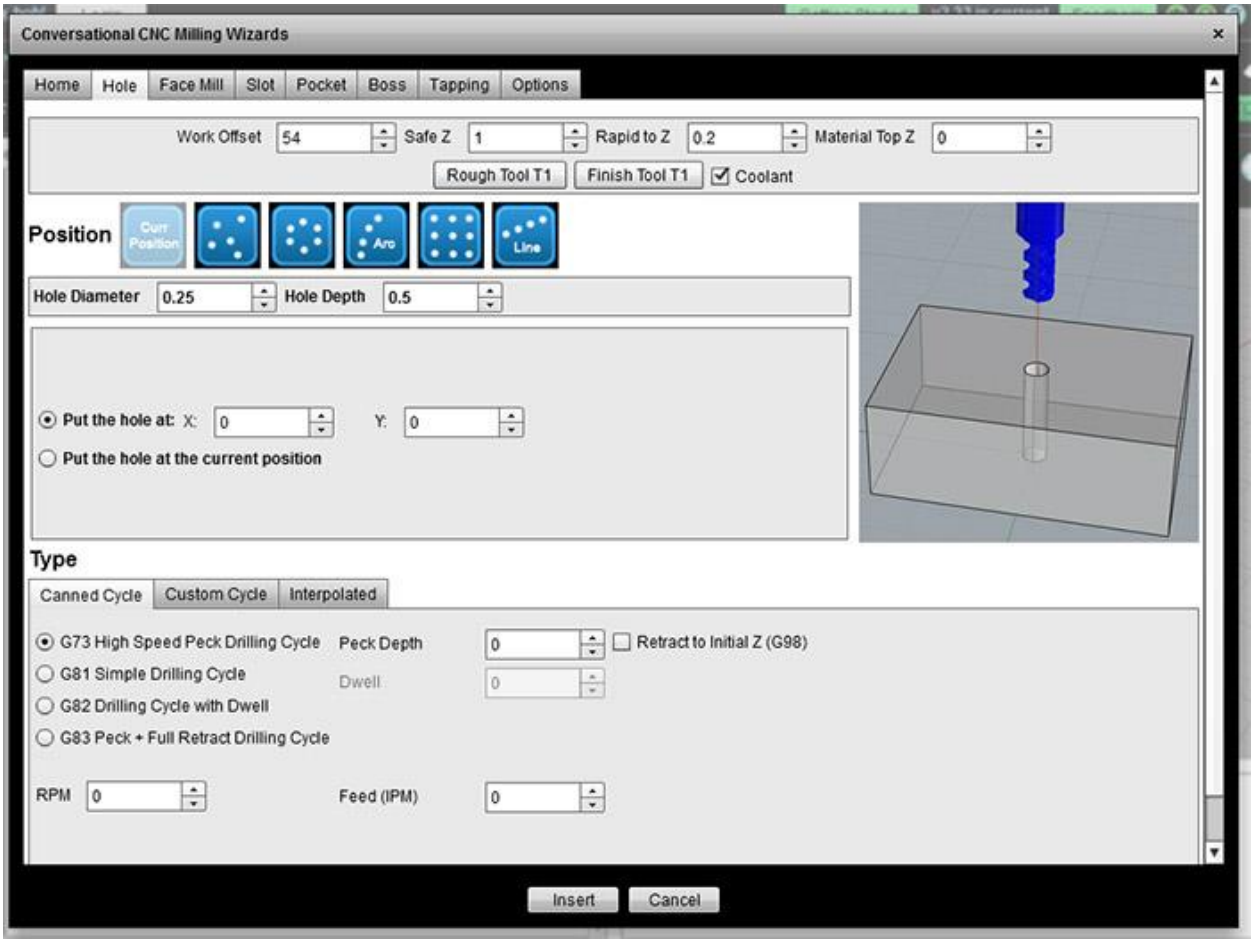
- Boss: Cut a rectangular or round boss.

- Tap: Tap a hole

- Thread Mill: Thread Mill internal or external threads

- Engrave: Do simple engraving

As you can see, quite a lot is possible with these basic options. If you click one, you'll get a popup Wizard like this one for the Hole Operation:



The Conversational CNC Hole Wizard

Now it's just a matter of filling in the blanks. You've got some basics at the top:

- Work Offset: Let's you use work offsets, you won't need them at first, so just leave it set to 54.

- Safe Z: This is the Z where the cutter can move freely without hitting anything. If 0 is the top of the workpiece, I usually make Safe Z 1" above.

- Rapid to Z: This is the height where we can rapid down to there, but XY motions must be at feed speeds. I make this 0.2" above the Material Top.

- Material Top Z: Cutting begins here and most CNC'ers like to make that value 0.

- Roughing/Finishing Tools and Coolant. You need to tell the Wizard which tools to use and whether to turn on Coolant.

Next is a set of options for positioning multiple holes:

- Put the hole at the Current Position

- Give a list of hole coordinates

- Make a circle of holes

- Place the holes on an arc

- Place the holes in a grid

- Place the holes in a line

Lastly, we get into the details of how the hole will be programmed:

- Via canned cycle

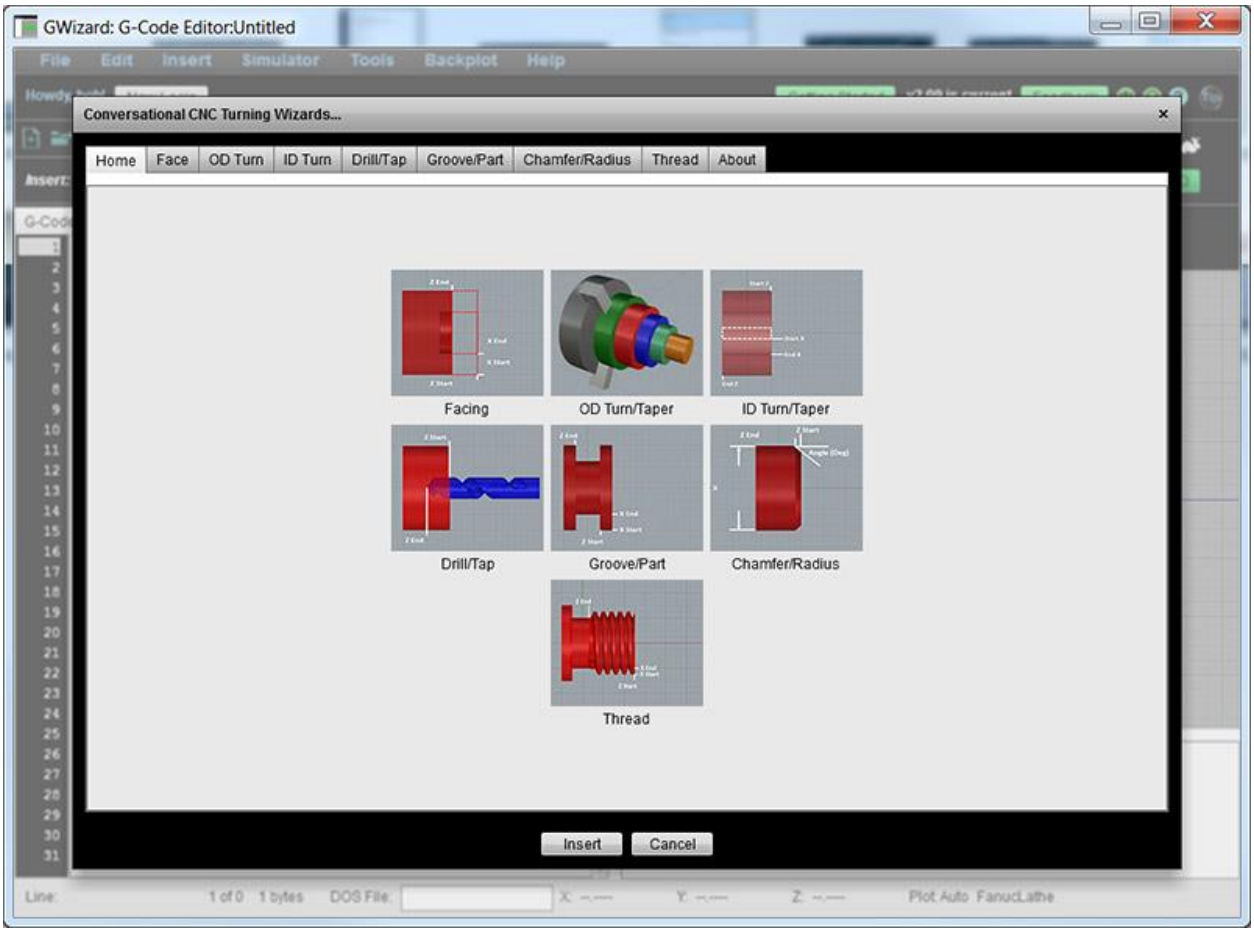
- Custom cycle: A powerful deep hole drilling capability is provided here

- Interpolated: Where an endmill is used to make holes potentially much larger than the diameter of the endmill. This is a powerful technique not available on Manual Machines!

Having answered the questions, you click the "Insert" button and finished g-code is placed in the file you're editing. It will be fully documented with comments and ready to run.

CONVERSATIONAL CNC FOR TURNING

Conversational CNC for a Lathe is very similar. Here is the graphical menu of lathe operations:



Conversational Operations for Lathe

The lathe has its own basic operations:

- Facing

- OD Turning and Tapers

- ID Turning and Tapers

- Drilling

- Tapping

- Grooving

- Parting

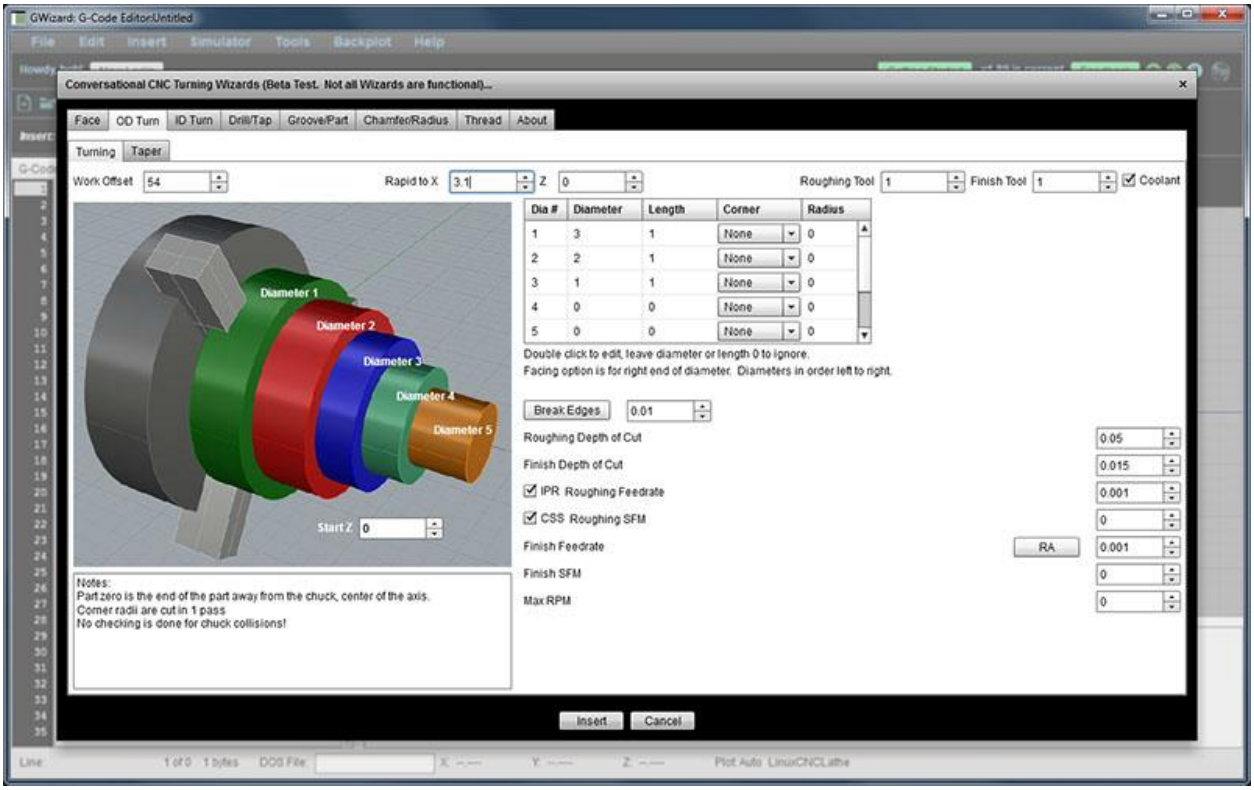
- Chamfer

- Radius

- Threading: ID and OD

It's so much easier to perform many of these operations using Conversational CNC than to try to use a Manual Lathe. You'd never get the taper attachment set up or threading dialed in before the CNC was done. For some of these things you'd need special tooling such as a ball turning fixture.

Using the Wizards is much like with the Mill Wizards. Here is the OD Turning Wizard for example:



A quick walkthrough of how to use the Lathe OD Turning Conversational CNC Wizard.

Where Can I Find Conversational CNC?

Conversational CNC is available from a lot of sources. It is an option for many machine tools, though it can seem expensive at the time until you think about the cost savings of not having to run through CADCAM. Click here for a great video showing how to use the Conversational CNC capabilities of the Tormach Lathe.

EXERCISES

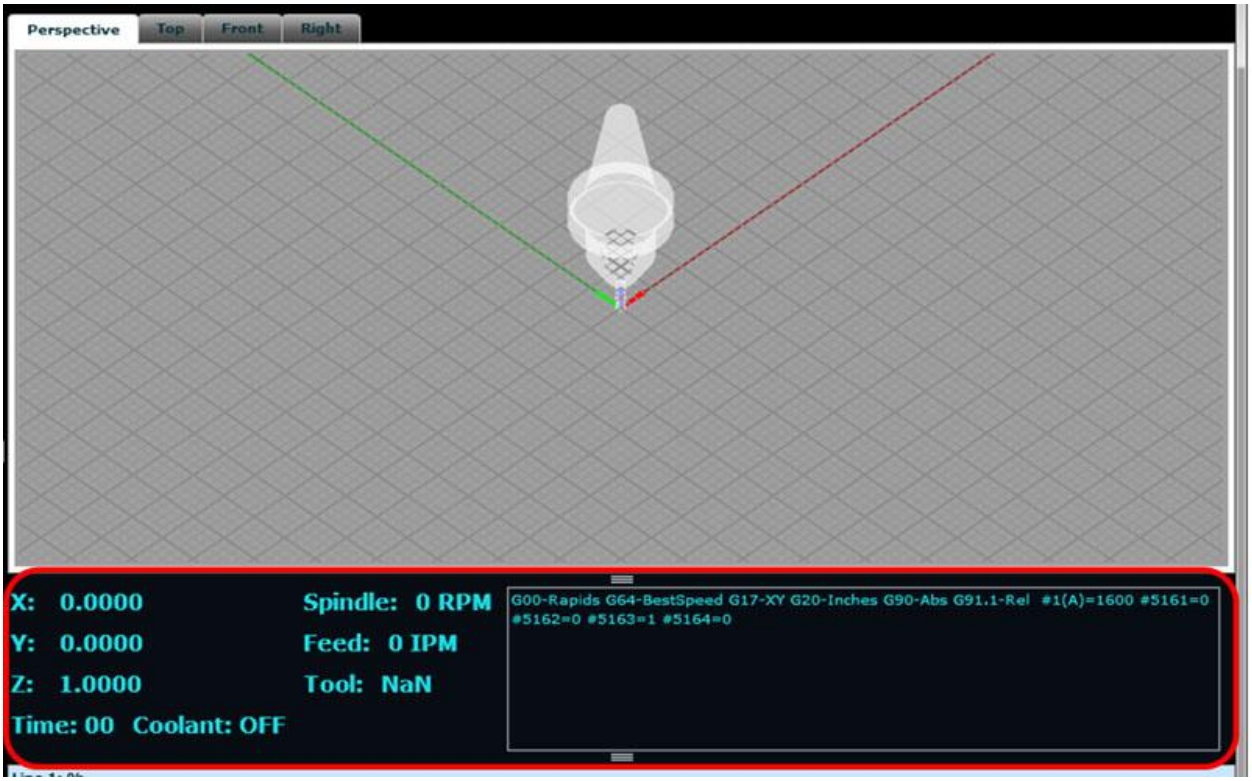
1. Give Conversational CNC a try. Fire up G-Wizard Editor's Conversational CNC Wizards and see if you can create g-code for some simple part. You may find the videos in our G-Wizard University helpful to getting started.

ONE SHOT G-CODES AND MODAL G-CODES

WHAT ARE MODES?

We've mentioned before that there are lots of "Modes" in G-Codes. A "Mode" simply means the CNC Controller is remembering some behavior from the last time you told it what to do. People often think modally, meaning if you don't specify every detail, they fall back to assuming you mean to do things the way you did the last time. Modes save time and make our g-code programs shorter.

When you run a G-Code program in the GWE simulator, it shows you which modes are in effect in the little box under the backplot:



The controller's state and G-Code Modes are shown Below the GWE Backplot...

There are many examples of modes in g-code, not the least of which are the coordinates of the current position. You don't always have to specify X, Y, and Z for every move. You only specify the coordinates that have actually changed.

The exact use of modes is one of the areas that differs from one G-Code Dialect to the next, so make sure you understand which things are modal in your g-code dialect and which are not.

WHAT IS A ONE-SHOT G-CODE?

Speaking of things which are not modal, we have a name for a big category of non-modal behavior which are called *One-Shot G-Codes*. A one-shot g-code is only in effect for the block it is used in, and then the mode goes back to whatever it was before the block was executed. Most of the one-shot g-codes are fairly advanced, so we'll save them for a later chapter.

EXERCISES

1. Get out the programming manual for your control and find at least one example of a modal g-code and one example of a one-shot g-code.

CNC EDITORS: CREATING HAND-TUNED G-CODE

INTRODUCTION

Once you're comfortable with MDI uses for g-code, it's time to get together some tools for doing more extensive g-code programming.

Even if most of your g-code is produced via CAM software, It's important more often than not to be able to do some hand tuning to the g-code. There's also a whole class of simple g-code programs that can be created without recourse to CAM, sometimes much faster.

Let's leave aside the issue of writing g-code from scratch for a minute, and look at some scenarios where it might be handy to tune up some CAM produced g-code. There's a bunch of different cases where this can be really handy and productive. Suppose, for example, that you want to change the feeds and speeds in some portion of the program and you don't want to have to run all the way back through the CAM package to do that. It's not hard to bring up the g-code in an editor and make the changes.

Or, let's say that you get the program down on the shop floor, you load it up in the controller, and you suddenly notice there's an error due to a bug in the CAM software. Often these are simple errors, even one single errant motion. Sometimes its faster just to fix these by editing the code rather than trying to get back into the CAM and "trick" it into not making the error. I know a machinist whose CAM kicks out an errant move whenever it moves to a new work offset. You can see it right in the g-code and delete it. He may have a bad post or some other problem, but he's talked to the CAM people several times and they haven't been able to help, so he just edits the programs.

Suppose the CAM programmer didn't account for the fixturing quite correctly. The part will be cut correctly, but some of the positioning moves are going to run into clamps or what not. It's pretty easy to program by hand around those obstacles.

These are all real simple examples. You can also quickly get to cases where a little hand programming might optimize a program quite a lot. Perhaps you can't quite get the CAM program to do things the way you want it to, but you can get it to do some parts and you can accomplish the rest with a little judicious hand programming.

A couple of other case for hand working g-code:

- You don't have the CAM any longer but need to be able to change the program.

- You want to adapt the program to the differences between controllers without re-running the CAM against a different post-processor.

The list goes on, but you get the idea. A good CNC Editor is a handy thing to have on hand!

FEATURE BUYING GUIDE

What are some of the key features to look for when shopping for a CNC Editor? They fall into these key categories, and we'll run through some of the more important features for each one.

- Text Editing Features: These features determine how easy it is to edit the text of your program.
- Informational and Power Editing Features: These features give you extra power in understanding your program, or in creating it quickly.
- Program Revision Features: The features help you to make mass revisions to your program, for example, to renumber the program lines.
- G-Code Simulator and Backplotter Features: These features allow you to watch the g-code program execute in simulation.

TEXT EDITING FEATURES

There are programmers out there using just Windows Notepad or perhaps a generic (non g-code) programmer's editor of some kind. Nothing wrong with that, but having a text editor that actually "knows" g-code and was designed for it can result in a more productive environment. Before we get too far into the g-code specific features, make sure your editor has the typical features any good text editor should have and that you're used to. The basics like clipboard support, multi-level undo, and a powerful search and replace capability. You're no doubt already used to some kind of editor, see how the CNC Editor you're checking into compares with it in terms of the features you're already used to.

Beyond those basics, here are some other capabilities you may want to have:

Keyboard Shortcuts

Touch typists realize pretty quickly that having to jump back and forth between mouse and keyboard slows them down quite a bit. Look for editors were you can accomplish most everything with either the keyboard or mouse so that you don't have to leave the keyboard if you don't want to.

Find and Replace

G-code programmers use find and replace a lot, so it's important to have a good one. CNCCookbook's G-Wizard Editor uses a special Find/Replace toolbar to make find/replace into a true power tool. With the toolbar, there is no popup obscuring part of your program that you have to peer around. You can seamlessly move back and forth between editing and find/replace as you move through the program. Accessing Find/Replace is easily done with the mouse via menu or toolbar and from the keyboard simply by pressing Ctrl + F.

Find/Replace Toolbar is easier than peering around a popup dialog...

Jumping in a G-Code Program

G-code has a pretty regular structure, and some features in it are so important it's worth having a special "Jump" command to move through that structure. For example, you may want to jump to the next g-code "N" block number, actual line number (e.g. N10 might be on the 1st physical line), next error in the program so you can fix it, or to the next Toolchange. Like Find/Replace, it's convenient to have Jump as a toolbar rather than a popup dialog, and it is convenient to access it either with the mouse or the Ctrl + J keyboard shortcut.

Jump toolbar...

Appending or Inserting another File

After a while, you'll have accumulated a whole collection of g-code files. Not long after you'll be wanting to combine these files in various ways. Make sure your CNC Editor can append a file to the one being edited or insert it in place.

Capitalization

A lot of g-code is written in capital letters, except for text inside comments. It's nice to have an editor that can automatically capitalize outside the comments to save where and tear on the shift key as you're typing.

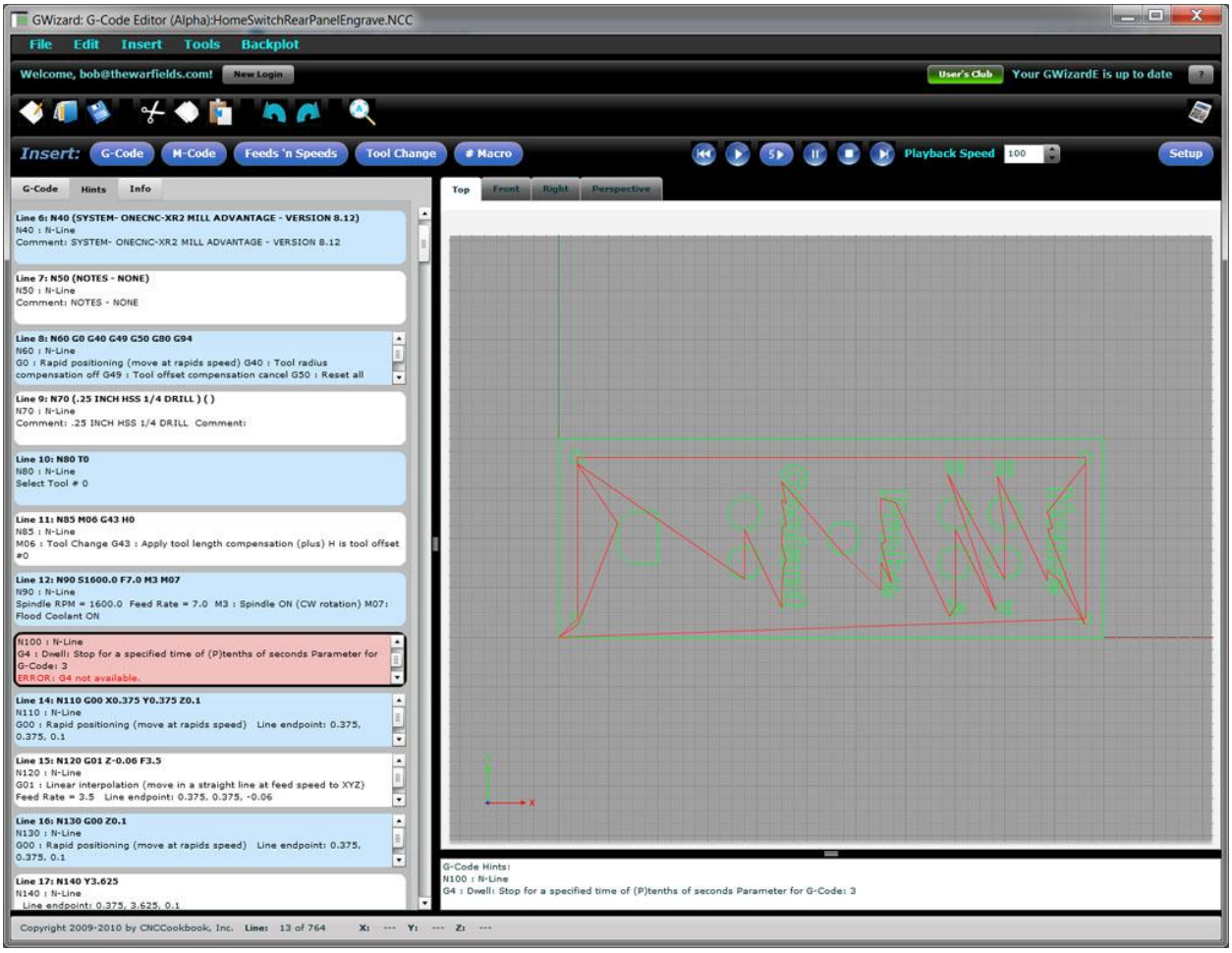
Informational and Power Editing Features

Whether you're a beginner or an expert, you won't always remember every last detail of your controller's g-code dialect--it's only a matter of degree that separates the beginners and the experts. Perhaps you can't quite remember which letters are used with a

particular canned cycle. Maybe you don't remember which G-Code is used for the peck drill cycle. This comes up whether you're reading or writing g-code, so you'll want features that help going both ways. In addition, a lot of useful information is missing from the g-code program. Sure, you could look it up or calculate it (quick, how many degrees does that arc subtend?), but why bother if your CNC Editor will just figure it out for you?

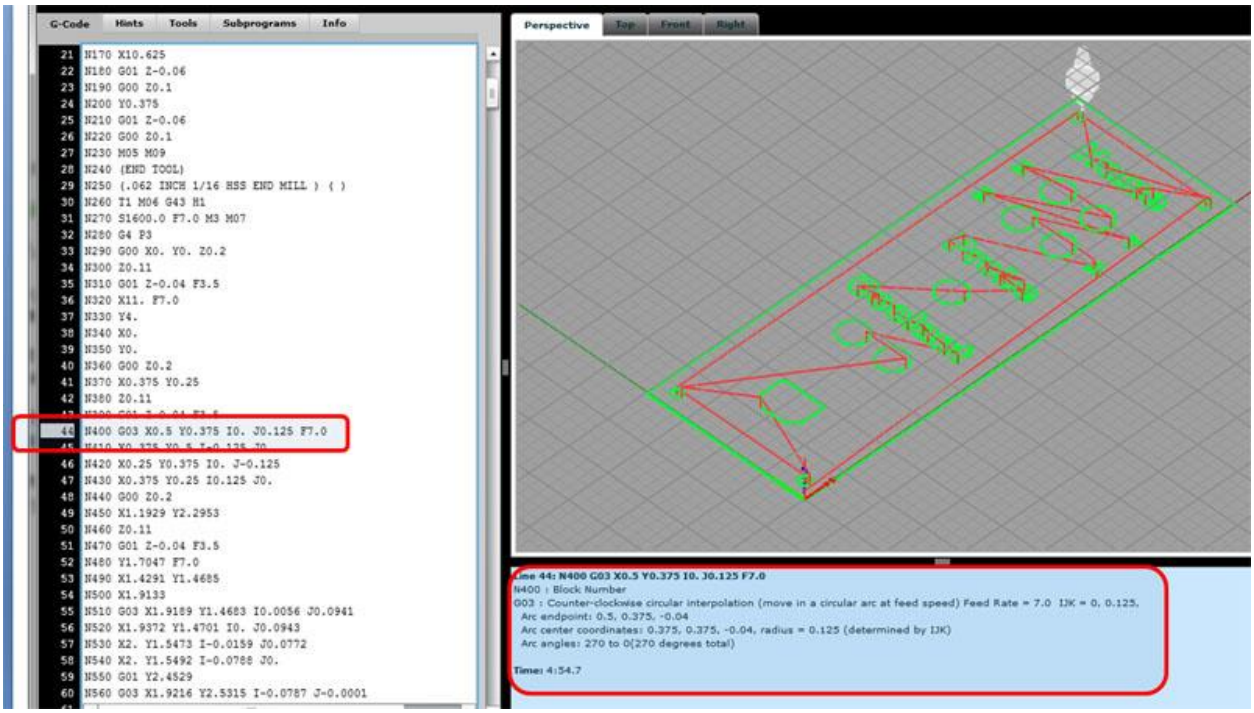
Hints: G-Code in Plain English

Hints are a feature that explains g-code in plain English. This is helpful for learning the g-code as well as for getting a refresher on the details you may have forgotten, and for collecting information that is otherwise not obvious. In the G-Wizard CNC Editor, you can view the Hint for the currently selected line of g-code, or you can switch to the Hints tab and see Hints for all the lines laid out together.



Hints View shows you a quick explanation of what the g-code does and it also shows error messages. For example, the current post does not support the G4 used in this program...

Here is a typical example of a Hint for a line containing an arc:

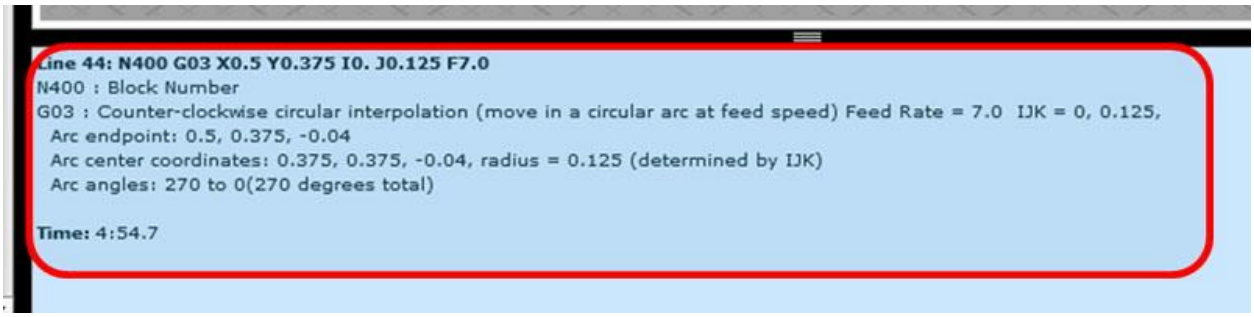


Line N400 has an arc. The Hint below the backplot tells all...

Suppose you haven't learned arcs yet, and are trying to follow the line of g-code:

N400 G03 X0.5 Y0.375 I0. J0.125 F7.0

The fact we don't yet know how arcs work makes the Hint even more valuable. Let's take a close look at the Hint:



The Hint for an Arc...

It tells us the following:

- In **bold** at the top is the original text of the line. This is useful if we're in Hints view and can't see the g-code line text.

Right below the bold, it starts telling us all about what the line does:

- The Block Number is N400

- G03 performs Counter-clockwise circular interpolation. In other words, it tells the machine to move the cutter in a circular arc at feed speed.

- The Feed Rate was set to 7.0 on this line.

- We see what the IJK values parse to be. In this case, we have I=0, J=0.125, and no K

- The endpoint of the arc after the move is 0.5, 0.375, -0.04

- The coordinates of the center of the arc are 0.375, 0.375, -0.04.

- The center was determined by IJK (as opposed to R, which is another way)

- The angle of the arc runs from 270 degrees to 0, a total of 270 degrees.

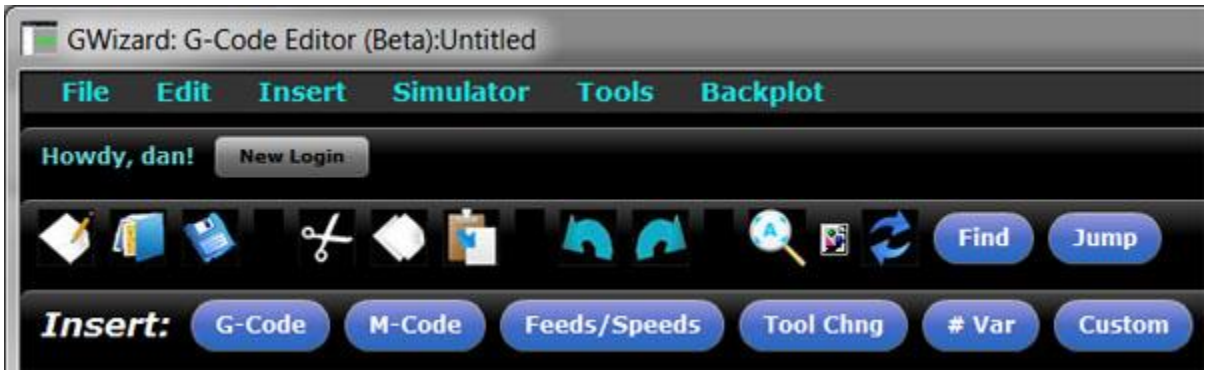
After all that, the Hint skips a line and gives us a time. This is telling us how long it will take the g-code program to reach this line and finish it if we execute on the machine. In this case, we can see the arc will be completed 4 minutes and 54.7 seconds after the program starts executing.

Phew, that's a lot of data spewed forth, but it can be extremely handy to have when you're trying to figure out what a g-code program is doing or what's wrong with one. And, like I said, it's hard for an expert to tell all of this stuff at a glance.

GWE gives this level of detail for each and every line of the program.

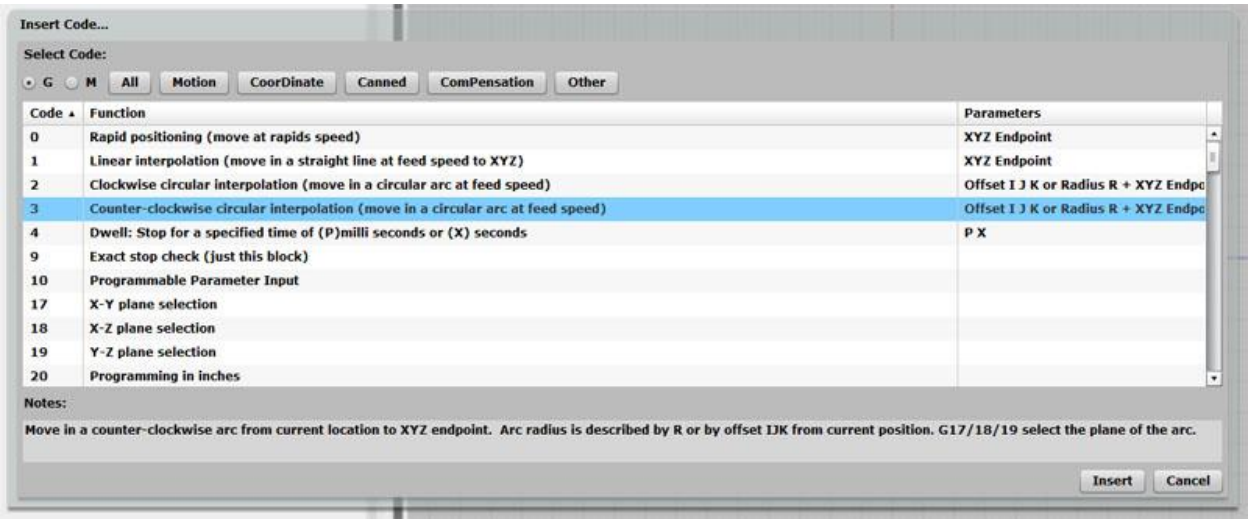
WIZARDS: REMEMBERING THE DETAILS FOR YOU

G-Code is powerful, but it is also somewhat arbitrary. Wizards are there to help you overcome the arbitrary part by remembering the details for you. If Hints are there to help with reading G-Code, Wizards are there to help with writing G-Code. In G-Wizard CNC Editor, the Wizards may be triggered either via Toolbar with the mouse or via Keyboard shortcut:



Select from 6 different Wizards on the Toolbar for G-Codes, M-Codes, etc...

After you've chosen the Wizard, it pops open with more options. For example, here is the G-Code Wizard:



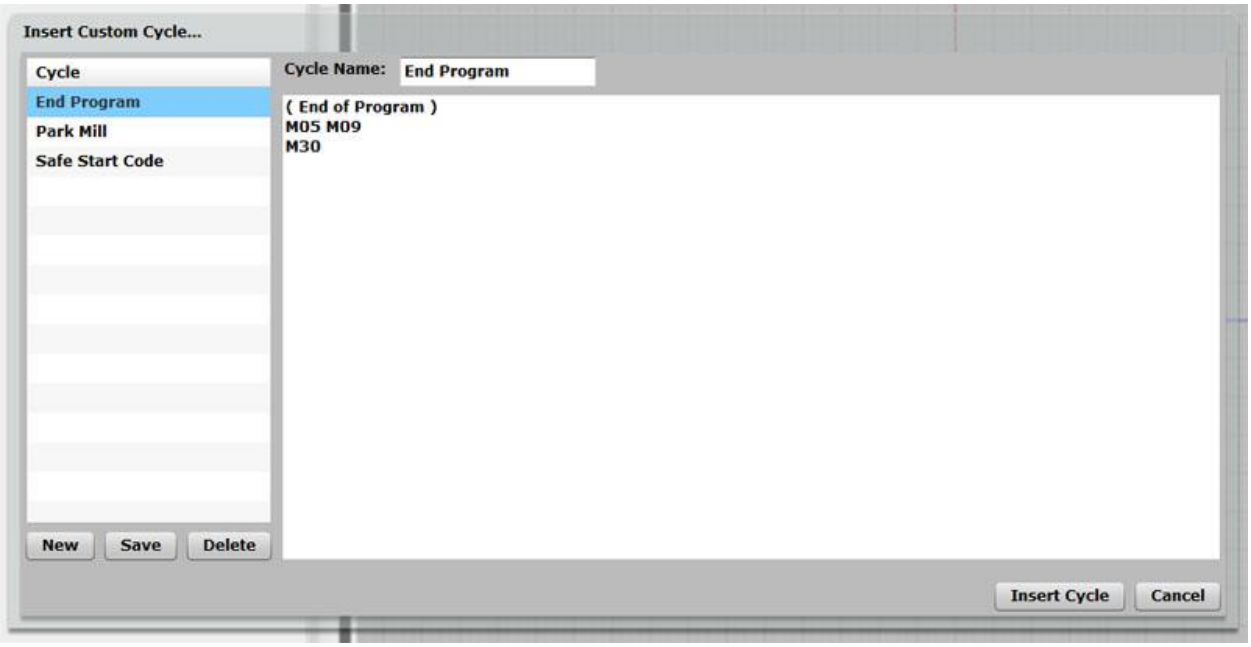
The Popup Wizard for G-Codes...

From the Wizard, you can just key in a number directly: for example, type Ctrl + G then keep typing "3" then enter and a G03 is entered. If you get in the habit of typing "Ctrl-G" instead of just "G", the Wizard will be there to catch you if you forget how the code works. If you close your eyes, you'd never know it was there. Once it's up, you can scroll through the options, look at them by category (e..g "Motion", CoorDinate", etc.),

see what parameters are used, and also see the hint information (under "Notes") associated with that G-Code.

CODE SNIPPETS AS CUSTOM CANNED CYCLES

Over time, you're going to build up a library of code snippets that you like to use when programming. G-Wizard makes it easy to manage these as "Custom Cycles". Just click the "Custom" Wizard and you'll see how it works:



You can create new cycles, edit the existing ones, or insert the code into your program. GWE provides three default cycles for you to start with:

- End Program: The set of steps you want to do at the end of every program.

- Park Mill: It's nice to configure some code that "parks" the mill table for easy loading and unloading by the operator.

- Safe Start Code: The steps you want to begin every program with so it starts out in a known safe state.

PROGRAM SUMMARY INFORMATION: WHAT'S THE BIG PICTURE?

In addition to detailed specific information on each g-code line, G-Wizard Editor also provides useful overall information about the program. GWE has an "Info" tab that holds the overall information. Take a look at the illustration to the right to see what the Info tab looks like. As you can see, it tells you a variety of information about your g-code program including:

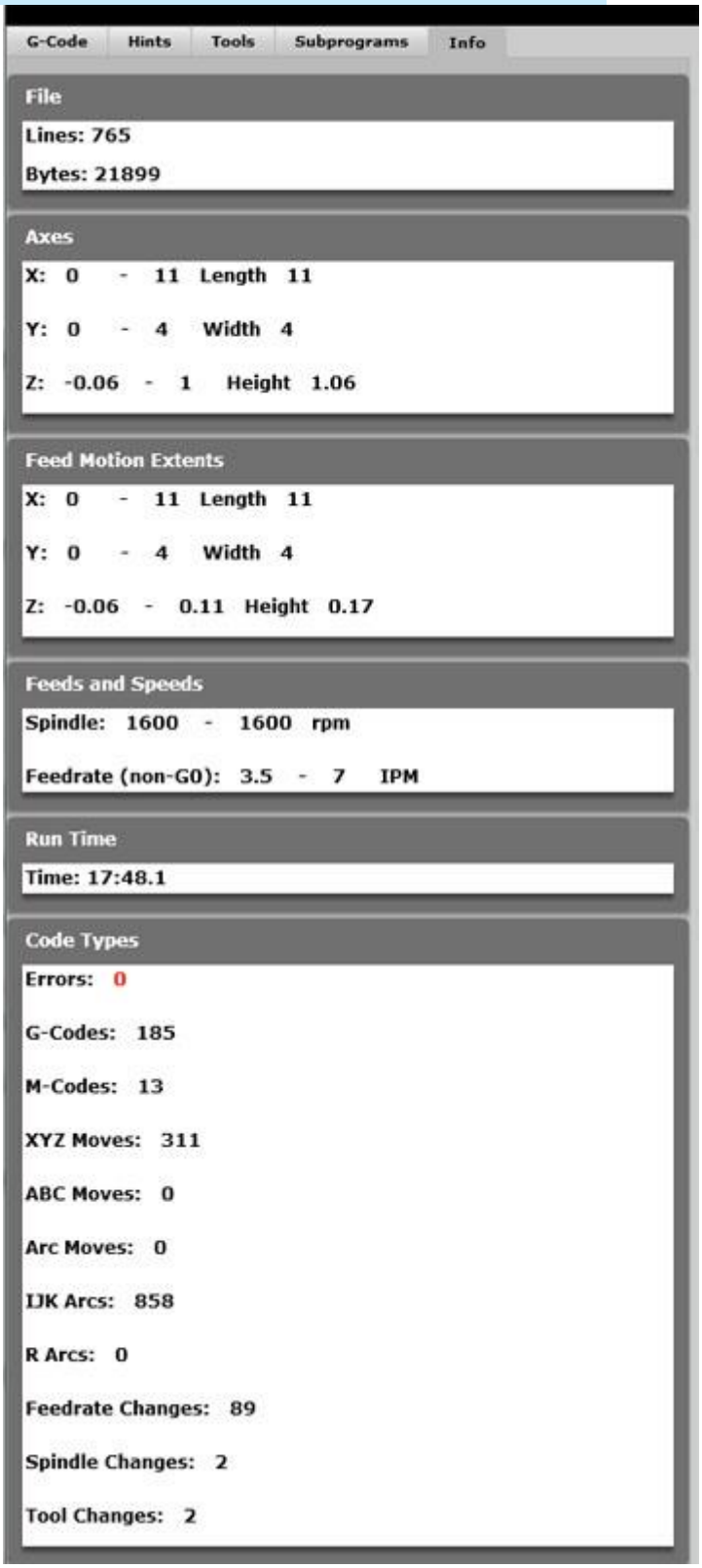
- It's size both in terms of lines and bytes. If you're trying to make your program fit in the limited RAM memory available to an older controller, it's important to be able to tell quickly how you're doing.

- For each axis, GWE will spell out the range of motion used as well as the length, width, and height these ranges imply. You get the information both overall and in terms of just feedrate motion. The rapids will nearly always extend outside the feedrate envelope, but the feedrate envelope may give you a good idea of the rough stock size needed by the program.

- You get to see the range of spindle rpm's used as well as the range of feedrates used in the program.

- You get to see the overall predicted run time of the program.

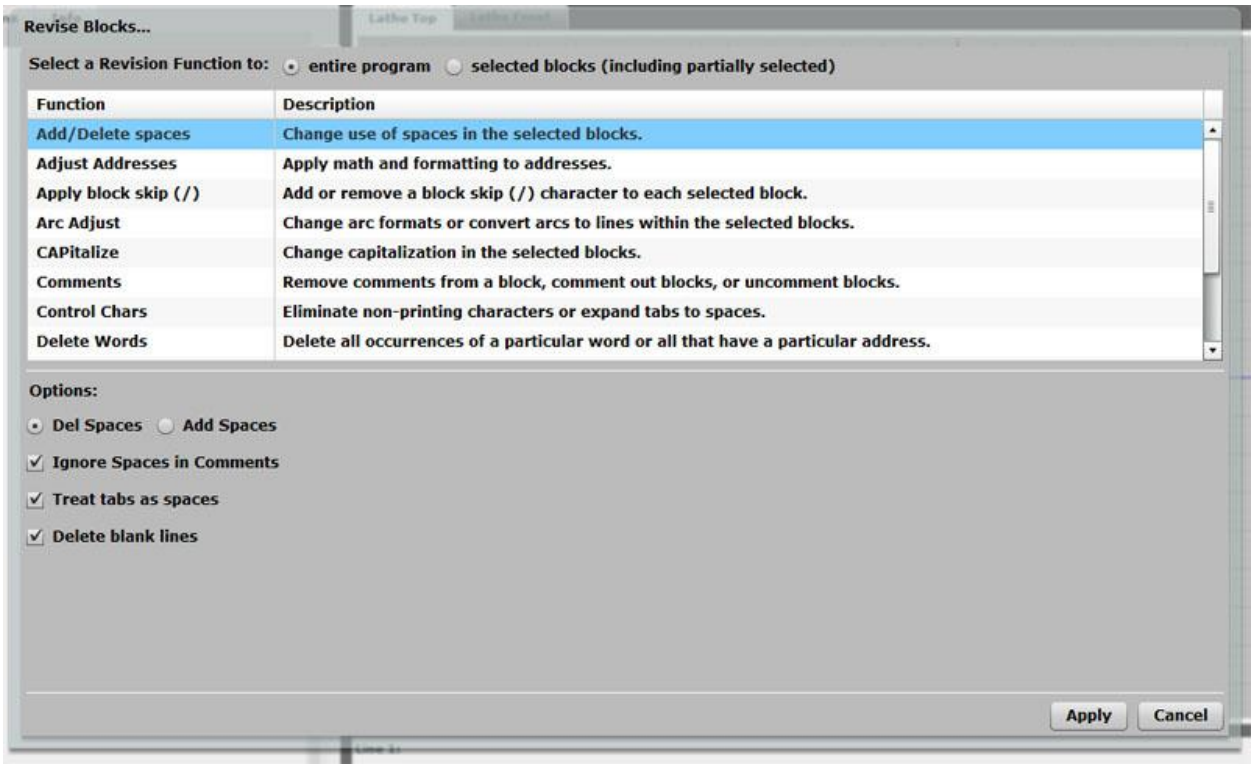
- Lastly, there is a count of the different classes of operations within the program, as well as a count of the number of errors GWE discovered in your program.



PROGRAM REVISION FEATURES

You've got a big g-code program, and you want to make some pretty major changes that could affect many lines of code. What features does your CNC Editor offer to facilitate those changes?

In GW CNC Editor, these features are called, naturally enough "Revisions". There's a long list of them available from the Tools Revisions popup:



Making G-Code Revisions is easy with the right tools...

Typical Revisions you'll want to be able to make include:

- Adding or deleting spaces and blank lines.
- Adding or removing block skip characters (/) over sections of g-code.
- Changing capitalization
- Removing comments from blocks and commenting out or uncommenting blocks.
- Eliminating control, garbage, and other non-printing characters.
- Expanding tabs to spaces
- Renumbering block sequence numbers ("N" numbers)
- etc.

G-Wizard Editor includes a long list of different Revisions, and you can check them all out on the Revisions Page.

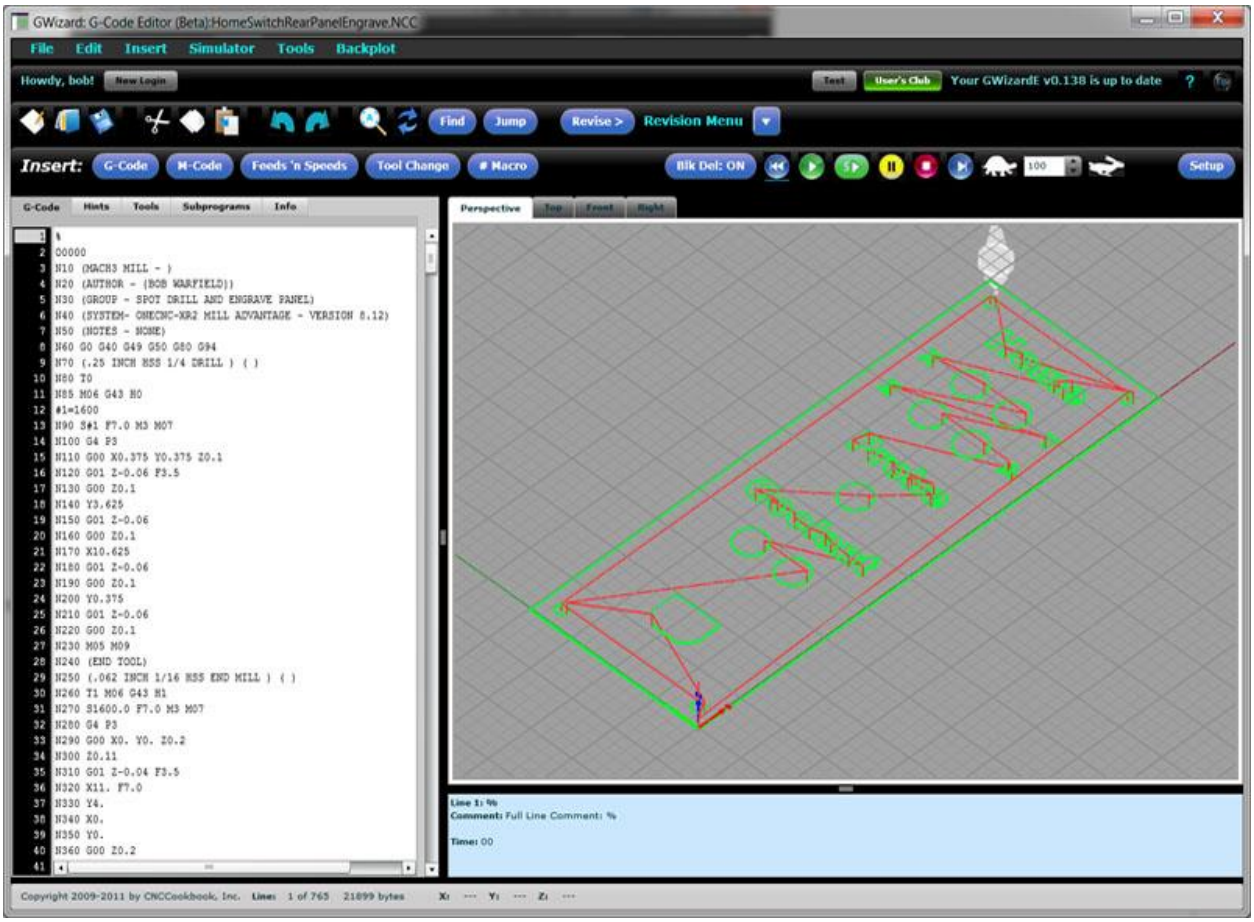
CNC SIMULATORS, BACKPLOTS, AND VIEWERS

THERE'S TIMES WHEN YOU WANT A SECOND OPINION

Any time something serious is going on medically, wouldn't you seek a second opinion? Considering the kinds of trouble your machine can get into if the g-code goes awry, shouldn't you get it a second opinion too? It's actually cheap and easy to get that second opinion. You need a piece of software that has variously been called a CNC Simulator, CNC Backplot, or G-Code Viewer. You'll also hear the terms "Verification", "Emulation", and "Virtual CNC" batted around, though they're a little less common than the other three. CNC Verification can sometimes have a little different meaning, referring to the high end of the market for this kind of software, so let's leave it to talk about until the end. Meanwhile, let's take a look at CNC Simulation Software, and we'll use that name for the rest of the tutorial.

WHAT IS A CNC SIMULATOR?

Think of a CNC Simulator (or g-code simulator if you prefer) as a piece of software that can execute g-code programs, but instead of controlling servo motors and a spindle on a machine, a simulator produces a graphical display of what the machine would do. That graphical display is often referred to as a backplot. It shows the toolpath your cutter will follow if the g-code program is executed. Here is a backplot produced by the g-code simulator that's built into the G-Wizard G-Code Editor:



The CNC Backplot is the graphical display on the right...

In this particular g-code program, we're doing some engraving for an electrical panel that's part of a CNC controller I put together.

CUSTOMIZATION: HAVING A "POST" IN THE CNC EDITOR

As we've discussed, g-code has "dialects" depending on which controller you use. You may be thinking that since all your machines are Haas, or all your controllers are Fanuc, that a Fanuc Simulator or a Haas Simulator would be fine. But you never know when you might bring in a different machine and be wishing your Simulator could deal with it. Also, your g-code editor and simulator software should be capable of understanding as many of these nuances as possible. In CAM software, the nuances are handled by what's called the "Postprocessor" or "Post". You want your editor to have a "Post" too. For more, seeour page on G-Code Dialects.

NAVIGATION IN A CNC SIMULATOR'S BACKPLOT

A good CNC Simulator can tell you all sorts of useful information. Let's go through some of the information we can glean about the electrical panel engraving program from the screen shot above.

First of all, the graphics can often tell you at a glance whether the tool path is what you're expecting. Be on the lookout for any sudden unexplained departures outside the envelope you're expecting. The red lines are rapids motion and are red because rapiding into the workpiece is a bad thing to be on the lookout for. Green represents motions made at feedrate speeds.

In G-Wizard editor you can rotate the backplot by holding down the right mouse button while moving the mouse on the Perspective view. In the other views, such as Top, Front, and Right, holding down the right mouse button lets you pan the view.

You may need to zoom in close to see what's happening. The mouse scroll wheel controls zoom. If you get lost in the forest of lines, simply press the icon on the toolbar that looks like a magnifying glass with an "A" in the middle. This is the "Zoom Extents" command, and it zooms the display and centers it based on the full extent of the motions for your program.

If you want to redraw the backplot from scratch, there is a "Backplot" button as well--the two arrows circling one another. Pressing the Backplot button forces re-parsing and re-plotting of the entire program.

The grid is there to give you an idea of position and size. It's calibrated according to whether your Post is currently set for Imperial or Metric. Down on the status line at the bottom are X, Y, and Z coordinates of the tip of the mouse cursor, so you can point at something on the backplot to see its coordinates.

WHY USE A G-CODE SIMULATOR?

Are you starting to get some ideas for how a g-code simulator could be useful to you? As you can see, a good simulator throws off a wealth of information to help you understand what's going on in the program. Machinists need to know these things for a variety of reasons.

Here are some possibilities to think about:

- **Learning and Training**: CNC Simulators are great for getting a better understanding of what the g-code is doing, how specific g-codes work, and in general, answering "What will it do if..." sorts of questions. If you're trying to learn the ins and outs of a controller that is slightly different than the one you "grew up with", a simulator can be a good place to start getting used to the differences of the G-Code Dialects.

- **Quick Sanity Check**: You just posted some g-code out of your CAM program. Why not bring it up for a quick check in a good CNC Simulator. Are there any obvious errors being flagged? Does a quick visual check indicate nothing obvious is amiss? Most CAM programs have bugs (heck, all software does, the CAM people certainly aren't immune). Most of the time, their backplots and simulations are not true G-Code simulations. They're just geometry plots. If there is any kind of error in the post processing, the CAM simulator will not show you a faithful rendition of what your machine will be doing. I knew one machinist whose CAM program would periodically throw a wild rapid move in when he used work offsets. It was an easy fix, provided he caught it before running it on the machine. It became not only easy, but essential to do the quick sanity check on a CNC Simulator before running each program. After all, in many cases, the CAM program isn't sitting right next to the machine. Do everything you can to make sure the program is right before trying to load it and execute it.

- **Debugging Hand Written Code**: Obviously if you're writing g-code by hand, a CNC Simulator can be a real labor saver. In fact, I can't imagine how you do it without one.

- **Tracking Down Subtle Errors**: You've got a g-code program. It appears to have no errors, your machine runs it okay, but when the resulting part is finished, there are problems. A CNC Simulator that can give you detailed information such as we worked through on the arc might help you to track down the source of the error and correct it, assuming the error stems from the g-code and not some other source.

- **Trying Out New Ideas**: Suppose you want to compare some different approaches to a part program. Is it faster to interpolate a hole to begin pocketing, or are you better of running an extra toolchange so you can make the hole faster with a 1" indexable drill? These kinds of things are ideal to compare with simulations, assuming they have the ability to accurately predict the run time of your g-code program. When you consider that some complex expensive parts can spend many hours if not days on machines, it only makes sense to figure out everything you can about how to optimize the job before you get to the point of putting it on a machine.

There are probably lots of other reasons, but the bottom line is you can quickly tell what's going on with your CNC program without risk to machine, materials, or tooling.

ERROR CHECKING FEATURES

Every error you find in your editor or simulator is an error you don't have to find at the machine, which saves you time and money. The more extensive the error checking your simulator can do, the better. It's ability to check errors will be limited by the precision of the Post. The more detail it understands about your controller, the more accurate it can call out issues that the controller may object to. Another important capability is being able to ignore any error message. Sometimes errors should be treated more like warnings. Sometimes they're not quite right because of controller parameters that may change the controller's behavior. Either way, it's convenient to be able to ignore them when you want to.

In the G-Wizard CNC Editor / Simulator, errors are displayed in the Hints view (see the prior chapter on CNC Editors for more on Hints).

TOOL DATA MANAGEMENT

This is a critical, and often overlooked function. Your editor should be able to maintain a Tool Crib or Tool Table. It should be possible to import and export these tables so they'll stay in sync with your CAM program and the Tool Changers on your machines. Just as running a CNC machine without knowledge of the tools in the changer is bound to lead to trouble, so is running your simulator that way. For more on the Tool Data Management capabilities of G-Wizard Editor and other software, be sure to check our two part series on TDM:

Part 1 is the basics of Tool Length and other Tool Data.

Part 2 goes into topics like Tool Presetters and software to manage Tool Data.

It's not part of the tutorial, but you should pop over and at least scan the articles some time so you know what the issues and concepts are for Tool Data Management.

WHAT IS G-CODE VERIFICATION?

As mentioned above, CNC Verification often refers to a higher level of functionality than straightforward CNC Simulation and Backplotting. A full CNC Verification package will include capabilities such as:

- Full 3D simulation of the cutting. This is a basic requirement for verification and could as much be considered part of higher end CNC simulation as verification.

- 3D Import and Display: Some capacity to import 3D models, usually in the form of .STL files (Stereo Lithography, a common 3D file format). For example, you might import models of the machine, fixtures, cutters, or the finished part.

- Gouge detection: Gouge detection involves detecting a number of conditions, all of them bad. If the tool moves too quick, say at rapids, into the workpiece, that's a gouge. If the tool moves into the fixtures (such as your Kurt Vise) or worse the machine itself, that's also a gouge. Good verification software detects all of these conditions.

- Tolerance verification: With tolerance verification, you can import a 3D solid model of your finished part and the verification software will tell you how the simulated cut model deviates from it. Ideally, it can show you a color coded diagram that is marked with all the places where the g-code program either cut too deep or not deep enough based on the tolerances you've established.

- Full Machine Kinematics Simulation. This is the ability to show a faithful 3D reproduction of the machine with accurate modeling of everything from the machine's physical travel limits on to even the exact performance envelope of the machine.

It's tough to say exactly where CNC Simulation ends and Verification begins, but just because a package calls itself a Verification package doesn't make it so. In particular, if it can't do the 3D import of fixtures and finished part, if it can't do gouge detection, and if it can't do tolerance verification, you shouldn't regard it as a verification package. It's a fancy simulation package. True Verification software tends to be quite expensive, so caveat emptor (beware the buyer) if you seem to have found a cheap one. It may be too good to be true.

Simulators and Verifiers are extremely useful tools for CNC. There are a lot of other software tools available too. See our Digital Tooling article for a good overview of the different types that are available.

EXERCISES

1. Get some sample g-code and load it into G-Wizard Editor so you can play with the backplotting capability. If you don't have any samples, try our sample file page for some downloads, including the files displayed in the screen shots for this G-Code tutorial.

2. Experiment with the different views available in G-Wizard Editor.

3. Experiment with both Mill and Lathe sample g-code. Remember, you'll have to change your machine profile from Mill to Lathe and be sure an appropriate controller profile is selected to view the different file types properly.

PART ZERO, TOUCH OFFS, AND ZEROING

LET'S START WITH PART ZERO, ALSO KNOWN AND PROGRAM ZERO

We've already discussed CNC Coordinate Systems in a previous article, so check back there if you haven't already covered that material.

Let's suppose you just got done drawing up a part in your CAD software, and you're about ready to generate some g-code for it. One of the key things to understand is where Part Zero is going to be. Your CAD program has some sort of coordinate system, and your part is positioned in the drawing relative to that coordinate system. If you've never done any CNC work before, you may not have paid much attention to that positioning. Perhaps you stuck the part well away from the 0, 0, 0 origin in the CAD program so it would be easier to see without the axis lines being too close.

You might want to reconsider that idea, at least until you get very comfortable with all the different coordinate systems that you'll be using for CNC. Instead, what you want to do is put your "Part Zero" (for now, the CAD System's origin or 0, 0, 0) some place that makes sense when you get ready to machine the material. When your g-code program refers to X0 Y0 Z0, that's your Part Zero. Later on we can get all fancy with Work Offsets and other ways of transforming the coordinates, but when you first start the machine, think of X0 Y0 Z0 as Part Zero.

There are a lot of different theories on where to put Part Zero, and it matters for how easy and natural your CNC work will be.

When milling, a lot of emphasis is on the Z axis. When Z = 0, where should that be in relation to the part?

One theory has Z = 0 being the top of the workpiece BEFORE MACHINING. This facilitates knowing when your cutter is cutting workpiece and when it is cutting air. Of course as you start making chips, you're also making air down below Z = 0, but it is still a comfort to know where that original boundary started.

Another theory prefers that Z = 0 be some feature that doesn't move and will not be milled away. For example, it might be the top of a vise jaw. This is handy should you need to remove your part for some reason. You won't have to re-reference the machine to a new Z0. It is also handy if you're machining workpieces with slightly differing dimensions. For example, even if you're making identical parts, you may be starting with rough sawn material. The exact coordinates of the top of such material will vary from workpiece to workpiece because sawing is not a precision operation.

Cookbook Recipe: I like to use a Part Zero that corresponds to the fixed jaw of my vise when I will use a vise for machining. Once you get used to making your CAD drawings with that in mind, it means you can walk up to the machine, stick a piece of material in

the vise, load a g-code program designed with that notion of Part Zero, and immediately begin machining after just homing the machine.

Since the vise generally stays put on the machine, there are no touch offs required, which is a nice productivity booster. If I do need to move the vise or change jaws, no worries, I can just re-zero on that location again.

Whatever you decide to use for your Part Zero, you need to be aware of it, and it is worth thinking about how to choose a Part Zero that might save you a little time or make things easier to understand.

WCS: Another Name for Part Zero

You'll also hear people refer to a "WCS" or "Work Coordinate System". Part Zero is the origin of a WCS. Put another way, it establishes a WCS by defining its origin. Your CAM program will have a way of specifying the WCS or Part Zero.

WHY YOU WANT A MACHINE WITH ACCURATE HOME SWITCHES

If you have a professional CNC machine, it's probably got Accurate Home Switches. But if you're a hobbyist who has built or retrofitted a machine, you may not have Home Switches or they may not be accurate enough for repeatable precision. If you don't, you've made yourself a whole lot of extra work and you'll soon want to add Home Switches.

Without them, any time you start the machine, you'll have to use an Edge Finder or other method to accurately locate Part Zero. You can't rely on the machine to remember it because it may have been shifted while the power was off, or it may have lost steps if it has stepper motors instead of servos. If you crash or have to emergency stop the machine, you'll have to manually re-zero it with that Edge Finder in order to get it re-zeroed.

All this is a real pain and time sink. You won't want to run a machine without Accurate Home Switches for very long!

WORK COORDINATES VS MACHINE COORDINATES

When you start up the machine, it doesn't necessarily know anything about your preferred coordinate system. What it does know is something called "Machine Coordinates". This is a fixed coordinate system that is baked into the machine. When you "Home" the machine, or "Reference the Axes", you are causing it to use its Home Switches to accurately locate itself relative to machine coordinates. If your machine doesn't automatically home when you start it up, it's a good idea to get used to the idea of homing it before you do much else. If you crash or emergency stop, it can also be a good idea to home the machine so it can pick up its lost position.

"Work Coordinates" are the coordinates you want to think about. In other words, Work Coordinates are the ones where the machine is at Part Zero when its display shows X0 Y0 Z0. For that reason, Part Zero may also be called Work Zero. You can establish Work Coordinates in a variety of ways. By "establish", I mean you can tell the machine how to equate Work Coordinates to Machine Coordinates.

A Work Coordinate system is something your machine will remember from one invocation to the next, though you probably shouldn't count on that unless you know for sure you can. Since I use the system of Part Zero matching a point on my vise jaws, I can start the machine and Home it and I know the Work Coordinates are what I expect. You also have the ability to establish multiple Work Coordinate systems, which is convenient for a lot of reasons. We'll talk more about using multiple Work Coordinate systems in a later article. For now, let's just focus on one.

ESTABLISHING A WORK COORDINATE SYSTEM VIA "TOUCH OFFS" OR "ZEROING"

Let's talk about establishing a Work Coordinate system via Touch Offs. We'll use my vise jaw system, just to make the discussion concrete, but the principle works for any work coordinate system.

Simply put, a "Touch Off" is where you use the cutter to locate the Work Zero. We do it one axis at a time, so let's start with the "Z" axis. There are lots of ways to do Touch Offs. Each has varying accuracy and requires you to work on your technique a bit. The Old School method uses paper--cigarette rolling papers were very thin and commonly available. Use a little dab of oil to hold the paper in place and slowly jog the spinning cutter until it moves the paper. Stop. The cutter is now located at the Zero, with the exception of the thickness of the paper. Some trial cutting and a micrometer will establish what that is. Be sure to use the same kind of paper each time so the thickness is repeatable.

A more modern and accurate method would involve the use of a gage block. Gage blocks are precision machined to a very high tolerance and will include an inspection report that tells how much error there is in the block.

DO NOT TRY TO TOUCH THE TOOL OFF THE GAGE BLOCK!

If you're using gage blocks, your cutter should not be spinning. But whether the cutter is spinning or not, it's bad for your expensive gage blocks and bad for your cutters. Instead, move the cutter up, stop moving, and try to slide the gage block between the cutter and the workpiece. At some point, you will have jogged the machine a little too far and can jog back until you can slide between the two.

Here's another tip from a reader (thanks Paul!) if you don't want to use gage blocks--try a wrist pin from an engine. They're machined from hardened material, they're precise, they usually have a fine finish, and you can roll them under the cutter to check the fit. In fact, from many standpoints, a cylinder or ball shape (large ball bearings are precise

too!) makes a lot of sense for this measurement as they're less sensitive to whether the surface under them is flat and level. Take your micrometer to establish the wrist pin's diameter and make sure it isn't worn too badly if it's used.

When you have located the machine in one axis at a point you want to "Zero", your CNC Control will have a way for you to tell it that's the zero for that axis. This is an important operation, so make sure you know how to do it on your controller. There's usually a one touch button to zero a given axis and perhaps another to zero all the axes.

Note that you don't have to measure strictly Part Zero. Your controller will have a way for you to enter an arbitrary value in and tell it that is where the tool tip is currently located. This is convenient for many cases and something you'll be doing fairly often as well as Zeroing. For example, you may want to enter the thickness of your cigarette paper instead of "0.0000."

EDGE FINDERS AND PROBES FOR ESTABLISHING WORK COORDINATES

You won't be CNC'ing for long before you'll be wishing for an Edge Finder or a Probe. These are tools that make it quick and easy to find the edge of some object so you can Zero on it. Edge Finders come in all shapes and sizes from simple spinning contraptions all the way to fancy, accurate, and easy to use contraptions such as the Haimer 3D "Taster". Yep, that's not a misprint, they call them "Tasters" from the original German.

Tormach has a good video tutorial on how to use a simple edge finder. A more versatile tool for these tasks is the so-called “3d Taster” made by Haimer. Check out this video to see how to use a 3D Taster. And for the ultimate in convenience for doing these kinds of operations, here is video of a Renishaw Probe being used to set up work offsets.

each of these tools is similar in purpose, just with increasing capability, automation, and expense. There are a wide variety of other tools available for precisely locating features on your parts and workpieces. Some are more specialized, such as the Blake Coaxial Indicator, which is used to locate bore centers.

You'll want to have some of these gadgets all fixed up in a tool holder and ready to pop into your spindle for job setups.

We won't spend much more time on such things as they're more properly part of CNC setup and general machinist measuring techniques than g-code programming per se.

EXERCISES

1. Take out your CNC machine's manual and figure out how to zero your CNC machine to establish Work Coordinates. Look up how to read Machine Coordinates versus Work Coordinates on the control panel too.

2. Try some touch offs on your machine. Use the corner of a piece of scrap stuck in a vise for starters until you get good at it.

3. If you have an edge finder, 3D Taster, or probe, give it a try as a way of precisely locating part zero.

4. Decide what your convention for Z = 0 and perhaps Part Zero will be and stick to it.

BASIC G-CODE PROGRAM STRUCTURE

BLOCKS = LINES OF G-CODE

A line of g-code is commonly called a "Block". Up until now, we've been typing in one block at a time rather than trying to create full g-code program. You can get a lot done that way. As I've been saying, it's like having a really nice manual machine with DRO's and Power Feeds on all axes. But, there's a lot more power waiting to be unleashed if we delve a little deeper into the g-code language.

So the first thing you have to know is how to create blocks since they're the basic unit of a g-code part program.

BEGINNING A BLOCK

A block can contain a number of optional parts. Certain parts of a block have to come first if they are to be present in the block. In particular, it can optionally begin with a Sequence Number, a Block Skip, a Program Number, or a Tape Start/End character ("%"). Let's look at the function of each of these beginning parts, and remember, a block doesn't have to have any of them.

Tape Start/End

The percent "%" is used to denote the beginning and end of a program. Put one as the first line (block) and one as the last line (block) of any g-code program. Not all controllers require this. Most will ignore anything after the "%", so you can put comments there. Such comments are referred to as being in the "Leader" section. All these references to "Leaders" and "Tape" may seem odd to relative newcomers to cnc. They have a historical origin dating back to when nc programs were stored on punched paper tape. There are still machines out there happily churning out parts that worked this way, but by now their controllers have often been upgraded or at least the tape reader has been replaced with a serial port or USB key front end.

Program Numbers

Programs and subprograms need a number for reference, and this is provided by the Program Number. A Program Number is an "O" word (in g-code we refer to the letters as "words" because each is a "word" telling the controller something to do). Your controller may or may not require a Program Number, but it is a good idea to provide one. A program that has the proper tape start/end and program numbers would look like this:

```
% A simple do-nothing g-code program

O1000
```

%

Not much to it, eh? We'll keep embellishing this little program to make our point as we add various constructs.

Block Skip

The block skip is a handy way to make blocks drop out of the program temporarily without deleting them. Just start the line with a slash ('/') and the controller can ignore the line. These block skips may optionally be numbered on some controllers, and potentially each number can be controlled via the operator control panel. For many Fanuc controllers, up to 9 different block skip numbers may be used. Here is a line with a rapids (G00) move that has been disabled via block skip:

/G00 X0 Y0 Z0

And here is the same line disabled by block skips 1 and 3:

/1/3G00 X0 Y0 Z0

These blocks skips are there so the operator can make quick changes or have optional sections of code that can be turned off and on from the control panel. You probably wont use them much until you're pretty far along in your g-code work, so for now, just be aware that they're available.

Sequence Numbers

We've saved the best for last: sequence numbers. A sequence number provides a unique way of identifying a particular block. You don't have to sequence all the blocks, but it is common to do so. If our simple do-nothing program suddenly had a line that did something, and had a sequence number, it might look like this:

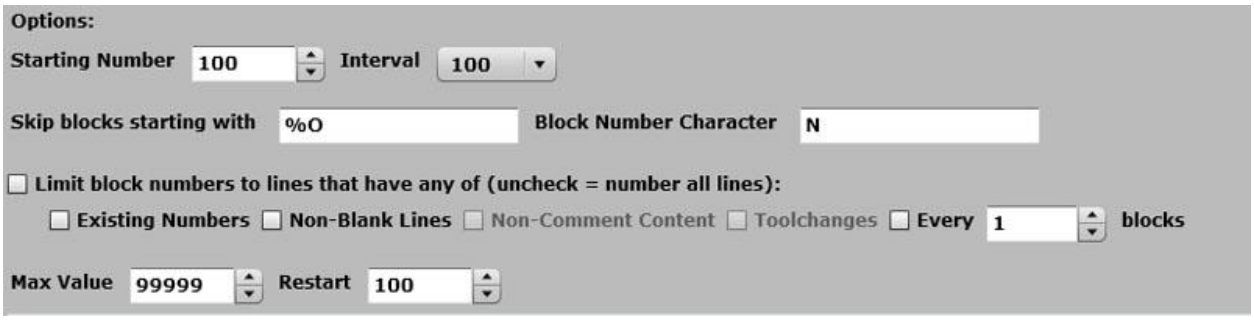
% A do something simple g-code program

O1000

N100 G00 X0 Y0 Z0

%

Sequence numbers will not ever have a decimal point or negative sign--they're always positive integers. Renumbering a program to keep the sequence numbers consistent can be a real pain, so rely on your g-code editor to do that work. Here is the sequence numbering revision panel from G-Wizard Editor:



G-Code Resequencing Command from GWE...

You can tell from the options some of the things you need to keep in mind--we don't sequence number program numbers or tape start/end lines, for example. If a line has a sequence number, the only thing that may come before it is a block skip character. Otherwise, the sequence number should be the first thing on the line.

Spaces in G-Code Programs

Now is a good time to mention that g-code ignores spaces in many cases. They're there simply to make the code more readable by us humans, so use them to your benefit. You can stick spaces between a word and the number that goes with the word (that number is called the "address", see Word Address Format below):

G00 can be G 00

X0 can be X 0

Decide what you like to see in terms of readability and stick with it. Depending on your g-code editor, it may have tools to help you manage this sort of formating automatically. G-Wizard CNC Simulator and Editor certainly does.

Leaving Out Optional G-Code Information to Save Memory

As mentioned, spaces are optional, sequence numbers are option unless they're being used for subprograms or macros (more on these later), and there are potentially other things in the g-code language that are optional. It's worth keeping in mind that if you're on an older controller, one of the constant challenges is the memory capacity of the controller. Stripping out spaces and unused sequence numbers can free up quite a few bytes by making your g-code programs smaller. Don't overlook the opportunity to do something like this if you have to in order to get the job done.

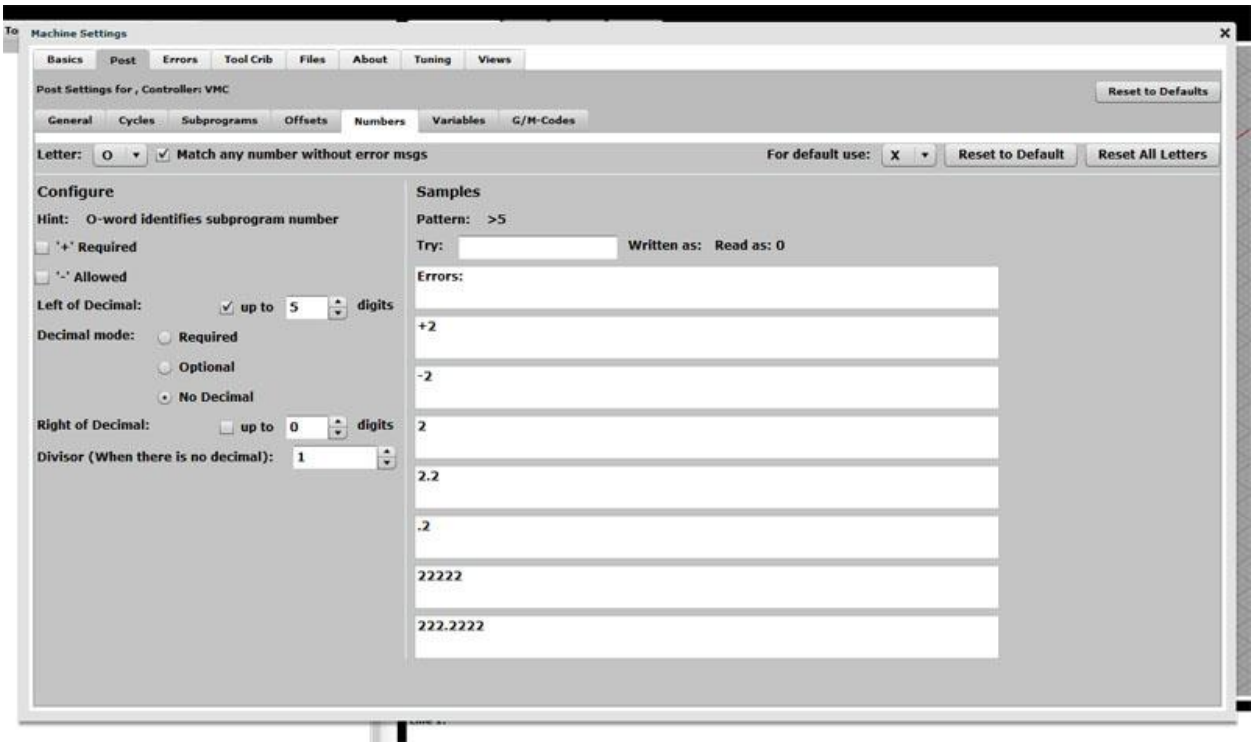
WORD ADDRESS FORMAT

As was mentioned, a LETTER in a g-code program is referred to as a WORD. Associated with each word is an ADDRESS, which is a number that goes with the word. For example, when we're talking coordinates, the address is the coordinate value for the axis: "X0" is the X-word with a "0" address.

Depending on the function of the Word, the address must take a certain format. For example, we mentioned that the addresses (sequence numbers) associated with "N" words have to be positive integers, they'll have no decimal points or signs.

Even the "G" word from which the name "G-Code" comes from has an address that determines which G-Code we're talking about--"G00" is the G-Word with a "00" address and initiates rapid motion modes.

A good CNC Simulator will have the ability to define the Word Address formats to fit whatever your controller expects (they can vary quite a lot from one g-code dialect to the next). G-Wizard Editor uses the following screen for defining Word Address Formats:



Setting up the Word Address format for the "M" word...

To use it, select the letter for the Word in the pulldown menu at top left. The configuration options are in the left column. Samples are provided on the right to show what's valid and what isn't.

BLOCKS DON'T NECESSARILY EXECUTE LEFT TO RIGHT

One of the things I struggled with when first learning g-code was the idea that while blocks mostly execute top to bottom unless you tell them to do something different explicitly via macros and subprograms, blocks don't necessarily execute left to right. In fact, each controller may define its own peculiar ordering, though they do try to be similar. Here, for example, is the order of execution for RS-274 G-Code:

1. Comment
1.1 Comment message
2. Set feed rate mode (G93, G94, G95).
3. Set feed rate (F).
4. Set spindle speed (S).
5. Select tool (T).
6. Change tool (M6).
7. Spindle on or off (M3, M4, M5).
8. Coolant on or off (M7, M8, M9).
9. Enable or disable overrides (M48, M49).
10. Dwell (G4).
11. Set active plane (G17, G18, G19).
12. Set length units (G20, G21).
13. Cutter radius compensation on or off (G40, G41, G42)
14. Cutter length compensation on or off (G43, G44, G49)
14.5 Do scaling G50/G51.
15. Coordinate system selection (G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3).
16. Set path control mode (G61, G61.1, G64)
17. Set distance mode (G90, G91).
18. Set retract mode (G98, G99).
18.6 Do G68/69 coordinate rotation. Scaling before rotation.
18.7 Do G15/16 polar coordinates.
19. Go to reference location (G28, G30) or change coordinate system data (G10) or set axis offsets
(G92, G92.1, G92.2, G94).

20. Perform motion (G0 to G3, G33, G34, G38.x, G73, G76, G80 to G89), as modified (possibly) by G53.

21. Stop (M0, M1, M2, M30, M60).
22. M97, M98, M99

As you can see, the order is all over the map. This was done to try to have operations that depend on one another happen in the preferred order. For example, if you are turning coolant on or off in the same line you're potentially going to perform a cutting motion, you'd want the coolant on before the cutting begins, not afterward.

Modes

As we've mentioned in some of the other articles, G-Code is a modal language. Many times when we execute a particular word, it sets up a mode that carries forward. After we execute "G00", we've put the controller into rapids mode. Any coordinate words will result in rapids motion to the coordinate. These modes are another reason to be aware of the order of execution of the words in a block as many words create modes that have an impact on other words that follow.

Forcing the Order of Execution

If you have any doubt about the order things are happening in, or more importantly, if you care what order the words are executed in, the best answer is to put them into consecutive blocks. This way there is no ambiguity. You could try to remember the arbitrary order for your controller, but that's a lot more error prone!

Don't Cram Too Much Into a Block!

A corollary to the ideas about forcing order of execution is not to try to cram too much into a block. The g-code often won't let you anyway--you can't just slam a whole bunch of moves into the same block, for example. So don't get in too much of a hurry. Keep things together that it makes sense to have together and let things where it doesn't make sense spread into their own blocks to make the code more clear.

WORD CONFLICTS AND CODE GROUPS

Since words are modal, the possibility of conflicts exists. For example, if you entered "G00" (rapids linear motion) and "G01" (feedrate linear motion) in the same block, there is a conflict because the controller can't do both. Generally, it will choose the last one but sometimes it will create an error.

Fanuc and others try to keep conflicts straight using what are called "Code Groups". If two words belong to the same group, then they conflict with each other and should not be used in the same block.

COMMENTS

Speaking of making sense and being clear, don't forget to add comments to your code in various places. Anything you place in parenthesis is a comment that is ignored when executing the g-code program. For example:

% A do something simple g-code program

O1000

N100 G00 X0 Y0 Z0 (Rapid to Part Zero)

%

The comment tells why we're moving to that particular location. There's no need to over-comment the code--things that are obvious shouldn't need comments. Try commenting anything that was hard to figure out when you wrote the g-code. The comments can serve as a reminder for what you learned and what you were trying to accomplish. If you're trying to read some g-code and scratching your head, add some comments when you figure it out. The next guy to read that code will thank you.

Comments as Headers Delineating Sections

Comments are a good way to delineate different sections of the g-code. Put one at the top that has all the salient information about your program that the operator and future programmers will find convenient to have:

%

(----------)

(A do something simple g-code program ----------)

(Written by Bob Warfield, July 18, 2011 ----------)

(----------)

O1000

N100 G00 X0 Y0 Z0 (Rapid to Part Zero)

%

A really complete program header ought to include things like:

- Program file name.

- Program author

- Version date

- Modification history--a line for each version telling what changed, who made the changes, and the date.

- Machine and control the program is intended for

- Units: inches or mm

- Setup information: What stock material is expected, what size, etc.?

- Work Coordinates: What's expected by way of work coordinates? Where is Part Zero?

- Customer info: Job shops will want information about what customer and job # the program is part of.

It's a good idea to have a header for each subprogram as well, though there is no need to repeat all the same information that's already in the main program's header.

CAM Programs and Other Software Often Create Special Comments

CAM Programs and other software will often use comments as a good place to store additional information that there is no other way to encode into the g-code. Here is a typical example from the HSMWorks CAM program:

```
%
O1000 (TANKCONSTSCALLOP)
(T1 D=0.4724 CR=0. - ZMIN=-0.77 - FLAT END MILL)
```

It's telling us the name of the CAM file was "TANKCONSTANTSCALLOP". Immediately below is a tool definition. This program uses exactly one tool "T1" with the following characteristics:

- Tool Diameter = 0.4724

- Corner Radius = 0

- Z Minimum = -0.77

- Tool Type = Flat End Mill

Here's an example from OneCNC:

```
%
O0000
N10 (MACH3 MILL - )
N20 (AUTHOR - {BOB WARFIELD})
N30 (GROUP - SPOT DRILL AND ENGRAVE PANEL)
N40 (SYSTEM- ONECNC-XR2 MILL ADVANTAGE - VERSION 8.12)
N50 (NOTES - NONE)
```

They went to a format that has more information for human operators.

Keep an eye out for these if you're editing g-code generated by a CAM program. They may give you some useful information and you may not want to delete these comment if they're something some other software may want to access for some reason.

CONCLUSION

You now have a solid foundation for using CNC machines as a manual machine substitute, and you've seen the basics of how g-code programs are put together from g-code blocks. That's a good starting point. From here on we'll be adding bricks to that foundation around special topics that will allow you to create more and more powerful g-code programs.

EXERCISES

1. Get your CNC Controller's manual and go through G-Wizard Editor's Word Address Formats to verify that GWE is configured correctly for your controller.

2. Put together a simple header format that you plan to use for your g-code programming. Use it to create a simple do-nothing program that can serve as the initial template when you create a new g-code file.

3. If you use a CAM program, investigate what its options are for headers in the g-code file it posts.

LINEAR MOTION WITH G00 AND G01

LINEAR MOTION IS STRAIGHT LINE MOTION

G-Code is all about motion, so let's take a look at the most common kind of motion found in part programs: LINEAR MOTION. Linear motion is simply motion in a STRAIGHT LINE. Motion is another one of those things in g-code that is modal. You tell the controller what kind of motion you'd like with a g-code and it remembers to always make that kind of motion until you tell it to change using another g-code. We've covered a lot of the basics in the chapter on MDI, but let's just buzz through a review and slightly deeper coverage of that material. It won't take long and it will give me a chance to talk about the commands from the perspective of part programs rather than manual machining.

G00 FOR FAST POSITIONING, G01 FOR SLOWER CUTTING MOTION

There are two codes intended to set up linear motion modes:

- G00 specifies *rapids* motion.

- G01 specifies *feed* motion.

Rapids motion tells your machine to move at its fastest possible speed. G00 is used to position the cutter near where you want to start cutting, but we never enter a cut with G00. Doing so by mistake ensures a broken cutter or worse as rapids motion is way too fast for any kind of cutting. Most controllers start up with G00 active when you first turn on the machine. That's because the part program has to manuever the cutter into position near the cut before you can begin removing material. Once the cutter is ready and you want to make cutting motions, you would typically use G01 to specify feed motion.

Older Machines May "Dog Leg" G00

Something to keep in mind if you're using an older controller is that it may "dog leg" G00. That means that rather than moving all the axes to create straight line motion to the target, it will move one axis at a time. Depending on what might be in the way (clamps, fixtures, vises, or raised portions of your workpiece), this can be very important to consider when programming G00 moves.

SPECIFY CUTTING SPEED WITH THE "F" WORD AND SPINDLE RPM WITH THE "S" WORD

No, silly, not that F-word!

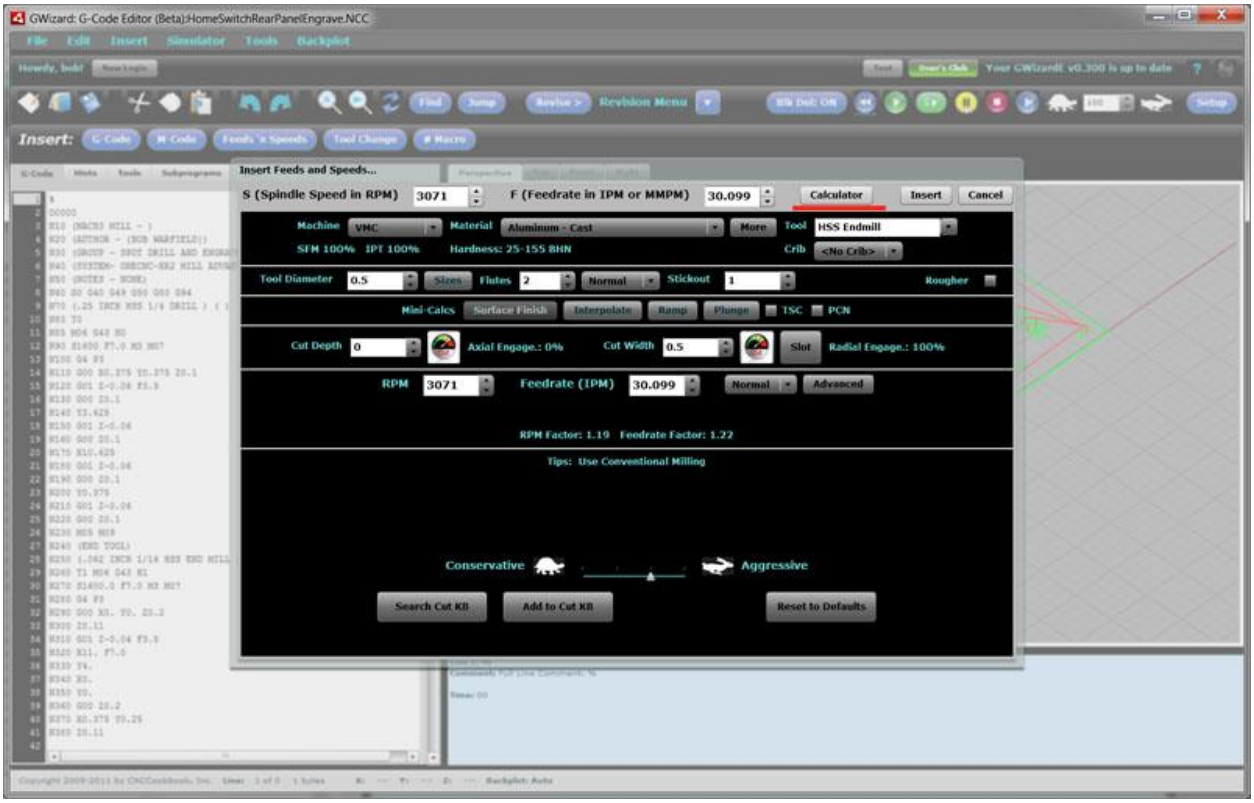
"F" as in "Feedrate". The speed at which your cutter moves while the G01 mode is active is called its feedrate. This rate is highly dependent on the type of material you are cutting, the type of cutter you're using, the spindle speed and a whole host of other factors. The feedrate word specifies feedrate in inches per minute for Imperial controllers and in millimeters per minute if your controller is set up for metric. It's easiest to determine the feedrate using a feeds and speeds calculator like our own G-Wizard. We have a free course on determining the best feeds and speeds for your machine, so be sure to check it out. The G-Wizard Calculator is also available inexpensively with a 30-day trial so you can play with it.

The other important parameter when setting up a cut is the spindle speed, which is determined by the "S" word. For now, consider that the default "S" address is rpm. There are specialized modes that let you specify spindle speed in other ways.

I like to set up the F and S words before issuing the G01. The words are modal just like G00 and G01, so you set them and then don't need to change them until you want a different speed or feed. Switching back and forth between G00 and G01 or the other kinds of motion doesn't affect the F and S settings either.

G-WIZARD EDITOR IS INTEGRATED WITH THE G-WIZARD FEEDS AND SPEEDS CALCULATOR

G-Wizard Editor includes Wizards to help insert all the different g-codes in the right formats. To add feeds and speeds type "Ctrl+P" (P for sPeeds and feeds). All our keyboard shortcuts have some kind of memory aid behind the choice of which keys to press, and they're listed on the keyboard page. Once typed, you get a popup that lets you enter a value for feedrate and spindle rpm at the top. If you press the "Calculator" button (underlined in red below), you get to access the G-Wizard Feeds and Speeds Calculator:



Integrating our feeds and speeds calculator with GWE...

Note that you have to be a registered G-Wizard Calculator user with a valid trial or subscription. Use the calculator as desired to calculate your feeds and speeds and then press the "Insert" button in the top row to add the "F" and "S" words with their addresses to your program.

SPECIFYING LINEAR MOTION WITH X, Y, AND Z

Note that simply specifying G00 or G01 does not cause any motion to happen--they merely tell the controller what type of motion is expected when you finally tell it where to move to. For actual motion you need to specify a destination using the X, Y, and Z words. We've talked about how the coordinate system works previously, so you should be quite familiar with XYZ moves. As a reminder, to move to the part zero, we might issue a command like this:

G00 X0Y0Z0

We can also do it this way:

G00 (Or use G01 if you want to go slower)

X0Y0Z0

When we specify multiple coordinates on a line, we get what's called INTERPOLATED MOTION or an INTERPOLATED MOVE. That fancy word just means more than one axis of the machine is moving at the same time. In fact, the controller will move them all at exactly the right speed relative to one another so that the cutter follows a straight line to the destination and moves at the feedrate. Imagine trying to do a coordinated 3 axis move at an exact feedrate on a manual machine. I know guys that can shift a stick shift on a car with their knee while both hands are busy (not recommended!), but I have yet to see a machinist do a coordinated 3 axis move with manual handwheels!

If we specify the same destination, but spread the coordinates over multiple lines, each line is a separate move:

G00

X0Y0 (Move to X0Y0 in one move, keeping Z constant)

Z0 (Move to Z0 in one move, keeping X and Y constant)

Remember, G00 and G01 are modal, so we only have to specify them when we want to change modes.

Careful With Z!

The concept of interpolated moves raises an interesting issue for the Z axis (or whichever axis controls your depth of cut). It's often a good idea to move the depth-of-cut-axis on its own, rather than as coordinated motion with other axes. Doing so just makes it easier to "eyeball" whether you're going to have a problem (collision is the

technical term) as the cutter gets close to the workpiece and fixturing. It's really hard for the human eye to judge motion in multiple axes, particularly if you have to move a long ways in X and Y and a relatively shorter distance in Z. By first moving in X and Y and then moving in Z as shown in the example above, it's much easier to judge whether an accidental collision is about to take place. You're also much less likely to hit some random object sticking up, like a clamp, if you keep the cutter high until you're directly over where you want to start cutting.

The downside to this approach is it might be slightly slower than the coordinated move. If you're trying to wring every last second out of a job, start out making 2 moves and then when you're sure you've got the job running smoothly, update the program on a future run.

Entering a Cut

While you will often see programs and machinists that take the cutter straight into the material to begin a cut, this is not the best approach for cutter life and surface finish. The Feeds and Speeds course talks more about this in the section on toolpaths, but ideal you'd like to enter with some sort of an arc move that gradually builds the cutting forces instead of hammering straight in with a plunge cut of some kind. To do that, we'll need to understand arc moves, the subject of our next chapter.

EXERCISES

1. If you haven't already, sign up for the G-Wizard Calculator's 30-day free trial and start going through the free Feeds and Speeds Course to learn more about feedrates.

2. Try some "etch-a-sketch" programming. Draw out a figure by hand on graph paper so you can see the coordinates easily. Now using the G-Wizard Editor, start trying to enter the correct G00 and G01 moves to create a cut that resembles that figure. Use G00 to position close to the starting point, set the Feed and Speed up, and then start making G01 cuts

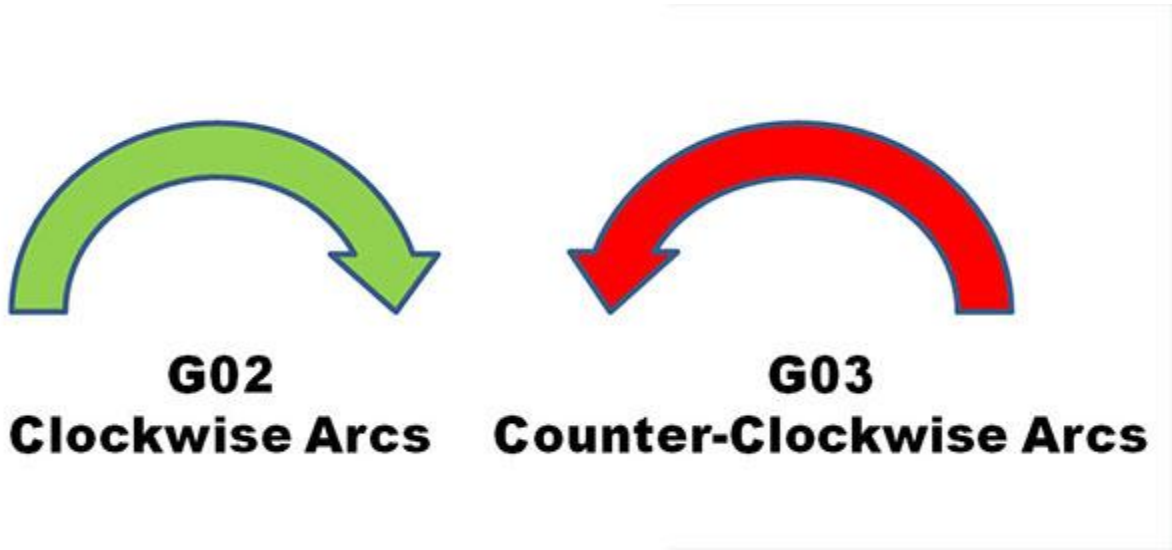
CIRCULAR ARCS WITH G02 AND G03

CIRCULAR INTERPOLATION IS MOTION ALONG A CIRCULAR ARC

Having just finished discussing linear interpolation, or motion in a straight line, we next come to circular interpolation, which is motion along a circular arc. Other than the fairly exotic ability to follow a "NURBS" path, most g-code controllers only support two kinds of motion: linear and circular. Circular interpolation is quite a bit more demanding on your machine as two axes have to be precisely coordinated. Drawing a complete circle involves not just coordinated motion but reversal of direction at each of the 4 quadrant points. These would be the points corresponding to 0, 90, 180, and 270 degrees. If the machine has any backlash at all, it will be obvious at these reversals because there will be a glitch in the cut there.

CIRCULAR MOTION IS A MODE INITIATED VIA G02 OR G03

Like linear motion (initiated by G00 and G01), circular motion is a mode initiated via G02 or G03. G02 establishes a mode for clockwise circular arcs. G03 establishes a mode for counter-clockwise circular arcs.

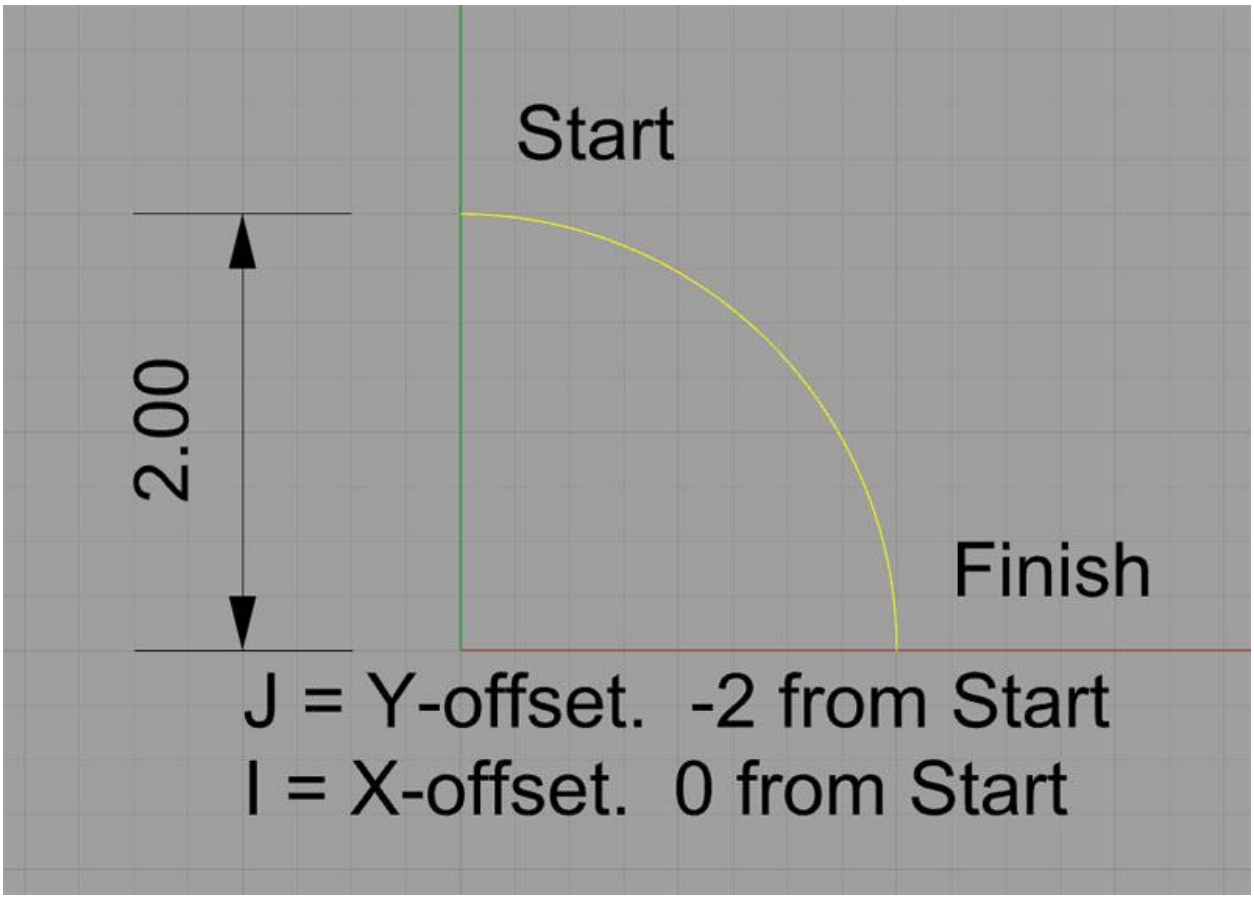


Defining An Arc For the CNC Controller

Once either the G02 or G03 mode is established, arcs are defined in G-Code by identifying their 2 endpoints and the center which must be equi-distant from each endpoint or an alarm will occur. The endpoints are easy. The current control point, or location when the block is begun establishes one endpoint. The other may be established by XYZ coordinates. The center is a bit more complex.

Defining the Center Via IJK Relative Offsets

The center is most commonly identified by using I, J, or K to establish *relative offsets* from the starting point of the arc to the center. Here is a typical clockwise arc:



Defining an arc's center with IJK...

This arc starts at X0Y2 and finishes at X2Y0. It's center is at X0Y0. We could specify it in g-code like this:

G02 (Set up the clockwise arc mode)

X2Y0 I0J-2.0

The I and the J specify relative coordinates from the start point to the center. In other words, if we add the I value to the starting point's X, and the J value to the starting point's Y, we get the X and Y for the center.

Defining the Center Via the Radius Using "R"

We can also define the center just by specifying the radius of the circle. In this case, our circle has a radius of 2, so the g-code might be simply:

G02

X2Y0 R2

Many of you will be deciding right here and now that since R is easier to understand and shorter to write, you're just going to use R and forget about IJK. But, the CNC teachers in the world will suggest that you should prefer IJK. Their argument is that when you use IJK, you get a double check that your arc is correct.

Why?

Because the controller gets to compute an actual set of coordinates for the center via IJK. Once it has the center's coordinates, it can check that it is equa-distant from both end points. The check of each of those two distances is the double check. In the case of the "R" format, the controller has no such double check. It has to chose a center that guarantees equal distance.

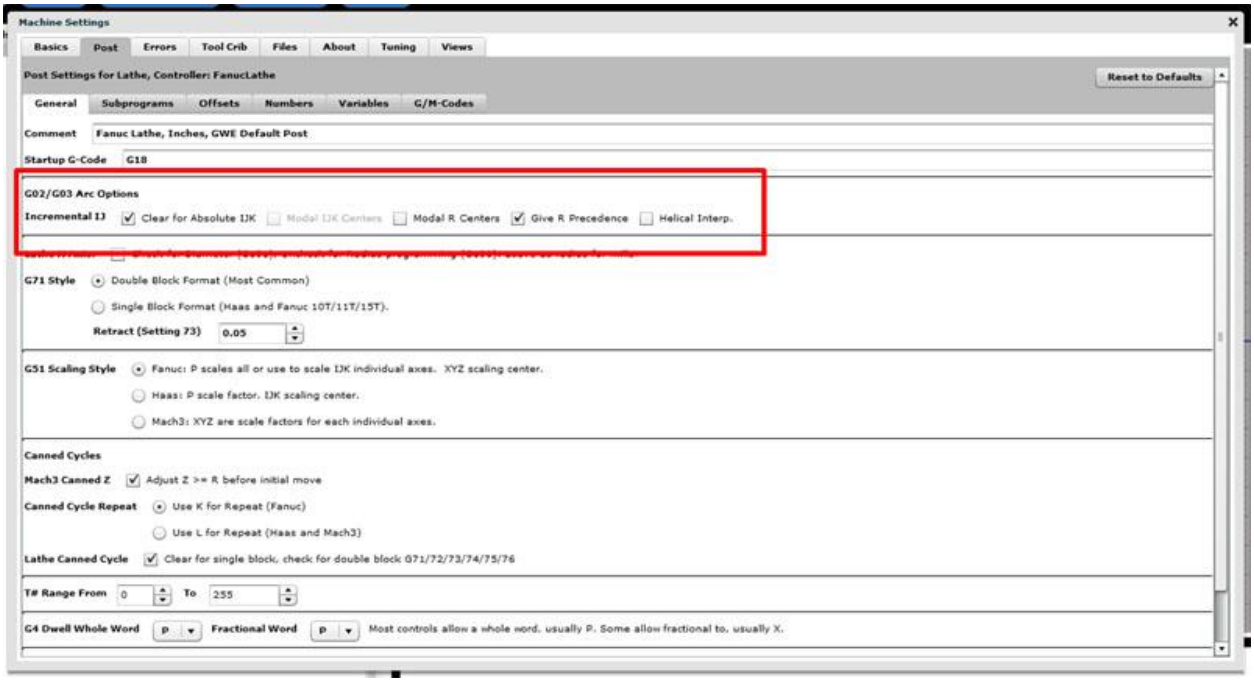
Personally, I don't know if I agree with the CNC instructors that this is providing any extra checking or not. I say go with whichever approach makes sense for your particular situation, but you should definitely be familiar and comfortable with both. You're going to need to be comfortable with relative coordinates anyway, as they're darned handy. May as well get comfortable now.

It's kind of like being told you should only use the 4-jaw chuck on a lathe when you first start out so you'll get very comfortable dialing it in. It's a good skill to be good at as a machinist!

When IJK Are Not Incremental and What About Having Both IJK And R? Plus, Other Modal Shenanigans and Arc Variations

This is another one of those places where lots of obscure things happen and you need to know what your controller will do without assuming anything. In general, the rule is supposed to be that if you have both IJK and R in the same block, R takes precedence and IJK is ignored. But there are controllers that don't work exactly that way, so be sure you know what's going on.

G-Wizard Editor let's you specify several parameters in its Post that determine how arcs work. Here is a screen shot of the setup options:



Arc Options for G-Code Simulation

Let's go over these options:

- **Incremental vs Absolute IJK**: We've discussed IJK as offering coordinates relative to the starting point for the center. Add the I to X, J to Y, and K to Z of the starting point and you get the center. Many controls also have the option for IJK to be the absolute coordinates of the center.

- **Modal IJK Centers**: When IJK are absolute center coordinates, some controllers will remember the last center defined, hence IJK is modal in that case. When using a control set up like this, you can just keep issuing XYZ commands for arcs without having to define a new center each time. It's not clear you'll save much though--how often do you want to do a bunch of arcs with the same center?

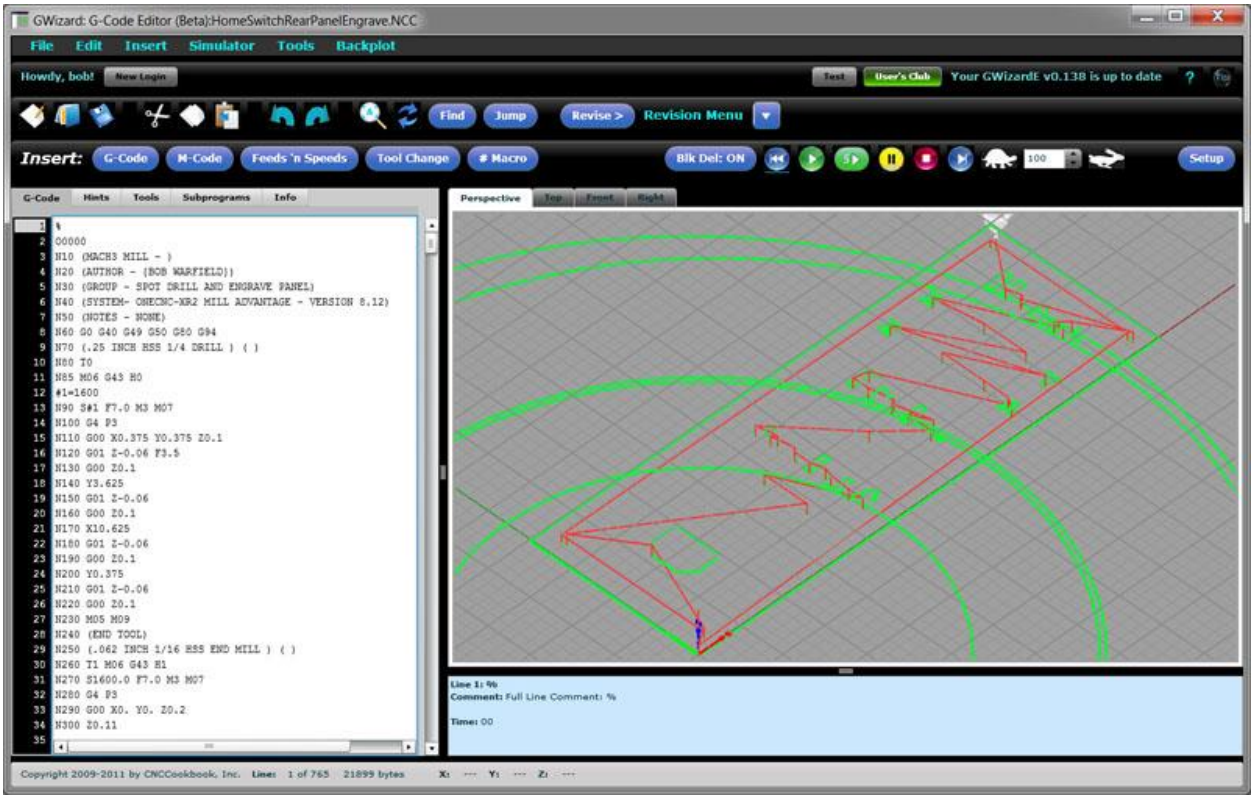
- **Modal R Centers**: Another variation on the modal center idea is to allow the radius defined by "R" to be modal. Whatever the last R used was, the controller remembers and uses that value again if no R is given. This seems more useful than modal IJK. For example, a pocket might have arcs for the corners that are all the same radius.

- **Give R Precedence**: As mentioned, most controllers will use "R" when both "R" and "IJK" are given in the same block. But this option allows you to change that precedence to IJK if your controller works that way instead.

- **Helical Interp.**: This option governs whether your controller allows helical interpolation.

THE MOST COMMON PROBLEM CONFIGURING A CAM POST OR CNC SIMULATOR: ABSOLUTE VS RELATIVE IJK

We've all had the experience (or will soon if you're a beginner) of looking at a backplot (or worse, seeing it in the actual tool motion which is pretty scary) and seeing the giant almost complete circles and no sign of the familiar part motions we expected to see. Here is a typical example:



If you see that sort of thing, like your g-code has been invaded by crop circles, the first thing to check is absolute versus relative IJK for arcs. The setting has to match between what the CAM produces and what the controller or simulator expects.

FRACTIONS OF A CIRCLE, QUADRANTS, AND CONTROLLERS

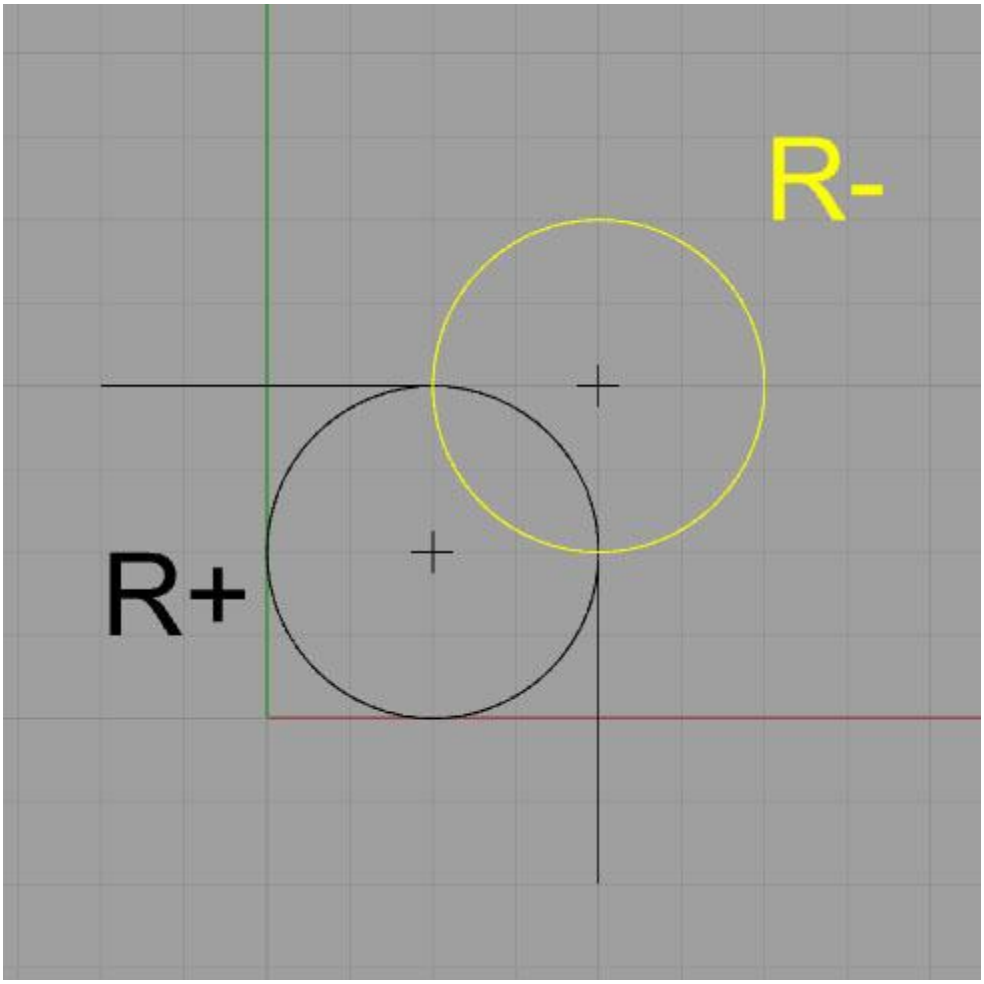
The first thing about an arc is it isn't possible to specify more than a 360 degree arc. There are some exceptions to this on some controllers for Helical Interpolation (see below), just because it can be useful for helixes. When a full circle is desired, set the start and end points equal to one another:

G01 X3.25 Y2.0

G02 X3.25 Y2.0 I-1.25 J0

Interestingly, you can't specify a full circle with the "R" notation. This is because there are an infinite number of circles that start and end at the same point of a particular radius, so the controller has no idea what the correct circle might be.

There is more funny business still with "R" and larger arcs. For example, an arc may still be of a particular radius and clockwise (or counter-clockwise direction), but the center is ambiguous if you travel more than 90 degrees. For example:



If R is negative, it takes the longer path (in yellow). Positive gets the shorter path.

Given the two choices shown, the controller chooses the path based on the sign of the radius. Negative forces the longer arc, positive the shorter. The negative sign forces the controller to seek a viable arc of more than 180 degrees.

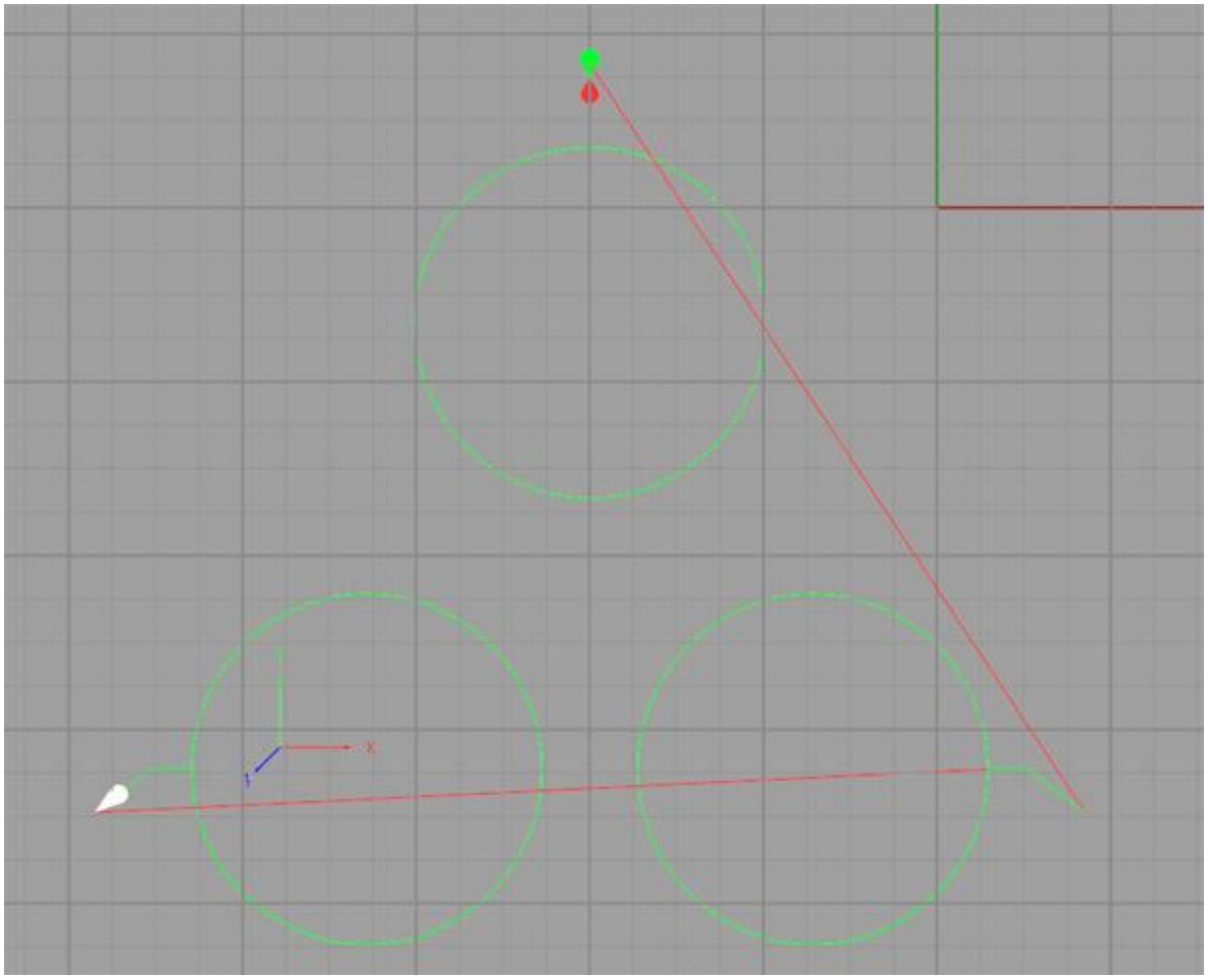
Some controllers are touchier still and will not program an arc that crosses a quadrant line. Hence, the largest angle an arc can follow is 90 degrees, and that angle must not cross 0, 90, 180, or 270 degrees. For angles of 90 degrees that cross a quadrant line, they must be broken into two pieces, with the join between the pieces being right on the quadrant line.

Full Circles with No XYZ

Full circles come about when the start and endpoints are identical and the center is specified via IJK (remember, R leads to an infinite number of circles). Given that you want the start and endpoint to be the same, you may not need to bother even specifying the end point with XYZ. Some controllers may require it, but most do not. Here's a simple g-code program that produces 3 circles in this way:

```
N45 G0 X-2. Y.75
N46 G1 Z-.5 F10.
N47 Y.5 F30. S2000
N48 G2 J-1.1
N49 G1 Y.75
N50 Z.2
N51 G0 X.75 Y-3.4
N52 G1 Z-.5 F10.
N53 X.5 F30.
N54 G2 I-1.1
N55 X.75
N56 Z.2
N57 G0 X-4.75 Y-3.4
N58 G1 Z-.5 F10.
N59 X-4.5 F30.
N60 G2 I1.1
N61 G1 X-4.75
N62 Z.2
```

And here's what the backplot looks like:

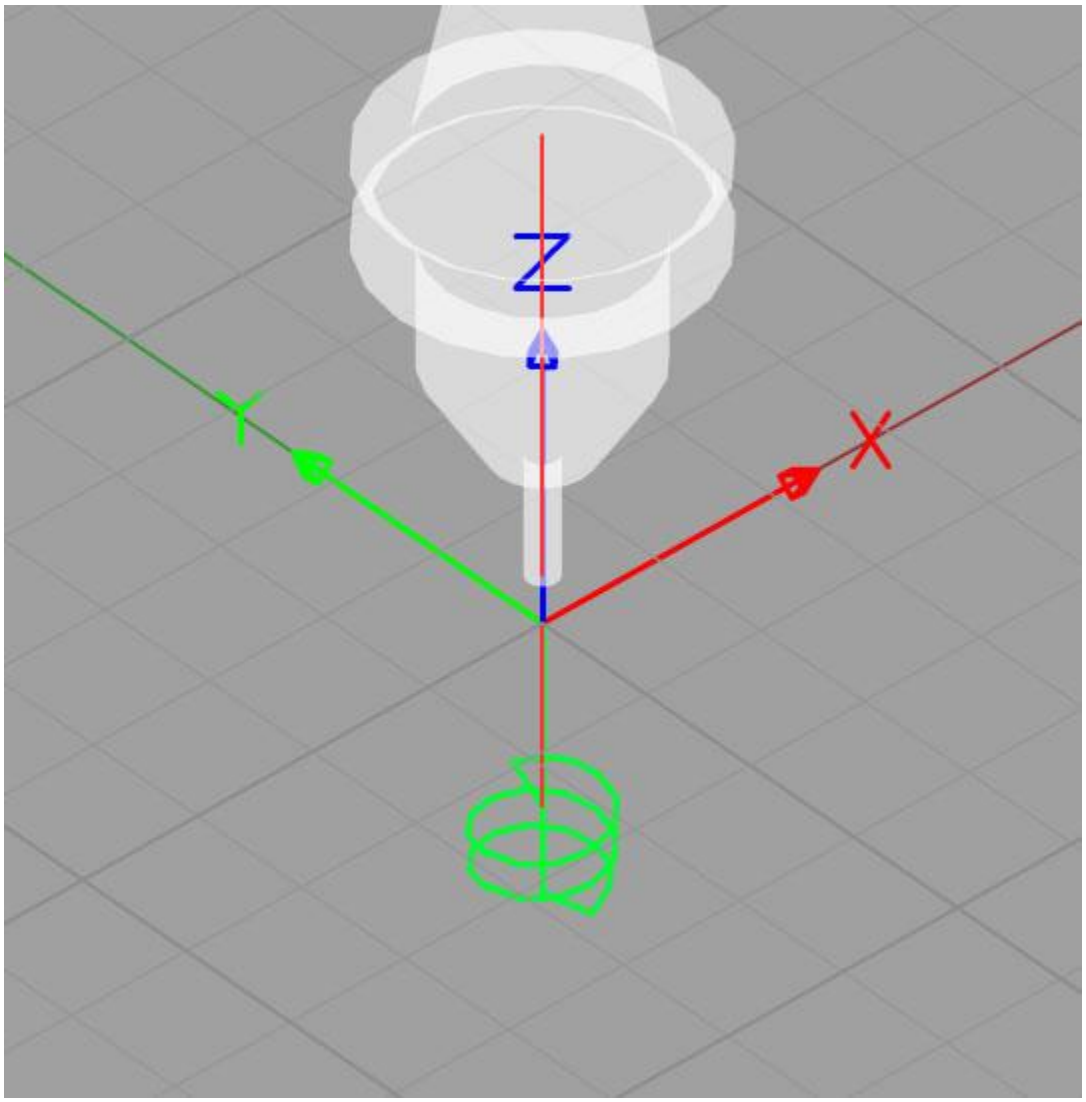


TIP TO MAKE ARC PROGRAMMING SIMPLER: START WITH SEGMENTS

When I'm laying out a toolpath, I prefer to leave the arcs until last. In place of each arc, I simply put a line segment whose endpoints correspond to the arc's endpoints. This makes it easy to get the rough sketch of the toolpath together quickly, and it often seems to make it easier to then go back and convert the lines to arcs once the basic structure is already in place.

HELICAL INTERPOLATION

A helix is an arc that continuously moves in a third dimension, like a screw thread. With helical interpolation, we specify such an arc with G02/G03 in order to move the cutter along a helix. This can be done for thread milling, interpolating a hole, or a variety of other purposes. Here is a backplot from a 1/4" NPT thread mill program:



Helix for thread milling...

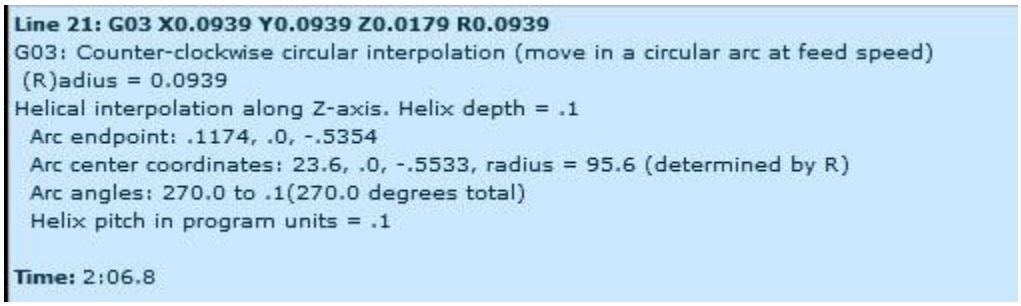
Here is a sample of the code from the thread milling program:

```
G01 G91 Z-0.6533 F100.
G01 G42 D08 X0.0235 Y-0.0939 F10.
G03 X0.0939 Y0.0939 Z0.0179 R0.0939
G03 X-0.1179 Y0.1179 Z0.0179 R0.1179
```

```
G03 X-0.1185 Y-0.1185 Z0.0179 R0.1185
G03 X0.1191 Y-0.1191 Z0.0179 R0.1191 F16.
G03 X0.1196 Y0.1196 Z0.0179 R0.1196
G03 X-0.1202 Y0.1202 Z0.0179 R0.1202 F26.
G03 X-0.1207 Y-0.1207 Z0.0179 R0.1207
G03 X0.1213 Y-0.1213 Z0.0179 R0.1213
G03 X0.1218 Y0.1218 Z0.0179 R0.1218
G03 X-0.0975 Y0.0975 Z0.0179 R0.0975
```

This is "R" (radius) format for the arcs, and note there is a Z coordinate to specify a depth change for the end point of each arc. This code uses relative motion (G91), so each "Z0.0179" moves the cutter 0.0179" deeper.

G-Wizard Editor provides some really useful information to help out with understanding helical interpolation. Here is the Hint from the third line (first arc move):



Note the thread pitch here is calculated as 0.1"

GWE will measure and tell you the helix pitch, which in this case is 0.100". That can be useful for identifying what sort of thread is being milled. We can also see that this particular arc runs from 270 degrees to a scosh more than zero (0.1 degrees).

We'll revisit thread milling in much more detail in a later chapter devoted entirely to the subject. For now, we just wanted you to be familiar with the idea that you can make helixes as well as flat two dimensional arcs.

MAKING TOOLPATHS YOUR MACHINE WILL BE HAPPIER WITH

Whenever the cutter changes direction, it adds a certain amount of stress. The cutter will bite into the material either more or less than it had been, depending on whether the directions changes towards the workpiece (or uncut material) or away from it. Your machine will be much happier if you program an arc rather than an abrupt straightline change of direction. Even an arc with a very small radius will allow the controller to avoid changing direction instantly, which can leave a mark in the finish in the best case and cause chatter or other problems in the worst case. For slight changes of direction, it may not be worth it. But the more abrupt the change, with 90 degrees being very abrupt, the greater the likelihood you should use an arc to ease through the turn.

Arcs are also a useful way to enter the cut, rather than having the cutter barge straight in. For information on entering the cut with an arc, see the toolpath page from the Milling Feeds and Speeds Course.

EXERCISES

1. Dig out your CNC controller manual and go through the arc settings to set up GWE to match your control's way of operating.

2. Do some etch-a-sketch experimentation with GWE. Create some toolpaths that include arcs until you're comfortable creating them.

RUNNING THE GWE G-CODE SIMULATOR TO VIEW AND DEBUG YOUR G-CODE PROGRAMS

HOW DO I TEST MY G-CODE PROGRAMS WITHOUT CRASHING MY MACHINE?

By now you've come along quite a ways. You're able to control your CNC using MDI as though it were a manual machine with DRO's and Power Feeds on every axis. You understand the basic G-Codes for motion and how to construct a line of g-code with sequence numbers and all the rest. You're ready to start creating more complex programs that get beyond what you might try to do on the fly with MDI. Perhaps you have a CAM program that's generating g-code and you're itching to try it out.

How do you know the program isn't going to crash the cutter into your workpiece, or worse an expensive vise or your machine's table?

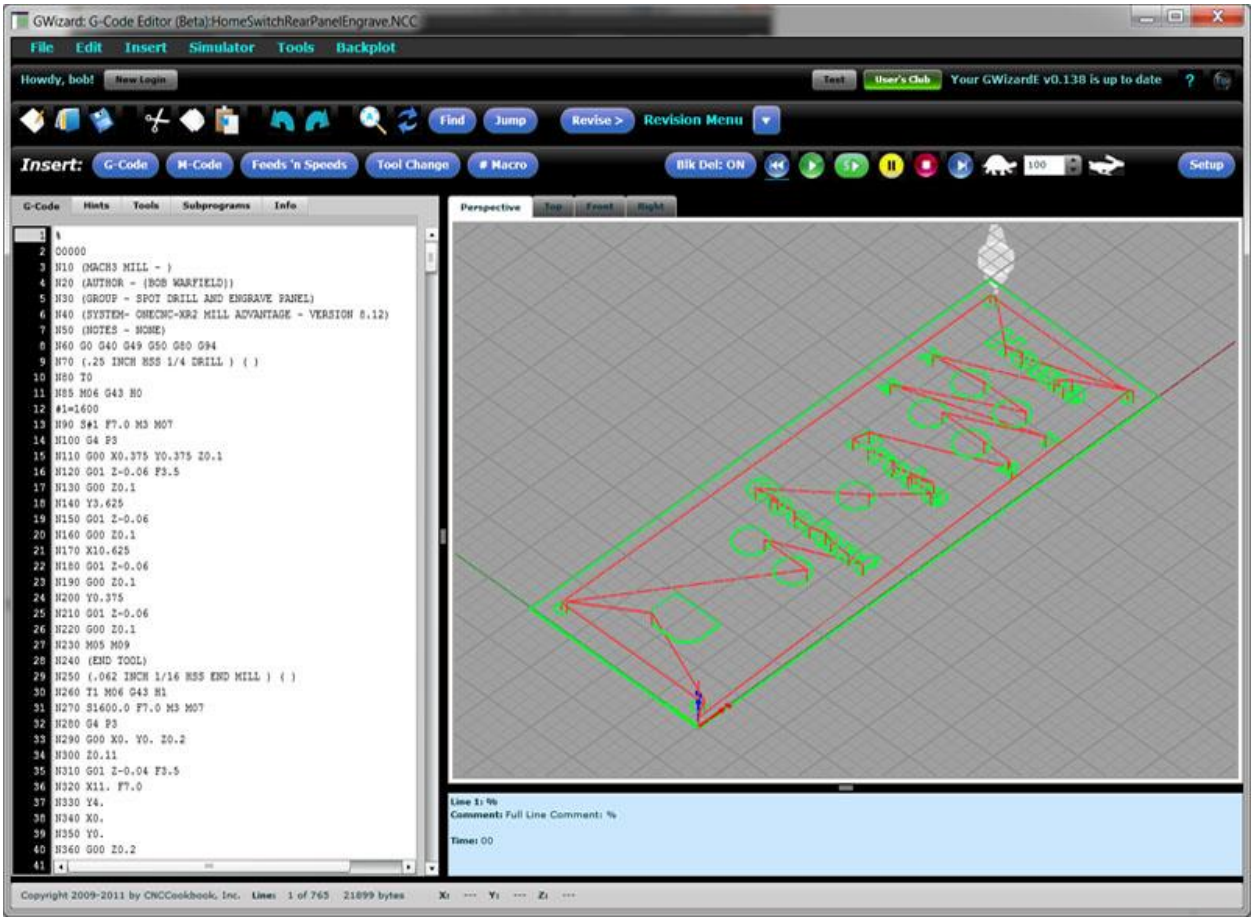
Your first order of priority in solving this problem is to make sure you get part zero set up properly, and that you're able to run the machine "cutting air" with one hand on the feedrate override. That means you will intentionally make part zero sit high above the part so that unless there is a severe error in your program, the cutter will never get close to anything but air. You'll use the feedrate override to run the machine very slowly through the program while it is cutting the air.

Being able to cut air is a great diagnostic tool to have, and until you're absolutely sure of your skills in setting up the program's part zero and dealing with different tool lengths, you'll want to be sure to carefully check each program you run by cutting air.

CUTTING AIR IS SLOW: IS THERE A BETTER WAY TO DIAGNOSE MY PROGRAMS?

Cutting Air can be extremely slow, and it uses valuable machine time that might otherwise be employed doing something productive like making chips. Once you're facile cutting air, and you know how to set up your machine with the proper part zero and deal with tool length offsets, you're going to get tired of cutting air and be wishing for a faster way to check your programs. The answer to this problem is that you'll want to use a G-Code Simulator, such as CNCCookbook's own G-Wizard Editor.

A good simulator will show you what's called a Backplot of your program. Here is a typical backplot of the g-code programs we have available for download and experimentation:



A backplot of a program to engrave a CNC connector panel...

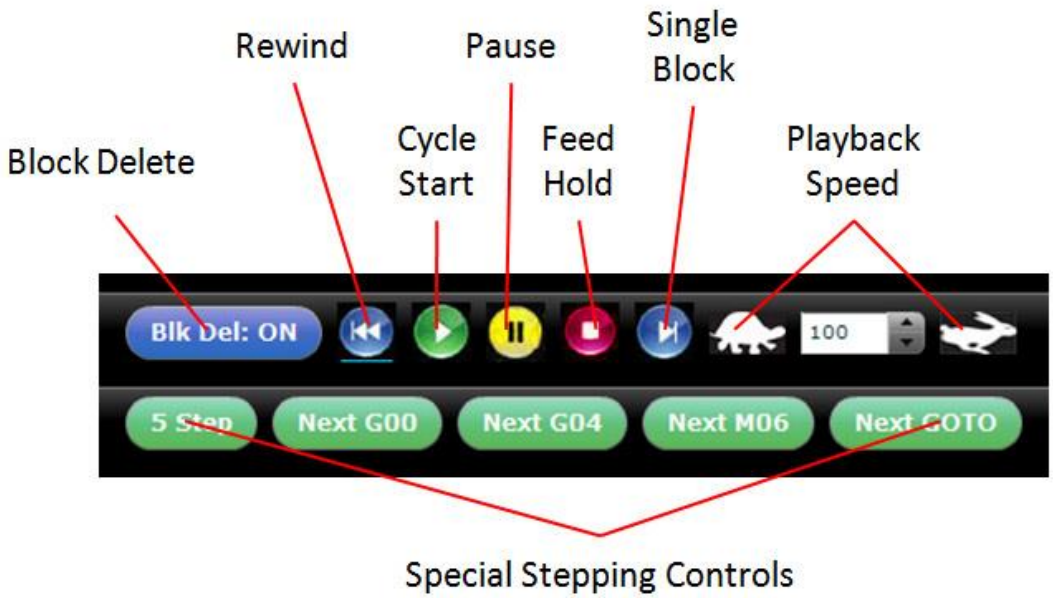
The red and green lines are showing you the motion of the tip of the cutter on this mill program. Red lines are for rapids: G0 movement. Green lines are for feed motion: G01 for straightlines and G02/03 for arcs. It's pretty easy to tell from the backplot whether there is some errant motion that will slam the tool tip down too far, for example.

DOESN'T MY CAM PROGRAM ALREADY HAVE A BACKPLOT OR SIMULATOR FOR THIS PURPOSE?

The qualified answer is, "Maybe." The issue is how the CAM program's backplot is generated. If your CAM program includes a true CNC g-code simulator, then yes, it does have a simulated backplot and by all means use it. The thing is, most CAM programs don't. They just plot the same geometry information that was used to create the g-code output by the postprocessor. This allows for subtle bugs to creep in that are not detectable in the CAM backplot. Because of that, a lot of experienced machinists insist on a separate simulated backplot as a sanity check for their g-code before they'll run it. It doesn't cost very much or take very long to have this peace of mind, so it's something you should consider.

SINGLE STEPPING TO UNDERSTAND G-CODE BETTER

A good simulator can do a lot more than just display a backplot; it can run through the program step by step and show you a great deal of information about what's going on. With GWE, a set of buttons similar to what is found on your CNC is used to control the simulation:



Simulator Controls...

To start a simulation, press the "Rewind" followed by "Cycle Start". You can control playback speed as a percentage of maximum with the Playback Speed controls. The Tortoise makes it run slow, the Hare makes it run fast and you can change the percentage as well. If you want to step through block by block, turn on the "Single

Block" button. It will turn red to signify it is active. To exit the simulator and show only the finished Backplot, just press "Feed Hold".

Using these controls you can go through your program block by block. The "Special Stepping Controls" give you the ability to move ahead to points of interest more quickly:

- "5-Step" causes 5 blocks at a time to execute in Single Block mode instead of just one at a time. You can change the number of blocks this button will execute via the Simulator Options menu.

- "Next G00" moves the simulation ahead to the next G00. Often a g-code program will alternate between cutting moves and G00 moves to position for the next cut, so this is a convenient way to move forward.

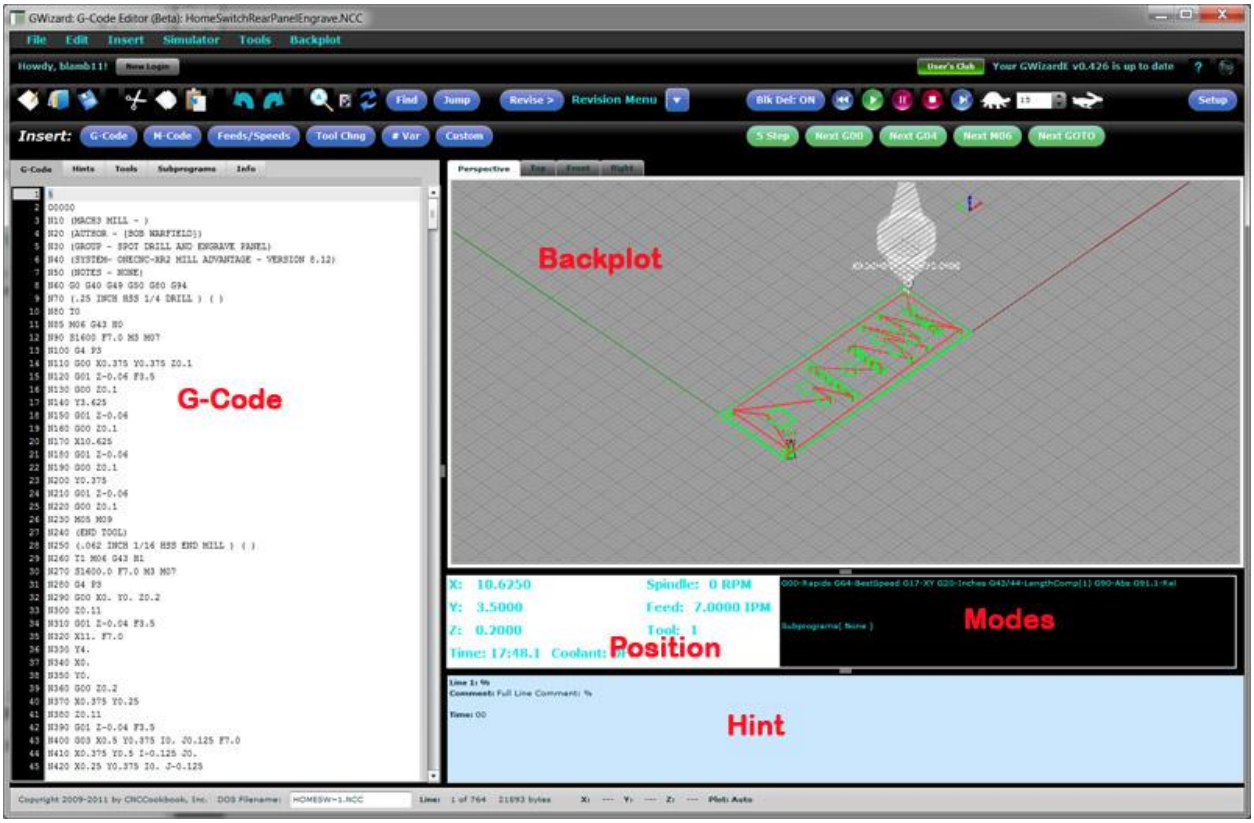
- "Next G04" moves the simulator ahead to the next Dwell (G04). Why is that helpful? Dwells can be inserted almost anywhere in a program without causing a problem, so insert a Dwell in a place you want to be able to go to easily while debugging your program and use this button to get there quickly.

- "Next M06" moves the simulator ahead to the next Toolchange.

- "Next GOTO" moves the simulator ahead to the next place where the program jumps to a subroutine or block number. It's handy if you're trying to debug macros.

SIMULATOR SCREEN AREAS

When the Simulator is running, GWE displays quite a lot of additional information in a variety of screen areas:



Simulator screen areas labeled in red...

Each screen area is called out in the illustration above in red. Here is the function of each:

G-Code

This area shows you the g-code. The line being executed is highlighted. The illustration shows the very top line highlighted. If you click on this pane, GWE will offer to stop the simulation so you can edit the code.

Backplot

This shows you the backplot of the moves made up to the currently executing line.

Position

Position is like the position display on your machine controller. It's showing you the X, Y, and Z coordinates, the Spindle RPM, the Feedrate, the current Tool, the Coolant status, and the time the program has run.

Modes

Modes tells you all the modal states of the controller. For example, is some work offset active? It also tells you the last few#variables that changed and what the values were that they changed to.

Hint

The Hint tells you in plain English what the g-code for the current line is expected to do. It also gives you a lot of other information that may not be obvious just looking at the program such as where an arc's center is located.

CONCLUSION

Using a simulator like G-Wizard Editor makes it much easier for you to tell by looking at the backplot whether the program is apt to make sudden moves outside the expected work envelope. It also helps you go through the program step-by-step and see what it's doing.

EXERCISES

1. Experiment on your machine cutting air to make sure you're familiar with how to set that up and use it as a diagnostic tool.

2. Run several CAM generated posts through a G-Code Simulator like G-Wizard Editor. How do the backplots compare to those produced within your CAM program?

3. Try single block mode to simulate some g-code programs. Follow what's updating in each of the screen areas as you step through.

TOOL CHANGES AND TOOL OFFSETS

CHANGING TOOLS IN G-CODE

Most CNC g-code programs will have one or more tool changes programmed into them. This will be true even if your machine doesn't have an automatic tool changer. That's because the tool change programming serves two purposes. First, in the case of a machine with an automatic tool changer (ATC), the tool changing g-codes tell the ATC to load a particular tool in the spindle or, in the case of a lathe, to rotate that tool into position on the lathe's turret. The second role of these codes is to set up a different tool length offset. The tool length offset tells the CNC machine how much the length of the current tool differs from tool #1. It will adjust its notion of where the tool tip is when you make your next move based on this.

Accomplishing these two functions can require anywhere from one to several g-codes. Let's go through each possible g-code and it's style of use.

T TOOL SELECT AND M06 TOOL CHANGE

The "T" word is commonly used to select a tool. On a VMC with ATC, usually the "T" word tells the mill to select that tool, but it requires the M06 word to be executed before the tool is actually changed. This gives the ATC a little bit of advanced warning, allowing it to rotate the new tool into position while the machine is busy doing something else, which can make for a faster tool change. On most lathes, and even for some mills, the M06 is not needed. On those machines, it is likely an error to use an M06.

Sample tool change using T + M06

N10 T12 (Select Tool #12)

N20 M06 (Change to selected tool)

If the machine doesn't use M06, the sample would look like this:

N10 T12

If you wanted to take advantage of the separate T and M06 to give your machine maximum time to get ready in order to make the tool change faster, you'd do something like this:

T12

M06

T14

(Machining with T12 in spindle, but T14 is ready for next M06)

M06

(Now T14 is loaded)

T02 (Setting up T02 for later)

(Machining with T14)

M06

(Now T02 is in spindle)

(etc.)

RANDOM MEMORY TOOL SELECTION

One trick some machines use to make tool changes go even faster is to set it up so tools don't have to go back into a particular pocket on the changer. Instead, the machine just sticks the tool in the nearest pocket so it doesn't have to take so long to rotate a pocket into position. The machine keeps track of which slot T02 is actually in, for example, and you can always refer to it as "T02"
no matter what slot it winds up in.

TOOL OFFSETS: GEOMETRY AND WEAR OFFSETS

Now what about that business of selecting a Tool Offset so the machine knows how long the tool is?

First thing to note is that there are actually 2 different length offsets--the Geometry Offset and the Wear Offset. The Geometry offset is associated with each tool by number and is the one that grossly determines tool length. The Wear Offset is a fine tuning change to the overall tool length that is used to compensate for wear and to give the machinist the ability to fine tune what's happening from part to part. For example, you may be turning a part to diameter and discover that after measuring the diameter with a micrometer it is slightly off. Perhaps it is 0.0013" too large. To correct, you can enter a wear offset of -0.0013 to cause the machine to come out much closer to the desired tolerance.

Let's ignore the Wear Offset for the moment. When you do your tool change, your machine controller will automatically apply the Geometry Offset based on the information contained in its tool table about each tool.

FANUC-STYLE COMBINED TOOL NUMBER AND OFFSET FOR TURNING

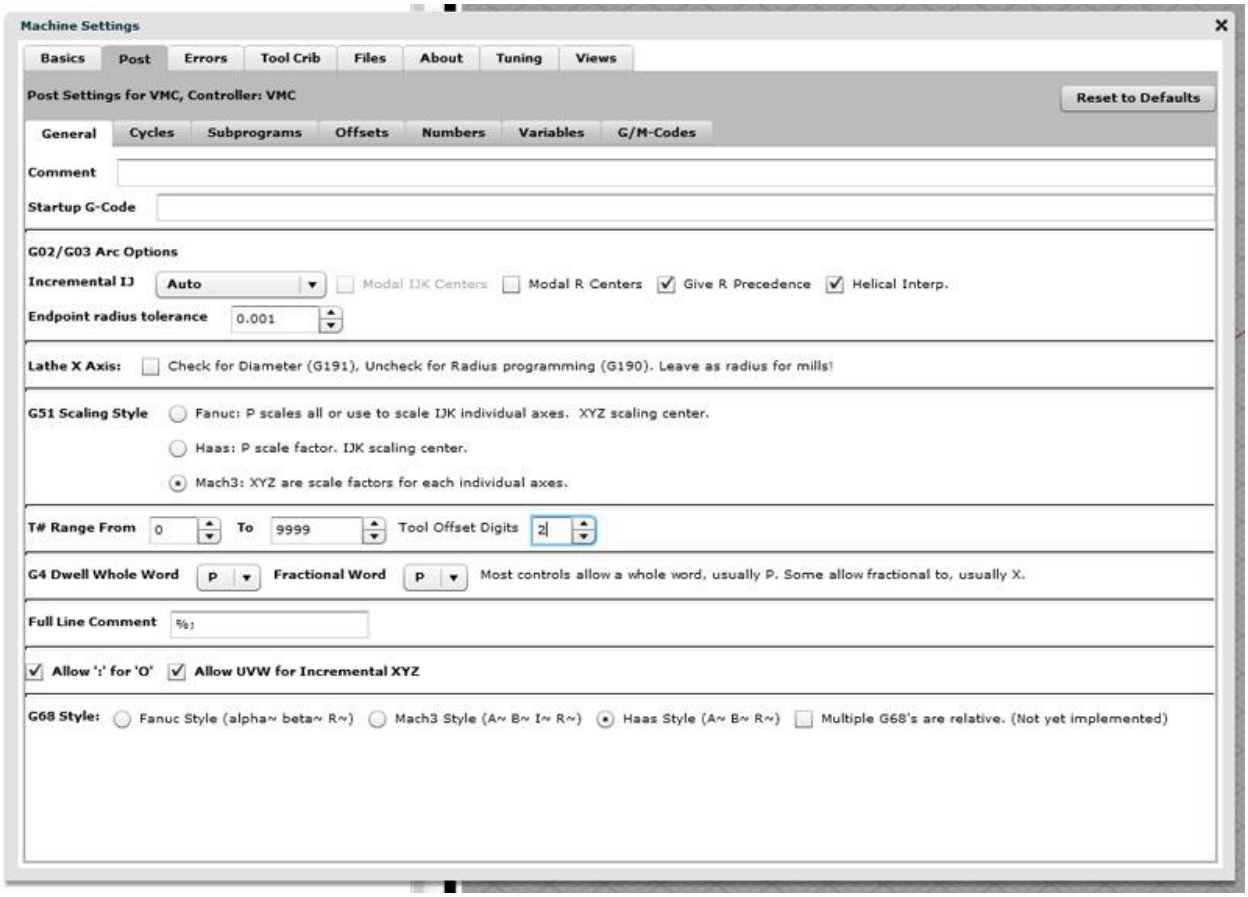
Now that we know about these offsets, let's take a look at specifying a toolchange on a lathe using Fanuc's syntax. It's pretty simple. To select tool #2, we'd use:

```
T0202
```

"Why is the "02" repeated twice?" you wonder. That second "02" can be any 2-digits, and it selects the wear offset to use. By convention, most programs make the wear offset the same as the tool number so you don't have to worry about matching them up.

G-WIZARD EDITOR/SIMULATOR TOOL CHANGE AND OFFSET POST OPTIONS

This is all pretty simple to set up in the G-Wizard Editor / Simulator's Post Options:



You have the ability to specify the range of T#'s as well as how many digits at the end of the T# are used to specify the offset.

EXERCISES

1. Get out your CNC controller manual and set up GWE to match your control's way of doing Tool Changes.

2. Write a sample g-code program containing a tool change and verify during GWE playback that the tool was changed.

3. For some extra credit, check out our 2-part article on Tool Data Management and start thinking about how you will manage your tool data.

EXTRA CREDIT

For more in-depth understand, be sure to check out these two articles:

Tool Length Offsets: Something every CNC machinist should know a lot about.

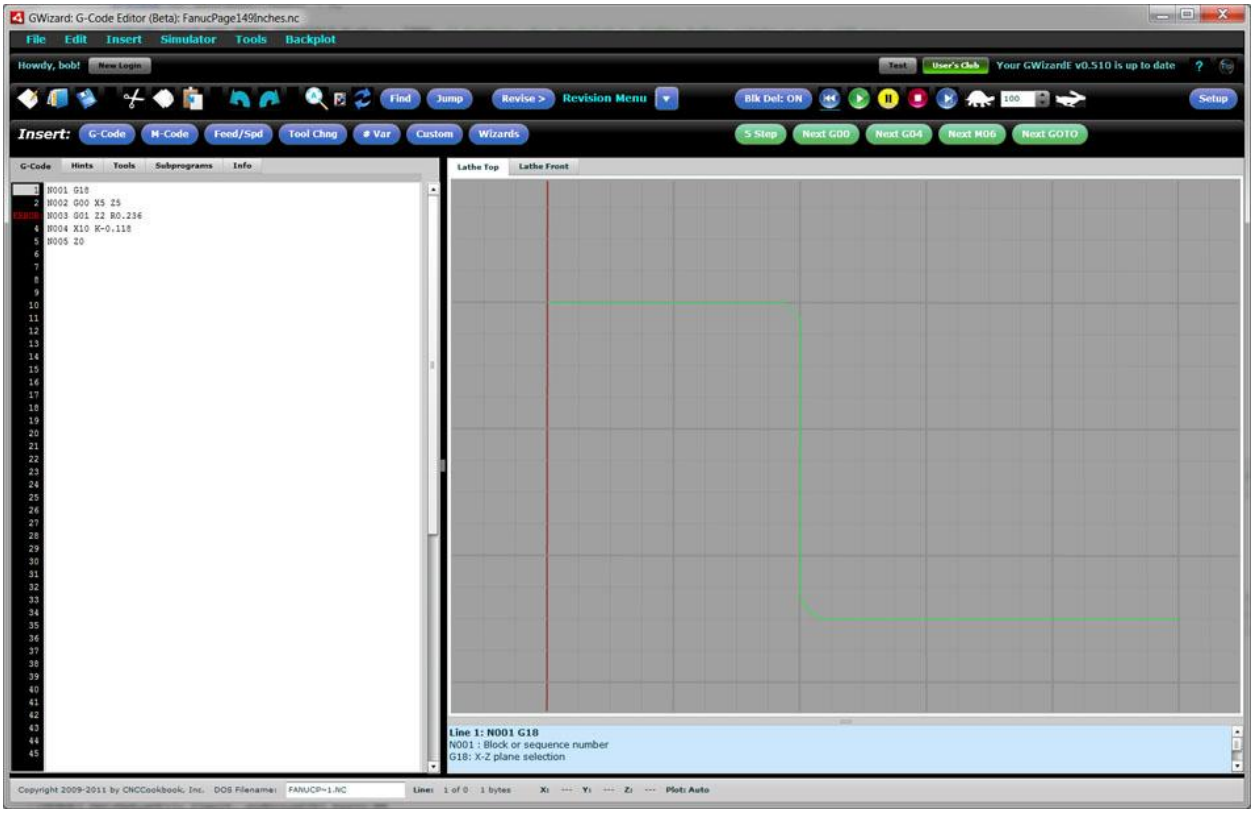
Tool Data Management: How do you keep up with all your tooling and the compensation information? What about tool presetting?

BASIC CNC LATHE PROGRAMMING

CNC LATHE AXES

CNC Lathes come in a variety of configurations, but for the basics, we'll stick to the simplest and most common setup--2 axes.

Simple CNC Lathes use a Z-axis, which is parallel to the spindle axis, and an X-axis, which is at right angles to the spindle. InG-Wizard Editor/Simulator, the display looks like this:



X-Axis runs top to bottom while Z-axis runs left and right...

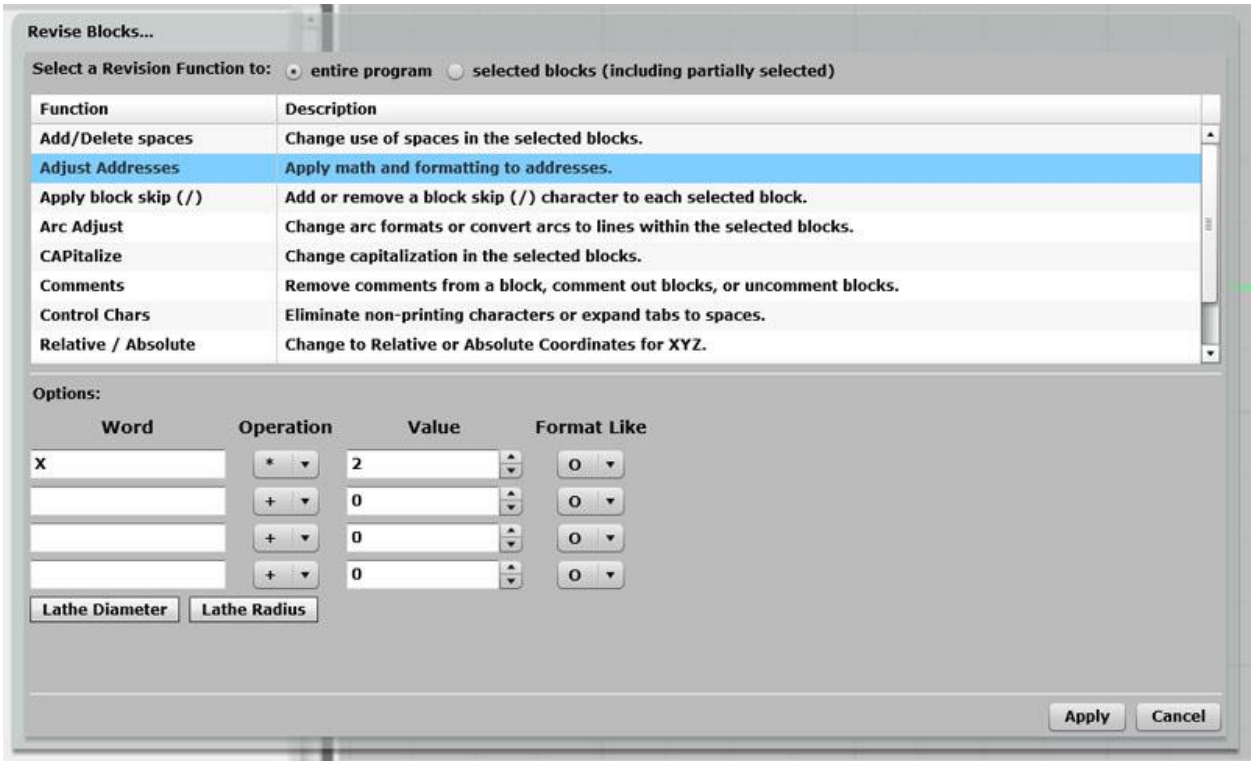
The G-Wizard Editor will automatically switch its axis display when you change the machine type from mill to lathe and vice versa.

The next step up from a 2-axis lathe would be a 3-axis lathe. Rather than add the Y-axis, a 3-axis lathe typically has X, Z, and C axes. C is a rotary axis parallel to the spindle axis and Z. In essence, it allows you to clock the spindle to specific angles. A three axis lathe with live toolign can do fairly arbitrary milling jobs on the part in the lathe chuck. We won't spend any more time talking about C-axis programming in this section because it's beyond the scope of a simple introduction to CNC turning.

DIAMETER VERSUS RADIUS MODE

Your CNC lathe's controller will be setup to start in either diameter or radius mode. In diameter mode, X-axis values specify diameters, while in radius mode, they specify a radius from the axis. As you would expect, diameter mode X-axis values are exactly twice what the radius mode values are, so it is important that you know which mode your lathe is using.

G-Wizard Editor has a revision command that lets you change programs back and forth between diameter and radius mode:



Lathe diameter mode is simple: multiply all the X's by 2...

As you can see, it ain't rocket science--we use the Adjust Addresses Revision to multiply all the X's by 2.

Some machines may also have the ability to change between radius and diameter mode using special g-codes.

BASIC G01 AND G02/03 MOVES: LINES AND ARCS

The CNC Lathe can make the same basic moves as the mill--G01 for lines and G02/03 for arcs (plus G00 for rapid motion in a straight line). The difference is you can mostly ignore the Y coordinate (though there are lathes that have Y too!). Just leave it out of the programs, and don't use the corresponding incremental "J" or "V" either.

Once you get used to it, lathe programming is quite a bit easier than mill programming. You are typically trying to create a profile of some kind. If your lathe has sophisticated canned cycles like G71 Rough Turning, this can be particularly easy, but even programming by hand isn't too bad.

PART ZERO ON CNC TURNING PROGRAMS

It's worth putting a little thought into where you'd like to put part zero on a turning program. The X-axis zero is always chosen to be the spindle center line when turning for all sorts of reasons.

For the Z-axis zero, there are three popular spots to choose from:

- Chuck face: The only advantage to this method is it is very each to touch the chuck face. But, the part is rarely against the chuck face, so this method introduces more complications in locating positions on the actual part.

- Chuck jaw face: This one is a little better, as it is both easy to measure and will also establish part zero at the end of the part that is pressed firmly against the jaw face. But, we can do even better.

- End of finished part: This is the most popular approach, the reason being a lot of parts have to be flipped in the chuck and machined on both ends. The disadvantage is that when machining starts, there is no finished part to touch off. Machinists typically use an allowance, or make a quick facing cut to create the finished end to get past that problem.

TOOL CHANGES AND TOOL SELECTION ON THE CNC LATHE

There are a couple of differences between typical CNC Lathe tool changes and Mill tool changes. First, the lathe uses no M06 command--the tool is changed as soon as the "T" word is executed. Second, many lathes encode a tool wear offset into the address following the "T" word:

Tttww

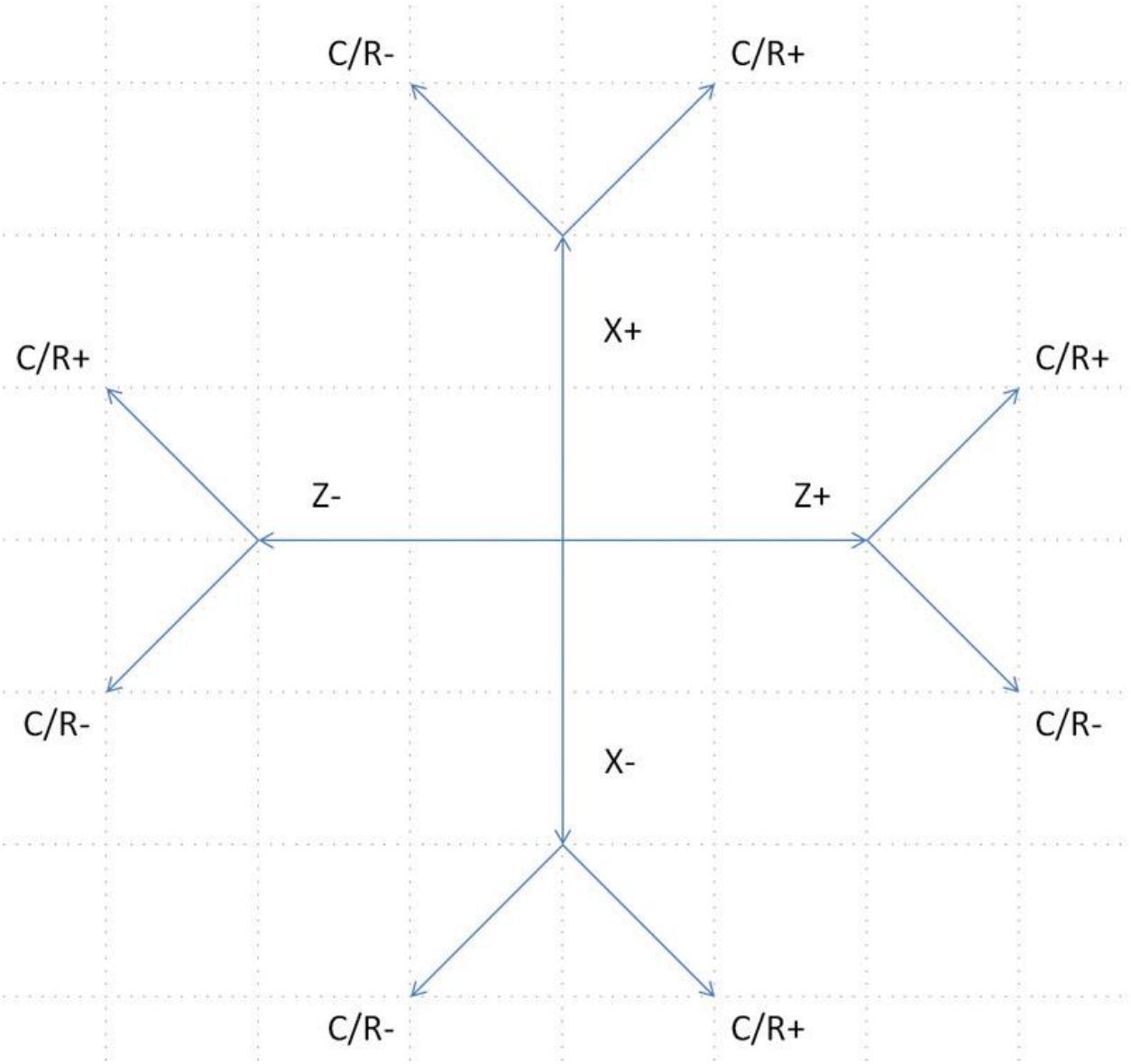
In the example "tt" signifies the tool number and "ww" signifies the wear offset. Different controls may be set up with differing numbers of digits to select the tool and wear offset.

CNC Lathes are set up this way so that each tool can have more than one wear offset. This is done because turning work often needs to be to very tight tolerances, and the same tool is often used for multiple features on the part. By using a different wear offset for each feature, the machinist can measure the parts as they come off and tweak the wear offset so each feature is to the specified tolerances.

This all assumes your lathe has a tool turret. It may not--some lathes use Gang Tooling. It's beyond the scope of this intro chapter, but gang tooling will be covered in a later article. Suffice to say that with gang tooling, the tools are mounted on the slide and the program selects the tool by explicitly programming slide motions on the X and Z axes.

AUTOMATIC CHAMFER AND CORNER ROUNDING WITH G01

When facing and turning at 90 degrees to one another, you'll get very sharp edges. It's very common for drawings to specify that these edges are to be broken with a chamfer or radius. Many controls provide a feature for just that purpose called automatic chamfer and automatic corner rounding. Once you get used to it, it's a very handy feature to have, but it takes a little getting used to how to set it up. I prefer to use "C" or "R" rather than I and K, so that's what I will explain here. We'll use this little chart:



To use the automatic corner break feature, the G01 must move in only one axis--X or Z. Using the chart, choose which axis is moving and in what direction. For each axis and direction, there are two possible chamfers (or corner radii). Choose the one you want

and that tells you the sign of the value. The "radius" is the amount of the value. Use the "C" word for a chamfer and the "R" word for a radius.

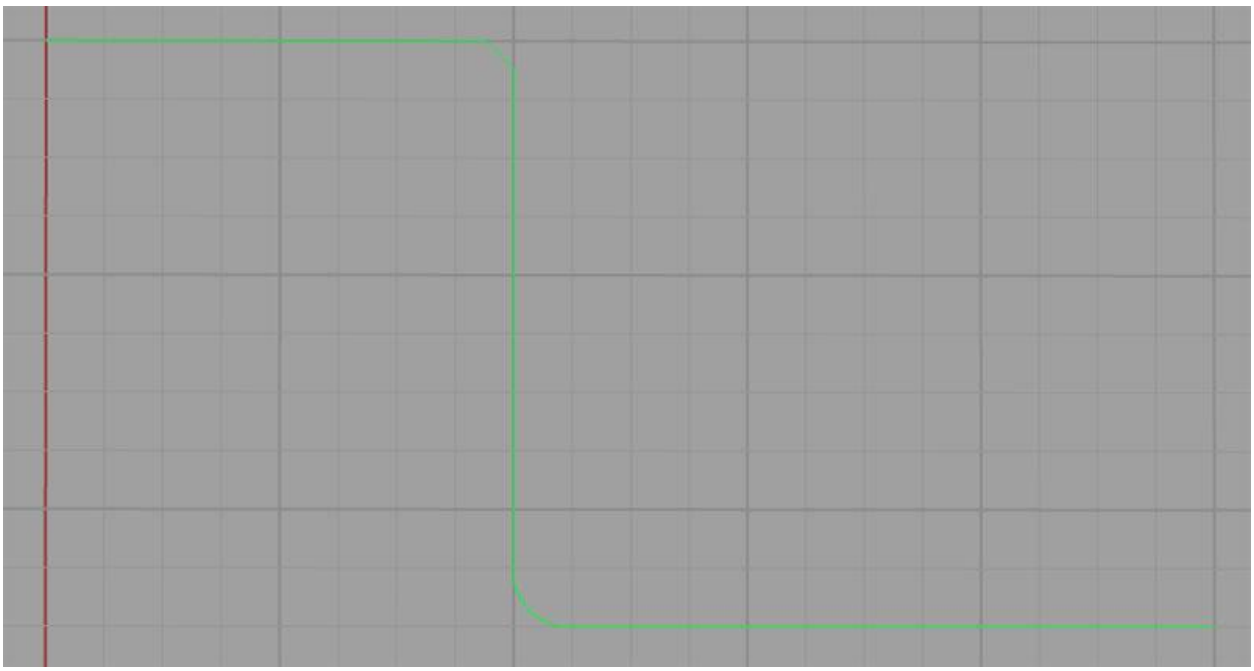
Let's try an example. Say we are moving up in X (so X+) and on our next move we'll go left in Z (Z-). Looking at the chart, we go up (X+) and then we want the "C/R-" branch. So, to give a chamfer, we use C-0.05 for an 0.05" chamfer. For a radius, we use R-0.05 for an 0.05 radius.

Here's another trick: if you're programming in GWE, just try out positive first and you can see whether it goes the right way. Switch to negative if it doesn't. That lets you dispense with the chart.

Okay, let's go over the example from the top of page screen shot. Here's the g-code:

```
N001 G18
N002 G00 X5 Z5
N003 G01 Z2 R0.236
N004 X10 C-0.118
N005 Z0
```

And here is the backplot from GWE:



G01 Automatic Chamfer and Corner Rounding Example...

Line N003 is a move purely in Z from right to left. We want to put a clockwise arc in with radius 0.236. Looking at the diagram, that means we want a positive R, so we use R0.236.

Next up is a chamfer at line N004. In this case, the chart tells us we need a negative "C", so we use C-0.118. Each side of the triangle with chamfer as hypotenuse is 0.118".

It's easy, try some examples yourself!

EXERCISES

1. If you don't already have GWE, take a moment now to sign up. It's free for at least 30-days and while in Beta test. We'll be using it for many of the exercises on each section of this course.

2. Review your CNC Lathe's programming manual and get used to how it's axes work.

3. Find out whether your lathe starts up in Diameter or Radius mode.

4. Using GWE, write some simple lathe programs that turn some simple profiles.

5. Modify the program you wrote in #4 to use the automatic corner break function and include both chamfers and radiused corners.

6. Review whether your CNC lathe's control has automatic chamfer and corner rounding available.

QUIZ ON BASIC GCODE PROGRAMMING

Congratulations—you've finished the Basic GCode Programming Course!

Now let's do a quick review in the form of our Quiz on Basic GCode Programming. It's online and should be pretty straightforward if you're familiar with the material. It's a good way to test yourself on what to go back and review further before continuing the the Intermediate GCode Programming Course.

RESOURCES

CNCCookbook Blog Posts Relating to GCode Programming: Lots more Tools, Examples, and detailed articles.

G-Wizard Editor: CNC Programming Software for g-coders and a CNC G-Code Simulator. We use it in this course to help teach G-Code.

G-Wizard Calculator: A CNC Machinist's Calculator

G-Code Reference for Mills

G-Code Reference for Turning

Sample G-Code Files: G-Code examples you can download and play with