Coordinate Spaces - a Guide

Roger Caffin, with contributions from others

Understanding all the work spaces and offsets system in Mach3 can be bewildering at first. The Mach3 Mill manual tries to explain all this in Chapter 7, but sometimes it seems you need to have fully understood Ch 7 before reading it, so you can notice the bits in Ch 7 which really matter. It's a common problem, and the usual explanation is that the manual was written by the programmer rather than by a user. To make matters worse, the manual adds

> 'The final way of setting a work offset is by typing a new value into an axis DRO ...
>
> You are advised not to use this final method until you are confident using work offsets that have been set up using the Offsets screen.'

Needless to say, the simplest method of resetting the coordinates is just that: typing into the DROs on the screen. I am sure many people start this way without understanding Offsets (I did). It works just fine for any simple machining task.

Unfortunately, if you want to do any macro programming, you find you cannot (easily) enter data into the DROs directly. Actually you can, using two different methods, including an undocumented feature which we will cover later, but not understanding how Offsets work can cause massive grief when your programming gets more sophisticated. So you have to RTFM. And you have to understand the difference between Machine Coordinates, the Controlled Point, the Workplace Offsets, 'other' Offsets and the Tool Offsets. So we will try to explain how it works.

I am approaching this from a CAD and mathematical perspective. The idea of 'spaces' is natural to me, but others will want a more mechanical or 'g-code' perspective. I have included as many relevant examples of g-code as I can.

We will assume you have some familiarity with Mach3 already, so a few short cuts can be taken in presenting some commands (eg G0 and G1). After looking at the 'spaces' perspective we will reiterate a bit over the g-codes. We end with some unexpected details about hidden parameters in Mach3 as they do relate. This may be the most useful part of the whole document!

Please note: this has been written for a Mill. Mach3 for a Lathe is a bit different: for a start the job spins around an axis. You can access various offsets via the Tool Table, but there is no Settings screen.

Coordinate Spaces

The whole idea of a coordinate space is fundamental to CNC use. Think of a sheet of graph paper laid out flat in front of you, with all those lines at right angles. The ones going sideways are conventionally called the X axis, and the ones going away from you are conventionally called the Y axis, and the ones going up and down are called the Z axis. The start of Ch 7 in the Mach3Mill_1.84.pdf manual has some excellent examples of this.

By convention, X values increase to the right, Y values increase going away from you, and Z values increase going upwards. You don't have to follow this convention, but if you don't then the conversation can get very confused. (I did start off back to front, but soon realised the error of my ways.) It did take me a little while to adapt.

Pick a point in the middle of that piece of graph paper, and call it the Origin. That means that X=0, Y=0 and Z=0 at that point. (You will have to imagine the Z axis I am afraid: we are fresh out of 3D graph paper.) Now you can specify any other point by its XYZ coordinates. A point X=4, Y=3 and Z=2 is often written as (4,3,2) for convenience. Such a point is 4 units to the right, 3 units away and 2 units upwards, all from the origin.

As to what units – that's up to you. Some use millimetres, some use inches, and I dare say a few might use something else (nanometres are good...). The difference is just a scale factor. A word of caution though: you really do NOT want to change the units you are using half-way through doing something. Utter confusion can result. And DO make sure you know what units you are using, or you might end up in the sort of trouble they had with the Hubble telescope. A single error with units there cost them a whole space shuttle mission to fix.

Controlled Point

The Controlled Point is what any CNC program is ALL about. The Controlled Point is the middle of the tip of the cutter as it spins. You can think of it as being the tip of a tiny V-point cutter if this helps. A CNC program or machine moves this Controlled Point around. The fact that the cutter is really a lump of very sharp carbide some 20 mm in diameter and spinning at 3,000 RPM is incidental: hopefully any material in the way will be removed without too much damage.

This does mean that issuing a command such as `G0 X10` will move the Controlled Point to X=10. That is the whole point of CNC after all. Normally the DROs on the screen show the values for this 'Controlled Point', so the command `G0 X10` will move the machine to where the X DRO shows 10 units. What really defines the value in the X DRO is more complicated. Note that while we will often talk about just the X axis, the comments really cover all the axes, even the rotary ones. I use 'X' as a short-hand.

In many places to follow I will use the term 'User Space' instead of 'Controlled Point'. User space or user coordinates or part space (the terms are largely interchangeable) is where you are working, and it may be offset from the Machine Coordinates. These offsets are the crux of the matter. Without these offsets a CNC machine would be worth only a fraction of its real value. One could say that the Controlled Point exists in the User Space – well, it does, it has to, but it is really a fairly null-content statement.

Also note that you can switch the DRO display to show 'Machine Coordinates' rather than 'User Space'. You will eventually need to do this. The equation across the top of the Diagnostics screen may help you understand a lot of this too.

Machine Coordinates & Home

While the Controlled Point relates to the space you are working in or the part you are machining, Mach3 really has no understanding of that. It's a program and it knows little about the physical machine, but it does know how to move the Controlled Point around – by outputting signals which move the axes. The movement of the Controlled Point (ie the axes) takes place in the Machine Coordinate space, and this is what Mach3 uses internally for all its movements. The coordinate values seen in the DROs on the screen are in the User Space (program space, part space, etc) and are there for the user. Keep this firmly in your mind. We will go deeply into how they relate.

But any coordinate system has to have an origin: a place where X=0 etc. This place is called Home. Home can be set in two main ways: by moving an axis until a Home switch on the

machine is triggered, or by telling Mach3 that 'right here' is Home. If you have defined Home pins in the Mach3 Ports&Pins screen, then the first method applies. If you want to load pallets onto the milling table with a robot for instance, you will have to have Home switches so the robot and the machine can relate to each other, day after day.

A common place for the Machine Coordinate Home is at the front left bottom corner of the available working volume. That puts the Home switches all at the ends of travel. However, while that is obvious, it can be inconvenient. Imagine you have a 4 m x 2 m router: it could take quite some time for the machine to reach 'Home'. In such cases many find it more convenient to define 'Home' as being in the middle of the table.

Home Switches

If you do not have Home switches – that is if they are not defined in Ports&Pins, the second method for defining Home applies. This is an arbitrary definition that 'right here' is 'Home'. You get it by clicking on the 'Ref All Home' button on the Program Run screen. It could be argued that not having Home switches can be more flexible, but it all depends. It can be fine for a hobbyist, but it creates problems in a 24/7 production environment.

Having Home switches which are not super-reproducible can actually be a hazard as the Home point may wander around a little. In such a case you might want the Home switches to be accurate to better than 10 microns – possibly to 1 micron in a precision metal-working machine.

Going to Home – G28, G30, G53

You can go to the Home position in three different ways. G30 will take you there in a straight line from where ever you are now. If there is something in the way – tough, big crash follows. Be cautious with G30.

G28 will also take you there, but via an intermediate position given in the command. This *may* let you move around an obstacle – or it may not. Read the fine manual and consider carefully. Personally, I favour very explicit move commands in such a case.

You can also go to the Home position with the command `G0 G53 X0 Y0 Z0`. Including the G53 in the command tells Mach3 to work in the Machine Coordinate space rather than the Controlled Point space - for straight line moves (G0, G1). Done blindly, this could be the same as G30 – crash. Fortunately this method is far more flexible as you can put any XYZ values there, not just zero. In particular, the command `G0 G53 X0 Y0` (without any Z movement) can be very useful as we explain below.

Absolute vs Incremental Distance Modes: G90 & G91

XYZ

In absolute distance mode, the axis values (X~ Y~ Z~ A~ B~ C~) in a command represent positions in terms of the currently active User Space, *relative to the origin for that user coordinate space*. (The ~ sign means 'some value'.) This is the normal mode of operation. You get into this mode via G90. (And yes, Mach3 can handle synchronised movement using up to 6 axes.)

So a `command G0 X10` will send the Controlled Point to X=10 from wherever it was. It does not matter whether the Controlled Point was at X=-50 or X=+80: it will go to where X=10 in the current User Space. (The X value will be quite different in the Machine Coordinate space.)

In incremental distance mode (reached via G91), the axis values (X~ Y~ Z~ A~ B~ C~) in a movement command usually represent increments from the current position, but there are numerous exceptions. For example, the command sequence `G91 / G0 Z10 / G90` (where the slashes represent the New Line character) will move the Controlled Point up by 10 units *from where ever it was*. That is, in incremental mode you are not going *to* a point on an axis, you are moving along that axis by the specified distance. This can be very useful for small moves, like retracting the cutter upwards from wherever it was last. That said, the incremental space can be a dangerous space to work in for any length of time (imho).

To summarise so far: G90 and G91 apply to the X, Y, Z, A, B & C variables. G90 puts them into 'absolute mode' while G91 puts them into 'incremental mode'.

A reminder: 'incremental' applies only to *actual moves*, not to other things.

IJK

Some commands (such as G02 and G03) allow you to use I, J & K variables as well. Consider going in an arc from your current position to X~ Y~. G02 and G03 will tell the machine whether the arc should be to the left or to the right, so that is clear. But what radius should the arc have, or where should the centre of the arc be? And just as important: how should you specify the centre of the arc?

The Mach3 manual for G91 says that the I & J values are always incremental, regardless of the Distance Mode, and that the K value is also incremental except for G87 (a special potted boring command which is not defined in the Mach manual). I do not think this is correct, which can be horribly confusing.

The interpretation of the I~ and J~ values can be altered by G90.1 and G91.1 commands. The Mach3 manual makes a hash of this in my opinion. Fortunately, Ger21 summarised this extremely clearly (in 2010) as follows (with my extensions in brackets)

G90= Absolute positioning (for X, Y, Z ...)
G91 = Incremental positioning (for X, Y, Z ...)
G90.1 = Absolute IJ
G91.1 = Incremental IJ

Please note: For the most part, only G02 and G03 use these IJK variables.

By way of example: if you want to specify both the end point of an arc and the centre of the arc in absolute coordinates, you would program `G90 G90.1`, and so on.

Also note: if you get 'crop circles' on your toolpath display, you have absolute and incremental modes mixed up. I am told that some CAM post-processors (still) get these wrong, and that this is a common cause for anguish by new users.

Yet another note: if the <distance from the current position to the centre of the arc> differs from the <distance from the centre of the arc to the destination> by more than a very small amount, Mach3 throws an error.

The alternative to using IJK values in G02 and G03 commands is to use the R format, where R is the radius. Some prefer the R format as it has slightly more flexibility and is not prone to rounding-off errors and does not create crop circles. On the other hand, it can be unwise to try to push the swept angle beyond 180 degrees. (So I do full circles in 2 steps – so what?)

(My thanks to Gerry and Tweakie for helping me understand this bit.)

Machine Coordinates vs User Space: Workspace Offsets

So how do these two spaces relate to each other? Basically as follows. We will use the term 'Controlled Point' here rather than User Coordinate because that's what the Mach manual uses.

Machine Coord = Controlled Point + Workspace Offset + Other Offset

or

Controlled Point = Machine Coord - Workspace Offset - Other Offset

What is the point here? What you are doing is moving the origin of the User Space. You might not want your job to have its origin at the Home position: you might want it offset a bit, for any of several very good reasons. Any time you position the cutter where *you* want the origin to be and zero the displays on the 'screen', you are defining an offset.

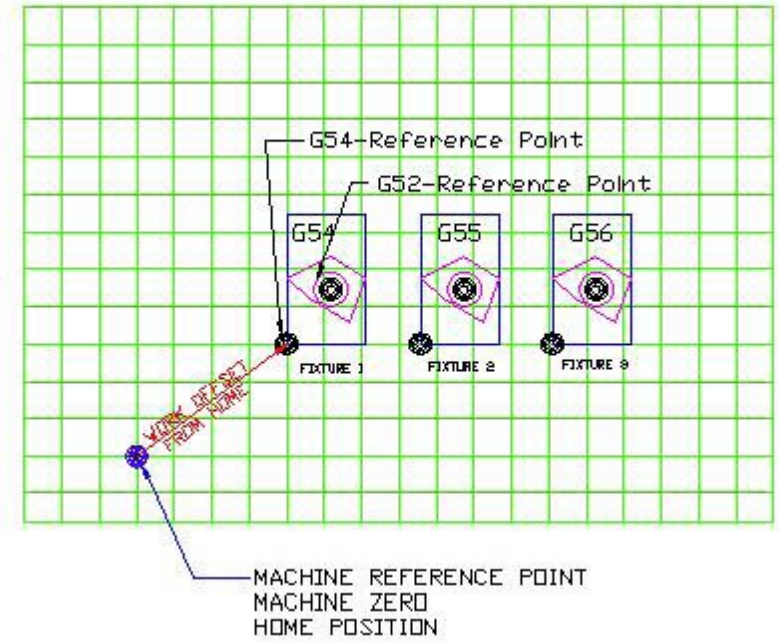

Figure 1: Workspace Offsets (drawing courtesy RICH)

So Mach3 (following the NIST Standard) has the idea of applying a Workspace Offset to the basic Machine Coord origin. Actually, Mach3 allows up to 254 workspace offsets (or 255 or 256) - the manual says different things in different places), with the first 6 being directly accessible by simple g-code commands. To go to Workspace #1, program G54; for Workspace #2 you program G55, etc. Workspace #6 is reached by G59; to reach Workspace #7 you program `G59 P7`, and so on. The offsets are all stored in a Workspace Table, which can be saved. (Why not just `G54 Pn` for the lot? More committees?)

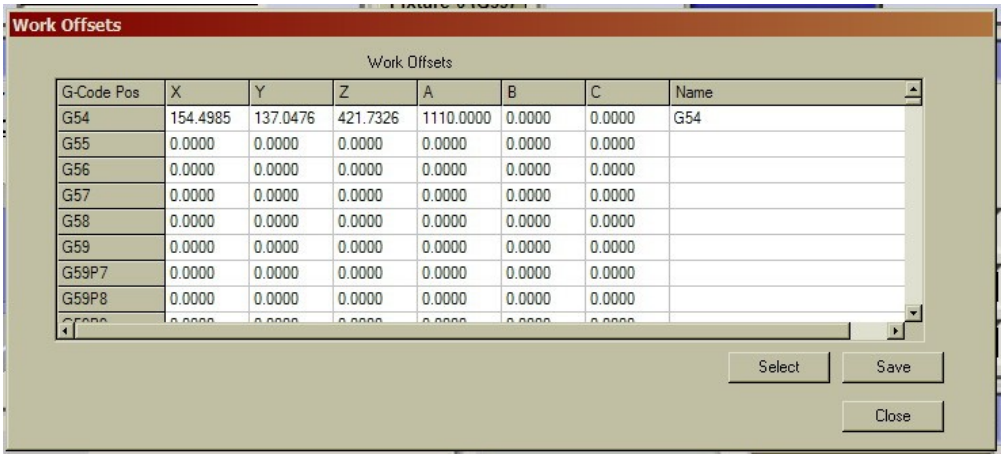

Figure 2: the Work Offsets table

To set the XYZ offsets for Workspace #n you program `G10 L2 Pn X~ Y~ Z~ A~ B~ C~`; you can omit the axes you do not want to affect and they will not be changed. So the Workspaces are fully programmable.

The first form of the above equations is what Mach3 uses internally. When your program says go to X=5 (in the user or part space), Mach3 has to convert this into Machine Coordinates.

It is worth noting that when you move the cutter so you can zero the DROs where you want the user space origin to be, you are in fact changing the parameters in the current Workspace offset.

'Other Offset' – G52

There is also the 'Other' Offset. This has two components: the second one is dealt with later. For the first component, imagine you have a group of 6 parts to machine, all in a line at equal spacings. Instead of changing the Workspace values or Workspace number (which is indeed quite OK as a method), you can apply a temporary Offset to the base Workspace. This is done using `G52 X~ Y~ Z~`, where the XYZ values are the offset, the distance between the parts. What G52 does is effectively to move the User Space origin (again).

What happens if you program `G52 X~` with the same value a second time? Nothing. You have defined the offset, and that is it. Going into incremental mode does not change this because incremental mode only applies to movements. So if you program `G0 X0 / G52 X10 / G0 X0`, the following happens. First you send the Controlled Point to where X=10. Then you move the origin to +10 units along the x axis. Nothing physically moves with this command: you have not given a motion command. Finally, the third command sends the controlled point to where X=10 according to the new position of the origin. You don't have to start at X=0 of course: this is just a simple example. If I have a row of parts to machine, all 20 mm apart in a line, then `G52 X20, G52 X40` etc, before calls to a single subroutine, are the way.

There is another way of doing this is to use `G92 X~`. This is a highly deprecated command, although it has its uses. What it does is to jam the given X value into the current X Coordinate Space DRO by altering the 'Other Offset'. In effect, G92 simulates typing new values into the DROs. But it can really foul up things if you try to mix G52 and G92 in a program, as they use the same special Offset variables! Total confusion is probable; some damage is likely. It is recommended that you do not use G92: use workspaces (G54 etc) and G52 instead.

A caution here: whereas the full sequence for G52 given above will move the X axis +10 units, the same sequence with G92 (`G0 X0 / G92 X10 / G0 X0`) will move the X axis -10 units (at the second G0 command). The X DRO was reading 0, the G92 command put +10 into it, so G0 X0 had to move 'left' to bring the DRO back to 0. One is the reverse of the other.

It can be most frustrating trying to set an axis value to something if you don't understand these offsets. Trust me, I know.

CAUTION: G52 BUG

Sadly, it must be admitted that there are bugs deep in Mach3 associated with the interaction between the G52 instruction and the ToolPath Display. This can lead to you spending hours trying to find out why the ToolPath Display is showing some strange movements which do not appear to be anywhere in your code. In fact, those movements are NOT in your code: they are bugs in how Mach drives the ToolPath Display. The only saving grace here is that, as far as I know, the bug only affects the Display and NOT the actual movement of the tool in real life.

Since there are bugs and I do not have access to the source code, the following is simply my best guess as to what is happening. First we look at the 'static' display created when Mach loads the program, then we look at the dynamic path traced on the screen when Mach executes the program. That makes two bugs.

Imagine you wish to repeat a set of operations at several places, so you put that set into a subroutine and call it several times, shifting the current 'origin' with the G52 instruction for each call of the subroutine. At the end you put a `G52 X0 Y0` to cancel all the shifts. This is what the G52 instruction is for.

Included at the end of the set of instructions, after the `G52 X0 Y0`, or possibly at the start of a subsequent operation, you use a `G0 Z(some safe value)` instruction to make sure the cutter is not going to crash into something. In practice this works very well (and I do it all the time), but what appears on the ToolPath Display has some very strange deviations.

Key to understanding this is in how Mach handles 'default values'. The instruction `G0 Z(new value)` is really read as `G0 X(old value) Y(old value) Z(new value)`. (You can throw A, B & C in here if you wish.) Mach has to adjust the Work Space when the `G52 X0 Y0` is issued, but apparently it does not do this properly in the Display driver. It seems that Mach changes the 'old values' (probably correctly), but does not use the updated values *in the Display driver* when executing the following `G0 Z(safe)` instruction. Instead of executing `G0 X(updated value) Y(updated value) Z(new)` it executes `G0 X(old value) Y(old value) Z(new value)`. Having executed one movement after the `G52 X0 Y0`, it then loads the updated values.

That is not the end of the story. The Display driver does two things: it creates the static tool path display when the code is loaded, and it traces out the dynamic movement of the tool as the program executes. The static tool path display has the bug mentioned above, but the dynamic path display compounds this bug. Sometimes it handles the G52 instruction correctly, but sometimes it misses the `G52 X0 Y0` reset. When this happens you see the dynamic path following the static path for a while, but then the dynamic path abruptly shifts to a new location where it continues to trace the otherwise correct tool path. It looks as though the dynamic path code sometimes fails to notice the origin reset.

Fortunately, Mach does use the updated values in the code which drives the Controlled Point, so all is well there. But some Mach programmers avoid the use of the G52 instruction because of these problems. That's a pity.

Tool Offsets

While Workplace Offsets relate to the coordinate system inside the CNC, Tool Offsets relate to the cutter itself. They were temporarily included in the above under 'Other Offsets' for convenience. However, Tool Offsets are for the length of the cutter, so they only affect the Z axis. (There is an X parameter in the command, but it has little to do with the X axis.) Again, this can be used in at least two different ways.

If you are using tooling mounted in holders such as BT30, then the Tool Offsets can be thought of as effectively the distance from the BT30 reference plane to the tool tips. As you change tooling you change which Tool Offset you use. In more practical terms, the Tool Offset is the difference in length between some nominal Tool #0 and other tools: you don't have to go back to the BT30 reference plane. When using Tool Offsets, the rest of the Workspace stays unchanged. In particular, note that the Tool Offset value affects the Controlled Point; it does NOT affect the Machine Coordinates. (The latter relate after all to some real or virtual Home switches on each axis.)

The Tool Offset value is set by the command `G10 L1 Pn Z(offset)`, where 'n' is the tool number. The Offset is an entry in another table, also saveable. You select which physical Tool you want to use with the command M6 ('change the tool') or with Tn M6 if you are using an ATC, but note that all this does is to change the physical tool: it does not change the Tool Offset. It is crucial to understand the difference. To change a Tool Offset you need to program `G43 Hn`, where n is the Tool number. (There is also a G44 command for use when the Tool offsets have the wrong sign - more committee in-fighting? I have not studied this G44 command.)

Can you load Tool 7 and Tool Offset 3? Yes. However, the results might not be exactly what you wanted (or wish to afford) – unless you have made the conscious decision to store the information about Tool 7 in Offset 3.

What this means is that the 'Other Offset' in the above section will also include the Tool Offset value, at least on the Z axis. Yes, that means there are no less than 3 different offsets to be applied to the Machine Coordinate to get the Controlled Point. Actually, the Tool Offsets Table also allows entries for tool wear in both Z and diameter, but Mach3 does not use this data afaik.

You can turn the Tool Offset system off by programming G49. Effectively that selects Tool 0, which has unalterable zero offset values. In many cases this will be perfectly OK, and may make life easier and safer. The Tool Offset table can also hold the Tool Diameter (X~), but that is not touched here.

And that leads to the other method of handling the concept of tool length compensation. You can turn the Tool Offset table off and do all offsetting via the Workspace. If you are sticking with just one or two manually-loaded cutters for the entire job, a direct calibration of a Workspace for each one is quite reasonable, especially for a fully manual operation. Position the tip of the cutter some fixed distance above the top of the part (feeler gauge, ground HSS rod, whatever) and type the thickness of the gauge into the Z DRO. Now Z=0 puts the tool tip just in contact. You can also do this using the Mach Mill Offsets screen. There are many ways to skin the cat here.

Rotations – G15/G16, G68/G69

Just when you thought you have it all worked out, someone goes and rotates the world on you. You can do that in g-code too. Caution: here be real dragons.

Imagine you want to draw a ring of little lines like graduations on a dial. The first line is easy: you go from X=10, Y=0 to X=11, Y=0. But for the rest you have to do some trigonometry. Well, no: you can get Mach3 to do the trig for you. And Mach3 offers two different ways to do it!

These rotations are not included in the definition of User Space given previously. Mach does the trig for you to convert all this back into Machine coordinates. It **has** to do these conversions in order to know how to drive the X, Y & Z motors. Unfortunately, as explained a little further on, Mach does not fully handle all rotations. There are some restriction.

A consequence of this realisation is that the G2 and G3 commands do not really create arcs. The X, Y & Z motors don't work that way. Instead Mach (effectively) breaks the arcs down into a very large number of very short straight line segments. Fortunately, the Exact Stop condition does NOT apply to the ends of all these short segments. If it did the machine would never get anywhere. This also explains why some older dinosaur controllers did not actually offer the G2 and G3 commands unless you paid lots of extra money for extra control boards in the controller box.

Polar Coordinate Moves – G15/G16

This first method tells Mach3 to interpret your commands differently. The command G16 tells Mach3 to go into polar coordinates; the command G15 cancels the polar coordinate mode. Thus the G16 part of the command sequence `G16/G0 X10 Y45/G15` tells Mach3 to go into polar mode, treat the X value as a radius and the Y value as an angle, and then drop out of the polar mode. The cutter will go to a point 10 units *from wherever it was beforehand* (the radius) at an angle of 45 degrees (to the X axis). That is, the starting position is treated as a temporary origin.

Caution: the Mach manual says you must not make X or Y moves other than with G0 or G1 when G16 is active (ie, no G2 or G3). But of course X will mean radius and Y will mean angle in any such moves.

With the command sequence as follows you could drill a ring of holes (assuming Z is above the surface which is at Z=0):

```
G16             % polar mode
G0 X10 Y10      % go to radius=10 at 10 degrees
G81 Z-3.0       % drill hole
G1 Y20          % rotate to radius=10, angle=20
G81 Z-3.0       % drill hole
G1 Y30          % rotate to radius=10, angle=30
G81 Z-3.0       % drill hole
...
G15             % exit polar mode
```

Note that we do not need to change the X parameter as the radius stays constant in this example. Granted, unless you are careful, this sort of code can look a bit strange. If you want to draw those little radial lines on the dial, try this (assuming that the surface of the dial is at Z=0 and the cutter starts at Z=1)

```
G16             % polar mode
G0 X10 Y0       % go to radius=10 at 0 degrees
```

```
m98 p10         % subroutine mark line
G0 X10 Y10      % go to radius=10 at 10 degrees
m98 p10         % subroutine mark line
G1 Y20          % rotate to radius=10, angle=20
m98 p10         % subroutine mark line
G1 Y30          % rotate to radius=10, angle=30
m98 p10         % subroutine mark line
G15             % exit polar mode
M30

O10
G1 Z-1
G1 X11
G1 Z1
G0 x10
M99
```

You can load this code and try it out. It works (provided you set up the feed and spin beforehand). But a small word of caution here. Mach3 permits subroutines, but don't nest them too deeply or it may get confused. That is a design defect in Mach3 which cannot be repaired. (Stack overflow? Maybe.)

According to the Mach3 manual, doing this in a Fanuc controller is a shade more complex. The instructions are not the same. So g-code is not always portable.

Having got somewhere and dropped out of polar mode, subsequent moves will be made along the conventional Cartesian axes. This might be more easily understood after the next section.

Coordinate space rotation – G68/G69

Whereas G15/G16 tell Mach to interpret the XY values differently, the G68/G69 commands actually rotate the whole coordinate space around the specified position. The command `G68 A10 B20 R45 I1` causes the whole user space to be rotated around the specified A,B point by R degrees. In this case the user space will be rotated 45 degrees around the point (10,20). It would seem that they use A & B here (instead of X & Y) to avoid confusion. If the I parameter is present, the angle of rotation is incremental, otherwise it is absolute. G69 cancels all rotation.

Can you issue the command `G68 R45 ?` That is, rotate the coordinates around the current position? Yes, you can. This means you can go to the centre of a circle and rotate the coordinate space around that point. Can you use G2 and G3 arc commands when the whole coordinate space has been rotated? YES!

The big difference is in what subsequent commands do. If G68 is in effect the X & Y axes in the Machine Coordinate space are rotated, but you can pretend that you are still working in an unrotated XY space. You can for instance use this to engrave numbers around a dial (using the Mach wizard for instance) such that they are always oriented to the dial markings. That's fine and very suitable for a *rotating* dial, but you would NOT use this to engrave the hours of the day around a watch dial. You don't want the hours rotated!

This is a very powerful command, but if you issue subsequent G68 commands about a different centre point I can almost guarantee that you will get lost. Good programming practice (imho) is to issue the G68, do what has to be done there, then immediately issue G69. Only after the G69 should you issue another G68 command. I have no idea what would happen if you tried to mix G15 and G68!

(Footnote: the Mach3 manual does have some typographical errors, including in this section.)

Restriction re use of G2 and G3

G2 and G3 may be used in the three planes as selected by G17 (XY), G18 (XZ) and G19 (YZ) to machine arcs. However, you cannot rotate the axes at the same time using G68. The axis of rotation for the G2/G3 circle must be parallel to one of the Machine Coordinate X, Y or Z axes. This restriction can be a right pain at times.

Test it for Yourself

This is all very well, but you may need to see it for yourself to really understand it all. This is very easy to do. Fire up Mach Mill and go to the Program Run screen. Move the table and spindle to somewhere near the middle. If you have Home switches, you will have to edit the following a bit. I will assume you do not. I will refer to the on-screen DROs as 'User Coords' for convenience.

Click on the Ref All Home button. The DROs will change. Click on the Machine Coords button and you should have all zeroes in the DROs. Click again back to get the 'User Coords'.

Now zero each of the X, Y & Z axes via the individual buttons on the screen. Check that the Machine Coords are still zero.

Go to the Offsets screen. You should have Current Work Offset = 1 (from the G54 in the Init string). Active Work Offset should read G54. The Part Offsets should all be zero. The user coords DROs above the Machine Coords button (on the right) should all be zero. Click on the Machine Coords button and you should still have all zeros.

Now go back to the Program Run screen and move the X & Y axes a bit. Alternately, go to the MDI screen and enter G0 X10 Y-15 (assuming these values are reasonable). The screen DROs should update. Leave the Z axis alone for the moment: you will see why shortly.

Click on the Machine Coords button and the DROs should not change. You have moved the Controlled Point a bit after all, and both User Coords and Machine Coords should reflect this.

Go to the Offsets screen and look at the DROs. Current Work Offsets will still be zero, since you have not changed anything there. The User Coords on the right will have changed. So far, as expected.

Now zero the User Coords on the right. They will go to zero, but the Current Work offsets will now show whatever the User Coords were. Why? Because you have not moved the machine (or the Controlled Point) when you 'zeroed the user DROs', and you have not moved the Ref Home position. If you want the user DROs to read zero, then Mach3 will change the values in the Work Offset to accommodate you.

Type a zero into one of the Current Work Offsets DROs. The value which was there will move back to the user DROs. It has to: you still have not moved the machine. You are still at the same distance from the Machine Home.

Tool Offsets are a shade more complex to understand at first. The Tool Offsets area is at the bottom right hand corner. The Tool DRO should read 0 and the Z offset should read 0.000. You cannot change the offset value for Tool 0. Also, if necessary, click on the Zero Z button on the user DROs. This all says that the Controlled Point for Tool 0 is at Z=0. We will call that the surface of the part.

Now type 1 into the Tool DRO, type 5 into the DRO under Gage Block Height and click on Set Tool Offset. You should get -5 in the Z offset DRO and +5 in the Z DRO. This says that if you have a Tool 1 which is 5 units shorter than Tool 0 when mounted in the spindle but you have NOT moved the Z axis, then the tip of Tool 1 will be +5 units above whatever was the surface of the part. A 5 unit high gage block will just fit under Tool 1. While you are at it, check the current Machine Coords: Z will still read 0 because you still have not moved the machine.

To bring the tip of Tool 1 down to the surface of the part you will need to apply a Z Offset of -5 units. This is now obvious from looking at the picture of the gage block. Change Tool number to 0 and the user Z DRO will go back to 0.

You can also test G52 here. Let's assume Work Offset 1 has a Y value of 0 and user Y DRO has a value of 0 (so Machine Coord Y=0). Go to the MDI screen and type G52 Y5. The user Y DROs will now read -5. You have told Mach3 to move the user origin for the Y axis 5 units north. The Machine Coords will still show Y=0, because you have not moved the machine. But if the USER origin is now 5 units to the north, while the machine has not moved, then the current user Y coord must be -5 units.

You cannot see the G52 value on the Offsets screen, but you can see it on the Diagnostics screen under the G92 label. Curiously, while you can change other DROs on the Diagnostics screen, it seems you cannot change the G52/G92 DROs there. This may be a quirk of Mach3.

You could go on to test the G15/G16 commands and the G68/G69 commands. Be aware that the tool path display in Mach3 is not always strictly accurate, and sometimes does slightly funny things. You get used to it...

Parameters

Mach3 allows about 10,000 user parameters (like #10, #123, #1000 etc). However, there are also some special cases outside the 10,000 range which seem to be poorly documented. Obviously, these are Mach3 special bypasses, and may not be in the NIST spec. If you know of any more, please, let me know!

Parameters #500 - #600: these seem to be saved in the XML file and restored the next time you fire up Mach3. They are 'persistent'.

Parameters #5181 - #5186: These are the G30 Home coordinates for X, Y, Z, A, B & C.
I would normally expect the G30 Home coordinates in Machine Space to be all 0, so why they are stored in parameters is not clear to me.
While the Mach documentation is not very clear, I believe they must refer to the Machine Coordinate space. We would not want the Home position to wander around just because we changed the Workspace number.

Parameters #5161 - #5166: These are the G28 Home coordinates for X, Y, Z, A, B & C. It may be that the G28 coordinates are actually for the intermediate position given in the G28 command, on the way to the final G30 coordinates.

Parameters #5191 - #5196: These *may* be the scale factors for X, Y, Z, A, B & C, although the definitions are greyed out in the Mach3Mill document. One would expect the scale factors to be somewhere after all. I have not tested these.

Parameters #5211 - #5216: These give you direct access to the offsets for the G52/G92 commands. Touching these without really, really knowing what you are doing could be exciting and expensive. Exactly how these parameters work is actually not entirely clear, so writing to

them may or may not do what you want. Better by far to use G52 explicitly – although being able to read the current offsets may have some value. Note that these (and subsequent) are 'offsets', so the question of User space or Machine space does not apply.

Parameter #5220: This is the current Work Offset number – value 1 – 255.

Parameters #5221 - #5226: These are the XYZABC values for Work Offset #1.

Parameters #5241 - #5246: These are the XYZABC values for Work Offset #2. This continues at steps of 20 for each Work Offset to #10301 - #10306 for Work Offset #255. I can understand the reasons for starting the X offset at nnnnn1, but I do not understand the reason for the step of 20 instead of 10 each time. I also note that the parameter range is nominally #1 - #10,320, so there are a few ?unused? parameters from #10-,307 to #10,320.

Parameters #15001 - #15255: These are linked (to a certain extent) with User DROs 1001 – 1255. You would normally only need to use these if you are doing some fancy screen creation.

Parameters #99nnn: These are also special, at least for some values of nnn. Noting that the addresses for the X, Y, Z DROs etc for macro commands are 800, 801, 802 etc (ie for the get/setOEMDRO commands), it turns out that you can also access the on-screen DROs from g-code by using the parameters #9980n. I have not explored other values for nnn. So executing `#99802=0` is the direct equivalent of `G92 Z0`.

Practical Defaults

Not knowing which Workspace offset you are in can make life difficult, but do not despair. The default initialisation string for Mach3 (top right hand corner of the Program Run screen) includes a G54 command. This will select Workspace #1. Many people will never leave that workspace – and will never *need* to leave that Workspace. That's fine. Not knowing what Tool offset you have set up can be equally exciting, so the default initialisation string for Mach3 also includes a G49 command to turn the whole Tool Offset system off. You can in fact do everything you want to do in the G49/G54 space alone.

The string also includes a G90, so you know what mode you are in.