```
-- SheetCAM postprocessor for bLouChip MillRight MegaV XL CNC Plasma machine.
-- This postscript was copied and modified from SheetCAM's GRBL plasma.scpost V7.0.19 std file.
-- Used and tested with GRBL v1.1i XYZA axis machine.
-- ================================================================================================

-- LDC vars

LdcVersion = "20.3-beta"

--[[ SheetCAMs Rotary-plasma plugin support has been added, but it is NOT required.
If you don't use the Rotary-plasma plugin, that is it is not an "Enabled" plugin, then
you should comment out the Rotary Plasma and Helper library stmts at the end of this scpost, as such:
   --package.path = sc.Globals:Get().thisDir .. "/plugins/RotaryPlasma/?.lua"
   --require("rotaryhelper")
If you do use the Rotary-plasma plugin, then those stmts are required.
Know that you can have the Rotary-plasma plugin "Enabled" all the time and just
select "Flat Sheet" in the plugin dialogue to run a 2D job, ie rotary is not used.

I have  tested this scpost with Rotary-plasma plugin for Flat Sheet, Round Pipe,
and Rectangle Tubing thus far, but pretty minimal testing, so be careful with use.
I have also tested it without the plugin enabled.
Please report problems in https://forum.sheetcam.com/ or send me a PM.

***  IMPORTANT ROTARY NOTE:
***  ZERO Z TO THE SURFACE OF THE CYLINDER PRIOR TO JOB START, NOT THE CENTER OF ROTATION.
***  For Rectangles, zero Z on the apex of a corner rotated to top dead center. In other words,
***  that is the maximum radius for the object.

Global vars defined here only get initialized at scpost load time.
By contrast, those vars in OnInit() function down below get intitalized at evey post process runtime.
]]

--[[ Air Supply Time, in minutes, is handy if your air compressor is under-sized for
for the plasma torch air flow demands during continuous runtimes. Set this to the maximum
continuous demand time on the air supply before the tank pressure falls to low to supply
the correct PSI to the torch. When this timer hits, the plasma job will Pause allowing
the operator to Resume the job via controller sw Cycle Start button or a physical Resume button
on the table control panel. The job will also Pause between SC Part cutouts, allowing the operator
to reach into the table to remove a cut part if so desired.
]]
LdcAirSupplyTime = 5.0  -- default minutes, air supply time to maintain continuous cutting.

--[[ The feedrate and shape cut lengths are calculated as a time and a timer in this scpost
accummulates those values to plant the grbl Pause stmt. In order to get somewhat accurate
timings, set the Torch Touchoff Time in minutes (seconds/60) to the time it takes your machine
to run through a pre-cut torch zeroing touchoff cycle.
]]
LdcTorchTouchoffTime =  8/60  -- minutes, seconds/60 if that is easier
LdcGrblPauseStmt = "M0"  -- use the Job Pause stmt for your controller
LdcThcONcmd  = "M7 (enable Z THC and anti-dive circuit)"
LdcThcOFFcmd = "M9 (disable Z THC and anti-dive circuit)"

--rotary.on
-- these global vars are copied from "Mach3 Rotary.scpost"
--  my understanding is that these vars are global to Rotary-plasma plugin and helper (p/h) macros.
rotaryAxis = " A"  -- usually A, include leading blank for readbility in the gcode file
unitsPerRev = 360  --Number of units(usually degrees) for one full revolution of the rotary axis
--rotary.off

--[[ The following variables can be defined external to this .scpost in SheetCAM Variables
Either in Operations using Set Variables op or via the program menu bar
options/window tabs of Options->JobOptions->Variables or Options->ApplicationOptions->Variables.
The defaults for these variables if not defined in the above Options can be seen above and in the
function OnInit() below. All values in this .scpost code are in mm, but this post works fine in
either mm or inches as defined in the gcode Output File Units of Metric or Inches in
Options->Machine->Post Processor.
============ External variables are described as:
]]

-- the gravity slide travel distance of the torch holder from top of material to actuating the probe switch
-- This is also referred to as the float distance on a floating torch mount.
post.DefineVariable("LdcTorchProbeTravel", sc.unitLINEAR, 1.0, 10.0)

-- causes arcs to be coded with short line segments chained together, replaces G2/G3 motion with G1 motion.
-- typically used if G2/G3 motion causes grbl error 33 or the like. The value specifies the line segment length.
post.DefineVariable("LdcArcAsMoves", sc.unitLINEAR, 0, 0.200)

--[[ THC present on machine or not.
On my CNC plasma table, this option ON(1) produces M4, S1000 (100% Spindle power),
M7, and M9 grbl stmts. When this option is OFF(0), the post produces M3 S1000 and M5 stmts.
Note that M7 grbl stmt is typically availabe by default only on 4 axis controllers
such as Atmega2560 or Teensy4.1 since they have extra I/O pins.
The 3 axis controllers such as Arduino Uno should use M8 if avaiable.
My machine uses the M4, Snnnn, M7, and M9 in conjunction with grbl config $32=1 (laser mode)
and a custom THC Z Anti-Dive circuit. The anti-dive circuit
prevents the THC from diving the torch into the work material during
G1 XY cornering and start/stop of cuts, basically anytime the feedrate due to acceleration/deceleration
falls below 85% of programmed value. Grbl manages this automatically when in Laser Mode and firing with M4.
My custom THC Z Anti-Dive circuit schematic is here:
   https://drive.google.com/file/d/1UKgOIl-9XRypoTO2jaWg7nsvEHAYx1RG/view?usp=sharing
```

```
The custom anti-dive circuit works in conjunction with a Proma Compact THC SD unit.
]]
post.DefineVariable("LdcThcPresent", sc.unit0DECPLACE, 0, 1)

--[[ THC enable ON(1) or OFF(1). This option is secondary to LdcThcPresent, thus LdcThcPresent must be YES(1).
This option allows for dynamic control of the THC if your controller supports that function.
Thus, if you'd like to turn off the THC for an Operation, use the SC Set Post Processor Variable operation
to change the value of this option.
My machine's THC enable is managed by the grbl Spindle Speed (S), when S850 or lower is in effect, the THC
is effectively OFF, greater than S850 is ON. The THC enable is also dynamically managed via the Spindle PWM
signal during runtime via the custom THC Z Anti-Dive circuit referenced above, provided S1000 stmt has been
issued and is in effect.
]]
post.DefineVariable("LdcThcEnable", sc.unit0DECPLACE, 0, 1)

-- similar to Path Rules 'On small shape' action, but allows for further customization of actions
-- The value used is the length of the small shape cut including leadin and leadouts.
post.DefineVariable("LdcSmallShape", sc.unitLINEAR, 0, 100)

-- a switch which allows testing gcode on a grbl controller not connected to a physical CNC machine
post.DefineVariable("LdcBenchTest", sc.unit0DECPLACE, 0, 1)

post.SetOptions(post.ARC_SEGMENTS) -- limit arc moves to 180 deg -- LDC
post.ForceExtension("nc")

function OnAbout(event)
   ctrl = event:GetTextCtrl()
   ctrl:AppendText("GRBL Plasma XYZA post processor, **LDC Customized\n")
   ctrl:AppendText("  Requires Rotary-Plasma plugin for Rnd Pipe and Rectangle tubing.\n")
   ctrl:AppendText("  But for Flatsheet 2D cutting, this post works WITH or WITHOUT the plugin.\n")
   ctrl:AppendText("  Works with or without a THC.\n")
   ctrl:AppendText("  Annotated and easily adaptable to your specific machine.\n")
   ctrl:AppendText("Rotary Z0 at max OD (surface), NOT rotary center line; thus works with short Z motion.\n")
   ctrl:AppendText("M4 or M3/M5 turn the torch on/off\n")
   ctrl:AppendText("Arc as Moves (G1) or Incremental IJ (G2/G3)\n")
   ctrl:AppendText("  ** external post variables\n")
   ctrl:AppendText("LdcTorchProbeTravel -- default mm, same as floating torch mount float distance\n")
   ctrl:AppendText("LdcThcPresent -- default, THC present on machine = 1, not present = 0\n")
   ctrl:AppendText("LdcThcEnable  -- default, THC enable ON = 1, OFF = 0, can be used for dynamic THC control at runtime\n")
   ctrl:AppendText("LdcArcAsMoves -- default mm, use 0 for G2/G3 arcs, else specify the line segment length for G1 motion\n")
   ctrl:AppendText("LdcSmallShape -- default mm, use 0 for no small shapes; not needed w THC Z Anti-Dive\n")
   ctrl:AppendText("LdcBenchTest  -- default OFF = 0; used for testing on grbl without a machine\n")
end

function OnInit()
   post.SetCommentChars ("()", "[]")  --make sure ( and ) characters do not appear in system text

   -- LDC job init block
   -- external post variables default values
   LdcTorchProbeTravel = 8.2
   LdcThcPresent = 1
   LdcThcEnable = 1
   LdcArcAsMoves = 0
   LdcSmallShape = 0
   LdcBenchTest = 0

   -- internal processing vars
   LdcShapeNum = 1
   LdcThcOn = false
   LdcSmallShapeCount = 0
   LdcCuttingTime = 0 -- the per cut time accumulater, the timer
   LdcOpFeedrate = 0
   LdcLastPartIndex = 0
   LdcAbsCutHeight = 1e+30
   LdcMaxRotRad = 0  -- maximum rotary radius, can be approx. 1/2 diagonal of rectangles and I-beams
   LdcIsRotaryPlugin = false
   LdcIsRotaryJob = false
   LdcRotZ0Surface = true
   LdcDidPenUp = false
   LdcProbeCmd = "G38.2 Z"  -- set default, report probe error if limit sw not made
   LdcFirstSafeZ = true

--rotary.on
   -- endA is a global var instantiated in the Rotary-plasma p/h, thus if it exits, the plugin is enabled.
   if endA ~= nil then
      LdcIsRotaryPlugin = true
      -- BeamType 0 = flatsheet, 1 = Rectangle, 2 = Round Pipe, 3 = I-Beam, 4 = Custom
      if var.BeamType > 0 then LdcIsRotaryJob = true end
   end
--rotary.off

   -- build a gcode file comment header section...
   post.Text("; Post: '", postName, "' V", LdcVersion, "\n")
   post.Text("; GCode file: '", fileName, "'\n")
   post.Text("; GCode datestamp: ", date, "- ", time, "\n")

   -- init gcode initial state and retract motion...
   post.Text("G00 G17 G40 G54 G90 M5 M9 F0\n")
   FRformat = "0.##"  -- feedrate format
   if scale == metric then
```

```
      post.Text ("G21  (metric mode)\n")
      format = "0.000"  -- alt format "0.0##";  accuracy to 3 decimals mm necessary to avoid grbl error33 on arcs
   else
      post.Text ("G20  (inch mode)\n")
      format = "0.0000"  -- alt format "0.0###";  accuracy to 4 decimals in inch mode
   end

   -- begin gcode block comment header...
   post.Text("; ***  ATTENTION OPERATOR  ***\n")
   post.Text(";   Be sure to Zero Z (torch tip) to top of material\n")

   if LdcIsRotaryJob then
--rotary.on
      -- get the cylinder radius or maximum radius of the rectangle, which is the apex
      -- of the corner including radius of corner. This position in the drawing should always be materialY1.
-- normally qryCHKEND() function does not need to be called in a scpost. I'm doing it here because I need the
-- maximum radius due to using/assuming Z0 as the surface of the rotary object, LdcMaxRotRad is the maximum rotary radius
-- used throughout my scpost as a correcting Z offset to endZ (set by SC with an assumed Z0 of center of rotation).
-- Very important: qryCHKEND() always takes the Y axis argument which is the Y position on the flat drawing, NOT the Y position
-- on the wrapped rotary object; it returns the radius to the object surface at that Y flat drawing position.
      LdcMaxRotRad = sc.QueryDll(qryCHKEND, materialY1, dllId)
--rotary.off

--rotary.ON optional
      post.Text("; Rotary Job -- ")
      if var.BeamType == 1 then
         post.Text("RECTANGLE TUBING ... Ready for machine testing\n")
         post.Text(";   Beam Height: ", post.FormatNumber(var.BeamHeight * scale, format),
                   ", Beam Width: ", post.FormatNumber(var.BeamWidth * scale, format),
                   ", Corner Radius: ", post.FormatNumber(var.CornerRadius * scale, format), "\n")
         post.Text(";   Wall Thickness: ", post.FormatNumber(var.WallThickness * scale, format),
                   ", Map Drawing To Inside: ", var.MapDrawingToInside, "\n")
         post.Text(";   WCS Zero setup:  Y0.00 A0.00 is the centerline of Rectangle Width\n")
         post.Text(";                     X0.00 is left edge of tubing\n")
         post.Text("; *** ZERO Z ON APEX OF TUBING CORNER ***\n")

         --var.LdcThcPresent =
         --var.LdcThcEnable =
         post.Text("; ***  OPERATOR: Be aware that this job contains Z motion during cutting.\n")
         post.Text("; ***            Adjust your THC use accordingly. \n")
      end
      if var.BeamType == 2 then
         post.Text("ROUND PIPE ... Ready for machine testing\n")
         post.Text("; *** ZERO Z ON SURFACE OF CYLINDER ***\n")
         post.Text(";   Pipe Diameter: ", post.FormatNumber(var.PipeDiameter * scale, format),
                   ", Wall Thickness: ", post.FormatNumber(var.WallThickness * scale, format), "\n")
         post.Text(";   Map Drawing To Inside: ", var.MapDrawingToInside, "\n")
      end
      if var.BeamType == 3 then
         post.Text("I-BEAM ... WARNING- NOT TESTED\n")
         post.Text(";   Beam Height: ", post.FormatNumber(var.BeamHeight * scale, format),
                   ", Beam Width: ", post.FormatNumber(var.BeamWidth * scale, format),
                   ", Web Thickness: ", post.FormatNumber(var.WebThickness * scale, format), "\n")
         post.Text(";   Flange Thickness: ", post.FormatNumber(var.FlangeThickness * scale, format),
                   ", Map Drawing To Inside: ", var.MapDrawingToInside, "\n")
      end
      if var.BeamType > 3 then
         post.Text("Custom setup, values unknown. WARNING- NOT TESTED\n")
      end
      post.Eol()
   end
--rotary.OFF

   post.Text("G28 G91 Z0  (retract Z for start of job)\n")
   post.Text("G90\n")
   --  alternate Z retract commands for start of job...
   --post.NonModalNumber("G0 Z", safeZ * scale, format)
   --   post.Text(" (retract Z for start of job)\n")

   post.Text(LdcGrblPauseStmt, "  (******* MACHINE PAUSE ********)\n")
   post.Text("; ****  OPERATOR:\n")

--rotary.ON optional
   if LdcIsRotaryJob then
      post.Text("; ****  Check that Maximum Radius of rotary object is: ",
         post.FormatNumber(LdcMaxRotRad * scale, format), "\n")
   end
--rotary.OFF

   post.Text("; ****  About to move XYZA to safeZ and first pierce position\n")
   post.Text("; **  Press Cycle Start or Resume button when ready to resume job\n")

   if var.LdcTorchProbeTravel ~= nil then
      LdcTorchProbeTravel = var.LdcTorchProbeTravel  -- set default from Options Vars
   end
   post.Text("; Torch Probe Travel = ",
      post.FormatNumber(LdcTorchProbeTravel * scale, format), "\n")

   if var.LdcThcPresent ~= nil then
      LdcThcPresent = var.LdcThcPresent  -- set from Options Vars
```

```
   end
   post.Text("; THC Present on machine, YES(1)/NO(0): ", LdcThcPresent, "\n")

   if (var.LdcThcEnable ~= nil) then
      LdcThcEnable = var.LdcThcEnable  -- set default from Options Vars
   end
   post.Text("; Dynamic THC Enable ON(1)/OFF(0) status: ", LdcThcEnable, "\n")

   if var.LdcAirSupplyTime ~= nil then
      LdcAirSupplyTime = var.LdcAirSupplyTime  -- set default from Options Vars
   end
   post.Text("; Air Supply Time: ",
      post.FormatNumber(LdcAirSupplyTime, "0.0"),  " minutes\n")

   if var.LdcATorchTouchoffTime ~= nil then
      LdcTorchTouchoffTime = var.LdcTorchTouchoffTime  -- set default from Options Vars
   end
   post.Text("; Torch Touchoff Time: ",
      post.FormatNumber(LdcTorchTouchoffTime, "0.000"), " minutes\n")

   if var.LdcArcAsMoves ~= nil then
      LdcArcAsMoves = var.LdcArcAsMoves  -- set default from Options Vars
   end
   post.Text("; Arcs As Moves = ",
      post.FormatNumber(LdcArcAsMoves * scale, format), " in gcode units\n")

   -- Path Rules "On small shape" has some side effects in that it changes the
   -- LeadIn style from Ramp to Plunge, and its confusing as to what defines a small shape.
   -- Thus, its better to implement small shape handling in this post if its ever needed.
   if (var.LdcSmallShape ~= nil) then
      LdcSmallShape = var.LdcSmallShape  -- set default from Options Vars
   end
   post.Text("; LDC Small Shape Size = ",
      post.FormatNumber(LdcSmallShape * scale, format), "\n")

   if  var.LdcBenchTest ~= nil then
      LdcBenchTest = var.LdcBenchTest
      if LdcBenchTest > 0 then
         post.Text("; *** BENCH/DESKTOP TESTING MODE ONLY, NO CNC MACHINE NEEDED ***\n")
         post.Text("; *** TORCH TOUCHOFF AND Z ZERO NOT ACCURATE, PROBE SWITCH NOT REQUIRED ***\n")
      end
   end
   post.Eol()
   -- end of gcode file comment header block
   -- end LDC

end


function OnFinish()
   post.Text ("S0 \n")
   post.Text ( LdcThcOFFcmd, "\n")
   post.Text ("G92.1 \n")
   post.Text ("; LdcSmallShape Count = ", post.FormatNumber( LdcSmallShapeCount, "0"), "\n")
   post.Text ("M5 M9 M30 \n")
   post.Eol()
end


function OnRapid()
--rotary.on
-- I'm calling this whole function as rotary specific for reasons noted:
-- this stmt suppresses an extra safeZ motion at the beginning of the job, I don't like it because this scpost
-- has already parked Z at the G28 Z0 position is max retraction for clearance on my table.
   if LdcIsRotaryPlugin and LdcFirstSafeZ then LdcFirstSafeZ = false return end

-- be sure to use ModalNumber() for posting all axes, this prevents most cases of otherwise repeated axis posts when there
-- is no motion for that axis. Also, be sure to use the same number formatting in all cases of the axis word in other functions
-- else you may get repeat axis words and values.
   post.Text ("G0")
      post.ModalNumber (" X", endX * scale, format)
      post.ModalNumber (" Y", endY * scale, format)

      if LdcIsRotaryJob and LdcDidPenUp then
         -- in cases of rotary rectangles, endZ depends on the angle of tube due to its corners.
         -- In cases of motion to safeZ, SC makes it relative from endZ.
         -- Although normally this is sufficient, it can be a close clearance; then adding the
         -- potential for THC induced Z error, this SC caluclated safeZ/endZ dimension
         -- may not clear the diagonal (corner) of subseqent rotational motion due to accumulated THC Z error.
         -- Thus a flag was set upon exit from OnPenUP() to know when a safeZ should be posted.
         -- Since Z0 is the vertical apex of corners, an absolute safeZ is alsways going to clear them.
         LdcDidPenUp = false
         post.ModalNumber (" Z", safeZ * scale, format)
      else
-- here is an example of posting Z position adjusted by the maximum rotary radius for the object,
-- thus normalizing the value such that Z0 is the surface of the object.
         post.ModalNumber (" Z", (endZ - LdcMaxRotRad) * scale, format)
      end

-- this test prevents the A0 axis words from posting when cutting "flatsheet".
```

```
      if LdcIsRotaryJob then
         post.ModalNumber (rotaryAxis, endA, "0.000")
      end
   post.Eol()
end
--rotary.off


function OnMove()
--rotary.on
-- a temp var because I use the result more than once.
   local relZ = endZ - LdcMaxRotRad

-- Again, make sure ModalNumber() stmts are consistent with text constant and format as in other functions,
-- else they don't work as intended.
   post.Text ("G1")
      post.ModalNumber(" X", endX * scale, format)
      post.ModalNumber(" Y", endY * scale, format)
      post.ModalNumber(" Z", relZ * scale, format)
      if LdcIsRotaryJob then
         post.ModalNumber(rotaryAxis, endA, "0.000")
      end
      post.ModalNumber(" F", feedRate * scale, FRformat)
   post.Eol()

-- optional to have a function which handles your THC control if necessary.
   LdcTHCctrl(relZ)
End
--rotary.off


function OnArc()
   if LdcArcAsMoves > 0 then post.ArcAsMoves(LdcArcAsMoves) return end

--rotary.on
-- similar to OnMove() rotary adjustments.

   local relZ = endZ - LdcMaxRotRad

   if(arcAngle <0) then
        post.Text ("G3")
   else post.Text ("G2") end
      post.ModalNumber (" X", endX * scale, format)
      post.ModalNumber (" Y", endY * scale, format)
      post.ModalNumber (" Z", relZ * scale, format)
      post.ModalNumber (" I", (arcCentreX - currentX) * scale, format)
      post.ModalNumber (" J", (arcCentreY - currentY) * scale, format)
      if LdcIsRotaryJob then
         post.ModalNumber (rotaryAxis, endA, "0.000")
      end
      post.ModalNumber(" F", feedRate * scale, FRformat)
   post.Eol()

   LdcTHCctrl(relZ)
End
--rotary.off


--rotary.on
-- this is optional, but notice I use it to detect when Z is at rotary cutHeight since my THC is not
-- integrated into the grbl firmware like some controllers. Rather, when I dynamically command my THC ON,
-- then grbl cannot move Z until I dynamically command the THC OFF. So to keep grbl relatively in sync with
-- its expected position of Z, don't turn THC ON until grbl is done moving Z.

function LdcTHCctrl(cz)
   -- Special THC enable/disable control logic...
   -- designed to be called after posting endX, endY, etc. coords, thus endX etc is the new "current" pos.
   -- The currentN(axis) vars do not reflect the wrapped rotary coordinates at this point.
   -- change this code to suit your controller and machine as need be.

   if LdcThcPresent > 0 and LdcThcEnable > 0 then
      -- if THC is OFF and Z is at cutHeight, good time to turn on the THC...
      if not LdcThcOn and math.abs(cz - LdcAbsCutHeight) < 0.001 then
         LdcThcOn = true
         post.Text ( LdcThcONcmd, "\n")

         -- this is good place to test for and handle a small shape...
         --[[
         if entityLength <= LdcSmallShape then
            LdcSmallShapeCount = LdcSmallShapeCount + 1
            -- do something in gcode
         end
         ]]
      end
   end
end
--rotary.off


function OnPenDown()
```

```
--rotary.on
   -- assumes entry with Z at pierceHeight adjusted for rotary position and vertical radius.
   -- vertical radius varies by rotary angle for rectange objects
   local relZ, myX, myY

   if LdcIsRotaryPlugin then
      relZ = rotaryVals.cz - LdcMaxRotRad
      myX  = rotaryVals.cx
      myY  = rotaryVals.cy
   else
      relZ = currentZ
      myX  = currentX
      myY  = currentY
   end
--rotary.off

   local probeBuffer = 5
   if LdcBenchTest > 0 then  -- if true then expected to be so for duration of job
      probeBuffer = 0.1
      LdcProbeCmd = "G38.3 Z"  -- torch probe cmd, no error reported if no limit switch
      LdcTorchProbeTravel = 0
   end

   -- conditionally insert a machine Pause, requiring operator intervention to Resume,
   -- to allow an under capacity air compressor time to build up its storage tank
   -- including the cut we're about to make
   LdcCuttingTime = LdcCuttingTime + (entityLength / LdcOpFeedrate) + LdcTorchTouchoffTime -- in minutes
   if LdcCuttingTime > LdcAirSupplyTime or partIndex > LdcLastPartIndex then
      LdcCuttingTime = (entityLength / LdcOpFeedrate)
      LdcLastPartIndex = partIndex
      post.Text("G0 Z", post.FormatNumber(safeZ * scale, format), "\n")
      post.Text(LdcGrblPauseStmt,
                " (******* MACHINE PAUSE ********)\n")
      post.Text("; **** OPERATOR ** Press Cycle Start or Resume button when ready to resume job)\n")
      post.Text("G0 Z", post.FormatNumber(relZ * scale, format), "\n")
   end

   post.Text("; LDC begin per-cut torch touchoff to set adjust Z0\n")

   post.Text("G91  (enter relative distance motion mode)\n")
   post.NonModalNumber(LdcProbeCmd, -(pierceHeight + LdcTorchProbeTravel + probeBuffer) * scale, format)
      post.ModalNumber(" F", 1000 * scale, "0.##")
      post.Text(" (probe fast to find sw)\n")
   post.NonModalNumber("G0 Z", probeBuffer * scale, format)
      post.Text(" (retract off probe switch)\n")
   post.Text("G04 P0.400 (wait for sw debounce)\n")
   post.NonModalNumber(LdcProbeCmd, -probeBuffer * scale, format)
      post.ModalNumber(" F", 50 * scale, FRformat) -- use ModalNumber(" F"... here so feedrate is corrected later
      post.Text(" (probe slow for accuracy)\n")
   post.Text("G90  (return to absolute distance motion mode)\n")

   if LdcBenchTest > 0 then post.Text("; ") end -- comment out the setting of G92 Z offset
--rotary.ON
   -- set G92 Z offset. Base calc from Z at pierceHeight which was the position upon entry to
   -- this function. This adjusts Z only by the probe amount, keeping Z0 near or on the
   -- max diameter OD if using rotary.
   post.NonModalNumber("G92 Z", (relZ - pierceHeight - LdcTorchProbeTravel) * scale, format)
      post.Text(" (adjust Z0 relative to current Z0 -probe delta)\n")

   --   Due to the previous Z probe events, a side effect
   --   of grbl reading the probe control point position is that the precision of all
   --   axes is lost to no better than the precision of a single axis Step.
   --   MillRight MegaV XY steppers are config'd at 57.288steps/mm or 0.0175mm/step, thus up to
   --   0.009mm on average has been lost by reading the probe coordinates which are in full step values.
   --   Exact to 3 decimal precision is good in general, but it is necessary to avoid grbl error33
   --   when executing a subsequent G2/G3 XYIJ center arc format stmt.
   post.NonModalNumber  ("G0 Z", relZ * scale, format)
      post.NonModalNumber (" X", myX * scale, format)
      post.NonModalNumber (" Y", myY * scale, format)
      if LdcIsRotaryJob then
         post.NonModalNumber (rotaryAxis, endA, "0.000")
      end
   post.Eol()

   -- calculate the cut height as an absolute coord adjusted to include rectangle rotary object
   LdcAbsCutHeight = relZ - pierceHeight + cutHeight

   -- grbl M4 vs. M3 is used in conjunction with grbl config $32=1 (laser mode) such that
   -- the Spindle PWM (SPWM) signal varies in duty cycle proportional to the realtime
   -- control point feedrate as grbl manages acceleration and deceleration
   -- motion through corners, arcs, and start/stop condition motion.
   -- The LDC Z THC Anti-Dive circuit uses M7 and SPWM duty cycle < 86% in order to block THC Z
   -- motion control during those conditions.
   if LdcThcPresent > 0 then
      post.Text("M4 S1000  (setup for speed variable SPWM signal, fire the torch)\n")
   else
      -- if using grbl on a 8 bit Arduino Uno board and grbl 1.1e+ version, then
      -- pin D11 is the torch trigger signal and it requires S1000 to go high, aka fire the torch.
      post.Text("M3 S1000 (setup for constant SPWM ON, fire the torch)\n")
```

```
   end
--rotary.OFF

   -- adjust pierce delay to minimize the size of the pierce hole during plunge vs. Ramp piercing style.
   -- Plunge piercing occurs often when cutting open shapes. In these cases, the total pierce delay
   -- includes the dwell time (no motion time) + the time it takes to lower the torch from pierce height to
   -- cut height since the torch remains over the pierce hole.
   if (leadinType == 0) then  -- assume this is a plunge leadin
      LdcPH2CHtime = (pierceHeight-cutHeight) / (plungeRate/60)  -- in seconds
      -- always use a minimum of some % of tool defined pierce delay in cases
      -- when the plunge feedrate is so slow that dross blowback into nozzle can occur.
      LdcPD = math.max((pierceDelay * 0.40), (pierceDelay - LdcPH2CHtime))
      LdcPDCmt = " (Plunge travel time: "
   else  -- assume this is a Ramp leadin (45deg), use the tool defined pierce delay
      LdcPH2CHtime = (pierceHeight-cutHeight)*1.41 / (plungeRate/60)  -- in seconds
      LdcPD = pierceDelay
      LdcPDCmt = " (Ramp travel time: "
   end
   post.NonModalNumber("G04 P", LdcPD, "0.0#")
      post.NonModalNumber(LdcPDCmt, LdcPH2CHtime, "0.0#")
      post.NonModalNumber(", Total PD:", LdcPD + LdcPH2CHtime, "0.0#")
      post.Text(")\n")
end


function OnPenUp()  -- LDC custom
   post.Text ("M5 (torch off) \n")
   if (endDelay > 0) then
      post.NonModalNumber ("G04 P", endDelay, "0.0#")
        post.Text (" (allow THC time to give Z control back to grbl) \n")
   end
--rotary.ON
   if LdcThcPresent > 0 then
      LdcThcOn = false
      post.Text ( LdcThcOFFcmd, "\n")
   end
   LdcDidPenUp = true
end
--rotary.OFF


function OnDrill()  -- not tested
   OnRapid()
   OnPenDown()
   endZ = drillZ
   OnMove()
   OnPenUp()
   endZ = safeZ -- LDC suspect stmt if rotary
   OnRapid()
end

function OnNewOperation() -- LDC custom
   post.Text(";** Next Operation **********************************************\n")
   post.Text(";*** Op name: " .. operationName .. "\n")
   LdcShapeNum = 1
end


function OnNewEntity()   -- LDC custom
   if (LdcShapeNum == 1) then
      post.NonModalNumber(";**** Pierce Delay: ", pierceDelay, "0.0#")
         post.NonModalNumber(", Kerf width: ", toolDia * scale, format)
      post.Eol()
      LdcOpFeedrate = feedRate
   end
   post.NonModalNumber(";===== Op index: ", operationIndex, "#")
      post.NonModalNumber(", Shape number: ", LdcShapeNum, "#")
   post.Eol()
   LdcShapeNum = LdcShapeNum + 1
end

--rotary.ON
-- required compile-time package for rotary axis support
--   must include here at end of scpost
-- comment out the following stmts if you do not have Rotary-plasma plugin Enabled in SC.
package.path = sc.Globals:Get().thisDir .. "/plugins/RotaryPlasma/?.lua"
require("rotaryhelper")
--rotary.OFF

-- EOF
```