Manufacturing Technology I

CNC Set-up and Operation

Section 1: Background Information

I. History of CNC Machining
II. Cartesian Coordinate System
III. Understanding the Coordinate System (Absolute and Relative coordinates)
IV. Introduction to CNC Machining “SRP vs. RP”
V. CAD, CAM and CNC
VI. Advantages and disadvantages of CNC Machining

Section 2: Before You Start Machining

I. Working with files and file formats
II. Selecting cutters
III. Spindle speed and feed rate
IV. Selecting materials
V. Clamping / Fixturing, Jigs
VI. Safety

Section 3: Introduction to CAD /CAM design and toolpath creation

I. Introduction to VCarve Pro
II. Introduction to creating toolpaths in VCarve Pro

Section 4: Running a part file on the CNC

I. Preparing the machine
II. Introduction to the Sb3 Shopbot Control Software
III. Transferring files to the machine PC
IV. Preparing material
V. Running a file

Section 5: Introduction to CNC Programming / G-Code

I. Using the line editor
II. Making changes to CNC code

Section 1: Background Information

History

CNC machines are very accurate and powerful industrial robots developed jointly by Mr. John Parsons, IBM and Massachusetts Institute of Technology Servomechanism Laboratory in the 1950's.

Most CNC machine tools use a language set by the Electronics Industry Association (EIA) in the 1960's. The official name of this language is RS-274D, but everyone refers it "***G-code***" or "G&M Code" because many of the words of this language begin with the letters G or M.

While many of the words used by different CNC machines are the same, there are differences between makes and models. This is due in part to machines having different configurations and options. For example, a machine with a chip conveyor will have words to turn the conveyor on and off, while a machine without a conveyor does not. So, while RS-274D is a standard, it is not rigid or enforced. Always refer to the machine documentation for the exact words and syntax for your CNC machine.

Most machines have a vocabulary of at least a hundred words, but only about thirty that are used often. These thirty or so words are best memorized because they appear in almost every CNC program and knowing them helps you work more efficiently.

The G-code language was developed when machine controls had very little memory. It was therefore designed to be as compact as possible. While at first this language may seem arcane, the modern machine tool language is the safest and most efficient way yet devised to control machine tool motion. ***G&M codes, along with coordinates and other parameters, comprise what is called a CNC program.***

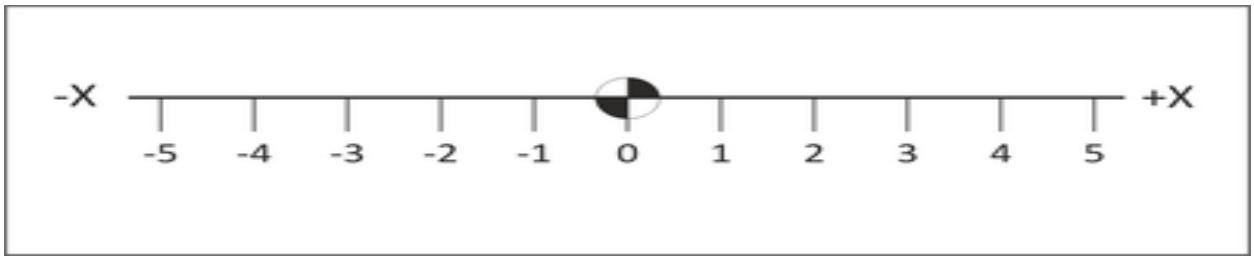
Cartesian Coordinate System

Motion is controlled along multiple **axes**, normally at least two (X and Y), and a tool spindle that moves in the Z (depth). The position of the tool is driven by direct-drive stepper motor or servo motors in order to provide highly accurate movements, or in older designs, motors through a series of step down gears. **Open loop control** works as long as the forces are kept small enough and speeds are not too great. On commercial metalworking machines, **closed loop controls** are standard and required in order to provide the accuracy, speed, and repeatability demanded.

CNC motion is based on a 3D Cartesian coordinate system.

Number Line

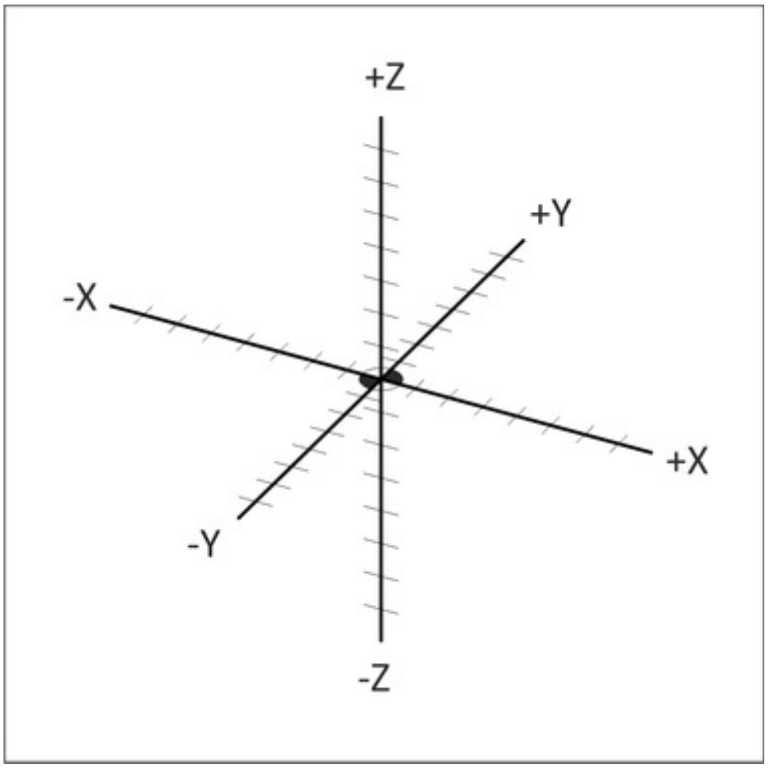
The basis of this system is the number line marked at equal intervals. The axis is labeled (X, Y or Z). One point on the line is designated as the ***Origin***. Numbers on one side of the line are marked as positive and those to the other side marked negative.



X-Axis Number Line

3D Cartesian Coordinate System

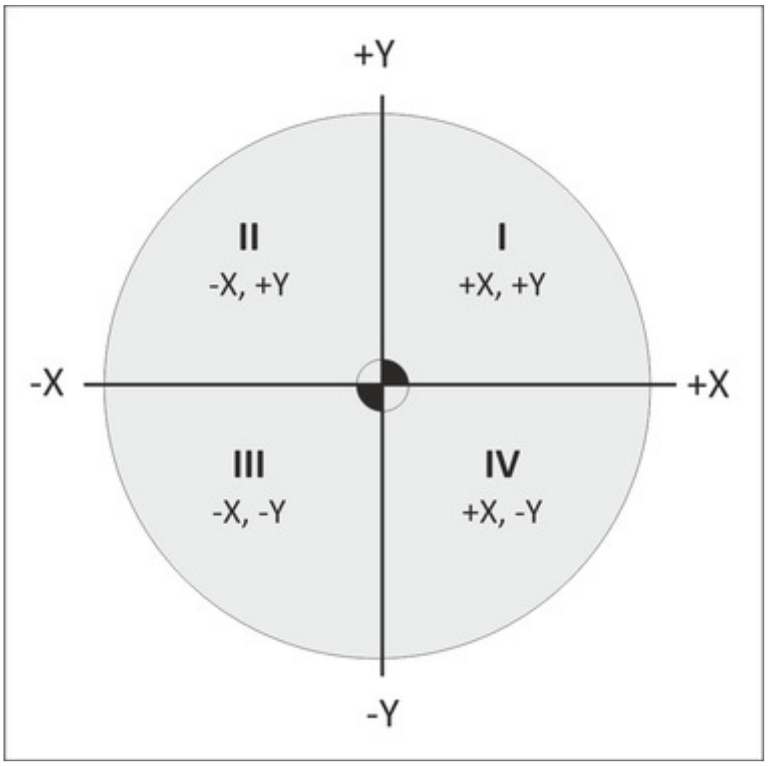
The Cartesian coordinate system consists of three number lines, labeled X, Y and Z, set at 90 degree angles to each other as shown in the figure below. The ***origin***, or ***Datum***, is where the three axes cross each other. The labels, orientations, and directions of the Cartesian coordinate system in this figure are typical of most Vertical Machining Center (VMC).



3D Cartesian Coordinate System

Quadrants

Any two axes form a plane. Planes are named by the axes that define them. For example, the figure shows the XY plane, which is the primary work plane for machining on a VMC. A plane can be divided into four quadrants, labeled I, II, III and IV with axes designations as shown in the illustration below.



Quadrants

Units

CNC Programs can be written in either Inch or Metric units. The machine can be switched with a single code to accept either.

In the United States, most programming is using inch units because most tooling is in inches and machinists are more familiar with the inch measurement system. Even if the part is designed in metric, it is usually converted to inch units for machining and metric tools are used only when no inch equivalent is available (for example when creating metric tapped holes).

The table below lists the units and maximum precision for inch and metric data used by CNC machines.

Units and Precision

Data Type	*Inch Units*		*Metric Units*	
Coordinate	inches	.0001	mm	.001
Speed	rev/min	1.	rev/min	1.
Feed	in/min	1.	mm/min	1.
Tap Feed	in/min	.001	mm/min	.01

Note; the machines in our shop generally run feeds and speeds in "inches per second" or "ips" not "inches per minute" or "ipm". The conversion is found below;

ips X 60 = ipm

Understanding the Coordinate System

Absolute vs Relative Coordinates

When writing coordinates it is standard practice to ***write them in the order of X, Y, and Z***.

When CNC programming the coordinate system must reference from a fixed point; this is called the ***origin*** or more commonly in manufacturing, the ***datum***. ***The datum is the position where X, Y, and Z all equal zero***. This is usually a point on the component and this position is usually decided by the manufacturing engineer or CNC programmer.

The coordinate system is almost always an absolute coordinate system. ***Absolute meaning all coordinates are measured from the datum or x. y and z zero points.***

Other coordinate system are found in CNC manufacturing; it is not unusual to find ***Incremental (Relative) coordinates*** used on many machines and it is possible to use Polar coordinates on most machines.

Relative or incremental coordinates do not refer back to the original datum, the position of the datum (or zero points) moves with the programmed coordinate. The machine moves towards a programmed position; when it gets to that position the position becomes X0Y0Z0 (the new datum). the next position is described from this new datum.

Introduction to CNC Machining

SRP vs RP

This course emphasizes an approach to CNC machining referred to as ***Subtractive Rapid Prototyping (SRP)***, also known as "***Subtractive Manufacturing***". SRP deals with small quantities of functional prototypes. Functional prototypes are made from materials like aluminum, steel and polycarbonate that cannot be produced with widely available additive ***Rapid Prototyping (RP)***, also called "***Additive Manufacturing***" processes such as SLA (Stereolithography) or FDM (Fused Deposition Modeling), what is commonly known these days as **3D Printing**.

SPR is not as simple to learn and use as RP. It takes more skill and often more time to apply. ***The main advantage of SRP is in materials. Almost anything can be machined***. SRP parts are not just visual aids, they are structural components that can be tested and assembled as part of working machines.

Another advantage of SRP is that it teaches real manufacturing constraints typical of the aerospace, biomedical, consumer goods, and electronics industries -all which use CNC for mass production, molds and other tooling. RP does not reflect these constraints. A part that is easy to rapid prototype may be extremely difficult, expensive, or even impossible to manufacture. ***SRP provides the designer with feedback about the manufacturability of design*** that can save considerable time and money as a part moves from concept to product.

Prototype vs. Production Machining

One of the biggest differences between making a few or many parts is in the design of work-holding fixtures. ***Prototype machining*** emphasizes quick, simple and cheap work holding solutions such as vises, clamps, screws or even glue or double-sided tape. High production parts ***(production machining)*** allow the cost of fixtures to be amortized over larger quantities to justify the cost of more elaborate and efficient fixtures.

CAD, CAM and CNC

What is CNC?

Definition; CNC Machining is a process used in the manufacturing sector that involves the use of computers to control machine tools. Tools that can be controlled in this manner include lathes, mills, routers, 3D printers, laser cutters, vinyl cutters and plasma torches. The CNC in CNC Machining stands for Computer Numerical Control.

In modern CNC systems, end-to-end component design is highly automated using computer-aided design (CAD) and computer-aided manufacturing (CAM) programs. The programs produce a computer file that is interpreted to extract the commands needed to operate a particular machine by use of a ***post processor***, and then loaded into the CNC machines for production. Since any particular component might require the use of a number of different tools – drills, saws, etc. – modern machines often combine multiple tools into a single "cell". In other installations, a number of different machines are used with an external controller and human or robotic operators that move the component from machine to machine. In either case, the series of steps needed to produce any part is highly automated and produces a part that closely matches the original CAD design.

What is CAM?

Computer-aided manufacturing (CAM) ***commonly refers to the use of numerical control (NC) computer software applications to create detailed instructions (G-code) that drive computer numerical control (CNC) machine tools for manufacturing parts.*** Manufacturers in a variety of industries depend on the capabilities of CAM to produce high-quality parts.

A broader definition of CAM can include the use of computer applications to define a manufacturing plan for tooling design, computer-aided design (CAD) model preparation, NC

programming, coordinate measuring machine (CMM) inspection programming, machine tool simulation, or post-processing. The plan is then executed in a production environment, such as direct numerical control (DNC), tool management, CNC machining, or CMM execution.

CAM systems can aid in creating, verifying, and optimizing NC programs for optimum machining productivity, as well as automate the creation of shop documentation.

What is CAD?

Computer-aided drafting (**CAD**) ***or computer-aided design and drafting (CADD), is the use of computer technology for design and design documentation. CAD software replaces manual drafting with an automated process.*** CAD software is used to increase the productivity of the designer, improve the quality of design, improve communications through documentation, and to create a database for manufacturing. CAD output is often in the form of electronic files for print, machining, or other manufacturing operations. The term **CADD** (for *Computer Aided Design and Drafting*) is also used.

CAD is an important industrial art extensively used in many applications, including automotive, shipbuilding, and aerospace industries, industrial and architectural design, prosthetics, and many more. CAD is also widely used to produce computer animation for special effects in movies, advertising and technical manuals, often called DCC digital content creation. The modern ubiquity and power of computers means that even perfume bottles and shampoo dispensers are designed using techniques unheard of by engineers of the 1960s. Because of its enormous economic importance, CAD has been a major driving force for research in computational geometry, computer graphics (both hardware and software), and discrete differential geometry.

The design of geometric models for object shapes, in particular, is occasionally called *computer-aided geometric design* (*CAGD*).

Its use in designing electronic systems is known as electronic design automation, or **EDA**. In mechanical design it is known as mechanical design automation (**MDA**) or **computer-aided drafting** (**CAD**), which includes the process of creating a technical drawing with the use of computer software.

CAD software for mechanical design uses either ***vector-based graphics*** to depict the objects of traditional drafting, or may also produce ***raster graphics*** showing the overall appearance of designed objects. However, it involves more than just shapes. As in the manual drafting of technical and engineering drawings, the output of CAD must convey information, such as materials, processes, dimensions, and tolerances, according to application-specific conventions.

Vector Graphics; is the use of polygons to represent images in computer graphics. Vector graphics are based on vectors, which lead through locations called control points or nodes. Each of these points has a definite position on the x and y axes of the work plane and determines the direction of the path; further, each path may be assigned various attributes, including such values as stroke color, shape, curve, thickness, and fill.

Raster Graphics; a raster graphics image is a dot matrix data structure representing a generally rectangular grid of pixels, or points of color, viewable via a monitor, paper, or other display medium. Raster images are stored in image files with varying formats

CAD may be used to design curves and figures in **two-dimensional** (2D) space; or curves, surfaces, and solids in **three-dimensional** (3D) space.

If you work in the architecture, MEP, or engineering fields, you've probably used 2D or 3D CAD programs. These programs can help you explore design ideas, visualize concepts through photorealistic renderings, and simulate how a design will perform in the real world. AutoCAD software was the first CAD program. Other CAD programs include Autodesk, Autocad, Solidworks, VCarve, Aspire, and Google Sketch Up to name a few.

CAD programs have different features depending on whether your design process involves 2D vector-based graphics or 3D modeling of solid surfaces. Most 3D CAD programs let you apply multiple light sources, rotate objects in three dimensions, and render designs from any angle.

Advantages of CNC Machining

CNC machines can be used continuously 24 hours a day, 365 days a year and only need to be switched off for occasional maintenance.

CNC machines are programmed with a design which can then be manufactured hundreds or even thousands of times. Each manufactured product will be exactly the same.

CNC machines can be programmed by advanced design software enabling the manufacture of products that cannot be made by manual machines, even those used by skilled designers / engineers.

A skilled engineer can make the same component many times. However, if each component is carefully studied, each one will vary slightly.

A CNC machine will manufacture each component as an exact match. Modern design software allows the designer to simulate the manufacture of his/her idea. There is no need to make a prototype or a model. This saves time and money.

Disadvantages of CNC Machining

CNC machines are more expensive than manually operated machines, although costs are slowly coming down.

The CNC machine operator only needs basic training and skills, enough to supervise several machines. In years gone by, engineers needed years of training to operate centre lathes, milling machines and other manually operated machines. This means many of the old skills are been lost.

Less workers are required to operate CNC machines compared to manually operated machines. Investment in CNC machines can lead to unemployment.

Many countries no longer teach pupils / students how to use manually operated lathes / milling machines etc... Pupils / students no longer develop the detailed skills required by engineers of the past. These include mathematical and engineering skills.

Section II: Before You Start Machining

Working with files and file formats

The CAD software packages that we use with our Shopbot CNC routers are VCarve Pro and Aspire. These two CAD design packages are essentially the same with the exception that Aspire gives you the ability to design true 3 dimensional organic shapes. We also use Autodesk Inventor HSM Pro as a CAD and CAM package for use with the Shopbot 5-axis CNC router. There are many different software packages available that you can use to design for our machines. You simply need to be aware what sort of file types are associated with VCarve, Aspire and Inventor HSM Pro.

The table below lists the ***file formats that may be imported into VCarve;***

File Suffix	Format summary
STL	CAD orientated 3D design packages and scanning systems
DXF	Many CAD systems – data must consist of meshes or triangles
3DS	3D Studio and many other animation orientated packages
OBJ	Wavefront
SKP	SketchUp software files
V3M	VectorArt 3D or Design & Make clipart models
LWO	Lightwave 3D Object model
3DM	Rhino 3D Model
WRL	Netscape 3D Live Picture

The table below lists the file formats that may be imported into Aspire;

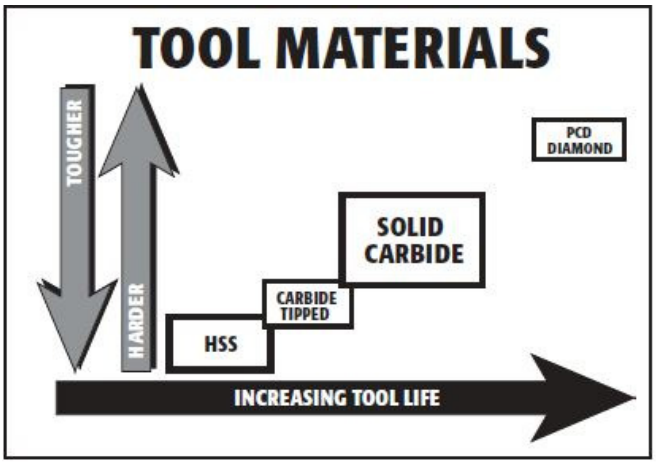
File Suffix	Format summary
STL	CAD orientated 3D design packages and scanning systems
DXF	Many CAD systems – data must consist of meshes or triangles
3DS	3D Studio and many other animation orientated packages
OBJ	Wavefront
SKP	SketchUp software files
V3M	VectorArt 3D or Design & Make clipart models
3DClip	Vectric Aspire Clipart files
LWO	Lightwave 3D Object model
3DM	Rhino 3D Model
WRL	Netscape 3D Live Picture

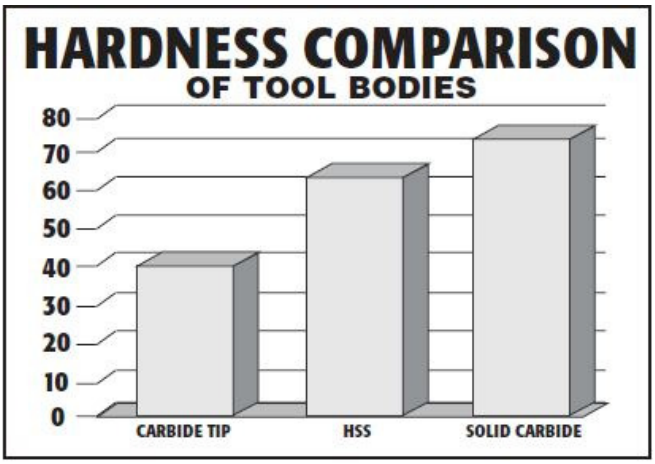
Selecting Cutting Tools

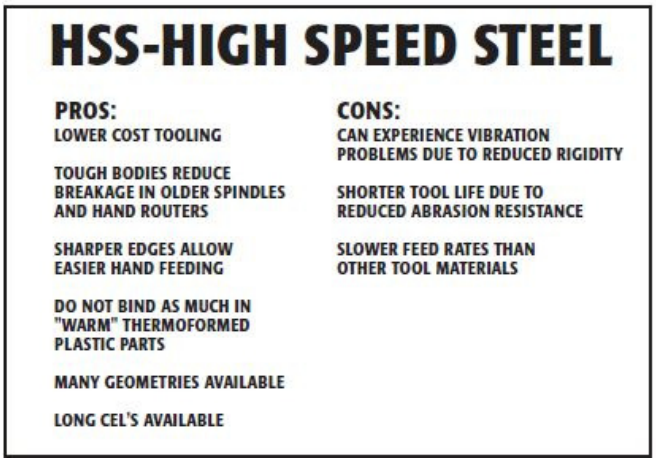
A wide range of tool types and configurations are available for CNC milling machines. Discussing every type, variation and use is beyond the scope of this course.

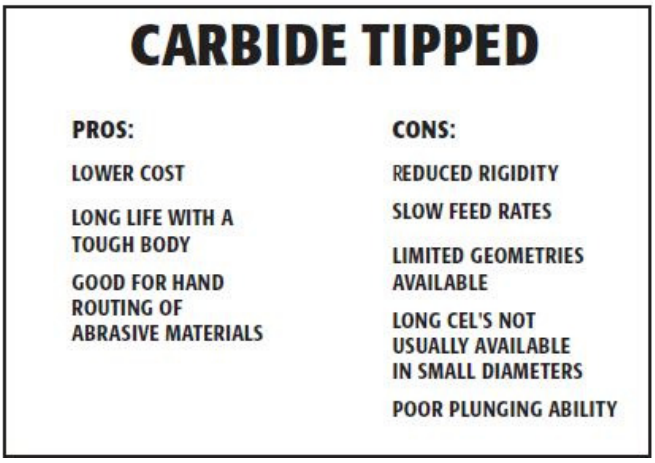
CUTTING TOOL MATERIALS

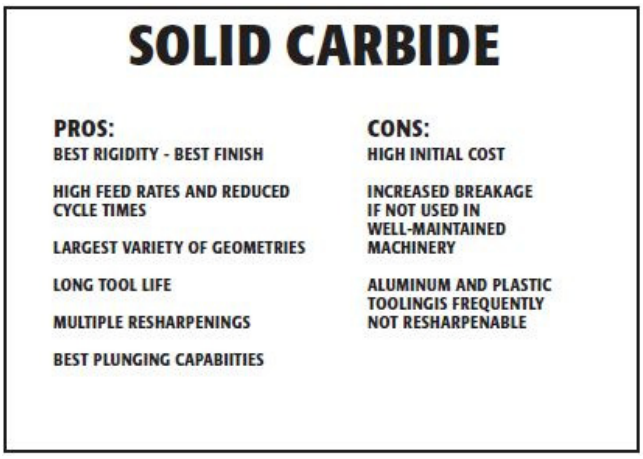
Tool materials utilized in the manufacture of router bits in the industrial market place involve high-speed steel, carbide tipped, solid carbide, and PCD diamond. The choice of tooling depends upon the relative characteristics of the material being machined and the equipment available for a specific application. High-speed steel and carbide tipped tools tend to fall into the category of manually fed router applications while solid carbide and PCD diamond is best applied to CNC operations. Basically, as hardness of tool material increases and toughness decreases, the tooling of the harder material functions better in the consistent feeding environment of CNC machinery.

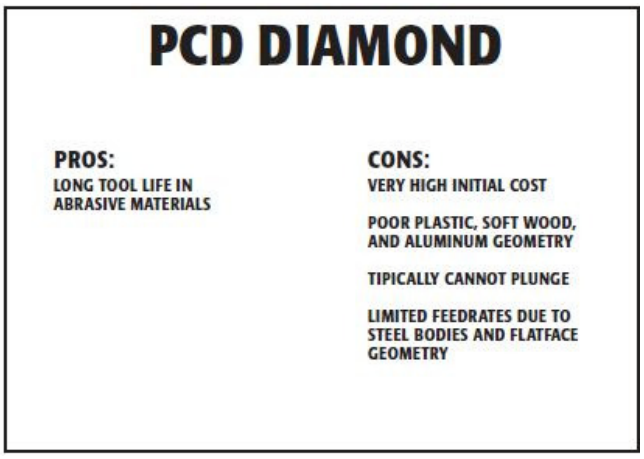








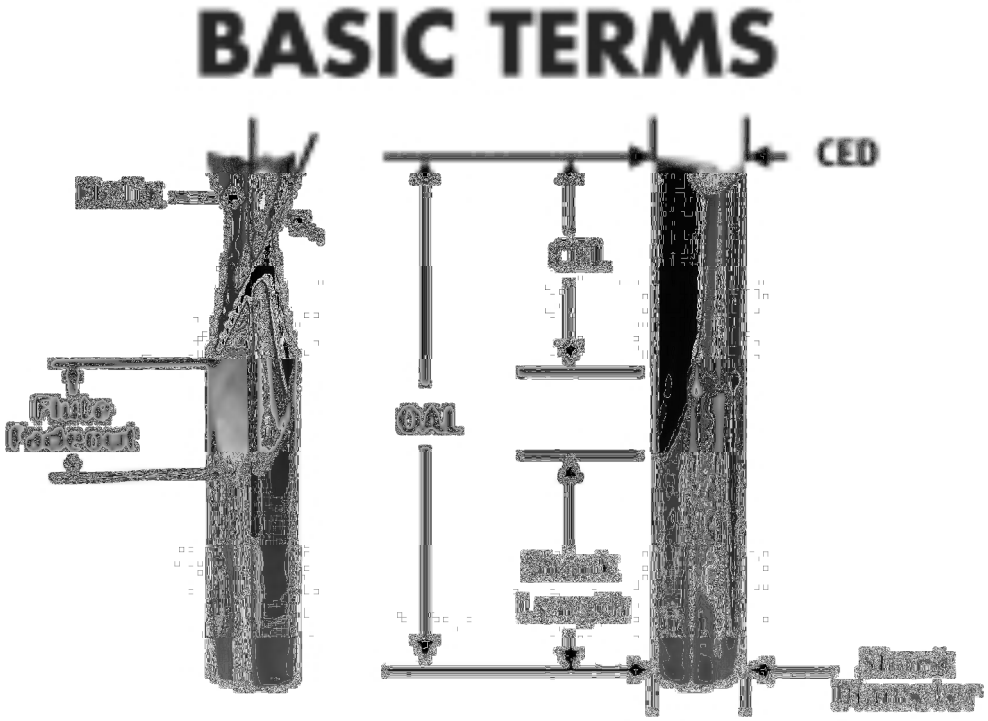




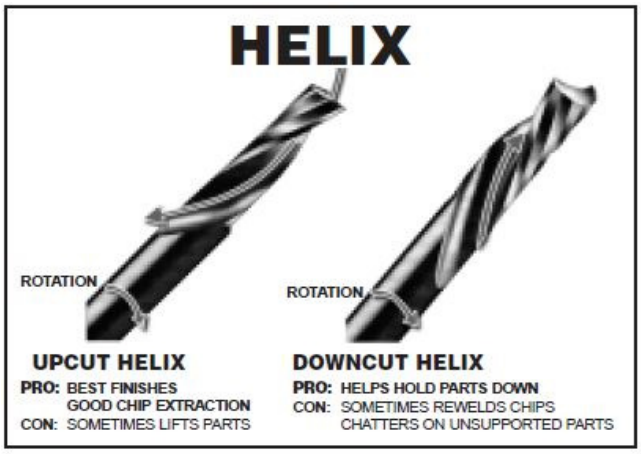
Cutting Tool Geometry

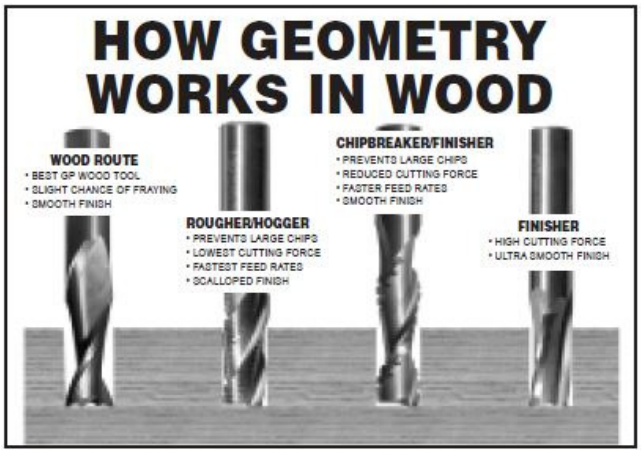
Cutting tool geometry influences many factors including the type of chip produced, the flow of the chip, the finish, and the actual force placed on the part. Consequently, it is important to

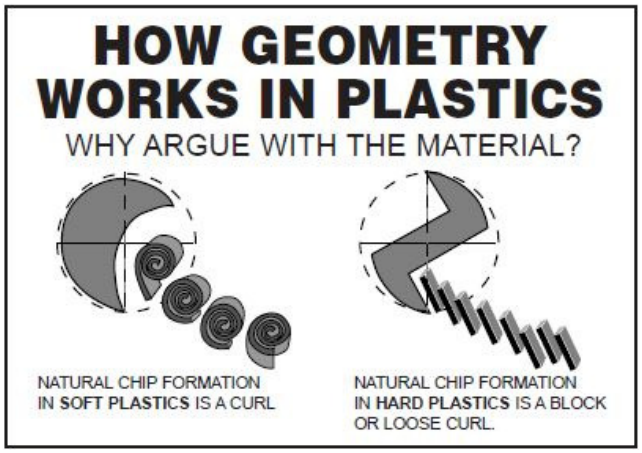
understand the basic terms associated with tool geometry and how these influences effect the machining process.



OAL= overall length CEL= cutting edge length CED = cutting edge dia.







Spindle Speed and Feed Rate

Working with Your Tool

ShopBot tools are used for a variety of purposes and in a variety of settings. For this reason, it is difficult to anticipate exactly how a tool should be configured for a particular application. In this section we review some of the general issues you will want to consider in setting up your tool for your specific projects. Keep in mind that the best information for your application will come from your previous experience working with your materials and cutting or machining characteristics and from doing some test cutting with different software and mechanical configurations of your tool.

Many of the same rules that apply to using a router by hand also work well when using a CNC machine. However, the advantage of programming the perfect cutting speed and maintaining that speed throughout a series of complex cuts puts you at a whole new level of accuracy and repeatability. For those of you experienced with routing by hand or mounting one under a table, many of these techniques will be second nature, but for those of you just starting out, you will need to consider the following variables very carefully.

Speeds, Feed-Rates and Rapid-Positioning

Before we even begin to consider what speed you want to be cutting at, recognize that there are limits to how fast your ShopBot will go. If you are not familiar with stepper motors, this will probably seem counter-intuitive but as stepper motors go faster, they lose power; the slower they go, the more power they have. Spindle equipped alpha tools can cut at 6 to 10 in/sec and jog at about 30 in/sec.

ShopBot has two Speed modes: One that is called 'Move' Speed and one that is called 'Jog' Speed. All movement is done either as 'M' Commands (Move Speed) or 'J'og Commands (Jog Speed). The movement in each case is equivalent (similar resolution and vectoring), but each Speed mode has a pre-assigned speed (Speeds are set with [MS], [JS], or [VS]). Move Speed is the speed that you've assigned for making movements that involve cutting or machining. ***Move Speed is the rate your material sees the cutter at and is thus your 'Feed Rate.'*** All ShopBot moves are fully 'speed vectored' which means that the cutter moves at the same continuous speed whether it is moving in the X axis the Y axis, on a diagonal, or in a circle or curve (unless speed is being 'ramped' up or down at the beginning or end of a movement, see below). The cutter velocity is constant.

Jog Speed is the speed designation for the rate at which the tool is moved when it is not cutting or machining material. Jog speed is normally set faster than Move Speed and Jogs are thus the method for the rapid positioning of your tool.

Choosing the correct Move Speed for your project is one of the most important choices in getting a good quality and efficient cut from your tool. There are a number of issues and considerations here. While we can offer some guidelines, there is nothing more important than your own experience with the cutter and material in making the decisions about cutting speeds. ShopBot tools working at 2"/sec or less have ample power to break bits and stall routers ... thus at ***Move Speeds the power of your ShopBot is not likely to be nearly as important an issue as your cutter type and size, the material you are using, the cutting direction, and speed and power of your router or spindle.***

At the two extremes in setting feed rates, you could be moving slow enough to burn or melt your material, or fast enough to give you poor edge quality, cause chatter, bog down your router and break bits. Somewhere between the two will be a move speed that has plenty of power to cut at a fast rate without sacrificing edge quality. For ¾" hardwoods, that might be between .75" and 1.25" per second and for ¾" plywood you might be traveling between 1" and 1.75" per second. For thicker or harder materials, a second pass may be in order. Here is where some trial cutting and close inspection will really help. Whatever your optimum speed turns out to be we recommend that you program that speed into the top of any file that you will be using repeatedly, so that you always have a reliable feed rate.

Using Chip Load Calculations to Improve Cutting

One of the challenges in getting good CNC cuts is in selecting the best cutting speed (feed rate) and router/spindle RPM. Selection can be facilitated using the guidance provided by manufacturers regarding the best 'chip load' for a particular cutter. Since ***chip load (the size of the chunk of material taken by a tooth of the cutter)*** reflects the combination of how fast the cutter is moving forward into the material and how fast it is turning ***(Chipload = Feedrate / [RPM x number of flutes])***, it gives you starting point speed values for testing to determine the most suitable parameters for any cutting situation. The ShopBot Chip Load Calculator simply

provides a quick way to explore these values, and is most useful if you know the approximate desired chip load for a particular cutter and material.
The following are some very rough chip load ranges for cutting various materials. Keep in mind that the most effective chip load depends on the specific geometry of the cutter, so you will want to check with the maker of your specific bit, and then keep your own records on what values work best for you.

A Suggested Feed Rate Starting Point. For working in most woods, we suggest you start testing at 1.7" per second for your XY Move Speed. Evaluate your bit and material to see if you can work at faster speeds and still get smooth cuts. For some tasks, it may be necessary to move more slowly. Set the Ramp Speed to .4. Ramp Speeds are not of real importance with the relatively slow movement speed at this cutting rate. But as you start moving faster, ramping can assure that you have a smooth start and stop.

A good starting Move Speed for your Z plunge will be about .5"/sec. This should give you plenty of drilling power. Set the Z Ramp Speed at .4.

TIP: It is hard to over-emphasize the importance of a sharp bit for smooth, clean cuts. Bits will seem to wear quickly because it is very easy to accumulate mileage with CNC cutting. Do not be tempted to use an old dull bit for reasons of economy. The cut will be rough and the cutter will exert extreme forces on your tool.

S*uggested Starting Jog Speeds*. Jog Speeds should be set to speed up your program run time by moving you to the next cut as quickly and efficiently as possible. Please note here that an efficient tool path can be just as important as a fast tool. The danger of going too fast with stepper motors is that there is an inverse relationship between speed and power. If you set your speed too high, your power will drop and you can risk stalling the motors and losing position.

Selecting Materials

Solid wood—Good old-fashioned, beautiful, natural wood is a great routing material. For millennia people have been carving intricate patterns into wood. CNC routers just made it a whole lot quicker and easier. Great hardwoods include: ash, oak, cherry, mahogany, hard maple, etc. Routable softer woods include pine, soft maple, fir, spruce, hemlock, cedar and redwood.

MDF—Medium density fiberboard (MDF) is a big player in the sign and exhibit industry. MDF is made by converting wood chips into fibers. Liquid resins are added to create a thick mat that is pressed into boards using heat and pressure. An advantage of MDF for the graphics industry is that the product is uniform throughout, including the edges. MDF edges are smooth and beautiful straight off of the router. One disadvantage of MDF is that paint is generally required to finish it, even if it will be covered with a digital graphic.

Plywood, HDO and MDO—Plywood is one of the earliest signage materials of the last century, and an excellent candidate for routing. Plywood is created with layers of veneers. These layers are alternated with the grain running at right angles, then glued together under pressure in large presses. For outdoor grade plywood, the adhesive used is generally a phenolic resin that is resistant to moisture, bugs, heat and cold. These synthetic resins are also impregnated into smooth top layers to create overlay plywood like HDO and MDO. Plywood's large size, high strength-to-weight ratio and dimensional stability have made it a staple of the signage and

display industry since its inception. Overlay plywood accepts paint and digital prints; however the layered edges need a lot of finishing to look good. Other routable wood-based composites include OSB, LDF, particle board and Masonite.

Plastics

Expanded PVC—Lightweight routable expanded PVC sheet products like Sintra, Komatex, Komacel and Celtec have virtually taken over the middle price point of the indoor display market. Light, tough, stable and very easy to work with, expanded PVC can also be fire resistant and directly printable. Routers cut through it like butter, leaving perfectly finished edges. And they are available in a variety of colors, meaning that no paint is needed for further savings.

Acrylics—Plastic acrylic sheet is another product that stands strong in the router marketplace. Beautiful, versatile and durable, these rigid sheets come in a large variety of thicknesses and colors. They make very stylish and unique displays that are easy to fabricate. Special router bits for plastic can polish and finish the edges as well. Some brands include Plexiglas, Acrylite, Lucite and Lexan.

Polycarbonates—This is one of the toughest materials on the market. Polycarbonates have excellent impact-, heat-, cold-, stain-, UV- and chemical-resistance. They represent a very long-term, outdoor-durable and flame-resistant solution. Polycarbonates cut easily on a CNC, but the edge finishes are not as clean as acrylics. The nature of the molecular structure of polycarbonate that creates its strength also creates a minor problem that makes the baby smooth edges of acrylics and MDF products more difficult to achieve.

Sign Foam—Sign foams are used quite differently than other materials in this category. Rather than just be used as substrates to carry graphics, they come in thick, easily routed slabs that are ideal for 3-D sculptural signage. In other words, the shapes are the graphics. This wonderful stuff has made a whole new dimensional genre of signage possible. And in comes in a wide variety of qualities, densities and sizes.

Other Routable Plastics—There are a number of other plastics that are considered routable and useful to sign makers. They include *ABS, polyethylene, PVC, polypropylene, polystyrene, HDPE and PETG.*

METAL

Aluminum—Most metals don't fit into the "easy to CNC" category, with the exception of aluminum and honeycomb aluminum. And for long-term outdoor durability it's hard to beat this common metal as well. It is completely impervious to any form of weather and 100 percent recyclable- a winning combination. Special CNC bits are available for this material that make cutting smooth and easy, albeit at a somewhat lower speed than many other substrates.

Other Metals—A number of other metals are considered routable, but may not be quite so easy to work as aluminum. These metals include *brass, copper, bronze and mild steel*.

Aluminum/Plastic—One of the most common router substrates are the class of composites that employ aluminum and plastic. Utilizing the light weight and moisture resistance of thermoplastics for the core and thin layers of painted or anodized aluminum for the outer surfaces, these relatively new products are perfect for sign production. They often have .012"

outer layers with 2mm, 3mm and 4mm core thicknesses. Sizes include 4'x8', 4'x10' and 5'x10' sheets. It is dimensionally stable and weather-proof yet much lighter than solid metal. Composites are a UL-recognized component for electrical signs, have a Class 1 and Class A fire rating and are NCHRP 350 approved for highway and safety signage.

Their layered structure gives them excellent strength and rigidity, meaning that less substructure is required during construction. They are easy to work with using ordinary shop and woodworking tools and come pre-painted for a completely hassle-free experience. CNC routers love these unique products and spin through them with ease. The layered edges may not be quite as beautiful as PVC or MDF, but they are smooth and clean. Some aluminum composite brands include Alupanel, AlumaCorr, Dibond, Graphic-Al and KomaAlu.

Clamping / Fixturing and Jigs

Often the terms "jig" and "fixture" are confused or used interchangeably; however, there are clear distinctions between these two tools. Although many people have their own definitions for a jig or fixture, there is one universal distinction between the two. Both jigs and fixtures hold, support, and locate the workpiece. ***A jig, however, guides the cutting tool. A fixture references the cutting tool.*** The differentiation between these types of workholders is in their relation to the cutting tool.

There are many techniques that can be used to hold your material to the work surface or "spoilboard". ***A* spoilboard *is a disposable work surface mounted atop the router's permanent table. The* spoilboard *is typically MDF and protects the router table from damage as well as being an expendable surface that can participate in workholding.***

Different machining processes and materials will dictate which hold-down technique should be used. Often times there are multiple choices for similar processes and materials.

Jigs; In simplest terms, a jig is any object that makes the basic tasks of woodworking easier. A jig is used in addition to the primary tool, be that a saw, router or other instrument. Using jigs provides a means of speeding up repetitive tasks, ensuring more accurate cutting or measuring, or repeating standard sizes and positions on work pieces. Jigs tend to guide a machine or process.

Examples of Jigs include featherboards, circle cutting jigs, router jigs, circular saw jigs and box joint jigs.

Hold-down / Clamping; When you are machining a single part and you don't need to be concerned about being able to place subsequent parts in exact locations, you can use a simple hold down or clamp system. This can consist of screws driven into the **spoilboard** to hold your material, spray glues or other adhesives, double sided tape, brad nails (nylon nails work great as they don't hurt tooling in the event of a collision) or wedge systems.

What you must keep in mind is if the part you are machining will consist of ***through-cuts. This is a process where you will be cutting completely through your material,*** such as in a profile-cut, that will allow you to remove your part from the surrounding material. If you are performing such an operation you must keep in mind that your part still needs to be held in place throughout the machining operation. This will require the use of either tabs placed in the toolpath, double sided tape or some type of adhesive.

Fixtures are workholding devices designed to hold, locate and support workpieces during manufacturing operations. Fixtures provide a means to reference and align the cutting tool to the workpiece but they do not guide the tool. Fixtures that have the added function of guiding the tool during manufacturing are called jigs. Fixturing devices include: Various standard clamps, chucks, and vises, metal plates containing dowel and/or tapped locating holes or key slots and dedicated fixtures with specific design and build requirements.

Safety

Rules for a safe workplace:

- ✓ Never use any equipment which you have not been trained to operate by a qualified person.
- ✓ Never tamper with a machine safety guard or switch.
- ✓ Use caution when handling cutting tools. They are very sharp. Never handle a tool by its cutting flutes.
- ✓ Never start or jog the machine until you have checked that the work area is clear.
- ✓ Never push the start button on the machine unless you are certain your setup is capable of safely holding the part against all cutting forces during machining.
- ✓ Use caution when running a new program: especially at the start of program and after a tool change.
- ✓ Always set the “Z” height after changing tooling or materials.
- ✓ Know where the emergency stop is on the machine and practice using it before you need it.
- ✓ Never run a machine alone or without other people within hearing distance.
- ✓ When working with someone else at the machine, clearly communicate who is running the machine.
- ✓ Never have one person touching the control while the other is working in the machine envelope.
- ✓ Never use an air hose to clear chips from a machine. Flying chips are dangerous to you and others.
- ✓ Lift with your legs, not your back.
- ✓ Never lift anything more than you can comfortably handle.
- ✓ Get help handling heavy or bulky objects.
- ✓ At the end of the program, command the machine to position the gantry away from operator and the finished part so it can be easily reached with the spindle out of the way.
- ✓ Never leave a running machine untended.
- ✓ Before shutting the machine down, remove any tools from the spindle.
- ✓ Report any injuries immediately.
- ✓ Remain alert. Think safety in everything you do.

Use these extra precautions when running a CNC program for the first time:

- ✓ Use machine Rapid and Feed override controls to slow the machine down.
- ✓ A major cause of crashes is setting the tool or fixture offset incorrectly. Pay particular attention to moves at the start of program and immediately after a tool change as the

tool moves towards the part. Remain at the machine with a hand on or near the emergency stop button.

- ✓ Stop machine motion at the first sign of trouble.
- ✓ Learn how to interpret the machine G-code and review your file prior to running it. Often times you can see a problem in the G-code before running a file and correct it.
- ✓ Always run your file in a "3D Offset" first. This will assure that your part is positioned and oriented correctly on the spoilboard and that there will be no collisions with hold-downs or fixtures.
- ✓ Check your file run time before running it. Make sure you have time to complete the job before you have to leave the shop.

Section 3: Introduction to CAD / CAM Design and Toolpath Software

Introduction to VCarve Pro / Aspire

VCarve Pro and Aspire are essentially the same program with one major difference. Aspire functions exactly the same as VCarve Pro but also allows for the design and toolpath creation for 3 Dimensional machining. For the purposes of this discussion we will only focus on the capabilities that exist in VCarve Pro.

Before you can cut anything with a CNC, you need to first create the design layout that the machine is going to follow to cut the parts. The software you choose will play a significant role in successfully creating projects with your CNC. Simply put, the design software will allow you to transform "pencil and paper" ideas to a set of instructions used to run the machine. When done correctly, the end result will be a physical product you can touch and hold that has value and purpose and a great sense of achievement.

Every CNC machine needs software to directly drive its movement; this is commonly referred to as the 'Control Software'. Some common generic third-party packages that do this include "Mach3" and "WINCNC". The control software that runs our Shopbot machines is called "Shopbot 3" or "Sb3".

Workflow

When you step back and look at a complete project from start to finish, you can identify a series of major steps that will form the "Workflow" to complete it. Having a good understanding of this process will help you start to appreciate where the different software packages and setup procedures fit into the overall creation of parts with your CNC.

Concept; This is the idea for what you are going to make. This may range from a specific customer requirement, something you have sketched on a napkin or a ready to go file that someone has already prepared. At this stage you need to try and think through the other processes in the job to help to get the best approach to achieving it. You should also assemble any reference material you will use to help design the part such as photos, data from the customer, design sketches etc.

Design / CAD; For the design you need create the computer data that will define either the 2D or 3D forms you want to cut on your CNC. This is done in what is typically called “CAD software” and you may also hear this type of software referred to as a drafting, drawing or design program.
The finish point of the Design stage is to have prepared all the 2D data (Vectors) or 3D data (Components) you require to start calculating the specific movements the CNC machine will follow, these moves are typically referred to as the “Toolpaths”.

Toolpaths; Once the design is complete, you will start to calculate the actual paths that will drive where the tool will move on the machine, as previously stated these are called “Toolpaths”. Creating your Toolpaths is the key stage in going from the virtual world of a computer design to the reality of the physical world. At this point you will start to take into account the shape and size of the tool, the type of movement you want the tool to make (the shape you want it to leave in the material) and appropriate settings for how fast the tool can be moved and how much material can be removed safely.

Once the Toolpaths have been calculated the software will let you preview how they will look in a virtual piece of material. This lets you check that they are doing what you expected. Once you are happy the Toolpaths are correct then they can be saved in a format that is appropriate for our Shopbot machine.

Machining; Once your toolpaths have been saved then you transfer them over to the computer that operates the CNC. ***You cannot run a file directly from a USB flash drive! You must copy the file to the computer then remove the flash drive before running your job!*** At this stage you need to set the CNC to match the job setup you specified in the CAD / CAM software. This will involve setting up your material in the right orientation, and making sure it will be secure while you’re cutting it. Then you need to load the correct tool and tell the machine where the X, Y and Z reference position is for the tool tip (normally this is the zero position for each axis), again this will be to replicate how it was set in the software so all the positions and sizes you specified in the software will be replicated at the machine.

Once the machine is setup correctly, the toolpath can be loaded into the SB3 control software and then executed. The machine will feed the co-ordinates of the toolpath to the machine to continually move its position and create the cuts you setup in your toolpaths. Running a toolpath may take less than a minute or potentially many hours depending on what type of operation it is. Once it is complete, you can run additional toolpaths and if required change the tool and reset the Z zero datum position for the new tool. Once you have run all your toolpaths you remove the material from the machine.

Creating a Design in VCarve Pro / Aspire

At this stage you need to create either a 2D design (Vectors), a 3D computer model (Components), or a combination of the two. This data will be used to calculate the paths the CNC will follow to actually cut the finished parts.

Depending on the reference material you have, you may start with a blank job and create everything within the software or you may be importing information such as an image to trace or an existing 2D or 3D design supplied from another software package.

Once you have an idea of what you are trying to produce, the next step would be to start the software and setup a new project, followed by doing the actual CAD (design) work.

The part is designed in a virtual area which is setup to represent the size of the piece of material you plan to machine it into (see image to right). This allows you to reference the correct size and position that the part will be cut in as you complete the layout. This includes the material size in Height, Width and Thickness (each of which relates to the X, Y and Z axis), where on this material will you set your machine's X and Y axes to zero (referred to as the XY Origin or Datum) and where you want to set the Z axis zero position for the tip of your router bit (tool).

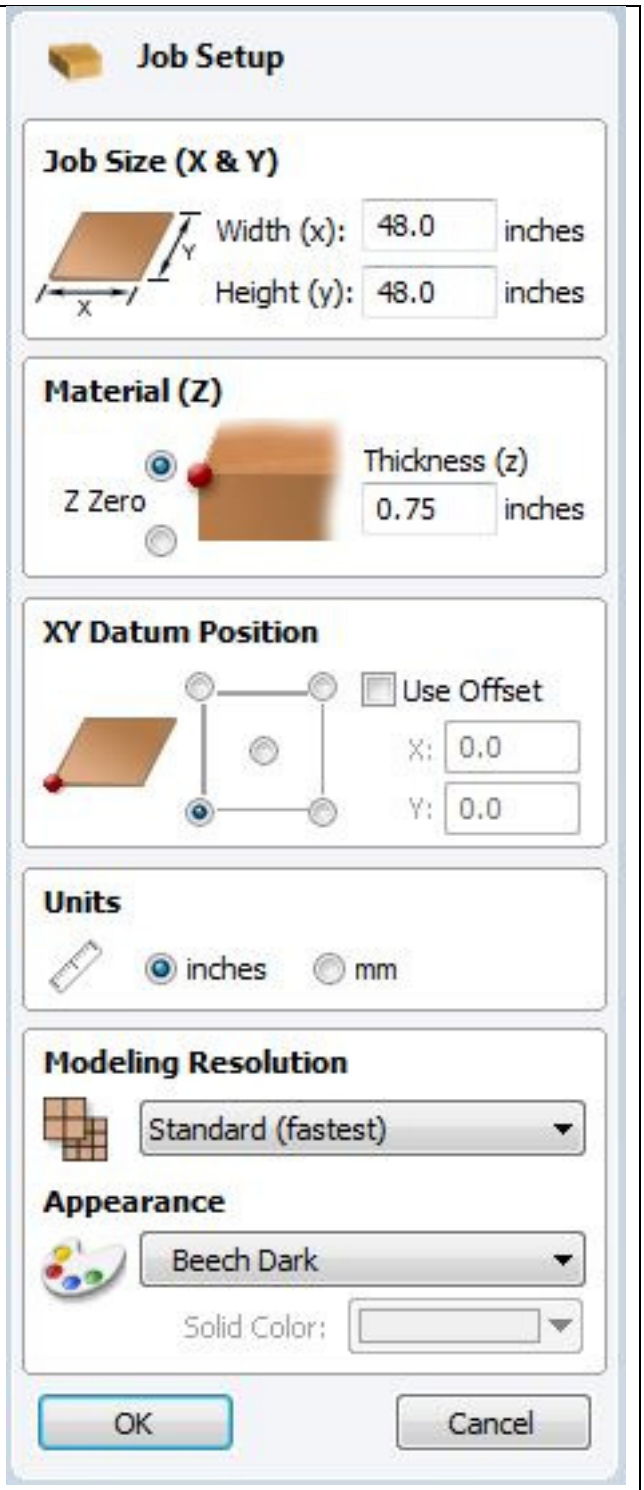
The size and location of all these points will be replicated when you set up the CNC, and this is how you relate the virtual positions in your software to the real positions on your CNC.

As the design evolves, these values can be changed if required and only needs to be finally set before you calculate the actual toolpaths. It will be important to make any final adjustments before calculating the toolpaths.

There are many considerations as you choose where to set the XY Origin and Z Zero points. There is no right or wrong answer here as much of this is personal preference, but here are some considerations that may help for a specific project:

Job Size (X & Y): When setting Material Size, it can be the actual job size or just large enough to accommodate the job and space to cut it out. If you are going to be cutting your job from a much larger size of material than its actual size, then it is suggested for the design phase that you setup a material size just slightly larger than your job will be. You can either change this before calculating the toolpaths or simply position the XY Origin where you need it on the larger sheet of material.

Z Zero: The program gives you a choice between the top of the material or the bottom of the material. This is also where you will set

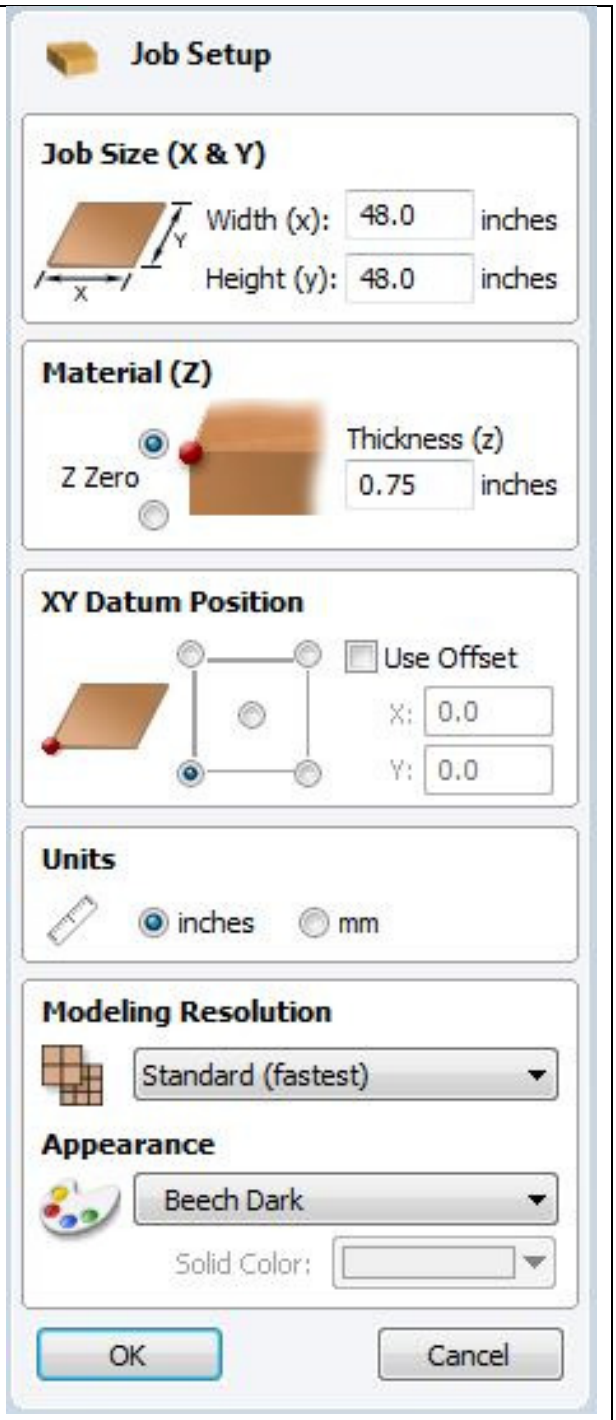


the thickness of your material. You need to measure the actual thickness of your material very carefully. This is critical when performing through-cuts.

Units: Under all but the most unusual circumstances we will be using "inches" as our units.

XY Origin: The programs give you 5 possible choices for this: The center of the material or one of the 4 corners. Most projects tend to use either the center or lower left corner. Again, this can be changed before the toolpaths are calculated.

Modeling Resolution: The more detailed the 3D project will be, the more model resolution may be required. The larger the 3D project, the more model resolution may be required. A good practice when creating a 3D model is to keep the Material Size just a little larger than the 3D model size. If this is not possible, such as when creating a curved molding, a higher model resolution may be required. The higher you set the resolution, the slower your model will render and the larger the file size will be. Your actual project is limited by the bit you choose to use.



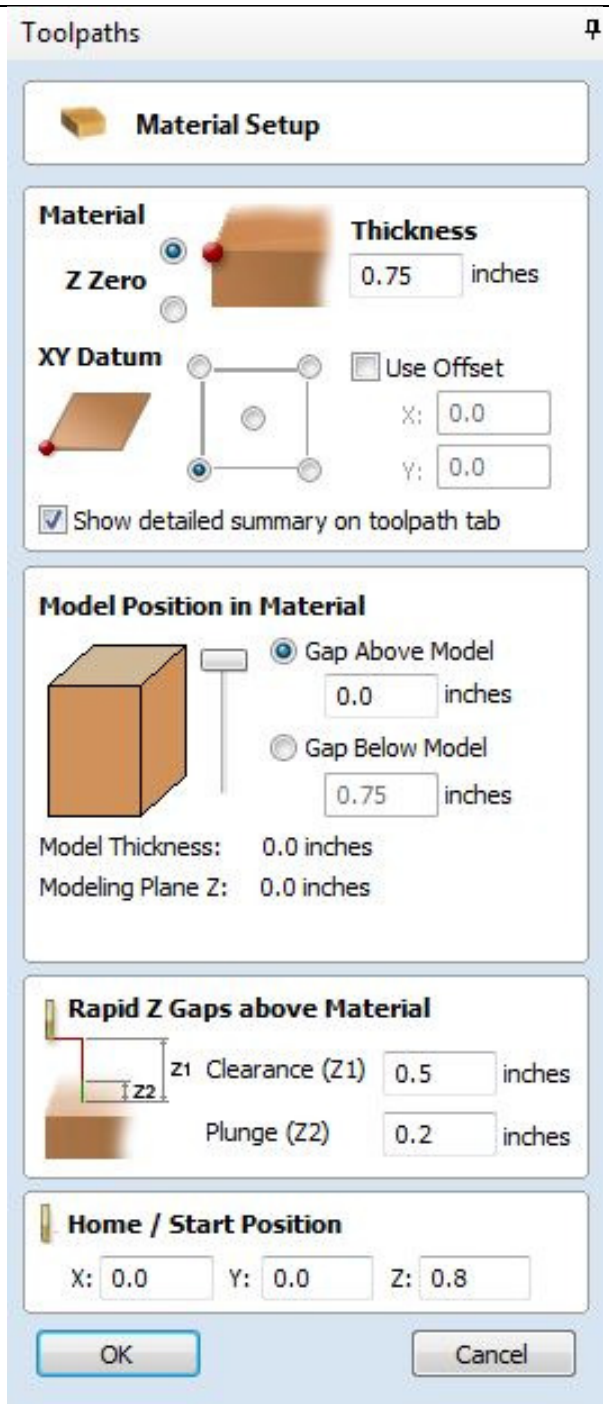
Project Design (CAD): This outline is not intended to cover the intricacies of designing a project using VCarve Pro or Aspire. For more detailed instruction on CAD design, please refer to the tutorials available in the course tutorial handout.

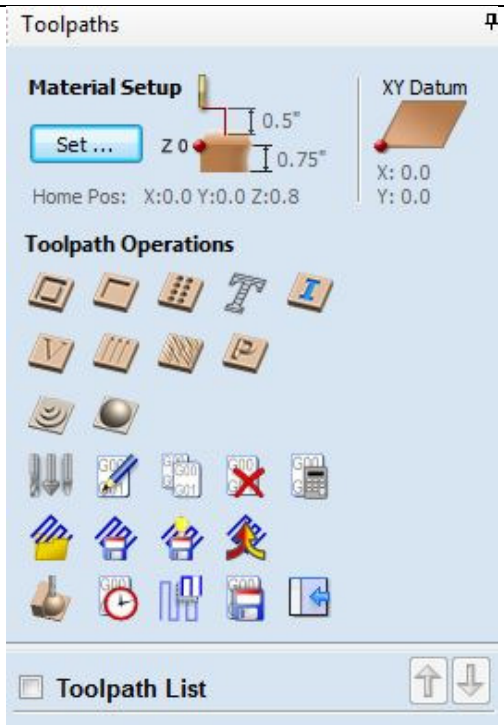
Introduction to Creating Toolpaths (CAM) in VCarve Pro / Aspire

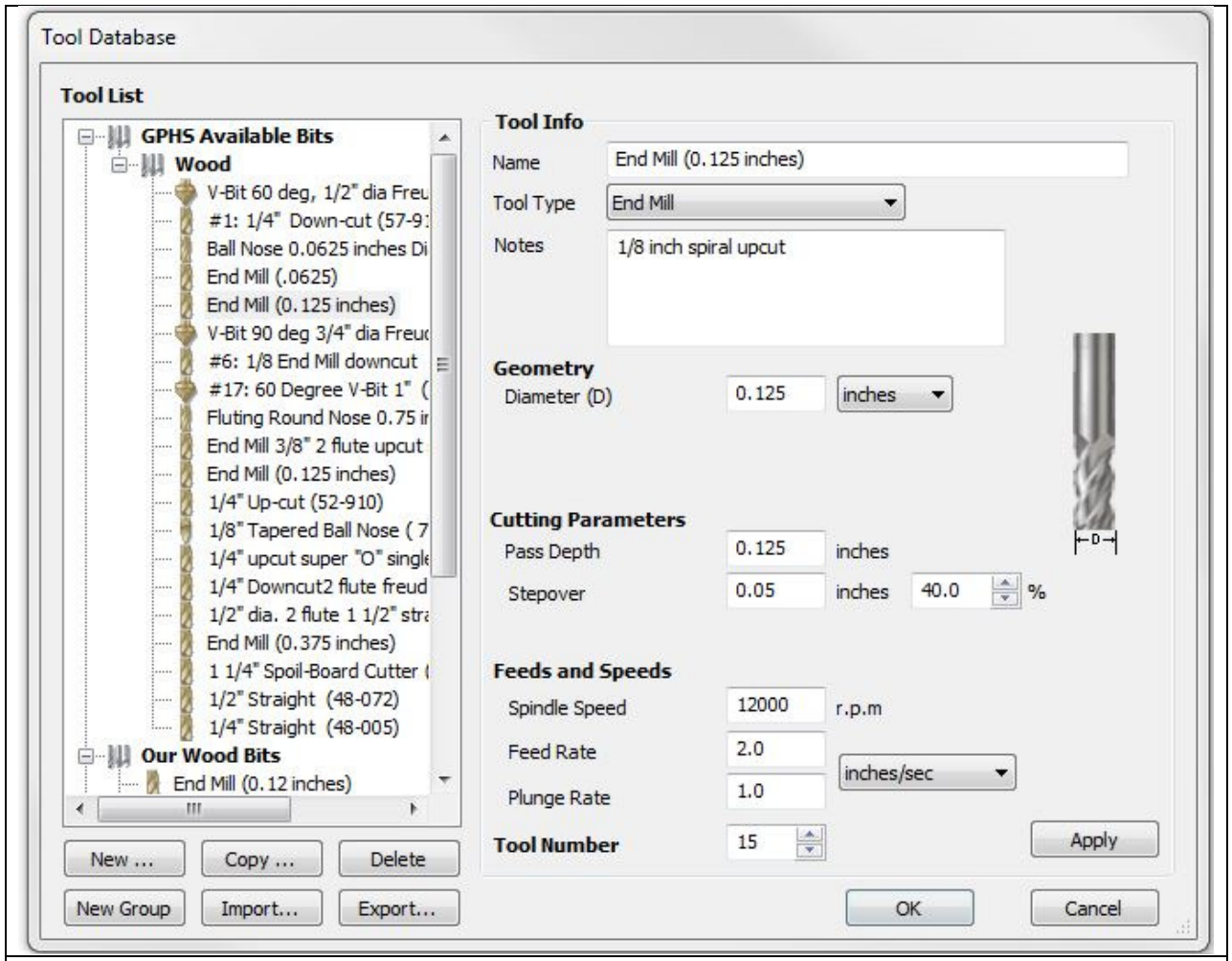
The first thing you need to do before you calculate any toolpaths is confirm your material setup and position, relative to how it will be set up on the machine itself.

Although you will have previously made some choices on this when you setup the job for the design layout, you will need to double check they are still correct and potentially make changes based on things that may have evolved as you completed the design. For instance if you set XY zero in the middle of the job for drawing, you may now decide to change this to one of the corners (typically lower left) for machining.

Once your material settings have been verified or edited, you are ready to start generating the Toolpaths. To do this you will run through this set of procedures.



Select the type of toolpath: Profile, VCarve, Pocket, etc. For example, a “VCarving” strategy can be used to engrave incised vector letters or a “Profile” strategy may be chosen to cut-out a vector shape. You will find good information on the different Toolpath types in the Help document and Reference Manual. Each one has different applications, there are many examples within the video tutorials that cover all the main uses and some specialized functions.	



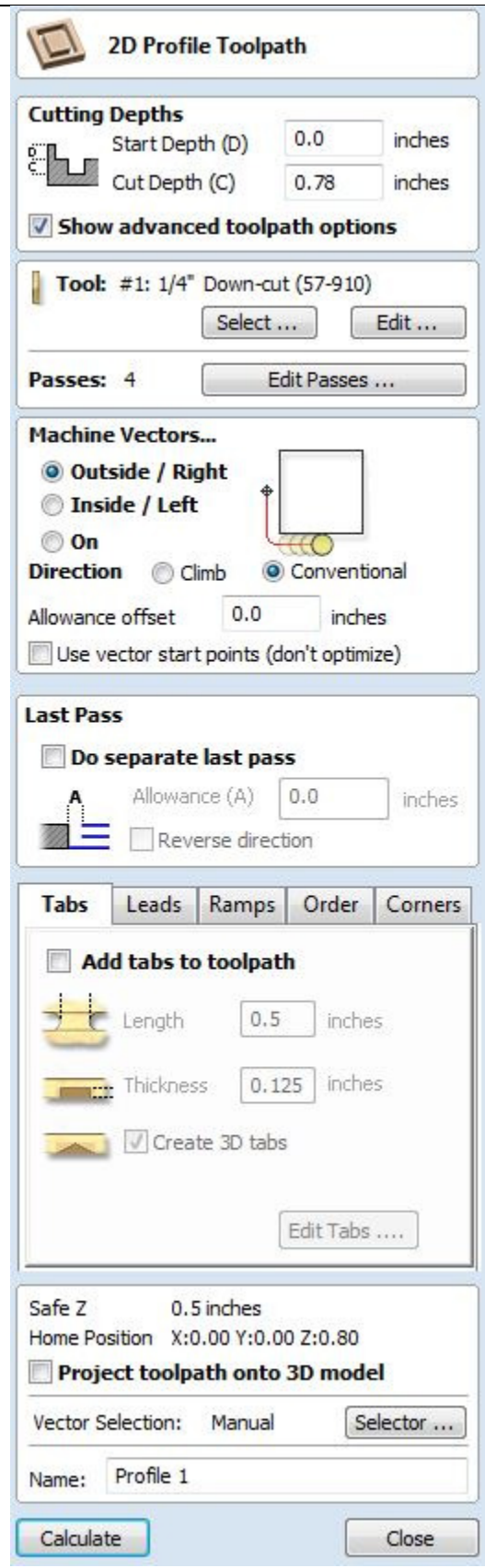
Select the correct router bit: Select a tool that will be best suited to machine the toolpath in the material you are using. Some toolpaths, by definition of what they do, are limited to the types of tools that can be assigned to them. For example a VCarve toolpath can only use a V-Bit or Ball Nose tool. For each tool selected, you can 'Edit' the settings for that bit to exactly match the requirements for that project. A list of tools, their type, size and appropriate settings are remembered by the software and accessed from the "Tool Database.

The specific tools that are available to us are listed under "GPHS Available Bits" and correspond to the numbered tool holder located at the machine.

Choose any modifying options for that specific toolpath: Each toolpath has options that can be set to customize it for a particular cut. These options vary and are dealt with in detail in the documentation and the tutorial videos.

Some toolpaths have very few modifiers and some have a lot of additional choices depending on the type of cut and how much control the user may require over it.
For instance you may want to add “Tabs” to a toolpath where you are cutting out a job using a “Profile” toolpath strategy. That would let you specify the size and position of small pieces of material that leave the cut object still partially attached to the original stock to hold it in place while it is being machined. This is just one example of the way a toolpath can be adjusted.

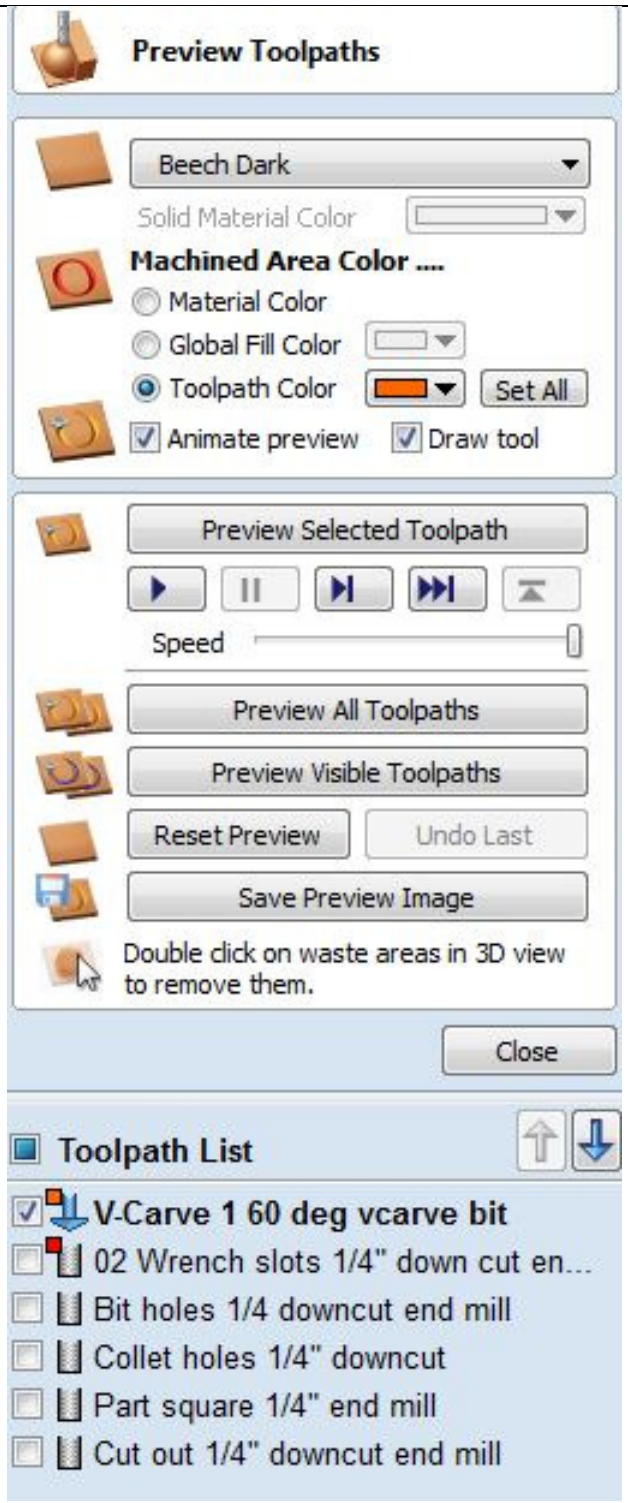
As a CNC is flexible enough to use in many different applications and to cut a range of material it’s important to have these options available when calculating different types of cut.



Calculate the toolpath and then preview: A powerful feature offered in all the Vectric programs is the Toolpath Preview; this allows you to accurately simulate the result of the toolpath on your computer screen in a virtual 3D piece of material.

Creating a simulation like this lets you check if the toolpath is correct based on the tool and settings you have chosen. If it does not look right in the preview then it will not be right when you cut it. This feature helps you avoid costly mistakes, a few seconds at the computer can save hours at the machine, plus prevent damage, potential safety issues and a ruined project.

This function can also be used to generate realistic rendered images to show your customer exactly what the part will look like when it's machined or even to use as images in promotional material to demonstrate the capabilities of what you make.

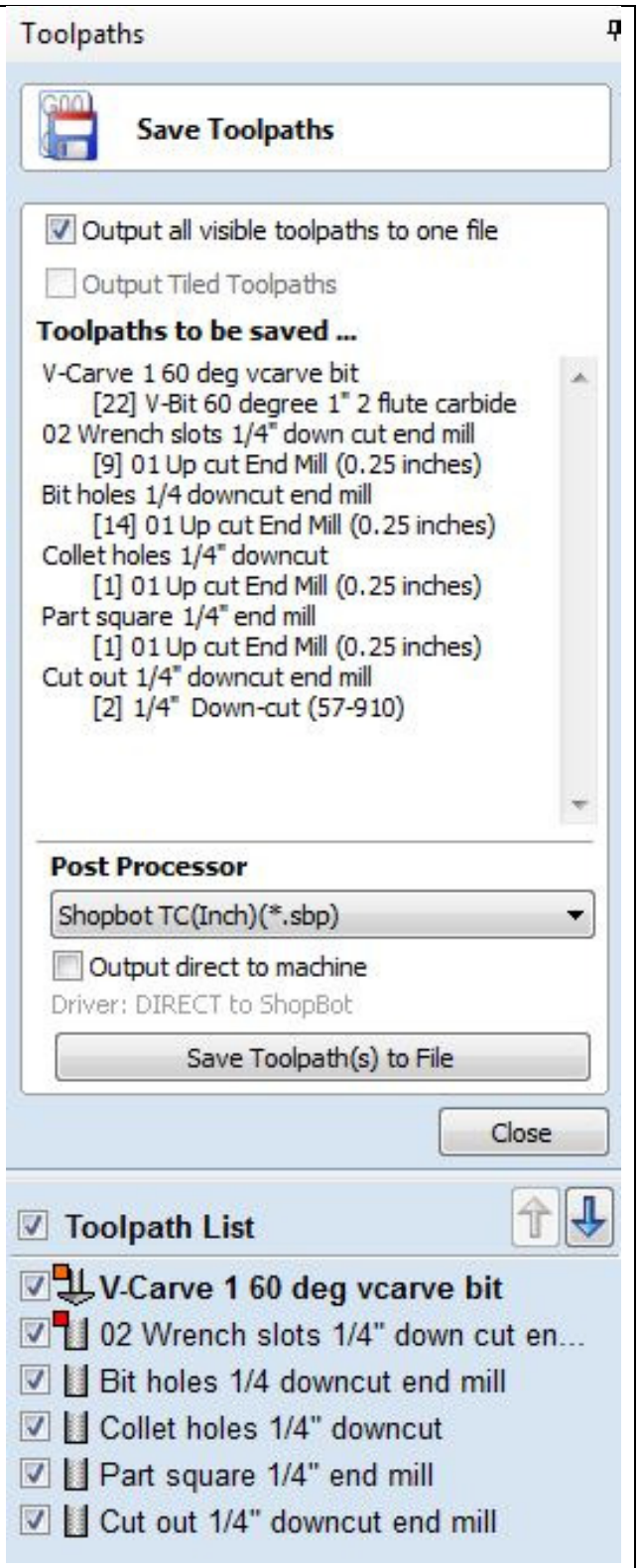


Post Processors; After you have created and calculated all the toolpaths and are happy with the Toolpath Preview, it is time to save them in the appropriate format for your CNC machine.

When you save your Toolpaths, you will be required to select what is referred to as a ***Post Processor*** from a list and this will determine the format that the Toolpath data will be saved in. You need to select the appropriate Post Processor from the list that matches your specific machine or the control software you use to drive your machine. This will create a file that has all the setup information and XY and Z coordinates for the toolpath formatted in a way that your CNC can understand and execute them as instructions to move.

Once you have selected the right Post Processor, the software will remember your choice so you only need to do this once.

Next you work through the list of toolpaths and save them as separate files, one for each toolpath. You may also choose to append multiple toolpaths together that use the same tool shape and size and save them as one file.



Section IV: Running a Part File on the CNC

Pre-Start

- Check the area around the machine for loose tools or material
- Check the machine for any red-tags / tag-outs and perform any required maintenence prior to starting

Start Up

- Turn on the PC and monitor. Log in and wait for alll processes to complete
- Turn on the power to the CNC and the spindle
- Open the Shopbot control software (Shopbot 3) and make sure it is communicating with the machine
- Move the spindle to a safe location and run a C5 routine (spindle warm-up routine) if this has not been performed within the last 4 hours

Load Material

- Place your material on the spoilboard and attach it using appropriate hold-down techniques

Load File and Tooling

- Insert your first cutting tool into the collet of the spindle
- Set your Z-Zero using the touch plate
- Set your X and Y axes zero points
- Load your file from the flash drive onto the desktop. Remove the flash drive from the PC
- Open your cut file in the control software

Run a 3D Offset

- After loading your cut-file into the software, run a 3D offset routine to be sure your zero points are properly set and your material is oriented correctly

Run your program file

- Do not leave the machine while it is running
- Do not place any part of your body inside the cutting envelope

Machine Shut Down

- Once your files have been run, remove your tooling from the spindle and put it in it's proper location
- Move the spindle to a safe location and remove your project
- Shut off the power to the machine and log out of the PC, shutting it down when you are done
- Clean up the machine and the surrounding area

Preparing the Machine: Getting the CNC machine ready to run involves several steps, not all of which are performed at the outset. The initial steps to follow when using the CNC are as follows;

Pre-Start:

- ✓ Check to make sure that there are no Lockout or Tagout devices on the machine or its power sources. If any part of the machine is unplugged (Computer, Spindle…) ***DO NOT TURN ON POWER OR OPERATE THE MACHINE IN ANY WAY!*** Notify your instructor before continuing.
- ✓ Make sure that the entire work area is clean and clear of any material, tools, fixtures, clamps, tooling or other items. The work area should be clear of any debris or material.
- ✓ Check the maintenance schedule / log to be sure that everything is up to date. If a maintenance item is due, notify your instructor before proceeding.
- ✓ Make sure that the area needed for gantry movement is clear of any obstructions.

Start-Up:

- ✓ Turn on the PC and the monitor. Log-on to the PC and wait for the start-up procedures to complete.
- ✓ Turn on the power to the CNC and to the spindle.
- ✓ Open the Shopbot 3 software. The software should recognize the CNC machine when you power up the CNC.
- ✓ Open the manual control keypad and move the gantry / spindle to a safe area for operation of the spindle. ***Make sure that any tooling and the collet are removed before moving to the next step.***
- ✓ In the Shopbot Command line (or using the drop down menu) run a C5 routine (spindle warm-up). This routine needs to be done if the spindle has not been run within the last 4 hours.

Load Material:

- ✓ You should already know how you intend to mount your material to the spoilboard or to any fixtures. Remember that if you are creating a part that requires a through-cut you need to have placed tabs in the toolpaths or have some means of securing individual parts to the work surface.
- ✓ Mount your material to the spoilboard securely. Keep orientation of grain and placement of any screws or clamps in mind. Always pre-drill (countersink or counterbore) your mounting screw holes prior to mounting fixtures or materials. Make sure everything is square to the axes if it is called for.

Load Cut –File and Tooling:

- ✓ Move the gantry / spindle to an easily accessible location and properly install your cutting tool. Pay careful attention to how far you insert your tooling into the collet.

- ✓ Using the manual control keypad, move the spindle to a location over your material that you can use to set the Z-zero height.

- ✓ Install the grounding clamp to the spindle body or the collet and test for proper gounding by touching the zero plate to the tip of the cutting tool***. Input light #1 should light up*** when you make contact with the tool. If it doesn't, you are not grounded properly and you need to move the grounding clip to a different position.

- ✓ Place the Z-zero plate on the surface of your material hold it down tightly with your hand in a safe location and activate the Z-zero routine. The spindle will slowly move down and touch-off on the plate. It will then retract and touch-off a second time. The Z-zero height has now been set. ***You must repeat these Z-zeroing steps any time you change tooling!***

- ✓ Using the manual control keypad, set your X and Y axes zero location.

- ✓ Load your drawing and your toolpath files onto the desktop from your flash drive. Once they are both loaded, remove the flash drive from the USB port. ***Do not run a file directly from the flash drive!*** You need to load both the .crv file (your drawings) and the .sbp file when running a job. It is much easier to make changes to a project at the machine than it is to go back to another PC and alter things.

- ✓ Once you have your toolpaths (the .sbp file), open the code in the Shopbot editor and take a look at the code. You can often spot problems in the code before running a file.

Run a 3D Offset:

- ✓ Once everything is in place and looks acceptable, load the file into the program and do a dry run or "***3D Offset***" of your job. ***Keep the emergency shut-off switch in your hand in the event of problems***. This allows you to check that your X and Y axes home locations are correct and that you will not collide with any fixtures or hold downs. ***This must be done on every file you run!***

Run Your Program File:

- ✓ You can now run your cut file. Keep your hands and body away from any moving parts ***(stay out of the cutting envelope).*** Do not leave the machine while your file is running and keep the emergency shut off within reach. ***Monitor your job!!!***

Machine Shut Down:

- ✓ Once your file has finished, move the spindle / gantry to an easily accessible location and remove the tooling and collet. Blow out / clean the collet and place the tooling and collet in their proper place.

- ✓ Move the gantry / spindle to a safe location and remove your project and any fixtures that will no longer be needed.

- ✓ Shut all power off to the machine and log-off the PC. Power the PC and monitor down.

- ✓ Clean-up the machine and the surrounding area.

Section V: Introduction to CNC Programming / G-Code
Using the Line Editor
Making Changes to CNC Code

Post Processor Editing Guide / Using the Line Editor: The post processor is the section of the program that converts the XYZ coordinates for the tool moves into a format that is suitable for a particular router or machine tool. This information is often referred to as "G-Code".

OpenSBP® Language Part Files offer virtually unlimited control of the tool and the execution of cutting instructions. This document describes working with Part Files and how to use the additional OpenSBP® 'Program Statements' to help control or program how a Part File runs.

OpenSBP Part Files (.sbp);

What is in a file? The list of instructions in a part file can be just a list of Commands entered from the keyboard using the two key-stroke shortcuts. On each line of the list there is the two-letter Command, followed in most cases by parameters giving the details of the action.

A Part File can be a short and simple list that looks like the following:

```
JZ, .5
J2, 10, 10
MZ, -.25
M2, 20, 10
M2, 20, 5
M2, 10, 5
M2, 10, 10
JZ, .5
```

These instruct the tool to cut a rectangle. The JZ, .5 pulls up the bit to make sure it is not in the material when positioning. Next, the tool will Jog to the XY location 10, 10 (assuming the ShopBot is configured for Absolute Distance moves). Then the bit is moved down to -.25 (a 'M'ove is used to cut into material), to drill into the material.

There are then four cutting speed moves (M2's) that will define a 10 x 5 rectangle, with the starting point being in the upper left corner of the rectangle, and going around the rectangle in a clockwise direction (if using a 1/4in bit, this would create a cut-out that is actually 9.75 x 4.75 because the center of the bit would be following the specified path). An OpenSBP® Part file can also, however, contain Programming Commands that can only appear in Part files and can't be entered at the keyboard...that's what the rest of this manual is about.

Using the Editor: The ShopBot reads files in 'text' format. That means that the file can be edited with any text editor or word-processor. However, the SbEdit text editor that comes with the ShopBot software is the recommended editor. Type [FE] to open the editor or type [FN] to open a new blank part file. SbEdit can be started from the Windows 'Start' menu or by double-clicking on any ShopBot Part File. If using a different word-processor, make certain that a

modified Part File is re-saved in generic 'text' format and not a special format. The file name extension for a ShopBot Part file is '.sbp'. Other ShopBot related files have file extensions that start with '.sb_' and end with another letter for descriptive convenience. The file extensions '.ini', '.opt', and '.sbd' have special purposes and are discussed elsewhere, but these too are text files in ShopBot Part File format.

The Part File Layout; A Part File can consist of a single line or a million lines, but can have only one Command per line. Upper and lower case letters are generally used for preference. Lines can be indented and skipped to add white space to help structure files and make them easy to read.

Comments can also be inserted to document notes for a particular file. To use a comment, type an ' (apostrophe) and everything after it in the line will be ignored when the file is read by ShopBot. Details for the use of remarks and comments can be found under "REM" in the Reference section. Also note the very useful special case of a comment after a "PAUSE" that provides a quick message to the tool user.

To really appreciate the possibilities for file layout, look at the sample files included in the c:\SbParts folder installed with the SB3 software. These sample files illustrate various formatting and organizational techniques. Sample files start with the name 'S_' (for 'Sample').

Note that when using ShopBot Commands, if any parameter follows the Command, the two letters of the Command are always followed by either a comma or at least one space. See the following example (note use of comments):

M2, 3, 4.2 'ok ... using comma as Command separator
M2 3, 4.2 'ok ... using space as Command separator
M23, 4.2 'WRONG ... no separator
M2,, 4.2 'This commands moves Y to 4.2 because first comma is the separator (no X value)
M2 , 4.2 'This commands moves Y to 4.2 because space is the separator (no X value)
M2, 4.2 'This commands moves X to 4.2 because comma is the separator (thus no Y value)
M2 4.2 'This commands also moves X to 4.2 (no Y value)

During execution of a Part File, information about the files being opened and run are displayed in the main window. This display identifies which line in each file is currently being read or executed.

Introducing Part File Programming Statements; Beyond the ShopBot execution Commands, there is an additional set of commands that can be used from within Part Files. These commands are called Program Statements. Programming Statements add flexibility to files. When used in ShopBot Part files, Program Statements amount to a mini programming language and are the primary topic of this Handbook.

The ShopBot Program Statements are modeled after similar functions in the BASIC computer programming language. In general, Program Statements provide enhanced ways to give the tool instructions.

Nesting or Embedding Part Files; A Part File can call up and execute another Part File - simply use the [FP] Command in the first file start the second file. When the second file finishes execution, action returns to the next line of the first. The process of starting one file with another is called "nesting" or "embedding" files. Typically, a 'master' Part File will call up various components that will be cut from the piece of material on the table. This master Part File can position the tool for

cutting the components and thus manage the layout of the project. An example of placing a single part in multiple locations is described in the sample code below. Use the ‘offset’ function in the [FP] Command to cause the file to be cut from the location that the master file moves the tool to. Nesting can be used to place different parts of a project in their correct location.

```
‘Start of master Part File ALLPARTS.SBP
  M2, 10, 4 ‘move to location to start first part
    FP, MYPART.SBP , , , , , 2 ‘execute with 2D offset
  M2, 20, 4 ‘move to start of second part
    FP, MYPART.SBP , , , , , 2 ‘execute with 2D offset
  M2, 30, 4 ‘... and so on
    FP, MYPART.SBP , , , , , 2
  M2, 40, 4
    FP, MYPART.SBP , , , , , 2
```

Nesting of files inside other files can be repeated multiple times up to eight levels deep. This means a part can be pulled up, and then within each part call up a repeated procedure such as a drilling and countersinking routine can be included. This process is called a sub-routine, sub-program, or procedure. Sub-routines are useful for organization and efficiency, and more generally provide a way to ‘structure’ Part File work. Subroutines can be used with a variety of master Part Files. For example, a drilling/countersinking routine can used for many different purposes including making the table-tops for ShopBots. This counter-sinking routine, is for just one hole, saved as a Part File. It is then used as a sub-routine within any file in which drilling or countersinking is necessary.

Math, User Variables (&), and System Variables (%);When creating a Part File, variables can specify parameters just like entering parameters at the main menu. Most mathematical functions are supported. When in doubt, try them using the ShopBot Calculator first. Variables can also be used in Programming Statements.

Using Variables; It is sometimes useful to indicate a value as a ‘variable’ rather than as a fixed number. A ‘variable’ is just a word or symbol used as a temporary stand-in for a number. For example, to cut out a fancy grating for the front of a series of cabinets, the grating might sometimes be cut from 3/8” material and sometimes in 1/2” material. There are dozens of Z plunges in the file and rewriting all of these plunge values for different depths of material would be time consuming. One way to approach this is to use a variable for the Z depth for the cut. Variables in SB3 are always designated with an ‘&’ (ampersand) as the first character and are created like: &Zdepth = -0.52 . Creating a variable like this at the start of a file assigns the value of -0.52 to every instance of the &Zdepth variable that follows in the file. So Z plunges would be written like: MZ, &Zdepth . Then to change the plunge depth in a file - even though there may be numerous plunges - it is only necessary to change the one variable definition at the top of the file.

New user variables can also be defined with mathematical expressions.

```
&MyVariable = 5.0
or
&MyNewVariable = &MyVariable * 2.31-2
```

Remember that variables must have an & as the first character. Letter case does not matter during ShopBot processing (internally all variables are converted to upper case as they are

processed). This means any lettering in a variable can be input in either upper or lower case. Letters and underscores are fine, but don't use special characters in variable names that might be confused with math symbols or Windows folder designations (e.g. no "\" or "*" or ":").

Programming variables are often defined in terms of specific 'types' based on what kind of numbers or character they are. For example, integer variables, string variables, floating point variables, etc… However, ShopBot variables are a generic variable type that is handled behind the scenes as a string variable or floating point number depending on what is appropriate in the situation (similar to 'variant' data type for those familiar with Visual Basic).

When mathematical operations are carried out, if a variable can be made into a number it is handled as a single precision floating point (7-8 digits precision). If using variables for counters and test for the end of the loop with an IF test, then test with an ">" or "<" rather than an "=" because the variable may not be stored as precisely the expected number (i.e. a number that you expect to be "1" might actually be "1.000001").

Strings variables are not case sensitive, and can be entered with or without quotation marks. For example:

&MyString = test string 'string variable without quotes
&MyString = "test string" 'or, string variable with quotes

Strings can be used for display, writing to files and some special functions. In most cases, defining a string in either manner will work fine. Strings can be combined (or "concatenated") using the & character. If:

&firstname = Bill

Then if there is a line that reads:

&newstring = "My name is " & &firstname

the value of &newstring would be "My name is Bill".

Part File variables are "persistent" and "global", retaining their value until that value is changed or the ShopBot software is shut down and are available to any Part File that is run. This can be very handy for occasions using the same value for all the files in a session like possibly &Zup for your safe Z-axis position. This also means, though, that if using a variable from a previous file, ensure it's value is known. The safest way to do this is to define all variables at the beginning of a file by giving it a value, a process referred to as "initializing." This also conveniently documents the list of variables that are used in any particular file.

System Variables; In addition to user variables, certain ShopBot System Variables can be accessed. These are variables maintained by the software and related to tool setup and operation. These values are accessed by a "%" (percent sign) followed by a number in parentheses (e.g., %(5)) and can be inserted where any value or parameter could normally be inserted in a file. Any system variable that involves a distance value (like the X-axis location) will be in the current user units.

Here's an example of getting the current Y location (in the current user units; inches or mm):

&Y_now = %(2)

ShopBot System Variables are frequently expanded and updated. The current System Variable list is provided at the end of this manual and updates can be found in the “Program Files\ShopBot\Developer Tools” folder after installing the ShopBot software.

Types of Programming Statements; All ShopBot Commands can be used in a Part File. In addition, a number of Program Statements are available to use in enhancing the functionality of Part Files. These programming statements are modeled after statements in the BASIC programming language. BASIC-like functions and control statements will be continuously added to ShopBot’s capabilities. Here is a brief description of current programming statements by category. The subsequent section provides a detailed alphabetical reference to programming commands.

Control / **Flow Statements;** These are statements that direct the action during the execution of a Part File, moving it from one place to another and stopping and starting it.

GOTO >sends action elsewhere
GOSUB ... RETURN >sends action to subroutine and returns it
PAUSE >pauses a file
END >ends a file
EXIT SHOPBOT >ends a file and exits ShopBot

Conditional Branching Statements; These are statements that perform a logical test or check for an Input Switch event and then appropriately re-direct the action in the Part File if something has happened.

IF ... THEN >logical IF test
ON INPUT >handles detection of an Input Switch change

Input / **Output Statements;** Control what is displayed or entered on the File Display line at the bottom of the the screen during the running of a Part File.

INPUT >get input from user/keyboard
PRINT >display a message or variable on the screen
‘ (or) REM >comment in file that can optionally be displayed
MSGBOX >display a Windows style message box
WARNING OFF >turn off warning light display
PLAY >play a sound or audio recording to alert the Operator (or just for fun)

File Statements; Statements used to manage input and output to files that are opened within and during the action of a Part File.

OPEN ... CLOSE >open or close a specified file
INPUT #_ >get information from the file
WRITE #_ >put information in the file

Reference: Part File Programming Control Statements; For consistency and clarity, upper case letters are used for Programming Statements and ShopBot Commands. However, in the actual processing of a Part File, case is ignored. In the following section, programming statements are in upper case letters,variable or parameters are in italics, and variable or parameter choices are separated by '|'s, and optional functions are in braces ('{'s).

CLOSE {#(openfilenumber)}

Closes a file that has previously been opened with OPEN statement. If used without the (#openfilenumber)

Close files when finished using them. It is particularly important when writing to a file to make sure all data is saved in case of a crash. All user files are automatically closed when Part Files finish.

Note:Read about the OPEN statement further down this list.

END
Terminates the execution of an individual Part File. It is sometimes useful in a file to place an explicit END statement to stop processing and make certain that processing can not move into a section of the Part File such as a section that should only be entered with a GOTO or GOSUB. The use of an END statement can also add clarity to the flow of a program. However, since Part Files are sequential and execute from the first line to the last (unless redirected by a GOSUB or GOTO statement), an END statement is not required in a Part File as the action of the file will terminate on execution of the last statement. Note that an END statement in a nested Part File that is being used as a sub-routine will end the execution of that File and return control to the Command Prompt or the higher level Part File that started or called it.

ENDALL
Terminates execution of all Part Files. In the case of nested files, ENDALL will end all files above and below the file in which the ENDALL statement occurs.

EXIT SHOPBOT
This statement will terminate the Part File and immediately exit ShopBot software. This statement is useful for occasions when the ShopBot software is started by another program and has started a file in ShopBot by passing the Part File name when SB.EXE was called (see Developer's section below for details). When the ShopBot task is completed and the EXIT SHOPBOT Program Command encountered, the computer will leave the ShopBot Program and return to action in the calling program.

GOSUB label
This statement invokes a subroutine and executes the file to shift to the first line under the designated label where label: is the name of a block of instruction and is positioned as the first line. Note that the label is actually used as the name of the block, and is followed by a colon, but it is not followed by a colon in the GOSUB statement. The block of code is terminated by the RETURN instruction, which causes execution to return to the line immediately below the GOSUB statement.

GOTO label
Causes execution of the file to shift to the first line under the designated label where label: is the name for a position elsewhere in the file. Note that where the label is actually used as the name of the new position it is followed by a colon, but not in the GOSUB statement.

IF statement to evaluate THEN ShopBot command | variable assignment | GOTO label
This control statement does a logical statement test, and, if the statement is "true," it executes a ShopBot command, assigns a value to a variable, or branches to a label - whichever action is specified after THEN. Statements to evaluate are in the form of "IF &Count = 20 THEN ...." Moreover, the logical comparisons [= , >, < , AND, OR, NOT] all can be used in the statement along with user variables, system variables, or computed values. Test statements using the ShopBot Calculator [UU] or F10 (-1 = "true"; 0 = "false"). At this point, various "action" choices for the true case are available, including branching to another section of the file. Play around with it to get a feel for how it controls the flow of program execution. Also, have a look how IF's are used in various sample files (the ones whose filenames begin with S_ in the ShopBot Folder).

Note: An IF statement may contain only one logical test. For example, it is not possible to test IF &varA => &varB THEN ... rather, for one logical condition can be tested at a time. For example, IF &varA > &varB THEN ...
See the section above on string variables for an explanation of how explicit quotes must be used
in IF tests of strings.

INPUT "Optional Text Message" &variablename { ,&variablename, &variablename, etc... up to 10 variables}

Gets input from a user and assigns it to a variable. The "Optional Text Message" is any text in quotes (without commas) and can let the operator know what sort of information is being requested. &variablename is a ShopBot user variable that will store that input, beginning with an ampersand ("&"). The INPUT statement calls up a message box that provides OK and Cancel buttons. OK places the value in the text box into the variable, and Cancel removes the box and ends the program. If the variable exists, its value is overwritten; if it does not exist, it is created. The user must separate input values with commas if more than one value is being input. If the text box is
empty and only one variable is expected, it treats that variable as an empty string and continues. If more than one variable is expected, one or more missing variables is treated as an error, and an empty box is handled as a Cancel. In both cases the Program is terminated.

There are two special cases for INPUT:

1. Including a variable in the Text Message of INPUT. If it is desirable, a ShopBot User Variable can be used to display dynamic information within the Text Message display. This is accomplished by putting a semicolon immediately after the end of the Text Message. The INPUT statement will
have the form:

INPUT “You Entered - “ & &variable & “ Is this Correct? Y or N”; &test1

In this case, the returned variable will be the response to the Y or N question. The &variable needs to be defined prior to the INPUT call to work correctly.

2. Including a default value in the INPUT Box. A default value can be placed in the input box by defining it before the INPUT call and using a colon in front of the default variable name, which will be the first variable in the list, occurring just before the other variables to be gotten from the user. This INPUT statement takes the form:

INPUT “Displaying default value - “: &default, &newvar

In both cases, as with the standard INPUT, there can be up to 10 user variables in the list of those being collected. The two special cases can be combined, using the semicolon call of the dynamic variable processing first.

INPUT #(openfilenumber), &variablename { ,&variablename, &variablename, etc }

Statement for reading data from a file that has been opened from within the ShopBot Part file with the OPEN for INPUT statement and having been assigned an openfilenumber. Variables are assigned as they are assigned with the INPUT Command. When all variables have been read, an open file should be closed with the CLOSE statement.

label:
A label provides a marker in the file that serves as both a section title and, more often, an entry point for a GOTO or GOSUB instruction. Labels can be any word or single group of characters, but labels must exist on a line all by themselves and have a colon (“:”) as their last character. See GOTO and GOSUB.

MSGBOX (body text, button type, title text)

The MSGBOX command creates a Windows style message box that can be customized, and places the value of the button that was clicked into a user variable named &msganswer. The body text is the text that appears in the main part of the message box and can be used to ask the User questions or give them information without them having to type in an answer the way they
would with the INPUT statement. Strings with variables can also be combined, either system or user, to make a more descriptive message, using the rules for combining strings as mentioned in the “Strings” section. The only restriction for the body text is that it can not contain a comma.
The second message box parameter is the Button type. This provides a lot of options to customize the look and action of the messagebox. Defaults include...

Style Parameter	Display
0 (or “OKOnly”)	OK button only
1 (or “OKCancel”)	OK and Cancel buttons
2 (or “AbortRetryIgnore”)	Abort, Retry, and Ignore
3 (or YesNoCancel”)	Yes, No, and Cancel buttons
4 (or “YesNo”)	Yes and No buttons
5 (or “RetryCancel”)	Retry and Cancel buttons
16 (or “Critical”)	Critical Message icon with OK button only
32 (or “Question”)	Warning Question icon with OK button only
48 (or “Exclamation”)	Warning Message icon with OK button only
64 (or “Information”)	Information Message icon with OK button only
256	Second button is default
512	Third button is default
768	Fourth button is default

These numbers (or the text options for the first 10) can be typed in as the second parameter in the command just the way they are to get the default message boxes. The table above shows the parameters in three groups. The first six define the button choices that the operator will see, the next
four define the icon that will appear in the box, and the last three define which button has focus. Adding the values of items from each group enables the ability to create custom boxes.

For instance, if typing “276” as the second parameter in the command (4+16+256), it will create a message box with Yes and No buttons (the value of 4), a ‘critical” icon (the value of 16), and the second button (No) selected by default (the value of 256). Any incorrect value will default to 0, a plain OK Only box.

The third parameter is the text that appears in the title block at the top of the message box. Keep this short and don’t include any “,” in the text.So if this command is placed in the file...

MSGBOX (Do you want to open the keypad?, YesNo, Start Keypad?)

...then this box would appear...

When a button is clicked by the User, the value of that button is saved.

If the command was..

MSGBOX (You can not cut plywood that thick with that size bit!,16,Thickness problem!)

…the message box would look like this…

When a button is clicked in the message box, the "value" of that button is saved as a string in a variable named "&msganswer" to use it in an IF test to act on the operator's answer. Possible values for that variable are…

- OK
- Cancel
- Abort
- Retry
- Ignore
- Yes
- No

…depending on the buttons that were available in the message box.

ON INPUT(switch# , 0 | 1 | 2 | 3) ShopBot command | variable assignment | GOTO label

This control statement defines what is to be done if action occurs for any Input Switch. Place in code before location where input detection is desired. Define by Input Switch number and switch condition whose occurence is to be detected; where 1 = ON and 0 = OFF. Then provide simple instruction such as GOTO (a label) or setting a variable. To clear the interrupt handling condition and instruction, use the same defining condition for the switch, but leave the instructions blank.

After a switch interrupt has occurred, the interrupt is automatically cleared. This prevents any unexpected reactivation (e.g. bounce) after the interrupt. To use the ON INPUT interrupt again, reset it with the same ON INPUT command.

A change at any of the switches will cause the ShopBot software to look for instructions for handling that particular change or event (The state for all eight Input and Output Switches is displayed in “lights” below the location readouts in the main red Location panel). For example, if Input Switch #2 had just been activated, and “ON INPUT(2,1) VA, 0.0” was set as the desired response (for example, the closure of a limit switch), then the X-axis location would be reset to 0 when the switch was triggered. In other words, ON INPUT( ) sets up the instructions for what to do “in the event of” a Switch closure (or opening). Switch # is the number of the input switch that the instructions apply to. 1 or 0 indicates whether the instructions are coming on (1) or going off (0).

The instruction identifying the desired response can consist of a ShopBot command, the assigning of a value to a variable, or branching to a label (The actions are similar to those possible after an IF ... THEN statement). The ON INPUT( ) statement should be placed near the beginning of the file or before a procedure segment. ON INPUT( ) itself does not cause execution of the instruction; it only establishes what to do if an Input Switch occurs. Note, too, that the action is not triggered unless a change in condition actually happens. The event handling specified in an ON INPUT( ) statement is active only for the duration of the file execution. When control is returned to the main menu, the specified ON INPUT( ) action is no longer active (although activation of the switches still will be indicated on the panel if the switches are enabled).

If a move is being executed at the time of the ON INPUT interrupt, motion will stop instantly (see the zzero.sbp routine for a good example). If a gradual, ramped stop is preferred, use the following switch conditions numbers, 3 = detect Input Switch coming ON and execute a ramped stop before continuing with instruction, 2 = detect Input Switch going OFF and execute a ramped stop before continuing with instruction.

OPEN “path&filename” FOR {INPUT/OUTPUT/APPEND} AS #{number}

ShopBot Files are sequential access text files and use standard BASIC syntax to open a user file for various purposes. If the ShopBot file is going to be written to the file, it should be opened FOR OUTPUT. If only reading data from it, it should be opened FOR INPUT. If adding information to an existing file without deleting it’s contents, open it FOR APPEND. If opening a file FOR OUTPUT and the file doesn’t already exist, then the ShopBot will create it - but if the file does exist, the data in it will be overwritten. If preserving the data in the file, either use another name or add data to the end using the FOR APPEND option.

The file number is the one being assigned to the file being written to and is used by the WRITE and PRINT statements to identify which file to work with. Up to nine open files are permitted (number = 1 to 9).

The path&filename can be a variable or specified file name. For path&filename, the path can be relative as in…

OPEN “myfile.sbp” FOR OUTPUT as #1

…which will look for “myfile.sbp” in the current part file folder, or …

OPEN “myfolder\myfile.sbp” FOR OUTPUT as #1

…which will look in the current part file folder for a subfolder named “myfolder” and then look for “myfile.sbp”. The full path to the file can also be used such as…

OPEN “C:\myfolder\myfile.sbp” FOR OUTPUT as #1

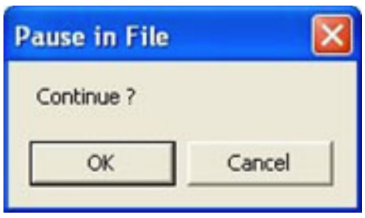
It’s important to remember that any file that was opened with the OPEN statement should be closed when finished using the CLOSE statement.

PAUSE {seconds; or, UNTIL Input switch#, state}

Creates a pause in the execution of a Part File. This Programming Instruction allows the operator to put a brief pause or a stop in the execution of a Part File.

PAUSE #. The number after the PAUSE is the duration of the pause in seconds and should be accurate to about 1/100th of a second. For example, programming a PAUSE 3 will cause a 3 second pause in the action and then the file execution will continue. If a comment line is inserted with an apostrophe just before the pause, that line will be displayed during the time the PAUSE is in effect.

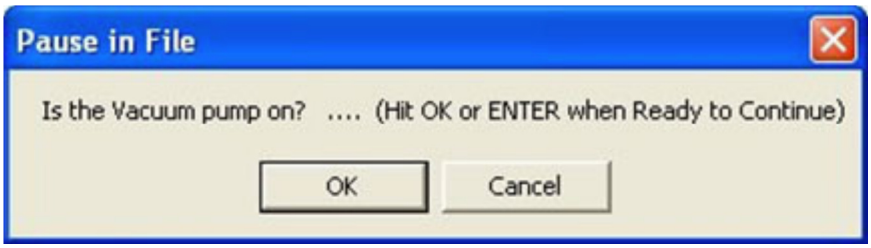
PAUSE. A PAUSE statement on a line without a number after it will cause the file to stop indefinitely, and a message box will appear with a choice of two keys … “OK” to continue, or “Cancel” to end the Part File. If the previous line is NOT a comment, the box will display “Continue?”



If the full line preceding the pause is a comment (‘), this comment line will be displayed in the box. For example, if making sure that a vacuum holddown pump is turned on before the file starts cutting, the following might be added…

```
‘ Is the Vacuum Pump on? .... (Hit OK or ENTER when Ready to Continue)
PAUSE
```

When the program gets to this place in the file, a message will appear to check the vacuum pump, and the ShopBot would stop until the OK button is clicked or the ENTER key is hit.



To just notify the operator about something in the file, without the operator having to act on it:

```
'Finished the first Cut-Out, Moving on to the Second
 PAUSE 5
```

In this case, the tool would stop for five seconds while displaying the message, then automatically continue to the next task.

PAUSE UNTIL works just like PAUSE (without a time) except that an Input Switch can be used as the equivalent of the OK input button. For example, programming **PAUSE UNTIL 5, 1** will PAUSE until the OK button is entered or the #5 Input Switch is closed. Any message in a preceding comment line will be displayed as with the simple **PAUSE** instruction. Use a 0 to detect an Input Switch opening, or 1 to detect a Switch closing.

By default, when motion is initiated after an Input Switch PAUSE, there will be a warning and short wait, just like at the beginning of a file to provide a safety interval. This can be eliminated by turning the wait off with **[SW]** before the call to function, then restoring it. See the sample file in the SbParts folder (sample_for_switch.sbp) for an illustration.

PLAY path&wavfilename

Plays a .wav sound file. The statement will start the execution of the sound and immediately proceed to the next instruction... to not proceed until after the sound finishes, add an appropriate length PAUSE after the PLAY statement. In the dev folder under ShopBot there is a helpful program called "wavlength.exe" that will help select and time sound files.

```
PLAY path&wavfilename
PAUSE time(sec)
```

For instance, to play the Windows "Tada" sound somewhere in the file and continue running while the sound was playing, add...

```
PLAY C:\WINDOWS\Media\tada.wav
```

...to the file. To stop the file while the sound played, add...

```
PLAY C:\WINDOWS\Media\tada.wav
PAUSE 2
```

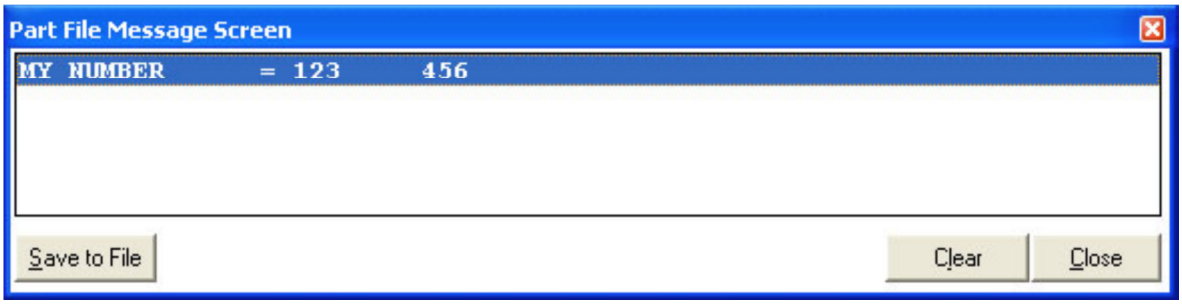
PRINT {variable or quote delimited text, variable or quote delimited text, etc...}

The PRINT Statement will display a variable or text and forces the PartFile Message Window to Display. If the first separator is a comma, this will be the default item separator and will cause each item in the line to be separated by a tab (five spaces). A semicolon as the last character will suppress the line feed, but if a comma is used first, a semicolon in any other position will raise an error.

If the first separator is a semicolon, this will be the default item separator but will produce no separation in the displayed item unless the separation was added as spaces as part of the print string. A semicolon as the last character will suppress the line feed. A comma later on in the parameters will be treated as an item to display. For example:

&firstvar = “My number”
PRINT &firstvar, “ = 123,456 “

with a comma after “&firstvar” to separate the parameters will be split at the commas and look like this...



With a semicolon separating the parameters like this...

&firstvar = “My number”
PRINT &firstvar; “ = 123,456 “

it will look like this...

If a variable is used in a PRINT statement that hasn’t been given a value then the variable name will appear in its place in the message screen.

The location and size of the message screen (measured in twips) can be specified using the [VD] command, either from the keyboard or within a part file. It can also be dragged to a new location and resized on the screen with the mouse and the new location and size will be remembered. To keep the Message Window from displaying, turn it off by setting the “Show Comments” setting to 0 in the [VD] command..

REM or more easily just, ‘

Indicates a Remark or Comment line. Use either the letters REM or an apostrophe (‘), but the (‘) is preferred. The text in a comment is not treated as a command and is ignored during the running/ cutting/execution of the file, even if the comment text contains one of the Shopbot Commands.

While creating Part Files, it is very helpful to make notes in the file. This is a process that programmers call ‘documenting’. To make a note in a file, type in an apostrophe (‘) and everything that follows it on that line will be ignored by the ShopBot software when the file is executed. Put in the apostrophe and make a note about what a particular line, or portion of

the file, is supposed to be doing. To make several lines of comments, put an apostrophe at the beginning of each line. A full-line comment is displayed at the bottom of the screen when a file is executing, so this is an easy way to remember things while the file is operating (see the 'PAUSE' instruction to learn how to make this display remain visible).

```
'This is an example of documenting a file
'You can use an apostrophe (or a REM) at start of a line
REM You can use an REM (or an apostrophe) at start of a line
'This file is to create a rectangle (written by me, 1/1/2000)
' ... none of the above effects the cutting of your file
' ... the line below is left blank for legibility

JZ, .5 'this line pulls the bit up
J2, 10, 10 'you can add a comment to a line.
CR, 10, 5, , , , -.25, , , 1
   'You can indent lines or comments and add blank lines
'The End
```

RETURN

Causes Part File execution to return to the line that is immediately below the GOSUB Statement that initiated action in a sub-routine. If subroutines are placed at the bottom of a Part File, put an END statement before them to make sure that execution of the Part File does not inadvertently flow into the sub-routine.

SHELL, "path and program and rest of command line (as single string variable or in quotes)", {display option}, {async or sync (use words ASYNC or SYNC)}

This Programming Statement starts an external program. First parameter gives the file to start (including path), followed as part of the same string with any command line parameters/options. These parameters are those specified by the program being called. They can be compiled into the string as ShopBot variables using normal string concatenation.

Display options are: 0 - hidden and focus hidden (use caution - this is a good way to lose control of the computer); 1 - normal focus and size (def); 2 - minimized with focus; 3 - maximized with focus; 4 - normal size, focus stays on ShopBot; 6 - minimized, focus stays on ShopBot.

ASYNC (def) or SYNC mode:

In normal ASYNC mode, the ShopBot part file will continue to execute after the outside file has been called.

In SYNC mode, ShopBot Part File will stop execution while called program executes and control will not be returned to ShopBot until the outside program is closed.

Examples:

```
SHELL, "sbedit.exe c:\sbparts\sample_shopbot_logo.sbp"

SHELL, "C:\windows\notepad.exe", 1, SYNC

&filepath = "c:\sbparts\sample_shopbot_logo.sbp"
```

```
&run_this = “sbedit.exe” & &filepath
SHELL, &run_this, 1, ASYNC
```

(Calling a Virtual Tool with variable:)

```
&speed = 11500
TR, &speed
```

WARNING OFF

Turns off the flashing warning display when it would be distracting, as it might be during execution of sections of a file that involve no movement. Any movement command turns the warning display back on.

WRITE #(openfilenumber)

Statement for writing data to a text file that has been opened from within the ShopBot Part file using the OPEN FOR OUTPUT statement and having been assigned an openfilenumber. An open file should be closed with the CLOSE statement.

There are two options for the WRITE# statement…separating the items with commas or with semicolons. See PRINT statement for examples of how this works.

WRITE #(openfilenumber), (data list, items separated by comma) ... This is the standard way to use the Write statement. Separate each data item with a comma and they will be evaluated and separated by commas in the output file.

WRITE #(openfilenumber); (data list; items separated by semicolons) ... This is an alternative Write function in which all spacing characters are suppressed except those explicitly supplied in data list. This version of the Statement provides the most explicit control of punctuation in the output file.

Notes: In both versions, a semicolon as the last character will suppress the line feed in the file being written to so that subsequent data will be added to the same line.

To generate **&myVar = 23.5** in the output file, use:

```
&someVar = 23.5
Write #1; “&myVar = “; &someVar
```

Or, to generate: M2, 23.5, 23.5 use

```
&someVar = 23.5
Write #1; “M2, “; &someVar; “, “; &someVar
```

Thus, the Write #?; version of the Statement (with a semicolon“;” after the file number) provides the most explicit control of punctuation in the output file. Alternatively, the last statement could have been produced more simply with:

```
&someVar = 23.5
Write #1, “M2”, &someVar, &someVar
```

This means that there cannot be a comma inside quotes if using a Write #?, Command (with a comma “,” after the file number). However, this can be accomplished using the more explicit controls of Write #?; (with a semicolon “;” after the file number) ...

Note: The variations in the WRITE and PRINT statement may seem confusing because the difference between the “;” and “,” versions are not immediately obvious…they can look like the same command.

ADDITIONAL REGISTRY COMMANDS. The following ShopBot Programming Instructions are provided to interact with the registry:

SetUsrVal, textvalue

Sets user-determined variable. Can be any text (or number) value with the variable being saved to the registry as a string.

- This value is not erased until done so by the user! (It persists even when ShopBot is closed and the computer turned off)
- The saved value can be text or a number
- When SetUsrVal is used to read from the value back, the registry string is placed in the designated ShopBot variable as a variant (string or number)
- The current value of uValue can be inspected in ShopBot using [UD]

SetUsrValClrd, textvalue

As with SetUsrVal, this sets a user-determined variable. However, this value is automatically cleared at the end/beginning of each file. Value can be any text (or number) value with the variable being saved to the registry as a string

- The saved value can be text or a number
- When GetUsrValClrd is used to read from the value back, the registry string is placed in the designated ShopBot variable as a variant (string or number)
- The automatic clearing provides a “new file” flag that can be set to “used” or other appropriate indication that some function or process has been already been done once in a file. Value can only be inspected while file is running.

SetSpindleStatus, 0-1

Sets ShopBot flag for Spindle Start-Up Dialog. If set to 0, the spindle is considered not active and Dialog should display. If 1, then spindle is considered active and spindle will restart without dialog.

GetUsrVal, &any_ShopBotVariable_name

Retrieves previously set UsrVal to a ShopBot Variable. See above.

GetUsrValClrd, &any_ShopBotVariable_name

Retrieves previously set UsrValClrd to a ShopBot Variable. See above.

GetUsrPath, &any_ShopBotVariable_name

Retrieves path to User Data Folder (defined by Windows).

GetAppPath, &any_ShopBotVariable_name

Retrieves application path for ShopBot program.

GetPartFileName, &any_ShopBotVariable_name

Retrieves most recent Part File read. This will bring up name of Part File from last session if no others have yet been run.

GetSpindleStatus, &any_ShopBotVariable_name

Retrieve the current SpindleStatus. 0 indicates the spindle is not active. Next start will evoke the spindle-start dialog. 1 indicates spindle is considered active. See above.

There is a more extensive set of Registry Variables available to developers who are using their own software to directly read and save to the registry or command line system. See section below on “Windows Registry Interface”.

Additional Part File Programming Considerations

Movement Blocks

When a Part File is read by ShopBot Software, the program’s first priority is to process all the tool movement commands in such a way that they can be most efficiently and smoothly executed. To execute smoothly, the software reads through the file and accumulates tool movement commands, storing them into memory in a pre-processed form. This stored list of moves is called a ‘movement stack’. While being stored, the moves are also analyzed to determine where acceleration and deceleration ramps should be placed, and how other features such as ‘tabbing’ should be applied. Establishing the movement stack is always the first priority of the software as files are read.

In most cases, arranging the file into movement stacks is completely handled by ShopBot. However, in certain situations it can be helpful to take control of how movement blocks are handled.

As a Part File is being read, a movement block is created when the first tool move is encountered (e.g. an M2). The software will then continue to read lines accumulating movement instructions. No movement will be executed until the whole stack has been read.

The end of the stack is determined by:

1. Reaching a Command that is not a Move, Cut, or SwitchON Command, or a special case Insert.

2. Or, by reaching the end of stack memory.

After the stack is read, all the moves will execute. If the block is terminated by a non-Move Command, then that next command will be executed. Following it, when a new Move Command is encountered in the file, the next movement block will be acquired and executed (e.g. a J2, is a Jog and not a Move; the Jog ends the present stack and the stack is executed, then the Jog is executed as a single stack, point-to-point move).

If reaching the limit of available memory punctuated the reading of the Part File, then the bit will be pulled to the 'Safe-Z' height (if this feature is active) at the end of execution of the stack. After the pull-up, an additional stack will be read and then the bit will be re-inserted for cutting that stack of moves.

There are a couple of situations that requires the interruption of the automatic handling of movement blocks. The [SC, 2] Command can be used to explicitly put a stack-end and stack-restart [SC,2] at a specific location in a Part File.

One reason to control stacks is to control an interruption in cutting in order to load more of the file, rather than it occurring in an awkward location. By placing SC,2's at locations in a file before memory is used up by the current block, the block will be executed and a new one started. This will force the blocking to occur at a location in the file where it will not have an adverse effect on cutting.

Structuring Files

Efficiency in Part File creation, the speed of de-bugging, and the ability to reuse Programming work will be enhanced by adding structural elements to the programming.

First, layout your file with indents, labels and comments to make it as easy to read as possible and to break it into logical chunks. Second, use Sub-routines to carry out repetitive processes and try to write these subs in a generic way so that they can be used in other work. Third, break projects into multiple Part Files that are called or controlled by a master file. This will provide you with ready to use sub-Part Files for your other projects.

Adding Setup Lines to the Start of a File

After creating a Part File that does some type of cutting or machining, consider putting a few setup lines at the beginning of the file, particularly if this is a file that will be used regularly. Because all ShopBot Commands can be used in a Part File, put all the instructions that might be used to configure the tool at the start of the file. To make sure that the tool is optimally configured for cutting the project, place Commands that set the speeds, cutter size, or other features of ShopBot operation in this file:

'Setup lines for this file

```
VS, 1.25, .45, , 2.75, .80 'to set speeds
VC, .25 'always a .25 in cutter for this file
SA 'distances will be absolute
```

The "Header Writer" Virtual Tool in the ShopBot software makes it easy to add header and footer information to part files.

Troubleshooting difficulties in Part Files

When creating a Part File, it may be necessary to debug it to remove little errors such as typos or, math or logic problems, etc. Debugging can sometimes be more difficult than the actual

creation of the file itself ... and it is certainly less fun. Here are some suggestions to help make debugging a part file easier:

1. Break Part File projects into smaller Files in order to test and correct them individually in a more manageable size.
2. Use the PRINT and PAUSE Statements to display values of variables at various places in the file to check to make sure that things are working as expected.
3. Step through Files using the [FG] Command to identify trouble places.
4. Inspect the values of variables with [UL] to make sure they are what they should be.
5. Get similar results with MOVE and PREVIEW Modes. It's probably best to work out all complicated files in PREVIEW, before running the tool.

Why Have a ShopBot Part File Format?

The reason for having a specific Part File format is to make the process of producing, understanding, and maintaining CNC files as easy as possible. The industrial standard for CNC files is referred to as 'G-code'. Though it's not promoted, the most recent versions of ShopBot software will run G-code as well as ShopBot Part file code. However, G-code is a language that was created long ago. Its origins have left g-code unnecessarily arcane and difficult to understand when just inspecting a file. Moreover, it is a very loosely implemented standard with virtually every manufacturer using a different version and implementing it in idiosyncratic ways.

We felt that a Command language with mnemonically meaningful commands which were the same Commands used for direct control of the tool offered the best option for new robotics tool users. These types of Commands would be easier to learn, easier to understand, and easier to use in everyday work, whether at the keyboard or creating Part Files.

In addition, we believed that if additional functionality were provided as 'programming' instructions in the format of the BASIC language, it would make them ready for use by anyone familiar with programming. This convention would also provide a clear set of prescriptions for the format in which functions should be added to the ShopBot control language. G-code has very limited programming capability and that which it has is awkward to use at best.

If you are already an experienced CNC programmer and prefer G-code or have software that generates G-code just go ahead and run it on ShopBot. Our converter [FC] also makes it possible to rapidly make a translation from a G-code file to a ShopBot formatted Part File so that you can have the file in this more useful format. You will probably find that when you inspect a Part File, it will make perfect sense to you because it functionally resembles a G-code file -- being primarily just a list of coordinates through which the tool is moved.

Most popular CAM programs and design programs that generate toolpaths now offer the option of outputting the more usable ShopBot Part File format. In addition, at least one CNC Control system can already use the more easy to work with ShopBot format. We expect that other CNC vendors will increasingly also adopt the .SBP language.

The ShopBot Part File language, officially known as OpenSBP®, is not a proprietary language. ShopBot has published it as an open standard and set up a system for expanding and developing the standard. We expect to see it increasingly integrated into user-oriented CNC systems.

ShopBot Log Files

By default, ShopBot software writes two different types of 'log' files: 1) A **Part File log** for each Part File that is run, which tracks the utilization of that specific file. 2) An **SbSystem log** that records the use of the ShopBot Control Software, the tool, and the files that are run.

Part File Logs. If writing of Part File logs is enabled [VD], each time a Part File is run in Move Mode or Preview, a record is made of that and saved in the file with the same name as the Part File but with a ".log" extension. Each time the file is run, the file is appended with the new information, which is contained in two text lines. Several information items about the file are recorded as well as whether the file was terminated before completion and what line the termination occurred.

SbSystem Log. If writing of the SbSystem log is enabled [VD], then when ShopBot is started, closed, or files run, entries are appended into the log. The log is stored by default as C:\Program Files\ShopBot\ShopBot 3\Bin\SbSys.log. This log file could be used to create a system for monitoring use of the ShopBot.

There are three types of lines in the SbSys file: 1) Start-up or mode switch line; 2) Part File line; and 3) Closing line.

1. The Start-up line has either "NFP" or "StartFile" as the first data entry, where NFP indicates that the software was started normally and not by a command line call that passed a file. A command line call startup will be indicated by "StartFile".

- The second data item indicates whether the start occurred into CUT or PREVIEW mode;
- Five data items follow that indicate the parameters passed if this was a command line start, otherwise these parameters are empty;
- Next items are TIME and DATE;
- Empty data item;
- Control Box Version if available;
- Software Version;
- Serial#, Operator, sys1,…, sys4 (FUTURE IMPLEMENTATION PLANNED); and
- ~####### (an internal check number used by ShopBot)

2. The Part File line describes any Part File that is run by the software. It begins with full path and name of the Part File.

- CUT or PREVIEW Mode;
- Elapsed time;
- Next is TIME and DATE;
- Empty data item;
- Control Box Version if available;
- Software Version;

- Serial#, Operator, sys1,…, sys4 (FUTURE IMPLEMENTATION PLANNED); and
- ~####### (an internal check number used by ShopBot)

3. The Closing line records when the Control system is exited. The first data entry indicates the type of Close (NORMAL for a user close; or Type of close as set-up by a command line startup - see below) “Total Time” - just a printed term;

- Elapsed time;
- TIME and DATE;
- Empty data item;
- Control Box Version if available;
- Software Version;
- Serial#, Operator, sys1,…, sys4 (FUTURE IMPLEMENTATION PLANNED); and
- ~####### (an internal check number used by ShopBot)

Advanced Start-Up Information

From the beginning, ShopBot has sought to make the product open for use, improvement, and modification by others and to make it highly adaptable to a range of applications. ShopBot supports the use of systems by outside developers in several ways: One priority is to support those who are interested in creating non-parametric and parametric design systems and cutting projects. For these developers, the functionality of the Part File language and built-in cutting routines is the reason for this manual. ShopBot will continue to expand the flexibility of the Part File language and the ease with which it can be adapted to creative projects.

A second priority has been to make the ShopBot Control Software open for use by outside programs. Along those lines, two simple but powerful systems have been created to allow outside programs to run and manage the ShopBot software.

Windows Command Line Options… Controlling ShopBot software from outside programs (a)

A ShopBot can be effectively started and run from other software (e.g. drawing programs, CAM programs, custom interfaces). As such, the actual operation of the ShopBot system can be transparent to the operator. Many companies have created custom interfaces that allow an operator to interact with a proprietary system and then runs the ShopBot when tool motion or action is needed.

The system works using the option for a “Command Line” startup of the ShopBot software (Sb3. exe). A number of startup parameters help optimize the use of ShopBot software, while the specifics of the required action are passed in a file(s).

Command line format:

SB3.EXE path&filename, Port, OpenMethod, timing, CloseMethod, DisplayMethod, Offset

Parameters:

- Path&Filename (can be sent with or without " "s)
- Port - (1 – 16) optional
- OpenMethod –
    - o 1 (default) = MOVE/CUT; with FP Fill-In Opened set to this file
    - o 2 = MOVE/PREVIEW; with FP Fill-In Opened set to this file
    - o 3 = (currently same as #2)
    - o 4 = MOVE/CUT; no stop for Fill-In
    - o 5 = MOVE/PREVIEW; no stop for Fill-In
    - o 6 = Only make file current Part File
- Timing (not used in Windows software)
- CloseMethod
    - o 0 (default) = No close after running passed file
    - o 1 = Close after file, but with confirm question
    - o 2 = Immediate close and exit from ShopBot at end of file
- DisplayMethod
    - o 0 (default) = whatever is current in shopbot.ini
    - o 1 = Force console on
    - o 2 = Force console off
    - o 9 = Force console to always be off (OEM version, no use of SB Command Keys, must read file)
- Offset (Only applies if MOVE/CUT or PREVIEW with no stop for Fill-In
    - o 0 = no offset (default)
    - o 1 = Offset 3D
    - o 2 = Offset 2D

More information and examples can be found in the SBStarter folder in the Developer Tools folder in the ShopBot install.

Windows Registry Interface…Controlling the ShopBot software from outside programs (b)

For developers, ShopBot has provided a Windows Registry-based system for communicating between an outside program and ShopBot.

STATUS FLAGS indicating ShopBot Status as binary coded info in single decimal byte:

ORDER(&decimal value) [starting with lsd] FileRunning (1), PreviewMode(2), KeyPadOpen(4), PauseinFile(8), StopHit(16), StackRunning(32)

READ REGISTRY "ShopBot", "UserData", "Status", Single byte as Text
-All bits cleared to "0" when ShopBot software starts

INPUT SWITCH FLAGS indicating Input Switch Condition [input switch 1 through 8; binary coded in single decimal byte]

READ REGISTRY “ShopBot”, “UserData”, “InputSwitches”, Single byte as Text

OUTPUT SWITCH FLAGS indicating Output Switch Condition [output switch 1 through 8; binary coded in single decimal byte]

READ REGISTRY “ShopBot”, “UserData”, “OutPutSwitches”, Single byte as Text

Using the Registry, a Standard ShopBot Command can be passed to ShopBot from another program. The Command will be read and executed as soon as ShopBot finishes executing any running PartFiles, and any Commands already sent via a Command Line Pass (Method a, above).

The Command is PASSED FROM THE REGISTRY by setting:

“ShopBot”, “UserData”, “uCommand”, command

- Where command is a two letter ShopBot Command followed by normally formatted ShopBot parameter settings.

- The command is erased when it is read by ShopBot, and is always initially cleared when the software starts.

Two additional pieces of information are also available from the registry:

“ShopBot”, “AppData”, “uAppPath”, fullpath

- Provides the full path to the folder containing the current ShopBot.ini and Problem.log files.

“ShopBot”, “UserData”, “uUsrPath”, fullpath

- Provides the full path to the folder containing the current S3.exe file.

System Variables [use as %(sys var#)]

Sys Var#		Var	Units	Possible Problem with Old Ver
1	Location X	userUnits		
2	Location Y	userUnits		
3	Location Z	userUnits		
4	Location a	userUnits		
5	Location b	userUnits		
6	Base Coord X	userUnits		
7	Base Coord Y	userUnits		
8	Base CoordZ	userUnits		
9	Base Coord a	userUnits		
10	Base Coord b	userUnits		
11	Min Table Limit X	userUnits		VL
12	Max Table Limit X	userUnits		
13	Min Table Limit Y	userUnits		
14	Max Table Limit Y	userUnits		
15	Min Table Limit Z	userUnits		
16	Max Table Limit Z	userUnits		
17	Min Table Limit a	userUnits		
18	Max Table Limit a	userUnits		
19	Min Table Limit b	userUnits		
20	Max Table Limit b	userUnits		*CUTTER VAR now #23
21	DISTANCE	0-1		absolute or relative
22	MODE	0-1		cut or preview
23	CUTTER Dia	userUnits		VC
24	SW Delay	sec		
25	UNIT	0-1		VD; Inches or Metric
26				
27	Number of Axes	3-5		VD; Number of axes
28	Safe_Z	userUnits		VC
29	Safe_A	userUnits		VC
30	Number of Axes to Ck Lim	1-5		VL
31	WUx	single		VU;Unit value is scaled by UNIT Multiplier (1 for inches ...)
32	WUy	single		

Sys Var#		Var	Units	Possible Problem with Old Ver
33	WUz	single		
34	WUa	single		
35	WUb	single		
36	Driver Channel 1	XYZA or B		VI; Driver Channel Assignments
37	Driver Channel 2	XYZA or B		
38	Driver Channel 3	XYZA or B		
39	Driver Channel 4	XYZA or B		
40	Driver Channel 5	XYZA or B		
41	Tabbing Dist UP	single		VB
42	Tabbing Dist Next	single		
43	Tabbing Size	single		
44	Tabbing Lead Size	single		
45				
46	Inp #3 Mode	0 - 1		std or limit checking
47	Inp #4 Mode	0 - 1		Normally closed or open
48	Software Version	string		
49	Firmware Version	integer		
50				From 3.8.1 no longer doing composite value
51	Inp #1	0-1		
52	Inp #2	0-1		
53	Inp #3	0-1		
54	Inp #4	0-1		
55	Inp #5	0-1		
56	Inp #6	0-1		
57	Inp #7	0-1		
58	Inp #8	0-1		
59				
60	Current File Name	string		FP
61	X Offset Current File	single		-same as start location for file
62	Y Offset Current File	single		
63	Z Offset Current File	single		
64				
65				
66	X Proportion Current File	single		
67	Y Proportion Current File	single		
68	Z Proportion Current File	single		
69				
70				

Sys Var#		Var	Units	Possible Problem with Old Ver
71	MoveSpeed X	userUnits		VS
72	MoveSpeed Y	userUnits		*special case write, verify
73	MoveSpeed Z	userUnits		
74	MoveSpeed a	userUnits		
75	MoveSpeed b	userUnits		
76	JogSpeed X	userUnits		
77	JogSpeed Y	userUnits		*special case write, verify
78	JogSpeed Z	userUnits		
79	JogSpeed a	userUnits		
80	JogSpeed b	userUnits		
81	MoveRamp X	userUnits		
82	MoveRamp Y	userUnits		*special case write, verify
83	MoveRamp Z	userUnits		
84	MoveRamp a	userUnits		
85	MoveRamp b	userUnits		
86	JogRamp X	userUnits		
87	JogRamp Y	userUnits		*special case write, verify
88	JogRamp Z	userUnits		
89	JogRamp a	userUnits		
90	JogRamp b	userUnits		
91	Mode Input #1	0-4	old 46-47	INPUT SWITCH Modes
92	Mode Input #2	0-4		
93	Mode Input #3	0-4		
94	Mode Input #4	0-4		
95	Mode Input #5	0-4		
96	Mode Input #6	0-4		
97	Mode Input #7	0-4		
98	Mode Input #8	0-4		
99	Torch Height	0-1		
100	Full Output Byte	byte		OUTPUTS // top and bottom nibble reversed??
101	status Out #1	0-1		
102	status Out #2	0-1		
103	status Out #3	0-1		
104	status Out #4	0-1		
105	status Out #5	0-1		
106	status Out #6	0-1		
107	status Out #7	0-1		

Sys Var#		Var	Units	Possible Problem with Old Ver
108	status Out #8	0-1		
109	Mode # 4	-1 to -4		mode for output #4
110				
111	Mode Output #1	0-5		OUTPUT SWITCH Modes
112	Mode Output #2	0-5		
113	Mode Output #3	0-5		
114	Mode Output #4	0-5		
115	Mode Output #5	0-5		
116	Mode Output #6	0-5		
117	Mode Output #7	0-5		
118	Mode Output #8	0-5		
119	Mode Output #9	0-5		
120	Mode Output #10	0-5		
121	Mode Output #11	0-5		
122	Mode Output #12	0-5		
123	IndexerON	0-1		INdexerON flag
124	Analog Input #1	0-9		ADC1 approx +0.000v
125	Analog Input #2	0-9		ADC2 approx +0.000v
126	Inp #9	0-1		
127	Inp #10	0-1		
128	Inp #11	0-1		
129	Inp #12	0-1		
130				
131	Mode Input #9	0-4		
132	Mode Input #10	0-4		
133	Mode Input #11	0-4		
134	Mode Input #12	0-4		
135	Upper Output Byte	byte	4 low bits are 4 out-put switch-es	
136	status Out #9	0-1		
137	status Out #10	0-1		
138	status Out #11	0-1		
139	status Out #12	0-1		
140				
141	Z/A Trigger ON Thresh	dist		
142	Z/A Trigger OFF Thresh	dist		
143				

Sys Var#		Var	Units	Possible Problem with Old Ver
144	Location c	userUnits		
145	Base Coord c	userUnits		
146				
147				
148				
149				
150	Step InterVal Divider			
151	Multiplier X	0-10	Unit Value Multiplier	
152	Multiplier Y	0-10		
153	Multiplier Z	0-10		
154	Multiplier A	0-10		
155	Multiplier B	0-10		
156	Multiplier C	0-10		
157				
158	Current uController Voltage	0-9		approx +0.000v
159	Current uController Temp	Centigrade		
160	xxxx current max xxxx			

Bibliography

http://cam.autodesk.com/docs/cncbook/en/

http://www.shopbottools.com/ShopBotDocs/files/commandreference.pdf

https://www.hennepintech.edu/program/awards/281