Along the G-Code Way: Rotary Angles

Although G-Code can calculate the coordinates of points located on a circle and elminate the need for a horizontal rotary table, it can't produce the spray nozzles seen in Photo 1. The Z-axis spindle of an ordinary vertical milling machine moves only vertically (hence the name!), which means that drilled holes must be perpendicular to the XY plane of the milling table: it can't drill at an angle.

When you need only a few holes at a few angles to the XY plane, you can reclamp the part for each hole. However, setting up each angle requires either a specialized fixture or some tedious adjustment, either of which would be completely unworkable for a dozen holes around a cylindrical nozzle.

In this column, I'll explain how to produce those nozzles by combining some G-Code loops with the rotary table setup in Photo 2. As before, you'll see how changing a few G-Code parameters can produce a wide variety of different nozzles with very little effort.

Dimensional Constraints

This project arose some years back when our water well began coughing up a disturbing amount of reddish grit and gunk. Among other problems, the well was inhabited by iron bacteria which, while harmless to humans, convert dissolved iron into slime. The first and simplest treatment was to chlorinate the well, washing the entire length of the casing from the top down. The usual recommendation involve lowering a garden hose into the well while recirculating a bleach solution.

However, the well head was deep inside a pit with a single 1/2-inch NPT pipe plug presenting the only access without major disassembly. I found that 1/2-inch ID PEX tubing had a 5/8-inch OD that barely fit through the opening, so I decided to put a spray nozzle at the end of a few hundred feet of PEX, with a garden hose adapter on the other end.

Photo 1: The original spray nozzle at the top of this picture used clunky copy-and-paste G-Code to create the holes. The other two use the looping G-Code presented in this column. (dsc05142 - Sample Nozzles.jpg)

I made the top nozzle in Photo 1 and attached it to the roll of tubing with standard fittings. PEX fittings use external compression clamps that were too large to pass through the pipe plug hole, so I made a compession band that fit around the tubing, inserted the fitting, heated the finished nozzle until it was soft, and squashed the clamp. That deformed the tubing around the fitting barbs enough to produce a mechanically secure joint and, after the tubing cooled, I removed the band. Unlike home plumbing, a little leakage doesn't matter inside the well!

PEX tubing comes in a large roll that puts a pronounced curve in even a short length. I forced the nozzle blank over an aluminum rod chucked in the lathe, started the lathe at low speed, and applied a hot-air gun until the curve relaxed. The two bottom nozzles in Photo 1 still have a slight curve, but they're samples I made for this column. If you have tighter specs, a sacrificial mandrel would provide better results.

I center-drilled a chunk of brass to support the right end of the tubing in the tailstock, as shown in Photo 3. Aligning the rotary table parallel to the X axis and setting the tailstock exactly on center are just as tedious as when you'll be doing manual milling.

However, Photo 2 shows that the rotary table's handwheel points to the rear of the mill, where it's completely inaccessible. The usual Sherline angle plate setup puts the rotary table on the right end of the XY table with the handwheel pointing forward, but in this case I won't be turning the crank: G-Code will do that for me. The motor-to-column clearance wasn't a problem with this project, but it could keep the spindle from getting close enough to the rotary table surface for some operations.

With the nozzle blank in place, the G-Code can start drilling... after we figure out where the holes should go.

Computing the Holes

The first few lines in Listing 1 define the number of holes in each row, the number of rows, and the overall

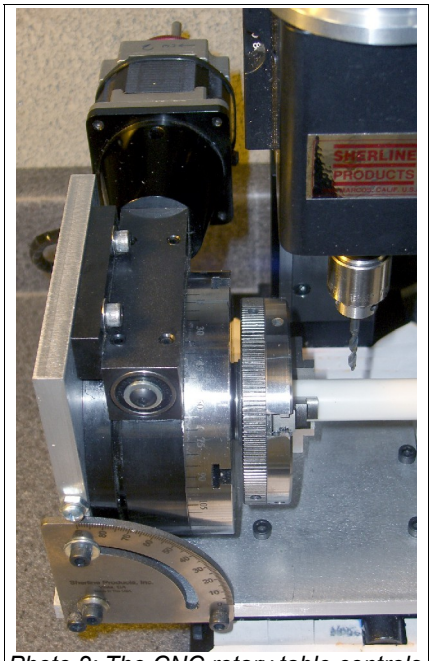

Photo 2: The CNC rotary table controls the angular position of the PEX tubing clamped in the three-jaw chuck. G-Code controls the motor, so the handwheel need not be accessible from the front of the setup. (cimg4575 - Vertical Rotary Table Setup.jpg)

Ed Nisley
ed.08.nisley@pobox.com

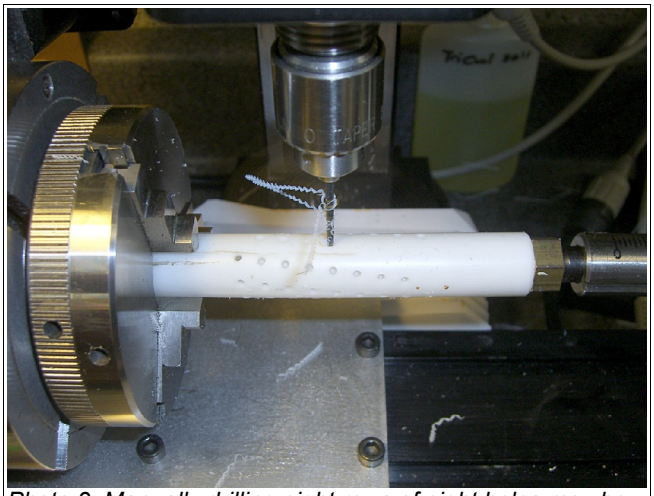

Photo 3: Manually drilling eight rows of eight holes may be a tedious project, but it requires just a few lines of G-Code. (cimg4515 - Drilling radial holes.jpg)

length of the hole pattern. The Sherline mill has about nine inches of X-axis travel, but the rotary table, chuck, and tailstock limit the maximum workpiece length to about seven inches. The `StartOffset` parameter defines the starting location for the holes along the X axis; it's your responsibility to ensure that the G-Code won't try to drill holes in the tailstock!

Given that information, the G-Code can compute the spacing between holes. The `XIncr` parameter gives the distance between hole centers along the X axis, while `XOffset` shifts successive rows so that no two holes line up around the tubing circumference.

The angular distance between rows, `ASpace`, is simply 360 degrees divided by the number of rows. For example, two rows will be separated by 180 degrees and five rows by 72 degrees. There's no need to worry about weird fractions: EMC2 can easily space seven rows 51.42857 degrees apart.

`AIncr` sets the angular space between successive holes in each row to the angle between the rows divided by the number of holes. Incidentally, if you want the helix of holes to wind counterclockwise, just change the sign of `ASpace` or `AIncr` and EMC2 will handle all the arithmetic for you.

Pop Quiz: compute the values of all those parameter for a nozzle with seven rows of five holes. Then fill out a table giving the X-axis and A-axis coordinates for each hole.

Although PEX tubing sizes are standardized, the inside and outside diameters are not tightly controlled to machine-shop levels. Worse, the tubing may become slightly out-of-round after being stored in a large coil, so `TubingOD` and `TubingID` define the measured maximum outside and inside diameters.

Because the tubing may remain slightly bowed after straightening on the mandrel, `TraverseMargin` sets a safe distance beyond the OD for rapid movements between the holes. The G-Code then computes `TraverseZ`, the actual Z-axis coordinate, from the tubing OD and the required clearance.

Not knowing much about hydrodynamics, I decided that the total area of all the holes should equal the area of the tubing's ID. A bit of pencil pushing showed that the required drill diameter was simply the tubing ID divided by the total number of holes, which the G-Code computes and stores in the `DrillDia` parameter. Tiny holes might make more sense in a metal nozzle, but I added an `IF-ENDIF` statement to enforce a minimum 1/8" drill size.

You can use the same logic to set minimum or maximum limits on other values: RPM, speeds-and-feeds, and computed dimensions.

The last few lines in Listing 1 compute the drill feed according to the rule-of-thumb that fixes the spindle RPM at 3000 and the feed at 100 times the drill diameter in either inches or millimeters. The Sherline doesn't have nearly enough power to push drills much larger than ¼ inch at that pace, but it does surprisingly well. Obviously, you must do some experimentation with your own equipment and materials.

With all those values in hand, it's time to actually drill some holes!

```
#<_HolesPerRow>           = 8
#<_NumRows>               = 4
#<_RowLength>             = 2.0
#<_StartOffset>           = 0.0

#<_XIncr>                 = [#<_RowLength> / #<_HolesPerRow>]
#<_XOffset>               = [#<_XIncr> / #<_NumRows>]

#<_ASpace>                = [360 / #<_NumRows>]
#<_AIncr>                 = [#<_ASpace> / #<_HolesPerRow>]

#<_TubingOD>              = 0.625
#<_TubingID>              = 0.465

#<_TraverseMargin>        = 0.1            (clearance beyond OD)
#<_TraverseZ>             = [[#<_TubingOD> / 2 ] + #<_TraverseMargin>]

#<_DrillDia>              = [#<_TubingID> / [#<_HolesPerRow> * #<_NumRows>]]

O900 IF [#<_DrillDia> LT 0.125]
#<_DrillDia>              = 0.125
O900 ENDIF

#<_DrillFeed>             = [100 * #<_DrillDia>]
#<_DrillRPM>              = 3000
#<_DrillZ>                = [[#<_TubingID> / 2] - [1.5 * #<_DrillDia>]]
```

Listing 1: These G-Code statement define the basic geometry of the nozzle. The program first computes DrillDia as the diameter that makes the area of all the holes equal to the tubing ID. The IF statement ensures that DrillDia is never smaller than 0.125 inches.

ed.08.nisley@pobox.com

Drilling Patterns

The EMC2 version of the RS274 language recently acquired a `REPEAT-ENDREPEAT` command that executes a sequence of G-Code statements a specific number of times. The G-Code you've seen in my previous columns used `DO-WHILE` loops for the same purpose. You can use either, but `REPEAT` works well when a parameter holds the exact number of iterations.

The G-Code in Listing 2 consists of two nested loops. The outer loop iterates once for each row of holes, as determined by the `NumRows` parameter, while the inner loop iterates once for each hole of the row, as set by `HolesPerRow`. As a result, the statements inside both loops execute once for each hole in the nozzle pattern: a single G81 statement drills all the holes.

The `RowIndex` parameter keeps track of the current row, while `HoleIndex` identifies the current hole within the row. As usual, both values start at zero to simplify the coordinate calculations. You could use these parameters in the `WHILE` part of `DO-WHILE` loops, as I've done in the past, but now they're simple counters.

The X-axis coordinate of each hole is the sum of three values: the starting point for the entire set of holes, the starting offset for the current row, and the hole's position within the row. The X-axis origin lies just to the right of the three-jaw chuck in Photo 2, at a spot where the rotary table doesn't collide with the spindle head.

The A-axis coordinate is sum of two values: the starting angle for the current row and the hole's position within the row. The A-axis orgin is arbitrary, because the tubing is a simple cylinder with no distinguishing marks.

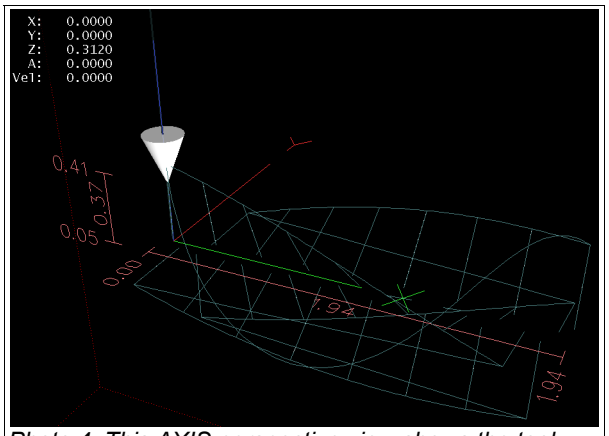

Photo 4: This AXIS perspective view shows the tool path for four rows of eight holes. The drill point is aligned at Z = TubingOD/2, the surface of the workpiece, just below the TraverseZ level. (Screenshot – Perspective - cropped.png)

Both the Y- and Z-axis origins are at the rotary table's center, which is the center line of the PEX tubing.

You could drill offset holes, perhaps to make a spinning nozzle, by drilling at a nonzero Y coordinate, but I'll leave that as an exercise. Hint: use an end mill to put a small flat in the tubing at each drill site so the drill bit doesn't skate away.

Photo 4 shows the AXIS perspective backplot of a nozzle with four rows of eight holes. You must add the `GEOMETRY = AXYZ` line to your machine's `INI` file to tell AXIS how to display angular motions in the backplot; the seting varies depending on the rotary table's alignment.

Photo 5 is the view looking toward the origin along the X axis, with the Z and Y axis markers sticking out to the top and right. This clearly shows the depth of each drill stroke

```
G20                             (inch units)

(msg,Verify origin X=0 and Z=0 at centerline)
M0
(debug,Load #<_DrillDia> drill, set #<_DrillRPM> rpm, hit Resume)
M0

G0 X0
G0 Z #<_TraverseZ>

F#<_DrillFeed>

#<RowIndex> = 0
O1000 REPEAT [#<_NumRows>]

#<HoleIndex> = 0
O1100 REPEAT [#<_HolesPerRow>]

#<XCoord> = [#<_StartOffset> + [#<RowIndex> * #<_Xoffset>] + [#<_XIncr> * #<HoleIndex>]]
#<ACoord> = [[#<RowIndex> * #<_ASpace>]  + [#<_AIncr> * #<HoleIndex>]]

G0 X#<XCoord> A#<ACoord>        (traverse to next hole)

G81 Z#<_DrillZ> R#<_TraverseZ>  (drill it)

#<HoleIndex> = [#<HoleIndex> + 1]
O1100 ENDREPEAT

#<RowIndex> = [#<RowIndex> + 1]
O1000 ENDREPEAT

G0 Z#<_TraverseZ>               (ensure clearance)
G0 X0 A0                        (unwind A axis)

M2
```

Listing 2: A pair of nested loops drill the holes along each row while turning the workpiece. The G-Code computes the hole locations from a few parameters in Listing 1.

ed.08.nisley@pobox.com

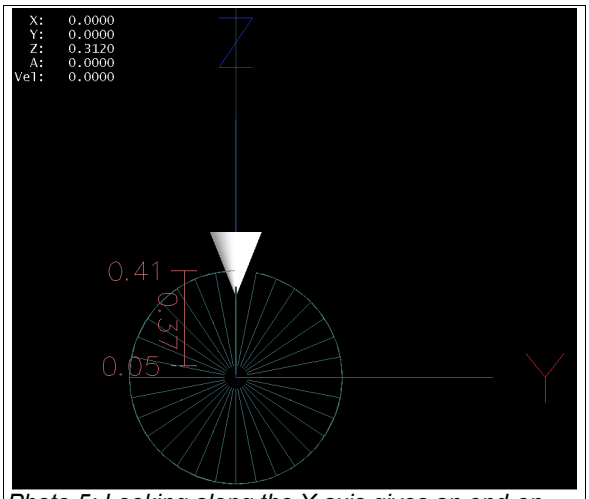

Photo 5: Looking along the X axis gives an end-on view of the 32 drill paths, equally spaced along four rows of eight holes each. The tool position is the same as in Photo 2. (Screenshot - Along X Axis - cropped.png)

toward the X axis at the middle of the tubing.

Angular Optimzation

The order of the two loops in Listing 2 is important, because the rotary table moves much more slowly than the linear axes. The actual machining times depend on the acceleration limits for the mill, but the program took about two minutes as shown and 3.5 minutes after simply interchanging the two loops.

EMC2 does not wrap A-axis values from 360 degrees back to 0, so commanding `G0 A360` when the current A coordinate is 0 will wind the table one complete revolution. That EMC2 design decision makes perfect sense, but you must account for each complete rotation and adjust the angle values accordingly.

For example, the `ACoord` computation in Listing 2 produces A-axis coordinates of 0, 90, 180, and 270 for a four-row nozzle. When you simply interchange the two loops, the rotary table will "unwind" from 270 back to 0 after drilling four holes, rather than "wind" forward the shorter distance to the same point at 360 degrees.

However, if you modify the code to always increment the angle, perhaps by introducing a new `AngleIndex` parameter that increments for each hole, the A axis will rotate 2880 degrees after drilling eight holes in each of four rows. The code moves back to X=0 and A=0 at the end of the program, so the rotary table will make eight complete and utterly unnecessary rotations!

Homework: modify the code as described, then try to optimize the results. Hint: `G92` may be your friend.

Because EMC2 can set any angle you want, it eliminates indexing plates, hole counting, and fiddly sector arms for the rotary table. Even better, it won't get distracted and lose track of which hole it just drilled midway through a job.

Go forth and rotate numerically!

Temporary Stop: M0

As it turned out, we decided to abandon the well and attach our house to the town water supply, so I never got to use that nozzle. Trenching the front yard was a major undertaking: I was happy to watch it happen from the living room.

Contact me at ed.nisley@ieee.org. Add "Digital Machinist" to your email subject line to help avoid spam filters along the way.

Sources & Resources

Downloadable file with the complete programs `DM5.1-Nisley Download.zip`
Enhanced Machine Controller project `www.linuxcnc.org`
Iron bacteria: `http://en.wikipedia.org/wiki/Iron_bacteria`
PEX tubing: `http://www.ppfahome.org/pex/index.html`