

# **Lobo CNC 3-Axis Milling Machine Controller and G-Code Interpreter for *PIC-SERVO*-based Motion Controllers**

## **CAUTION**

This program is a prototype application provided by the Lobo CNC Project for hobbyists building their own CNC systems. It is not a failsafe product and is not intended for production use. It should be used with the utmost of caution, and only by users familiar with automated equipment who understand the associated risks. There are no warranties whatsoever regarding its operation. The user of this software shall not use it in any situation which could cause property damage, injury or loss of life, and the user shall take sole responsibility for any liabilities resulting from its use.

## **1.0 Overview**

PSCNC is a complete example program for operating a 3-axis milling machine. It runs under Windows 95 through Windows 7 and interfaces to three ***PIC-SERVO S3*** servo motor controllers. It includes the following features:

- Digital position readout panel displays positions in program or raw machine coordinates
- Controls for setting the origin of the program coordinate system
- Allows the use of hand cranks while still displaying the position
- Manual jogging controls
- G-Code Display window and immediate single-line G-Code execution
- Dynamic feed rate override, rapid speed override
- Continuous contouring of tangent and near tangent tool path segments
- Tool length compensation

The G-Code interpreter only implements a basic set G-Codes and M-Codes, but should be sufficient for most hobby applications. The source code is provided for Borland C++ Builder 5.0 as an example, and may be modified as needed.

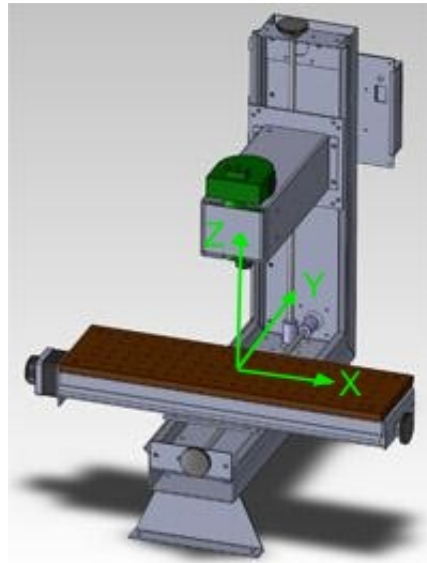
## **2.0 Setting Up for Use With Your Machine**

### **2.1 Coordinate Systems**

The coordinate system used by PSCNC is shown in *Figure 1* below. The orientation of this coordinate system is fixed, but the program origin may be reset by the user. The origin for the machine coordinate system corresponds to the position of the tool tip (using a zero length tool) when the machine is located at its positive limit stops.

All motion in the X, Y or Z directions refers to the motion of the tool tip relative to the coordinate system fixed in the table, even though it is the table that may be actually moving. For

example, a positive X motion will move the table to the left, causing the tool tip to move along the positive X axis.



*Figure 1 - PSCNC Coordinate System.*

## **2.2 Hardware Setup**

The PSCNC program is designed for use with the S3x3 three-axis machine controller board. This board has three PIC-SERVO S3 motor controller channels, and connects to your PC via USB. Please refer to the S3x3 Motor Control Board documentation for details on this board.

The PSCNC program communicates using a USB Virtual COM Port. When you first plug in your USB connection to the S3x3 board and turn on the power, Windows should attempt to find the appropriate Virtual COM Port driver. More modern versions of Windows will do this successfully (if internet connected). For older versions of Windows and unconnected PCs, you should go to: [www.ftdichip.com/Drivers/VCP.htm](http://www.ftdichip.com/Drivers/VCP.htm) to download the appropriate driver.

Once Windows has installed the driver, you need to set the Virtual COM Port number for the board. With the board power turned on, go to the Windows device manager and expand the list of Ports (COM & LPT) available. You should see listed a USB Serial Port. Right-click on the port and display its properties. Under the Port Settings tab, click on the Advanced button to display more port settings. There will be a drop-down box with the Port number listed (eg: COM3). It is recommended to switch the virtual COM port number to COM5 or COM6 to avoid conflicts with other devices.

Also while adjusting the Advanced Settings, you should lower the Receive and Transmit buffer sizes to 64, as this will speed up communications. You should also lower the Latency Timer to 8 msec.

## 2.3 Software Setup

To install, simply unzip all the files in PSCNC.ZIP into a single folder. There is no formal installation process.

It is strongly recommended that while running PSCNC, you have all networking turned **off** and have **closed** all other applications. This will prevent Windows from interrupting the real-time monitoring required for safe machining operation.

### **The Initialization File**

The file PSCNC.INI contains all of the information required to operate the PSCNC program with your particular milling machine. The default values are already set up for the Lobo CNC milling machine – the only value you need to change is the COM port number.

This file can be edited with a text editor such as Notepad. Each line contains a parameter name followed by a ‘:’, followed by the parameter value. Only change the parameter values. You should make a backup copy of the original version of the PSCNC.INI in case you accidentally corrupt the file. This file should be located in the same working directory as the program file PSCNC.EXE.

The initialization file contains the following parameters:

<i>Parameter</i>	<i>Description</i>
comport	Use ‘COM1:’ through ‘COM9:’ to match the PC COM port number you set with the Device Manager.
xaxis, yaxis, zaxis	Module addresses used for each of the axes (1, 2 or 3 for each axis)
bufsize	The maximum number of path points to be stored in each <b>PIC-SERVO S3</b> . A <i>bufsize</i> of 30 path points with a path frequency of 60 Hz will require your PC to load more path points at least every ½ second. Use more path points (up to 85) for slower PC’s, fewer path points for a more responsive feed rate override.
pfreq	Path frequency should be set to 30 or 60 only. 60 Hz provides a more accurate path, 30 Hz requires less computation.
maxang	Maximum angle, in degrees, between tool path segments for them to be considered tangent. If the angle between adjacent line segments is less than this angle, and if contouring is turned on, the path will be executed smoothly without stopping. Lower values improve smoothness, higher values will reduce the overall cycle time. 10 degrees is a reasonable value.
xscale, yscale, zscale	Scale factors equal to the number of encoder counts per inch (or millimeter). If a positive number of encoder counts causes a negative motion in that axis, then the corresponding scale factor should be negative. The default values for the Lobo CNC machine are in inches.
xaccel, yaccel, zaccel	Acceleration values to use for each axis, in inches (or millimeters) per second per second.
xmaxvel,	Maximum velocities for each axis in inches (or millimeters) per second.

y <sub>maxvel</sub> , z <sub>maxvel</sub>	The is the velocity used for rapid motions, and the greatest of these velocities is used as a limit on the overall feed rate. (If these values differ, it is up to the user to make sure that any programs do not exceed the maximum velocity of the slower axes.)
x <sub>max</sub> , y <sub>max</sub> , z <sub>max</sub>	Maximum operating positions, in inches (or millimeters), relative to the home position. Note that because the homing procedure homes the mill to the positive limit stops, these values will be negative. The maximum operating position should be far enough away from any hard limit stops so that when this position is reached, there is still room to smoothly decelerate to a stop without crashing.
x <sub>min</sub> , y <sub>min</sub> , z <sub>min</sub>	Minimum operating positions, in inches (or millimeters), relative to the home position. Note that because the homing procedure homes the mill to the positive limit stops, these values will be negative, and also more negative than the maximum positions. The minimum operating position should be far enough away from any hard limit stops so that when this position is reached, there is still room to smoothly decelerate to a stop without crashing.
x <sub>kp</sub> , y <sub>kp</sub> , z <sub>kp</sub>	Proportional gains used by the servo controllers. (Range: 0 - 32,767) These values should be determined using the NMCTest program.
x <sub>kd</sub> , y <sub>kd</sub> , z <sub>kd</sub>	Derivative gains used by the servo controllers. (Range: 0 - 32,767) These values should be determined using the NMCTest program.
x <sub>ki</sub> , y <sub>ki</sub> , z <sub>ki</sub>	Integral gains used by the servo controllers. (Range: 0 - 32,767) These values should be determined using the NMCTest program.
x <sub>il</sub> , y <sub>il</sub> , z <sub>il</sub>	Integration limits used by the servo controllers. (Range: 0 - 32,767) These values should be determined using the NMCTest program.
x <sub>ol</sub> , y <sub>ol</sub> , z <sub>ol</sub>	Output limits used by the servo controllers. (Range: 0 - 255) These values should be determined using the NMCTest program. (Usually set to 255)
x <sub>cl</sub> , y <sub>cl</sub> , z <sub>cl</sub>	Current limits used by the servo controllers. (Range: 0 - 255) These values should be determined using the NMCTest program. (Use 0 if amplifiers have no current feedback.)
x <sub>el</sub> , y <sub>el</sub> , z <sub>el</sub>	Position error limits, in encoder counts, used by the servo controllers. (Range: 0 - 16,383) These values should be determined using the NMCTest program.
x <sub>sr</sub> , y <sub>sr</sub> , z <sub>sr</sub>	Servo rate divisors used by the servo controllers. All should be set to 1.
x <sub>dc</sub> , y <sub>dc</sub> , z <sub>dc</sub>	Deadband compensation used by the servo controllers. (Range 0 - 255) Usually set to 0 except for when using SS-Drive controllers.
x <sub>padv</sub> , y <sub>padv</sub> , z <sub>padv</sub>	Phase advance settings used by SS-Drive type controllers. Set to 0 for all other types of <b>PIC-SERVO</b> controllers.
x <sub>poff</sub> , y <sub>poff</sub> , z <sub>poff</sub>	Phase offset settings used by SS-Drive type controllers. Set to 0 for all other types of <b>PIC-SERVO</b> controllers.

### ***The Tool Description File***

The PSCNC program supports tool length compensation for up to 20 tools. G-Code programs will use an “M06 Tnn” command to indicate a tool change, and a “G43 Hnn” command to select a tool offset. The relative tool lengths are specified in the file PSCNC.TLL. (You can make a backup copy of the sample file and then edit your version with a text editor.) This file should have exactly 20 lines looking like:

```
T01  0.00
T02  2.623
T03  1.445
.
.
.
T20  0.00
```

By convention, tool T01 should have length zero, and all other tool lengths will be relative to the length of T01. Tools longer than T01 will have positive length entries, and tools shorter will have negative entries. PSCNC.TLL should be located in the same working directory as the program file PSCNC.EXE

Note: you can choose to not use the tool length compensation and leave the tool at the default T01. In this case, every time you change tools, you will position your tool tip to the appropriate zero position and reset the Z Origin.

## **3.0 Using PSCNC**

### ***3.1 Homing Procedure***

When you first run PSCNC, you will be prompted to turn on the power to the motor controllers. At this time, you should turn on the power for the S3x3 board. After clicking on OK, all motors will twitch back & forth a small amount as the motor commutation is initialized.

Next, you will be asked if you want to “home the mill”. The homing procedure will move each axis slowly up against its limit stop, and then move all axes back into the allowable working range of motion. Once homing is complete, the software will monitor the position of the mill to prevent it from moving outside the allowable range of motion.

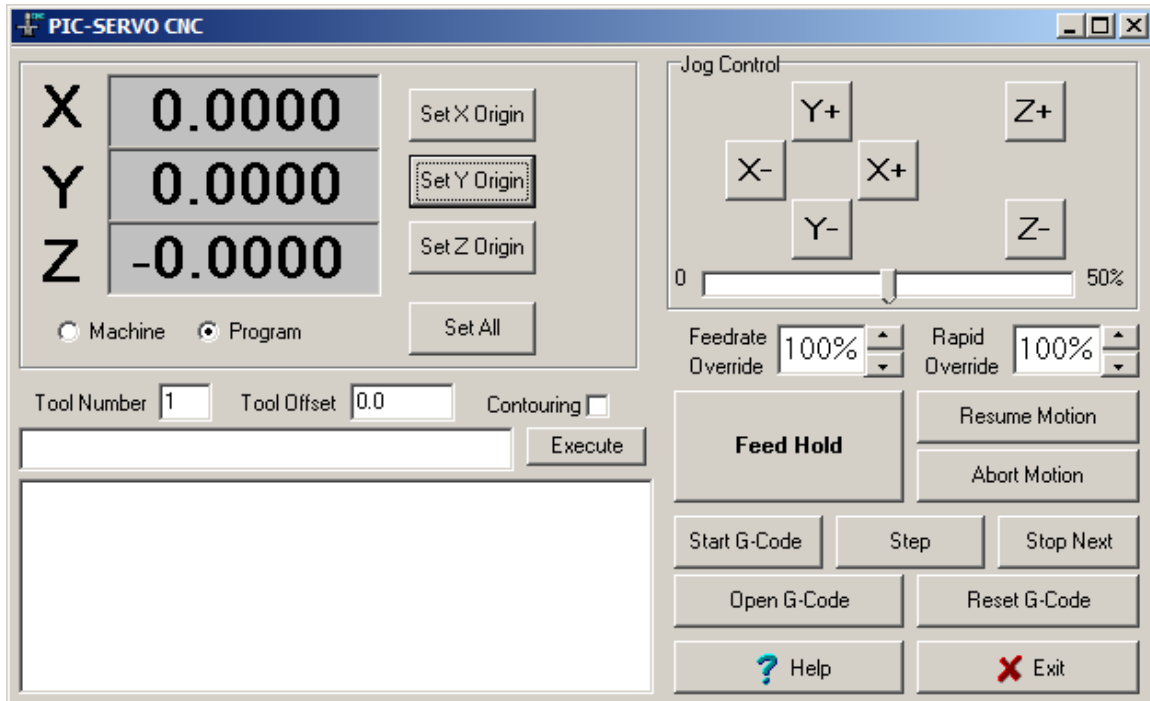
Note that the homing does not require limit switches – it instead detects when the motor runs into the hard limit stop. If any of the axes has excessive friction, it may stop its homing motion before hitting the end of the range of motion. If this happens, you should adjust the assembly of the axis to eliminate the excess friction.

If you do not elect to run the homing procedure, the software cannot monitor the position of the mill and it will be very easy to accidentally crash the mill into its hard stops. Therefore, it is recommended that you always opt to home the mill.

Note that even after executing the homing procedure, it is possible to crash the tool into your part, other fixturing, or the table. In particular, the software does not account for the current tool length, and you should use extra caution when using longer tools.

### 3.2 The Control Panel

Shown in *Figure 2* below is the PSCNC control panel. It is broken into four regions: 1) the digital readout panel with origin controls, 2) the jogging control panel, 3) the G-Code display window, and 4) the G-Code execution controls.



*Figure 2 - PSCNC Control Panel.*

#### Digital Readout Panel

The digital readout panel displays milling machine position based on the actual motor positions. This position will reflect any servo positioning errors, but it will not reflect any backlash in your drive train.

Coordinates can be displayed in either absolute machine coordinates or in program coordinates relative to the user settable origin. If the mill has been homed, the displayed machine coordinates will always be negative, because each axis has its zero position defined at the location of the positive limit switch.

Program coordinates are displayed relative program origin. All lines of G-Code programs will also be interpreted relative to the program origin. To set the program origin, move an axis of the milling machine to the desired zero position and click on the corresponding "Set Origin" button. The program coordinate position for that axis will then read zero. You can use the "Set All" to simultaneously set all origin coordinates.

You can also set a program position coordinate to a specific value by clicking on the displayed value. You will then be prompted to enter a new coordinate value.

Note that the Z program coordinate displays the coordinate of the tool tip using tool length compensation. Typically, you should use a tool length offset of 0.00 for tool T01, and teach the height of your part using tool T01. All other tool heights will then be relative to the length of tool T01.

You will note that there are no controls for enabling or disabling the servos on each of the axes. By default, this version of PSCNC leaves the axes turned off when they are not actively being driven under computer control. This means that any time an axis is idle, you can simply turn the hand wheels for manual positioning. The position will automatically be tracked in the Digital Readout Panel. Manually positioning the axes for accurately setting the origin is much easier than using the Manual Jog Controls described below. Manual positioning is also quite convenient for machining by hand.

### Jog Controls – X, Y & Z

The jog controls allow you to move individual axes at a constant velocity. Down-clicking with the **left** mouse button and holding the button will move the corresponding axis in the positive or negative direction. Releasing the button will decelerate the axis to a stop. The speed can be controlled using the slider bar underneath the jog buttons. The manual jog speed is limited to 50% of the maximum programmable feed rate.

Down-clicking with the **right** mouse button and holding the button will move the corresponding axis at the **maximum** traverse speed. This is useful for gross positioning of the axes. However, because it takes some distance to decelerate from higher speeds, you should release the button well before your goal, and then press the left mouse button to jog at a more controlled speed.

Through a combination of jog controls and manually turning the hand wheels, you can very quickly and accurately position the axes for setting origins or for manual machining.

### G-Code Display Window

The G-Code display window displays 7 lines of your G-Code program, with the current line highlighted. This is a display window only, and the code cannot be edited. (You should use the Notepad or other editor to create and edit G-Code programs.)

The text entry window just above the G-Code display allows you to enter in a single line of G-Code and then execute it immediately. This is quite useful for moving to exact coordinates (using G00 or G01), for tool offset adjustments (using G43), or for setting feed rate parameters (Fx.xx).

The G-Code display area also shows the current tool number (set with the M06 command) and the current tool length (independently set with the G43 command).

## G-Code Execution Controls

The G-Code execution controls, next to the G-Code display window, allow you to open G-Code programs, start and stop programs, step through programs one line at a time, and reset the program to its beginning.

This area also has feed rate controls for dynamically increasing the tool feed rate or pausing executions. The up-down buttons next to the feed rate override display allow you to increase or decrease the actual feed rate from 10% to 400% of the programmed value. Changes to the feed rate override can be made while the mill is moving. The speed for rapid motions has a separate override which works in a similar way, except that the speed change will not take place until the next rapid motion.

The “Feed Hold” button will pause the motion of the mill along its programmed trajectory. Hitting the “Resume Motion” button will continue execution along the same path. Hitting the “Abort Motion” button while in a feed hold state will stop the execution of the program altogether. (If you want to stop the program execution, but only after the current motion has finished, you should use the “Stop Next” button.)

## Contouring

One advanced feature of the PSCNC program is its ability to merge line segments and arcs which are tangent, or nearly tangent, into a single smooth motion without stopping at each end-point. Clicking on the “Contouring” checkbox will enable this feature. The ‘maxang’ parameter in the initialization file specifies the maximum angle, in degrees, between line segments for them to be considered nearly tangent. You should turn off this feature if you have programmed very small radius arcs, because they could result in very high accelerations, and possibly give you a servo positioning error. Contouring can be turned on and off manually while a program is running, or it can be turned on and off by the program itself using M-Codes M21 and M22.

With contouring turned on, there are several conditions under which the programmed tool path will be broken into separate motions:

- 1) Two consecutive tool paths are not tangent.
- 2) A rapid motion (G00) is encountered.
- 3) A dwell command (G04) is encountered
- 4) A feed rate change is encountered.
- 5) Any M-Code is encountered.

When one of these conditions is encountered, the previous motion will decelerate to a stop before the next motion will be started.

## Help & Exit

The “Help” button will launch the Adobe Acrobat Reader application for viewing the file PSCNC.PDF. PSCNC.PDF should be in the same working directory as the program file.

The “Exit” button will reset all of the motor controllers and terminate the program. The same action will take place by clicking on the ‘X’ in the upper right corner of the application window. If the program terminates abnormally, as will happen if communications with the servo controller



are interrupted, you may need to turn the motor controller's logic power off and then on again before restarting the program.

### **3.3 The G-Code Interpreter**

The G-Code interpreter executes the G-Code program once it has been loaded. It only supports a subset of the standard G-Codes, but the most used ones are implemented. The interpreter also handles the execution of M-Codes. These are auxiliary codes are used to control other machine functions such as tool changers, spindle and coolant control, as well as some program flow control functions. Only a few M-Codes are actually supported, and when they are encountered, program execution will pause and a message will be printed out.

G-Codes were designed at a time when computer memory was very expensive. Therefore, programs are allowed to be very sparse, and nearly unreadable. Another unfortunate problem with G-Codes is that different G-Code interpreters will operate in slightly different ways. The PSCNC G-Code interpreter tries follows most of the standards, but there are a few exceptions and other suggestions to follow:

- 1) Most G-Code interpreters will interpret a number without a decimal point as that number divided by 1000. For example, 2139 will actually be interpreted as 2.139. However, numbers with a decimal point will be interpreted at face value. The PSCNC interpreter interprets *all* numbers at their face value, regardless of a missing decimal point.
- 2) G-Codes are normally broken into different groups. Multiple G-Codes may appear on the same line of G-Code, but only one command from any one group is allowed. The PSCNC interpreter allows this to some degree, but there may be ambiguities in which command is executed first. Therefore, it is far safer and more predictable to only use one G-Code per line.
- 3) Some G-Codes are said to be 'modal', meaning that once the G-Code is active, additional data can appear without needing to re-state the G-Code on each line. For example, a G00 rapid move is usually followed on the same line by X, Y or Z data. If no other command from the same group appears, additional X, Y and Z data may be entered without an additional G00 code. Again, this programming practice can introduce ambiguity and is discouraged.
- 4) M-Codes are usually allowed on the same line a G-Codes. The PSCNC interpreter allows only one M-Code on a line, and it always gets executed after the G-Code. Again, putting multiple codes on one line can introduce ambiguity and is discouraged.
- 5) Comments are delimited with '(' and ')' and must appear on a single line. Comments are encouraged given how unreadable G-Codes tend to be.
- 6) The maximum G-Code file size is about 2.5 megabyte. Some CAM systems which automatically generate G-Code files may produce files larger than this. In these cases, the files can be broken up into multiple files which are loaded and executed separately.
- 7) All G-Code letter characters (G, X, Y, Z, M, F, etc.) must be capitalized and have a space in front of them or appear at the beginning of a line.
- 8) All programs must end with a M00 or and M30.

## G-Codes Implemented

The G-Codes implemented in PSCNC appear below. G-Codes not listed below are ignored by the interpreter.

### **G00** Rapid Motion

Initiates a rapid, uncoordinated move to the specified X, Y and Z coordinates.

#### *Examples:*

```
G00 X1.0 Y2.0 (moves to X=1.0, Y=2.0, Z is unchanged)
G00 Z-0.5      (moves down to Z=-0.5, X and Y unchanged)
```

### **G01** Linear Motion

Moves in a straight line to the specified X, Y and Z coordinates at the specified feed rate. Feed rates are specified in inches (or millimeters) per *minute*.

#### *Examples:*

```
G01 X1.0 Y2.0 F10.0 (moves in a straight line to X=1.0, Y=2.0 with)
                    ( a feed rate of 10 inches per minute )
G01 Z-0.5           (moves down to Z=-0.5 at the previously set)
                    ( feed rate )
```

### **G02** Clockwise Circular Arc

Moves along a clockwise circular arc to the specified X, Y and Z coordinates at the specified feed rate. Arcs can be specified using an I, J, K format or using an R format.

With the I, J, K format, the center point of the arc is specified using I, J and K values which are *relative* to the starting point of the arc. (The I value is the distance along the X axis, the J value is the distance along the Y axis, and the K value is the distance along the Z axis.) The distance from the starting point to the center point must be the same as the distance from the ending point to the center, or else an error will be generated. The plane in which the arc lies is selected using G-Codes G17, G18 or G19. To specify a complete circle, the X, Y and Z coordinates can be omitted, or set equal to the starting point. The motion will be clockwise with respect to looking down upon the selected plane.

#### *Examples:*

```
(Starting point: X = 1.0, Y = 0.0, Z = 0.0)
G17 F30.0          (select the X-Y plane, set the feed rate to 30 in/min)
G02 X0.0 Y-1.0 I-1.0 (creates a 90 degree arc to the point X=0.0, )
                    ( Y=-1.0 with a radius of 1.0, centered about the )
                    ( point X=0, Y=0 )
G19                (select the Y-Z plane )
G02 J1.0            (create a complete circle, starting and ending at )
                    ( X=0.0, Y=-1.0, Z=0.0, with a radius of 1.0 )
```

With the R format, only an X, Y, Z goal point is specified along with a radius (R value). The controller then calculates to location of the center point so that the center-to-end point distance is equal to the center-to-start point distance. You cannot create an arc of more than 180 degrees with a single R-format arc. (You would need to use multiple arcs.) The R value must be greater

than 0.0. If the R value is less than  $\frac{1}{2}$  the start-to-end point distance, a 180 degree arc will be executed.

*Example:*

```
(Starting point: X = 1.0, Y = 0.0, Z = 0.0)
G17 F30.0      (select the X-Y plane, set the feed rate to 30 in/min)
G02 X0.0 Y1.0 R1.0 (creates a 90 degree arc to the point X=0.0, )
                  ( Y=1.0 with a radius of 1.0, centered about the )
                  ( point X=0, Y=0 )
```

### **G03 Counter-clockwise Circular Arc**

Moves along a counter-clockwise circular arc to the specified X, Y and Z coordinates at the specified feed rate. Arcs can be specified using an I, J, K format or using an R format.

With the I, J, K format, the center point of the arc is specified using I, J and K values which are *relative* to the starting point of the arc. (The I value is the distance along the X axis, the J value is the distance along the Y axis, and the K value is the distance along the Z axis.) The distance from the starting point to the center point must be the same as the distance from the ending point to the center, or else an error will be generated. The plane in which the arc lies is selected using G-Codes G17, G18 or G19. To specify a complete circle, the X, Y and Z coordinates can be omitted, or set equal to the starting point. The motion will be counter-clockwise with respect to looking down upon the selected plane.

*Examples:*

```
(Starting point: X = 1.0, Y = 0.0, Z = 0.0)
G17 F30.0      (select the X-Y plane, set the feed rate to 30 in/min)
G02 X0.0 Y-1.0 I-1.0 (creates a 90 degree arc to the point X=0.0, )
                    ( Y=-1.0 with a radius of 1.0, centered about the )
                    ( point X=0, Y=0 )
G19            (select the Y-Z plane )
G02 J1.0        (create a complete circle, starting and ending at )
                ( X=0.0, Y=-1.0, Z=0.0, with a radius of 1.0 )
```

With the R format, only an X, Y, Z goal point is specified along with a radius (R value). The controller then calculates to location of the center point so that the center-to-end point distance is equal to the center-to-start point distance. You cannot create an arc of more than 180 degrees with a single R-format arc. (You would need to use multiple arcs.) The R value must be greater than 0.0. If the R value is less than  $\frac{1}{2}$  the start-to-end point distance, a 180 degree arc will be executed.

*Example:*

```
(Starting point: X = 1.0, Y = 0.0, Z = 0.0)
G17 F30.0      (select the X-Y plane, set the feed rate to 30 in/min)
G02 X0.0 Y1.0 R1.0 (creates a 90 degree arc to the point X=0.0, )
                  ( Y=1.0 with a radius of 1.0, centered about the )
                  ( point X=0, Y=0 )
```

#### **G04 Dwell**

Pauses the program for some number of seconds specified with the P parameter. The actual dwell will only be approximately equal to the specified time.

##### *Example:*

```
G04 P1.5           (pause the program execution for about 1.5 seconds)
```

#### **G17 X-Y Plane Selection**

Selects the X-Y plane for circular motions. The starting and ending points for circular motions must have the same Z value. Clockwise or counterclockwise will be interpreted as the direction of motion when looking down along the Z axis.

##### *Example:*

```
G17               (no other parameters needed)
```

#### **G18 Z-X Plane Selection**

Selects the Z-X plane for circular motions. The starting and ending points for circular motions must have the same Y value. Clockwise or counterclockwise will be interpreted as the direction of motion when looking down along the Y axis.

##### *Example:*

```
G18               (no other parameters needed)
```

#### **G19 Y-Z Plane Selection**

Selects the Y-Z plane for circular motions. The starting and ending points for circular motions must have the same X value. Clockwise or counterclockwise will be interpreted as the direction of motion when looking down along the X axis.

##### *Example:*

```
G19               (no other parameters needed)
```

#### **G43 Tool Length Compensation**

Adjusts the length of the tool used to the specified H value. The H value corresponds the length found in the tool length file PSCNC.TLL. All subsequent Z axis coordinates will be with respect to the new tool tip.

##### *Example:*

```
(old tool length = 0.0, tool #2 length = 1.5)
G00 Z0.0          (move the old tool to a height of 0.0)
M06 T02           (change to tool #2)
G43 H02           (use the length of tool #2)
G00 Z0.0          (moves the Z axis up by 1.5 so that the new tool tip
                  (is in the same location as the old tool tip)
```

#### **M-Codes Implemented**

The M-Codes implemented in PSCNC appear below. If an M-Code not listed below is encountered, program execution will stop, and an error message will appear.

**M00** Program Pause

Causes the program execution to stop. Execution can be resumed by hitting the “Start G-Code” button. M00 or M30 should appear as the last line of every program.

**M03** Spindle On

Stops the program execution and prompts the user to turn on the spindle. (No actual spindle control is implemented.). Execution can be resumed by hitting the “Start G-Code” button.

**M05** Spindle Off

Stops the program execution and prompts the user to turn off the spindle. (No actual spindle control is implemented.). Execution can be resumed by hitting the “Start G-Code” button.

**M06** Tool Change

Stops the program execution and prompts the user to change the tool. (No actual tool change is implemented.). The tool length compensation must be changed separately using G43. Execution can be resumed by hitting the “Start G-Code” button.

**M21** Contouring On

Turns on the path contouring feature. Path contouring can also be turned on and off via the control panel.

**M22** Contouring Off

Turns off the path contouring feature. Path contouring can also be turned on and off via the control panel.

**M30** Program End and Reset

Causes the program execution to halt, and the program will be reset to the beginning. M30 or M00 should appear as the last line of every program.

## G-Code Program Examples

### *Rectangular Path with Rounded Corners*

G17	(Choose X-Y Plane)
M06 T03	(Choose tool #3)
G43 H03	(Use tool length for tool #3)
M03	(Prompt user to turn on spindle)
G00 X0.5 Y0.0 Z0.25	(Rapid move to a point above the part)
G01 Z-0.1 F5.0	(Plunge cut into part slowly)
G01 X2.5 F10.0	(Cut along X at a faster rate)
G03 X3.0 Y0.5 J0.5	(Cut a 90 degree arc about the ( center point X=2.5, Y=0.5 )
G01 Y1.0	(Cut in the Y direction)
G03 X2.5 Y1.5 I-0.5	(Cut second corner radius)
G01 X0.5	(Cut upper side of the rectangle)
G03 X0.0 Y1.0 J-0.5	(Cut third corner radius)
G01 Y0.5	(Cut the last side)
G03 X0.5 Y0.0 I0.5	(Cut the last corner radius)
G01 Z0.25 F30.0	(Raise cutter above part quickly)
M05	(Prompt user to turn off spindle)
M30	(Stop and reset program)

### *Center Drill and Finish Drill a Hole Pattern*

M06 T01	(Choose tool #1 - center drill)
G43 H01	(Use tool length for tool #1)
M03	(Prompt user to turn on spindle)
G00 X1.5 Y1.5 Z0.25	(Rapid move to a point above 1st hole)
G01 Z-0.25 F5.0	(Slowly plunge center drill)
G01 Z0.25 F30.0	(Raise tool quickly)
G00 X-1.5 Y1.5	(Rapid move to 2nd hole)
G01 Z-0.25 F5.0	
G01 Z0.25 F30.0	
G00 X-1.5 Y-1.5	(Rapid move to 3rd hole)
G01 Z-0.25 F5.0	
G01 Z0.25 F30.0	
G00 X1.5 Y-1.5	(Rapid move to 4th hole)
G01 Z-0.25 F5.0	
G01 Z0.25 F30.0	
M05	(Prompt user to turn off spindle)
G00 Z3.0	(Raise spindle up for tool change)
M06 T02	(Choose tool #2 - 0.250 dia. drill)
G43 H02	(Use tool length for tool #2)
M03	(Prompt user to turn on spindle)
G00 X1.5 Y1.5 Z0.25	(Rapid move to a point above 1st hole)
G01 Z-0.5 F5.0	(Slowly plunge the drill)
G01 Z0.25 F30.0	(Raise tool quickly)
G00 X-1.5 Y1.5	(Rapid move to 2nd hole)
G01 Z-0.5 F5.0	
G01 Z0.25 F30.0	
G00 X-1.5 Y-1.5	(Rapid move to 3rd hole)
G01 Z-0.5 F5.0	
G01 Z0.25 F30.0	
G00 X1.5 Y-1.5	(Rapid move to 4th hole)
G01 Z-0.5 F5.0	
G01 Z0.25 F30.0	
M05	(Prompt user to turn off spindle)
M30	(Stop and reset program)

### *Multi-plane Contoured Motion*

This program does nothing useful, but it demonstrates a smooth single path consisting of several arcs lying in different planes.

G00 X0 Y0 Z0	(Move to the program origin)
M21	(Turn on contouring feature)
G17	(Select the X-Y plane)
G02 X0.5 Y0.5 I0.5 F60.0	(Arc in X-Y)
G18	(Select the Z-X plane)
G02 X1.0 Z0.5 K0.5	(Arc in Z-X)
G19	(Select the Y-Z plane)
G03 Y0.0 Z1.0 J-0.5	(Arc in Y-Z)
G17	(Select the X-Y plane)
G02 X0.0 Y0.0 I-0.5	(Arc in X-Y)
G19	(Select the Y-Z plane)
G02 Y0.0 Z0.0 K-0.5	(Final arc back to the origin)
M30	(Stop and reset program)