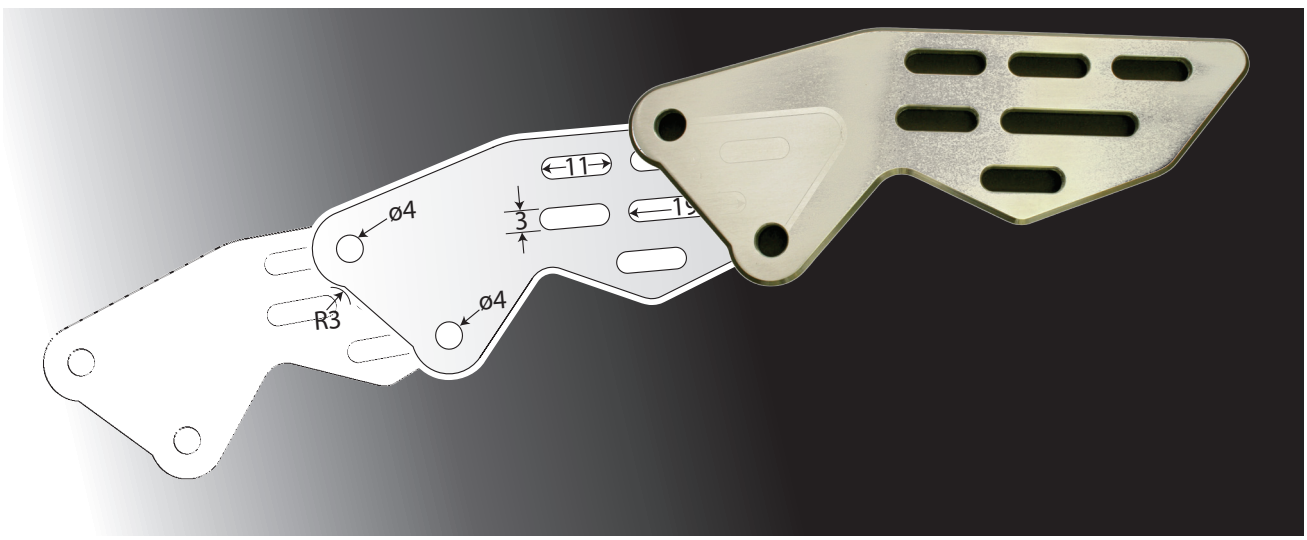
CNC in the Workshop

©
All text and images copyright of
Marcus Bowman
except where stated otherwise.



Part 11

Part 11

In this part of the series, we continue looking at circular paths but find out how to re-use a routine for milling a circel at 0, 0 in another place. Then we progress to packaging code in subroutines so that it can be used repeatedly.

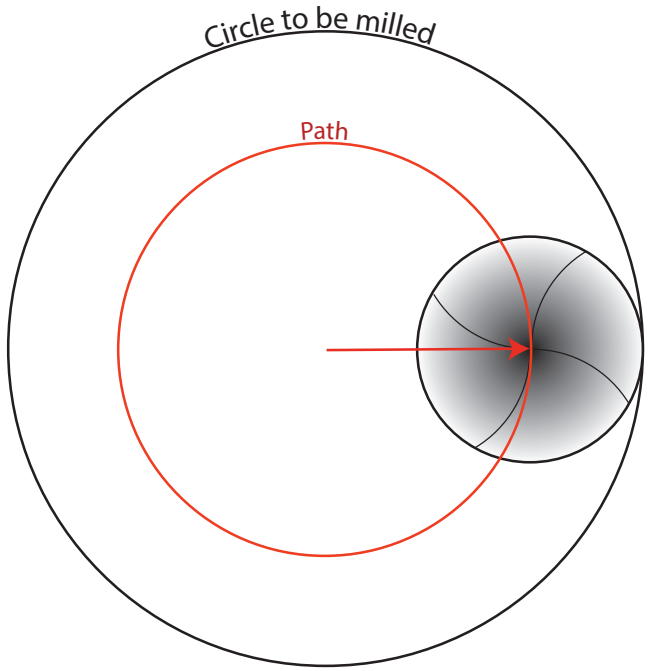


Fig 52: An initial move takes the cutter to the right, to touch the periphery of the circle.

THE USEFULNESS OF A CENTRE AT (0,0)

Now that we know how to make the Controlled Point (CP) move in a circular path, we can machine circles, cut circular pockets and peripheries. All very useful.
But some jobs use the same circular path more than once, and it would be handy to be able to use the same path in a different position.

The G2 and G3 commands describe a circular path and take the form
G2 X~ Y~ I~ J~ Z~ where ~ is replaced by a number.
X~ and Y~ are the co-ordinates of the End point of the circular path of the CP, and the offsets I~ and J~ are calculated from the co-ordinates of the Current Point and the Centre of the circular path. Changing the centre but keeping the same radius will mean the Current Point and the End Point will also change (not surprising, since the circular path is to be in a different place) but it will not change the I or J values, because the circle has the same radius, so the distance from the Centre to the start (the Current Point) will be the same no matter the location of the circle.

If the cutter is moved to the right, to touch the periphery (fig 52):
the I value will be: Xcentre – Xstart, which is the same as the radius of the path of the CP (although it may have a negative sign); and
the J value will be 0 because Ycentre has the same value as Ystart .

To create a circular path to machine a circle of radius 12mm with its centre at (0, 0), by using a 7mm radius cutter (14mm diameter) running inside the circle (fig 53), requires the commands:
(ignoring Z for the moment)
G0 X5 Y0 (Start point)
G2 X5 Y0 I-5 J0 (5, 0 is the End point as well)

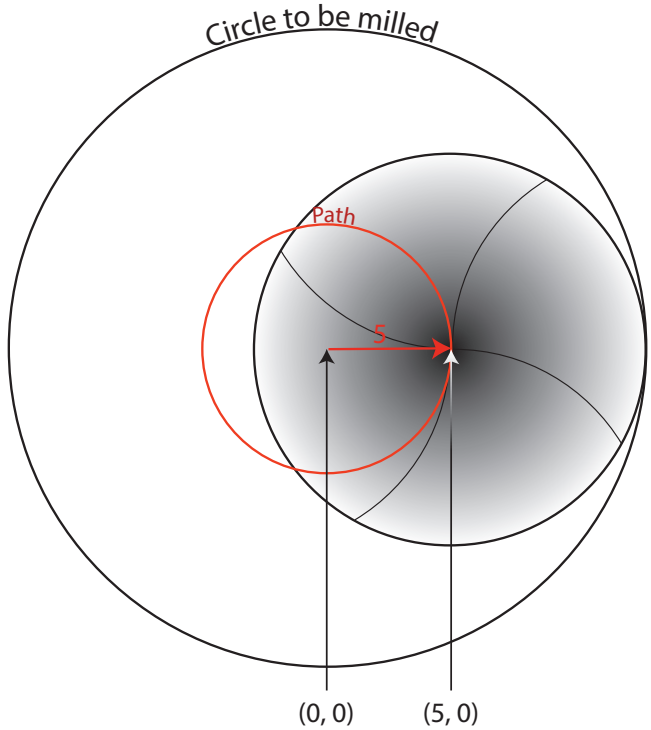


Fig 53: Positioning the 7mm radius cutter so that when it follows the path shown in red it machines a circle of radius 12mm.

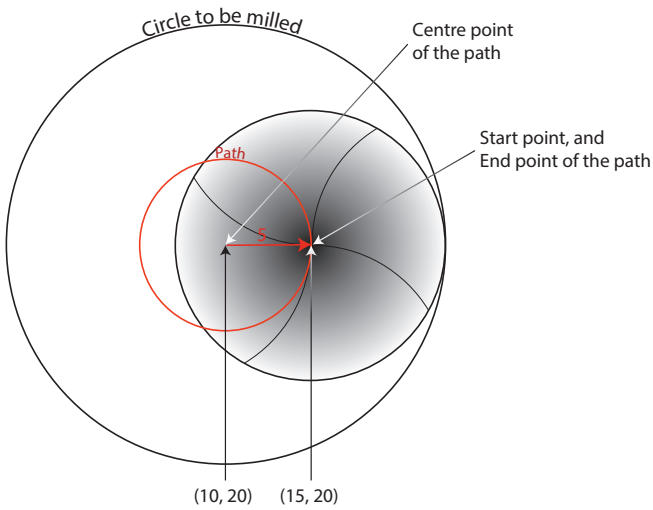


Fig 54: Using the same cutter to machine the same cirlce with a different centre point.

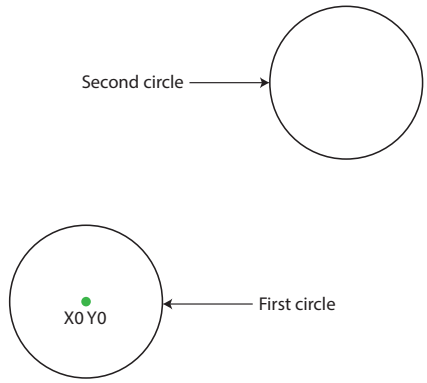


Fig 55: These circles are identical except for the co-ordinates of their centres.

If the same path is to be used to machine an identical circle with centre at (10, 20), using the same cutter, (fig 54) the commands would be:
G0 X15 Y20 (Start point)
G3 X15 Y20 I-5 J0 (15, 20 is the End point as well)

The same path for the same circle with its centre at (20, 40) would be:
G0 X25 Y40
G2 X25 Y40 I-5 J0

So it's only the centre that changes, and that only affects the X and Y co-ordinates of the Start and End points (which are the same point, for a full circle). The I and J values remain the same.
In fact, the X value is always the X co-ordinate of the centre plus 5, and the Y value is always the Y value of the centre, so everything is simply offset by the X and Y co-ordinates of the centre of the circle.
That's handy, because the G92 command can apply an offset to the whole co-ordinate system.

The command takes this form:
G92 X~ Y~ Z~
and it creates an offset to the whole co-ordinate system, so that the Current point ends up with the co-ordinates given in the command after X, Y and Z.
G92 X0 Y0 Z0 makes the co-ordinates of the current point X0 Y0 Z0
So if there is no G92 offset in effect,
G0 X10 Y20 Z15
G92 X0 Y0 Z0
effectively makes the current point at X10 Y20 Z15 take the co-ordinates X0 Y0 Z0

G92 is cancelled using the command G92.1
(There are 3 ways of cancelling the G92 command, and they have different effects, but we will use G92.1 for the moment.)
If there is no G92 offset in use,
G0 X10 Y20 Z15
G92 X12 Y33 Z0
makes the current point (X10 Y20 Z15) have the co-ordinates X12 Y33 Z0

G92.1 removes the offset and makes that point X10 Y20 Z15 once again.

This is a useful command, especially for paths centred on X0 Y0, so we can use it to cut the same circle, using the same commands, but in a different place (fig 55).

t works like this:
Set the centre of the first circle to X0 Y0 (probably by setting the Work Origin there).
Machine that first circle.
Move the CP to where the centre of the second circle should be.
Use G92 X0 Y0 to set the X and Y co-ordinates to zero at that point.
Use the same commands as last time, to machine the circle.
Cancel the offset by using G92.1
Move the CP to where the centre of the third circle should be.
Use G92 X0 Y0 to set the X and Y co-ordinates to zero at that point.
Use the same commands as last time, to machine the circle.
Cancel the offset by using G92.1

The pattern is the same, no matter the number of circles. In fact, this works for all paths, not just circular ones, so this is an extremely useful command.

The co-ordinates of the centres of the circles are all set initially in the "normal" co-ordinate system, with no G92 in effect. The normal co-ordinate system applies at all times, except between the G92 and G92.1 statements.

Notice that, in the example, Z is not changed because Z is not mentioned in the G92 command.
Omitting any reference to Z simply leaves it as it was.
That would also be true for X and Y.
Any co-ordinate not explicitly mentioned in the G92 command will not be altered.

CLOCK STAND SEAT

Here's a real example of the use of G92. It's a very simple workpiece which requires two circular pockets, 8mm deep, in different places (photo 81).

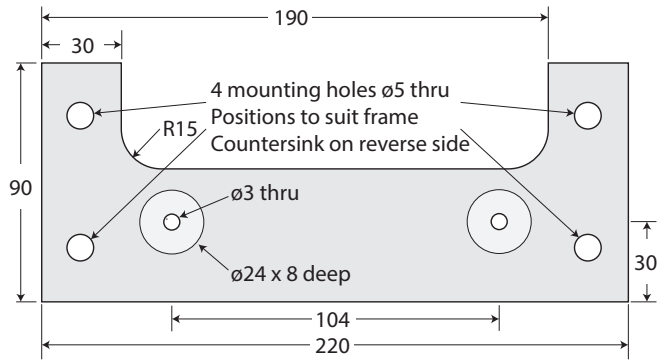
Photo 81: The finished clock stand seat.

The item shown is a seat for a clock stand, and it has been cut from a sheet of 18mm MDF. The seat is the top plate of a simple clock stand (photo 82) and two screws pass

Photo 82: The simple clock stand.

through the base, from below, and into the lower two clock frame pillars, to hold the clock mechanism firmly in place. Under the head of each screw there is a large washer, to spread the load. The length of each screw is limited, though, and the original seat (still in the clock case) was only 10mm thick, so the recesses allow the screw heads and washers to reach the threaded holes in the frame pillars.

Fig 56: The dimensions of the clock stand seat.



Material: MDF 18mm thick

The layout of the workpiece is as shown in fig 56, and the Work Origin is set to X0 Y0 at the centre of the left hand ø24 x 8 deep circular pocket. Photo 83 shows the setup on the mill.

The recesses are 24mm in diameter, and the cutter is an 18mm diameter router bit, so the commands to cut the first circle are:

G0 Z10 (Safe Z)
G0 X0 Y0 (Centre of circle)
G0 X3 Y0 Z0.1 (cutter edge aligned with circumference)
G2 X3 Y0 I-3 J0 Z-2 (spiral cut)
G2 X3 Y0 I-3 J0 Z-4 (spiral cut)
G2 X3 Y0 I-3 J0 Z-6 (spiral cut)
G2 X3 Y0 I-3 J0 Z-8 (spiral cut)
G2 X3 Y0 I-3 J0 Z-8 (level cut)
G0 Z10
G0 X0 Y0

Now move the CP to the centre of the second circle:

G0 X104 Y0
Create an offset to make those co-ordinates X0 Y0
but leave the Z height unchanged
G92 X0 Y0
Use the same commands as shown above, to machine the second circle.
Then cancel the offset
G92.1

Go back to the starting position
G0 X0 Y0
which should take the CP back to the centre of the first circular pocket. That's a useful check, because of the G92 offset is still

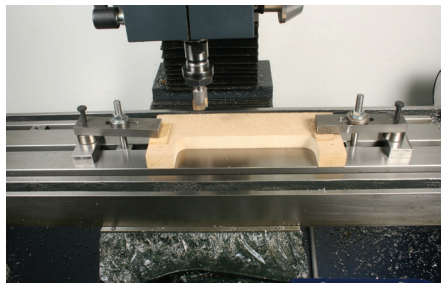


Photo 83: Set up ready to mill.

in effect the cutter will not move. It is only when the offset has been removed that Mach3 will recognise X0 Y0 as being the original Work origin at the centre of the first circular pocket.

Photo 84 shows the result.

The router bit should be run at over 12,000rpm but that's unachievable on my main spindle, so just run it as fast as you can. The finish improves with speed, but the lower speed is fine for this job.

G92 is a powerful command, if used with care. It applies a global offset, so the values of all co-ordinates are changed when G92 is in effect. It is wise to cancel the offset at the earliest opportunity, using G92.1

Our Initialisation Sequence (MEW 209) already contains a G92.1 command, as a precaution, just to make sure there are no offsets in operation when a program runs.

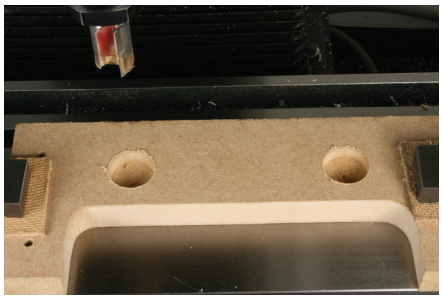


Photo 84: The counterbores machined.

Although this example only creates two holes, G92 is not limited to single small paths. Once in effect, it offsets all co-ordinates, so a more complex path could easily be reproduced many times by using the same method.

PACKAGING THE CODE

It is obvious that the last example used the same lines of code more than once. That's fine, but it becomes a bit of a pain typing the same commands more than once. Cutting and pasting the whole block of commands to machine one hole reduces the typing, and the room for error, but leaves us with a program which may work well, but is difficult to read and to follow. We can get around this quite neatly by creating a subroutine, or block of code which can be typed just once then used as many times as required, afterwards.

To identify the section of code, it has an identifying number at the beginning, and a command at the end. Here's what it might look like:

O123 (The start of the subroutine with reference number 123)
----code to move the CP around a path-------
M99 (signals the end of the subroutine)

In O123, the O is a capital letter and the 123 is a number.

M99 is always the command used at the end of a subroutine; the number doesn't change. Although we gaily talk about the G-code language, and although the majority of commands do begin with G, the O code denotes a flow control command related to the logic of the sequence of commands in a program. There are M codes, F, S and T codes too, so why we favour G is a mystery.

Reference numbers can be anything in the range 1 to 9999.
To avoid confusion, subroutines go after the M30 statement at the end of a program so they do not appear in the normal flow of commands in the main program.

So our code to machine the circle (above) can be put into a subroutine like this:

O123 (Subroutine to machine a circular pocket centred at the Current Point)
G92 X0 Y0 (make the current point X0 Y0)
G0 X3 Y0 Z0.1
G2 X3 Y0 I-3 J0 Z-2
G2 X3 Y0 I-3 J0 Z-4
G2 X3 Y0 I-3 J0 Z-6
G2 X3 Y0 I-3 J0 Z-8
G2 X3 Y0 I-3 J0 Z-8
G0 Z10
G0 X0 Y0
G92.1 (Remove offsets and restore the original co-ordinates)
M99

To use a subroutine, use the command M98 P~ substituting the reference number of the subroutine in place of the ~ character. So M98 P123 would call up the code from subroutine 123 and carry it out. When it has finished (i.e. when the subroutine reaches the M99 command) Mach3 returns to where it was in the main program and carries on from there.
The following program will cut two circular pockets, one at X0 Y0 and the other at X104 Y0. The Work Origin is set initially to X0 Y0 at the centre of the first circle.

<<Initialisation Sequence goes here>>
F200 (set feed rate)
S3000 (set spindle speed)
M3 (turn spindle on)
G0 Z10 (Safe Z)
G0 X0 Y0 (Centre of the first pocket)
M98 P123 (machine the first pocket)
G0 X104 Y0 (Centre of the second pocket)
M98 P123 (machine the second pocket)
M5 (spindle off)
M30

O123
<<put the code to cut the periphery here>>
M99

That has the same sequence as the programs we have used before, but the structure is neater and, as we will see, it allows us to use other powerful techniques.

Using the subroutine is termed "calling" the subroutine; so M98 P123 calls subroutine 123. It's a computer programmer's term, really, but useful nevertheless, because it is generally understood, at least in the world of code.

Clean code

One of the important things about subroutines is:
clean entry / clean exit
clean, in this case, meaning predictable.
That means the subroutine should begin its movements from a predictable start point (in this case, the Current Point), and it should finish at a predictable end point (in this case, the former Current Point).
The only exception to this rule is when we have deliberately created just that: an exception. Even then, there must be a reason for that, and the start and end points each time the subroutine is called must be predictable. The subroutine should leave no loose ends, and should cause no unexpected effects for

the rest of the program. That's why the subroutine cancels offsets before it ends, so that the co-ordinate system which was in effect before the subroutine was called, is once again in effect, and we can carry on in the knowledge that where we are is where we thought we were, and the CP has the same co-ordinates as before, within the same co-ordinate system.

In practice, this means we need to consider where the CP is before we call a subroutine, and think about how to get to a suitable position before the subroutine does it work. Similarly, at the end we need to think about where the CP will be and whether we need to do some additional moves before continuing with the other parts of the program. The more of this that can be done inside the subroutine, the better.

In the subroutine shown above, we must always take the CP to where the centre of the circle will be, before calling the subroutine. The subroutine immediately redefines the Current Point, making it X0 Y0 by applying an offset. Note that G92 works out the offset for itself.

As the subroutine finishes, it takes the CP back to where it was when the subroutine started, then removes the offsets so that the CP has its original co-ordinates once more. The subroutine causes no other effects, so it is a clean subroutine with a predictable entry and exit.

The main program could easily be extended to machine further, identical, pockets in other places, so if you want to create a tool holder with identical pockets, this would be a good method of creating the holes.

If the code inside a subroutine describes a different path, it can be as large or small as you like; and it can be any shape you wish. This technique is not restricted to circles or any other specific shapes.

There are more examples on the website.

Getting a finish

The example fragments of code, above, use G2 commands because that means the teeth on the cutter, which is revolving clockwise (as viewed from above), bite into the work as they are moving towards it. That's conventional milling.

Using the G3 command would mean the cutter was still revolving clockwise (because spindle direction is set independently of the direction in which the CP is moving, using M3 or M4 commands), while the CP motion is anticlockwise. That would be climb milling.

Normally, we use conventional milling. That would mean using G2 commands to cut with the cutter inside the circle, cutting a circular pocket or an inside arc. G3 commands are normally used to guide a cutter around the outside of the work, because contact is normally with the teeth on the opposite side of the cutter.

Relative speeds make this a bit academic, in my view, because if the peripheral speed of a cutter is, say, 100,000mm/min (equivalent to a ø10 cutter at 3000rpm machining aluminium) and the feed rate is 200mm/min, the speed at which a tooth will meet the material for conventional milling is: 103,000mm/min, and for climb milling 97,000mm/min. Not a significant difference there, and well within the kind of margins encountered when trying to set that speed accurately.

Backlash does come into it, with climb milling acting to pull the slides in the "wrong" direction, floating them away from firm contact with a leadscrew.

However; there is no doubt at all that the finish provided by climb milling is different to conventional milling, and there is often a noticeable difference in favour of climb milling.

One good approach is to use conventional milling until the final finishing cut, and to make that a light cut using climb milling. That means cutting undersize, until the final light finishing cut which will bring the job to size and give it a fine finish. It's a bit of a footer doing this manually, but no bother at all using CNC.

So, returning to our example of a ø24 pocket and a ø18 cutter, move the cutter out just a little less than the full distance, and cut the pocket using a similar set of commands as before. Then move the cutter out to the final position and take the final cut. When the last cut has been completed, move the cutter away from the finished surface before lifting it out of the pocket, to prevent a mark on the wall of the pocket.

Here's what a modified subroutine might look like:

```
O123 (Subroutine to machine a circular
pocket centred at the Current Point)
G92 X0 Y0
(Use conventional milling for the roughing
cuts)
G0 X2.9 Y0 Z0.1
G2 X2.9 Y0 I-2.9 J0 Z-2
G2 X2.9 Y0 I-2.9 J0 Z-4
G2 X2.9 Y0 I-2.9 J0 Z-6
G2 X2.9 Y0 I-2.9 J0 Z-8
G2 X2.9 Y0 I-2.9 J0 Z-8
G0 X0 Y0 Z0.1
(Use climb milling for finishing cuts)
G0 X3 Y0
G3 X3 Y0 I-3 J0 Z-2
G3 X3 Y0 I-3 J0 Z-4
G3 X3 Y0 I-3 J0 Z-6
G3 X3 Y0 I-3 J0 Z-8
G3 X3 Y0 I-3 J0 Z-8
G0 X0 Y0 Z10
(Finally, remove the offsets)
G92.1
M99
```