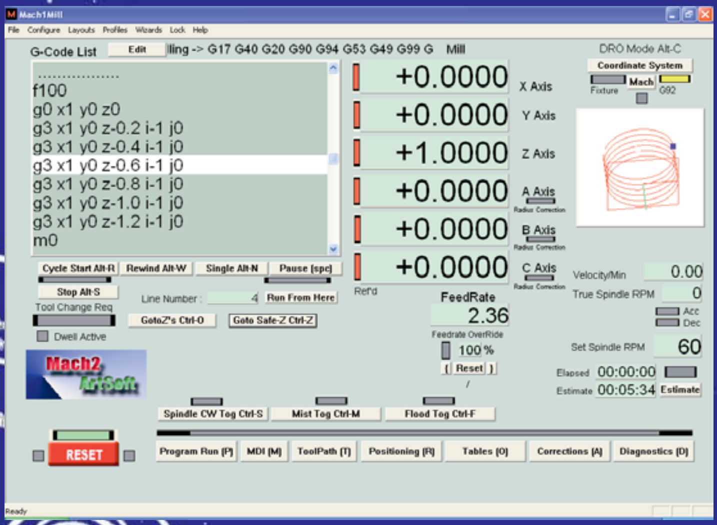
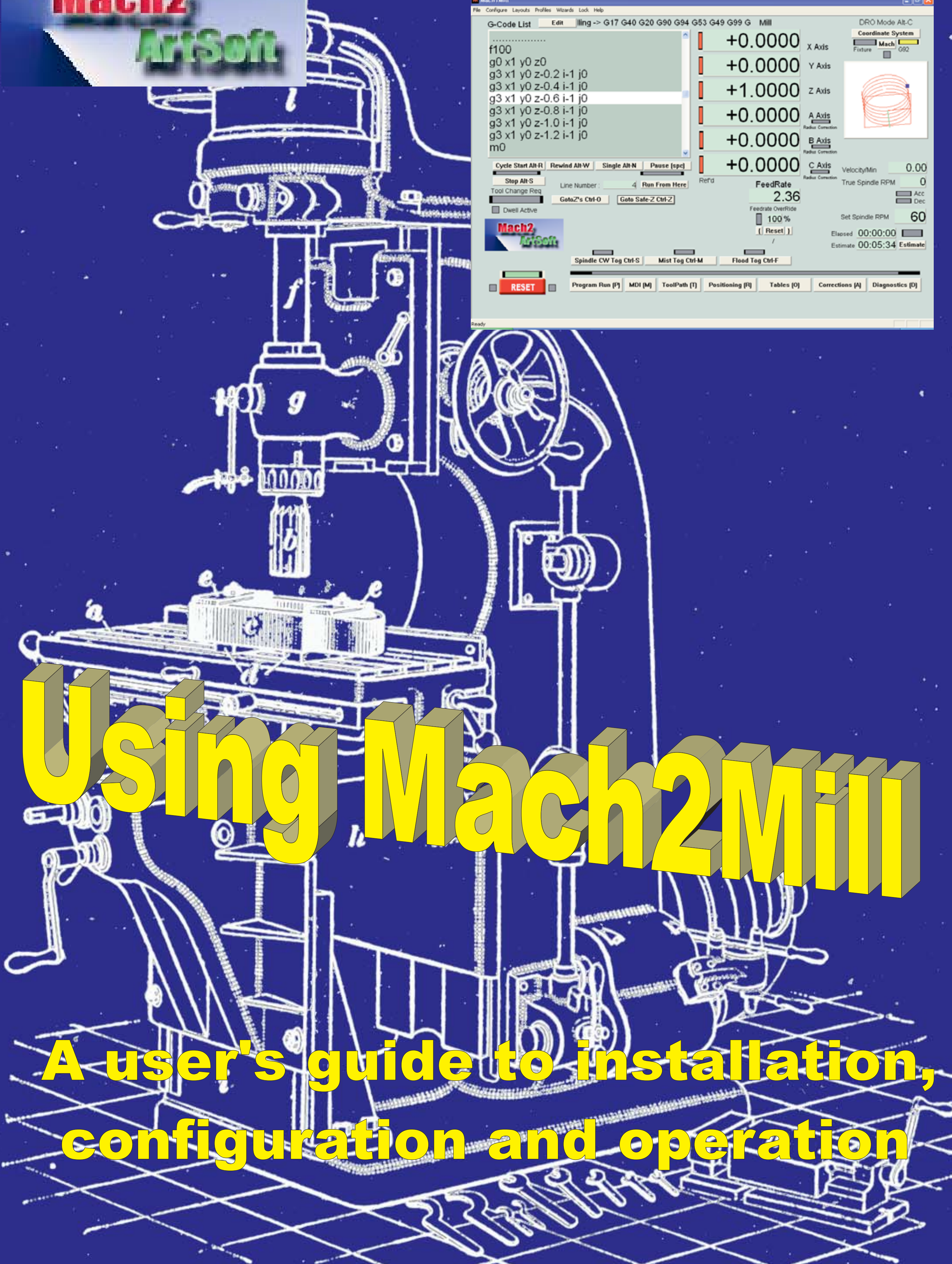
Mach2 ArtSoft



```
.................
f100
g0 x1 y0 z0
g3 x1 y0 z-0.2 i-1 j0
g3 x1 y0 z-0.4 i-1 j0
g3 x1 y0 z-0.6 i-1 j0
g3 x1 y0 z-0.8 i-1 j0
g3 x1 y0 z-1.0 i-1 j0
g3 x1 y0 z-1.2 i-1 j0
m0
```

X Axis +0.0000
Y Axis +0.0000
Z Axis +1.0000
A Axis +0.0000
B Axis +0.0000
C Axis +0.0000
Line Number 4
FeedRate 2.36
Feedrate OverRide 100 %
Velocity/Min 0.00
True Spindle RPM 0
Set Spindle RPM 60
Elapsed 00:00:00
Estimate 00:05:34



Using Mach2Mill

A user's guide to installation, configuration and operation

CNC Milling Users' Guide

or

The nurture, care and feeding of the Mach2 controlled CNC Mill

Alpha status documentation under development. All comments and suggestions welcomed by japrentice@iee.org or fenerty@artofcnc.ca

Front cover: A vertical mill circa 1914
Back cover: The old way of coordinating motion on mill table and a rotary axis

Contents

1. Preface

Any machine tool is potentially dangerous. Computer controlled machines are potentially more dangerous than manual ones because, for example, a computer is quite prepared to rotate an 8" unbalanced cast iron four-jaw chuck at 3000 rpm, to plunge a panel-fielding router cutter deep into a piece of oak or to mill the clamps holding your work to the table!

This manual tries to give you guidance on safety precautions and techniques but because we do not know the details of your machine or local conditions we can accept no responsibility for any machine or any damage or injury caused by its use. It is your responsibility to ensure that you understand the implications of what you design and build and to comply with any legislation and codes of practice applicable to your country or state.

If you are in any doubt you must seek guidance from a professionally qualified expert rather than risk injury to yourself or to others.

This document is intended to give enough details about how the Mach2Mill software interacts with your machine tool, how it is configured for different axis drive methods and about the input languages and formats supported for programming to enable you to implement a powerful CNC system on a machine with up to six controlled axes. Typical machine tools that can be controlled are mills, routers, plasma cutting tables.

Although Mach2Mill can control the two axes of a lathe for profile turning or the like, a separate program and supporting documentation is being developed to support the full functionality of a lathe.

Certain portions of text a printed "greyed out". They generally describe features found in some machine controllers which are not presently implemented in Mach2. The description of a greyed out feature here is not to be taken as a commitment to implement it at any given time in the future.

Thanks are due to numerous people including the original team who worked at National Institute for Standards and Testing (NIST) on the EMC project and the users of Mach2 without whose experience, materials and constructive comments this manual could not have been written.

ArtSoft Corporation is dedicated to continual improvement of its products so suggestions for enhancements, corrections and clarifications will be gratefully received.

This version of the manual relates to Mach2Mill Beta 9.

2. Introducing CNC machining systems

2.1 Parts of a machining system

This chapter will introduce you to terminology used in the rest of this manual and allow you to understand the purpose of the different components in a numerically controlled milling system.

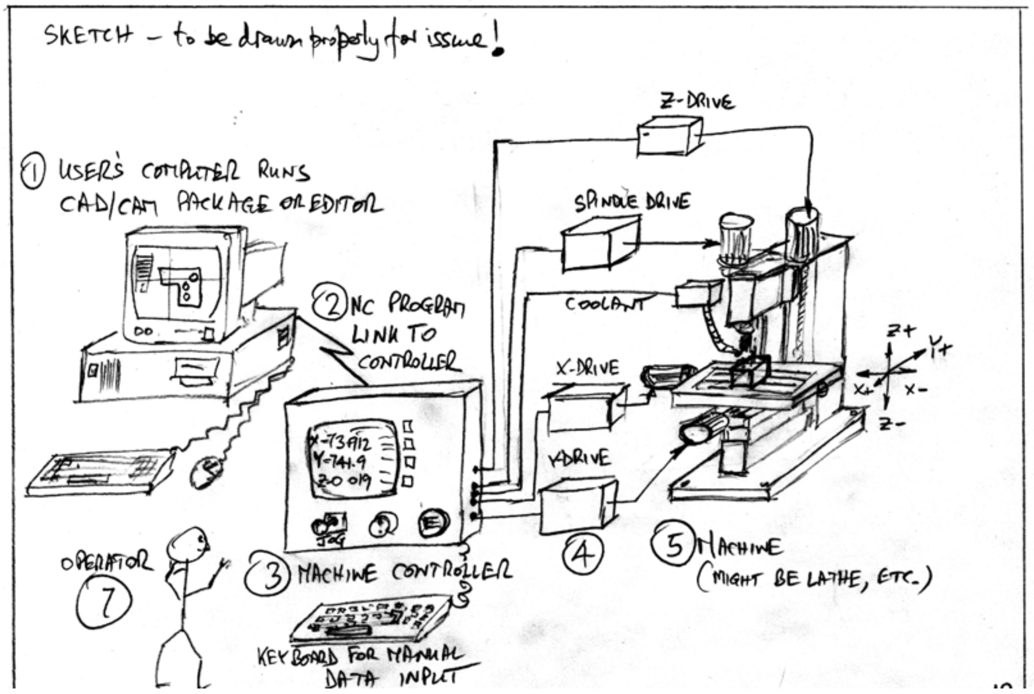

SKETCH – to be drawn properly for issue!
Z-DRIVE
① USER'S COMPUTER RUNS CAD/CAM PACKAGE OR EDITOR
SPINDLE DRIVE
②NC PROGRAM LINK TO CONTROLLER
COOLANT
X-DRIVE
Y-DRIVE
Z+
Y+
X+
X-
Z-
X-73912
Y-741.9
Z0 019
JOG
OPERATOR ⑦
③ MACHINE CONTROLLER
④
⑤MACHINE (MIGHT BE LATHE, ETC.)
KEYBOARD FOR MANUAL DATA INPUT

Figure 1.1 - Typical NC machining system - DIAGRAM TO BE REDRAFTED BEFORE ISSUE!!

The main parts of a system for numerically controlled milling are shown in figure 1.1

The designer of a part generally uses a Computer Aided Design/Computer Aided Manufacturing (CAD/CAM) program or programs on a computer (1). The output of this program, which is a Part Program and is often in "G-code" is transferred (by a network or perhaps floppy disc) (2) to the Machine Controller (3). The Machine Controller is responsible for interpreting the Part Program to control the tool which will cut the workpiece. The axes of the Machine (5) are moved by screws, racks or belts which are powered by servo motors or stepper motors. The signals from the Machine Controller are amplified by the Drives (4) so that they are powerful enough and suitably timed to operate the motors.

Although a milling machine is illustrated, the Machine can be a router or a plasma or laser cutter. A separate manual describes Mach2 controlling a lathe

Frequently the Machine Controller can control starting and stopping of the spindle motor (or even control its speed), can turn coolant on and off and will check that a Part Program or Machine Operator (7) are not trying to move any axis beyond its limits.

The Machine Controller also has controls like buttons, a keyboard, potentiometer knobs or a joystick so that the Operator can control the machine manually and start and stop the running of the Part Program. The Machine Controller has a display so that the Operator knows what is happening.

Because the commands of a G-code program can request complicated co-ordinated movements of the machine axes the Machine Controller has to be able to perform a lot of calculations in "real-time" (e.g. cutting a helix requires a lot of trigonometrical calculation). Historically this made it an expensive piece of equipment.

2.2 How Mach2 fits in

Mach2 is a software package which runs on a PC and turns it into a very powerful and economical Machine Controller to replace (3) in figure 1.1.

To run Mach2 you need Windows XP (or Windows 2000) ideally running on a 1GHz processor with a 1024 x 768 pixel resolution screen. A desktop machine will give much better performance than most laptops and be considerably cheaper. You can, of course use this computer for any other functions in the workshop (such as (1) in figure 1.1 - running a CAD/CAM package) when it is not controlling your machine.

The drivers for your machine's axis motors must accept step pulses and a direction signal. Virtually all stepper motor drivers work like this, as do modern DC and AC servo systems with digital encoders. **Beware** if you are converting an old NC machine whose servos may use resolvers to measure position of the axes as you will have to provide a complete new drive for each axis.

3. Try out the Mach2 Machine Controller software

You are still reading this so evidently you think Mach2 might be an asset in your workshop! The best thing to do now it to download a free demonstration version of the software and try it out on your computer. You do not need a machine tool to be connected up, indeed for the present it is better not to have one.

If you have bought a complete system from a reseller then some or all of these installation steps may have be done for you already.

3.1 Installation

Mach2 is distributed from ArtSoft Corp. by downloading via the Internet. You download the package as one self installing file (which, in the present release, is less than 3 megabytes). This will run for an indefinite period as a demonstration version with a few limitations on the size of job that can be undertaken and the specialist features supported. When you purchase a licence this will "unlock" the demonstration version you have already installed and configured. Full details of pricing and options are on the ArtSoft website www.artofcnc.ca

3.1.1 Downloading

Download the package from www.artofcnc.ca using the right mouse button and *Save Target as…* to put the self-installing file in any convenient working directory (perhaps Windows\Temp).

When the file has downloaded it can be immediately run by using the *Open* button on the download dialog or this dialog can be closed for later installation. When you want to do the installation you merely run the downloaded file. For example you could run Windows Explorer (right click *Start* button), and double-click on the downloaded file in the working directory.

3.1.2 Installing

You do not need a machine tool connected yet. If you are just starting it would be better not to have one connected. Note where the cable or cables from the machine tool are plugged into your PC. Switch off the PC , the machine tool and its drives and unplug the 25 pin connector(s) from the back of the PC. Now switch the PC back on.

When you run the downloaded file you will be guided through the usual installation steps for a Windows program such as accepting the license conditions and selecting the folder for Mach2. On the Setup Finished dialog you should ensure that *Initialise System* is checked and click *Finish.* You will now be told to reboot before running any Mach2 software.

3.1.3 The vital re-boot

This reboot is **vital**. If you do not do it then you will get into great difficulties which can only be overcome by using the Windows Control Panel to uninstall the driver manually. **So please reboot now.**

If you are interested in knowing why the reboot is required then read on, otherwise skip to the next section.

Although Mach2 will appear to be a single program when you are using it, it actually consists of three parts: a driver which is installed as part of Windows like a printer or network driver, a graphical user interface (GUI) and an OCX which accepts messages from and sends replies to the GUI. The reasons for having three parts are complex (for example it is possible for experts to write their own programs which will control Mach2 without its GUI) but the driver is the most important and ingenious part.

Mach2 must be able to send very accurately timed signals to control the axes of the machine tool. Windows likes to be in charge and runs normal user programs when it has nothing better to do itself. So Mach2 cannot be a "normal user program"; it must be at the lowest level inside Windows (that is it handles interrupts). Furthermore to do this at the high speeds possible (each axis can be given attention 45,000 times per second) the driver needs to tune its own code. Windows does not approve of this (it's a trick that viruses play) so it has to be asked to give special permission. This process requires the reboot. So if you have not done the re-boot then Windows will give the Blue Screen of Death and the driver will be corrupt. The only way out of this will be to manually remove the driver.

Having given these dire warnings, it is only fair to say that the reboot is only required when the driver is first installed. If you update your system with a newer version then the reboot is not vital. The install sequence does however still ask you to do it. Windows XP boots so quickly that it is not much hardship to do it every time.

3.1.4 Convenient desktop icons

So you **have** rebooted! It is now worthwhile to setup some icons for desktop shortcuts to the Mach2 programs.

Use Windows Explorer (right-click *Start*) and by right-clicking on the .EXE file for the Mill create a shortcut to the file. Repeat this for the file OCXDriverTest.exe. Drag these shortcuts onto your desktop.

3.1.5 Testing the installation

It is now highly recommended to test the system. Mach2 is not a simple program. It takes great liberties with Windows in order to perform its job; this means it will not work on all

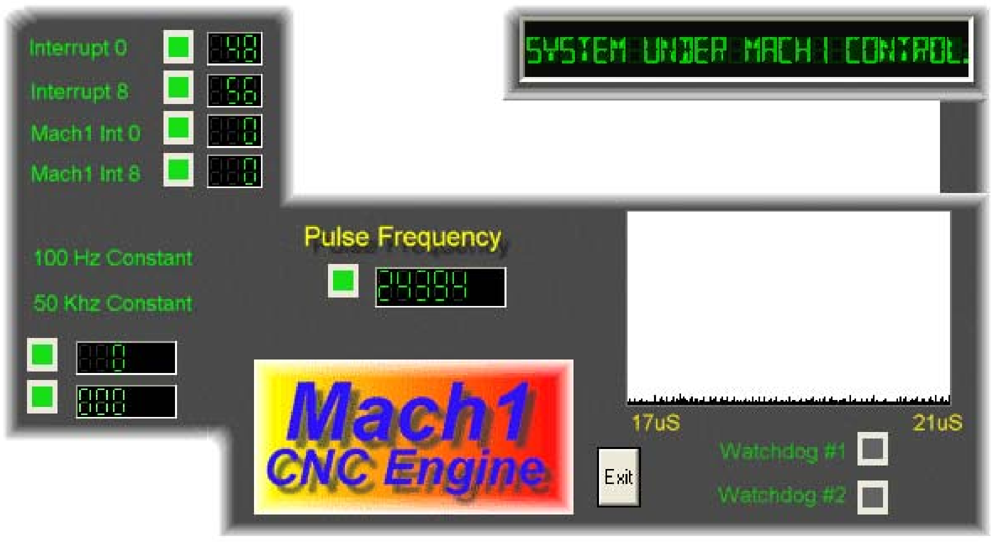


Figure 3-1 - OCXTest screenshot

systems due to many factors. QuickTime's system monitor running in the background can kill it, for example, and there will be other programs which you probably are not even aware are on your system that can do the same. Windows can and does start many processes in the background; some appear as icons in the systray and others do not show themselves in any way. Because of this it is important, though not mandatory, that you test your system when you suspect something is wrong or you just want to check that an install went well.

Double click the OCXDriverTest icon you set up. Its screen shot is in figure 3.1.

You can ignore all the boxes with the exception of the pulse timer. It should be fairly steady around 24,600Hz, but may vary around, even wildly, on some systems. This does not mean the pulse timer is necessarily unsteady, it may mean that the computer is heavily loaded or slow to begin with, since Mach2 takes the highest priority in the system, the clock may be shunted down to a priority slow enough that its one second is a variable length of time. Since the pulse count is based on one second of Windows time, variations in Windows time will make the pulse count look like its swinging around a lot even when it is rock solid. Basically, if you see a similar screen to the one above, everything is working well so **close the OCXDiverTest program and skip to the section Screens** in this manual.

Windows "experts" might be interested to see a few other things. The square window on your right is a type of timing analyzer. When the timer is not running it is black. When it is running it displays a line along the bottom with small variations pushing upwards. These small variations are the changes in timing from one interrupt cycle to another. There should be no lines longer than ¼ inch or so on most systems. Even if there are variations its possible they are below the threshold necessary to create timing jitters so a movement test would be performed in that case to see if jogging is smooth.

The left side, shows Int 0 as vector 48 and Int 8 as vector 56. These are the old IDT vectors used by windows prior to Mach1 taking them over. If these boxes read zero, then you are either using a newer, more powerful motherboard or a dual Pentium system which means the driver has switched to APIC timing as an alternative method and has not redirected the old interrupts as they are lower priority then the vector Mach1 took.

You may have 2 or 3 things happen to you when running the test which may indicate a problem.

1. “Driver not found or installed, contact Art.”, this means that the driver is not loaded into Windows for some reason. This can occur on XP systems which have a corruption of their driver database, reloading Windows is the cure in this case. Or, you may be running Win2000. Win2000 has a bug/"feature" which interferes with loading the driver. It may need to be loaded manually -s ee the next section

2. When the system says, taking over…3…2…1.. and then reboots, one of two things has occurred. Either you didn’t reboot when asked (told you!!) or the driver is corrupted or unable to be used in your system. In this case follow the next section and remove the driver manually, then re-install. If the same thing happens, please notify ArtSoft using the e-mail link on www.artofcnc.ca and you will be given guidance.
   A few systems have motherboards which have hardware for the APIC timer but whose BIOS code does not use it. This will confuse Mach2 install. A special diver (Mach2.sys) is available for download which forces use of the older i8529 interrupt controller. Un-install the standard Mach2.sys as explained below and manually install the special one. You will need to repeat this process whenever you download an upgraded version of Mach2 as installing the new version will replace the special driver.

3.1.6 Notes for manual driver installation and un-installation

You only need to read and do this section if you have not successfully run the OCXDriverTest program.

The driver (Mach2.sys) can be installed and uninstalled manually using the Windows control panel. The dialog boxes differ slightly between Windows 2000 and Windows XP but the steps are identical.

- Open the Control panel and double-click on the icon or line for *System*.
- Select *Hardware* and click *Add Hardware wizard*. (As mentioned before Mach2's driver works at the lowest level in Windows). Windows will look for any new actual hardware (and find none).
- Tell the wizard you have already installed it and then proceed to the next screen.
- You will be shown a list of hardware. Scroll to the bottom of this and select *Add a new hardware device* and move to the next screen.
- On the next screen you do not want Windows to search for the driver so select *Install the hardware that I manually select from a list (Advanced)*
- The list you are shown will include an entry for *Mach1/2 pulsing engine*. Select this and go to the next screen.

- Click *Have disc* and on the next screen point the file selector to your Mach2 directory (C:\Mach2 by default). Windows should find the file *Mach2.inf*. Select this file and click *Open*. Windows will install the driver.

The driver can be uninstalled rather more simply.

- Open the Control panel and double-click on the icon or line for System.
- Select *Hardware* and click *Device Manager*
- You will be shown a list of devices and their drivers. *Mach1 Pulsing Engine* has the driver *Mach2 Driver* under it. Use the **+** to expand the tree if necessary. Right-click on Mach2 Drive gives the option to uninstall it. This will remove the file Mach2.sys from the Windows folder. The copy in the Mach2 will still be there.

There is one final point to note. Windows remembers all the information about the way you have configured Mach2 in the Windows Registry. This information is not deleted by un-installing the driver and deleting other Mach2 files so it will remain whenever you upgrade the system. However in the very unlikely event that you need a totally clean installation from scratch then you need to delete the entry in the Windows Registry.

An error in editing the Registry could force you to re-install Windows and all your applications so proceed with the greatest care. Seek advice from an expert if you are in any doubt.

All the information relating to Mach2 is under HKEY_CURRENT_USER\Software\Mach2 and can be modified or deleted by the Windows Regedit program.

3.2 The screens

You are now ready to try out a "dry run" Mach2. It will be much easier to show you how to set up your actual machine tool when you have experimented with Mach2 like this. You can "pretend" to machine and learn a lot even if you haven't got a CNC machine tool yet. If you have got one, then do make sure it is not connected to the PC.

Mach2 is designed so that it is very easy to customise its screens to suit the way you work. This means that the screens you see may not look exactly like those in Appendix 1. If there are major difference then your system supplier should have given you a set of screenshots to match your system.

Double-click the Mach2Mill icon to run the program. You should see the Mill Program Run screen similar to that in Appendix 1 (but with the various DROs set to zero, no program loaded etc.).

Notice the red Reset button. It will have a flashing Red/Green LED (simulation of a light emitting diode) above it and some yellow LEDs lit. If you click the button then the yellow LEDs go out and the flashing LED turns to solid green. Mach2 is ready for action!

If you cannot reset then the problem is probably something plugged into your parallel port or ports (a "dongle" perhaps) or the PC has previously had Mach2 installed on it with an unusual allocation of port pins to the Emergency Stop (EStop signal). You will need to seek help or read the start of Chapter 5.

3.2.1 Types of object on screens

You will see that the Program Run screen is made up of the following types of object:

- Button (e.g. Reset, Stop Alt-S, etc.)
- DRO or Digital Readout. Anything with a number displayed will be a DRO. The main ones are, of course the current positions of the X, Y, Z, A, B & C axes.
- LEDs (in various sizes and shapes)
- G-code display window (with its own scroll bars)
- Toolpath display (blank square on your screen at the moment)

There is one further control that is not on the Program Run screen:

- MDI (Manual Data Input) line

Buttons and the MDI line are your inputs to Mach2.

DROs can be displays by Mach2 or used as inputs by you. The background colour changes when you are inputting.

The G-code window and Toolpath displays are for information from Mach2 to you. You can, however, manipulate both of them (e.g. scrolling the G-code window, zooming, rotating and panning the Toolpath display)

3.2.2 Using buttons and shortcuts

On the standard screens every button has a keyboard shortcut. This will be shown after the name on the button itself or in a label near it. Pressing the named key when the screen is displayed is the same as clicking the button with the mouse. You might like to try using the mouse and keyboard shortcuts to turn on and off the spindle, to change the feedrate override and to reset it to 100% and to switch to the MDI screen. Notice that letters are sometimes combined with the Control or Alt keys. Although they are shown as uppercase (for ease of reading) you do **not** use the shift key when using the shortcuts.

If a button does not appear on the current screen then its keyboard shortcut is not active.

There are certain special keyboard shortcuts which are global across all screens. Chapter 5 shows how these are set up.

3.2.3 Data entry to DRO

You can enter new data into any DRO by clicking in it with the mouse (or using the global shortcut key (? by default) to select DROs and selecting the one you want with the arrow keys)

Try entering a feedrate like 45.6 on the Program Run screen. You press the *Enter* key to accept the new value or the *Esc* key to revert to the previous one. The Backspace and Delete are not used when inputting to DROs.

Caution: It is not always sensible to put you own data into a DRO. For example the display of actual spindle speed is computed by Mach2. Any value you enter will be overwritten. You can put values into the axis DROs but you should not do it until you have read Chapter 7 in detail. This is **not** a way of moving the tool!

3.3 Jogging

You can move the tool relative to any place on your work by hand using various types of Jogging. Of course, on some axes the tool itself will move and on others it will be the machine table or slides. We will use the words "move the tool" here for simplicity.

Several of the screens display the jogging controls. These are laid out in slightly different ways but contain the following elements. Figure 3.2 gives an possible view of them.

You can recognise the jog controls by the illuminated ball. If you click or drag your mouse on it then the main axes of the machine (X, Y in Mill) will move. The speed will depend how far you are from the square in the middle of the icon. Thus, for example, clicking the mouse in the top righthand corner of the icon will move axes X and

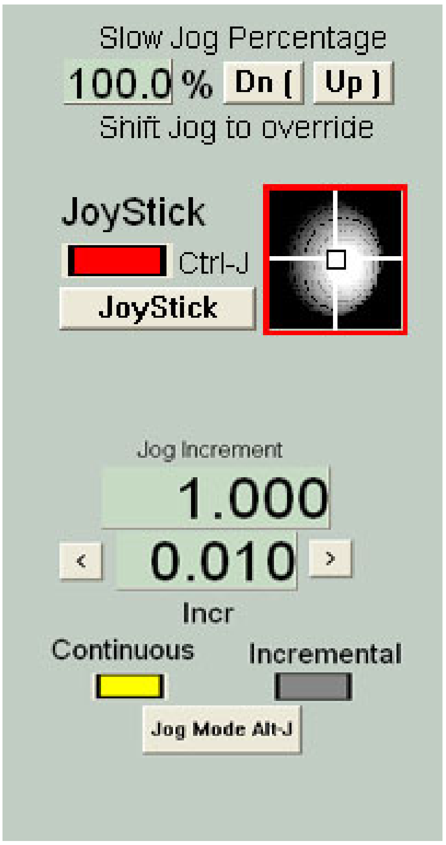


Figure 3.2 - Jog controls (simulated view)

Y at speed. You will see the axis DROs responding.

You can use the keyboard for jogging. The arrow keys are set by default to give you jogging on the main axes. You can configure these keys (see Chapter 5) to suit your own preferences.

In figure 3.2 you will see that the *Continuous* LED is shown lit. The *Jog Mode* button toggles between *Continuous* and *Incremental* modes,

In Continuous mode the chosen axis will jog for as long as you hold the key down. The speed of jogging is set by the *Slow Jog Percentage* DRO. You can enter any value from 0.1% to 100% to get whatever speed you want. The Up and Dn buttons beside this DRO will alter its value in 5% steps. If you depress the *Shift* key then the jogging will occur at 100% speed whatever the override setting. This allows you to quickly jog to near your destination and the position accurately.

In Incremental mode each press of a jog key will move the axis by the distance indicated in the *Jog Increment* DRO. You can set this to whatever value you like. Note that holding the key depressed gives repeated incremental jogs. The value below *Jog Increment* (labelled *Incr*) is a constant that can be added to or subtracted from it using the buttons beside it or their shortcuts. This is an alternative to typing a value into the Jog Increment DRO.

The final option for jogging is a joystick connected to the PC games port or USB. Mach2 will work with any Windows compatible "analog joystick" (so you could even control your X axis by a Ferrari steering wheel!). The appropriate Windows driver will be needed for the joystick device. The 'stick is enabled by the *Joystick* button and, for safety, must be in the central position when it is enabled.

If you have an actual joystick and it has a throttle control then this can be configured either to control the jog override speed or the control the feed rate override (see Chapter 5 again). Such a joystick is a cheap way of providing very flexible manual control of your machine tool.

Now would be a good time to try all the jogging options on your system. Don't forget that there are keyboard shortcuts for the buttons, so why not identify them and try them. You should soon find a way of working that feels comfortable.

3.4 Manual Data Input (MDI)

Use the mouse or keyboard shortcut to display the MDI (Manual Data Input) screen.

This has a single line for data entry. You can click in it to select it or use the global shortcut (? by default). You can type any valid line that could appear in a Part Program and it will be executed when you press *Return*. You can discard the line by pressing *Esc*. The *Backspace* key can be used for correcting mistakes in your typing.

If you know some G-code commands then you could try them out. If not then try:

```
G0 X1.6 Y2.3
```

Which will move the tool to co-ordinates X = 1.6 units and Y - 2.3 units. (it is G zero not G letter O). You will see the axis DROs move to the new co-ordinates.

Try several different commands (or G0 to different places). If you use the up or down arrow keys while in the MDI line you will see that Mach2 scrolls you back and forwards through the history of commands you have used. This makes repeating a command easy without having to re-type it.

An MDI line (or Block) can have several commands on it and they will be executed in the "sensible" order as defined in Chapter 11 not necessarily from left to right. For example setting a feed speed by something like F2.5 will take effect before any feed speed movements even if the F2.5 appears in the middle or even at the end of the block (line). If in doubt type several MDI commands in one by one.

3.5 Running a G-code program

Now it is time to run a Part Program. You will normally be able to edit programs without leaving Mach2 but as we have not yet configured it to say which editor to use it is easiest to set up the program outside Mach2.

Use Notepad to enter the following lines into a text file and save it in a convenient folder as `spiral.txt`

```
f100
g0 x1 y0 z0
g3 x1 y0 z-0.2 i-1 j0
g3 x1 y0 z-0.4 i-1 j0
g3 x1 y0 z-0.6 i-1 j0
g3 x1 y0 z-0.8 i-1 j0
g3 x1 y0 z-1.0 i-1 j0
g3 x1 y0 z-1.2 i-1 j0
m0
```

Use the File>Load G-code menu to load this program. You will notice that it is displayed in the G-code window.

On the *Program Run* screen you can try the effect of the *Start Cycle*, *Pause*, *Single*, *Stop*, and *Rewind* buttons and their shortcuts.

As you run the program you may notice that the highlighted line moves in a peculiar way in the G-code window. Mach2 reads ahead and plans its moves to avoid the toolpath having to slow down more than in necessary. This lookahead is reflected in the display and when you pause.

You can go to any line of code scrolling the display so the line is highlighted. You can then use *Run from here*.

3.6 Toolpath display

3.6.1 *Viewing the toolpath*

The Program Run screen has a blank square on it when Mach2 is first loaded. When the Spiral program is loaded you will see it change to a circle inside a square. You are looking straight down on the toolpath for the programmed part, i.e in Mach2Mill you are looking perpendicular to the X-Y plane.

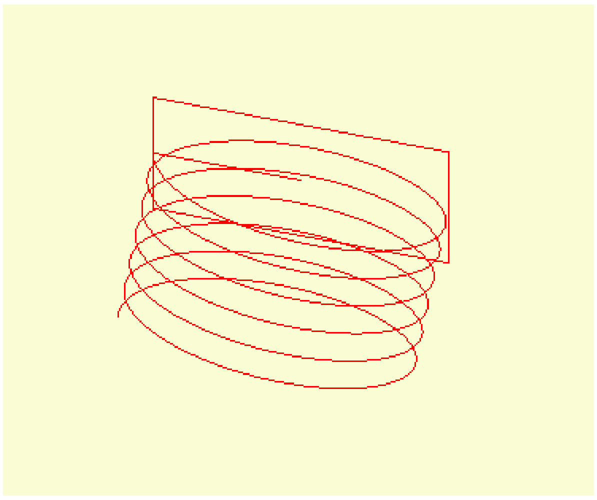


Figure 3.3 Toolpath from Spiral.txt

If you use the mouse to drag the cursor leftwards over the window then this is like moving your head to the left so you are looking obliquely onto X-Y and you will be able to see that the circle is actually a spiral cut downwards (in the negative Z direction). Each of the G3 lines in the Spiral program above draws a circle while simultaneously lowering the tool 0.2 in the Z direction. You can also see the initial G0 move which is a straight line.

When you run the part program the path that the tool is following is traced in green on the toolpath display. This green tracing is cleared when you alter the view of the toolpath and zoom level so you can easily clear green from the path if you just want to see subsequent movement of the tool.

Dragging the cursor in the toolpath window in the up and down direction is like moving your head in the Y direction over the path of the tool in space. This allows you to visualise where the tool will move and is moving for very complex parts.

Notice that you cannot rotate the plane you are looking at about the Z axis so you cannot see a conventional isometric view of the toolpath.

3.6.2 Panning and Zooming the toolpath display

The toolpath display can be zoomed by dragging the cursor in its window with the Shift key depressed.

The toolpath display can be panned in its window by dragging the cursor with the Right mouse button.

Double-clicking the toolpath window restores the display to the original perpendicular view with no zoom applied.

Calculating and displaying the toolpath uses quite a lot of the central processor's time which, on a slow computer, you might need for actually controlling the axes. A button is provided on the Diagnostics screen to disable the redrawing if the toolpath.

3.7 Other screen features

Finally it is worth browsing through all the screens. As a small challenge you might like to see if you can identify the following useful features:

- A button for estimating the time that a Part Program will take to run on the actual machine tool
- The controls for overriding the feedrate selected in the Part Program
- DROs which give the extent of movement of the tool in all axes for the loaded Part Program
- A group of buttons for displaying what co-ordinate system the axis DROs display. You will need to read Chapter 7 for the meaning of the different systems
- A screen that lets you set up information like where you want the Z axis to be put to make X and Y moves safe from hitting clamps etc.
- A screen that lets you monitor the logic levels (zero and one) on all Mach2s inputs and outputs.

4. Hardware issues and connections to your machine tool

If you have bought a machine that is already equipped to be run by Mach2 then you will probably not need to read this chapter (except out of general interest). Your supplier will have given you some documentation on how to connect the parts of your system together.

Read this chapter to discover what Mach2 expects it is going to control and how you can connect up standard components like stepper motor drivers and micro-switches. We will assume that you can understand simple schematic circuit diagrams; if not, then now is the time to get some help.

On the first reading you might not want to bother with sections after 4.6. This section tells you about the hardware aspects of connections. Chapter 5 gives details of configuring Mach2 to use the connected items.

4.1 Safety - again

Any machine tool is potentially dangerous. Computer controlled machines are potentially more dangerous than manual ones because, for example, a computer is quite prepared to rotate an 8" unbalanced cast iron four-jaw chuck at 3000 rpm, to plunge a panel-fielding router cutter deep into a piece of oak or to mill the clamps holding your work to the table!

This manual tries to give you guidance on safety precautions and techniques but because we do not know the details of your machine or local conditions we can accept no responsibility for any machine or any damage or injury caused by its use. It is your responsibility to ensure that you understand the implications of what you design and build and to comply with any legislation and codes of practice applicable to your country or state.

If you are in any doubt you must seek guidance from a professionally qualified expert rather than risk injury to yourself or to others.

4.2 What Mach2 can control

Mach2 is designed to control machines like milling machines and lathes. The characteristics of these machines used by Mach2 are:

- Some user controls. An emergency stop (EStop) button **must** be provided on every machine
- Two or three axes which are at right angles to each other (referred to as X, Y and Z)
- A tool which moves relative to a workpiece. The origin of the axes is fixed in relation to the workpiece. The relative movement can, of course, be by (i) the tool moving (e.g. the quill of a milling spindle moves the tool in the Z direction or a lathe tool mounted on a cross-slide and a saddle moves the tool in the X and Z directions) or (ii) by the table and workpiece moving (e.g. on a knee type mill the table moves in the X, Y and Z directions)

And optionally:

- Some switches to say when the tool is in the "Home" or "Referenced" position
- Some switches to define the limits of permitted relative movement of the tool
- A controlled "spindle". The "spindle" might rotate the tool (mill) or the workpiece (lathe).
- Up to three additional axes. These can be defined as Rotary (i.e. their movement is measured in degrees) or Linear. One of the additional linear axis can be slaved to the X or Y or Z axis. The two will move together at all times in response to a

Part Program's moves and to your jogging but they will each be referenced separately. (see *Configuring slaved axes* for more details).

- A switch or switches which interlock the guards on the machine
- Controls for the way coolant is delivered (Flood and/or Mist)
- A probe in the tool holder that allows digitising of an existing part
- Encoders, such as linear glass scales, which can display the position of parts of the machine

The connections between your machine and the PC running Mach2 are made through the parallel (printer) port(s) of the computer. A simple machine will only need one port; a complex one will need two.

Mach2 will control all six axes, co-ordinating their simultaneous movement with linear interpolation or perform circular interpolation on two axes (out of X, Y or Z) while simultaneously linearly interpolating the other four with the angle being swept by the circular interpolation. The tool can thus move in a tapering helical path if required! The feed rate during these moves is maintained at the value requested by your Part Program, subject to limitations of the acceleration and maximum speed of the axes. You can move the axes by hand with various jogging controls.

If the mechanism of your machine is like a robot arm or a hexapod then Mach2 **will not** be able to control it because of the kinematic calculations that would be needed to relate the "tool" position in X, Y and Z co-ordinates to the length and rotation of the machine arms..

Mach2 can switch the spindle on, rotating in either direction, and switch it off. It can also control the rate at which it rotates (rpm) and monitor its angular position for operations like cutting threads.

Mach2 can turn the two type of coolant on and off.

Mach2 will monitor the EStop and can take note of the operation of the reference switches, the guard interlock and limit switches

Mach2 will store the properties of up to 256 different tools. If, however, your machine has an automatic tool changer or magazine then you will have to control it yourself.

4.3 The EStop control

Every machine tool must have one or more Emergency Stop (EStop) buttons; usually with a big red mushroom head. They must be fitted so that you can easily reach one from wherever you might be when you are operating the machine.

Each EStop button should stop all activity in the machine as quickly as is safely possible; the spindle should stop rotating and the axes should stop moving. This should happen **without** relying on software so we are talking about relays and contactors. The circuit should tell Mach2 what you have done and there is a special, mandatory input for this. It will generally not be good enough to turn off the AC power for an EStop because the energy stored in DC smoothing capacitors can allow motors to run on for some considerable time.

The machine should not be able to run again until a "reset" button has been pressed. If the EStop button locks when pushed then the machine should not start when you release it by turning its head.

It will not generally be possible to continue machining a part after and EStop but you and the machine will at least be safe.

4.4 The PC parallel port

4.4.1 The parallel port and its history

When IBM designed the original PC (160k floppy disc drive, 64kbytes of RAM!) they provided an interface for connecting printers using a 25 conductor cable. This is the foundation of the Parallel port we have on most PCs today. As it is a very simple way of transferring data it has been used for many things other than connecting printers. You can transfer files between PC, attach copy protection "dongles", connect peripherals like scanners and Zip drives and of course control machine tools using it. The USB is taking over many of these functions and this conveniently leaves the parallel port free for Mach2.

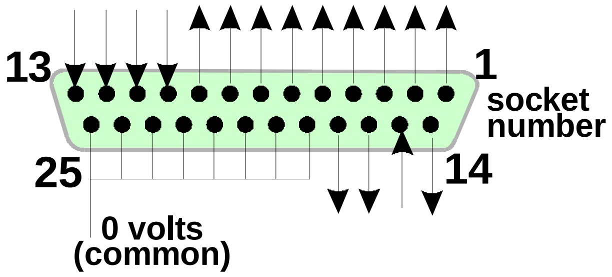


Figure 4.1 - Parallel port female connector seen from back of PC

The connector on the PC is a 25 way female "D" connector. Its sockets seen from the back of the PC are shown in figure 4.1. The arrows give the direction of information flow relative to the PC. Thus, for example, pin 15 is an input to the PC

4.4.2 Logic signals

You may wish to skip to the next heading on first reading and return here if you have to get involved with the nitty-gritty of interface circuits. It will probably be useful to read it with the documentation for you axis drive electronics.

All the signals output by Mach2 and input to it are binary digital (i.e. zeros and ones) These signals are voltages supplied by the output pins or supplied to the input pins of the parallel port. These voltages are measured relative to the computer's 0 volt line (which is connected to pins 18 to 25 of the port connector).

The first successful family (74xx series) of integrated circuits used TTL (transistor-transistor logic). In TTL circuits, any voltage between 0 and 0.8 volts is called "lo" and any voltage between 2.4 and 5 volts is called "hi". Connecting a negative voltage or anything above 5 volts to a TTL input will produce smoke.[1] The parallel port was originally built using TTL and to this day these voltages define its "lo" and "hi" signals. Notice that in the worst case there is only 1.6 volts difference between them. It is, of course, arbitrary whether we say that a "lo" represents a logic one or a logic zero. However, as is explained below, "lo" = one is actually better in most practical interface circuits.

For an output signal to do anything, some current will have to flow in the circuit connected to it. When it is "hi" current will flow **out** of the computer. When it is "lo" current will flow **into** the computer. The more current you have flowing the harder it is to keep the voltage near zero so the nearer to the permitted limit of 0.8 volts "lo" will become. Similarly a "hi" will be lower and nearer to the 2.4 volts lower likit. So with **too** much current the difference between "lo" and "hi" will be even less than 1.6 volts and things will become unreliable. Finally, it's worth noting you are allowed roughly 20 times more current flowing into a "lo" than you are allowed flowing out of a "hi".

So this means that it is best to assign logic 1 to be a "lo" signal. Fairly obviously this is called **active lo** logic. The main **disadvantage** of it is that the device connected to the parallel port has to have a 5 volt supply to it. This is sometimes taken from the PC game port socket or from a power supply in the device that is connected.

Turning to input signals, the computer will need to be supplied with some current (< 40 microamps) for "hi" inputs and will supply some (< 0.4 milliamps) for "lo" inputs.

4.4.3 Electrical noise and expensive smoke

Even if you skipped the previous section you had better read this one!

You will see that pins 18 to 25 are connected to the 0 volt side of the computer's power supply. All signals inside and outside the PC are relative to this. If you connect many long

[1] Some people think that integrated circuits work in some way by using smoke. Certainly no one has ever seen one work after the smoke has escaped!

wires to it, especially if they run near wires carrying high currents to motors, then these wires will have currents flowing in then that create voltages which are like noise and can cause errors. You might can even crash the computer.

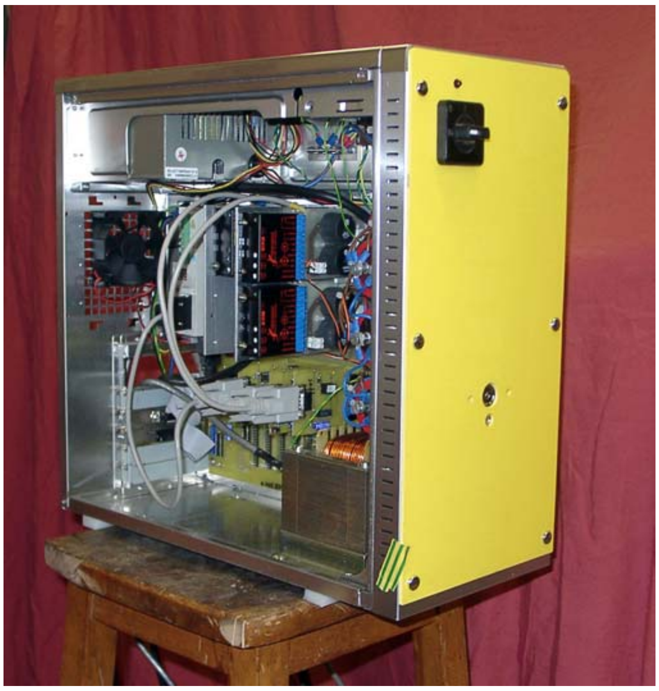


Figure 4.2 - Cased breakout board, drive electronics and power supply

The axis and perhaps spindle drives, which you will connect to Mach2 through your parallel port, are likely to work at between 30 and 240 volts and they will be able to supply currents of many amps. Properly connected they will do no harm to the computer but an accidental short circuit could easily destroy the entire computer mother-board and even the CD-ROM and hard drives as well.

For these **two** reasons you are very strongly advised to buy a device called an "isolating breakout board" which will provide you with terminals that are easy to connect to, a separate 0 volt (common) for the drives, home switches etc. and will take care of the permitted current in and out of the port. This breakout board, your drive electronics and power supply should be neatly installed in a metal case to minimise the risk of interference to your neighbours' radio and television signals. If you build a "rat's nest" you are inviting short circuits and tragedy. Figure 4.2 shows how a PC case, including some useful fans, can be converted for this role.

Here ends the sermon!

4.5 Axis drive options

4.5.1 Steppers and Servos

There are two possible types of motive power for axis drives:

- ♦ Stepper motor
- ♦ Servo motor (either AC or DC)

Either of these types of motor can then drive the axes through leadscrews (plain- or ball-nut), belts, chains or rack and pinion. The mechanical drive method will determine the speed and torque required and hence any gearing between the motor and machine.

Properties of a stepper motor drive are:

1. Low cost
2. Simple 4-wire connection to motor
3. Low maintenance
4. Motor speed limited to about 1000 rpm and torque limited to about 1000 ounce inches. (7 Nm). Getting the maximum speed depends on running the motor or the drive electronics at their maximum permitted voltage. Getting the maximum torque depends on running the motor at its maximum permitted current (amps)
5. For practical purposes on a machine tool steppers need to be driven by a chopped micro-stepping controller to ensure smooth operation at any speed with reasonable efficiency.

6. Open loop control means it is possible to lose steps under high loading and this may not immediately be obvious to the machine user.

On the other hand a servo motor drive is:

1. Relatively expensive (especially if it has an AC motor)
2. Needs wiring for both the motor and encoder
3. Maintenance of brushes is required on DC motors
4. Motor speed 4000 rpm plus and unlimited torque (if your budget can stand it!)
5. Closed loop control so drive position is always known to be correct (or a fault condition will be raised)

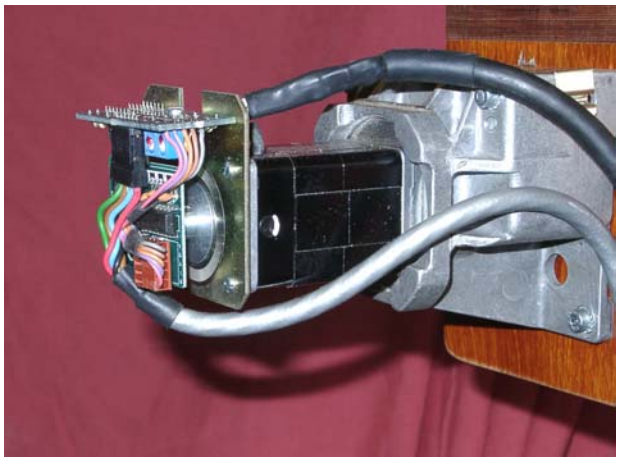


Figure 4.3 - Small DC servo motor with encoder (left) and gearbox

In practice stepper motor drives will give satisfactory performance with conventional machine tools up to a Bridgeport turret mill or a 6" centre height lathe unless you want exceptional accuracy and speed of operation.

It is worth giving two warnings here. Firstly servo systems on old machines are probably not digital; i.e. they are not controlled by a series of step pulses and a direction signal. To use an old motor with Mach2 you will need to discard the resolver (which gave the position) and fit a quadrature encoder and you will have to replace **all** the electronics. Secondly beware of secondhand stepper motors unless you can get manufacturer's data for them. They might be designed for 5-phase operation, may not work well with a modern chopped micro-stepping controller and might have a much lower rated torque than the same size of modern motor.. Unless you can test them, you may find that they have been accidentally demagnetised and so be useless. Unless you are really confident of your skills and experience, then the axis drives should be current products bought from suppliers who will support them. If you buy **right** then you will only need to buy **once**.

4.5.2 Doing Axis drive calculations

A full set of calculations for the axis drives would be very complicated and anyway you probably do not have all the necessary data (e.g what is the maximum cutting force you want to use). Some calculation is, however, necessary for success.

If you are reading the manual for an overview then you might like to skip this section.

Example 1 - MILL TABLE CROSS SLIDE

We start with checking the minimum possible move. This is an absolute limit to the accuracy of work done on the machine and then check rapid speeds and torque.

As an example suppose you are designing a mill cross-slide (Y axis) drive. You are going to use a screw with a 0.1" pitch single start thread and a ball nut. You want to aim for a minimum move of 0.0001" (resolution on diameter of two tenths of a thou). This is $^1/_{1000}$ of a revolution of the motor shaft if it is coupled directly to the screw.

With stepper motor

The minimum step with a stepper motor depends on how it is controlled. There are usually 200 full steps per revolution. You need to use micro-stepping for smooth running over the full range of feed speeds and many controllers will allow you to have 10 micro-steps per full step. This system would give $^1/_{2000}$ of a revolution as the minimum step which is fine.

Next look at the possible rapid feed speed. Assume, conservatively, that the maximum motor speed is 500 rpm. This would give a rapid of 50 inches/minute or about 15 seconds for the full slide travel. This would be satisfactory although not spectacular.

At this speed the micro-stepping motor drive electronics need 16,666 (500 * 200 * 10 / 60) pulses per second. On a 1 GHz PC, Mach2 can generate 35,000 pulses per second simultaneously on each of the six possible axes. So there are no problems here.

You now have to choose the torque that the machine will require. One way to measure this is to set up the machine for the heaviest facing cut you think you will ever make and, with a long lever (say 12") on the slide handwheel, turn it at the end with a spring balance (of set of spring kitchen scales). The torque for the cut (in ounce-inches) is the balance reading * 12. The other way is to use a motor size and specification that you know works on someone else's machine with the same type of slide and screw!

As the rapid feed speed was reasonable you could consider slowing it down by 2:1 gearing (perhaps by a toothed belt drive) which would nearly double the available torque on the screw.

With servo motor

Again we look at the size of one step. A servo motor has an encoder to tell its drive electronics. This consists of a slotted disc and will generate four pulses for each slot in the disc. Thus a 300 slot disc (which is fairly low for commercial encoders) will generate 1200 pulses per revolution of the motor shaft.

The drive electronics for the servo will usually turn the motor by one encoder pulse per input step. Some high specification servo electronics can multiply and/or divide the step pulses by a constant (e.g. one step pulse moves by 5 encoder pulses or 36/17 pulses). This is often called **electronic gearing**.

As the maximum speed of a servo motor is around 4000 rpm we will certainly need a speed reduction on the mechanical drive. 5:1 would seem sensible. This gives a movement of 0.0000167" per step which is much better than that required (0.0001")

What maximum rapid speed will we get? With 35,000 step pulses per second we get 5.83 revolutions [35000/(1200 * 5)] of the leadscrew per second. This is OK at about 9 seconds for 5" travel of the slide. Notice, however, that the speed is limited by the pulse rate from Mach2 **not** the motor speed. This is only about 1750 rpm in the example. The limitation would be even worse if the encoder gave more pulses per revolution. It will often be necessary to use servo electronics with electronic gearing to overcome this limitation.

Finally one would check on available torque. On a servo motor less safety margin is required than with a stepper motor because the servo cannot suffer from "lost steps". If the torque required by the machine is too high then the motor may overheat or the drive electronics raise an over-current fault.

Example 2 - ROUTER GANTRY DRIVE

For a gantry router might need a travel of 60" on this axis and a ballscrew for this length will be expensive and difficult to protect from dust. Many designers might go for a chain and sprocket drive.

We might choose a minimum step of 0.0005". A drive chain sprocket of 20 teeth with $^1/_4$" pitch chain gives 5" gantry movement per revolution of the sprocket. A stepper motor (ten micro-steps) gives 2000 steps per revolution so a 5:1 reduction (belt or gear box) is needed between the motor and sprocket shaft. [0.0005" = 5"/(2000 x 5)]

With this design if we get 500 rpm from the stepper then the rapid feed of 60" would, neglecting acceleration and deceleration time, take a reasonable 8.33 seconds.

The torque calculation on this machine is more difficult than with the cross slide as, with the mass of the gantry to be moved, inertia, during acceleration and deceleration, is probably more important than the cutting forces. The experience of others or experiments will be the best guide. If you join the ArtSoft user group for Master5/Mach1/ Mach2 on Yahoo! you will have access to the experience of hundreds of other users.

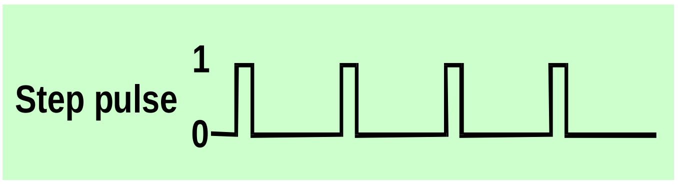


Figure 4.4 - Step pulse waveform

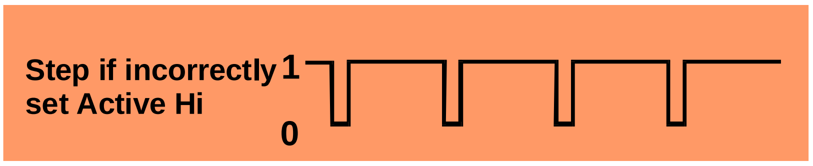


Figure 4.5 - Wrongly configured output alters step waveform

4.5.3 How the Step and Dir signals work

One pulse (logic 1) appears on the Step output for each step that the axis is to make. The Dir output will have been set before the step pulse appears.

The logic waveform will be like that shown in figure 4.4. The gap between the pulses will be smaller the higher the speed of the steps.

Drive electronics usually use the Active Lo representation for Step and Dir signals. Mach2 should be setup so these outputs are Active Lo. If this is not done then the Step signal still goes up and down but the drive thinks that the gaps between the pulses are the pulses and vice-versa and this often causes very rough or unreliable running of the motor. The "inverted" pulses are shown in figure 4.5.

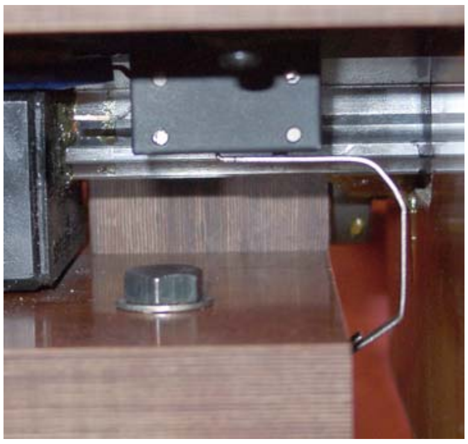


Figure 4.6 - Limit switch - microswitch mounted on the table is tripped by bed of machine

4.6 Limit and reference switches

4.6.1 Strategies

Limit switches are used to prevent any linear axis moving too far and so causing damage to the structure of the machine. You can run a machine without them but the slightest mistake setting up can cause a lot of expensive damage.

An axis may also have a **Reference switch**. Mach2 can be commanded to move one (or all) axes to the reference position. This will need to be done whenever the system is switched on so that it knows where the axes are currently positioned. If you do not provide a Reference switch then you will have to jog the axes by eye to a reference position. The Reference switch for an axis can be at the 0.0 co-ordinate in which case referencing also **homes** the axis.

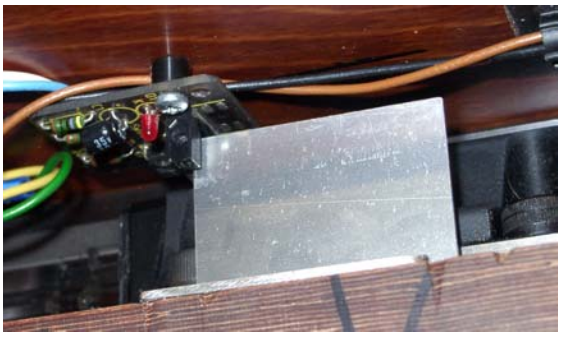


Figure 4.7 - Optical switch on table with vane on bed of machine

Thus each axis could need three switches (i.e. limit switches at the two ends of travel and a reference switch). So a basic mill would require nine parallel port inputs for them. This is not much good as the parallel port only has 5 inputs! The problem can be solved in two ways:

- The limit switches are connected to external logic (perhaps in the drive electronics) and this logic switches off the drives when the limit is reached. The separate reference switches are connected inputs to Mach2
- One pin can share all the inputs for an axis and Mach2 is responsible for controlling both limits and referencing.

The first method is best for a very large, expensive or fast machine where you might not wish to trust software to prevent mechanical damage. Switches connected to drive electronics can be intelligent and only allow motion away from a switch when the limit is hit. This is safer than disabling the limits so a user can jog the machine off its limits.

With the second method it is still possible to use only 3 inputs to Mach2 for a 3-axis mill (4 for a gantry type machine - see Slaving) and only two switches are required as one limit and reference can share a switch.

4.6.2 The switches

There are several choices you need to make when selecting switches:

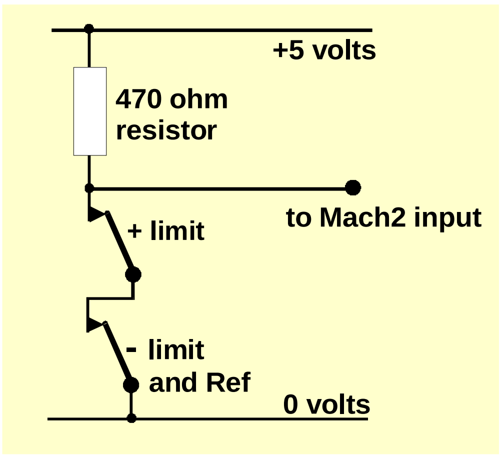


Figure 4.8 - Two NC mechanical switches give logic OR

If you are going to have two switches sharing an input then they need to be connected so the signal is a logic "1" if **either** switch is operated (i.e. the logical OR function). This is easy with mechanical switches. If they have normally closed contacts and are wired in series as shown in figure 4.8, then they will give an Active Hi signal if either switch is operated. Note that for reliable operation you need to "pull up" the input to the parallel port. As mechanical switches can carry a significant current a value of 470R is shown which gives a current of about 10 milliamps. As the wiring to the switches might be quite long and liable to pickup of noise make sure that you have a good connection to the 0 volt side of your input (the frame of your machine tool will not be satisfactory) and consider using shielded cable with the shield connected to the main ground terminal of your controller.

If you use electronic switches like a slotted detector with a LED and photo-transistor, then you will need some sort of an OR gate (which could be a "wired-or" if an Active Lo input is driven by open collector transistors).

Optical switches, if out of the way of coolant, should be OK on a metalworking machine but are liable to malfunction with wood dust.

Don't use magnetic switches (reed switches or Hall effect devices) on a machine that may cur ferrous metal of the swarf will "fuzz-up" the magnet.

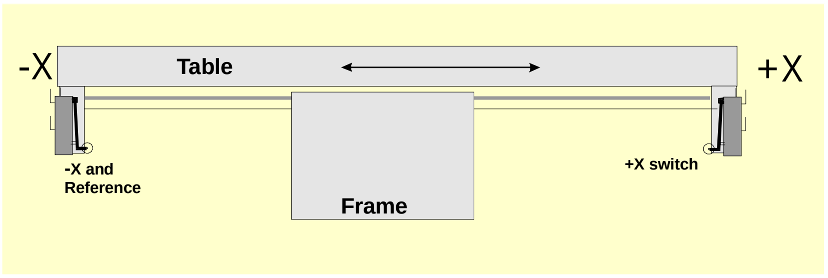


Figure 4.9 - Two switches operated by frame with overtravel avoided by mechanical stops

The repeatability of the operating point, particularly with mechanical switches, is very dependent on the quality of the switch and the rigidity of its mounting and actuating lever. The setup in Figure 4.6 would be very imprecise. The repeatability is very important for a switch to be used as a reference, particularly on a lathe cross-slide See the discussion of *Tool Tables* for lathes for a way of maintaining accuracy greater than that of the reference switch.

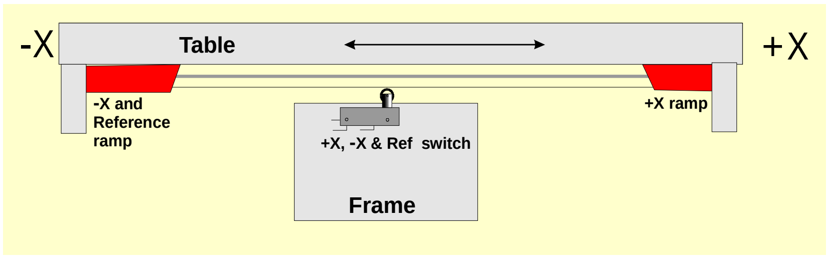


Figure 4.10 - Ramps operating one switch

Overtravel is the movement of the switch that occurs after it has operated. With a limit switch it can be caused by the inertia of the drive. On an optical switch like figure 4.7 then provided the vane is long enough there will be no difficulties. A microswitch can be given arbitrary overtravel by operating a roller on it by a ramp (see figure 4.10). The slope of the ramp does, however, reduce the repeatability of operation of the switch. It is often possible to use one switch for both limits by providing two ramps or vanes.

4.6.3 Where to mount the switches

The choice of mounting position for switches is often a compromise between keeping them away from swarf and dust and having to use flexible rather than fixed wiring.

For example figures 4.6 and 4.7 are both mounted under the table, despite the fact that they need a moving cable, as they are much better protected there.

You might find it convenient to have one moving cable with the wires in it for two or more axes (e.g. the X and Y axes of a gantry router could have switches on the gantry itself and a very short cable loop for the Z axis could then join the other two). Do not be tempted to share a multi-way cable between motor and switch wiring. You may want to run two separate cables together and this will not cause trouble if both a shielded (with braid or foil) and the shields are grounded to one common point at the electronic drives.

You might find it helpful to look at commercial machines and pictures of examples on the ArtSoft Yahoo! group for more ideas and techniques for switches.

4.6.4 How Mach2 uses shared switches

For a full understanding of this you will also have to read the section on configuring Mach2, but the basic principle is easy. You connect the two limit switches to one input (or have one switch and two vanes or ramps). You define, to Mach2, a direction as the direction to travel to move when looking for a reference switch. The limit switch (vane or ramp) at that end of the axis is also the reference.

In normal use when Mach2 is moving an axis and sees its limit input become active it will stop running (like an EStop) and display that a limit switch has been tripped. You will be unable to move the axes unless:

1) *Auto limit override* is switched on (by a button on the Corrections screen). In this case you can click Reset and jog off the limit switch. You should then reference the machine
2) You click *Override limits*. A red flashing LED warns you of the temporary override. This will again allow you Reset and to jog off the switch and will then turn itself and the flashing LED off. Again you should reference the machine. An input can also be defined to override the limit switches.

Note, however, that you will not be prevented, in either case, from jogging further onto the switch and maybe crashing the axis in a mechanical stop. **Take great care**.

4.6.5 *Referencing in action*

When you request referencing (by button or G-code) the axis (or axes) will travel (at a selectable low speed) in the defined direction until the switch operated. The axis will then move back in the other direction so as to be off the switch. During referencing the limits do not apply.

When you have referenced an axis then zero or some other value can be loaded into the axis DRO as its absolute machine co-ordinate. If you use zero then the reference switch position is also the Home position of the axis. If the reference goes in the negative direction of an axis (usual for X and Y) the you might get referencing to load something like -0.5" into the DRO. This means that the Home is half an inch clear of the limit. This wastes a bit of the axis travel but if you overshoot, when jogging to Home, you will not accidentally trip the limits. See also Software Limits as another way of solving this problem.

If you ask Mach2 to reference **before** you jog off the switch then it will travel in the opposite direction (because it says that you are already on the Reference switch) and stop when you get off the switch. This is fine when you have a separate Reference switch or are on the limit at the reference end of the axis. If, however, you are on the other Limit switch (and Mach2 cannot know this as they are shared) then the axis moves for ever away from the actual reference point until it crashes. So the advice is: **always jog carefully off the limit switches, then reference.**

4.6.6 *Other Reference and Limit options and hints*

Reference switch not near limit switch

It is sometimes not very convenient to have the reference switch at a limit of travel. Consider a large moving column floor mill or a big planer-mill. The Z travel on the column might be 8 feet and could be quite slow without affecting the overall cutting performance of the machine. If, however, the reference position is the top of the column then referencing might involve nearly 16 feet of slow Z travel. If the reference position was chosen half way up the column then this time can be halved. Such a machine would have a separate reference switch for the Z axis (thus requiring another input on the parallel port but still only four inputs in a three axis machine) and would use the ability of Mach2 to set any value for an axis DRO, after referencing, to make Home the top of the column.

This technique might also be used on a lathe to avoid having to move the tailstock to the end of the bed when referencing.

Separate high accuracy reference switch

The X axis on a lathe might have a separate reference switch to achieve the required accuracy.

Limit switches of multiple axes connected together

Because Mach2 does not take any notice of **which** limit of which axis has tripped, then all the limits can be ORed together and fed into one limit input. Each axis can then have its own reference switch connected to the reference input. A three axis machine still only needs four inputs.

Slaving

On a gantry type miller or router where the two "legs" of the gantry are driven by separate motors then each motor should be driven by its own axis. Suppose the gantry moves in the Y direction then axis A should be defined as a linear (i.e. non-rotational) axis and A should be slaved to Y - see the chapter on Configuring Mach2 for details. Both axes should have limit and reference switches. In normal use both Y and A will be sent exactly the same step and direction commands by Mach2. When a Reference operation is performed then the axes will run together until the final part of referencing which is moving just off the reference switches. Here they will move so that each stops the same distance off its own switch. Referencing will therefore correct any racking (i.e. out of squareness) of the gantry which might have occurred when the machine is switched off or due to lost steps.

4.7 Spindle control

There are three different ways in which Mach2 can control your "spindle" or you can ignore all of these and control it manually.

1. Turn motor On (Clockwise or Counterclockwise) and Turn motor Off
2. Motor controlled by Step and Direction pulses (i.e. spindle motor is a servo)
3. Motor controlled by a pulse width modulated signal

1. On/Off motor control

M3 and a screen button will request that the spindle starts in a clockwise direction. M4 will request that the spindle starts in an counterclockwise direction. M5 requests that the spindle stops. M3 and M4 can be configured to activate external output signals which can be associated with output pins on the parallel ports. You then wire these outputs (probably via relays) to control the motor contactors for your machine.

Although this sounds straightforward, in practice **you need to be very careful**. Unless you really need to run the spindle "backwards" it would be better to treat M3 and M4 as the same or to allow M4 to activate a signal which you do not connect to anything.

Clearly it is possible, in an error situation, for the clockwise and counterclockwise signals to be active together. This may cause the contactors to short the mains supply. Special mechanically interlocked reversing contactors can be obtained and if you are going to allow your spindle to run counterclockwise then you need to use one. Another difficulty is that the "G-code" definition says that it is legal to issue an M4 when the spindle is running clockwise under an M3 (and vice-versa). If your spindle drive is an AC motor just changing the direction when running at full speed is going to impose very large forces on the mechanical drive of the machine and will probably blow the AC fuse or trip a circuit breaker. For safety you need to introduce time delays on the operation of the contactors or use a modern inverter drive which allows you to change direction with a running motor.

See also the note about the limited number of External Activation Signals in the section on Coolant.

2. Step and Direction motor control

If your spindle motor is a servomotor with a step and direction drive (like the axis drives) then you can configure two output signals to control its speed and direction of rotation. Mach2 will take account of a variable step pulley drive or gearbox between the motor and the spindle. For full details see Motor Tuning in chapter 5

3. PWM motor control

As an alternative to Step and Direction control, Mach2 will output a pulse width modulated signal whose duty cycle is the percentage of full speed that you require. You could, for example, convert the duty cycle of the signal to a voltage (PWM signal on for 0% of time gives 0 volts 50% gives 5 volts and 100% gives 10 volts) and use this to control an induction motor with a variable frequency inverter drive. Alternatively the PWM signal could be used to trigger a triac in a simple DC speed controller. The PWM signal is output on the spindle Step pin. You will need to take special precautions to switch off the motor at low speeds as the Motor Clockwise/Counterclockwise outputs are not used.

4.8 Coolant

Output signals can be used to control valves or pumps for flood and mist coolant. These are activated by screen buttons and/or M7, M8, M9.

Note: Mach2 only has three External Activation signals which must be shared between the four functions of *Spindle clockwise*, *Spindle counterclockwise*, *Flood coolant* and *Mist coolant* together with any additional things that you want to control through your own Macros (e.g. *Shuttle pallet* etc.)

4.9 Plasma torch height control

Mach2 will control a plasma cutting torch and table. Some external logic is required to assess the correct torch height. This might be a measurement of the arc voltage. Mach2 has an input to say that the torch is ready to have its height controlled and a Move down and Move up signal. These signals will apply a correction to the Z value commanded by the Part Program to achieve the torch height defined by the external logic despite working over a sheet which might not be flat.

If you do not have extensive experience with this type of equipment then the earlier advice on buying a complete, already interfaced and supported product is **particularly important** on account of the high voltages and large currents being used by the torch.

4.10 Digitise probe

Mach2 can be connected to a digitising probe to make a measuring and model digitising system. There is an input signal that indicates that the probe has made contact and provision (not yet implemented) for an output to request that a reading is taken by a non-contact (e.g. laser) probe.

To be useful the probe needs to have an accurately spherical end (or at least a portion of a sphere) mounted in the spindle with its center accurately on the centerline of the spindle and a fixed distance from a fixed point in the Z direction (e.g. the spindle nose). To be capable of probing non metallic materials (such as a model to be digitised will be made from) the probe requires to make (or break) a switch with a minute deflection of its tip in any (XY or Z) direction). Ir the probes is to be used with an automatic toolchanger then it needs to be "cordless".

These requirements are a major challenge for the designer of a probe to be built in a home workshop and commercial probes are not cheap.

4.11 Linear (glass scale) encoders

Mach2 has three pairs of inputs to each of which an encoder with quadrature outputs can be connected (typically these might be "glass scale" encoders - see figure 4.11). Mach 2 will display the position of each of these encoders on a dedicated DRO. These values can be loaded from and saved to the main axis DROs.

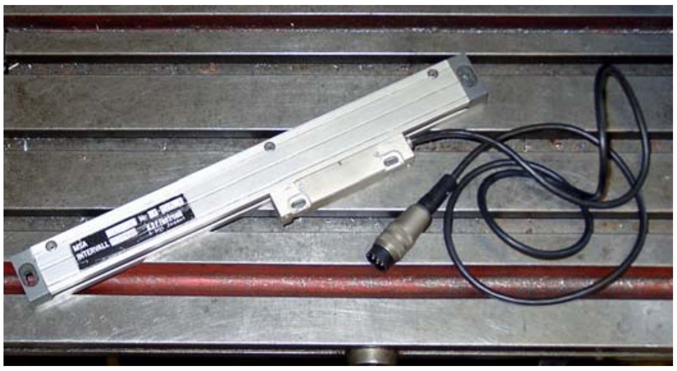


Figure 4.11 - Glass scale encoder (awaiting installation)

Inside the case of the encoder is a glass (or sometimes plastic) strip ruled with lines (e.g often 10 microns wide) separated by the same sized clear space. A light shining on a phototransistor through the ruling would give a signal like A in figure 4.12. One complete cycle corresponds to a movement of 20 microns.

Another light and phototransistor located 5 microns away from the first one would give signal B a quarter of a cycle out from A (hence the name *quadrature*)

A full explanation is rather long, but you will notice that a signal changes every 5 microns of movement so the resolution of the scale is 5 microns. We can tell which way it is moving by the sequence of changes. For example if B goes from lo to hi when A is hi (point *x*) then we are moving to the right of the marked start whereas if B goes from hi to lo when A is hi (point *y*) then we are moving to the left of the start.

Mach2 expects logic signals. Some glass scales (e.g certain Heidenhain models) give an analogue sinewave. This allows clever electronics to interpolate to a higher resolution than 5 microns. If you want to use these than you need to square off the waveform with an operational amplifier/comparator. TTL output encoders will connect directly to the input pins of the parallel port but, as noise will give false counts, they are better interfaced via what is known as a Schmitt trigger chip. The scales require a DC supply (often 5 volts) for the lights and any driver chips in them.

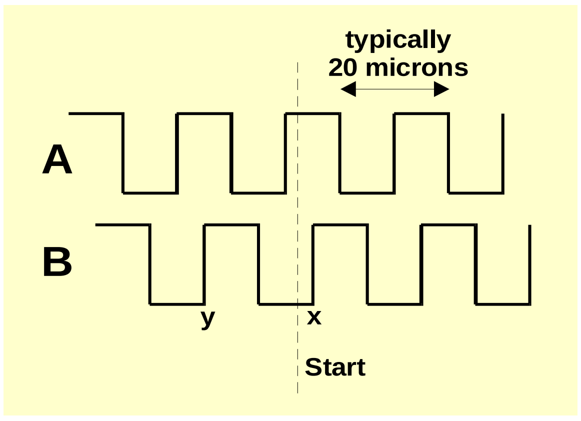


Figure 4.12 - Quadrature signals

Notice:

(a) that you can not easily use a linear scale as the feedback encoder for a servo drive as the slightest backlash or springiness in the mechanical drive will make the servo unstable.

(b) it is not easy to connect the rotary encoders on the servo motor to the encoder DROs. This would be attractive for manual operation of the axes with position readout. The problem is that the 0 volt (common) inside the servo drive used for the motor encoders is almost certainly not the same 0 volt as your PC or breakout board. Connecting them together will cause problems - don't be tempted to do it!

(c) the main benefit of using linear encoders on linear axes is that their measurements do not depend on the accuracy or backlash of the drive screw, belt, chain etc.

4.12 Spindle index pulse

Mach2 has an input for a pulse generated each revolution of the spindle. It uses this to co-ordinate the movement of the tool and work when cutting threads and for orientating the tool for the back boring canned cycle. It can be used to control feed on a per-rev rather than per-minute basis.

4.13 Charge pump - a pulse monitor

Mach2 will output a constant pulse train whose frequency is at least 25kHz whenever it is running correctly. This will not be there if the program has not been loaded yet, does an EStop or fails in some way. You can use this signal to charge a capacitor through a diode pump (hence the name) or, probably more simply, keep triggering a retriggerable monostable whose output enables your axis and spindle drives etc. In essence this output gives you a lot of confidence that all other outputs of Mach2 are meaningful.

4.14 Other functions

Mach2 defines up to four Activation input signals which you can assign for your own use. For example they can be tested in user written macros. Input #1 can be used to inhibit running of the part program. It might be connected to the guards on your machine.

The three external activation outputs have already been mentioned under Spindle and Coolant. Any spares can be used by you and controlled in user written macros.

And a final thought - before you get carried away with implementing too many of the features in this chapter, remember that you do not have an unlimited number of inputs/outputs. Even with two parallel ports there are only ten inputs for supporting all functions.

5. Configuring Mach2 to work with your machine tool and drives

If you have bought a machine tool with a computer running Mach2 then you will probably not need to read this chapter (except out of general interest). Your supplier will probably have installed the Mach2 software and set it up and/or will have given you detailed instructions on what to do.

You are recommended to ensure that you have a paper copy of how Mach2 is configured should you ever need to re-install the software from scratch. Mach2 will save the information in a file for you or you can use the blank "screens" in appendix 5 to record the information yourself.

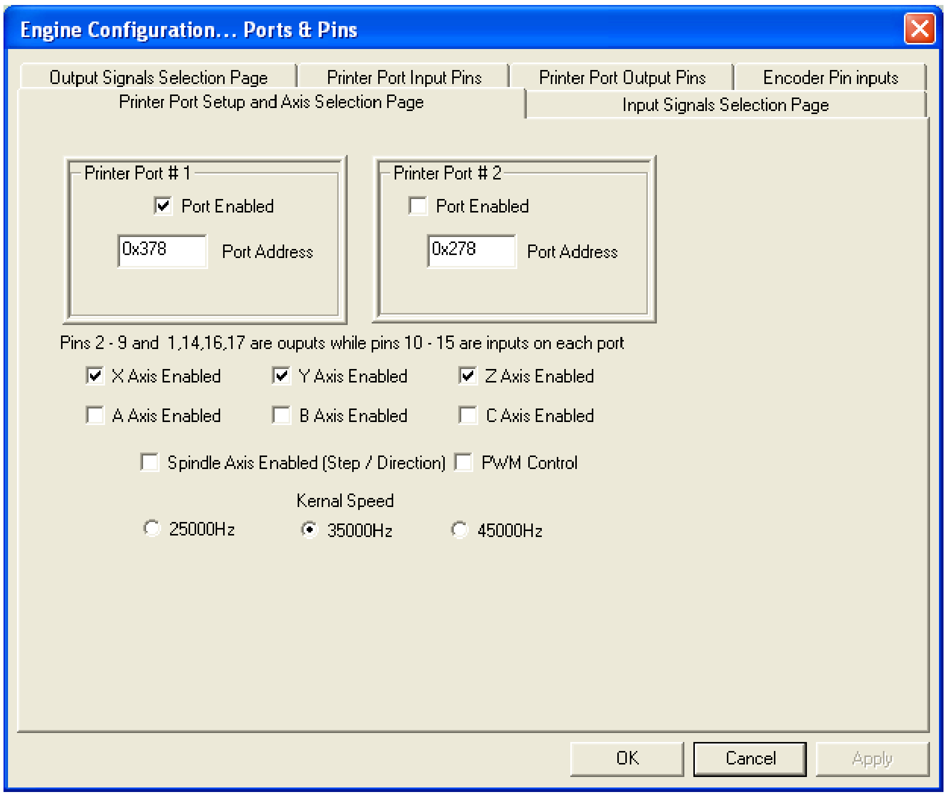
5.1 A configuration strategy

This chapter contains a lot of very fine detail. You should, however, find that the configuration process is straightforward if you take it step-by-step testing as you go. A good strategy is to skim through the chapter and then work with it on your computer and machine tool. We will assume that you have already installed Mach2 for the dry running described in chapter 3.

Virtually all the work you will do in this chapter is based on dialog boxes reached from the Configure menu. These are identified by, for example, Configure>Logic which means that you choose the Logic entry from the Configure menu.

5.2 Initial configuration

The first dialog to use is Configure>Ports and Pins. This dialog has many tabs but the initial one is as shown in figure 5.1.


Engine Configuration... Ports & Pins

Printer Port # 1
[✓] Port Enabled
0x378 Port Address

Printer Port # 2
[ ] Port Enabled
0x278 Port Address

Pins 2 - 9 and 1,14,16,17 are ouputs while pins 10 - 15 are inputs on each port

[✓] X Axis Enabled [✓] Y Axis Enabled [✓] Z Axis Enabled
[ ] A Axis Enabled [ ] B Axis Enabled [ ] C Axis Enabled
[ ] Spindle Axis Enabled (Step / Direction) [ ] PWM Control

Kernal Speed
25000Hz [✓ 35000Hz] 45000Hz
Output Signals Selection Page
Printer Port Input Pins
Printer Port Output Pins
Encoder Pin inputs
Printer Port Setup and Axis Selection Page
Input Signals Selection Page
OK
Cancel
Apply

Figure 5.1 - Ports and Axis selection tab

5.2.1 Defining addresses of port(s) to use

If you are only going to use one parallel port and it is the one on your computer's motherboard then the default address of Port 1 of 0x378 (i.e. Hexadecimal 378) is almost certainly correct.

If you are using one or more PCI add-on cards then you will need to discover the address each is responds to. There seem to be no standards! Run the Windows Control Panel from the Windows *Start* button. Double click on *System* and choose the *Hardware* tab. Click the *Device Manager* button. Expand the tree for the item "Ports (COM & LPT)".

Double click the first LPT or ECP port. Its properties will be displayed in a new window. Choose the Resources tab. The first number in the first IO range line is the address to use. Note the value down and close the Properties dialog.

If you are going to use a second port repeat the above paragraph for it.

Close the Device Manager, System Properties and Control Panel windows.

Enter your first port's address (with the 0x prefix to say it is Hexadecimal) and if necessary check Enabled for port 2 and enter its address.

Now click the *Apply* button to save these values. This is most important. **Mach2 will not remember values when you change from tab to tab or close the Port & Pins dialog unless you Apply.**

5.2.2 Defining axes to be used

5.2.2.1 Controlled axes

Check the axes which you are going to control. You will almost certainly want X, Y & Z. If you have a gantry type machine with two motors/screws for the gantry then you will need another axis to slave to the main drive. If you have a rotary table or dividing head then you will want to reserve an axis to it. It is usual to allocate A as a rotary axis that turns about the X axis, B for Y or C for Z but this is only a convention which you can break if you wish.

5.2.2.2 Spindle

The spindle is not strictly an axis but it is configured here. If you are going to use stop/start control or indeed turn it on and off and set its speed manually then you leave both the Spindle Axis enable and PWM control boxes unchecked. In this case the S word in you part program will not be able to control the actual spindle speed although it can tell Mach2 what speed you have set it to which can be used to define a feed-per-rev feedrate.

If your spindle is driven by a servo or stepper motor (i.e. by step and direction pulses) then check the Enabled box and leave PWM unchecked.

If you are going to control your spindle speed by a Pulse Width Modulated (PWM) variable speed drive then you should check both boxes.

See also the Index Pulse input pin for details of measuring actual spindle speed.

5.2.3 Defining engine frequency

The Mach2 driver can work at a frequency of 25,000 Hz (pulses per second), 35,000 Hz or 45,000 Hz depending on the speed of your processor and other loads placed on it when running Mach2. Computers with a 1 GHz clock speed will almost certainly run well at 35,000 Hx so choose this unless you have a slower computer when you should choose 25,000.Hz You can up the speed and re-tune your motors when you find the system works properly at the initial speed.

Don't forget to click the *Apply* button before proceeding.

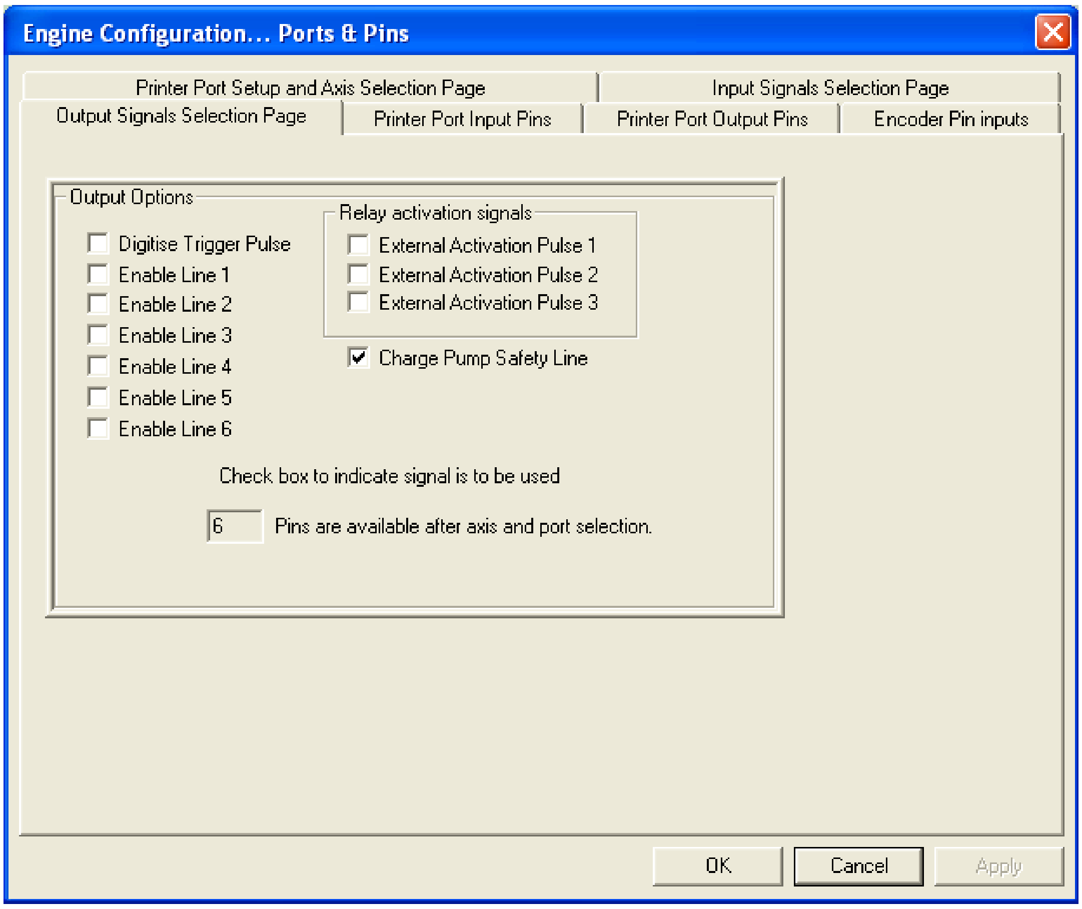
5.3 Defining input and output signals that you will use

Now you have established the basic configuration it is time to define which input and output signals you will be going to use. You will not need to allocate pins on the ports until the next stage.

The documentation for you breakout board may give guidance on what outputs to use if it has been designed for use with Mach2.

5.3.1 Output signals to be used

First view the *Output Signals Selection Page* tab. This will look like figure 5.2.


Engine Configuration... Ports & Pins

Printer Port Setup and Axis Selection Page | Input Signals Selection Page | Output Signals Selection Page | Printer Port Input Pins | Printer Port Output Pins | Encoder Pin inputs

Output Options
- [ ] Digitise Trigger Pulse
- [ ] Enable Line 1
- [ ] Enable Line 2
- [ ] Enable Line 3
- [ ] Enable Line 4
- [ ] Enable Line 5
- [ ] Enable Line 6

Relay activation signals
- [ ] External Activation Pulse 1
- [ ] External Activation Pulse 2
- [ ] External Activation Pulse 3

- [✓] Charge Pump Safety Line

Check box to indicate signal is to be used

6 Pins are available after axis and port selection.

OK Cancel Apply

Figure 5.2 Output Signals selection

You will probably only want to use one Enable output (as all the axis drives can work from it). Indeed if you are using the pulse monitor feature then you may enable your axis drives from its output.

The relay activation signals are for use to control a stop/start spindle (clockwise and optionally counterclockwise) and the Flood and Mist coolant pumps or valves. Notice that you have only got three signals so must compromise on which of the four functions you will control. If you are going to share spindle and coolant, or indeed control them by hand, then you can check as many or few signals as you wish.

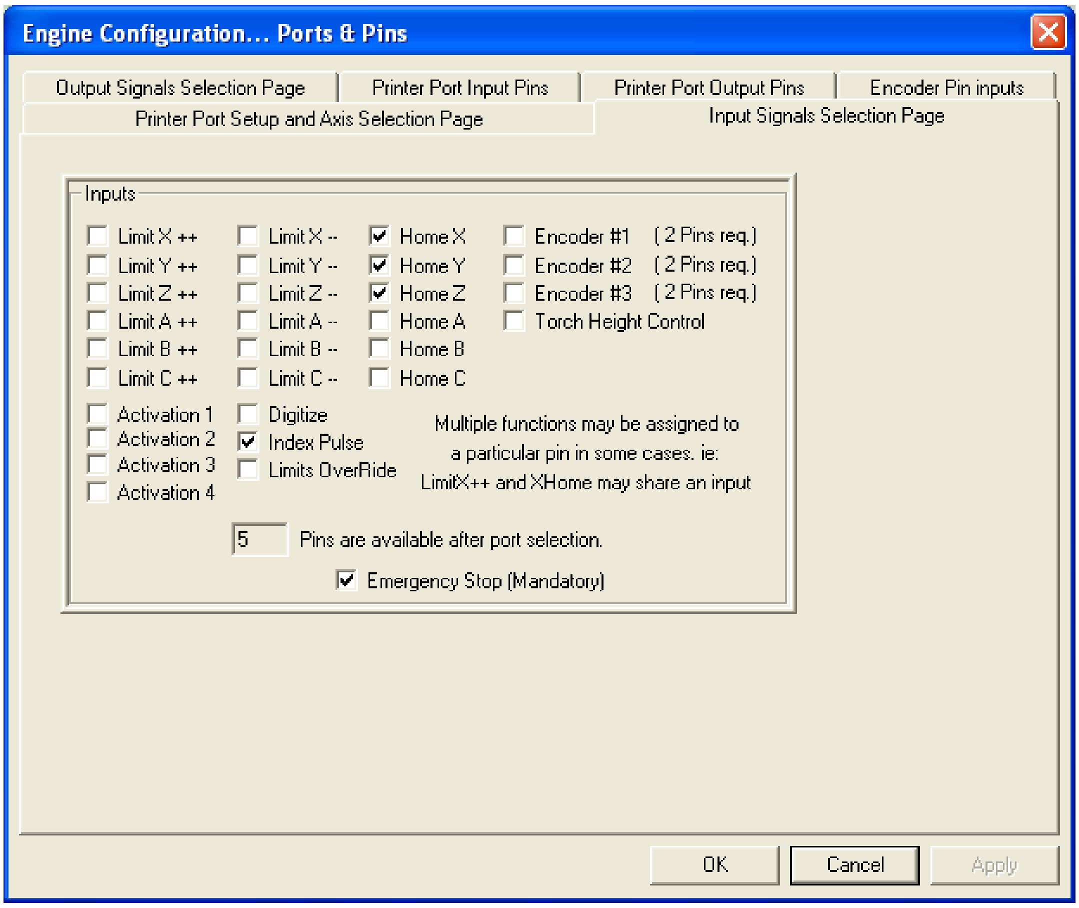
The *Charge Pump Safety* line should be checked if your breakout board accept this pulse input to continually confirm correct operation of Mach2.

You do not have to define the outputs for driving your axes (and perhaps spindle) as this was implicitly done when you defined the axes you are going to use.

Click the *Apply* button to save the data on this tab.

5.3.2 Input signals to be used

Now select the Input Signals Selection Page tab. This will look like figure 5.3.


Engine Configuration... Ports & Pins

Output Signals Selection Page | Printer Port Input Pins | Printer Port Output Pins | Encoder Pin inputs | Printer Port Setup and Axis Selection Page | Input Signals Selection Page

Inputs

☐ Limit X ++	☐ Limit X --	✓ Home X	☐ Encoder #1 (2 Pins req.)
☐ Limit Y ++	☐ Limit Y --	✓ Home Y	☐ Encoder #2 (2 Pins req.)
☐ Limit Z ++	☐ Limit Z --	✓ Home Z	☐ Encoder #3 (2 Pins req.)
☐ Limit A ++	☐ Limit A --	☐ Home A	☐ Torch Height Control
☐ Limit B ++	☐ Limit B --	☐ Home B	
☐ Limit C ++	☐ Limit C --	☐ Home C	
☐ Activation 1	☐ Digitize		
☐ Activation 2	✓ Index Pulse		
☐ Activation 3	☐ Limits OverRide		
☐ Activation 4			

Multiple functions may be assigned to a particular pin in some cases. ie: LimitX++ and XHome may share an input

5 Pins are available after port selection.

✓ Emergency Stop (Mandatory)

OK | Cancel | Apply

Figure 5.3 - Input signal selection

We assume that you have used one of the strategies from chapter 4.6. If the limit switches are connected together and trigger an EStop or disable the axis drives via the drive electronics then you do not check any of the Limit inputs. You will probably have reference switches on the X, Y and Z axes (maybe also one of the others if you are going to slave an axis for a gantry). Check the "Home" boxes for these axes. If you are combining limits and the reference switch then you should check *Limit --*, the *Limit ++* and *Home* for each axis.

True rotary axes, which you will define later, will not have limits but you may choose to configure a reference switch (say at 0 degrees on a dividing head)

The *Activation input #1* can be used to inhibit running a part program when safety guards are not in place. The other three (and #1 if not used for the guard interlock) are available for your own use and can be tested in the code of macros. You may wish to configure them later.

Check *Digitise* if you are going to connect a probe.

Check *Index Pulse* if you have a sensor that give a pulse for each revolution of the machine spindle. This can be used to display true spindle speed and to set the feedrate relative to this speed.

Check *Limits Override* if you are letting Mach2 control your limit switches and you have an external button which you will press when you need to jog off a limit. If you have no switch then you can use a screen button to achieve the same function.

Check *Torch Height Control* if you are going to use the height control features for a Plasma Torch.

Check the relevant *Encoder* inputs (#1 to #3) if you are going to connect "glass scale" encoders to directly measure the position of your axes. You do not check these boxes for the encoders on your servo motors in a servo drive system.

You **must** check the *Emergency Stop* box.

If you have one parallel port then you have 5 available inputs; with two ports there are 10. You will be able to share Limits and Home signals. Notice that encoders require two input per axis (for the A and B quadrature signals that they generate). It is very common to find that you are severely short of input signals and you may have to compromise on having things like a physical Limit Override switch to save signals!

Click the *Apply* button to save the data on this tab.

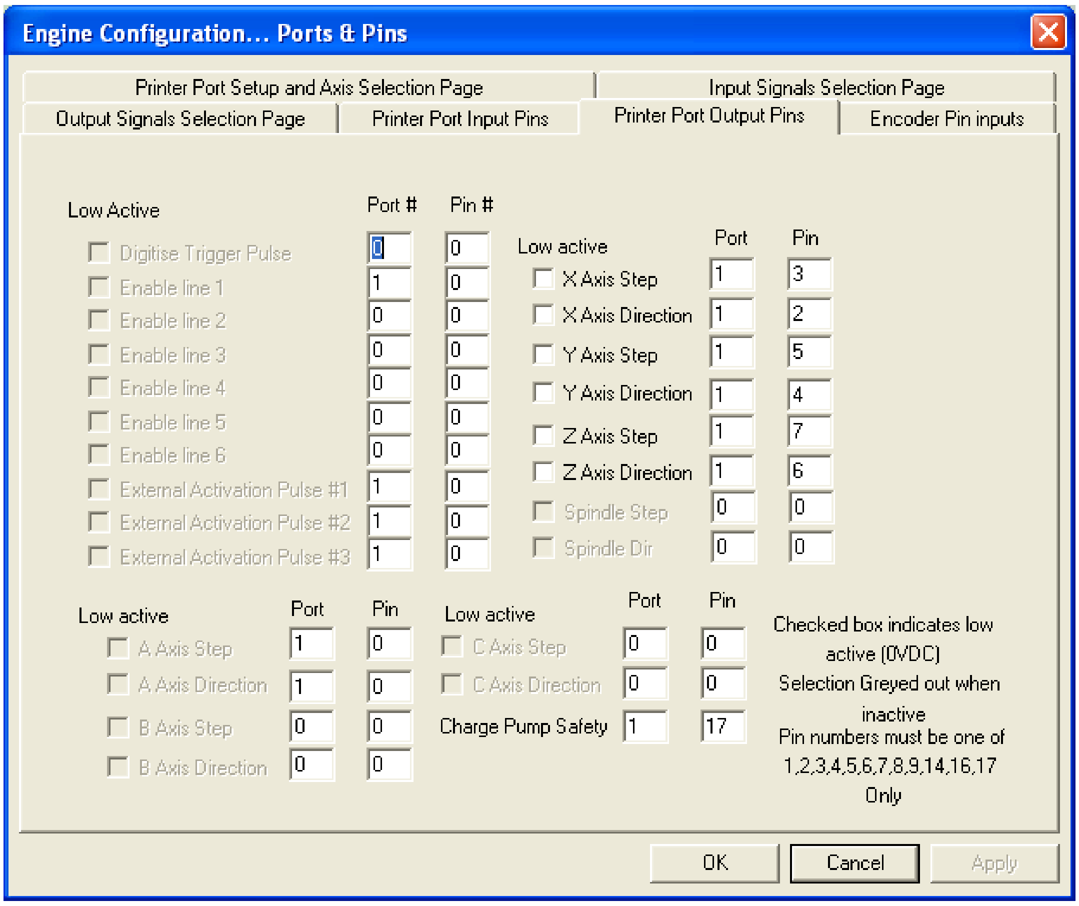
5.4 Allocating Inputs and Outputs to ports and pins

Next you have to define which pins on the parallel port(s) are to be used for the output and input signals you have said you need.

This mapping of signals to pins obviously depends on the actual connections from the 25D connectors of the ports, through any breakout board to your switches and axis drives. You will need to refer to the documentation of your breakout board and/or drive electronics to understand these connections.

5.4.1 Defining output pins

You enter the pins that you want to use for each output signal on the *Printer Port Output Pins* tab. This is illustrated in figure 5.4.


Engine Configuration... Ports & Pins
Printer Port Setup and Axis Selection Page
Input Signals Selection Page
Output Signals Selection Page
Printer Port Input Pins
Printer Port Output Pins
Encoder Pin inputs
OK
Cancel
Apply

Low Active	Port #	Pin #
Digitise Trigger Pulse	0	0
Enable line 1	1	0
Enable line 2	0	0
Enable line 3	0	0
Enable line 4	0	0
Enable line 5	0	0
Enable line 6	0	0
External Activation Pulse #1	1	0
External Activation Pulse #2	1	0
External Activation Pulse #3	1	0

Low active	Port	Pin
X Axis Step	1	3
X Axis Direction	1	2
Y Axis Step	1	5
Y Axis Direction	1	4
Z Axis Step	1	7
Z Axis Direction	1	6
Spindle Step	0	0
Spindle Dir	0	0

Low active	Port	Pin
A Axis Step	1	0
A Axis Direction	1	0
B Axis Step	0	0
B Axis Direction	0	0

Low active	Port	Pin
C Axis Step	0	0
C Axis Direction	0	0
Charge Pump Safety	1	17

Checked box indicates low active (0VDC)
Selection Greyed out when inactive
Pin numbers must be one of 1,2,3,4,5,6,7,8,9,14,16,17 Only

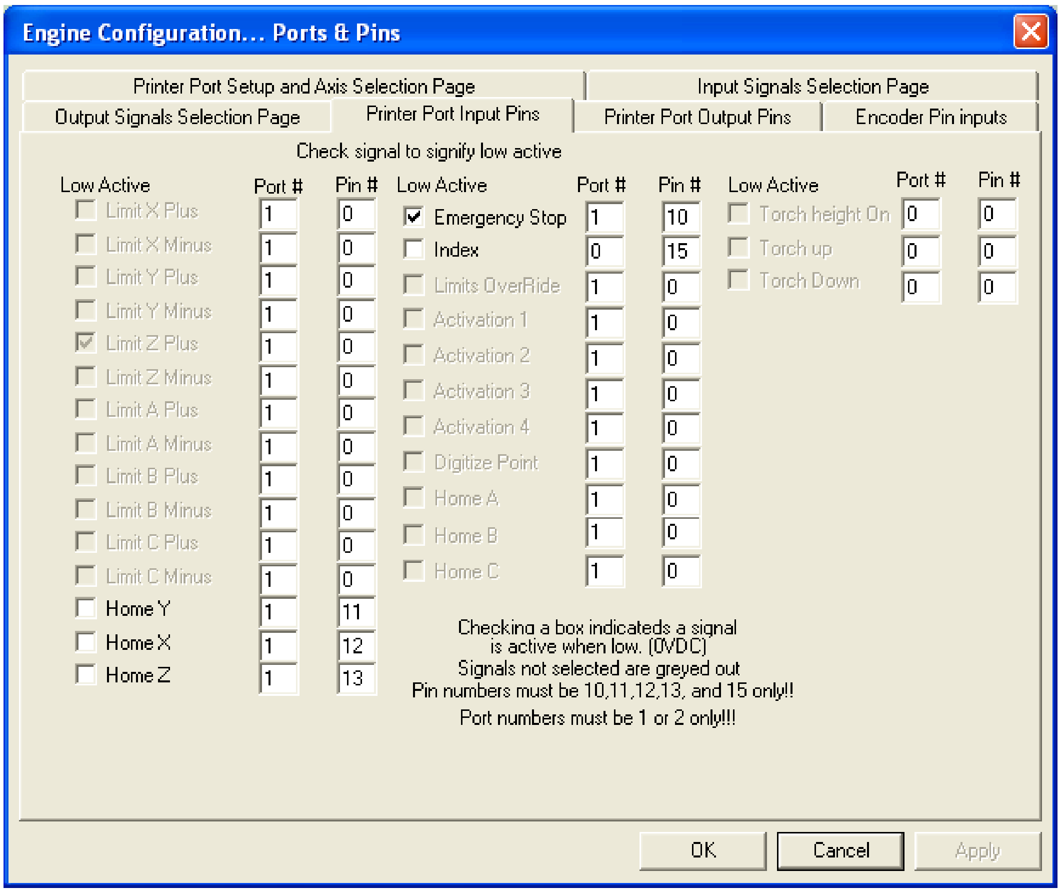
Figure 5.4 - Parallel Port Output Pins assignment

The screen shows the pins that may be used for output. Functions that were not selected in the earlier stage will be greyed out. If you only have one parallel port then the Port number must always be 1 otherwise you can use 1 or 2. If you want an logic "one" to pull the signal "Lo" then you should check the *Low active* box. As discussed in chapter 4 this logic convention is often used with TTL as it is less susceptible to noise.

Click the *Apply* button to save the data on this tab.

5.4.2 Defining input pins

Now define the input pins on the *Printer Port Input Pins* tab. This is illustrated in figure 5.5.



Engine Configuration... Ports & Pins

Printer Port Setup and Axis Selection Page | Input Signals Selection Page
Output Signals Selection Page | Printer Port Input Pins | Printer Port Output Pins | Encoder Pin inputs

Check signal to signify low active

Low Active	Port #	Pin #	Low Active	Port #	Pin #	Low Active	Port #	Pin #
☐ Limit X Plus	1	0	✓ Emergency Stop	1	10	☐ Torch height On	0	0
☐ Limit X Minus	1	0	☐ Index	0	15	☐ Torch up	0	0
☐ Limit Y Plus	1	0	☐ Limits OverRide	1	0	☐ Torch Down	0	0
☐ Limit Y Minus	1	0	☐ Activation 1	1	0			
✓ Limit Z Plus	1	0	☐ Activation 2	1	0			
☐ Limit Z Minus	1	0	☐ Activation 3	1	0			
☐ Limit A Plus	1	0	☐ Activation 4	1	0			
☐ Limit A Minus	1	0	☐ Digitize Point	1	0			
☐ Limit B Plus	1	0	☐ Home A	1	0			
☐ Limit B Minus	1	0	☐ Home B	1	0			
☐ Limit C Plus	1	0	☐ Home C	1	0			
☐ Limit C Minus	1	0						
☐ Home Y	1	11						
☐ Home X	1	12						
☐ Home Z	1	13						

Checking a box indicateds a signal
is active when low. (0VDC)
Signals not selected are greyed out
Pin numbers must be 10,11,12,13, and 15 only!!
Port numbers must be 1 or 2 only!!!

OK | Cancel | Apply

Figure 5.5 - Parallel Port Input Pin assignment

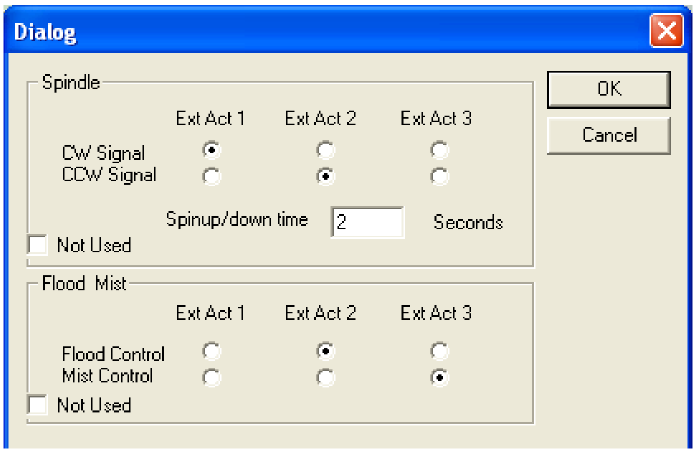
If you are sharing the reference switch with one of the limit switches and have the limit switches for a axis "ORed" together then you will put the same pin number in *Limit Plus*, *Limit Minus* and *Home* for each axis. Mach2 interprets this input differently depending on whether you have just reset the system and when you are performing a Reference operation on the axis concerned.

If you want to a logic one to be detected when an input in "Lo" (which is very common, as with output signals) then you should check the appropriate *Low active* box.

Click the *Apply* button to save the data on this tab.

Finally if you are connecting encoders complete the *Encoder Pin Inputs* tab to define how they are connected. There is no need to indicate Active Low here. If the encoders DROs count the wrong way when you test them you merely need to reverse the Quadrature #1 and Quadrature #2 values.

Click the *Apply* button to save the data on this tab.



Dialog

Spindle

	Ext Act 1	Ext Act 2	Ext Act 3
CW Signal	●	○	○
CCW Signal	○	●	○

Spinup/down time 2 Seconds

☐ Not Used

Flood Mist

	Ext Act 1	Ext Act 2	Ext Act 3
Flood Control	○	●	○
Mist Control	○	○	●

☐ Not Used

OK | Cancel

Figure 5.6 - Output Devices dialog

5.4.3 Assign External Activation Outputs

Display the dialog Configure>Output Devices. This will be like figure 5.6. Use the radio buttons to allocate the spindle and coolant options to the required outputs.

5.4.4 Testing

Your software is now configured sufficiently for you to do some simple tests with the hardware. If it is convenient to connect up the inputs from the reference switches then do so now.

Run Mach2Mill and display the Diagnostics screen. This has a bank of LEDs displaying the logic level of the inputs and outputs. Ensure that the external Emergengy Stop signal is not active (Red *Emergency* LED not flashing) and press the red *Reset* button on the screen. Its LED should stop flashing.

If you have associated any External Activation outputs with coolant or spindle rotation then you can use the relevant buttons on the diagnostic screen to turn the outputs on and off. The machine should also respond or you can monitor the voltages of the signals with a multimeter.

Next operate the reference or the limit switches. You should see the appropriate LEDs glow yellow when their signal is active.

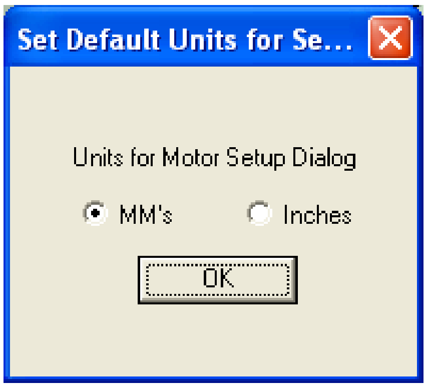
These tests will let you see that your parallel port is correctly addressed and the inputs and outputs are appropriately connected.

If you have two ports and all the test signals are on one then you might consider a temporary switch of your configuration so that one of the reference or limit switches is connected via it so that you can check its correct operation. Don't forget the *Apply* button when doing this sort of testing. If all is well then you should restore the proper configuration.

If you have problems you should sort them out now as this will be much easier that when you start trying to drive the axes. If you do not have a multi-meter then you will have to buy or borrow a logic probe or a D25 adaptor with actual LEDs which let you monitor the state of its pins. In essence you need to discover if the signals in and out of the computer are incorrect (i.e. Mach2 is not doing what you want or expect) or the signals are not getting between the D25 connector and your machine tool (i.e. a wiring or configuration problem with the breakout board or machine). 15 minutes help from a friend can work wonders in this situation even if you only carefully explain to him/her what your problem is and how you have already looked for it! You will be amazed how often this sort of explanation suddenly stops with words like "…… Oh! I see what the problem must be, it's ….."

5.5 Defining the setup units

With the basic functions working it's time to configure the axis drives. The first thing to decide is whether you wish to define their properties in Metric (millimetres) of Inch units. You will be able to run part programs in either units whichever you choose. The maths for configuration will be slightly easier if you choose the same system as your drive train (e.g. ballscrew) was made in. So a screw with 0.2" lead (5 tpi) is easier to configure in inches than in millimetres. Similarly a 2mm lead screw will be easier in millimetres. The multiplication and/or division by 25.4 is not difficult but is just something else to think about.



Units for Motor Setup Dialog

[✓ MM's] Inches

Figure 5.7 - Setup Units dialog

There is, on the other hand, a slight advantage in having the setup units be the units in which you usually work. This is that you can lock the DROs to display in this system whatever the part program is doing (i.e. switching units by G20 and G21).

So the choice is yours. Use Configure>Setup Units to choose MMs or Inches (see figure 5.7). Once you have made it you must not change it without going back over the following steps or total confusion will reign!

5.6 Tuning motors

Well after all that detail it's now time to get things moving - literally! This section describes setting up your axis drives and, if its speed will be controlled by Mach2, the spindle drive.

The overall strategy for each axis is: (a) to calculate how many step pulses must be sent to the drive for each unit (inch or mm) of movement of the tool or table, (b) to establish the maximum speed for the motor and (c) to set the required acceleration/deceleration rate.

We advise you to deal with one axis at a time. You might wish to try running the motor before it is mechanically connected to the machine tool.

So now connect up the power to your axis driver electronics and double check the wiring between the driver electronics and you breakout board/computer. You are about to mix high power and computing so it is better to be safe than smokey.

5.6.1 Calculating the steps per unit

The number of steps Mach2 must send for one unit of movement depends on the mechanical drive (e.g. pitch of ballscrew, gearing between the motor and the screw), the properties of the stepper motor or the encoder on the servo motor and the micro-stepping or electronic gearing in the drive electronics. We look at these in turn.

5.6.1.1 Calculating mechanical drive

You are going to calculate the number of revolutions of the motor shaft (***motor revs per unit***) to move the axis by one unit. This will probably be greater than one for inches and less than one for millimetres but this makes no difference to the calculation which is easiest done on a calculator anyway.

For a **screw and nut** you need the raw pitch of the screw (i.e. thread crest to crest distance) and the number of starts. Inch screws may be specified in threads per inch (tpi). The pitch is 1/tpi (e.g. the pitch of an 8 tpi screw is 1 ÷ 8 = 0.125")

If the screw is multiple start multiply the raw pitch by the number of starts to get the effective pitch. The ***effective screw pitch*** is therefore the distance the axis moves for one **revolution of the screw.**

If the screw is directly driven from the motor then the ***motor shaft pitch*** (i.e. the distance moved by one **revolution of the motor**) is the same as the effective screw pitch. If the motor has a gear or belt drive to the screw with N_m teeth on the motor gear and N_s teeth on the screw gear then:

motor shaft pitch = *effective screw pitch* x $N_m \div N_s$

For example suppose our 8 tpi screw is connected to the motor with a toothed belt with a 48 tooth pulley on the screw and an 16 tooth pulley on the motor then the motor shaft pitch would be 0.125 x 16 ÷ 48 = 0.04166666" (**Hint**: keep all the figures on your calculator for the next stage of calculation)

As a metric example suppose our two star screw has 5 millimetres between thread crests and it is connected to the motor with 24 tooth pulley on the motor shaft and a 48 tooth pulley on the screw. The *effective screw pitch* is 5 x 2 = 10mm. So the *motor shaft pitch* is 10 x 24÷48 = 5mm.

Finally for the screw drive:

motor revs per unit = 1 ÷ *motor shaft pitch*

In the 8tpi example, we get 24 revs per unit (actually a cheap calculator may give something like 24.000038 or so due to internal rounding errors) and for the metric screw 0.2 revs per unit.

For a **toothed belt or chain** drive the calculation is similar.

Find the pitch of the belt teeth or chain links. Belts are available in metric and imperial pitches with 5 or 8 millimetres common metric pitches and 0.375" ($^3/_8$") common for inch belts and for chain. We will call this ***belt pitch***.

If the number of teeth on the sprocket/pulley on the motor is N_m then:

motor revs per unit = 1 ÷ (*belt pitch* x N_m)

So, for example with a $^3/_8$" chain and a 13 tooth motor sprocket the *motor revs per unit* = 1 ÷ (0.375 x 13) = 0.2051282 In passing we observe that this is quite "high geared" and the motor might need an additional reduction gearbox to meet the torque requirements. In this case you multiple the motor revs per unit by the reduction ratio of the gearbox. For example a 10:1 box would give 2.051282 revs per inch.

For **rotary axes** (e.g. rotary tables or dividing heads) the unit is the degree. You need to calculate based on the worm ratio. This is often 90:1. So with a direct motor drive to the worm one rev gives 4 degrees so *Motor revs per unit* would be 0.25. A reduction of 2:1 from motor to worm would give 0.5 revs per unit.

5.6.1.2 Motor steps per revolution

The basic resolution of all modern stepper motors is 200 steps per revolution (i.e. 1.8^o per step).

The basic resolution of a servo motor depends on the encoder on its shaft. The encoder resolution is usually quoted in lines per revolution (i.e. the number of slits in its disc) Because the output is actually two quadrature signals the effective resolution will be **four** time this value. You would expect a basic resolution of around 300 lines per revolution.

5.6.1.3 Mach2 steps per motor revolution

We very strongly recommend that you use micro-stepping drive electronics for stepper motors. If you do not do this and use a full- or half-step drive then you will need much larger motors and will suffer from resonances that limit performance at some speeds.

Some micro-stepping drives have a fixed number of micro-steps (typically 10) others can be configured. In this case you will find 10 to be a good compromise value to choose. This means that Mach2 will need to send 2000 pulses per revolution for a stepper axis drive.

Some servo drives require one pulse per quadrature pulse from the motor encoder (thus giving around 1200 steps per rev for a 300 line encoder. Others include electronic gearing where you can multiply the input steps by an integer value and, sometimes, the divide the result by another integer value. The multiplication of input steps can be very useful with Mach2 as the speed of small servo motors with a high resolution encoder can be limited by the maximum pulse rate which Mach2 can generate.

5.6.1.4 Mach2 steps per unit

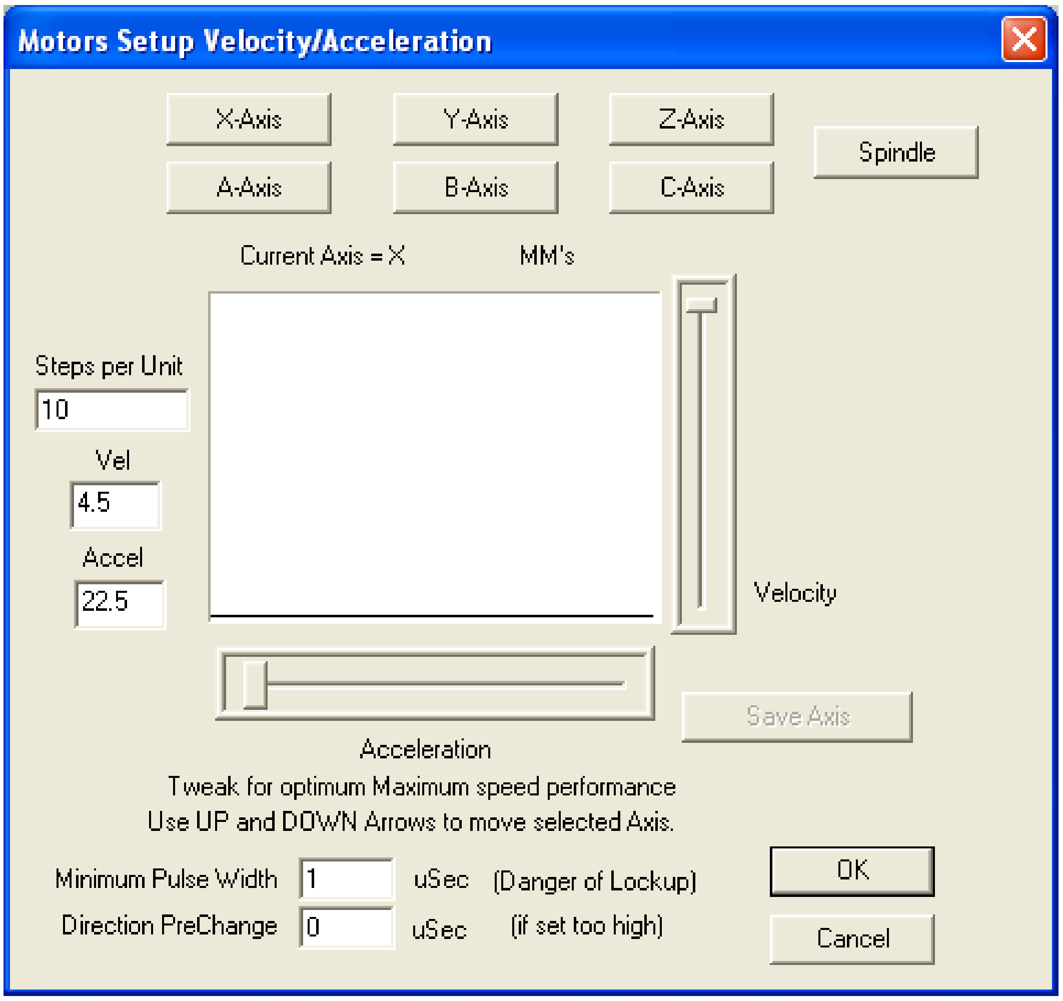
So now we can finally calculate:

Mach2 steps per unit = *Mach2 steps per rev* x *Motor revs per unit*

Figure 5.8 shows the dialog for Configure>Motor Tuning. Click a button to select the axis which you are configuring and enter the calculated value of *Mach2 steps per unit* in the box to the left hand side of the graph. This value does not have to be an integer so you can achieve as much accuracy as you wish. To avoid forgetting later click *Save Axis.*

5.6.2 Setting the maximum motor speed

Still using the Motor Tuning dialog, if you move the Velocity slider you will see a graph of velocity against time for a short imaginary move. The axis accelerates, maybe runs at full speed and then decelerates. Set the velocity to maximum for now. Use the Acceleration slider to alter the rate of acceleration/deceleration (they are always the same as each other)


Motors Setup Velocity/Acceleration

Current Axis = X MM's

Steps per Unit: 10

Vel: 4.5

Accel: 22.5

Tweak for optimum Maximum speed performance
Use UP and DOWN Arrows to move selected Axis.

Minimum Pulse Width: 1 uSec (Danger of Lockup)
Direction PreChange: 0 uSec (if set too high)

Figure 5.8 - Motor tuning dialog

As you use the sliders the values in the *Vel* and *Accel* boxes are updated. *Vel* is in units per second. *Accel* is in units per second^2 The maximum velocity you can display will be limited by the maximum pulse rate of Mach2. Suppose you have configured this to 35,000 Hz and 400 steps per unit then the maximum possible *Vel* is 87.5.

This maximum is, however, not necessarily safe for your motor, drive mechanism or machine; it is just Mach2 running "flat out". You can make the necessary calculations or do some practical trial. Let's try it out first.

5.6.2.1 Practical trials of motor speed

You saved the axis after setting the Steps per unit. OK the dialog and make sure that everything is powered up. Click the Reset button so its LED glows continuously.

Go back to Configure>Motor Tuning and select your axis. Use the Velocity slider to have the graph about 20% of maximum velocity. Press the cursor Up key on your keyboard. The axis should move in the Plus direction. If it runs away the choose a lower velocity. If it crawls then choose a higher velocity. The cursor Down key will make it run the other way (i.e. the Minus direction).

If the direction is wrong then, Save the axis and **either** (a) change the Low Active setting in Configure>Ports and Pins Printer, Port Output Pins tab (and *Apply* it) or (b) check the appropriate box in Configure>Motor Reversals for the axis that you are using or for a stepper motor (c) switch off and reverse **one** pair of connections to the motor from the drive electronics.

If a stepper motor hums or screams then you have wired it wrong or are trying to drive it much too fast. The labelling of stepper wires (especially 8 wire motors) is sometimes very confusing. You need to refer to the motor and driver documentation.

If a servo motor runs away at full speed or flicks and indicates a fault on its driver then its armature (or encoder) connections need reversing (see the servo electronics documentation for more details). Remember the advice to buy current and properly supported products - buy right, buy once!

Most drives will work well with a 1 microsecond minimum pulse width and need no delay before the *Dir* pin is changed (*Direction PreChange*). If you have problems with the test moves (e.g. motor seems too noisy) first check that your step pulses are not inverted (by *Low active* being set incorrectly for Step on the Printer port Output Pins tab of Ports and Pins) then you might try increasing the pulse width to, say, 5 microseconds. The Step and Direction interface is very simple but, because it "sort of works" when configured badly, can be difficult to fault-find without being very systematic and/or looking at the pulses with an oscilloscope.

5.6.2.2 Motor speed calculations

There are many things which define the maximum speed of an axis:

- ♦ Maximum allowed speed of motor (perhaps 4000 rpm for servo or 1000 rpm for stepper)
- ♦ Maximum allowed speed of ballscrew (depends on length, diameter, how its ends are supported
- ♦ Maximum speed of belt drive or reduction gearbox
- ♦ Maximum speed which drive electronics will support without signalling a fault
- ♦ Maximum speed to maintain lubrication of machine slides

The first two in this list are most likely to affect you. You will need to refer to the manufacturers' specifications, calculate the permitted speeds of screw and motor and relate these to units per second of axis movement. Set this maximum value in the *Vel* box of Motor Tuning for the axis involved.

The free Master5/Mach1/Mach2 online forum is a useful place to get advice from other Mach2 users, world-wide, on this sort of topic.

5.6.3 Deciding on acceleration

5.6.3.1 Inertia and forces

No motor is able to change the speed of a mechanism instantly. A torque is needed to give angular momentum to the rotating parts (including the motor itself) and torque converted to force by the mechanism (screw and nut etc.) has to accelerate the machine parts and the tool or workpiece. Some of the force also goes to overcome friction and, of course, to make the tool cut.

Mach2 will accelerate (and decelerate) the motor at a given rate (i.e. a straight line speed time curve) If the motor can provide more torque than is needed for the cutting, friction and inertia forces to be provided at the given acceleration rate then all is well. If the torque is insufficient then it will either stall (if a stepper) or the servo position error will increase. If the servo error gets too great then the drive will probably signal a fault condition but even if it does not then the accuracy of the cutting will suffer. This will be explained in more detail shortly.

5.6.3.2 Testing different acceleration values

Try starting and stopping your machine with different settings of the Acceleration slider in the Motor Tuning dialog. At low accelerations (a gentle slope on the graph) you will be able to hear the speed ramping up and down.

5.6.3.3 Why you want to avoid a big servo error

Most moves made in a part program are coordinated with two, or more, axes moving together. Thus in a move from X=0, Y=0 to X=2, Y=1, Mach2 will move the X axis at twice the speed of the Y axis. It not only coordinates the movements at constant speed but ensures that the speed required relationship applies during acceleration and deceleration but accelerating all motions at a speed determined by the "slowest" axis.

If you specify too high an acceleration for a given axis then Mach2 will assume it can use this value but as, in practice, the axis lags behind what is commanded (i.e. the servo error is big) then the path cut in the work will be inaccurate.

5.6.3.4 Choosing an acceleration value

It is quite possible, knowing all the masses of parts, moments of inertia of the motor and screws, friction forces and the torque available from the motor to calculate what acceleration can be achieved with a given error. Ballscrew and linear slide manufacturers' catalogues often include sample calculations.

Unless you want the ultimate in performance from your machine, we recommend setting the value so that test starts and stops sound "comfortable". Sorry it's not very scientific but it seems to give good results!

5.6.4 Saving and testing axis

Finally don't forget to click *Save Axis* to save the acceleration rate before you move on.

You should now check your calculations by using the MDI to make a defined G0 move. For a rough check you can use a steel rule. A more accurate test can be made with a Dial Test Indicator (DTI)/Clock and a slip gage block.

Suppose you are testing the X axis and have a 4" gage block.

Use the MDI screen to select inch units and absolute coordinates. (G20 G90) Set up a block on the table and Jog the axis so the with the DTI probe touches it. Ensure you finish by a move in the minus X direction.

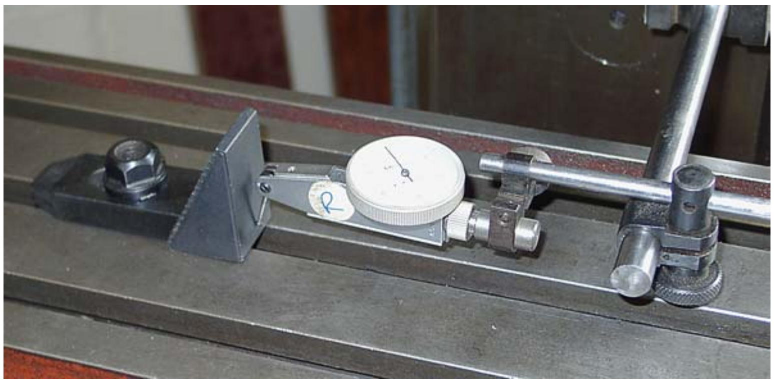


Figure 5.9 - Establishing a zero position

Rotate the bezel to zero the reading. This is illustrated in figure 5.9.

Now use the Mach2 MDI screen and click the G92X0 button to set an offset and hence zero the X axis DRO.

Move controlled point to X=4.5 by G0 X4.5. The gap should be about half an inch. If it is not then there is something badly wrong with your calculations of the Steps per Unit value. Check and correct this.

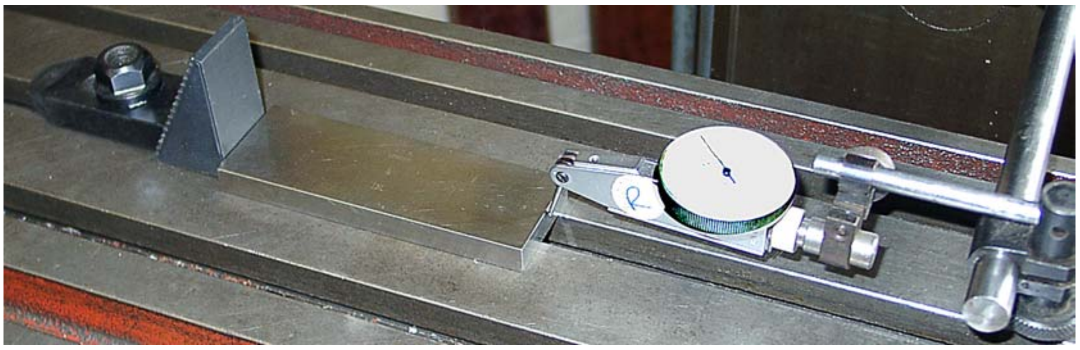


Figure 5.10 - Gage block in position

Insert the gage block and move to X-4.0 by G0 X4. This move is in the X minus direction as was the jog so the effects of backlash in the mechanism will be eliminated. The reading on the DTI will give your positioning error. It should only be a thou or so. Figure 5.10 shows the gage in position.

Remove the gage and G0 X0 to check the zero value. Repeat the 4" test to get an set of, perhaps, 20 values and see how reproducible the positioning is. If you get big variations then there is something wrong mechanically. If you get consistent errors then you can fine tune the Steps per Unit value to achieve maximum accuracy.

Next you should check that the axis does not lose steps in repeated moves at speed. Remove the gage block. Use MDI to G0 X0 and check the zero on the DTI.

Use the editor to input the following program:

```
F1000 (i.e. faster than possible but Mach2 will limit speed)
G20 G90 (Inch and Absolute)
M98 P1234 Q50 (run subroutine 50 times)
M30 (stop)
O1234
G1 X4
G1 X0 (do a feed rate move and move back)
M99 (return)
```

Click *Cycle Start* to run it. Check that the motion sounds smooth.

When it finishes the DTI should of course read zero. If you have problems then you will need to fine tune the maximum velocity of acceleration of the axis.

5.6.5 Repeat configuration of other axes

With the confidence you will have gained with the first axis you should be able to quickly repeat the process for the other axes.

5.6.6 Spindle motor setup

If the speed of your spindle motor is fixed or controlled by hand then you can ignore this section. If the motor is switched on and off, in either direct, by Mach2 then this will have been setup with the activation outputs.

If Mach2 is to control the spindle speed either by a servo drive that accepts Step and Direction pulses or by a Pulse Width Modulated (PWM) motor controller then this section tells you how to configure your system.

5.6.6.1 Motor speed, spindle speed and pulleys

The Step and Direction and PWM both allow you to control the speed of the motor. When you are machining what you and the part program (the S word) are concerned with is the speed of the spindle. The motor and spindle speed are, of course, related by the pulleys or gears connecting them. We will use the term "pulley" to cover both sorts of drive in this manual.

Mach2 cannot know without being told by you, the machine operator, what pulley ratio is selected at any given time so you are responsible for this. Actually the information is given in two steps. When the system is configured (i.e. what you are doing now) you define up to four available pulley combinations. These are set by the physical sizes of the pulleys or ratios in the

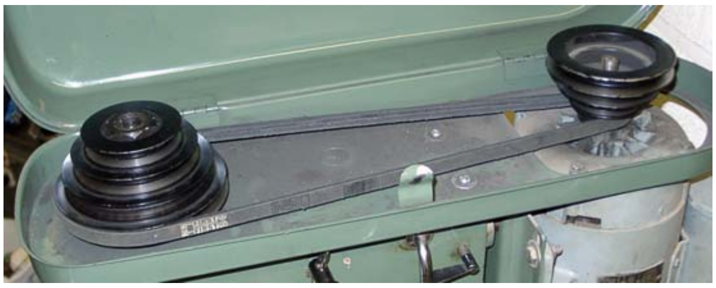
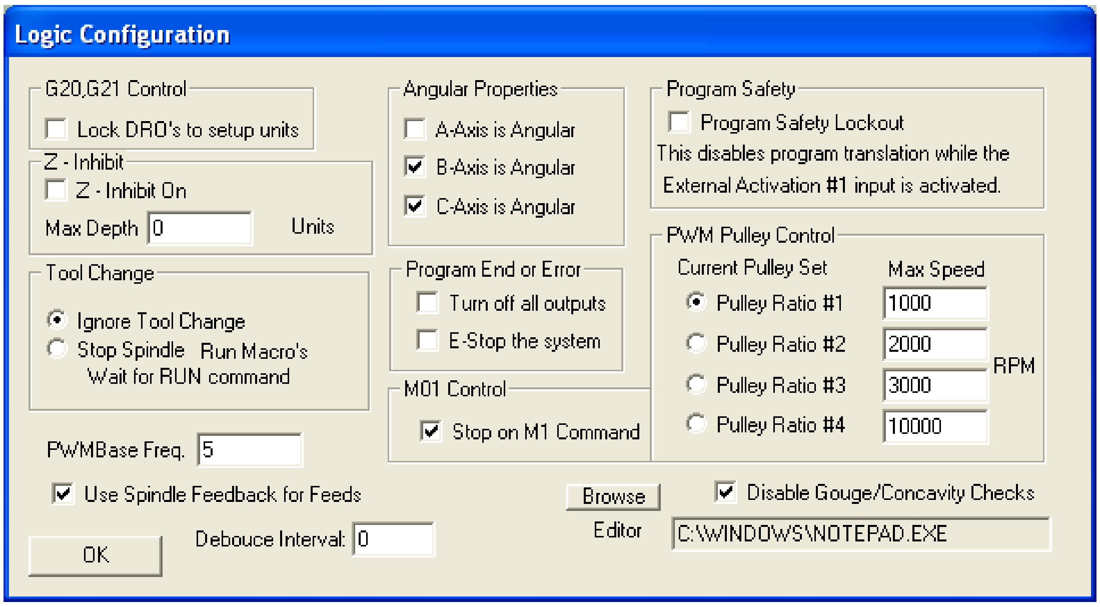


Figure 5.11 - Pulley spindle drive

geared head. Then when a part program is being run the operator defines which pulley (1 to 4) is in use.

The machine's pulley ratios are set on the Configure>Logic dialog (figure 5.12). On the Logic Configuration dialog the maximum speed of the four pulley sets is defined together with the default one to be selected. The maximum speed is the speed at which the **spindle** will rotate when the motor is at full speed. Full speed is achieved by 100% pulse width in PWM and the set *Vel* value on Motor Tuning "spindle Axis" for Step and Direction.



Logic Configuration

G20,G21 Control
☐ Lock DRO's to setup units

Z - Inhibit
☐ Z - Inhibit On
Max Depth: 0 Units

Tool Change
● Ignore Tool Change
○ Stop Spindle Run Macro's Wait for RUN command

PWMBase Freq.: 5

☑ Use Spindle Feedback for Feeds

Debouce Interval: 0

Angular Properties
☐ A-Axis is Angular
☑ B-Axis is Angular
☑ C-Axis is Angular

Program End or Error
☐ Turn off all outputs
☐ E-Stop the system

M01 Control
☑ Stop on M1 Command

Program Safety
☐ Program Safety Lockout
This disables program translation while the External Activation #1 input is activated.

PWM Pulley Control

Current Pulley Set	Max Speed
● Pulley Ratio #1	1000
○ Pulley Ratio #2	2000
○ Pulley Ratio #3	3000
○ Pulley Ratio #4	10000

RPM

☑ Disable Gouge/Concavity Checks

Editor: C:\WINDOWS\NOTEPAD.EXE

Figure 5.12 - Logic Configuration dialog

As an example, suppose the position we will call "Pulley 1" is a step down of 5:1 from motor to spindle and the maximum speed of the motor is 3600 rpm. Pulley 1 maximum speed on Logic Configuration will be set to 720 rpm (3600 ÷ 5). Pulley 4 might be a step up of 4:1. With the same motor speed its maximum speed would be set to 14,400 rpm (3600 x 4). The other pulleys would be intermediate ratios. The pulleys do not need to be defined in increasing speeds but the numbers should relate in some logical way to the controls on the machine tool.

We will return to the other controls on Configure>Logic later.

Mach2 uses the pulley ratio information as follows:

- When the part program executes an S word or a value is entered into the set speed DRO then the values is compared with the maximum speed for the currently selected pulley. If the requested speed if greater than the maximum then an error occurs.
- Otherwise the percentage of the maximum for the pulley that has bee requested and this is used to set the PWM width or the Step pulses are generated to produce that percentage of the maximum motor speed as set in Motor Tuning for the "Spindle Axis".

As an example suppose the max spindle speed for Pulley #1 is 1000 rpm. S1100 would be an error. S600 would give a pulse width of 60%. If the maximum Step and Direction speed is 3600 rpm then the motor would be "stepped" at 2160 rpm (3600 x 0.6).

5.6.6.2 Pulse width modulated spindle controller

To configure the spindle motor for PWM control, check the Spindle Axis Enabled and PWM Control boxes on the Port and Pins, Printer Port and Axis Selection Page tab (figure 5.1). Don't forget to *Apply* the changes. Define an output pin on the Output Signals Selection Page tab (figure 5.4) for the Spindle Step. This pin must be connected to your PWM motor control electronics. You do not need one for Spindle Direction so set this pin to 0. *Apply* the changes.

Now move to the Configure Logic dialog and locate the *PWMBase Freq* box. The value in here is the frequency of the squarewave whose pulse width is modulated. This is the signal which appears on the Spindle Step pin. The higher the frequency you choose here the faster your controller will be able to respond to speed changes but the lower the "resolution" of chosen speeds. The number of different speeds is the *Engine pulse frequency* ÷ *PWMBase freq*. Thus for example if you are running at 35,000 Hz and set the PWMBase to 50 Hz there are 700 discrete speeds available. This is almost certainly sufficient on any real system as a motor with maximum speed of 3600 rpm could, theoretically, be controlled in steps of less than 6 rpm.

5.6.6.3 Step and Direction spindle controller

To configure the spindle motor for Step and Direction control, check the Spindle Axis Enabled boxes on the Port and Pins, Printer Port and Axis Selection Page tab (figure 5.1). Leave PWM Control unchecked. Don't forget to *Apply* the changes. Define output pins on the Output Signals Selection Page tab (figure 5.4) for the Spindle Step and Spindle Direction These pin must be connected to your motor drive electronics as for axis drives. *Apply* the changes.

Now move to Configure>Motor Tuning for the "Spindle Axis". The units for this will be one revolution. So the Steps per Unit are the number of pulses for one rev (e.g. 2000 for a 10 times micro-stepping drive or 4 x the line count of a servomotor encoder or the equivalent with electronic gearing).

The *Vel* box should be set to the number of revs per second at full speed. So a 3600 rpm motor would need to be set to 60. This is not possible with a high line count encoder on account of the maximum pulse rate from Mach2. (e.g. a 100 line encoder allows 87.5 revs per second on a 35,000 Hz system). The spindle will generally require a powerful motor whose drive electronics is likely to include electronic gearing which overcomes this constraint.

The Acceleration box can be set by experiment to give a smooth start and stop to the spindle.

5.6.6.4 Testing the spindle drive

If you have a tachometer or stroboscope then you can measure the spindle speed of your machine. If not you will have to judge it by eye and using your experience.

On Mach2 Corrections screen choose a pulley that will allow 900 rpm. Set the belt or gearbox on the machine to the corresponding posituion. On the Program Run screen set the spindle speed required to 900 rpm and start it rotating. Measure or estimate the speed. If it is wrong you will have to revisit your calculations and setup.

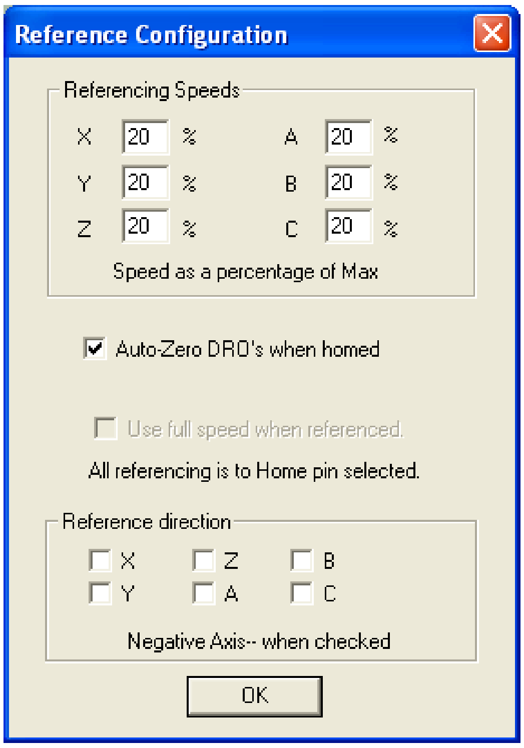
You might also check the speeds on all the pulleys in the same way but with suitable set speeds.

5.7 Other configuration

5.7.1 *Configure referencing*

5.7.1.1 Referencing speeds and direction

The Configure>Referencing dialog allows you to define what happens when a



Referencing Speeds: X 20 %, Y 20 %, Z 20 %, A 20 %, B 20 %, C 20 %

[✓] Auto-Zero DRO's when homed

[ ] Use full speed when referenced.

All referencing is to Home pin selected.

Reference direction: [ ] X [ ] Y [ ] Z [ ] A [ ] B [ ] C

Figure 5.13 - Referencing

reference operation (G28, G28.1 or a screen button) is performed. Figure 5.13 shows the dialog. The referencing speed is used to avoid crashing into the stop of an axis at full speed when looking for the reference switch. When you are referencing Mach2 has no idea of the position of an axis. The direction it moves in depends on the Reference Direction check boxes. If the relevant box it checked then the axis will move in the minus direction until the Home input becomes active. If the Home input is already active then it will move in the plus direction. Similarly if the box is unchecked then the axis moves in the plus direction until the input is active and the minus direction if it is already active.

If the *Auto Zero DROs* when homed checkbox then the axis DROs will be set to the Reference Switch location values defined on the Corrections screen (rather than actual Zero)

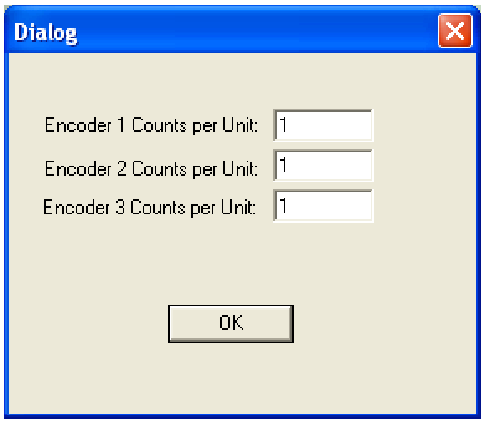
5.7.1.2 Position of reference switches

As described above the reference switches of one or more axes do not need to be at machine zero. This can be useful to minimise the referencing time on a long slow axis (like the Z axis of a floor mill). The absolute coordinates of the reference switches are defined on the Corrections screen.

It is, of course, necessary to have separate limit switches if the limit switch is not at the end of an axis.

5.7.2 Configure Encoders

If you have defined inputs for encoders (other than those on servo motors) then you must specify the number of pulses that will be input per unit (as defined in Setup Units) of movement. This is the number of "lines" on the encoder scale per unit (not the resolution implied by the quadrature multiplication by four). Thus for example a glass scale ruled at 20 micron spacing would be defined in millimetre units as 50 counts and in inch units as 1270 counts. The count value can be non-integral to allow for odd scales or unusual mechanisms (e.g. a rotary encoder operated by a pulley and steel tape) See figure 5.14.



Encoder 1 Counts per Unit: 1
Encoder 2 Counts per Unit: 1
Encoder 3 Counts per Unit: 1

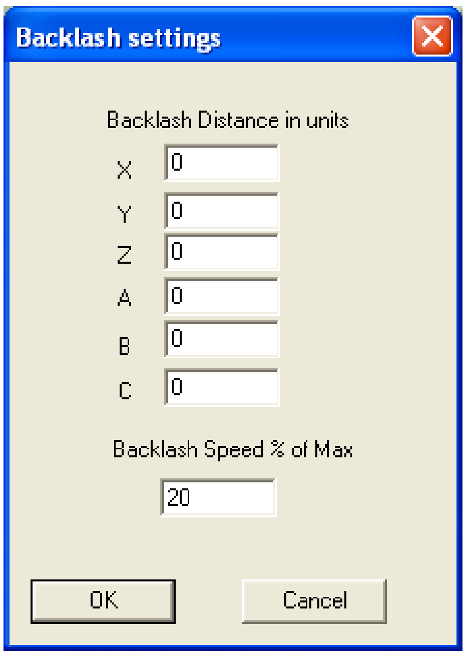
Figure 5.14 - Encoder configuration

Beware with high resolution encoders or encoders on high speed axes. The pulse rate must not, in any circumstance, exceed half the Mach2 pulse frequency. Thus at 35,000 Hz the maximum input frequency is 17.5 kHz which on 20 micron scales corresponds to 0.7 metres per second on the axis.

5.7.3 Configure Backlash

Mach2 will attempt to compensate for backlash in axis drive mechanisms by attempting to approach each required coordinate from the same direction. While this is useful in applications like drilling or boring it cannot overcome problems with the machine in continuous cutting.

The Configure>Backlas dialog allows you to give an estimate of the distance which the axis must back up by to ensure the backlash is taken up when the final "forward" movement



Backlash Distance in units
X 0
Y 0
Z 0
A 0
B 0
C 0
Backlash Speed % of Max
20

Figure 5.15 - Backlash configuration

is made. The speed at which this movement is to be made is also specified. See figure 5.15

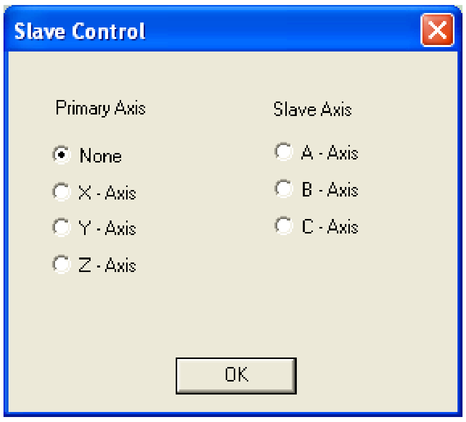
Backlash compensation is a "last resort" when the mechanical design of your machine cannot be improved!

5.7.4 Configure Slaving

Large machines such as gantry routers or mills often need two drives, one on each side of the gantry itself. If these become out of step then the gantry will "rack" and its cross axis not be perpendicular to the long axis.

You can configure Mach2 so one drive (say the X axis) is the main drive and can slave another to it (perhaps the C axis configured as linear rather than rotary).

During normal use the same number of step pulses will be sent to the master and slave axes with the speed and acceleration being determined by the "slower" of the two.


Slave Control

Primary Axis: [✓ None] X - Axis, Y - Axis, Z - Axis

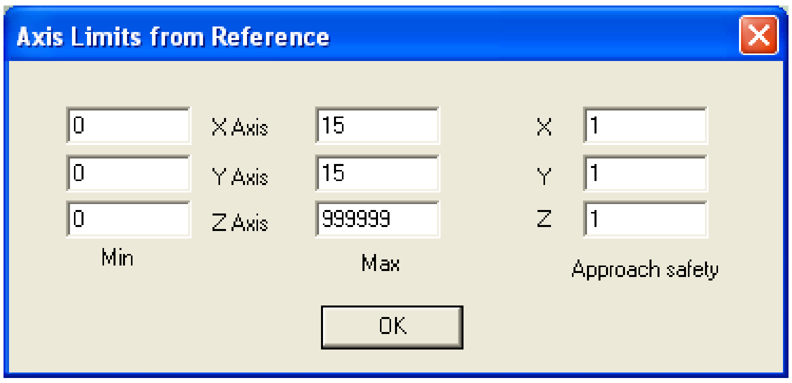
Slave Axis: A - Axis, B - Axis, C - Axis

Figure 5.16 - Slaving configuration

When a reference operation is requested they will move together until the reference switch of one is detected. This drive will position just off the switch in the usual way but the other axis will continue until its switch is detected when it will be positioned off it. Thus the pair of axes will be "squared up" to the reference switch positions and any racking which has occurred be eliminated.

5.7.5 Configure Soft Limits

As discussed above most implementations of limit switches involve some compromises and hitting them accidentally will require intervention by the operator and may require the system to be reset and referenced. Soft limits provide a protection against this sort of inconvenient accident.


Axis Limits from Reference

Min		Max		Approach safety
0	X Axis	15	X	1
0	Y Axis	15	Y	1
0	Z Axis	999999	Z	1

Figure 5.17 - Soft Limits configuration

The software will refuse to allow the axes to move outside the declared range of the soft limits of the X, Y and Z axes. These can be set in the range -999999 to + 999999 units for each axis. When jogging motion gets near to the limit then its speed will be reduced when inside an *Approach Safety* zone. These values are defined on the Configure>Soft Limits dialog. See figure 5.17.

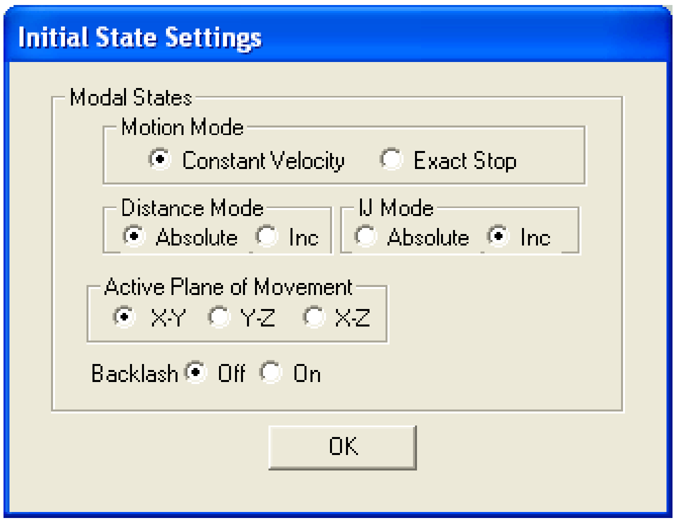
If a part program attempts to move outside a soft limit then it will stop movement (i.e. clip) until the program moves the controlled point back within the limits. In other words a program violation of a soft limit does not raise an error.

5.7.6 Configure Initial State

Configure>State opens a dialog which allows you to define the modes when Mach2 is loaded. It is shown in figure 5.18.

Motion mode: *Constant velocity* sets G64, *Exact Stop* sets G61

Distance mode: *Absolute* sets G 90, *Inc* sets G91


Initial State Settings

Modal States

Motion Mode: [✓ Constant Velocity] Exact Stop

Distance Mode: [✓ Absolute] Inc

IJ Mode: Absolute [✓ Inc]

Active Plane of Movement: [✓ X-Y] Y-Z X-Z

Backlash [✓ Off] On

OK

Figure 5.18 - Initial State configuration

Active plane: X-Y sets G17, Y-Z sets G19, X-Z sets G18

In addition you can set the interpretation to be placed on I & J in arc moves. This is provided for compatibility with different CAM post-processor and to emulate other machine controllers. In *Inc IJ* mode I and J (the center point) are interpreted as relative to the starting point of a center format arc. This is compatible with NIST EMC. In *Absolute IJ* mode I and J are the coordinates of the center in the current coordinate system (i.e. after application of fixture, tool and G92 offsets). If circles away fail to be cut properly (especially obvious if they are far from the origin) then the IJ mode is not compatible with you part program.

Backlash (compensation): can be set on of off here (you might have looked for it on Configure Logic!)

5.7.7 Configure other Logic items

Several functions on the Configure Logic dialog (see figure 5.12) have been discussed already. The remainder are covered here.

G20/G21 Control: If Lock DROs to set up units is checked then uven though G20 and G21 will alter the way X, Y, Z etc. words are interpreted (inch or millimetre) the DROs will always display in the Setup Unit system.

Z-Inhibit: Do not use this function in the current version of Mach2

Tool change: An M6 tool change request can be ignored or used to call the M6 macros (q.v.)

Angular properties: An axis defined as angular is measured in degrees (that is to say G20/G21 do not alter the interpretation of A, B, C words)

Program end or error: defines action(s) at end of part program

Stop on M01 Command: This is the "Optional Stop" switch

Use spindle feedback for feeds: This should only be checked if an index pulse from the spindle is connected to the Index input pin. When checked, if Feed per Rev is selected (e.g. on the Program Run screen) then the index will be used to define a revolution of the spindle. If not checked then the speed called for by the S word (or Set Speed DRO) will be used to determine the feed. **Note:** the S word can be used even if the spindle motor is manually controlled.

Debounce interval: Is the number of Mach 2 pulses that a switch must be stable for its signal to be considered valid. So for a system running at 35,000 Hz , 100 would give about a 3 millisecond debounce (100 ÷ 35000 = .0029 secs).

Program safety: When checked enables Activation Input #1 as a safety cover interlock.

Disable Gouge/Concavity checks: Art to explain to me (JAP)!!

Editor: The filename of the executable of the editor to be called by the G-code edit button. The Browse button allows a suitable file (e.g. C:\windows\notepad.exe) to be found.

6. Loading, running and editing a Part Program

Here we will look at the operator's use of the Program, MDI and Positioning screens. This will probably be written last to have as much stability in screens as possible before it is done.

7. Co-ordinate systems, tool table and fixtures

This chapter explains how Mach2 works out where exactly you mean when you ask the tool to move to a given position. It introduces the idea of a co-ordinate system, defines the Machine Co-ordinate System and shows how you can specify the lengths of each Tool, the position of a workpiece in a Fixture and add your own variable Offsets.
You may find it heavy going on the first read. We suggest that you try out the techniques using your own machine. It is not easy to do this just "dry" running Mach2 as you need to see where an actual tool is.

7.1 Machine co-ordinate system

You have seen that most Mach2 screens have DROs labelled "X Axis", "Y Axis" etc. If you are going to make parts accurately and minimise the chance of your tool crashing into anything you need to understand exactly what these values mean at all times when you are setting up a job or running a part program.

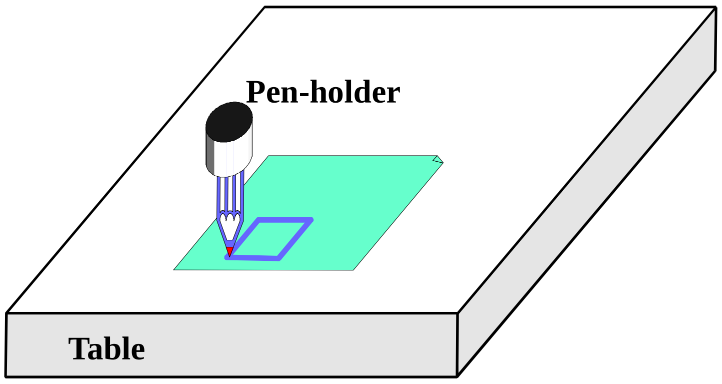


Figure 7.1 - Basic Drawing Machine

This is easiest to explain looking at a machine. We have chosen an imaginary machine that make it easier to visualise how the co-ordinate system works. Figure 7.1 shows what it is like.

It is a machine for producing drawings with a ballpoint or felt tipped pen on paper or cardboard. It consists of a fixed table and a cylindrical pen-holder which can move left and right (X direction), front and back (Y direction) and up and down (Z-direction). The figure shows a square which has just been drawn on the paper.

Figure 7.2 shows the Machine Co-ordinate System which measures (lets say in inches) from the surface of the table at its bottom left hand corner. As you will see the bottom left corner of the paper is at X=2, Y=1 and Z=0 (neglecting paper thickness). The point of the pen is at X=3, Y=2 and it looks as though Z=1.3.

If the point of the pen was at the corner of the table then, on this machine, it would be in its **Home** or **Referenced** position. This is not very practical but we will come back to where Home might actually be put on a real machine.

The point of the pen, like the end of a cutting tool, is where things happen and is called the **Controlled Point**. The Axis DROs in Mach2 **always** display the co-ordinates of the Controlled Point relative to some co-ordinate system. The reason you are having to read this

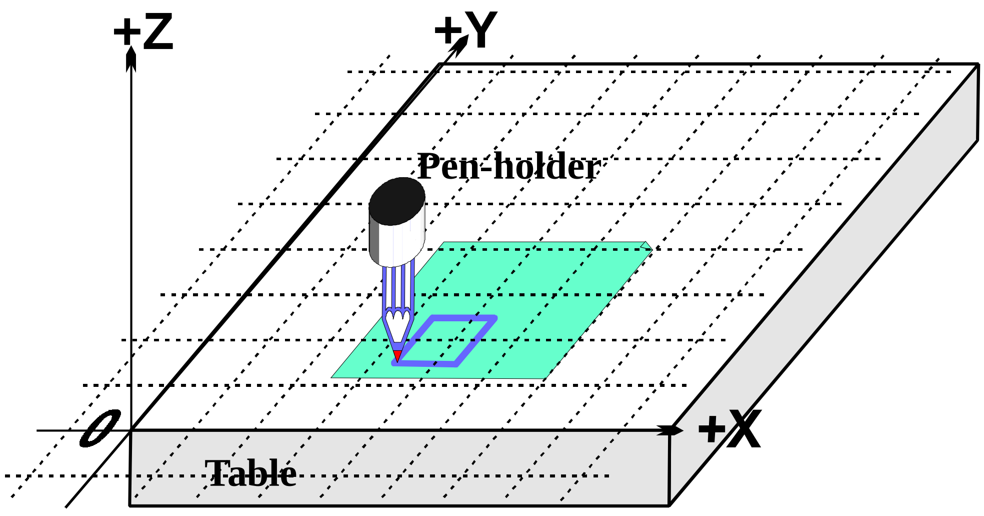


Figure 7.2 Machine co-ordinate system

chapter is that it is not always convenient to have the zeros of the measuring co-ordinate system at a fixed place of the machine (like the corner of the table in our example).

A simple example will show why this is so.

The following Part Program looks, at first sight, suitable for drawing the 1" square in Figure 7.1:

```
N10 G20 F10 G90 (set up imperial units, a slow feed rate etc.)
N20 G0 Z2.0 (lift pen)
N30 G0 X0.8 Y0.3 (rapid to bottom left of square)
N40 G1 Z0.0 (pen down)
N50 Y1.3 (we can leave out the G1 as we have just done one)
N60 X1.8
N70 Y0.3 (going clockwise round shape)
N80 X0.8
N90 G0 X0.0 Y0.0 Z2.0 (move pen out of the way and lift it)
N100 M30 (end program)
```

Even if you cannot yet follow all the code it is easy to see what is happening. For example on line N30 the machine is told to move the Controlled Point to X=0.8, Y=0.3. By line N60 the Controlled Point will be at X=1.8, Y=1.3 and so the DROs will read:

X Axis 1.8000 Y Axis 1.3000 Z Axis 0.0000

The problem, of course, is that the square has not been drawn on the paper like in figure 7.1 but on the table near the corner. The Part Program writer has measured from the corner of the paper but the machine is measuring from its Home/Reference position.

7.2 Fixture offsets

Mach2, like all machine controllers, allows you to move the origin of the co-ordinate system or, in other words where it measures from (i.e. where on the machine is to considered to be zero for moves of X, Y Z etc.)

This is called **offsetting** the co-ordinate system.

Figure 7.3 shows what would happen if we could offset the Current Co-ordinate system to the corner of the paper. **Remember** the G-code always moves the Controlled Point to the numbers given in the Current Co-ordinate system.

As there will usually be some way fixing sheets of paper, one by one, in the position shown this offset is called a **Fixture offset** and the 0, 0, 0 point is the **Fixture origin.**

This offsetting is so useful that there are several ways of doing it using Mach2 but they are all organised using the Tables screen (see Appendix 1 for screenshot)

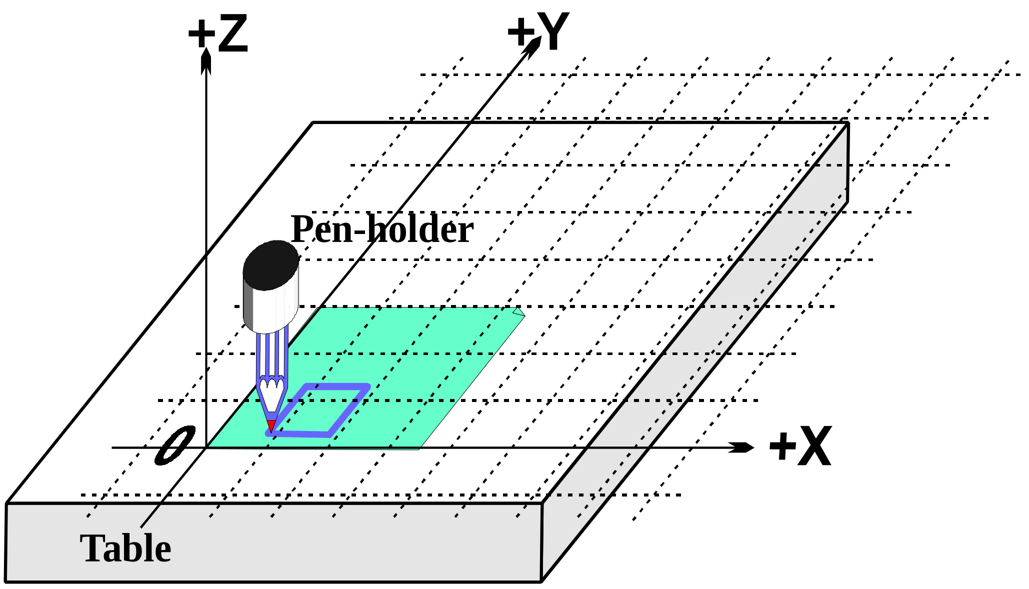


Figure 7.3 - Co-ordinate system origin offset to corner of paper

7.2.1 Setting Fixture origin to a given point

The most obvious way is done in two steps:

1. Display the Tables screen. Move the Controlled Point (pen) to where you want the new origin to be. This can be done by jogging or, if you can calculate how far it is from the current position you can use G0s with manual data input
2. Click the Touch button next to each of the axes in the Fixture part of the screen. On the first Touch you will see that "Fixture 1" is selected, the LED lights to show that the fixture offset is ON, the existing co-ordinate of the Touched axis is put into the Offset DRO and the axis DRO reads zero. Subsequent Touches on other axes copy the Current Co-ordinate to the offset and zero that axis DRO.

If you wonder what has happened the following may help. When the fixture is switched on then its offset values are added the numbers in the axis DROs (i.e. the current co-ordinates of the controlled point) to give the absolute machine co-ordinates of the controlled point. Mach2 will display the absolute co-ordinates of the controlled point if you click the *Mach* (for machine) under the *DRO Display Type* heading button. The LED flashes to warn you that the co-ordinates are absolute ones. You can switch off the Fixture offsets and then switch them on again by typing "1" into the fixture number DRO.

There is another way of setting the offsets which can be used if you know the position of where you want the new origin to be.

The corner of the paper is, by eye, about 2.6" right and 1.4" above the Home/Reference point at the corner of the table. Let's suppose that these figures are accurate enough to be used.

1. Type 1 into the Fixture number DRO. This will turn it on. Type 2.6 and 1.4 into the X and Y Offset DROs. The Axis DROs will change (by having the offsets subtracted from them). Remember you have not moved the absolute position of the Controlled point so its co-ordinates must change when you move the origin.
2. If you want to you could check all is well by using the MDI line to G00 X0 Y0 Z0. The pen would be touching the table at the corner of the paper.

We have described using Fixture number 1. You can use any numbers from 1 to 255. Only one is in use at any time and this can be chosen on the Tables screen or by using G-codes (G54 to G59 P253) in your Part Program.

So, to recap the example, by offsetting the Current Co-ordinate system by a Fixture offset we can draw the square at the right place on the paper wherever we have taped it down to the table.

7.2.2 Home in a practical machine

As mentioned above, although it looks tidy at first sight, it is often not a good idea to have the Home Z position as the surface of the table. Mach2 has a button to *Reference all* the axes (or you can Reference them individually). For an actual machine which has Reference switches installed this will move each linear axes (or chosen axis) until its switch is operated then move slightly off it. The Machine Co-ordinate system origin is then set to given X, Y, Z etc. values - frequently 0.0 - in which case referencing also homes the machine. You can define a non-zero value for the reference switches if you want but ignore this for now!

The Z reference switch is generally set so the Controlled Point is at the highest Z position above the table. Of course if the reference position is Machine Co-ordinate Z=0.0 then all the working positions are lower and will be negative Z values in Machine Co-ordinates.

Again if this is not totally clear at present do not worry. Having the Controlled Point (tool) out of the way when referenced/home is obviously practically convenient and it is easy to use the Fixture offset to set a convenient co-ordinate system for work on the table.

7.3 What about different lengths of tool?

If you are feeling confident so far then it is time to see how to solve another practical problem.

Suppose we now want to add a red rectangle to the drawing.

We jog the Z axis up and put the red pen in the holder in place of the blue one. Sadly the red pen is longer than the blue one so when we go to the Current Co-ordinate System origin the tip smashes into the table. (Figure 7.5)

Mach2, like other CNC controllers, has a way for storing information about the tools (pens in our system). This **Tool Table** allows you to tell the system about up to 256 different tools.

On the Tables screen you will see space for a Tool number and information about the tool. The information shown will be relevant to the type of machine you are controlling. For a mill the important thing is the offset in the Z direction of the tool (i.e. its length). On a lathe the tool cutting tip can be offset in both the X and Z directions.

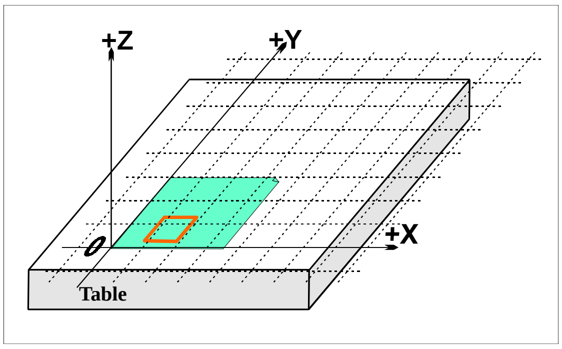


Figure 7.4 - Now we want another color

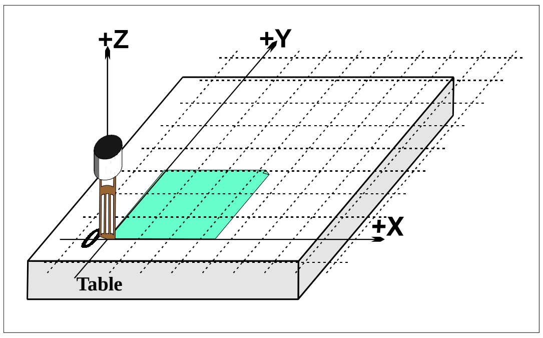


Figure 7.5 - Disaster at 0,0,0!

Information about the tool diameter or cutting tip radius is also stored which used for Cutter Compensation (q.v.)

Setting the Tool Z (and for lathe X) offsets is very similar to setting up Fixture offsets. There are some difference which will be described later.

7.3.1 Presettable tools

We will assume your machine has a tool-holder system which lets you put a tool in at exactly the same position each time. This might be a mill with lots of chucks or something like an Autolock chuck (figures 7.10 and 7.11 - where the centre-hole of the tool is registered against a pin) or a lathe with a Dickson quick-change tool post (figure 7.6). If your tool position is different each time then you will have to set up the offsets each time you change it. This will be described later.

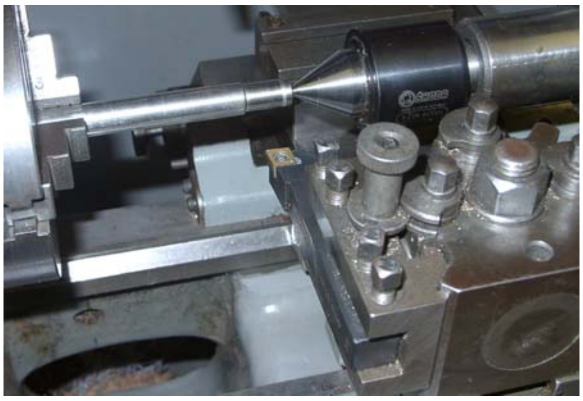


Figure 7.6 - Presettable lathe tool holder

In our drawing machine, suppose the pens register in a blind hole that is 1" deep in the pen holder. The red pen is 4.2" long and the blue one 3.7" long.

1. Suppose the machine has just been referenced/homed and a fixture offset defined for the corner of the paper with Z = 0.0 being the table using the bottom face of the empty pen holder. You would jog the Z axis up say to 5" and fit the blue pen. Enter "1" (which will be the blue pen) in the Tool number DRO but not turn the offset LED On yet. Jog the Z down to touch the paper. The Z axis DRO would read 2.7 as the pen sticks 2.7" out of the holder. Then "Touch" for the Z offset of the Tool table. This would load the (2.7") into the Z offset of Tool #1. Clicking the Offset On/Off toggle would light the LED and apply the tool offset and so the Z axis DRO will read 0.9 You could draw the square by running the example Part Program as before.
2. Next to use the red pen you would jog the Z axis up (say to Z = 5.0 again) to take out the blue pen and put in the red. Physically swapping the pens obviously does not alter the axis DROs. Now you would, switch Off the tool offset LED, select Tool #2 , jog and touch at the corner of the paper. This would set up tool 2's Z offset to 3.2". Switching On the offset for Tool #2 again will display Z = 0.0 on the axis DRO so the Part Program would draw the red square (over the blue one).
3. Now that tools 1 and 2 are set up you can change them as often as you wish and get the correct Current Co-ordinate system by selecting the appropriate tool number and switching its offsets on. This tool selection and switching on and off of the offsets can be done in the Part Program (M6, G43 and G49)

7.3.2 Non-presettable tools

Some tool holders do not have a way of refitting a given tool in exactly the same place each time. For example the collet of a router is usually bored too deep to bottom the tool and a conventional lathe tool post does not have an easy way op knowing where to put the tool. In this case it may still be worth setting up the tool offset (say with tool #1) each time it is changed. If you do it this way you can still make use of more than one fixture (see 2 and 3 pin fixtures illustrated below). If you do not have a physical fixture it may be just as easy to redefine the Fixture offsets each time you change the tool.

7.4 Operational differences between Fixture tables and Tool tables

In reading the description, you may have noticed differences in the control of fixtures and tools. If you wonder why this is so then read on in this section!

The differences are a result of a way that standard G-codes are defined. The differences are summarised below:

♦ Tools number from 0. In G-code there is no way of turning off fixtures so Mach2 defines fixture #0 as having no offsets. Hence user defined fixtures number from 1.

♦ When you chose a fixture number, it is (has to be) automatically turned on. The *Fixture off* button therefore chooses Fixture 0.

- Tool offsets are turned on and off by G43 and G49 so the chosen tool number does not automatically have to be turned on.

7.5 Drawing lots of copies - real fixtures

Now imagine we want to draw on many sheets of paper. It will be difficult to tape each one in the same place on the table and so will be necessary to set the Fixture offsets each time. Much better would be to have a plate with pins sticking out of it and to use pre-punched paper to register on the pins. You will probably recognise this as an example of a typical fixture which has long been used in machine shops. Figure 7.7 shows the machine so equipped. It would be common for the fixture to have dowels or something similar so that it always mounts in the same place on the table.

We could now move Current Co-ordinate system by setting the Fixture offsets of Fixture #1 to the corner of the paper on the actual fixture. Running the example program would draw the square exactly as before. This will of course take care of the difference in Z co-ordinates caused by the thickness of the fixture. We can put new pieces of paper on the pins and get the square in exactly the right place on each with no further setting up.

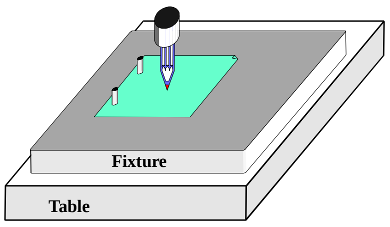


Figure 7.7 - Machine with two pin fixture

We might also have another fixture for three-hole paper (Figure 7.8) and might want to swap between the two and three pin fixtures for different jobs so Fixture #2 could be defined for the corner of the paper on the three pin fixture.

You can, of course define any point on the fixture as the origin of its offset co-ordinate system. For the drawing machine we would want to make the bottom left corner of the paper be X=0 & Y=0 and the top surface of the fixture be Z=0.

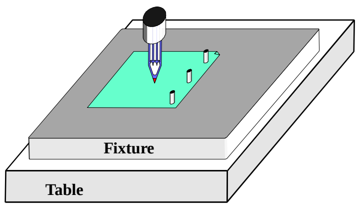


Figure 7.8 - Three pin fixture

It is common for one physical fixture to be able to be used for more than one job. Figure 7.9 shows the two and three hole fixtures combined. You would of course have two entries in the Fixture Table corresponding to the offsets to be used for each. In figure 7.8 the Current Co-ordinate system is shown set for using the two-hole paper option.

7.6 Saving your definitions of Fixtures and Tools

When you are dealing with physical fixtures and with tools holders that allow fixed positioning for each different tool you may wish to store the offset information in tables which persist when you shut down Mach2 and/or the computer and run it again. Buttons on the Tools screen allow you to look at all the entries in your tables and to Save

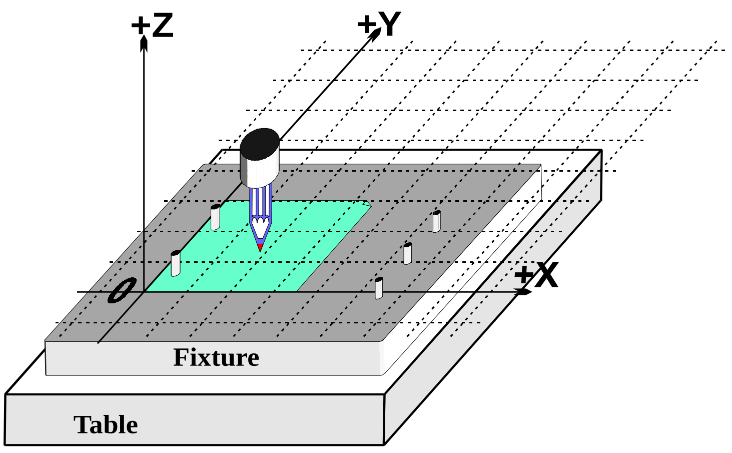


Figure 7.9 - A double fixture

them on disc.

7.7 Practicalities of "Touching"

7.7.1 End mills and lathe tools

On a manual machine tool it is quite easy to feel on the handles when a tool is touching the work but for accurate work it is better to have a feeler or slip gage so you can tell when it is being pinched. This is illustrated on a mill in figure 7.10.

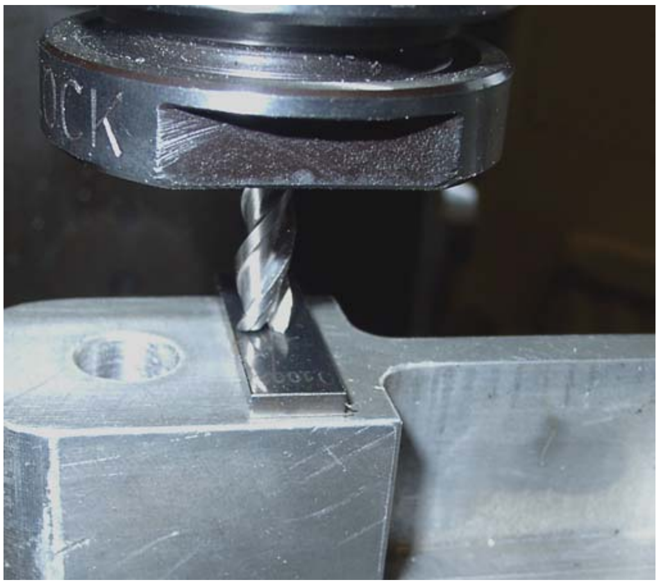


Figure 7.10 - Using a slip gage when touching Z offset on a mill

On the Tables screen you can enter the thickness of this feeler or slip gage into the *Touch Correction* DRO and turn the correction On. When you set an offset DRO for a tool or fixture then the thickness of the gage will be allowed for. A LED flashes to warn you that the correction is active whenever you do a touch.

For example suppose you had the axis DRO Z = -3.518 with the 0.1002" slip lightly held. You enter 0.1002 in the *Touch Correction* DRO, turn it On and click Touch for the Z offset of Fixture #1. After the touch the axis DRO reads Z = 0.1002 (i.e the Controlled Point is 0.1002) and Fixture #1 is On with its Z offset -3.6182.

7.7.2 Edge finding

It is very difficult to accurately set a mill to an edge in X or Y due to the flutes of the tool. A special edge-finder tool helps here, Figure 7.11 shows the minus X edge of a part being found.

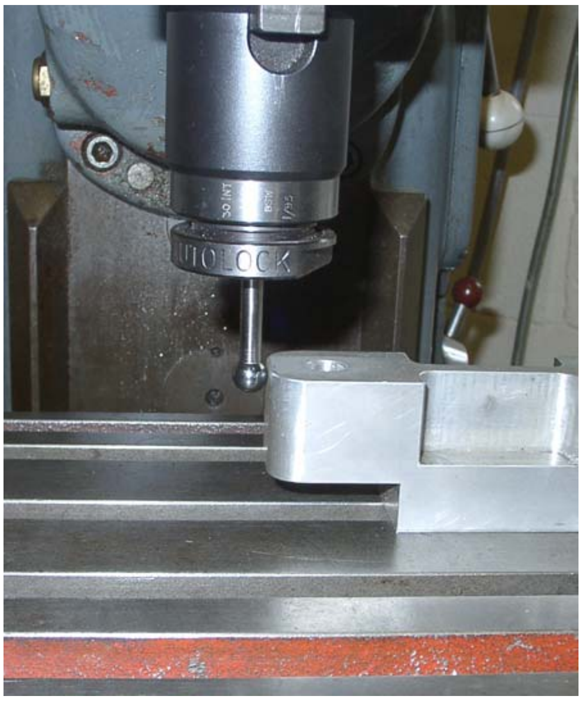


Figure 7.11 - Edge-finder in use on a mill

The Touch Correction can be used here as well. You will need the radius of the probe tip and the thickness of any feeler or slip gage. If you touch on the "negative" side of the part (like in the example) then the correction will be a negative value. A touch on the other side will have a positive correction.

7.8 Workpiece offsets

There is one further way of offsetting the Controlled Point. This is most frequently used within a Part Program using G-codes 92, 92.1, 92.3 or 92.4. When you use G92 you tell Mach2 what you want the co-ordinates of the Controlled Point to be. This does **not** move the tool it just adds another set of offsets to the origin of the Current Co-ordinate system.

The simplest example is, at a given point, to set X & Y to zero with G92 but you can set any values. The effect is, of course to alter where, on the workpiece, the moves of the part program take the tool. G92 offsets are, therefore, sometimes called **Workpiece Offsets**.

It is possible to set the G92/Workpiece Offsets by typing the desired value of a co-ordinate of the Controlled Point into the corresponding Axis DRO on any screen. Remember no movement takes place it is just that another set of offsets is applied to the origin.

The easiest way to cancel G92 offsets is to enter "G92.1" on the MDI line.

Warning! Almost everything that can be done with G92 offsets typing into DROs can be done better using Fixtures. Many operators find it hard to keep track of three sets of offsets (Fixture, Tool and G92) and if you get confused you will soon break either your tool or worse your machine!

You may remember that the *Mach* (on some screens *Machine*) button in the DRO Display Type group shows you the Controlled Point in absolute machine co-ordinates. If the *G92* LED is lit then axis DROs are showing the Controlled Point in the Current Co-ordinate system with all offsets (Tool, Fixture and G92 applied). This is the most useful display as all moves are in this co-ordinate system. If the *Fixture* LED is lit then axis DROs show the controlled point with the Tool and Fixture offsets applied. Part Program moves are **not** in this co-ordinate system but the display is useful as it shows the effects of the G92 you may have applied. If you have not entered any values into the axis DROs or used G92 then Fixture and G92 will show the same values.

To summarise you will usually run the machine showing the Current Co-ordinate System values (G92 LED lit) and can check things relative to the absolute machine co-ordinates using the *Mach/Machine* button.

7.9 Tool diameter and tool tip radius

These are two values in the information in the Tool Table. A brief explanation will help you see the reason for them. For details you need to refer to the chapter of Cutter Compensation

7.9.1 Tool diameter

Suppose the blue square drawn using our machine is the outline for a hole in the lid of a child's shape-sorter box into which a blue cube will fit. Remember G-codes move the Controlled Point. The example Part Program drew a 1" square. If the tool is a thick felt pen then the hole will be significantly smaller than 1" square. See figure 7.12.

The same problem obviously occurs with an endmill/slot drill. You may want to cut a pocket or be leaving an island. These need different compensation.

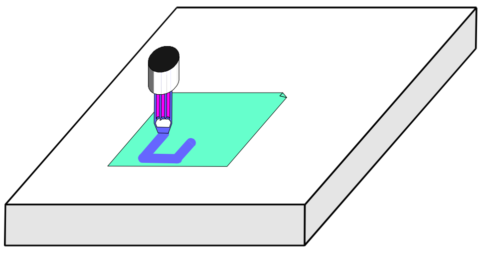


Figure 7.12 - Using a large diameter tool (felt pen)

This sounds easy to do but in practice there are many devils in the detail concerned with the beginning and end of the cutting. It is usual for your CAD/CAM software to deal with these issues. Mach2, however, allows a Part Program to compensate for the diameter of the chosen tool with the actual cutting moves being specified as, say, the 1" square. This feature is important if the author of the Part Program does not know the exact diameter of the cutter that will be used (e.g. it may be smaller than nominal due to repeated sharpening). The tool table lets you define the diameter of the tool. See Cutter Compensation chapter for full details.

7.9.2 Tip radius

This applied to the Mach2Lathe. For full details see its manual. The turning tool will not have a perfectly sharp corner. If it is cutting with constant X or facing with constant Z then this does not cause a problem but the dimensions of a chamfer or radius are affected by the tip radius. The Tool Table lets you define the radius of the tool tip.

8. Optimising your Mach2 and machine

In this chapter we will look at making Mach2 easy to use on your machine and for your type of work. The topics covered include: Screen Designer, macro writing and other "ease of use" customisations.

You should be aware that the more changes you make to your system the harder it will be for someone to support it from a distance (e.g. by e-mail) so you need to aim for a balance. Add your own customisation bit by bit as you become confident with using Mach2

8.1 Screen Designer

Mach2Mill comes with a standard set of screen layouts to suit many uses. You are strongly advised to use a screen resolution of 1024 x 768 pixels if you can, but screens are provided for 800 x 600 and 640 x 480 pixels.

The Screen Designer lets you change the layout of any or all of the information displayed on screen by Mach2. You can, if you wish, design a complete set of custom screens but, as there are more than 500 individual objects on the standard screens, most users will probably only want to make detailed changes to the layout.

The screen to use is loaded into Mach2 using the Layouts>Load Layout menu. Screen layouts are stored in files with the .SET extension. The Screen Editor application is used to edit .SET files.

8.1.1 Screen Editor basics

The Screen Editor allows you to create and edit a set of eight screens which form a .SET file. A screen consists of a collection of *Controls*. The Controls supported are:

1) DRO (digital read-out) - to display and optionally enter numerical values
2) Button - to request an action
3) Bitmap button - action but with image rather than text giving function
4) Bitmap - display a picture
5) Joystick - to control manual jogging
6) Label - to identify other objects or be a placeholder for a text display
7) LED (light emitting diode) - On/Off or warning indicator
8) MDI (manual data input line) - to allow input of "G-code" line
9) G-code list - to display current part program text
10) Toolpath - to display path followed by tool in current part program.

The screens are numbered 1 to 8 and screen P can be used to place controls which will appear on ever screen (P is for Persistent). This is used, for example, for the buttons used to change from screen to screen.

8.1.2 Try out the Editor

Close Mach2, if it is running, and use the shortcut you installed in the desktop to run the Screen editor.

You should see a mainly blank screen with a menu bar, a Tool bar (top of screen), a Status bar (bottom of screen) and a pallet of buttons. The pallet can be dragged around the screen to a convenient place out of the way of controls that you are editing. The Toolbar and Status bar can be hidden and viewed using the View menu. You will only need to do this if you want to place controls very near the bottom of your finished screen.

As a trial choose File>New to get a blank screen. Click a button on the pallet and the click anywhere on the screen. You will get a default sized control of the type chosen. You need to press a button and click for each control you want.

By default you will be putting controls on Screen #1. Use the numbered buttons on the Toolbar to select other screens and place controls on them. Try placing some controls on the "P screen" and see how they appear on screens 1 to 8.

If you click on a control you have placed you will see that it becomes selected and is drawn with the traditional sizing handles. It can be moved around the screen and resized by dragging the outline or the relevant handles. Multiple objects can be selected using Shift-click. One selected object in a set can be deselected by Control-click.

The selected objects can be cut or copied to a clipboard and the contents of the clipboard can be pasted onto the same or another of the eight screens. Notice, that although you can run more than one copy of Screen Designer at one time editing different .SET files, each has its own clipboard so you cannot move controls from one file to another using the clipboard.

When you resize a control with the handles then its contents are scaled to fit the new size. This is how you decide on the font size for DROs, labels etc.

8.1.3 *Getting a tidy visual effect*

The Screen Editor has several features to ensure a neat looking screen design. These are accessed from the icons at the right hand end of the toolbar.

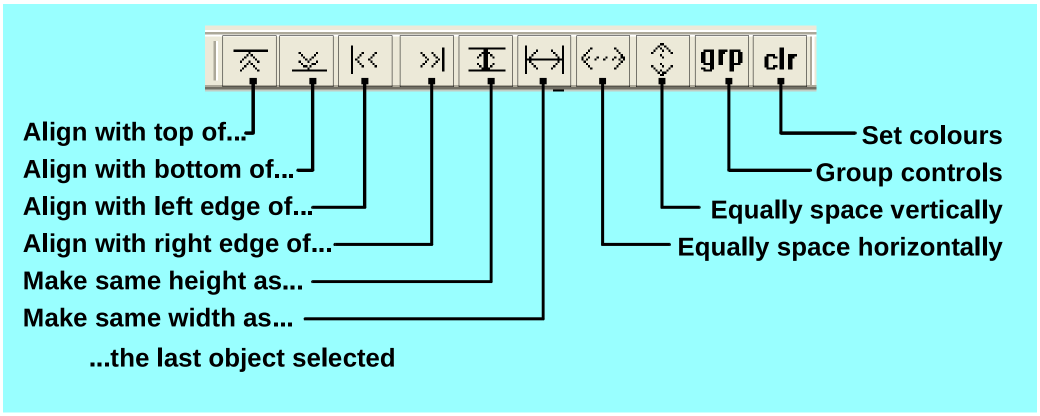


Figure 8.1 - Screen Designer alignment tools

With the exception of the "Clr" - for Color button they all act on a multiple selection of controls. These will often be of the same type (e.g. six DROs for the six axes) but do not need to be.

The **last** control selected in the multiple selection is special as it is the *master control* of the selection and other controls will be moved, sized etc. in relation to it.

8.1.3.1 Alignment icons

The first four icons on the bar align the appropriate edge of selected controls with that same edge of the master control. For example "Align with left edge" would move all the controls selected horizontally so all the left hand sides were in a vertical line with the left edge of the master one. Figure 8.2 shows the DROs just after drawing. They are then selected with the top one chosen last and Align Left Edge clicked. Figure 8.3 shows the result.

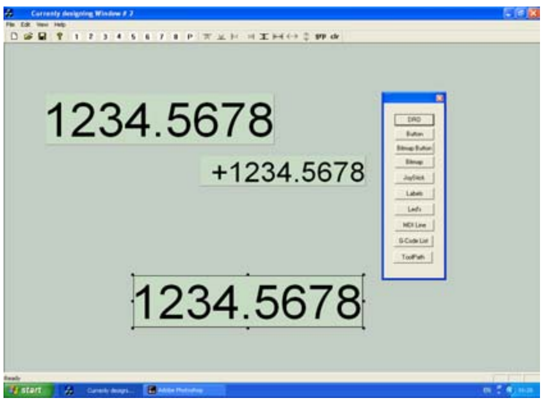


Figure 8.2 - Ragged DROs as drawn

You need to be slightly careful as if you align left and, say, top then all the controls will be on top of each other. If this happens by accident then you need to deselect them by clicking on a blank bit of screen, select them one by one and drag them apart.

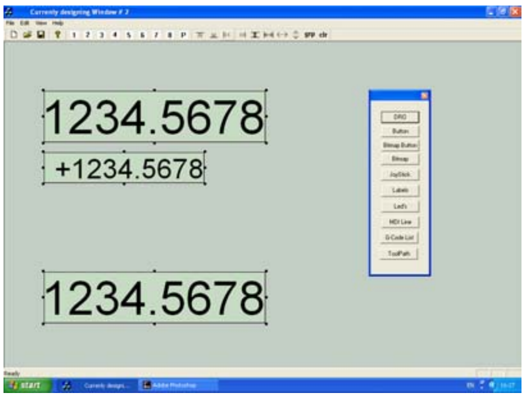


Figure 8.3 - Justified DROs

8.1.3.2 Sizing icons

In the example the controls are different sizes. This would not happen if you had used the clipboard to duplicate them but it is easy to make them all the same width and or height as the master of the selection. Figures 8.4 and 8.5 show this done in two stages with the two "Make same.." icons.

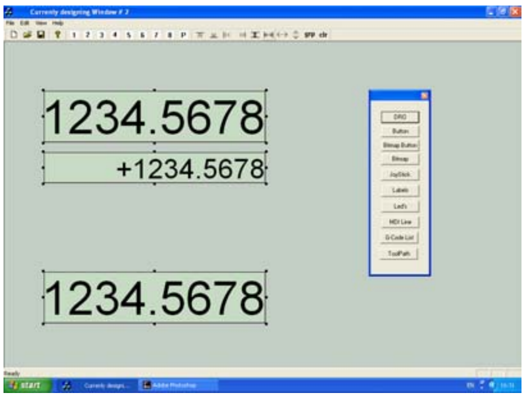


Figure 8.4 - Same width DROs

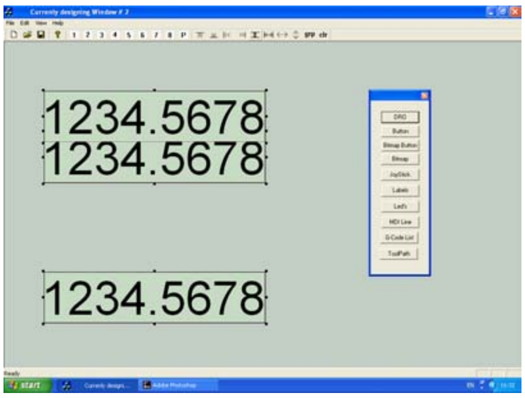


Figure 8.5 - Same size DROs

8.1.3.3 Spacing controls uniformly

Finally the "Equally space …" icons can be used to space the selected controls equally. The result of this is shown in figure 8.6.

It is worth spending a bit of time experimenting with these icons on different types of control to get a feel for what can be done to make a neat layout with very few clicks of the mouse.

You should get into the habit of saving your layout frequently as the current version of the Screen Designer has no Edit>Undo facility.

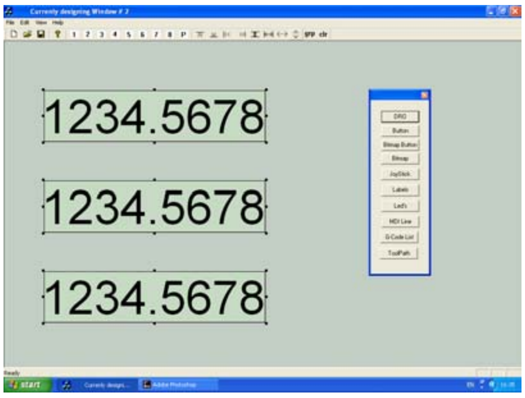


Figure 8.7 - Equally spaced controls

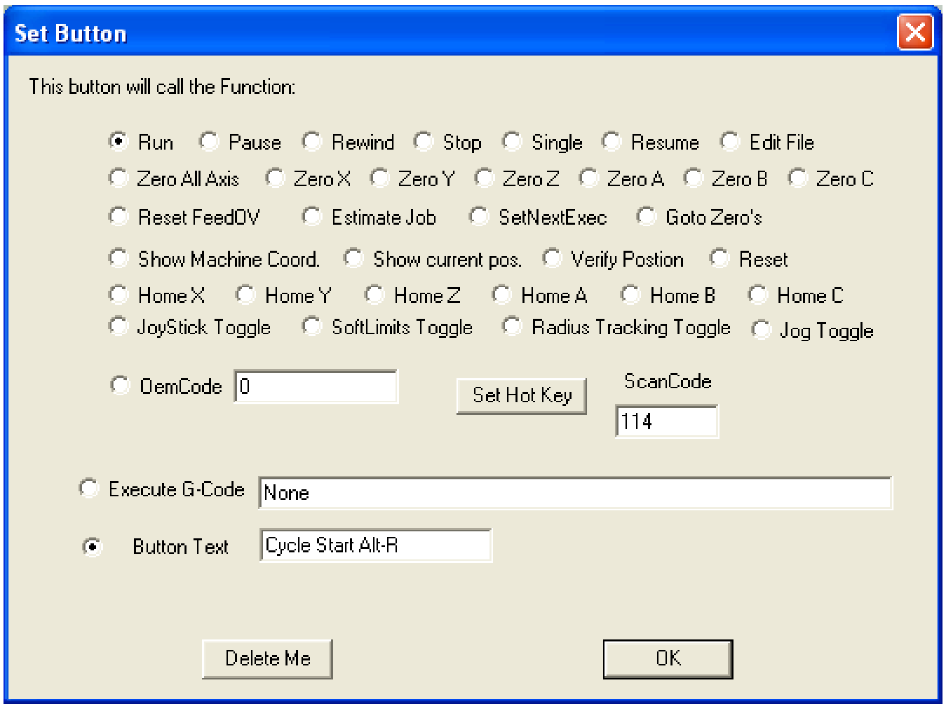
It is very important to notice that Screen Designer will close without prompting you to save modified screens. It is therefore **your responsibility** to do these Saves before you click the close box unless you want to lose all your work!

8.1.4 Making the controls work

It is now time to see how to make control "work". It is easiest to do this by looking at the standard Mach2 screens. Assuming you have a 1024 x 768 screen, save your trial screens (perhaps in file `Junk.set`) and open `Mach2\1024.set`

You will be shown Screen #1 - the "Program Run (P)" screen. See which screens are displayed for screens 2 to 7. **Do not save this work** as you are using the real screen layout!

Double-click the *Cycle Start* button on screen #1. You will be shown its properties which will look similar to figure 8.8.


Set Button

This button will call the Function:

[✓ Run] Pause Rewind Stop Single Resume Edit File

Zero All Axis Zero X Zero Y Zero Z Zero A Zero B Zero C

Reset FeedOV Estimate Job SetNextExec Goto Zero's

Show Machine Coord. Show current pos. Verify Postion Reset

Home X Home Y Home Z Home A Home B Home C

JoyStick Toggle SoftLimits Toggle Radius Tracking Toggle Jog Toggle

OemCode: 0

Set Hot Key

ScanCode: 114

Execute G-Code: None

[✓ Button Text]: Cycle Start Alt-R

Delete Me OK

Figure 8.8 - Properties of standard *Cycle Start* button

8.1.4.1 By named function

Notice there is a long list of radio buttons for functions that a button can call. *Cycle Start* uses the one called "Run". One possibility is OEM code which we will look at later. The button has a title given in its "Button Text" box and can be activated by a "Hot key" (Alt-R in fact). The user is reminded of this in the title - some buttons are very small so the shortcut is given in a separate label. Mach2 identifies keys by their "Scan code". Alt-R is code 114. If you click *Set Hot Key* and type any other key (or Ctrl/Alt key combination) this will set a new hot key for *Cycle Start*. You will see its scan code. For safety set it back to Alt-R.

The list of shortcuts used on the standard screens is given in Appendix 2.

Close the dialog by OK (beware of the Delete Me button!)

8.1.4.2 By OEM code

Now look at the properties of the *Flood Tog* button by double clicking it. You will notice that its function is defined as OEM code 113 in the list of radio buttons. OEM codes are a way of extending the range of buttons for more esoteric purposes without the list of names becoming much too long. The allocation of named functions and OEM codes to buttons on the standard screens is given in Appendix 3.

If you look at the properties on the *MDI (M)* button you will see that its OEM code is 2. Codes 1 to 8 are used to select the eight possible screens. If you look at the codes in all the screen selection buttons you will see which screen # is allocated to each and this will tie up with the rather odd order you will have discovered at the beginning of this section.

8.1.4.3 By G-code line

There is a final way of making a button work. You can provide some G-code to be executed when it is clicked. As an example look at the G92X0 button on the MDI screen. You will see that it (not unexpectedly!) issues the command "G92X0". You can define buttons to do whatever you want that can be done by a single MDI line.

Beware: If you provide hidden G-code like this, then any scaling which is active on axes at the time you click the button will be applied to the co-ordinates built into your button. The problem does not, of course, apply to zero values whatever scaling is applied.

8.1.5 Properties of other types of control

Now work your way through the different types of control and by inspecting the properties of examples of each type you will be able to see how Mach2 standard screens are built. Some controls are very simple (e.g. the joystick ball) while others are complex (LEDs are different colors, can flash etc.).

8.1.6 Properties of Intelligent Labels

If a label has some reserved text in it then this text is replaced when Mach2 runs by information about what is happening (e.g. the name of the part program which is loaded).

The following intelligent labels are defined in the current version:

File (the part program), Error (the last Mach2 error message), Mode (the current modes of the system), Profile (the name of the current profile file)

The intelligent label text is case sensitive. For example "FILE" displays the word FILE but "File" displays the current part program's file name.

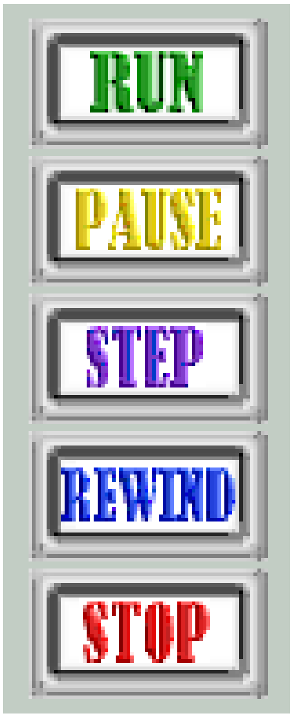


Figure 8.9 -Bitmap buttons

8.1.7 DRO groups

Mach2 is designed so it can be operated without a mouse or other pointing device although if one is provided it will probably be best to use it.

DROs can be selected by a hot key and the cursor keys can be used to move around them. To make this as convenient as possible they are combined into groups. Left and right keys go from group to group and up and down keys move within a group. The ***Grp*** icon in Screen Designer allows you to allocate a group number to a multiply selected group of DROs.

8.1.8 Use of Bitmaps

8.1.8.1 Bitmap buttons

A bitmap button can be used in place of any normal button to give emphasis to the function of the button. For an example see figure 8.9

8.1.8.2 Visual grouping with bitmaps

A set of related DROs buttons LEDs etc. can be placed "in" a frame or bezel by placing a suitable bitmap "underneath". A example is shown in figure 8.10. You should complete the placing of the controls before drawing the bitmap as otherwise you will not be able to select the handles to resize the controls.

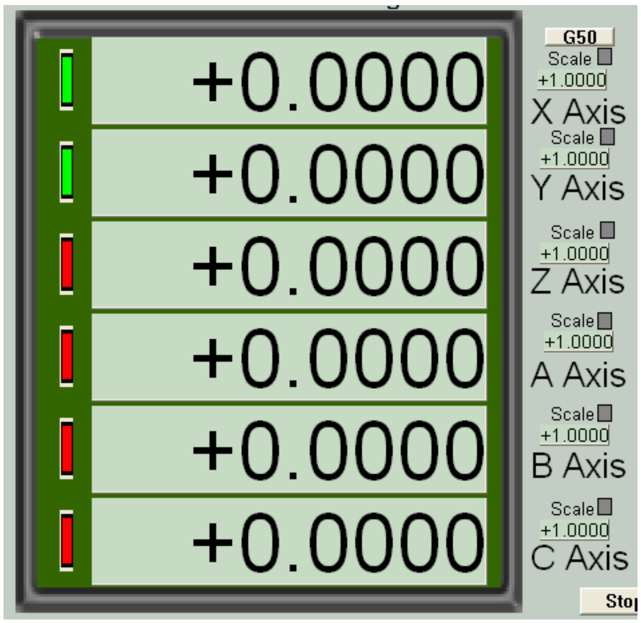


Figure 8.10 - Bitmap frame for DROs

You can use one bitmap file for many bezels although stretching a square one to become very long and thin will distort the width of the border so you might need to have more than one basic shape. Try to minimise the different styles and colour schemes or your screens will look cluttered and be more difficult to use.

8.1.8.3 Dynamic changes with bitmaps

The Screen Designer does not copy the bitmaps when you place them on the screen at design time; it just sets up a link to the image file in the Mach2\Bitmaps folder. The bitmap is loaded when Mach2 itself is run. This means that you can replace the bitmaps with ones of your own design and by giving them the names used by the original designer of the screen you can customise the appearance of your system without having to run Screen Designer or having to place or size the bitmaps. Figure 8.11 shows this done to the set of DROs shown in figure 8.10

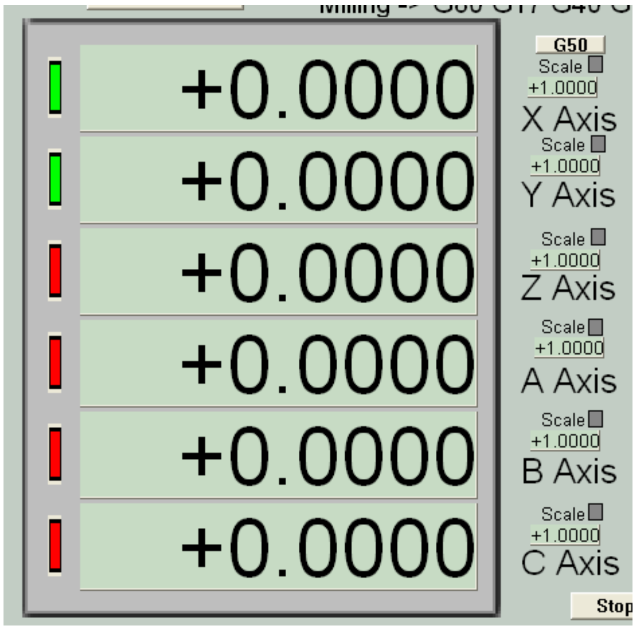


Figure 8.11 - An alternative frame link in dynamically

8.1.9 Colors

The Screen Designer ***Clr*** icon lets you define the colour scheme for your screens and controls. This is a global setting; it does not relate to the selected items.

8.1.10 Finally - caution

Do remember to save frequently and before exiting Screen Designer to avoid loss of your careful work. Beware of accidentally clicking the *New* button on the toolbar if using the *Save* icon on it!

8.2 Mach2 Macros

If Mach2 encounters an M-code that it does not understand (e.g. M66) then it will look for a file in the Mach2 Macros folder with a `.M1S` extension (e.g. `M66.m1s`) and execute the VB Script code in this.

Macro skeletons are provided for handling M6 tool changes and other simple functions like setting output signals.

You can write macros of your own which range from performing simple tasks (e.g. implementing non-standard M-codes from other controllers) to performing complex calculations on data which will be used in the part program. This might simulate, for example, engine turning as geometric pattern to decorate an object.

Warning: It is not advisable to write macros if you want you part programs to be portable to other controllers.

8.3 Other customisation

8.3.1 Global hotkeys

Configure>Set Axis Hotkeys allows you to set up the keys which will jog the axes when the jog ball is displayed on the current screen. Click the button for the axis and direction you want to set and then type the keystroke to be used. It is your responsibility yo make sure they are unique and do not clash with other hotkeys you might want to use. See appendix for list of standard hotkeys.

You can also define hotkeys which will select the MDI window, will select a DRO and will display the File>Load G-code dialog. These are useful if you are controlling Mach2 without a mouse or similar pointing device.

8.3.2 Wizards

Mach2 has a facility for a range of Wizards to automate complex tasks.

In the current release a wizard is provided to write a part program to digitise a 3D object.

Section to be provided later

9. DXF and image file import and Engraving

This covers importing DFX and bitmaps and other engraving issues. I will need help here as I have no engraving experience yet. JAP

10. Cutter compensation

Cutter compensation is a feature of Mach2 which you many never have to use. Most CAD/CAM programs can be told the nominal diameter of your mill and will output part programs which cut the part outline or pocket which you have drawn by themselves allowing for the tool diameter.

Having compensation in Mach2 allows you to: (a) use a tool different in diameter from that programmed (e.g. because it has be reground) or (b) to use a part program that describes the desired outline rather than the path of the center of the tool (perhaps one written by hand).

However, as compensation is not trivial, it is described in detail in this chapter should you need to use it.

10.1 Introduction to compensation

As we have seen Mach1 controls the movement of the Controlled Point. In practice no tool (except perhaps a V-engraver) is a point so cuts will be made at a different place to the Controlled Point depending on the radius of the cutter.

It is generally easiest to allow your CAD/CAM software to take account of this when cutting out pockets our the outline of shapes.

Mach2 does, however, support calculations to compensate for the diameter (radius) of the cutter. In industrial applications this is aimed at allowing for a cutter which, through regrinding, is not exactly the diameter of the tool assumed when the part program was written. The compensation can be enabled by the machine operator rather than requiring the production of another part program.

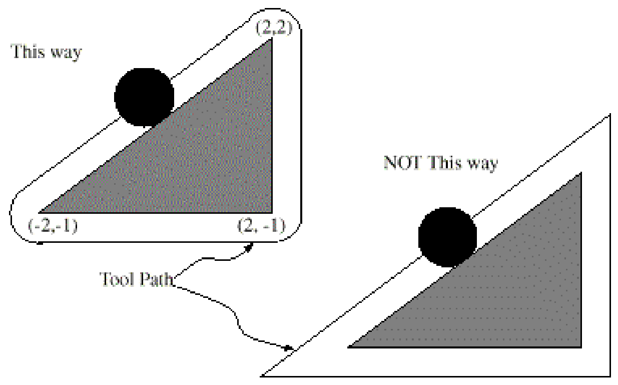


Figure 10.1 - Two possible toolpaths to cut triangle

Of the face of it, the problem should be easy to solve. All you need to do is to offset the controlled point by an appropriate X and Y to allow for the tool radius. Simple trigonometry gives the distances depending on the angle the direction of cut makes to the axes.

In practice it is not quite so easy. There are several issues but the main one is that the machine has to set a Z position before it starts cutting and at that time it does not know the direction in which the tool is going to be moving. This problem is solved by providing "pre-entry moves" which take place in waste material of the part. These ensure that the compensation calculations can be done before the actual part outline is being cut. Choice of a path which runs smoothly into the part's outline also optimises the surface finish. An exit move is sometimes used to maintain the finish at the end of a cut.

10.2 Two Kinds of Contour

Mach2 handles compensation for two types of contour:

- ♦ The contour given in the part program code is the edge of material that is not to be machined away. We will call this type a "material edge contour". This is the sort of code that might be "hand written"
- ♦ The contour given in the NC code is the tool path that would be followed by a tool of exactly the correct radius. We will call this type a "tool path contour". This is the sort of code that a CAD/CAM program might produce if it is aware of the intended cutter diameter

The interpreter does not have any setting that determines which type of contour is used, but the numerical description of the contour will, of course, differ (for the same part geometry) between the two types and the values for diameters in the tool table will be different for the two types.

10.2.1 Material Edge Contour

When the contour is the edge of the material, the outline of the edge is described in the part program. For a material edge contour, the value for the diameter in the tool table is the actual value of the diameter of the tool. The value in the table must be positive. The NC code for a material edge contour is the same regardless of the (actual or intended) diameter of the tool.

Example1:

Here is an NC program which cuts material away from the outside of the triangle in figure 10.1 above. In this example, the cutter compensation radius is the actual radius of the tool in use, which is 0.5, The value for the diameter in the tool table is twice the radius, which is 1.0.

```
N0010 G41 G1 X2 Y2 (turn compensation on and make entry move)
N0020 Y-1 (follow right side of triangle)
N0030 X-2 (follow bottom side of triangle)
N0040 X2 Y2 (follow hypotenuse of triangle)
N0050 G40 (turn compensation off)
```

This will result in the tool following a path consisting of an entry move and the path shown on the left going clockwise around the triangle. Notice that the co-ordinates of the triangle of material appear in the NC code. Notice also that the tool path includes three arcs which are not explicitly programmed; they are generated automatically.

10.2.2 Tool Path Contour

When the contour is a tool path contour, the path is described in the part program. It is expected that (except for during the entry moves) the path is intended to create some part geometry. The path may be generated manually or by a CAD/CAM program, considering the part geometry which is intended to be made. For Mach2 to work, the tool path must be such that the tool stays in contact with the edge of the part geometry, as shown on the left side of figure 10.1. If a path of the sort shown on the right of figure 10.1 is used, in which the tool does not stay in contact with the part geometry all the time, the interpreter will not be able to compensate properly when undersized tools are used.

For a tool path contour, the value for the cutter diameter in the tool table will be a small positive number if the selected tool is slightly oversized and will be a small negative number if the tool is slightly undersized. As implemented, if a cutter diameter value is negative, the interpreter compensates on the other side of the contour from the one programmed and uses the absolute value of the given diameter. If the actual tool is the correct size, the value in the table should be zero.

Tool Path Contour example:

Suppose the diameter of the cutter currently in the spindle is 0.97, and the diameter assumed in generating the tool path was 1.0. Then the value in the tool table for the diameter for this tool should be -0.03. Here is an NC program which cuts material away from the outside of the triangle in the figure.

```
N0010 G1 X1 Y4.5 (make alignment move)
```

```
N0020 G41 G1 Y3.5 (turn compensation on and make first entry
             move)
N0030 G3 X2 Y2.5 I1 (make second entry move)
N0040 G2 X2.5 Y2 J-0.5 (cut along arc at top of tool path)
N0050 G1 Y-1 (cut along right side of tool path)
N0060 G2 X2 Y-1.5 I-0.5 (cut along arc at bottom right of tool
             path)
N0070 G1 X-2 (cut along bottom side of tool path)
N0080 G2 X-2.3 Y-0.6 J0.5 (cut along arc at bottom left of
             tool path)
N0090 G1 X1.7 Y2.4 (cut along hypotenuse of tool path)
N0100 G2 X2 Y2.5 I0.3 J-0.4 (cut along arc at top of tool
             path)
N0110 G40 (turn compensation off)
```

This will result in the tool making an alignment move and two entry moves, and then following a path slightly inside the path shown on the left in figure 10.1 going clockwise around the triangle. This path is to the right of the programmed path even though G41 was programmed, because the diameter value is negative.

10.2.2.1 First Move

The algorithm used for the first move when the first move is a straight line is to draw a straight line from the destination point which is tangent to a circle whose center is at the current point and whose radius is the radius of the tool. The destination point of the tool tip is then found as the center of a circle of the same radius tangent to the tangent line at the destination point. This is shown in figure 10.2. If the programmed point is inside the initial cross section of the tool (the circle on the left), an error is signalled.

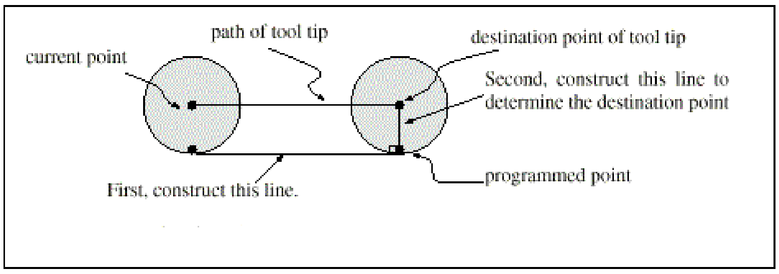

path of tool tip
destination point of tool tip
current point
Second, construct this line to determine the destination point
programmed point
First, construct this line.

Figure 10.2 - First cutter compensation move - Straight

If the first move after cutter radius compensation has been turned on is an arc, the arc which is generated is derived from an auxiliary arc which has its center at the programmed center point, passes through the programmed end point, and is tangent to the cutter at its current location. If the auxiliary arc cannot be constructed, an error is signalled. The generated arc moves the tool so that it stays tangent to the auxiliary arc throughout the move. This is shown in figure 10.3.

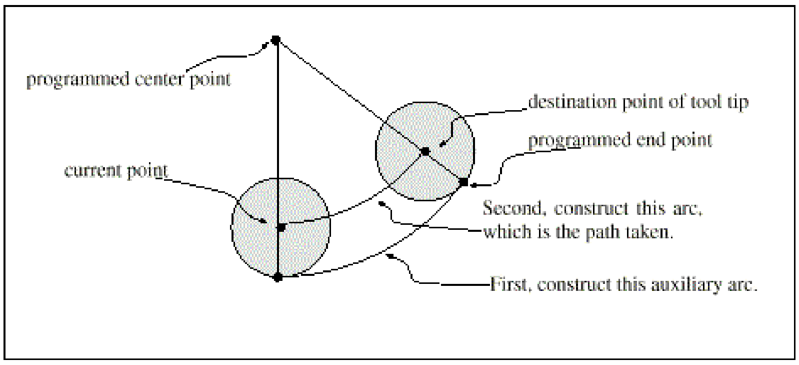

programmed center point
destination point of tool tip
programmed end point
current point
Second, construct this arc, which is the path taken.
First, construct this auxiliary arc.

Figure 10.3 - First cutter compensation move - Arc

Regardless of whether the first move is a straight line or an arc, the Z axis may also move at the same time. It will move linearly, as it does when cutter radius compensation is not being used. Rotary axis motions (A, B, and C axes) are allowed with cutter radius compensation, but using them would be very unusual. After the entry moves of cutter radius compensation, the interpreter keeps the tool tangent to the programmed path on the appropriate side. If a convex corner is on the path, an arc is inserted to go around the corner. The radius of the arc is half the diameter given in the tool table. When cutter radius compensation is turned off, no special exit move takes place.

The next move is what it would have been if cutter radius compensation had never been turned on and the previous move had placed the tool at its current position.

10.2.2.2 Programming Entry Moves

In general, an alignment move and two entry moves are needed to begin compensation correctly. However, where the programmed contour is a material edge contour and there is a convex corner on the contour, only one entry move (plus, possibly, a pre-entry move) is needed. The general method, which will work in all situations, is described first. We assume here that the programmer knows what the contour is already and has the job of adding entry moves.

General Method

The general method includes programming an alignment move and two entry moves. The entry moves given above will be used as an example. Here is the relevant code again:
N0010 G1 X1 Y4.5 (make alignment move to point C)

```
N0020 G41 G1 Y3.5 (turn compensation on and make first entry
                  move to point B)
N0030 G3 X2 Y2.5 I1 (make second entry move to point A)
```

See figure 10.4. The figure shows the two entry moves but not the alignment move. First, pick a point A on the contour where it is convenient to attach an entry arc. Specify an arc outside the contour which begins at a point B and ends at A tangent to the contour (and going in the same direction as it is planned to go around the contour). The radius of the arc should be larger than half the diameter given in the tool table. Then extend a line tangent to the arc from B to some point C, located so that the line BC is more than one radius long. After the construction is finished, the code is written in the reverse order from the construction.

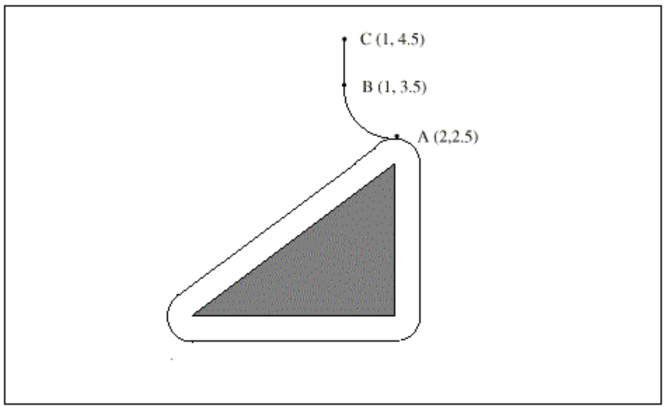


Figure 10.4 - The entry moves (omitting first alignment move)

Cutter radius compensation is turned on after the alignment move and before the first entry move. In the code above, line N0010 is the alignment move, line N0020 turns compensation on and makes the first entry move, and line N0030 makes the second entry move.

In this example, the arc AB and the line BC are fairly large, but they need not be. For a tool path contour, the radius of arc AB need only be slightly larger than the maximum possible deviation of the radius of the tool from the exact size. Also for a tool path contour, the side chosen for compensation should be the one to use if the tool is oversized. As mentioned earlier, if the tool is undersized, the interpreter will switch sides.

Simple Method

If the contour is a material edge contour and there is a convex corner somewhere on the contour, a simpler method of making an entry is available. See figure 10.5. First, pick a convex corner, D. Decide which way you want to go along the contour from D. In

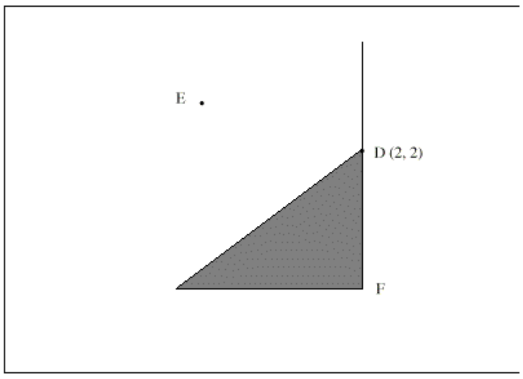


Figure 10.5 - Simple entry move

our example we are keeping the tool to the left of the contour and going next towards F. Extend the line FD (if the next part of the contour is an arc, extend the tangent to arc FD from D) to divide the area outside the contour near D into two regions. Make sure the center of the tool is currently in the region on the same side of the extended line as the material inside the contour near D. If not, move the tool into that region. In the example, point E represents the current location of the center of the tool. Since it is on the same side of line DF as the shaded triangle, no additional move is needed. Now write a line of NC code that turns compensation on and moves to point D.

```
    N0010 G41 G1 X2 Y2 (turn compensation on and make entry
               move)
```

This method will also work at a concave corner on a tool path contour, if the actual tool is oversized, but it will fail with a tool path contour if the tool is undersized.

11. Mach 2 G- and M-code language reference

This section defines the language (G-codes etc.) that are understood and interpreted by Mach2.

Certain functionality which was defined for machines in the NIST NMC (Next Generation Controller) architecture but is not presently implemented my Mach2 is given in grey type in this chapter. If this functionality is important for your application then please let ArtSoft Corporation know your needs and they will be included in our development planning cycle.

11.1 Some definitions

11.1.1 Linear Axes

The X, Y, and Z axes form a standard right-handed co-ordinate system of orthogonal linear axes. Positions of the three linear motion mechanisms are expressed using co-ordinates on these axes.

11.1.2 Rotational Axes

The rotational axes are measured in degrees as wrapped linear axes in which the direction of positive rotation is counterclockwise when viewed from the positive end of the corresponding X, Y, or Z-axis. By "wrapped linear axis," we mean one on which the angular position increases without limit (goes towards plus infinity) as the axis turns counterclockwise and deceases without limit (goes towards minus infinity) as the axis turns clockwise. Wrapped linear axes are used regardless of whether or not there is a mechanical limit on rotation.

Clockwise or counterclockwise is from the point of view of the workpiece. If the workpiece is fastened to a turntable which turns on a rotational axis, a counterclockwise turn from the point of view of the workpiece is accomplished by turning the turntable in a direction that (for most common machine configurations) looks clockwise from the point of view of someone standing next to the machine.

11.1.3 Scaling input

It is possible to set up scaling factors for each axis. These will be applied to the values of X, Y, Z, A, B, C, I, J and R words whenever these are entered. This allows the size of features machined to be altered and mirror images to be created - by use of negative scale factors.

The scaling is the first thing done with the values and things like feed rate are always based on the scaled values.

The offsets stored in tool and fixture tables are not scaled before use. Scaling may, of course, have been applied at the time the values were entered (say using G10).

11.1.4 Controlled Point

The controlled point is the point whose position and rate of motion are controlled. When the tool length offset is zero (the default value), this is a point on the spindle axis (often called the gauge point) that is some fixed distance beyond the end of the spindle, usually near the end of a tool holder that fits into the spindle. The location of the controlled point can be moved out along the spindle axis by specifying some positive amount for the tool length offset. This amount is normally the length of the cutting tool in use, so that the controlled point is at the end of the cutting tool.

11.1.5 Co-ordinated Linear Motion

To drive a tool along a specified path, a machining system must often co-ordinate the motion of several axes. We use the term "co-ordinated linear motion" to describe the situation in which, nominally, each axis moves at constant speed and all axes move from their starting positions to their end positions at the same time. If only the X, Y, and Z axes (or any one or two of them) move, this produces motion in a straight line, hence the word "linear" in the term. In actual motions, it is often not possible to maintain constant speed because acceleration or deceleration is required at the beginning and/or end of the motion. It is feasible, however, to control the axes so that, at all times, each axis has completed the same fraction of its required motion as the other axes. This moves the tool along same path, and we also call this kind of motion co-ordinated linear motion.

Co-ordinated linear motion can be performed either at the prevailing feed rate, or at rapid traverse rate. If physical limits on axis speed make the desired rate unobtainable, all axes are slowed to maintain the desired path.

11.1.6 Feed Rate

The rate at which the controlled point or the axes move is nominally a steady rate which may be set by the user. In the Interpreter, the interpretation of the feed rate is as follows unless inverse time feed rate (G93) mode is being used:

- For motion involving one or more of the linear axes (X, Y, Z and optionally A, B, C), without simultaneous rotational axis motion, the feed rate means length units per minute along the programmed linear XYZ(ABC) path
- For motion involving one or more of the linear axes (X, Y, Z and optionally A, B, C), with simultaneous rotational axis motion, the feed rate means length units per minute along the programmed linear XYZ(ABC) path combined with the angular velocity of the rotary axes multiplied by the appropriate axis Cirrection Diameter multiplied by pi ($\pi = 3.14152...$); i.e the declared "circumference" of the part
- For motion of one rotational axis with X, Y, and Z axes not moving, the feed rate means degrees per minute rotation of the rotational axis.
- For motion of two or three rotational axes with X, Y, and Z axes not moving, the rate is applied as follows. Let dA, dB, and dC be the angles in degrees through which the A, B, and C axes, respectively, must move. Let D = sqrt ($dA^2 + dB^2 + dC^2$). Conceptually, D is a measure of total angular motion, using the usual Euclidean metric. Let T be the amount of time required to move through D degrees at the current feed rate in degrees per minute. The rotational axes should be moved in co-ordinated linear motion so that the elapsed time from the start to the end of the motion is T plus any time required for acceleration or deceleration.

11.1.7 Arc Motion

Any pair of the linear axes (XY, YZ, XZ) can be controlled to move in a circular arc in the plane of that pair of axes. While this is occurring, the third linear axis and the rotational axes can be controlled to move simultaneously at effectively a constant rate. As in co-ordinated linear motion, the motions can be co-ordinated so that acceleration and deceleration do not affect the path.

If the rotational axes do not move, but the third linear axis does move, the trajectory of the controlled point is a helix.

The feed rate during arc motion is as described in Feed Rate above. In the case of helical motion, the rate is applied along the helix. Beware as other interpretations are used on other systems.

11.1.8 Coolant

Flood coolant and mist coolant may each be turned on independently. They are turned off together.

11.1.9 Dwell

A machining system may be commanded to dwell (i.e., keep all axes unmoving) for a specific amount of time. The most common use of dwell is to break and clear chips or for a spindle to get up to speed.

11.1.10 Units

Units used for distances along the X, Y, and Z axes may be measured in millimetres or inches. Units for all other quantities involved in machine control cannot be changed. Different quantities use different specific units. Spindle speed is measured in revolutions per minute. The positions of rotational axes are measured in degrees. Feed rates are expressed in current length units per minute or in degrees per minute, as described above.

Warning: We advise you to check very carefully the system's response to changing units while tool and fixture offsets are loaded into the tables, while these offsets are active and/or while a part program is excecuting

11.1.11 Current Position

The controlled point is always at some location called the "current position" and Mach2 always knows where that is. The numbers representing the current position are adjusted in the absence of any axis motion if any of several events take place:

- Length units are changed (but see Warning above)
- Tool length offset is changed
- Co-ordinate system offsets are changed.

11.1.12 Selected Plane

There is always a "selected plane", which must be the XY-plane, the YZ-plane, or the XZ-plane of the machining system. The Z-axis is, of course, perpendicular to the XY-plane, the X-axis to the YZ-plane, and the Y-axis to the XZ-plane.

11.1.13 Tool Table

Zero or one tool is assigned to each slot in the tool table.

11.1.14 Tool Change

Mach2 allows you to implement a procedure for implementing automatic tool changes using macros or to change the tools by hand when required.

11.1.15 Pallet Shuttle

Mach2 allows you to implement a procedure for implementing pallet shuttle using macros.

11.1.16 Path Control Modes

The machining system may be put into any one of two path control modes: (1) exact stop mode, (2) constant velocity mode. In exact stop mode, the machine stops briefly at the end of each programmed move. In constant velocity mode, sharp corners of the path may be rounded slightly so that the feed rate may be kept up.

11.2 Interpreter Interaction with controls

11.2.1 Feed and Speed Override controls

Mach2 commands which enable (M48) or disable (M49) the feed and speed override switches. It is useful to be able to override these switches for some machining operations. The idea is that optimal settings have been included in the program, and the operator should not change them.

11.2.2 Block Delete control

If the block delete control is ON, lines of code which start with a slash (the block delete character) are not executed. If the switch is off, such lines are executed.

11.2.3 Optional Program Stop control

The optional program stop control (see Configure Logic) works as follows. If this control is ON and an input line contains an M1 code, program execution is stopped at the end on the commands on that line until the *Cycle Start* button is pushed.

11.3 Tool File

Mach2 maintains a tool file for each of the 256 tools which can be used.

Each data line of the file contains the data for one tool. This allows the definition of the tool length (Z axis), tool diameter (for Mill) and tool tip radius (for Lathe)

11.4 The language of Part Programs

11.4.1 Overview

The language is based on lines of code. Each line (also called a "block") may include commands to the machining system to do several different things. Lines of code may be collected in a file to make a program.

A typical line of code consists of an optional line number at the beginning followed by one or more "words." A word consists of a letter followed by a number (or something that evaluates to a number). A word may either give a command or provide an argument to a command. For example, `G1 X3` is a valid line of code with two words. "G1" is a command meaning "move in a straight line at the programmed feed rate," and "X3" provides an argument value (the value of X should be 3 at the end of the move). Most commands start with either G or M (for General and Miscellaneous). The words for these commands are called "G codes" and "M codes."

The language has two commands (M2 or M30), either of which ends a program. A program may end before the end of a file. Lines of a file that occur after the end of a program are not to be executed.

11.4.2 Parameters

A Mach2 machining system maintains an array of 10,320 numerical parameters. Many of them have specific uses. The parameter which are associated with fixtures are persistent over time. Other parameters will be undefined when Mach2 is loaded. The parameters are preserved when the interpreter is reset. The parameters with meanings defined by Mach2 are given in figure 11.1

11.4.3 Co-ordinate Systems

The machining system has an absolute co-ordinate system and 255 program co-ordinate (fixture) systems.

You can set the offsets of tools by `G10 L1 P~ X~ Z~.` The P word defines the tool number to be set.

You can set the offsets of the fixture systems using `G10 L2 P~ X~ Y~ Z~ A~ B~ C~` The P word defines the fixture to be set. The X, Y, Z etc words are the co-ordinates for the origin of for the axes in terms of the absolute co-ordinate system.

You can select one of the first seven fixtures by using G54, G55, G56, G57, G58, G59. Any of the 255 fixtures can be selected by `G59 P~` (e.g. `G59 P23` would select fixture 23). The absolute co-ordinate system can be selected by `G59 P0`.

Parameter number	Meaning	Parameter number	Meaning
5161	G28 home X	5281	Fixture 4 X
5162	G28 home Y	5282	Fixture 4 Y
5163	G28 home Z	5283	Fixture 4 Z
5164	G28 home A	5284	Fixture 4 A
5165	G28 home B	5285	Fixture 4 B
5166	G28 home C	5286	Fixture 4 C
5181	G30 home X	5301	Fixture 5 X
5182	G30 home Y	5302	Fixture 5 Y
5183	G30 home Z	5303	Fixture 5 Z
5184	G30 home A	5304	Fixture 5 A
5185	G30 home B	5305	Fixture 5 B
5186	G30 home C	5306	Fixture 5 C
5211	G92 offset X	5321	Fixture 6 X
5212	G92 offset Y	5322	Fixture 6 Y
5213	G92 offset Z	5323	Fixture 6 Z
5214	G92 offset A	5324	Fixture 6 A
5215	G92 offset B	5325	Fixture 6 B
5216	G92 offset C	5326	Fixture 6 C
5220	Current Fixture number		
5221	Fixture 1 X		*And so on every 20 values until*
5222	Fixture 1 Y		
5223	Fixture 1 Z		
5224	Fixture 1 A	10281	Fixture 254 X
5225	Fixture 1 B	10282	Fixture 254 Y
5226	Fixture 1 C	10283	Fixture 254 Z
5241	Fixture 2 X	10284	Fixture 254 A
5242	Fixture 2 Y	10285	Fixture 254 B
5243	Fixture 2 Z	10286	Fixture 254 C
5244	Fixture 2 A	10301	Fixture 255 X
5245	Fixture 2 B	10302	Fixture 255 Y
5246	Fixture 2 C	10303	Fixture 255 Z
5261	Fixture 3 X	10304	Fixture 255 A
5262	Fixture 3 Y	10305	Fixture 255 B
5263	Fixture 3 Z	10306	Fixture 255 C
5264	Fixture 3 A		
5265	Fixture 3 B		
5266	Fixture 3 C		

Figure 11.1 - System defined parameters

You can offset the current co-ordinate system using G92 or G92.3. This offset will then apply fixture co-ordinate systems. This offset may be cancelled with G92.1 or G92.2.

You can make straight moves in the absolute machine co-ordinate system by using G53 with either G0 or G1.

11.5 Format of a Line

A permissible line of input code consists of the following, in order, with the restriction that there is a maximum (currently 256) to the number of characters allowed on a line.

- an optional block delete character, which is a slash "/" .
- an optional line number.

- any number of words, parameter settings, and comments.
- an end of line marker (carriage return or line feed or both).

Any input not explicitly allowed is illegal and will cause the Interpreter to signal an error or to ignore the line.

Letter	Meaning
A	A-axis of machine
B	B-axis of machine
C	C-axis of machine
D	tool radius compensation number
F	feedrate
G	general function (see Table 5)
H	tool length offset index
I	X-axis offset for arcs X offset in G87 canned cycle
J	Y-axis offset for arcs Y offset in G87 canned cycle
K	Z-axis offset for arcs Z offset in G87 canned cycle
L	number of repetitions in canned cycles key used with G10
M	miscellaneous function (see Table 7)
N	line number
O	Subroutine label number
P	dwell time in canned cycles dwell time with G4 key used with G10
Q	feed increment in G83 canned cycle repetitions of subroutine call
R	arc radius canned cycle retract level
S	spindle speed
T	tool selection
X	X-axis of machine
Y	Y-axis of machine
Z	Z-axis of machine

Figure 11.2 - Word initial letters

Spaces and tabs are allowed anywhere on a line of code and do not change the meaning of the line, except inside comments. This makes some strange-looking input legal. For example, the line `g0x +0. 12 34y 7` is equivalent to `g0 x+0.1234 y7`

Blank lines are allowed in the input. They will be ignored.

Input is case insensitive, except in comments, i.e., any letter outside a comment may be in upper or lower case without changing the meaning of a line.

11.5.1 Line Number

A line number is the letter N followed by an integer (with no sign) between 0 and 99999 written with no more than five digits (000009 is not OK, for example). Line numbers may be repeated or used out of order, although normal practice is to avoid such usage. A line number is not required to be used (and this omission is common) but it must be in the proper place if it is used.

11.5.2 Subroutine labels

A subroutine label is the letter O followed by an integer (with no sign) between 0 and 99999 written with no more than five digits (000009 is not OK, for example). Subroutine labels may be used in any order but must be unique in a program although violation of this rule may not be flagged as an error. Nothing else except a comment should appear on the same line as a subroutine label.

11.5.3 Word

A word is a letter other than N or O followed by a real value.

Words may begin with any of the letters shown in figure 11.2. The table includes N and O for completeness, even though, as defined above, line numbers are not words. Several letters (I, J, K, L, P, R) may have different meanings in different contexts.

A real value is some collection of characters that can be processed to come up with a number. A real value may be an explicit number (such as 341 or -0.8807), a parameter value, an expression, or a unary operation value. Definitions of these follow immediately. Processing characters to come up with a number is called "evaluating". An explicit number evaluates to itself.

11.5.3.1 Number

The following rules are used for (explicit) numbers. In these rules a digit is a single character between 0 and 9.

- A number consists of (1) an optional plus or minus sign, followed by (2) zero to many digits, followed, possibly, by (3) one decimal point, followed by (4) zero to many digits - provided that there is at least one digit somewhere in the number.
- There are two kinds of numbers: integers and decimals. An integer does not have a decimal point in it; a decimal does.
- Numbers may have any number of digits, subject to the limitation on line length. Only about seventeen significant figures will be retained, however (enough for all known applications).
- A non-zero number with no sign as the first character is assumed to be positive.

Notice that initial (before the decimal point and the first non-zero digit) and trailing (after the decimal point and the last non-zero digit) zeros are allowed but not required. A number written with initial or trailing zeros will have the same value when it is read as if the extra zeros were not there.

Numbers used for specific purposes by Mach2 are often restricted to some finite set of values or some to some range of values. In many uses, decimal numbers must be close to integers; this includes the values of indexes (for parameters and carousel slot numbers, for example), M codes, and G codes multiplied by ten. A decimal number which is supposed be close to an integer is considered close enough if it is within 0.0001 of an integer.

11.5.3.2 Parameter Value

A parameter value is the hash character # followed by a real value. The real value must evaluate to an integer between 1 and 10320. The integer is a parameter number, and the value of the parameter value is whatever number is stored in the numbered parameter.

The # character takes precedence over other operations, so that, for example, `#1+2` means the number found by adding 2 to the value of parameter 1, not the value found in parameter 3. Of course, `#[1+2]` does mean the value found in parameter 3. The # character may be repeated; for example `##2` means the value of the parameter whose index is the (integer) value of parameter 2.

11.5.3.3 Expressions and Binary Operations

An expression is a set of characters starting with a left bracket [and ending with a balancing right bracket]. In between the brackets are numbers, parameter values, mathematical operations, and other expressions. An expression may be evaluated to produce a number. The expressions on a line are evaluated when the line is read, before anything on the line is executed. An example of an expression is:

```
[1+acos[0]-[#3**[4.0/2]]]
```

Binary operations appear only inside expressions. Nine binary operations are defined. There are four basic mathematical operations: addition (+), subtraction (-), multiplication (*), and division (/). There are three logical operations: non-exclusive or (OR), exclusive or (XOR), and logical and (AND). The eighth operation is the modulus operation (MOD). The ninth operation is the "power" operation (**) of raising the number on the left of the operation to the power on the right.

The binary operations are divided into three groups. The first group is: power. The second group is: multiplication, division, and modulus. The third group is: addition, subtraction, logical non-exclusive or, logical exclusive or, and logical and. If operations are strung together (for example in the expression `[2.0/3*1.5-5.5/11.0]`), operations in the first group are to be performed before operations in the second group and operations in the second group before operations in the third group. If an expression contains more than one operation from the same group (such as the first / and * in the example), the operation on the left is performed first. Thus, the example is equivalent to: `[((2.0/3)*1.5)-(5.5/11.0)]` which simplifies to `[1.0-0.5]` which is 0.5.

The logical operations and modulus are to be performed on any real numbers, not just on integers. The number zero is equivalent to logical false, and any non-zero number is equivalent to logical true.

11.5.3.4 Unary Operation Value

A unary operation value is either "ATAN" followed by one expression divided by another expression (for example `ATAN[2]/[1+3]`) or any other unary operation name followed by an expression (for example `SIN[90]`). The unary operations are: ABS (absolute value), ACOS (arc cosine), ASIN (arc sine), ATAN (arc tangent), COS (cosine), EXP (e raised to the given power), FIX (round down), FUP (round up), LN (natural logarithm), ROUND (round to the nearest whole number), SIN (sine), SQRT (square root), and TAN (tangent). Arguments to unary operations which take angle measures (COS, SIN, and TAN) are in degrees. Values returned by unary operations which return angle measures (ACOS, ASIN, and ATAN) are also in degrees.

The FIX operation rounds towards the left (less positive or more negative) on a number line, so that `FIX[2.8]=2` and `FIX[-2.8]=-3`, for example. The FUP operation rounds towards the right (more positive or less negative) on a number line; `FUP[2.8]=3` and `FUP[-2.8]=-2`, for example.

11.5.4 Parameter Setting

A parameter setting is the following four items one after the other:

- a pound character #
- a real value which evaluates to an integer between 1 and 10320,
- an equal sign = , and
- a real value. For example "#3 = 15" is a parameter setting meaning "set parameter 3 to 15."

A parameter setting does not take effect until after all parameter values on the same line have been found. For example, if parameter 3 has been previously set to 15 and the line `#3=6 G1 x#3` is interpreted, a straight move to a point where x equals 15 will occur and the value of parameter 3 will be 6.

11.5.5 Comments and Messages

Printable characters and white space inside parentheses is a comment. A left parenthesis always starts a comment. The comment ends at the first right parenthesis found thereafter.

Once a left parenthesis is placed on a line, a matching right parenthesis must appear before the end of the line. Comments may not be nested; it is an error if a left parenthesis is found after the start of a comment and before the end of the comment. Here is an example of a line containing a comment: `G80 M5 (stop motion)`

An alternative form of comment is to use the two characters `//` The remainder of the line is treated as a comment

Comments do not cause the machining system to do anything.

A comment contains a message if "MSG," appears after the left parenthesis and before any other printing characters. Variants of "MSG," which include white space and lower case characters are allowed. The rest of the characters before the right parenthesis are considered to be a message. Messages should be displayed on the message display device. Comments not containing messages need not be displayed there.

11.5.6 Item Repeats

A line may have any number of G words, but two G words from the same modal group may not appear on the same line.

A line may have zero to four M words. Two M words from the same modal group may not appear on the same line.

For all other legal letters, a line may have only one word beginning with that letter.

If a parameter setting of the same parameter is repeated on a line, `#3=15 #3=6`, for example, only the last setting will take effect. It is silly, but not illegal, to set the same parameter twice on the same line.

If more than one comment appears on a line, only the last one will be used; each of the other comments will be read and its format will be checked, but it will be ignored thereafter. It is expected that putting more than one comment on a line will be very rare.

11.5.7 Item order

The three types of item whose order may vary on a line (as given at the beginning of this section) are word, parameter setting, and comment. Imagine that these three types of item are divided into three groups by type.

The first group (the words) may be reordered in any way without changing the meaning of the line.

If the second group (the parameter settings) is reordered, there will be no change in the meaning of the line unless the same parameter is set more than once. In this case, only the last setting of the parameter will take effect. For example, after the line `#3=15 #3=6` has been interpreted, the value of parameter 3 will be 6. If the order is reversed to `#3=6 #3=15` and the line is interpreted, the value of parameter 3 will be 15.

If the third group (the comments) contains more than one comment and is reordered, only the last comment will be used.

If each group is kept in order or reordered without changing the meaning of the line, then the three groups may be interleaved in any way without changing the meaning of the line. For example, the line `g40 g1 #3=15 (so there!) #4=-7.0` has five items and means exactly the same thing in any of the 120 possible orders - such as `#4=-7.0 g1 #3=15 g40 (so there!)`- for the five items.

11.5.8 Commands and Machine Modes

Mach2 many commands cause a machining system to change from one mode to another, and the mode stays active until some other command changes it implicitly or explicitly. Such commands are called "modal". For example, if coolant is turned on, it stays on until it is explicitly turned off. The G codes for motion are also modal. If a G1 (straight move) command is given on one line, for example, it will be executed again on the next line if one

The modal groups for M codes are:

- group 4 = {M0, M1, M2, M30} stopping
- group 6 = {M6} tool change
- group 7 = {M3, M4, M5} spindle turning
- group 8 = {M7, M8, M9} coolant (special case: M7 and M8 may be active at the same time)
- group 9 = {M48, M49} enable/disable feed and speed override switches

In addition to the above modal groups, there is a group for non-modal G codes:

- group 0 = {G4, G10, G28, G30, G53, G92, G92.1, G92.2, G92.3}

Figure 11.3 - Modal groups

or more axis words is available on the line, unless an explicit command is given on that next line using the axis words or cancelling motion.

"Non-modal" codes have effect only on the lines on which they occur. For example, G4 (dwell) is non-modal.

11.6 Modal Groups

Modal commands are arranged in sets called "modal groups", and only one member of a modal group may be in force at any given time. In general, a modal group contains commands for which it is logically impossible for two members to be in effect at the same time - like measure in inches vs. measure in millimetres. A machining system may be in many modes at the same time, with one mode from each modal group being in effect. The modal groups are shown in figure 11.3.

For several modal groups, when a machining system is ready to accept commands, one member of the group must be in effect. There are default settings for these modal groups. When the machining system is turned on or otherwise re-initialized, the default values are automatically in effect.

Group 1, the first group on the table, is a group of G codes for motion. One of these is always in effect. That one is called the current motion mode.

It is an error to put a G-code from group 1 and a G-code from group 0 on the same line if both of them use axis words. If an axis word-using G-code from group 1 is implicitly in effect on a line (by having been activated on an earlier line), and a group 0 G-code that uses axis words appears on the line, the activity of the group 1 G-code is suspended for that line. The axis word-using G-codes from group 0 are G10, G28, G30, and G92.

Mach2 displays the current mode at the top of each screen.

11.7 G Codes

G codes of the Mach2 input language are shown in figure 11.4 and are the described in detail.

The descriptions contain command prototypes, set in `courier` type.

In the command prototypes, the tilde (~) stand for a real value. As described earlier, a real value may be (1) an explicit number, 4.4, for example, (2) an expression, [2+2.4], for example, (3) a parameter value, #88, for example, or (4) a unary function value, acos[0], for example.

In most cases, if axis words (any or all of `X~, Y~, Z~, A~, B~, C~`) are given, they specify a destination point. Axis numbers relate to the currently active co-ordinate system, unless explicitly described as being in the absolute co-ordinate system. Where axis words are optional, any omitted axes will have their current value. Any items in the command prototypes not explicitly described as optional are required. It is an error if a required item is omitted.

In the prototypes, the values following letters are often given as explicit numbers. Unless stated otherwise, the explicit numbers can be real values. For example, `G10 L2` could equally well be written `G[2*5] L[1+1]`. If the value of parameter 100 were 2, `G10 L#100` would also mean the same. Using real values which are not explicit numbers as just shown in the examples is rarely useful.

If `L~` is written in a prototype the "~" will often be referred to as the "L number". Similarly the "~" in `H~` may be called the "H number", and so on for any other letter.

If a scale factor is applied to any axis then it will be applied to the value of the corresponding X, Y, Z, A, B, C word and to the relevant I, J, K or R words when they are used.

11.7.1 Rapid Linear Motion - G0

For rapid linear motion, program `G0 X~ Y~ Z~ A~ B~ C~`, where all the axis words are optional, except that at least one must be used. The G0 is optional if the current motion mode is G0. This will produce co-ordinated linear motion to the destination point at the current traverse rate (or slower if the machine will not go that fast). It is expected that cutting will not take place when a G0 command is executing.

It is an error if:

- all axis words are omitted.

If cutter radius compensation is active, the motion will differ from the above; see Cutter Compensation. If G53 is programmed on the same line, the motion will also differ; see Absolute Co-ordinates.

11.7.2 Linear Motion at Feed Rate - G1

For linear motion at feed rate (for cutting or not), program `G1 X~ Y~ Z~ A~ B~ C~`, where all the axis words are optional, except that at least one must be used. The G1 is optional if the current motion mode is G1. This will produce co-ordinated linear motion to the destination point at the current feed rate (or slower if the machine will not go that fast).

It is an error if:

- all axis words are omitted.

If cutter radius compensation is active, the motion will differ from the above; see Cutter Compensation. If G53 is programmed on the same line, the motion will also differ; see Absolute Co-ordinates.

11.7.3 Arc at Feed Rate - G2 and G3

A circular or helical arc is specified using either G2 (clockwise arc) or G3 (counterclockwise arc). The axis of the circle or helix must be parallel to the X, Y, or Z-axis of the machine co-ordinate system. The axis (or, equivalently, the plane perpendicular to the axis) is selected with G17 (Z-axis, XY-plane), G18 (Y-axis, XZ-plane), or G19 (X-axis, YZ-plane). If the arc is circular, it lies in a plane parallel to the selected plane.

Summary of G-codes	
G0	Rapid positioning
G1	Linear interpolation
G2	Clockwise circular/helical interpolation
G3	Counterclockwise circular/Helical interpolation
G4	Dwell
G10	Co-ordinate system origin setting
G12	Clockwise circular pocket
G13	Counterclockwise circular pocket
G17	XY Plane select
G18	XZ plane select
G19	YZ plane select
G20	Inch unit
G21	Millimetre units
G28	Return home
G28.1	Reference axes
G30	Return home
G38.2	Straight probe
G40	Cancel cutter radius compensation
G41	Start cutter radius compensation left
G42	Start cutter radius compensation right
G43	Apply tool length offset (plus)
G49	Cancel tool length offset
G53	Move in absolute machine co-ordinate system
G54	Use fixture offset 1
G55	Use fixture offset 2
G56	Use fixture offset 3
G57	Use fixture offset 4
G58	Use fixture offset 5
G59	Use fixture offset 6 / use general fixture number
G61	Exact stop mode
G61.1	Exact stop mode
G64	Constant velocity mode
G80	Cancel motion mode (including canned cycles)
G81	Canned cycle - drilling
G82	Canned cycle - drilling with dwell
G83	Canned cycle - peck drilling
G84	Canned cycle - right hand rigid tapping
G85	Canned cycle - boring, no dwell, feed out
G86	Canned cycle - boring, spindle stop, rapid out
G87	Canned cycle - back boring
G88	Canned cycle - boring, spindle stop, manual out
G89	Canned cycle - boring, dwell, feed out
G90	Absolute distance mode
G91	Incremental distance mode
G92	Offset co-ordinates and set parameters
G92.1	Cancel G92 and set parameters to zero
G92.2	Cancel G92 and do not reset parameters
G92.3	Apply existing parameters to offset co-ordinate system
G93	Inverse time feed mode
G94	Feed per minute time mode
G98	Initial level return after canned cycles
G99	R-point level return after canned cycles

Figure 11.4 - Table of G codes

If a line of code makes an arc and includes rotational axis motion, the rotational axes turn at a constant rate so that the rotational motion starts and finishes when the XYZ motion starts and finishes. Lines of this sort are hardly ever programmed.

If cutter radius compensation is active, the motion will differ from the above; see Cutter Compensation.

Two formats are allowed for specifying an arc. We will call these the center format and the radius format. In both formats the G2 or G3 is optional if it is the current motion mode.

11.7.3.1 Radius Format Arc

In the radius format, the co-ordinates of the end point of the arc in the selected plane are specified along with the radius of the arc. Program `G2 X~ Y~ Z~ A~ B~ C~ R~` (or use G3 instead of G2). R is the radius. The axis words are all optional except that at least one of the two words for the axes in the selected plane must be used. The R number is the radius. A positive radius indicates that the arc turns through 180 degrees or less, while a negative radius indicates a turn of 180 degrees to 359.999 degrees. If the arc is helical, the value of the end point of the arc on the co-ordinate axis parallel to the axis of the helix is also specified.

It is an error if:

- ♦ both of the axis words for the axes of the selected plane are omitted,
- ♦ the end point of the arc is the same as the current point.

It is not good practice to program radius format arcs that are nearly full circles or are semicircles (or nearly semicircles) because a small change in the location of the end point will produce a much larger change in the location of the center of the circle (and, hence, the middle of the arc). The magnification effect is large enough that rounding error in a number can produce out-of-tolerance cuts. Nearly full circles are outrageously bad, semicircles (and nearly so) are only very bad. Other size arcs (in the range tiny to 165 degrees or 195 to 345 degrees) are OK.

Here is an example of a radius format command to mill an arc:

```
G17 G2 x 10 y 15 r 20 z 5.
```

That means to make a clockwise (as viewed from the positive Z-axis) circular or helical arc whose axis is parallel to the Z-axis, ending where X=10, Y=15, and Z=5, with a radius of 20. If the starting value of Z is 5, this is an arc of a circle parallel to the XY-plane; otherwise it is a helical arc.

11.7.3.2 Center Format Arc

In the center format, the co-ordinates of the end point of the arc in the selected plane are specified along with the offsets of the center of the arc from the current location. In this format, it is OK if the end point of the arc is the same as the current point. It is an error if:

- ♦ when the arc is projected on the selected plane, the distance from the current point to the center differs from the distance from the end point to the center by more than 0.0002 inch (if inches are being used) or 0.002 millimetre (if millimetres are being used).

The center is specified using the I and J words. There are two ways of interpreting them. The usual way is that I and J are the center relative to the current point at the start of the arc. This is sometimes called *Incremental IJ mode.* The second way is that I and J specify the center as actual co-ordinates in the current system. This is rather misleadingly called *Absolute IJ mode.* The IJ mode is set using the Configure>State… menu when Mach2 is set up. The choice of modes are to provide compatibility with commercial controllers. You will probably find Incremental to be best. In Absolute it will, of course usually be necessary to use both I and J words unless by chance the arc's centre is at the origin.

When the XY-plane is selected, program `G2 X~ Y~ Z~ A~ B~ C~ I~ J~` (or use G3 instead of G2). The axis words are all optional except that at least one of X and Y must be used. I and J are the offsets from the current location or coordinates - depending on IJ mode (X and Y directions, respectively) of the center of the circle. I and J are optional except that at least one of the two must be used. It is an error if:

- X and Y are both omitted,
- I and J are both omitted.

When the XZ-plane is selected, program `G2 X~ Y~ Z~ A~ B~ C~ I~ K~` (or use G3 instead of G2). The axis words are all optional except that at least one of X and Z must be used. I and K are the offsets from the current location or coordinates - depending on IJ mode (X and Z directions, respectively) of the center of the circle. I and K are optional except that at least one of the two must be used. It is an error if:

- X and Z are both omitted,
- I and K are both omitted.

When the YZ-plane is selected, program `G2 X~ Y~ Z~ A~ B~ C~ J~ K~` (or use G3 instead of G2). The axis words are all optional except that at least one of Y and Z must be used. J and K are the offsets from the current location or coordinates - depending on IJ mode (Y and Z directions, respectively) of the center of the circle. J and K are optional except that at least one of the two must be used. It is an error if:

- Y and Z are both omitted,
- J and K are both omitted.

Here is an example of a center format command to mill an arc in Incremental IJ mode:

```
 G17 G2 x10 y16 i3 j4 z9
```

That means to make a clockwise (as viewed from the positive z-axis) circular or helical arc whose axis is parallel to the Z-axis, ending where X=10, Y=16, and Z=9, with its center offset in the X direction by 3 units from the current X location and offset in the Y direction by 4 units from the current Y location. If the current location has X=7, Y=7 at the outset, the center will be at X=10, Y=11. If the starting value of Z is 9, this is a circular arc; otherwise it is a helical arc. The radius of this arc would be 5.

The above arc in Absolute IJ mode would be:

```
G17 G2 x10 y16 i10 j11 z9
```

In the center format, the radius of the arc is not specified, but it may be found easily as the distance from the center of the circle to either the current point or the end point of the arc.

11.7.4 Dwell - G4

For a dwell, program `G4 P~` . This will keep the axes unmoving for the period of time in seconds specified by the P number. It is an error if:

- the P number is negative.

11.7.5 Set Co-ordinate System Data Tool and Fixture tables - G10

See details of tool and fixture offsets for further information on co-ordinate systems

To set the offset values of a tool, program
`G10 L1 P~ X~ Z~` A~, where the P number must evaluate to an integer in the range 0 to 255 - the tool number - Offsets of the tool specified by the P number are reset to the given. The A number will reset the tool tip radius. Only those values for which an axis word is included on the line will be reset.. The Tool diameter cannot be set in this way.

To set the co-ordinate values for the origin of a fixture co-ordinate system, program
`G10 L2 P~ X~ Y~ Z~ A~ B~ C~`, where the P number must evaluate to an integer in the range 1 to 255 - the fixture number - (Values 1 to 6 corresponding to G54 to G59) and all axis words are optional. The co-ordinates of the origin of the co-ordinate system specified by the P number are reset to the co-ordinate values given (in terms of the absolute co-ordinate system). Only those co-ordinates for which an axis word is included on the line will be reset.

It is an error if:

- ♦ the P number does not evaluate to an integer in the range 0 to 255.

If origin offsets (made by G92 or G92.3) were in effect before G10 is used, they will continue to be in effect afterwards.

The co-ordinate system whose origin is set by a G10 command may be active or inactive at the time the G10 is executed.

The values set will not be persistent unless the tool or fixture tables are saved using the buttons on Tables screen.

Example: `G10 L2 P1 x3.5 y17.2` sets the origin of the first co-ordinate system (the one selected by G54) to a point where X is 3.5 and Y is 17.2 (in absolute co-ordinates). The Z co-ordinate of the origin (and the co-ordinates for any rotational axes) are whatever those co-ordinates of the origin were before the line was executed.

11.7.6 Clockwise/counterclockwise circular pocket - G12 and G13

These circular pocket commands are a sort of canned cycle which can be used to produce a circular hole larger than the tool in use or with a suitable tool (like a woodruff key cutter) to cut internal grooves for "O" rings etc.

Program `G12 I~` for a clockwise move and `G13 I~` for a counterclockwise move.

The tool is moved in the X direction by the value if the I word and a circle cut in the direction specified with the original X and Y co-ordinates as the centre. The tool is returned to the centre.

Its effect is undefined if the current plane is not XY.

11.7.7 Plane Selection - G17, G18, and G19

Program G17 to select the XY-plane, G18 to select the XZ-plane, or G19 to select the YZ-plane. The effects of having a plane selected are discussed in under G2/3 and Canned cycles

11.7.8 Length Units - G20 and G21

Program G20 to use inches for length units. Program G21 to use millimetres.

It is usually a good idea to program either G20 or G21 near the beginning of a program before any motion occurs, and not to use either one anywhere else in the program. It is the responsibility of the user to be sure all numbers are appropriate for use with the current length units.

11.7.9 Return to Home - G28 and G30

A home position is defined (by parameters 5161-5166). The parameter values are in terms of the absolute co-ordinate system, but are in unspecified length units.

To return to home position by way of the programmed position, program `G28 X~ Y~ Z~ A~ B~ C~` (or use G30). All axis words are optional. The path is made by a traverse move from the current position to the programmed position, followed by a traverse move to the home position. If no axis words are programmed, the intermediate point is the current point, so only one move is made.

11.7.10 Reference axes G28.1

Program `G28.1 X~ Y~ Z~ A~ B~ C~` to reference the given axes. The axes will move at the current feed rate towards the home switch(es), as defined by the Configuration. When the absolute machine coordinate reaches the value given by an axis word then the feed rate is set to that defined by Configure>Config Referencing. Provided the current absolute position is approximately correct, then this will give a soft stop onto the reference switch(es).

11.7.11 Straight Probe - G38.2

11.7.11.1 The Straight Probe Command

Program `G38.2 X~ Y~ Z~ A~ B~ C~` to perform a straight probe operation. The rotational axis words are allowed, but it is better to omit them. If rotational axis words are used, the numbers must be the same as the current position numbers so that the rotational axes do not move. The linear axis words are optional, except that at least one of them must be used. The tool in the spindle must be a probe.

It is an error if:

- ♦ the current point is less than 0.254 millimetre or 0.01 inch from the programmed point.
- ♦ G38.2 is used in inverse time feed rate mode,
- ♦ any rotational axis is commanded to move,
- ♦ no X, Y, or Z-axis word is used.

In response to this command, the machine moves the controlled point (which should be at the end of the probe tip) in a straight line at the current feed rate toward the programmed point. If the probe trips, the probe is retracted slightly from the trip point at the end of command execution. If the probe does not trip even after overshooting the programmed point slightly, an error is signalled.

After successful probing, parameters 5061 to 5066 will be set to the co-ordinates of the location of the controlled point at the time the probe tripped.

11.7.11.2 Using the Straight Probe Command

Using the straight probe command, if the probe shank is kept nominally parallel to the Z-axis (i.e., any rotational axes are at zero) and the tool length offset for the probe is used, so that the controlled point is at the end of the tip of the probe:

- ♦ without additional knowledge about the probe, the parallelism of a face of a part to the XY-plane may, for example, be found.
- ♦ if the probe tip radius is known approximately, the parallelism of a face of a part to the YZ or XZ-plane may, for example, be found.
- ♦ if the shank of the probe is known to be well-aligned with the Z-axis and the probe tip radius is known approximately, the center of a circular hole, may, for example, be found.
- ♦ if the shank of the probe is known to be well-aligned with the Z-axis and the probe tip radius is known precisely, more uses may be made of the straight probe command, such as finding the diameter of a circular hole.

If the straightness of the probe shank cannot be adjusted to high accuracy, it is desirable to know the effective radii of the probe tip in at least the +X, -X, +Y, and -Y directions. These quantities can be stored in parameters either by being included in the parameter file or by being set in a Mach2 program.

Using the probe with rotational axes not set to zero is also feasible. Doing so is more complex than when rotational axes are at zero, and we do not deal with it here.

11.7.11.3 Example Code

As a usable example, the code for finding the center and diameter of a circular hole is shown in figure 11.5. For this code to yield accurate results, the probe shank must be well-aligned with the Z-axis, the cross section of the probe tip at its widest point must be very circular, and the probe tip radius (i.e., the radius of the circular cross section) must be known precisely. If the probe tip radius is known only approximately (but the other conditions hold), the location of the hole center will still be accurate, but the hole diameter will not.

In figure 11.5 an entry of the form <description of number> is meant to be replaced by an actual number that matches the description of number. After this section of code has executed, the X-value of the center will be in parameter 1041, the Y-value of the center in parameter 1022, and the diameter in parameter 1034. In addition, the diameter parallel to the X-axis will be in parameter 1024, the diameter parallel to the Y-axis in parameter 1014, and the difference (an indicator of circularity) in parameter 1035. The probe tip will be in the hole at the XY center of the hole.

The example does not include a tool change to put a probe in the spindle. Add the tool change code at the beginning, if needed.

```
N010 (probe to find center and diameter of circular hole)
N020 (This program will not run as given here. You have to)
N030 (insert numbers in place of <description of number>.)
N040 (Delete lines N020, N030, and N040 when you do that.)
N050 G0 Z <Z-value of retracted position> F <feed rate>
N060 #1001=<nominal X-value of hole center>
N070 #1002=<nominal Y-value of hole center>
N080 #1003=<some Z-value inside the hole>
N090 #1004=<probe tip radius>
N100 #1005=[<nominal hole diameter>/2.0 - #1004]
N110 G0 X#1001 Y#1002 (move above nominal hole center)
N120 G0 Z#1003 (move into hole - to be cautious, substitute G1 for G0 here)
N130 G38.2 X[#1001 + #1005] (probe +X side of hole)
N140 #1011=#5061 (save results)
N150 G0 X#1001 Y#1002 (back to center of hole)
N160 G38.2 X[#1001 - #1005] (probe -X side of hole)
N170 #1021=[[#1011 + #5061] / 2.0] (find pretty good X-value of hole center)

N180 G0 X#1021 Y#1002 (back to center of hole)
N190 G38.2 Y[#1002 + #1005] (probe +Y side of hole)
N200 #1012=#5062 (save results)
N210 G0 X#1021 Y#1002 (back to center of hole)
N220 G38.2 Y[#1002 - #1005] (probe -Y side of hole)
N230 #1022=[[#1012 + #5062] / 2.0] (find very good Y-value of hole center)
N240 #1014=[#1012 - #5062 + [2 * #1004]] (find hole diameter in Y-direction)

N250 G0 X#1021 Y#1022 (back to center of hole)
N260 G38.2 X[#1021 + #1005] (probe +X side of hole)
N270 #1031=#5061 (save results)
N280 G0 X#1021 Y#1022 (back to center of hole)
N290 G38.2 X[#1021 - #1005] (probe -X side of hole)
N300 #1041=[[#1031 + #5061] / 2.0] (find very good X-value of hole center)
N310 #1024=[#1031 - #5061 + [2 * #1004]] (find hole diameter in X-direction)

N320 #1034=[[#1014 + #1024] / 2.0] (find average hole diameter)
N330 #1035=[#1024 - #1014] (find difference in hole diameters)
N340 G0 X#1041 Y#1022 (back to center of hole)
N350 M2 (that's all, folks)
```

Figure 11.5 - Code to Probe Hole

11.7.12 Cutter Radius Compensation - G40, G41, and G42

To turn cutter radius compensation off, program `G40`. It is OK to turn compensation off when it is already off.

Cutter radius compensation may be performed only if the XY-plane is active.

To turn cutter radius compensation on left (i.e., the cutter stays to the left of the programmed path when the tool radius is positive), program `G41 D~` To turn cutter radius compensation on right (i.e., the cutter stays to the right of the programmed path when the tool radius is positive), program `G42 D~` The D word is optional; if there is no D word, the radius of the tool currently in the spindle will be used. If used, the D number should normally be the slot number of the tool in the spindle, although this is not required. It is OK for the D number to be zero; a radius value of zero will be used.

It is an error if:

- the D number is not an integer, is negative or is larger than the number of carousel slots,

- ♦ the XY-plane is not active,
- ♦ cutter radius compensation is commanded to turn on when it is already on.

The behavior of the machining system when cutter radius compensation is on is described in the chapter of Cutter Compensation.

11.7.13 Tool Length Offsets - G43 and G49

To use a tool length offset, program `G43 H~`, where the H number is the desired index in the tool table. It is expected that all entries in this table will be positive. The H number should be, but does not have to be, the same as the slot number of the tool currently in the spindle. It is OK for the H number to be zero; an offset value of zero will be used.

It is an error if:

- ♦ the H number is not an integer, is negative, or is larger than the number of carousel slots.

To use no tool length offset, program `G49`

It is OK to program using the same offset already in use. It is also OK to program using no tool length offset if none is currently being used.

11.7.14 Move in Absolute Co-ordinates - G53

For linear motion to a point expressed in absolute co-ordinates, program `G1 G53 X~ Y~ Z~ A~ B~ C~` (or similarly with G0 instead of G1), where all the axis words are optional, except that at least one must be used. The G0 or G1 is optional if it is the current motion mode. G53 is not modal and must be programmed on each line on which it is intended to be active. This will produce co-ordinated linear motion to the programmed point. If G1 is active, the speed of motion is the current feed rate (or slower if the machine will not go that fast). If G0 is active, the speed of motion is the current traverse rate (or slower if the machine will not go that fast).

It is an error if:

- ♦ G53 is used without G0 or G1 being active,
- ♦ G53 is used while cutter radius compensation is on.

See relevant chapter for an overview of co-ordinate systems.

11.7.15 Select Fixture Co-ordinate System - G54 to G59 & G59 P~

To select fixture co-ordinate system 1, program `G54`, and similarly for the first six fixtures. The system-number-G-code pairs are: (1-G54), (2-G55), (3-G56), (4-G57), (5-G58), (6-G59)

To access any of the 255 fixtures (1 - 255) program `G59 P~` where the P word gives the required fixture number. Thus G59 P5 is identical in effect to G58. If no fixture offset is required then program `G59 P0`

It is an error if:

- ♦ one of these G-codes is used while cutter radius compensation is on.

See relevant chapter for an overview of co-ordinate systems.

11.7.16 Set Path Control Mode - G61, G61.1, and G64

Program `G61` or `G61.1` to put the machining system into exact stop mode, or `G64` for constant velocity mode. It is OK to program for the mode that is already active. These modes are described in detail above.

11.7.17 Cancel Modal Motion - G80

Program `G80` to ensure no axis motion will occur. It is an error if:

- ♦ Axis words are programmed when G80 is active, unless a modal group 0 G code is programmed which uses axis words.

11.7.18 Canned Cycles - G81 to G89

The canned cycles G81 through G89 have been implemented as described in this section. Two examples are given with the description of G81 below.

All canned cycles are performed with respect to the currently selected plane. Any of the three planes (XY, YZ, ZX) may be selected. Throughout this section, most of the descriptions assume the XY-plane has been selected. The behavior is always analogous if the YZ or XZ-plane is selected.

Rotational axis words are allowed in canned cycles, but it is better to omit them. If rotational axis words

are used, the numbers must be the same as the current position numbers so that the rotational axes do not move.

All canned cycles use X, Y, R, and Z numbers in the NC code. These numbers are used to determine X, Y, R, and Z positions. The R (usually meaning retract) position is along the axis perpendicular to the currently selected plane (Z-axis for XY-plane, X-axis for YZ-plane, Y-axis for XZ-plane). Some canned cycles use additional arguments.

For canned cycles, we will call a number "sticky" if, when the same cycle is used on several lines of code in a row, the number must be used the first time, but is optional on the rest of the lines. Sticky numbers keep their value on the rest of the lines if they are not explicitly programmed to be different. The R number is always sticky.

In incremental distance mode: when the XY-plane is selected, X, Y, and R numbers are treated as increments to the current position and Z as an increment from the Z-axis position before the move involving Z takes place; when the YZ or XZ-plane is selected, treatment of the axis words is analogous. In absolute distance mode, the X, Y, R, and Z numbers are absolute positions in the current co-ordinate system.

The L number is optional and represents the number of repeats. L=0 is not allowed. If the repeat feature is used, it is normally used in incremental distance mode, so that the same sequence of motions is repeated in several equally spaced places along a straight line. In absolute distance mode, L > 1 means "do the same cycle in the same place several times," Omitting the L word is equivalent to specifying L=1. The L number is not sticky.

When L>1 in incremental mode with the XY-plane selected, the X and Y positions are determined by adding the given X and Y numbers either to the current X and Y positions (on the first go-around) or to the X and Y positions at the end of the previous go-around (on the repetitions). The R and Z positions do not change during the repeats.

The height of the retract move at the end of each repeat (called "clear Z" in the descriptions below) is determined by the setting of the retract mode: either to the original Z position (if that is above the R position and the retract mode is G98), or otherwise to the R position.

It is an error if:

- ♦ X, Y, and Z words are all missing during a canned cycle,
- ♦ a P number is required and a negative P number is used,
- ♦ an L number is used that does not evaluate to a positive integer,
- ♦ rotational axis motion is used during a canned cycle,
- ♦ inverse time feed rate is active during a canned cycle,
- ♦ cutter radius compensation is active during a canned cycle.

When the XY plane is active, the Z number is sticky, and it is an error if:

- ♦ the Z number is missing and the same canned cycle was not already active,
- ♦ the R number is less than the Z number.

When the XZ plane is active, the Y number is sticky, and it is an error if:

- he Y number is missing and the same canned cycle was not already active,
- the R number is less than the Y number.

When the YZ plane is active, the X number is sticky, and it is an error if:

- the X number is missing and the same canned cycle was not already active,
- the R number is less than the X number.

11.7.18.1 Preliminary and In-Between Motion

At the very beginning of the execution of any of the canned cycles, with the XY-plane selected, if the current Z position is below the R position, the Z-axis is traversed to the R position. This happens only once, regardless of the value of L.

In addition, at the beginning of the first cycle and each repeat, the following one or two moves are made:

- a straight traverse parallel to the XY-plane to the given XY-position,
- a straight traverse of the Z-axis only to the R position, if it is not already at the R position.

If the XZ or YZ plane is active, the preliminary and in-between motions are analogous.

11.7.18.2 G81 Cycle

The G81 cycle is intended for drilling. Program `G81 X~ Y~ Z~ A~ B~ C~ R~ L~`

- Preliminary motion, as described above.
- Move the Z-axis only at the current feed rate to the Z position.
- Retract the Z-axis at traverse rate to clear Z.

Example 1. Suppose the current position is (1, 2, 3) and the XY-plane has been selected, and the following line of NC code is interpreted.

```
G90 G81 G98 X4 Y5 Z1.5 R2.8
```

This calls for absolute distance mode (G90), old "Z" retract mode (G98) and calls for the G81 drilling cycle to be performed once. The X number and X position are 4. The Y number and Y position are 5. The Z number and Z position are 1.5. The R number and clear Z are 2.8. The following moves take place.

- a traverse parallel to the XY-plane to (4,5,3)
- a traverse parallel to the Z-axis to (4,5,2.8)
- a feed parallel to the Z-axis to (4,5,1.5)
- a traverse parallel to the Z-axis to (4,5,3)

Example 2. Suppose the current position is (1, 2, 3) and the XY-plane has been selected, and the following line of NC code is interpreted.

```
G91 G81 G98 X4 Y5 Z-0.6 R1.8 L3
```

This calls for incremental distance mode (G91), old "Z" retract mode and calls for the G81 drilling cycle to be repeated three times. The X number is 4, the Y number is 5, the Z number is -0.6 and the R number is 1.8. The initial X position is 5 (=1+4), the initial Y position is 7 (=2+5), the clear Z position is 4.8 (=1.8+3), and the Z position is 4.2 (=4.8-0.6). Old Z is 3.0

The first move is a traverse along the Z-axis to (1,2,4.8), since old Z < clear Z.

The first repeat consists of 3 moves.

- a traverse parallel to the XY-plane to (5,7,4.8)
- a feed parallel to the Z-axis to (5,7, 4.2)
- a traverse parallel to the Z-axis to (5,7,4.8)

The second repeat consists of 3 moves. The X position is reset to 9 (=5+4) and the Y position to 12 (=7+5).

- a traverse parallel to the XY-plane to (9,12,4.8)
- a feed parallel to the Z-axis to (9,12, 4.2)
- a traverse parallel to the Z-axis to (9,12,4.8)

The third repeat consists of 3 moves. The X position is reset to 13 (=9+4) and the Y position to 17 (=12+5).

- a traverse parallel to the XY-plane to (13,17,4.8)
- a feed parallel to the Z-axis to (13,17, 4.2)
- a traverse parallel to the Z-axis to (13,17,4.8)

11.7.18.3 G82 Cycle

The G82 cycle is intended for drilling. Program

```
G82 X~ Y~ Z~ A~ B~ C~ R~ L~ P~
```

- Preliminary motion, as described above.
- Move the Z-axis only at the current feed rate to the Z position.
- Dwell for the P number of seconds.
- Retract the Z-axis at traverse rate to clear Z.

11.7.18.4 G83 Cycle

The G83 cycle (often called peck drilling) is intended for deep drilling or milling with chip breaking. The retracts in this cycle clear the hole of chips and cut off any long stringers (which are common when drilling in aluminum). This cycle takes a Q number which represents a "delta" increment along the Z-axis. Program

```
G83 X~ Y~ Z~ A~ B~ C~ R~ L~ Q~
```

- Preliminary motion, as described above.
- Move the Z-axis only at the current feed rate downward by delta or to the Z position, whichever is less deep.
- Rapid back out to the clear Z.
- Rapid back down to the current hole bottom, backed off a bit.
- Repeat steps 1, 2, and 3 until the Z position is reached at step 1.
- Retract the Z-axis at traverse rate to clear Z.

It is an error if:

  - the Q number is negative or zero.

11.7.18.5 G84 Cycle

The G84 cycle is intended for right-hand tapping with a tap tool. Program

```
G84 X~ Y~ Z~ A~ B~ C~ R~ L~
```

- Preliminary motion, as described above.
- Start speed-feed synchronization.
- Move the Z-axis only at the current feed rate to the Z position.
- Stop the spindle.
- Start the spindle counterclockwise.
- Retract the Z-axis at the current feed rate to clear Z.
- If speed-feed synch was not on before the cycle started, stop it.
- Stop the spindle.

- Start the spindle clockwise.

The spindle must be turning clockwise before this cycle is used. It is an error if:

- the spindle is not turning clockwise before this cycle is executed.

With this cycle, the programmer must be sure to program the speed and feed in the correct proportion to match the pitch of threads being made. The relationship is that the spindle speed equals the feed rate times the pitch (in threads per length unit). For example, if the pitch is 2 threads per millimetre, the active length units are millimetres, and the feed rate has been set with the command F150, then the speed should be set with the command S300, since 150 x 2 = 300.

If the feed and speed override switches are enabled and not set at 100%, the one set at the lower setting will take effect. The speed and feed rates will still be synchronized.

11.7.18.6 G85 Cycle

The G85 cycle is intended for boring or reaming, but could be used for drilling or milling. Program `G85 X~ Y~ Z~ A~ B~ C~ R~ L~`

- Preliminary motion, as described above.
- Move the Z-axis only at the current feed rate to the Z position.
- Retract the Z-axis at the current feed rate to clear Z.

11.7.18.7 G86 Cycle

The G86 cycle is intended for boring. This cycle uses a P number for the number of seconds to dwell. Program `G86 X~ Y~ Z~ A~ B~ C~ R~ L~ P~`

- Preliminary motion, as described above.
- Move the Z-axis only at the current feed rate to the Z position.
- Dwell for the P number of seconds.
- Stop the spindle turning.
- Retract the Z-axis at traverse rate to clear Z.
- Restart the spindle in the direction it was going.

The spindle must be turning before this cycle is used. It is an error if:

- the spindle is not turning before this cycle is executed.

11.7.18.8 G87 Cycle

The G87 cycle is intended for back boring. Program

```
G87 X~ Y~ Z~ A~ B~ C~ R~ L~ I~ J~ K~
```

The situation, as shown in Figure 11.6 is that you have a through hole and you want to counterbore the bottom of hole. To do this you put an L-shaped tool in the spindle with a cutting surface on the UPPER side of its base. You stick it carefully through the hole when it is not spinning and is oriented so it fits through the hole, then you move it so the stem of the L is on the axis of the hole, start the spindle, and feed the tool upward to make the counterbore. Then you stop the tool, get it out of the hole, and restart it.

This cycle uses I and J numbers to indicate the position for inserting and removing the tool. I and J will always be increments from the X position and the Y position, regardless of the distance mode setting. This cycle also uses a K number to specify the position along the Z-axis of the controlled point top of the counterbore. The K number is a Z-value in the current co-ordinate system in absolute distance mode, and an increment (from the Z position) in incremental distance mode.

- Preliminary motion, as described above.
- Move at traverse rate parallel to the XY-plane to the point indicated by I and J.
- Stop the spindle in a specific orientation.

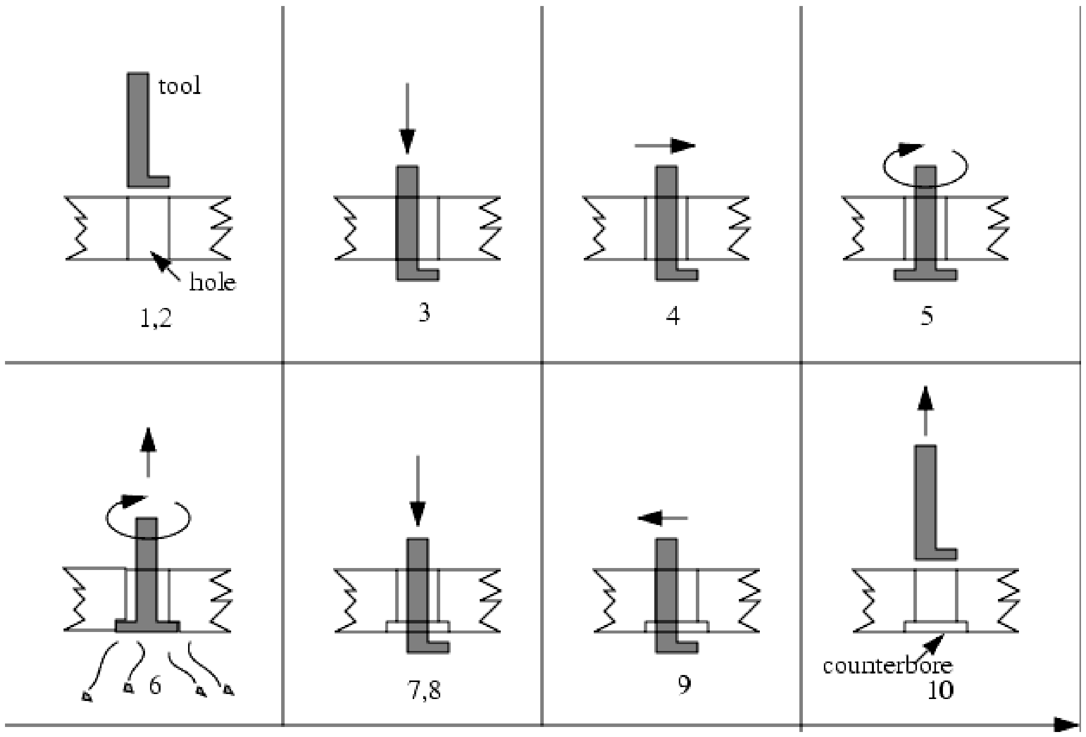


Figure 11.6 - G87 back boring sequence

- ♦ Move the Z-axis only at traverse rate downward to the Z position.
- ♦ Move at traverse rate parallel to the XY-plane to the X,Y location.
- ♦ Start the spindle in the direction it was going before.
- ♦ Move the Z-axis only at the given feed rate upward to the position indicated by K.
- ♦ Move the Z-axis only at the given feed rate back down to the Z position.
- ♦ Stop the spindle in the same orientation as before.
- ♦ Move at traverse rate parallel to the XY-plane to the point indicated by I and J.
- ♦ Move the Z-axis only at traverse rate to the clear Z.
- ♦ Move at traverse rate parallel to the XY-plane to the specified X,Y location.
- ♦ Restart the spindle in the direction it was going before.

When programming this cycle, the I and J numbers must be chosen so that when the tool is stopped in an oriented position, it will fit through the hole. Because different cutters are made differently, it may take some analysis and/or experimentation to determine appropriate values for I and J.

11.7.18.9 G88 Cycle

The G88 cycle is intended for boring. This cycle uses a P word, where P specifies the number of seconds to dwell. Program `G88 X~ Y~ Z~ A- B~ C~ R~~ L~ P~`

- ♦ Preliminary motion, as described above.
- ♦ Move the Z-axis only at the current feed rate to the Z position.
- ♦ Dwell for the P number of seconds.
- ♦ Stop the spindle turning.
- ♦ Stop the program so the operator can retract the spindle manually.
- ♦ Restart the spindle in the direction it was going.

11.7.18.10 G89 Cycle

The G89 cycle is intended for boring. This cycle uses a P number, where P specifies the number of seconds to dwell. program `G89 X~ Y~ Z~ A~ B~ C~ R~ L~ P~`

- ♦ Preliminary motion, as described above.
- ♦ Move the Z-axis only at the current feed rate to the Z position.
- ♦ Dwell for the P number of seconds.
- ♦ Retract the Z-axis at the current feed rate to clear Z.

11.7.19 Set Distance Mode - G90 and G91

Interpretation of Mach2 code can be in one of two distance modes: absolute or incremental.

To go into absolute distance mode, program `G90`. In absolute distance mode, axis numbers (X, Y, Z, A, B, C) usually represent positions in terms of the currently active co-ordinate system. Any exceptions to that rule are described explicitly in this section describing G-codes.

To go into incremental distance mode, program `G91`. In incremental distance mode, axis numbers (X, Y, Z, A, B, C) usually represent increments from the current values of the numbers.

I and J numbers always represent increments, regardless of the distance mode setting. K numbers represent increments in all but one usage (the G87 boring cycle), where the meaning changes with distance mode.

11.7.20 Co-ordinate System or Workpiece Offsets - G92, G92.1, G92.2, G92.3

See the chapter on co-ordinate systems for full details.

To make the current point have the co-ordinates you want (without motion), program `G92 X~ Y~ Z~ A~ B~ C~`, where the axis words contain the axis numbers you want. All axis words are optional, except that at least one must be used. If an axis word is not used for a given axis, the co-ordinate on that axis of the current point is not changed. It is an error if:

- ♦ all axis words are omitted.

When G92 is executed, the origin of the currently active co-ordinate system moves. To do this, origin offsets are calculated so that the co-ordinates of the current point with respect to the moved origin are as specified on the line containing the G92. In addition, parameters 5211 to 5216 are set to the X, Y, Z, A, B, and C-axis offsets. The offset for an axis is the amount the origin must be moved so that the co-ordinate of the controlled point on the axis has the specified value.

Here is an example. Suppose the current point is at X=4 in the currently specified co-ordinate system and the current X-axis offset is zero, then `G92 X7` sets the X-axis offset to -3, sets parameter 5211 to -3, and causes the X-co-ordinate of the current point to be 7.

The axis offsets are always used when motion is specified in absolute distance mode using any of the fixture co-ordinate systems Thus all fixture co-ordinate systems are affected by G92.

Being in incremental distance mode has no effect on the action of G92.

Non-zero offsets may be already be in effect when the G92 is called. They are in effect discarded before the new value is applied. Mathematically the new value of each offset is A+B, where A is what the offset would be if the old offset were zero, and B is the old offset. For example, after the previous example, the X-value of the current point is 7. If `G92 X9` is then programmed, the new X-axis offset is -5, which is calculated by [[7-9] + -3]. Put another way the `G92 X9` produces the same offset whatever G92 offset was already in place.

To reset axis offsets to zero, program `G92.1` or `G92.2` G92.1 sets parameters 5211 to 5216 to zero, whereas G92.2 leaves their current values alone.

To set the axis offset values to the values given in parameters 5211 to 5216, program `G92.3`

You can set axis offsets in one program and use the same offsets in another program. Program G92 in the first program. This will set parameters 5211 to 5216. Do not use G92.1 in the remainder of the first program. The parameter values will be saved when the first program exits and restored when the second one starts up. Use G92.3 near the beginning of the second program. That will restore the offsets saved in the first program.

11.7.21 Set Feed Rate Mode - G93 and G94

Two feed rate modes are recognized: units per minute and inverse time. Program `G94` to start the units per minute mode. Program `G93` to start the inverse time mode.

In units per minute feed rate mode, an F word on the line is interpreted to mean the controlled point should move at a certain number of inches per minute, millimetres per minute, or degrees per minute, depending upon what length units are being used and which axis or axes are moving.

In inverse time feed rate mode, an F word means the move should be completed in [one divided by the F number] minutes. For example, if the F number is 2.0, the move should be completed in half a minute.

When the inverse time feed rate mode is active, an F word must appear on every line which has a G1, G2, or G3 motion, and an F word on a line that does not have G1, G2, or G3 is ignored. Being in inverse time feed rate mode does not affect G0 (rapid traverse) motions. It is an error if:

- ♦ inverse time feed rate mode is active and a line with G1, G2, or G3 (explicitly or implicitly) does not have an F word.

11.7.22 Set Canned Cycle Return Level - G98 and G99

When the spindle retracts during canned cycles, there is a choice of how far it retracts:

1. retract perpendicular to the selected plane to the position indicated by the R word, or
2. retract perpendicular to the selected plane to the position that axis was in just before the canned cycle started (unless that position is lower than the position indicated by the R word, in which case use the R word position).

To use option (1), program `G99` To use option (2), program `G98` Remember that the R word has different meanings in absolute distance mode and incremental distance mode.

11.8 Built-in M Codes

M codes interpreted directly by Mach2 are shown in figure 11.7.

M-code	Meaning
M0	Program stop
M1	Optional program stop
M2	Program end
M3	Rotate spindle clockwise
M4	Rotate spindle counterclockwise
M5	Stop spindle rotation
M6	Tool change (by two macros)
M7	Mist coolant on
M8	Flood coolant on
M9	All coolant off
M30	Program end and Rewind
M48	Enable speed and feed override
M49	Disable speed and feed override
M98	Call subroutine
M99	Return from subroutine

Figure 11.7 - Built in M-codes

11.8.1 Program Stopping and Ending - M0, M1, M2, M30

To stop a running program temporarily (regardless of the setting of the optional stop switch), program `M0`.

To stop a running program temporarily (but only if the optional stop switch is on), program `M1`.

It is OK to program M0 and M1 in MDI mode, but the effect will probably not be noticeable, because normal behavior in MDI mode is to stop after each line of input, anyway.

If a program is stopped by an M0, M1, pressing the cycle start button will restart the program at the following line.

To end a program, program `M2` or `M30`. Both of these commands have the following effects.

- ♦ Axis offsets are set to zero (like G92.2) and origin offsets are set to the default (like G54).
- ♦ Selected plane is set to XY (like G17).
- ♦ Distance mode is set to absolute (like G90).
- ♦ Feed rate mode is set to Units per minute mode (like G94).
- ♦ Feed and speed overrides are set to ON (like M48).
- ♦ Cutter compensation is turned off (like G40).
- ♦ The spindle is stopped (like M5).
- ♦ The current motion mode is set to G1 (like G1).
- ♦ Coolant is turned off (like M9).

No more lines of code in the file will be executed after the M2 or M30 command is executed. Pressing cycle start will start the program back at the beginning of the file.

11.8.2 Spindle Control - M3, M4, M5

To start the spindle turning clockwise at the currently programmed speed, program `M3`.

To start the spindle turning counterclockwise at the currently programmed speed, program `M4`.

To stop the spindle from turning, program `M5`.

It is OK to use M3 or M4 if the spindle speed is set to zero. If this is done (or if the speed override switch is enabled and set to zero), the spindle will not start turning. If, later, the spindle speed is set above zero (or the override switch is turned up), the spindle will start turning. It is OK to use M3 or M4 when the spindle is already turning or to use M5 when the spindle is already stopped.

11.8.3 Tool change - M6

Provided tool change requests are not to be ignored (as defined in Configure>Logic), Mach2 will call a macro (q.v) M6Start when the command is encountered. It will then wait for Cycle Start to be pressed, execute the macro M6End and continue running the part program. You can provide Visual Basic code in the macros to operate your own mechanical tool changer if you wish.

If tool change requests configured to be ignored then M6 has no effect.

11.8.4 Coolant Control - M7, M8, M9

To turn mist coolant on, program `M7`.

To turn flood coolant on, program `M8`.

To turn all coolant off, program `M9`.

It is always OK to use any of these commands, regardless of what coolant is on or off.

11.8.5 Override Control - M48 and M49

To enable the speed and feed override, program `M48`. To disable both overrides, program `M49`. It is OK to enable or disable the switches when they are already enabled or disabled.

11.8.6 Call subroutine - M98

To call a subroutine program `M98 P~ Q~` The program must contain an O line with the number in the P word. This O line indicates the start of the subroutine. It will normally be with other subroutines and follow either an M2, M30 or M99 so it is not reached directly by the flow of the program.

The Q word gives the number of times that the subroutine is to be called before continuing with the line following the M98. If the Q word is omitted then its value defaults to 1.

By using parameters or incremental moves a repeated subroutine can make several roughing cuts around a complex path or cut several identical objects from one piece of material.

Subroutine calls may be nested. That is to say a subroutine may contain a M98 call to another subroutine. As no conditional branching is permitted it is not meaningful for subroutines to call themselves recursively.

11.8.7 Return from subroutine

To return from a subroutine program `M99` Execution will continue after the M98 which called the subroutine.

It is an error if:

- ♦ M99 is written in the main program, i.e. not in a subroutine.

11.9 Macro M-codes

11.9.1 Macro overview

If any M-code is used which is not in the above list of built-in codes then Mach2 will attempt to find a file named "`Mxx.M1S`" in the Macros folder. If it finds the file then it will execute the VB script program it finds within it.

The Configure>Macros menu item displays a dialog which allows you to see the currently installed macros, to Load, Edit and Save or Save As the text. The dialog also has a Help button which will display the VB functions which can be called to control Mach2. For example you can interrogate the position of axes, move axes, interrogate input signals and control output signals.

The usual VB Script data and control structures can be used in the program of a macro. For details consult documentation for Windows available at:
http:// msdn.microsoft.com/scripting/

11.9.2 Macro VB Script limitations

Because the macro is executed in a separate thread not connected to your Mach2 window you cannot use the MsgBox and InputBox functions to communicate with the user.

Debugging scripts is difficult. You might find it useful to use the `On Error Resume Next` followed by a call to the Mach2 `Question` function to display the error message stored in `err.description`

11.9.3 Writing macros

New macros can be written using an external program like Notepad and saved in the Macros folder or you can load an existing macro within Mach2, totally rewrite it and save it with a different file name.

A simple macro to toggle External Output 3 is given below.

```
Rem  EXTACT3  is signal  9

If IsActive (9)Then
      DeActivateSignal (9)
Else
      ActivateSignal (9)
Endif
```

11.10 Other Input Codes

11.10.1 Set Feed Rate - F

To set the feed rate, program `F~` The application of the feed rate is as described in G98 and G99 above.

11.10.2 Set Spindle Speed - S

To set the speed in revolutions per minute (rpm) of the spindle, program `S~` The spindle will turn at that speed when it has been programmed to start turning. It is OK to program an S word whether the spindle is turning or not. If the speed override switch is enabled and not set at 100%, the speed will be different from what is programmed. It is OK to program S0; the spindle will not turn if that is done. It is an error if:

- the S number is negative.

If a G84 (tapping) canned cycle is active and the feed and speed override switches are enabled, the one set at the lower setting will take effect. The speed and feed rates will still be synchronized. In this case, the speed may differ from what is programmed, even if the speed override switch is set at 100%.

11.10.3 Select Tool - T

To select a tool, program T~ where the T number is slot number for the tool. The tool is not changed automatically. It is OK, but not normally useful, if T words appear on two or more lines with no tool change. It is OK to program T0; no tool will be selected. This is useful if you want the spindle to be empty after a tool change. It is an error if:

- a negative T number is used,
- a T number larger than 255 is used.

11.11 Order of Execution

The order of execution of items on a line is critical to safe and effective machine operation. Items are executed in the order shown in figure 11.8 if they occur on the same line.

Order	Item
1	Comment (including message)
2	Set feed rate mode (G93, G94)
3	Set feed rate (F)
4	Set spindle speed (S)
5	Select tool
6	Tool change (M6) and Execute M-code macros
7	Spindle On/Off (M3, M4, M5)
8	Coolant On/Off (M7, M8, M9)
9	Enable/disable overrides (M48, M49)
10	Dwell (G4)
11	Set active plane (G17, G18, G18)
12	Set length units (G20, G21)
13	Cutter radius compensation On/Off (G40, G41, G42)
14	Tool table offset On/Off (G43, G49)
15	Fixture table select (G54 - G58 & G59 P~)
16	Set path control mode (G61, G61.1, G64)
17	Set distance mode (G90, G91)
18	Set canned cycle return level mode (G98, G99)
19	Home, or change co-ordinate system data (G10), or set offsets (G92, G94)
20	Perform motion (G0 to G3, G12, G13, G80 to G89 as modified by G53
21	Stop (M9, M1, M2, M30)

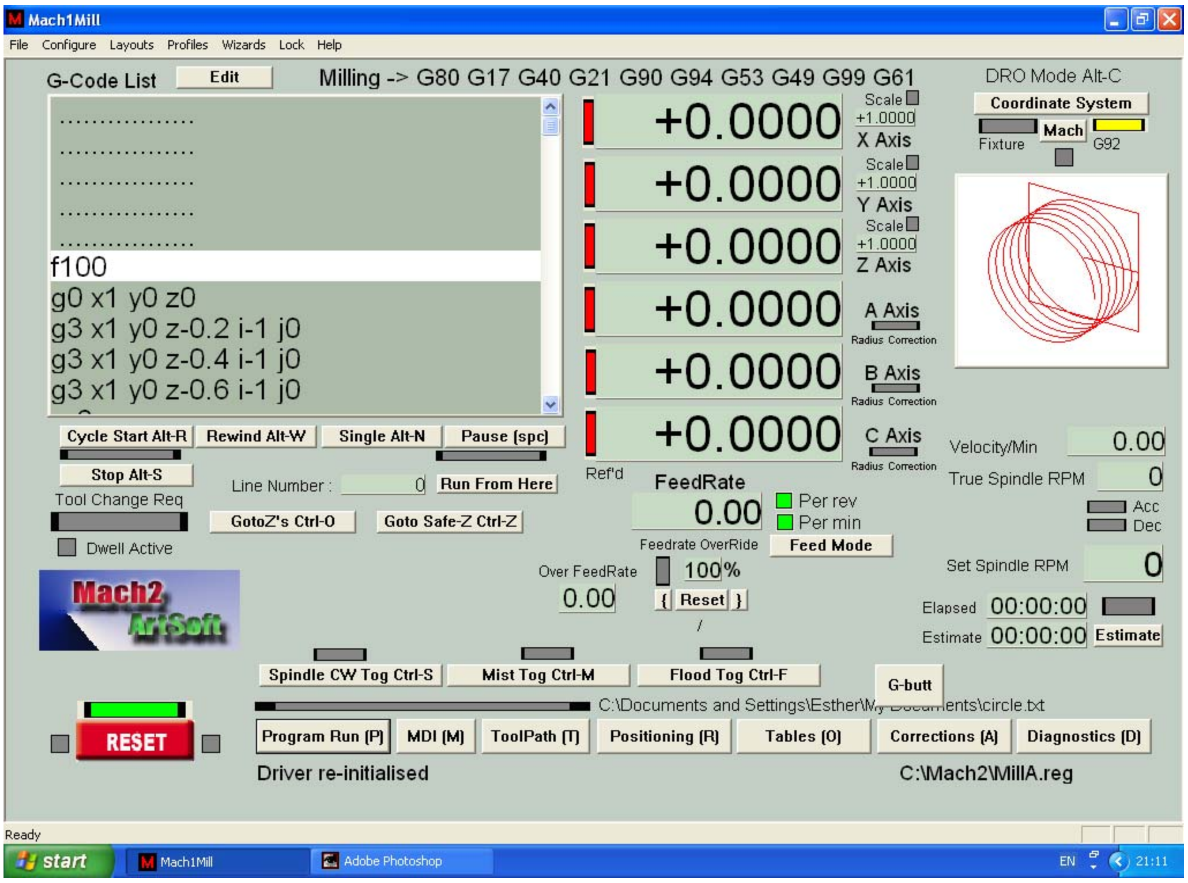
Table 11.8 - Order of execution on a line

11.12 Error Handling

This section describes error handling in Mach2.

Tmach2 tends to ignore things that it does not understand. If a command does not work as expected or does not do anything check that you have typed it correctly. Common mistakes are GO, instead of G0 i.e. letter O instead of zero) and too many decimal points in numbers. Mach2 does not check for axis overtravel (unless software limits are in use) or excessively high feeds or speeds. Nor does it does not detect situations where a legal command does something unfortunate, such as machining a fixture.

12. Appendix 1 - Mach2 screenshot pullout



Milling -> G80 G17 G40 G21 G90 G94 G53 G49 G99 G61

```
f100
g0 x1 y0 z0
g3 x1 y0 z-0.2 i-1 j0
g3 x1 y0 z-0.4 i-1 j0
g3 x1 y0 z-0.6 i-1 j0
```

X Axis +0.0000 (Scale +1.0000), Y Axis +0.0000 (Scale +1.0000), Z Axis +0.0000 (Scale +1.0000), A Axis +0.0000, B Axis +0.0000, C Axis +0.0000

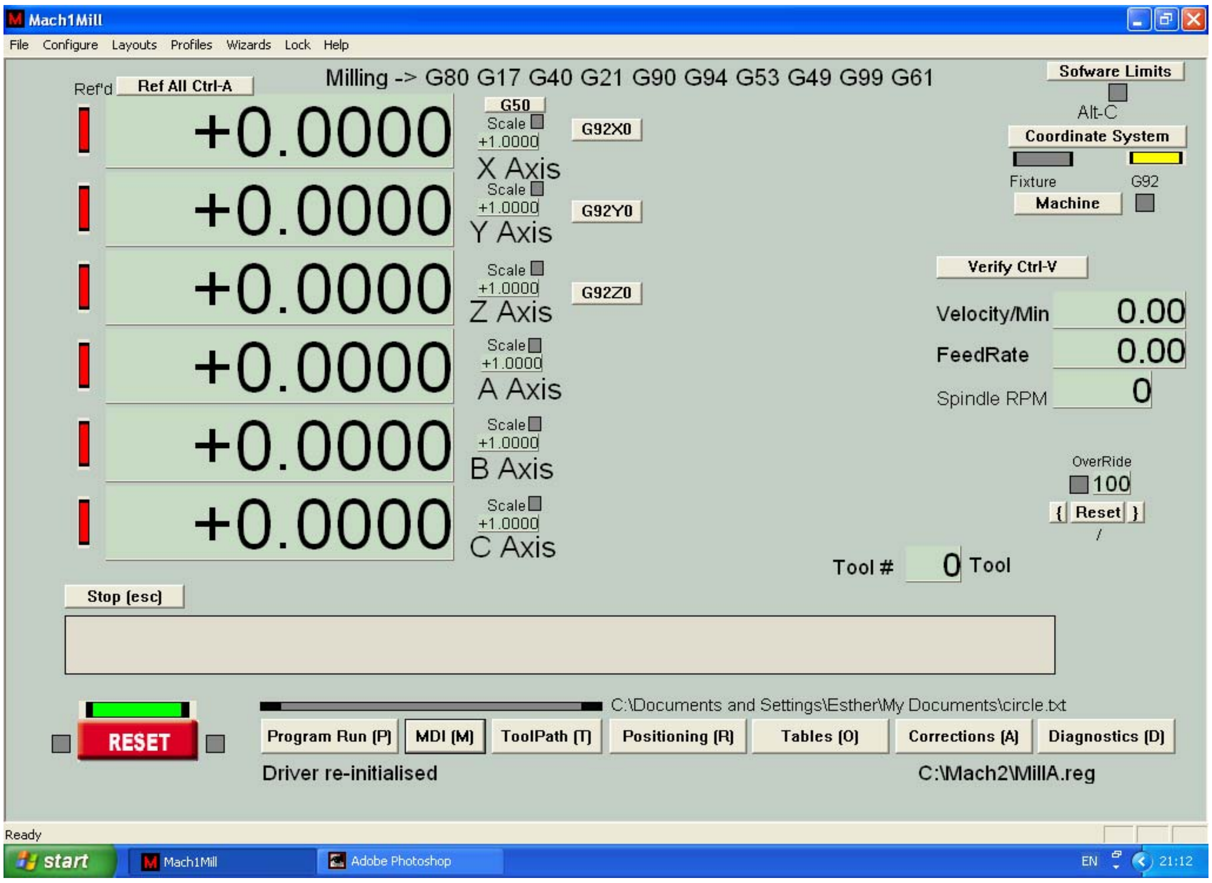
Line Number : 0 — FeedRate 0.00 — Over FeedRate 0.00 — Feedrate OverRide 100% — Velocity/Min 0.00 — True Spindle RPM 0 — Set Spindle RPM 0 — Elapsed 00:00:00 — Estimate 00:00:00

C:\Documents and Settings\Esther\My Documents\circle.txt

Driver re-initialised

C:\Mach2\MillA.reg

Mill Program Run screen



Milling -> G80 G17 G40 G21 G90 G94 G53 G49 G99 G61

X Axis +0.0000 (Scale +1.0000), Y Axis +0.0000 (Scale +1.0000), Z Axis +0.0000 (Scale +1.0000), A Axis +0.0000 (Scale +1.0000), B Axis +0.0000 (Scale +1.0000), C Axis +0.0000 (Scale +1.0000)

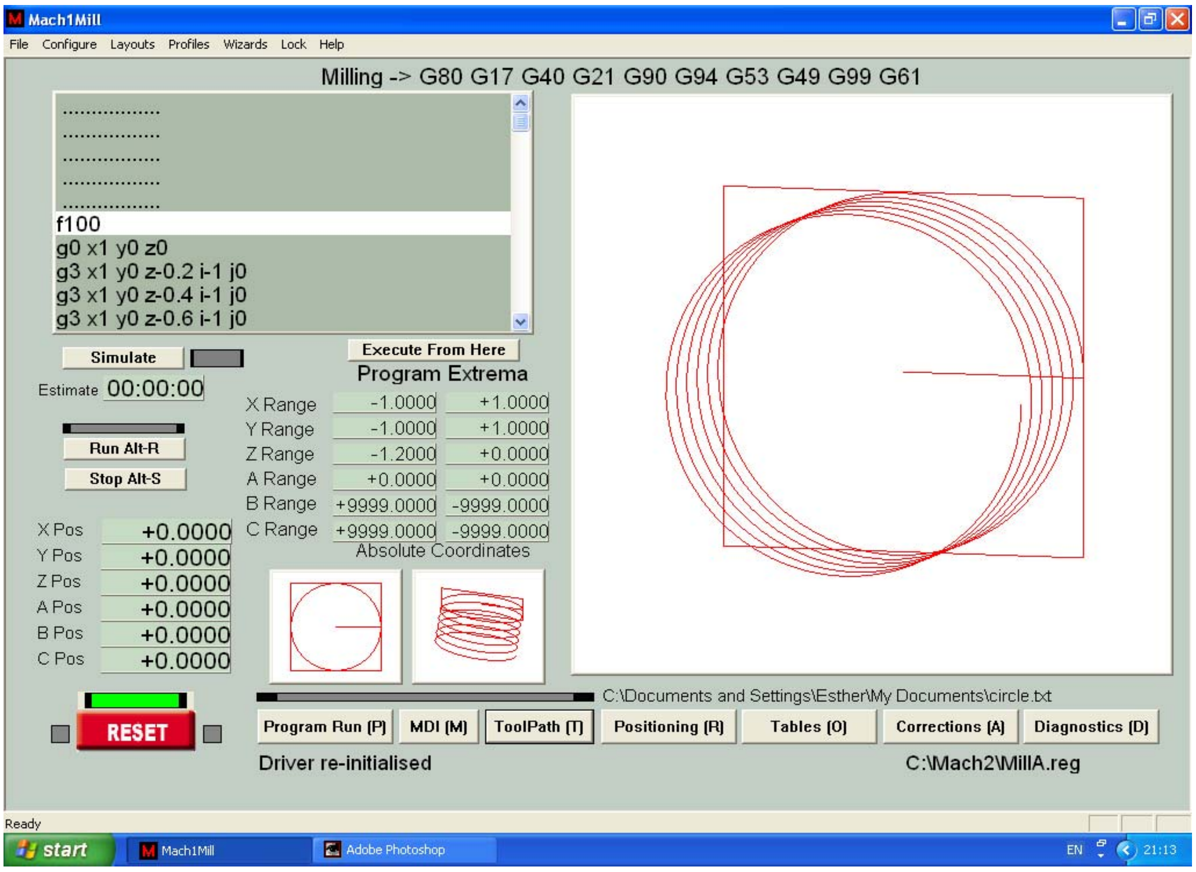
Velocity/Min 0.00 — FeedRate 0.00 — Spindle RPM 0 — OverRide 100 — Tool # 0

C:\Documents and Settings\Esther\My Documents\circle.txt

Driver re-initialised

C:\Mach2\MillA.reg

Mill MDI screen



Milling -> G80 G17 G40 G21 G90 G94 G53 G49 G99 G61

```
.................
.................
.................
.................
.................
f100
g0 x1 y0 z0
g3 x1 y0 z-0.2 i-1 j0
g3 x1 y0 z-0.4 i-1 j0
g3 x1 y0 z-0.6 i-1 j0
```

Estimate 00:00:00

Program Extrema

X Range	-1.0000	+1.0000
Y Range	-1.0000	+1.0000
Z Range	-1.2000	+0.0000
A Range	+0.0000	+0.0000
B Range	+9999.0000	-9999.0000
C Range	+9999.0000	-9999.0000

Absolute Coordinates

X Pos	+0.0000
Y Pos	+0.0000
Z Pos	+0.0000
A Pos	+0.0000
B Pos	+0.0000
C Pos	+0.0000

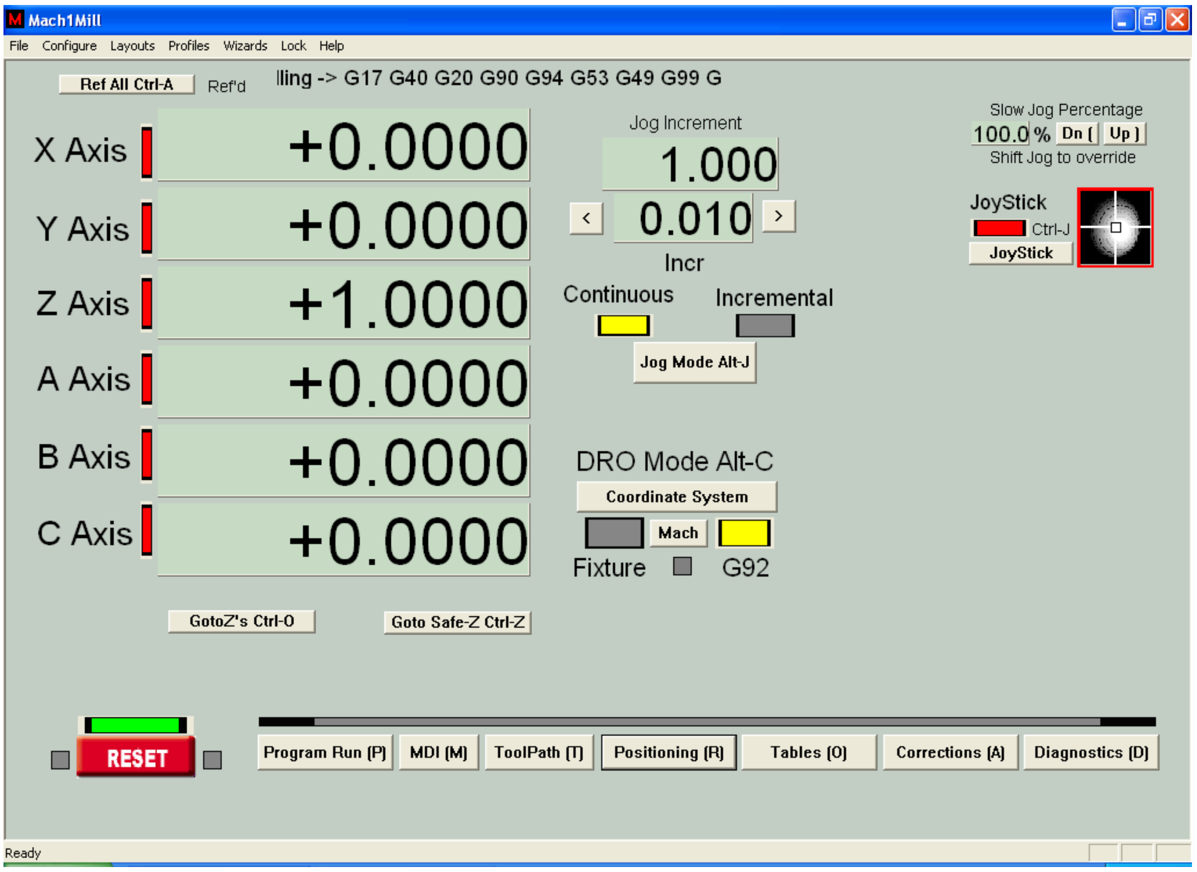
C:\Documents and Settings\Esther\My Documents\circle.txt

Driver re-initialised

C:\Mach2\MillA.reg

Ready

Mill Toolpath screen



lling -> G17 G40 G20 G90 G94 G53 G49 G99 G

X Axis	+0.0000
Y Axis	+0.0000
Z Axis	+1.0000
A Axis	+0.0000
B Axis	+0.0000
C Axis	+0.0000

Jog Increment 1.000

< 0.010 >

Incr

Continuous Incremental

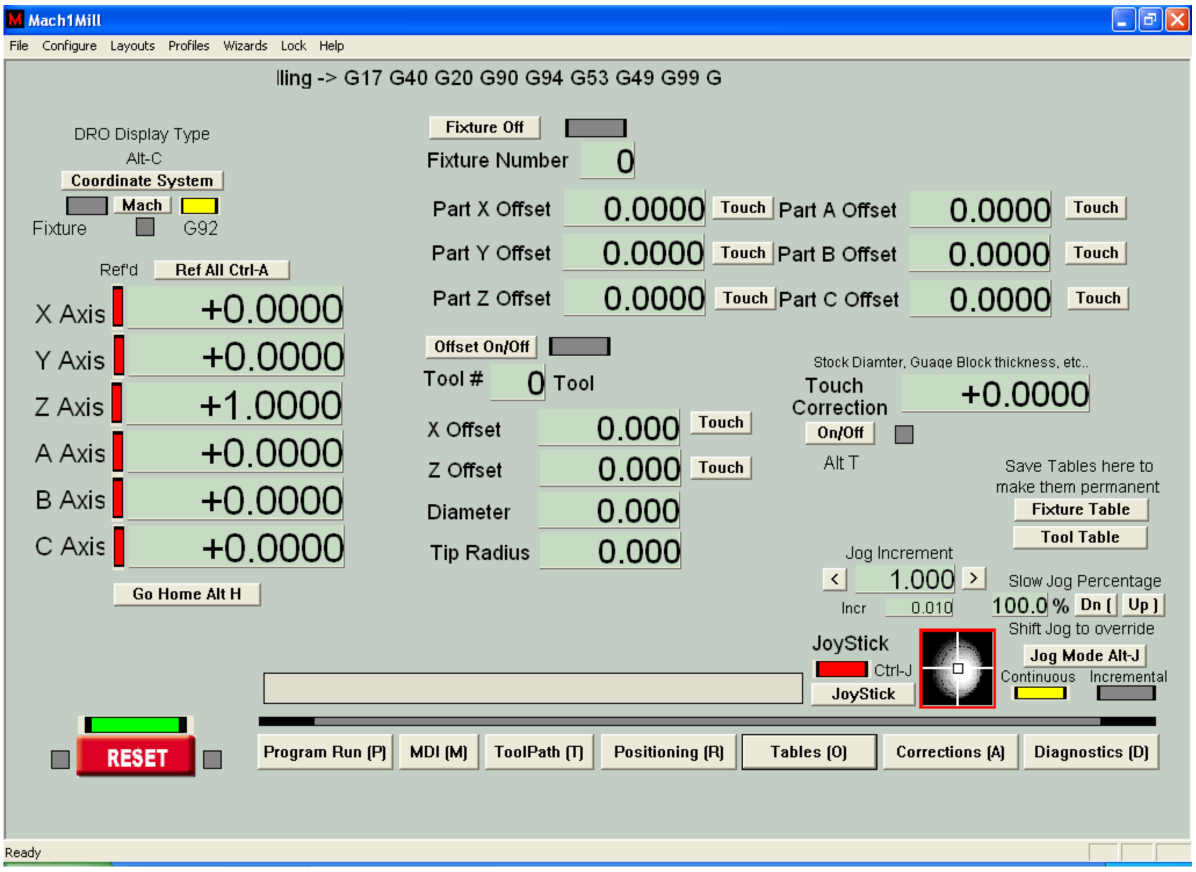
Slow Jog Percentage 100.0 % Dn [Up]
Shift Jog to override

JoyStick Ctrl-J
JoyStick

DRO Mode Alt-C
Coordinate System
Mach
Fixture G92

Ready

Mill Positioning screen


Mach1Mill

lling -> G17 G40 G20 G90 G94 G53 G49 G99 G

DRO Display Type Alt-C

Fixture G92

X Axis +0.0000
Y Axis +0.0000
Z Axis +1.0000
A Axis +0.0000
B Axis +0.0000
C Axis +0.0000

Fixture Number 0

Part X Offset 0.0000
Part Y Offset 0.0000
Part Z Offset 0.0000
Part A Offset 0.0000
Part B Offset 0.0000
Part C Offset 0.0000

Tool # 0 Tool

X Offset 0.000
Z Offset 0.000
Diameter 0.000
Tip Radius 0.000

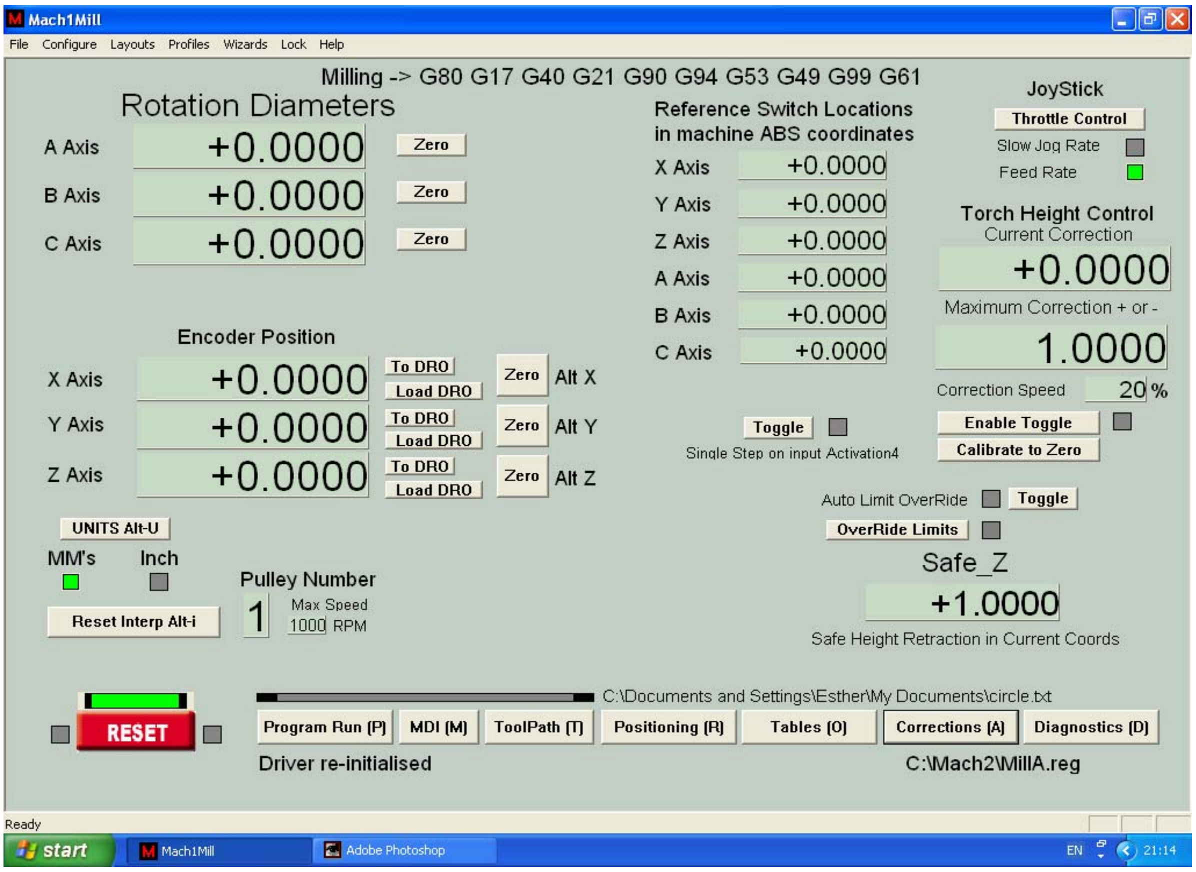
Stock Diamter, Guage Block thickness, etc..
Touch Correction +0.0000
Alt T

Save Tables here to make them permanent

Jog Increment 1.000
Incr 0.010
Slow Jog Percentage 100.0 %
Shift Jog to override

Ready

Mill Tables screen


Mach1Mill

Milling -> G80 G17 G40 G21 G90 G94 G53 G49 G99 G61

Rotation Diameters
A Axis +0.0000
B Axis +0.0000
C Axis +0.0000

Encoder Position
X Axis +0.0000
Y Axis +0.0000
Z Axis +0.0000

Reference Switch Locations in machine ABS coordinates
X Axis +0.0000
Y Axis +0.0000
Z Axis +0.0000
A Axis +0.0000
B Axis +0.0000
C Axis +0.0000

JoyStick
Slow Jog Rate
Feed Rate

Torch Height Control
Current Correction +0.0000
Maximum Correction + or - 1.0000
Correction Speed 20 %

Single Step on input Activation4

Auto Limit OverRide

Safe_Z +1.0000
Safe Height Retraction in Current Coords

MM's Inch

Pulley Number 1
Max Speed 1000 RPM

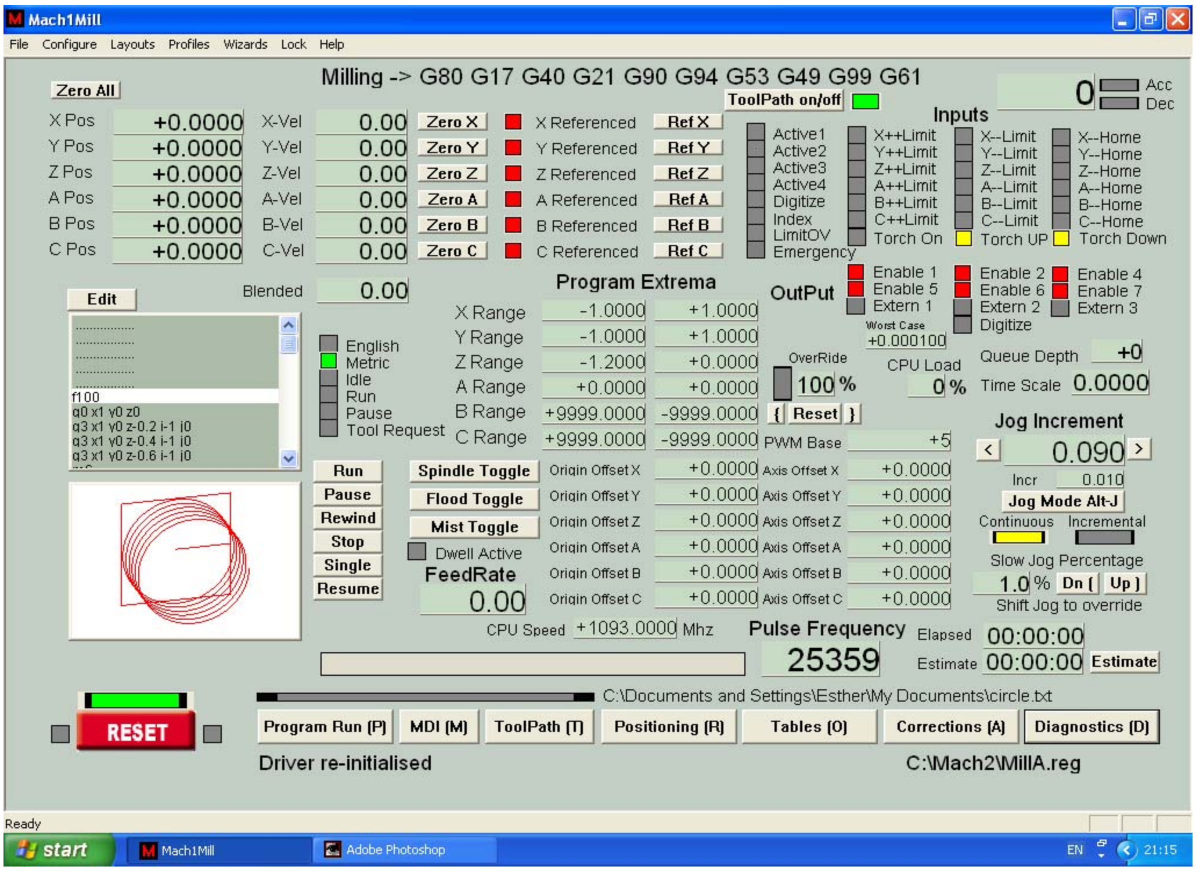
C:\Documents and Settings\Esther\My Documents\circle.txt

Driver re-initialised

C:\Mach2\MillA.reg

Ready

Mill Corrections screen



Milling -> G80 G17 G40 G21 G90 G94 G53 G49 G99 G61

X Pos	+0.0000	X-Vel	0.00
Y Pos	+0.0000	Y-Vel	0.00
Z Pos	+0.0000	Z-Vel	0.00
A Pos	+0.0000	A-Vel	0.00
B Pos	+0.0000	B-Vel	0.00
C Pos	+0.0000	C-Vel	0.00

🟥 X Referenced
🟥 Y Referenced
🟥 Z Referenced
🟥 A Referenced
🟥 B Referenced
🟥 C Referenced

Blended 0.00

```
f100
g0 x1 y0 z0
g3 x1 y0 z-0.2 i-1 j0
g3 x1 y0 z-0.4 i-1 j0
g3 x1 y0 z-0.6 i-1 j0
```

▨ English
🟩 Metric
▨ Idle
▨ Run
▨ Pause
▨ Tool Request

Program Extrema

X Range	-1.0000	+1.0000
Y Range	-1.0000	+1.0000
Z Range	-1.2000	+0.0000
A Range	+0.0000	+0.0000
B Range	+9999.0000	-9999.0000
C Range	+9999.0000	-9999.0000

▨ Dwell Active
FeedRate 0.00

Origin Offset X	+0.0000	Axis Offset X	+0.0000
Origin Offset Y	+0.0000	Axis Offset Y	+0.0000
Origin Offset Z	+0.0000	Axis Offset Z	+0.0000
Origin Offset A	+0.0000	Axis Offset A	+0.0000
Origin Offset B	+0.0000	Axis Offset B	+0.0000
Origin Offset C	+0.0000	Axis Offset C	+0.0000

CPU Speed +1093.0000 Mhz

Pulse Frequency 25359

0 Acc Dec

Inputs

▨ Active1	▨ X++Limit	▨ X--Limit	▨ X--Home
▨ Active2	▨ Y++Limit	▨ Y--Limit	▨ Y--Home
▨ Active3	▨ Z++Limit	▨ Z--Limit	▨ Z--Home
▨ Active4	▨ A++Limit	▨ A--Limit	▨ A--Home
▨ Digitize	▨ B++Limit	▨ B--Limit	▨ B--Home
▨ Index	▨ C++Limit	▨ C--Limit	▨ C--Home
▨ LimitOV	▨ Torch On	🟨 Torch UP	🟨 Torch Down
▨ Emergency			

OutPut

🟥 Enable 1	🟥 Enable 2	🟥 Enable 4
🟥 Enable 5	🟥 Enable 6	🟥 Enable 7
▨ Extern 1	▨ Extern 2	▨ Extern 3
	▨ Digitize	

Worst Case +0.000100
OverRide 100 %
CPU Load 0 %
Queue Depth +0
Time Scale 0.0000
PWM Base +5

Jog Increment 0.090
Incr 0.010
Continuous Incremental
Slow Jog Percentage 1.0 %
Shift Jog to override

Elapsed 00:00:00
Estimate 00:00:00

C:\Documents and Settings\Esther\My Documents\circle.txt

Driver re-initialised

C:\Mach2\MillA.reg

Ready

Mill Diagnostics screen

13. Appendix 2 - Standard shortcut codes

Shortcut	Function	Command/Op/OEMCode	Screen
(	Slow Jog Dn	112	ROD
)	Slow Jog Up	111	ROD
/	Feedrate reset	Reset Feed Ov	PMD
<	Inc Inc Down	101	ROD
<esc>	Stop	Stop	M
<space>	Pause	Pause program	P
>	IncInc Up	100	ROD
{	Feed lower	109	PMD
}	Feed raise	108	PMD
A	Corrections	6	PMTROAD
Alt-C	Coord System	Show Machine Coord	PR
Alt-H	Go Home	138	O
Alt-I	Reset Interp	102	A
Alt-J	Jog mode toggle	103	ROD
Alt-N	Single	Single step program	P
Alt-R	Cycle start	Run program	T
Alt-R	Cycle start	Run program	P
Alt-S	Stop	Stop program	PT
Alt-T	Touch Corr Toggle	148	O
Alt-U	Units	106	A
Alt-W	Rewind	Rewind program	P
Ctrl-A	Ref all	105	MO
Ctrl-F	Flood	113	P
Ctrl-J	Joystick Toggle		RO
Ctrl-M	Mist	114	P
Ctrl-O	GotoZs		P
Ctrl-S	Spindle CW	110 "M3"	P
Ctrl-V	Verify	Verify position	M
Ctrl-Z	Goto Safe Z		P
D	Diagnostics	5	PMTROAD
M	MDI	2	PMTROAD
O	Tables	7	PMTROAD
P	Program Run	1	PMTROAD
R	Positioning	4	PMTROAD
T	Toolpath	3	PMTROAD

14. Appendix 3 - Screen Designer OEM control codes

Type	Function	Code		Screen
Button	A Ref A	Home A		D
Button	Auto Lim Override Toggle	149		A
Button	B Ref B	Home B		D
Button	C Ref C	Home C		D
Button	Coord System	Show Machine Coord	Alt-C	PR
Button	Corrections	6	A	PMTROAD
Button	Cycle start	Run program	Alt-R	T
Button	Cycle start	Run program	Alt-R	P
Button	D Flood	113		D
Button	D Mist	114		D
Button	D Pause	Pause		D
Button	D Resume	Resume		D
Button	D Rewind	Rewind		D
Button	D Run	Run		D
Button	D Single	Single		D
Button	D Spindle	110		D
Button	D Stop	Stop		D
Button	Diagnostics	5	D	PMTROAD
Button	Edit G-code	115		PD
Button	Enc Load X	125		A
Button	Enc Load Y	127		A
Button	Enc Load Z	129		A
Button	Enc To X	126		A
Button	Enc To Y	128		A
Button	Enc To Z	130		A
Button	Estimate	Estimate Job		PD
Button	Execute from here	Set Next Exec		T
Button	Feed lower	109	{	PMD
Button	Feed raise	108	}	PMD
Button	Feedrate reset	Reset Feed Ov	/	PMD
Button	Fixture Off	137		O
Button	Fixture Tab Save	122		O
Button	Flood	113	Ctrl-F	P
Button	G50	"G50"		M
Button	G92X0	"G92X0"		M
Button	G92Y0	"G92Y0"		M
Button	G92Z0	"G92Z0"		M
Button	Go Home	138	Alt-H	O
Button	Goto Safe Z		Ctrl-Z	P
Button	GotoZs		Ctrl-O	P
Button	Inc Inc Down	101	<	ROD
Button	IncInc Up	100	>	ROD
Button	Jog mode toggle	103	Alt-J	ROD
Button	Joy Throttle select	147		A
Button	Joystick Toggle		Ctrl-J	RO
Button	Mach coords	107		PR
Button	MDI	2	M	PMTROAD
Button	Mist	114	Ctrl-M	P
Button	OverRide Limits	150		A
Button	Part A Offset Touch	142		O

Button	Part B Offset Touch	143		O
Button	Part C Offset Touch	144		O
Button	Part X Offset Touch	139		O
Button	Part Y Offset Touch	140		O
Button	Part Z Offset Touch	141		O
Button	Pause	Pause program	<space>	P
Button	Positioning	4	R	PMTROAD
Button	Program Run	1	P	PMTROAD
Button	Ref all	105	Ctrl-A	MO
Button	Reset Interp	102	Alt-I	A
Button	Rewind	Rewind program	Alt-W	P
Button	Rot A zero	0 - BUG		A
Button	Rot B zero	0 - BUG		A
Button	Rot C zero	0 BUG		A
Button	Run from here	SetNextExec		P
Button	Simulate	Estimate Job		T
Button	Single	Single step program	Alt-N	P
Button	Slow Jog Dn	112	(	ROD
Button	Slow Jog Up	111	)	ROD
Button	Software limits	119		M
Button	Spindle CW	110 "M3"	Ctrl-S	P
Button	SS on Act4 Toggle	151		A
Button	Stop	Stop program	Alt-S	PT
Button	Stop	Stop	<esc>	M
Button	Tables	7	O	PMTROAD
Button	Tool Offset Tog	136		O
Button	Tool Path Toggle	132		D
Button	Tool Tab Save	121		O
Button	Tool X Offset Touch	145		O
Button	Tool Z Offset Touch	146		O
Button	Toolpath	3	T	PMTROAD
Button	Torch Cal Zero	124		A
Button	Torch Enable Toggle	123		A
Button	Touch Corr Toggle	148	Alt-T	O
Button	Units	106	Alt-U	A
Button	Verify	Verify position	Ctrl-V	M
Button	X Ref X	Home X		D
Button	Y Ref Y	Home Y		D
Button	Z Ref Z	Home Z		D
Button	Zero A	Zero A		D
Button	Zero All	Zero All Axis		D
Button	Zero B	Zero B		D
Button	Zero C	Zero C		D
Button	Zero X	Zero X		D
Button	Zero Y	Zero Y		D
Button	Zero Z	Zero Z		D
DRO	A axis Ref Sw DRO	36	%+.4f	A
DRO	A DRO	A position	%+.4f	PMO
DRO	A DRO	A position	%+.4f	TD
DRO	A Fixture Orig Off DRO	Current A Offset	%+.4f	D
DRO	A G92 Axis Off DRO	19	%+.4f	D
DRO	A Scale DRO	62	%+.4f	M
DRO	A Vel DRO	A Velocity	%.2f	D
DRO	Amax	13	%+.4f	TD
DRO	Amin	7	%+.4f	TD
DRO	B axis Ref Sw DRO	37	%+.4f	A

DRO	B DRO	B position	%+.4f	TD
DRO	B DRO	B position	%+.4f	PMO
DRO	B Fixture Orig Off DRO	Current B offset	%+.4f	D
DRO	B G92 Axis Off DRO	20	%+.4f	D
DRO	B Scale DRO	63	%+.4f	M
DRO	B Vel DRO	B Velocity	%.2f	D
DRO	Blended Vel DRO	Toolpath Velocity	%.2f	D
DRO	Bmax	14	%+.4f	TD
DRO	Bmin	8	%+.4f	TD
DRO	C axis Ref Sw DRO	38	%+.4f	A
DRO	C DRO	C position	%+.4f	T
DRO	C DRO	C position	%+.4f	PMO
DRO	C Fixture Orig Off DRO	Current C Offset	%+.4f	D
DRO	C G92 Axis Off DRO	21	%+.4f	D
DRO	C Scale DRO	64	%+.4f	M
DRO	C Vel DRO	C Velocity	%.2f	D
DRO	Cmax	15	%+.4f	TD
DRO	Cmin	9	%+.4f	TD
DRO	CPU Load DRO	28	%3.0f	D
DRO	CPU Spd DRO	53	%+,4f	D
DRO	Elapsed DRO	Elapsed time	%.2f	PD
DRO	Enc X DRO	29	%+.4f	A
DRO	Enc Y DRO	30	%+.4f	A
DRO	Enc Z DRO	31	%+.4f	A
DRO	Estimate DRO	Estimated Time	%.2f	PTD
DRO	Feed DRO	Feedrate	%4.2f	PMD
DRO	Fixture # DRO	46	%3.0f	O
DRO	Joc Inc Inc	1	%.3f	ROD
DRO	Jog Inc DRO	Jog Increment	%.3f	ROD
DRO	Line no DRO	CurrGCode	% .0f	P
DRO	Part A Offset DRO	50	%.4f	O
DRO	Part B Offset DRO	51	%.4f	O
DRO	Part C Offset DRO	52	%.4f	O
DRO	Part X Offset DRO	47	%.4f	O
DRO	Part Y Offset DRO	48	%.4f	O
DRO	Part Z Offset DRO	49	%.4f	O
DRO	Pulley DRO	56	%+1.0f	A
DRO	Pulse Freq DRO	2	%5.0f	D
DRO	PWM Base DRO	24	%+4.0f	D
DRO	Queue Depth DRO	22	%+3f	D
DRO	Rot A diameter	A Axis radius	%+.4f	A
DRO	Rot Bdiameter	B Axis radius	%+.4f	A
DRO	Rot C diameter	C Axis radius	%+.4f	A
DRO	Safe Z DRO	54	%+.4f	A
DRO	Slow Jog % DRO	3	%3.1f	ROD
DRO	Spindle request DRO	SpindleSpeed	%.0f	PM
DRO	Time Scale DRO	23	%+.4f	D
DRO	Tool	Current Tool Number	%2.0f	MO
DRO	Tool Dia DRO	43	%.3f	O
DRO	Tool Tip Rad DRO	44	%.3f	O
DRO	Tool X Offset DRO	41	%.3f	O
DRO	Tool Z Offset DRO	42	%.3f	O
DRO	Torch Corr Sp DRO	25	%2.0f	A
DRO	Torch Height Corr DRO	26	%+.4f	A
DRO	Torch Height Max DRO	27	%+,4f	A
DRO	Touch Corr DRO	45	%+.4f	O

DRO	True spindle DRO	39	%.0f	PD
DRO	Velocity DRO	Toolpath Velocity	%4.2f	PM
DRO	Worst Case DRO	49	%+.6f	D
DRO	X axis Ref Sw DRO	33	%+.4f	A
DRO	X DRO	X position	%+.4f	TD
DRO	X DRO	X position	%+.4f	PMO
DRO	X Fixture Orig Off DRO	Current X Offset	%+.4f	D
DRO	X G92 Axis Off DRO	16	%+.4f	D
DRO	X Scale DRO	59	%+.4f	PM
DRO	X Vel DRO	X Velocity	%.2f	D
DRO	Xmax	10	%+.4f	TD
DRO	Xmin	4	%+.4f	TD
DRO	Y axis Ref Sw DRO	34	%+.4f	A
DRO	Y DRO	Y position	%+.4f	TD
DRO	Y DRO	Y position	%+.4f	PMOD
DRO	Y Fixture Orig Off DRO	Current Y Offset	%+.4f	D
DRO	Y G92 Axis Off DRO	17	%+.4f	D
DRO	Y Scale DRO	60	%+.4f	PM
DRO	Y Vel DRO	Y Velocity	%.2f	D
DRO	Ymax	11	%+.4f	TD
DRO	Ymin	5	%+.4f	TD
DRO	Z Vel DRO	Z Velocity	%.2f	D
DRO	Z axis Ref Sw DRO	35	%+.4f	A
DRO	Z DRO	Z position	%+.4f	PMO
DRO	Z DRO	Z position	%+.4f	TD
DRO	Z Fixture Orig Off DRO	Current Z Offset	%+.4f	D
DRO	Z G92 Axis Off DRO	18	%+.4f	D
DRO	Z Scale DRO	61	%+.4f	PM
DRO	Zmax	12	%+.4f	TD
DRO	Zmin	6	%+.4f	TD
LED	A radius corr. LED	20		P
LED	A ref LED	A ref		PMOD
LED	A Scale LED	44		M
LED	A++ Limit LED	A++		D
LED	A-- Home LED	A Home		D
LED	A-- Limit LED	A--		D
LED	Active 1 LED	Active1		D
LED	Active 2 LED	Active2		D
LED	Active 3 LED	Active3		D
LED	Active 4 LED	Active4		D
LED	Auto Lim LED	33		A
LED	B radius corr. LED	22		P
LED	B radius corr. LED	21		P
LED	B ref LED	B ref		PMOD
LED	B Scale LED	45		M
LED	B++ Limit LED	B++		D
LED	B-- Home LED	B Home		D
LED	B-- Limit LED	B--		D
LED	C ref LED	C ref		PMOD
LED	C Scale LED	46		M
LED	C++ Limit LED	C++		D
LED	C-- Home LED	C Home		D
LED	C-- Limit LED	C--		D
LED	Digitise In LED	Digitise		D
LED	Digitise Out LED	Digitise		D
LED	Dwell LED	Dwell		PD

LED	Emergency LED	19		D
LED	Enable 1 LED	Enable 1		D
LED	Enable 2 LED	Enable 2		D
LED	Enable 3 LED	Enable 3		D
LED	Enable 4 LED	Enable 4		D
LED	Enable 5 LED	Enable 5		D
LED	Enable 6 LED	Enable 6		D
LED	Estimating LED	18		PT
LED	Feed override LED	17		PMD
LED	Fixture LED	Mach Coor		PR
LED	Fixture on LED	29		O
LED	Flood LED	13		P
LED	G92 LED	10		PR
LED	Idle LED	Idle		D
LED	Inch LED	English		AD
LED	Index LED	Index		D
LED	Jog mode Cont LED	14		ROD
LED	Jog mode Incr LED	15		ROD
LED	Joy Feedrate LED	31		A
LED	Joy Slow Jog LED	30		A
LED	Joystick enable LED			RO
LED	Limit OV LED	LimitOV		D
LED	Long Reset LED 2	Estop		PMTROAD
LED	Mach coords warn LED	16		PR
LED	Mist LED	12		P
LED	MMs LED	Metric		AD
LED	OverRile Limits LED	34		A
LED	Pause LED	Pause program		PD
LED	Reset LED 1	Estop		PMTROAD
LED	Reset LED 3	Estop		PMTROAD
LED	Reset LED 4	Estop		PMTROAD
LED	Software limits LED	23		M
LED	Spindle CW LED	11		P
LED	SS on Act4 LED	35		A
LED	Start LED	Prog running		PD
LED	Tool change LED	ToolCh		PD
LED	Tool Offset on LED	28		O
LED	Tool Path LED	27		D
LED	Torch Down LED	38		D
LED	Torch En LED	24		A
LED	Torch On LED	36		D
LED	Torch Up LED	37		D
LED	Touch Coor On LED	32		O
LED	True spindle Acc LED	25		PD
LED	True spindle Dec LED	26		PD
LED	X ref LED	X ref		PMOD
LED	X Scale LED	41		PM
LED	X++ Limit LED	X++		D
LED	X-- Home LED	X Home		D
LED	X-- Limit LED	X--		D
LED	Y ref LED	Y ref		PMO
LED	Y Scale LED	42		PM
LED	Y++ Limit LED	Y++		D
LED	Y-- Home LED	Y Home		D
LED	Y-- Limit LED	Y--		D
LED	Z ref LED	Z ref		PMOD

LED	Z Scale LED	43		PM
LED	Z++ Limit LED	Z++		D
LED	Z-- Home LED	Z Home		D
LED	Z-- Limit LED	Z--		D

15. Appendix 4 - Sample relay based control logic

This may be a mistake but I do want to make sure people have a valid EStop strategy

16. Appendix 5 - Mach2 defined VB script function

The following functions are available to writers of macro scripts.

`StraightFeed(`**`double`**` x, `**`double`**` y, `**`double`**` z, `**`double`**` a, `**`double`**` b, `**`double`**` c)`

This will perform a feedrate move to X1,Y2,Z3…etc

`double`` GetSafeZ()`

This will return the current Safe_z to the VB routine.

`SetSafeZ(`**`double`**` SafeZ)`

This will set the Safe_Z

`SetCurrentTool(`**`short`**` Tool)`

Will return currently tool

`short`` GetSelectedTool()`

Will return tool selected but not yet activated.

`double`` GetToolChangeStart(`**`short`**` Axis)`

Will return the position of an axis when a toolchange was called for.

`StraightTraverse(`**`double`**` x, `**`double`**` y, `**`double`**` z, `**`double`**` a, `**`double`**` b, `**`double`**` c)`

Rapid move.

`double`` ToolLengthOffset()`

Gets the tool offset length currently in effect if any.

`double`` CommandedFeed()`

Gets current federate

`SetFeedRate(`**`double`**` Rate)`

Sets current FeedRate

`ActivateSignal(`**`short`**` Signal)`

Activates an output Pin.

`BOOL`` IsActive(`**`short`**` Signal)`

Checks a pin to see if its active.

`DeActivateSignal(`**`short`**` Signal)`

Turns an output pin off.

`SystemWaitFor(`**`short`**` Signal)`

Tells Mach2 to wait for an event. Should be last statement in a macro if used.

`double`` Param1()`

If Macro was called with P Q L parameters, they can be retrieved in the VB script by Param1(), Param2() and Param3() respectively

`VerifyAxis(`**`BOOL`**` Silent)`

Do a verification run. If silent is true, do not report the outcome, just correct the axis position.

`double`` GetVar(`**`short`**` var)`

Get a variable from the interpreter.

`SetVar(`**`short`**` var, `**`double`**` value)`

Set a Varibale in the interpreter.

double Question(**LPCTSTR** Text)

Ask a question, get an answer.

short QueueDepth()

Depth of planner queue is returned.

OpenDigFile()

Open a digitize point cloud file. User is prompted for filename.

CloseDigFile()

Close the file.

THCOn()

Turn on THC control

THCOff()

Turn off THC control.

17. Appendix 6 - Personal record of pins used

This allows you to record which pins on the parallel ports you use for which functions. If you complete it, it could save a lot of time when your memory has grown dim!

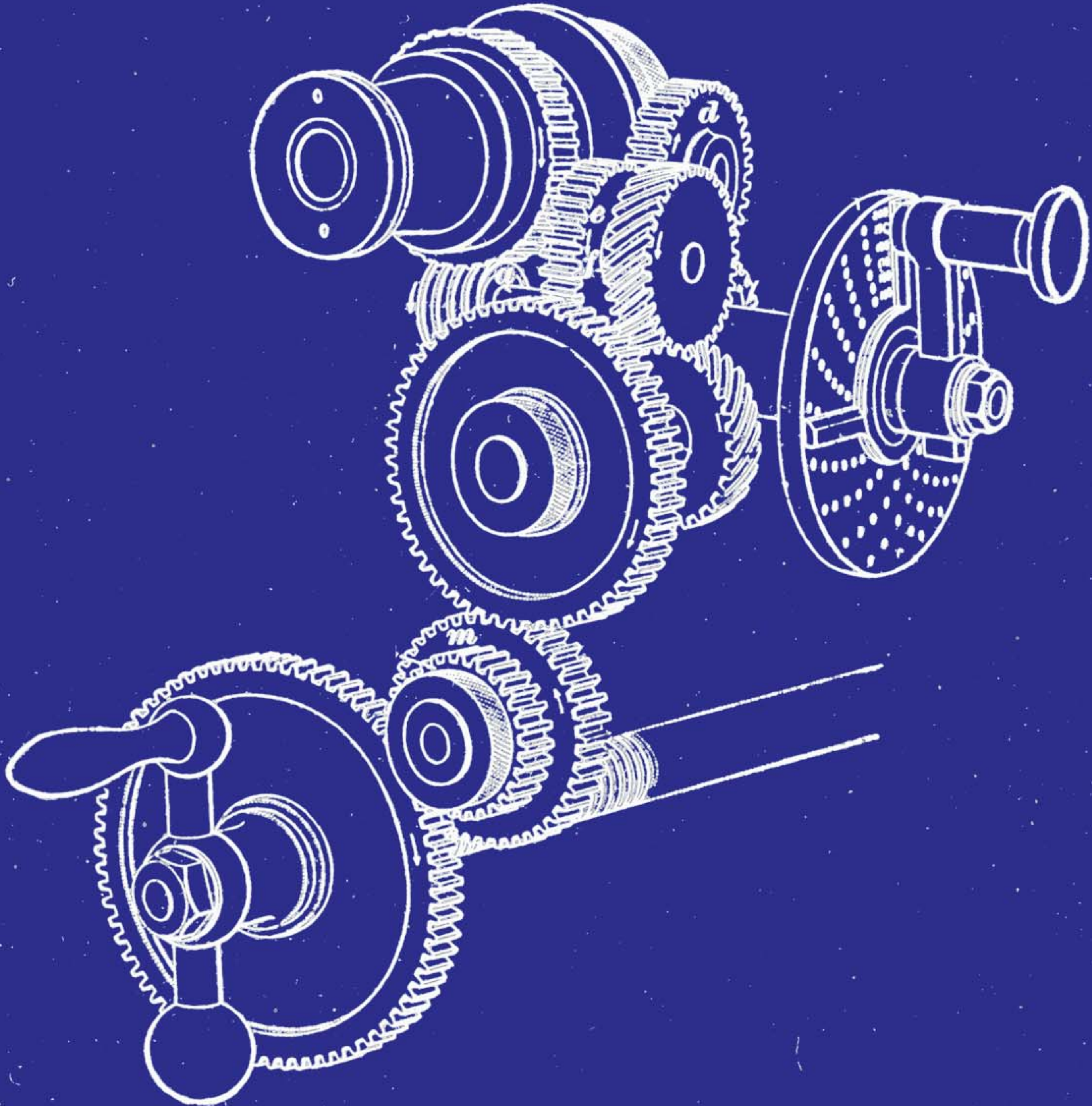
18. Revision history

Rev @10.4	30 May 2003	Chapter 5 added - configuration
Rev @9.6	27 May 2003	Various user suggestions and clarifications
Rev @9.5	23 May 2003	Scaling description added VB script functions added
Rev @9.1	21 May 2003	Major tidying of text for grammar etc. Chapter 8 on Screen Designer added Info on macros added Shortcut and OEM appendices added
Rev @8.2	16 May 2003	Chapter 11 definition of G- M- codes added Chapter 10 Cutter compensation added Contents added User suggestions added Lathe references removed/minimised
Rev @7.3	12 May 2003	Version includes Chapter 3 on "playing dry" with demo system and PDF bookmarks for sections
Rev @6.1	9 May 2003	Abortive attempt to convert to use Master documents. Word 97 is riddled with bugs in this area!
Rev @5.1	7 May 2003	Various user list suggestions incorporated. Index added. Crossref character style added for OEM to find cross-references to update.
Rev @4.1	6 May 2003	Chapter 4 added covering interface issues and designs. Skeletons of other sections published
Rev @3.1	30 April 2003	Ch 7 Revised to cover new Tools screen for Tool & Fixture tables
Rev @2.1	24 April 2003	Initial release of Co-ordinate systems, tool tables and fixtures chapter for comment

19. Index

Hint: Where there is a choice, most index entries are made using the name of a thing (e.g. Axis drive) rather than an action (e.g. Tuning) so you will get better results thinking about the part on which you want information. Thus looking for "Axis drives - tuning" will give better results than looking for "Tuning - axis drives". For important information both entries will probably appear.

If you have difficulty because you try to look something up and the index entry is missing, please take a moment to e-mail fenerty@artofcnc.ca {or for alpha and beta documentation japrentice@iee.org} with a note of (a) the words you were looking up and (b) where in the manual you found the information you wanted - assuming you did!


d
e
m