PIC-SERVO CNC: 3-Axis Milling Machine Controller and G-Code Interpreter for *PIC-SERVO (SC, CMC)* Motion Controllers

CAUTION

This program is provided by **JEFFREY KERR, LLC** as an example only, intended for users designing similar systems. It is not intended as a failsafe product for general use, and as such, it should be used for evaluation purposes only and only with the utmost of caution. It is *not* supported by **JEFFREY KERR, LLC**, and we make 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

PIC-SERVO CNC (PSCNC) is a complete example program for operating a 3-axis milling machine. It runs under Windows 95/98/NT/2000/XP and interfaces to three ***PIC-SERVO SC*** or ***PIC-SERVO CMC*** (coordinated motion control) 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
- Servo disable controls 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 in-depth evaluation purposes. The source code is provided for Borland C++ Builder 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.

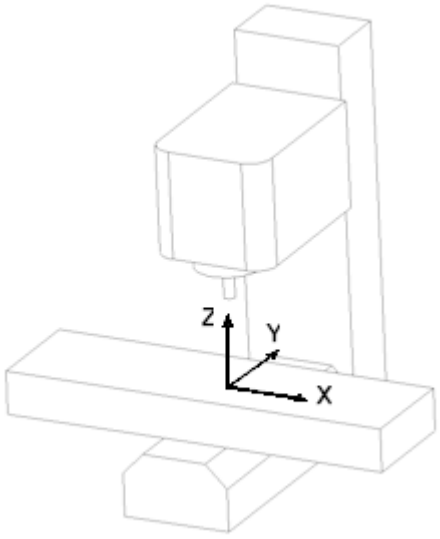

Figure 1 - PSCNC Coordinate System.

2.2 Hardware Setup

The PSCNC program expects to find three ***PIC-SERVO SC*** or ***PIC-SERVO CMC*** motor controllers (one each for the X, Y and Z axes) connected to your PC's COM port through a Z232-485 converter. Please refer to the ***PIC-SERVO*** Motor Control Board documentation for details on interconnecting the boards, and connecting your motors. No other controller boards should be connected.

Before running the PSCNC software, you should run the NMCTest.exe test utility program. This will allow you to verify that all of your hardware is connected and working properly. This program will also allow you to make test motions for each axis, and select the best servo control gains for your motor and mechanism. You will need to review the ***PIC-SERVO*** chip set data sheet for assistance in selecting servo gains.

When running the NMCTest program, you will need to determine the following information which you will then enter into the PSCNC initialization file:

1. The assigned addresses for each axis
2. The servo gain parameters used for each axis
3. The scale factor and direction of motion for each axis (in terms of encoder counts per inch or mm)
4. The maximum velocity to use for each axis (converted to inches or millimeters per second)
5. The maximum acceleration to use for each axis (converted to inches or millimeters per second per second)
6. The working range of motion of each axis (converted to inches or millimeters)

In general, the default servo gain values and velocity and acceleration values in the NMCTest program are good starting points. Make several test motions with each axis, varying the different parameters one by one to come up with the best parameters for your machine.

Lastly, to use the homing feature of the PSCNC program, you should connect two limit switches to each of the axes - with Limit 1 of the PIC-SERVO board connected to the switch triggered at the end of the positive range of motion and Limit 2 connected to the switch triggered at the end of the negative range of motion. The switches should be connected "normally closed", with the connection opening when the switch is hit.

2.3 Software Setup: The Initialization File

The file PSCNC.INI contains all of the information required to operate the PSCNC program with your particular milling machine. 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:

Parameter	*Description*
comport	Use 'COM1:', 'COM2:', 'COM3:', or 'COM4:' to select the PC com port you intend to use.
xaxis, yaxis, zaxis	Module addresses used for each of the axes (1, 2 or 3 for each axis)
xiotype, yiotype, ziotype	These parameters specify which type of ***PIC-SERVO*** controller or driver is being used for each axis. You should use the following values: ***PIC-SERVO CMC***: 12 ***PIC-SERVO SC*** with PWM & Direction output: 0 ***PIC-SERVO SC*** with 3-Phase output: 16 ***PIC-SERVO SC*** with Antiphase PWM output: 32
bufsize	The maximum number of path points to be stored in each ***PIC-SERVO***. A *bufsize* 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.
xaccel, yaccel, zaccel	Acceleration values to use for each axis, in inches (or millimeters) per second per second.
xmaxvel, ymaxvel, zmaxvel	Maximum velocities for each axis in inches (or millimeters) per second. 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.)
xmax, ymax, zmax	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.
xmin, ymin, zmin	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.
xkp, ykp, zkp	Proportional gains used by the servo controllers. (Range: 0 - 32,767) These values should be determined using the NMCTest program.
xkd, ykd, zkd	Derivative gains used by the servo controllers. (Range: 0 - 32,767) These values should be determined using the NMCTest program.
xki, yki, zki	Integral gains used by the servo controllers. (Range: 0 - 32,767) These values should be determined using the NMCTest program.
xil, yil, zil	Integration limits used by the servo controllers. (Range: 0 - 32,767) These values should be determined using the NMCTest program.
xol, yol, zol	Output limits used by the servo controllers. (Range: 0 - 255) These values should be determined using the NMCTest program. (Usually set to 255)
xcl, ycl, zcl	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.)
xel, yel, zel	Position error limits, in encoder counts, used by the servo controllers. (Range: 0 - 16,383) These values should be determined using the NMCTest program.
xsr, ysr, zsr	Servo rate divisors used by the servo controllers. All should be set to 1.
xdc, ydc, zdc	Deadband compensation used by the servo controllers. (Range 0 - 255) Usually set to 0 except for when using SS-Drive controllers.

xpadv, ypadv, zpadv	Phase advance settings used by SS-Drive type controllers. Set to 0 for all other types of ***PIC-SERVO*** controllers.
xpoff, ypoff, zpoff	Phase offset settings used by SS-Drive type controllers. Set to 0 for all other types of ***PIC-SERVO*** controllers.

2.4 Software Setup: the Tool Description File

The PSCNC program supports tool length compensation for up to 20 tools. G-Code programs will us an "M06 T*nn*" command to indicate a tool change, and a "G43 H*nn*" 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

3.0 Using *PIC-SERVO CNC*

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 both the logic supply and the motor power supply.

Next, you will be asked if you want to "home the mill". The homing procedure will move each axis slowly up against its positive limit switch, and then move all axes back to the most positive end of the 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.

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 and servo enable controls, 2) the jogging control panel, 3) the G-Code display window, and 4) the G-Code execution controls.

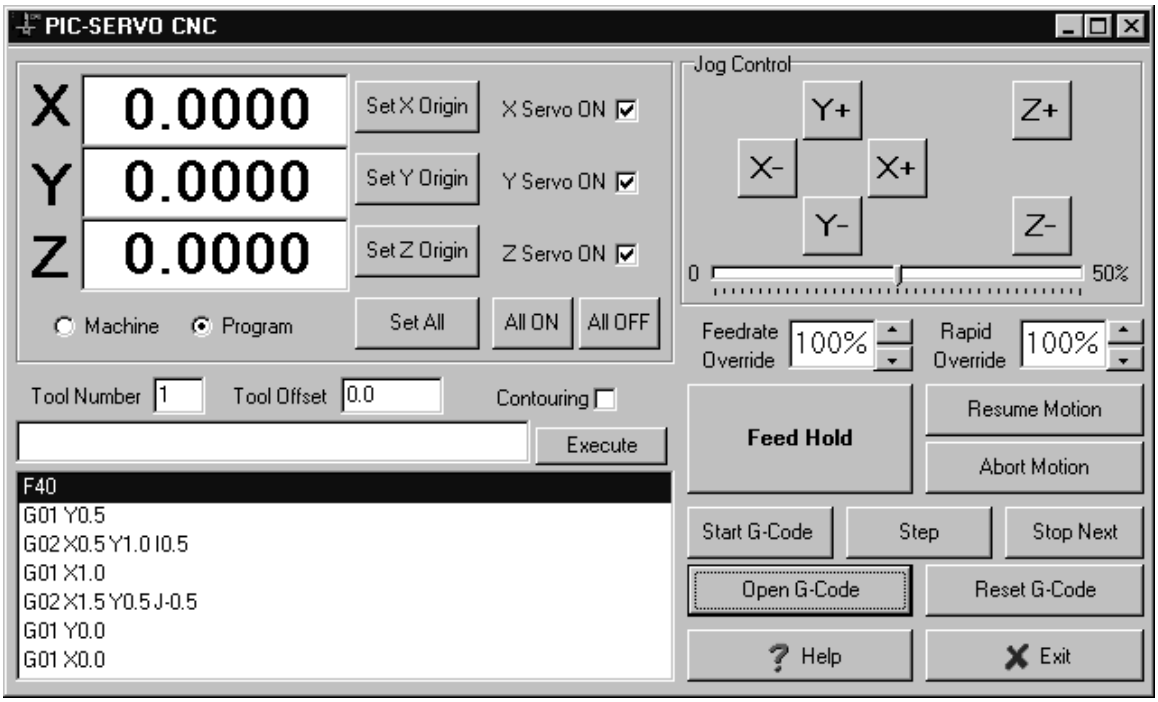

Figure 2 - PSCNC Control Panel.

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.

Note that the Z program coordinate display 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.

The digital readout panel also has controls for enabling or disabling the servo controllers. Even with the servo controllers disabled, the current motor positions will be displayed. If your milling machine has hand cranks, it is often most convenient to disable the servos and position the mill by hand for setting the origins. Once in position, the servos for any or all axes can be re-enabled, and the origins set as needed.

When a servo is disabled (either manually or due to a servo error condition) the position will be displayed with a grey background. It will be displayed with a white background when the servo is enabled.

If the milling machine has not been homed, the program will start off with all servos disabled, along with most of the other controls. To enable the servos, you must click on each "Servo On" check box, or click on the "All On" button to enable all servos.

Manual Jog Controls

The manual jog controls allow you to move individual axes at a constant velocity. Down-clicking and holding one of the buttons will move the corresponding axis in the positive or negative direction. 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.

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 1 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 inched (or millimeters) per *minute*.

Examples:

```
G01 X1.0 Y2.0 F30.0  (moves in a straight line to X=1.0, Y=2.0 with)
                     ( a feed rate of 30 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. 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 an 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      )
```

G03 Counterclockwise Circular Arc

Moves along a counterclockwise circular arc to the specified X, Y and Z coordinates at the specified feed rate. 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 counterclockwise 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 Y1.0 I-1.0 (creates an 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)
```

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)
```