

# 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

All queries, comments and suggestions welcomed via support@artofcnc.ca

Mach Developers Network (MachDN) is currently hosted at: <a href="http://groups.yahoo.com/group/mach1mach2cnc/files/MachDN/">http://groups.yahoo.com/group/mach1mach2cnc/files/MachDN/</a>

© 2003 Art Fenerty and John Prentice

Front cover: A vertical mill circa 1914 Back cover (if present): The old, gear, way of co-ordinating motion on mill table and a rotary axis

For Mach2Mill Release 4.00

Manual Revision A4.5

1.	Preface	1-1
2.	Introducing CNC machining systems	2-2
2.1	Parts of a machining system	2-2
2.2	How Mach2 fits in	2-2
3.	An overview of Mach2 Machine Controller software	3-1
3.1	Installation	3-1
3.1.1	Downloading	
3.1.2	2 Installing	
3.1.3	Converient desiter icons	
3.1.5	5 Testing the installation	3-2
3.1.6	Notes for manual driver installation and un-installation	
3.2	Screens	
3.2.1	Types of object on screens	
3.2.2	2 Using buttons and shortcuts	
3.2.3	Data entry to DRO	
3.3	Jogging	
3.4	Manual Data Input (MDI)	
3.5	Running a G-code program	
3.6	Toolpath display	
3.6.1	Viewing the toolpath	
3.6.2	Panning and Zooming the toolpath display	
3.7	Other screen features	
4.	Hardware issues and connections to your machine tool .	4-1
4.1	Safety - emphasised	4-1
4.2	What Mach2 can control	4-1
4.3	The EStop control	4-2
44	The PC parallel nort	4-3
441	The parallel port and its history	4-3
4.4.2	2 Logic signals	
4.4.3	Electrical noise and expensive smoke	4-4
4.5	Axis drive options	4-5
4.5.1	Steppers and Servos	4-5
4.5.2	2 Doing Axis drive calculations	4-6
4.5.3	How the Step and Dir signals work	4-7
4.6	Limit and Home switches	4-8
4.6.1	Strategies	
4.6.2	2 The switches	
4.6.3 4.6.4	w nere to mount the switches How Mach2 uses shared switches	
4.6.5	Referencing in action	
4.6.6	Other Home and Limit options and hints	
4.7	Spindle control	4-11
4.8	Coolant	4-13

	4.9	Plasma torch height control (THC)	. 4-1.	3
	4.10	Digitise probe	. 4-14	4
	4.11	Linear (glass scale) encoders	. 4-14	4
	4.12	Spindle index pulse	. 4-14	5
	4.12	Charge numer a nulse monitor		5
	4.13	Charge pump - a puise monitor	. 4-1.	5
	4.14	Other functions	. 4-15	5
5.	1	Configuring Mach2 for your machine and drives	5-'	1
	5.1	A configuration strategy	5-1	1
	5.2	Initial configuration	5-1	1
	521	Defining addresses of port(s) to use	e -	2
	522	Defining axes to be used	5 1	2
	5.2	2.2.1 Controlled axes	5-2	2
	5.2	2.2.2 Spindle	5-2	2
	5.2.3	Defining engine frequency	5-2	2
	53	Defining input and output signals that you will use	5	2
	5.5	Output and output signals that you will use	3 5 /	3 7
	5.5.1	Input signals to be used	3-: 5	) 1
	5.5.2	input signals to be used	5	+
	5.4	Allocating Inputs and Outputs to ports and pins	5-:	5
	5.4.1	Defining output pins	5-0	6
	5.4.2	Defining input pins	5-0	6
	5.4	I.2.1 Emulator setup	5-0	6
	5.4	1.2.2 Parallel input setup	5-'	7
	5.4.3	Encoder inputs	۵-۵	8
	5.4.4 5.4.5	Assign External Activation Outputs	3-0 5 (	ð a
	5.5	Defining the setup units	5-9	9
	5 (		5 1(	n
	5.0	Tuning motors	. 5-10	0
	5.0.1	Calculating the steps per unit.	. 3-10	0
	5.0	5.1.1 Calculating meter steps per revolution	. 3-10	1
	5.0	5.1.2 Calculating Motol steps per revolution	. 3-1	1 1
	5.0	5.1.5 Calculating Wach2 steps per motor revolution	. 5-1 5-1	1
	562	Setting the maximum motor speed	5-12	2
	5.6	5.2.1 Practical trials of motor speed	. 5-12	2
	5.6	5.2.2 Motor maximum speed calculations	. 5-1.	3
	5.6.3	Deciding on acceleration	. 5-13	3
	5.6	5.3.1 Inertia and forces	. 5-13	3
	5.6	5.3.2 Testing different acceleration values	. 5-1.	3
	5.0	5.3.3 Why you want to avoid a big servo error	. 5-1.	3
	5.6	0.3.4 Choosing an acceleration value	. 5-14	4
	5.6.4	Saving and testing axis	. 5-14	4
	5.6.5	Repeat configuration of other axes	. 3-1:	5
	5.0.0	5.6.1 Motor speed spindle speed and pulleys	. 3-1. 5_1	5 5
	5.0	5.6.2 Pulse width modulated spindle controller	. 5-1. 5-1	3 7
	5.0 5.6	5.6.3 Step and Direction spindle controller	. 5-1	ź
	5.0	5.6.4 Testing the spindle drive	. 5-1'	7
	57	Other configuration	<b>5</b> 10	0
	J.1 571	Configura referencing	. 5-18 5-14	0
	J./.1	Configure referencing speeds and direction	. 3-18 5 10	ð Q
	5.	7.1.1 Referencing specus and differential and an and a specus and differential and	. 3-10 5_10	0 8
	570	Configure Encoders	. J-10 5_19	8
	5.1.2	2.1 Encoders for position display	5_10 5_19	8
	5.7	2.2 Encoder for jogging	5-19	9
	J.,			/
	5.7.3	Configure Backlash	. 5-19	9

	5.	7.4	Configure Slaving	19
	5.	7.5	Configure Soft Limits	20
	5.	7.6	Configure Initial State	20
	5.	7.7	Configure other Logic items	21
	5.8		How the Profile information is stored	23
	0.0			
c			Mach2 controls and running a part program	4
ο.			Machz controls and running a part program	- 1
	6.1		How the controls are explained in this chapter	5-1
	6.	1.1	Screen switching controls	5-1
		6.1.	.1.1 Reset	5-1
		6.1.	.1.2 Labels	5-1
		6.1.	.1.3 Screen selection buttons	5-1
	6.	1.2	Axis control family	<b>j</b> -1
		6.1.	.2.1 Coordinate value DRO	5-2
		6.1.	.2.2 Referenced	5-2
		6.1.	.2.3 Scale	5-3
		6.1.	2.4 Diameter/Radius correction	5-3
		6.1.	.2.5 G92 button	5-3
	6.	1.3	Coordinate system display mode family	5-3
	6.	1.4	Jogging control family	-3
		6.l.	.4.1 Hotkey jogging	)-4
		6.1.	.4.2 Parallel port encoder handwheel jogging	)-4
		0.1.	4.3 Joystick jogging	)-4 · 4
		0.1.	.4.4 Jog rate	)-4 5 5
		6.1	4.5 JogDall	55
	6	1.5	Feed control family	5-5
	0.	61	5.1 Feed Units per minute	5 5-5
		6.1	52 Feed Units per rev	, <i>5</i>
		6.1	53 Feed display	, o 5-6
		6.1	5.4 Feed override 6	5-6
	6.	1.6	Program Running control family	5-6
		6.1.	.6.1 Cycle Start	6-6
		6.1.	.6.2 Pause	6-6
		6.1.	.6.3 Stop	6-6
		6.1.	.6.4 Single	5-7
		6.1.	.6.5 Line Number	5-7
		6.1.	.6.6 Run from here	5-7
		6.1.	.6.7 Set next line	5-7
		6.1.	.6.8 Tool change request	5-7
		6.1.	.6.9 Dwell active	5-7
		6.1.	.6.10 M1 Optional Stop	5-7
		6.1.	.6.11 Block Delete	-7
	-	6.1.	.6.12 Ignore tool change	9-8
	6.	1.7	Auxiliary (Spindle & Coolant) control family	<b>-8</b>
	6.	1.8	G-code and Loolpath control family	8-8
	6.	1.9	File control family	-9
	6.	1.10 2 1	rixture and tool table control family	)-9 : 0
		0.1.	10.2 Teals	)-9 : 0
		0.1. 6 1	10.3 Rotational Diameter control family	י-א 10
	6	0.1.	Plasma Torch Height control family	10
	0.	 61	11.1 The controls 6-	10
		61	11.2 THC in operation 6-	10
		61	.11.3 Hints on G-code and running the system 6-	11
	6	1.12	Limits and miscellaneous control family 6-	12
	0.	6.1	.12.1 Input Activation 4	12
		6.1	12.2 Override limits	12
		6.1	.12.3 Soft limits enable	12
	6.	1.13	Reference switch location control family	12
	6.	1.14	System Settings control family	13
		6.1.	.14.1 Units	13
		6.1.	.14.2 Safe Z	13
		6.1.	.14.3 CV Mode/Angular Limit	13

	6.	1.14.4 Offline	6-13
	6.1.15	Encoder control family	6-13
	6.1.16	Automatic Z control family	6-13
	6.1.1/ 6.1.18	Laser Irigger output family	6-14
	0.1.10		0-14
	6.2	Loading a G-code part program	6-14
	6.3	Editing a part program	6-14
	6.4	Manual preparation and running a part program	6-15
	6.4.1	Inputting a hand-written program	6-15
	6.4.2	Before you run a part program	6-15
	6.4.3	Running your program	6-15
	6.5	Importing other files	6-15
7.		Coordinate systems, tool table and fixtures	7-1
	7.1	Machine coordinate system	7-1
	7.2	Fixture offsets	
	/.2.1	Setting Fixture origin to a given point	/-3
	1.2.2	nome in a practical machine	/-4
	7.3	What about different lengths of tool?	7-4
	7.3.1	Presettable tools	7-5
	7.3.2	Non-presettable tools	7-5
	7.4	Operational differences between Fixture tables and Tool tables	7-5
	7.5	Drawing lots of copies - real fixtures	7-6
	7.6	Saving your definitions of Fixtures and Tools	7-6
	7.7	Practicalities of "Touching"	7-7
	7.7.1	End mills	7-7
	7.7.2	Edge finding	7-7
	7.8	Workpiece offsets	7-7
	7.9	Tool diameter and tool tip radius	7-8
	7.9.1	Tool diameter	7-8
	7.9.2	Tip radius	7-9
8.		Optimising your Mach2 and machine	8-1
	8.1	Communication "routes"	8-1
	8.1.1	Electrical connections	
	8.1.2	Keystroke connections	8-1
	8.	1.2.1 Keystrokes	8-1
	8.	1.2.2 Keystrokes and Shortcuts (Hotkeys)	8-3
	8.1.3	The KeyGrabber and profilers	8-3
	0.1.4 8	VB Script connections	8-4 8_4
	8.	1.4.2 Mach2 macro	8-4
	8.1.5	Windows' control	8-5
	8.2	Coding VB Script programs	8-5
	8.2.1	A simple button script	8-5
	8.2.2	Sample macros	8-6
	8.	2.2.1 A simple macro	8-6
	8.	2.2.2 More complex macro	8-6
	8.2.3	A common confusion with VB Script and a hint	8-7
	0.2.4 &	2.4.1 To execute G or M-codes from a script	ð-ð 8_8
	8.	2.4.2 For accessing the screen controls	8-8
	8.	2.4.3 Interrogating Mach2 internal variable	8-10
	8.	2.4.4 Access to the machine G-code parameter block	8-10

	8.2	2.4.5 Arguments of macro call	8-11
	8.2	4.6 Information from and to the user	8-11
	8.2	2.4.7 Input/Output signals, a serial port and "foreign" ports	8-12
	8.2	2.4.8 Serial port	8-12
	8.4 0 7	2.4.9 Foreign ports	8-13 9 12
	0.2	2.4.10 Waiting realures.	0-13 8_13
	825	A more complicated macro example	8-13 8-13
	8.2.6	Script Snags and Hints	
	8.2	2.6.1 What Windows/Mach2 does with your macro	8-14
	8.2	2.6.2 Script error reporting	8-15
	8.2	2.6.3 Stuck in a rut?	8-15
	8.2	2.6.4 Reporting errors to users	8-15
	83	Screen Designer	8-16
	831	Screen Editor basics	8-16
	832	Try out the Designer	8-17
	8.3.3	Making the controls work	8-17
	8.3	3.3.1 Key scan codes	8-17
	8.3	3.3.2 Defining function by G-code or VB Script	8-18
	8.3	5.3.3 Defining function by name	8-19
	8.3	5.3.4 Defining buttons by OEM code	8-20
	8.3.4	Getting a tidy visual effect	8-20
	8.3	5.4.1 Alignment icons	8-20
	8.3	5.4.2 Sizing icons	8-20
	8.3	.4.3 Spacing controls uniformly	8-21
	8.3.5	Properties of other types of control.	8-21
	8.3.0	Properties of Intelligent Labels	8-22 8-22
	8.2	5.0.1 System labels	8-22 8_22
	837	DRO grouns	8-22 8-22
	838	Use of Bitmans	8-22
	8.3	3.8.1 Bitmap buttons	8-22
	8.3	3.8.2 Visual grouping with bitmaps	8-23
	8.3	3.8.3 Identifying controls by the background bitmap	8-24
	8.3	5.8.4 Dynamic changes with bitmaps	8-24
	8.3.9	Advanced features for setting up controls	8-24
	8.3.10	Colors	8-25
	8.4	Conversational programming	8-25
	0.7		0.05
	8.5	Other customisation	8-25
	8.5.1	Global hotkeys	8-25
~		DVE UDOL and images file imagest	• •
9.		DXF, HPGL and image file import	9-1
	9.1	Introduction	
	92	DXF import	0_1
	0.2.1	File loading	0 2
	9.2.1	File loading	
	9.2.2	Conversion options	
	1.4.5		
	924	Generation of G-code	9-3
	9.2.4	Generation of G-code	
	9.2.4 9.3	Generation of G-code	
	9.2.4 9.3 9.3.1	Generation of G-code HPGL import	
	9.2.4 9.3 9.3.1 9.3.2	Generation of G-code HPGL import About HPGL Choosing file to import	9-3 9-4 9-4 9-4
	9.2.4 9.3 9.3.1 9.3.2 9.3.3	Generation of G-code	9-3 9-4 9-4 9-4 9-5
	9.2.4 9.3 9.3.1 9.3.2 9.3.3 9.3.4	Generation of G-code HPGL import About HPGL Choosing file to import Import parameters Writing the G-code file	9-3 9-4 9-4 9-4 9-5 9-5
	9.2.4 9.3 9.3.1 9.3.2 9.3.3 9.3.4 9.4	Generation of G-code HPGL import About HPGL Choosing file to import Import parameters Writing the G-code file Bitmap import (BMP & JPEG)	9-3 9-4 9-4 9-4 9-5 9-5 9-5 9-5
	9.2.4 9.3 9.3.1 9.3.2 9.3.3 9.3.4 9.4 9.4.1	Generation of G-code HPGL import About HPGL. Choosing file to import Import parameters Writing the G-code file Bitmap import (BMP & JPEG) Choosing file to import	9-3 9-4 9-4 9-4 9-5 9-5 9-5 9-5 9-5
	9.2.4 9.3 9.3.1 9.3.2 9.3.3 9.3.4 9.4 9.4.1 9.4.2	Generation of G-code HPGL import About HPGL Choosing file to import Import parameters Writing the G-code file. Bitmap import (BMP & JPEG) Choosing file to import. Choose type of rendering	9-3 9-4 9-4 9-5 9-5 9-5 9-5 9-6 9-6
	9.2.4 9.3 9.3.1 9.3.2 9.3.3 9.3.4 9.4 9.4.1 9.4.2 9.4.3	Generation of G-code HPGL import About HPGL Choosing file to import Import parameters Writing the G-code file Bitmap import (BMP & JPEG) Choosing file to import Choose type of rendering Raster and spiral rendering	9-3 9-4 9-4 9-5 9-5 9-5 9-5 9-6 9-6 9-6 9-7
	9.2.4 9.3 9.3.1 9.3.2 9.3.3 9.3.4 9.4 9.4.1 9.4.2 9.4.3 9.4.4	Generation of G-code HPGL import	9-3 9-4 9-4 9-5 9-5 9-5 9-5 9-6 9-6 9-6 9-7 9-7

10.	Cutter compensation	10-1
10.1	Introduction to compensation	10-1
10.2	Two Kinds of Contour	10-1
10.2.1	Material Edge Contour	10-2
10.2.2	Tool Path Contour	10-2
10	.2.2.1 First Move	10-3
10	.2.2.2 Programming Entry Moves	10-4
11.	Mach 2 G- and M-code language reference	11-1
11.1	Some definitions	11-1
11.1.1	Linear Axes	11-1
11.1.2	Rotational Axes	11-1
11.1.3	Scaling input	11-1
11.1.4	Controlled Point	11-1
11.1.5	Co-ordinated Linear Motion	11-2
11.1.6	Feed Rate	11-2
11.1.7	Arc Motion	11-2
11.1.8		11-2
11.1.9	Dwell	11-3
11.1.10	Current Desition	11-5
11.1.11	Selected Plane	11-3 11_3
11.1.12	Tool Table	11-3
11.1.19	Tool Change	11-3
11.1.15	Pallet Shuttle	11-3
11.1.16	Path Control Modes	11-3
11.2	Interpreter Interaction with controls	11_4
11.2	Food and Spood Override controls	11 <i>A</i>
11.2.1	Block Delete control	11-4 11_1
11.2.2	Ontional Program Ston control	11-4
11.3	Tool File	11-4
11.4	The language of part programs	11-4
11.4.1	Overview	11-4
11.4.2	Parameters	11-5
11.4.3	Coordinate Systems	11-6
11.5	Format of a Line	11-7
11.5.1	Line Number	11-7
11.5.2	Subroutine labels	11-7
11.5.3	Word	11-7
11	.5.3.1 Number	11-7
11	.5.3.2 Parameter Value	11-8
11	.5.3.3 Expressions and Binary Operations	11-8
11	.5.3.4 Unary Operation Value	11-9
11.5.4	Parameter Setting	11-9
11.5.5	Comments and Messages	11-9
11.5.6	Item Repeats	11-9
11.5./	Commands and Machine Modes	11-10
11.6	Modal Groups	11-10
117	C Codes	11 11
11.7 1	G Cours.	11-11
11./.l 11.7.2	Linear Motion at Faed Pate G1	11-11
11./.2 1172	Are at Feed Rate - G2 and G3	11-13
11./.5	7 3 1 Radius Format Arc	11_13
11	7 3 2 Center Format Arc	11-14
11.7.4	Dwell - G4	11-15
11.7.5	Set Coordinate System Data Tool and Fixture tables - G10	11-15
11.7.6	Clockwise/counterclockwise circular pocket - G12 and G13	11-15

11.7.7	Exit and Enter Polar mode - G15 and G16	11	-16
11.7.8	Plane Selection - G17, G18, and G19	11	-16
11.7.9	Length Units - G20 and G21	11	-16
11.7.10	Return to Home - G28 and G30	11	-17
11.7.11	Reference axes G28.1	11.	-17
11./.12	Straight Probe – G31	11	-1/
11	7.12.2 Line Straight Probe Command	11	-1/
11	7.12.2 Using the Straight Piote Command	11	-1/ 10
11 7 13	Cutter Padius Comparisation G40 G41 and G42	11	10
11.7.13	Tool Length Offsets - G43 and G49	11	-19
11.7.14	Scale factors G50 and G51	11	_19
11.7.15	Move in Absolute Coordinates - G53	11	-19
11.7.10	Select Fixture Coordinate System - G54 to G59 & G59 P~	11	-20
11.7.18	Set Path Control Mode - G61, and G64	11	-20
11.7.19	Cancel Modal Motion - G80	11.	-20
11.7.20	Canned Cycles - G81 to G89	11-	-20
11	.7.20.1 Preliminary and In-Between Motion	11	-21
11	.7.20.2 G81 Cycle	11-	-21
11	.7.20.3 G82 Cycle	11	-22
11	.7.20.4 G83 Cycle	11	-22
11	.7.20.5 G84 Cycle	11	-23
11	.7.20.6 G85 Cycle	11	-23
11	.7.20.7 G86 Cycle	11	-23
11	.7.20.8 G87 Cycle	11.	-24
11	.7.20.9 G88 Cycle	11	-25
11 7 21	.7.20.10G89 Cycle	11	-25
11./.21	Coordinate System or Workniege Offsets CO2 CO2 1 CO2 2 CO2 3	11	-25
11.7.22	Set Feed Rate Mode - G93 G94 and G95	11	-23 -26
11.7.23	Set Canned Cycle Return Level - G98 and G99	11	-20
11.7.21			- /
11.8	Built-in M Codes	11	-27
11.8.1	Program Stopping and Ending - M0, M1, M2, M30	11	-27
11.8.2	Spindle Control - M3, M4, M5	11.	-28
11.8.3	Tool change - M6.	11.	-28
11.8.4	Coolant Control - M /, M8, M9.	11	-28
11.8.3	Override Control M48 and M40	11	-28
11.8.0	Call subrouting M08	11	-29
11.0.7	Return from subroutine	11.	-29
11.0.0			2)
11.9	Macro M-codes	11	-29
11.9.1	Macro overview	11	-29
11.10	Other Input Codes	11.	-29
11 10 1	Set Feed Rate - F	11.	-29
11 10 2	Set Spindle Speed - S	11.	-30
11.10.3	Select Tool - T	11.	-30
			21
11.11	Error Handling	11·	-31
11.12	Order of Execution	11	-31
12.	Supplied macros and other extensions	12	2-1
12.1	Contact probe digitising	. 1	2-1
12.2	Laser probe digitising	. 1	2-1
12.3	Standard Macro definitions	. 1	2_7
12.0		• 1	
13.	Appendix 1 - Mach2 screenshot pullout	13	3-1
14.	Appendix 2 - Standard shortcut codes	14	4-1

15.	Appendix 3 - Screen Layout files (.SET & .LSET)	15-1
15.1	Roles of Screen Designer and Mach2ScreenTweak	
15.2	Using Mach2ScreenTweak	
15.2.	1 Introduction	
15.2.	2 Installation	
15.2.	3 The main screen and its buttons	
15.2.	4 Manipulate all screens in layout	
	15.2.4.1 Open .SET	
	15.2.4.2 CSV Save	
	15.2.4.3 Save As	
	15.2.4.4 Edit All Controls	
15.2	15.2.4.5 Edit Undo	
13.2.	15 2 5 1 Delete	
	15.2.5.1 Detete:	
	15.2.5.2 Wrote Op Down instantial	
15.2.	6 The Additional Lavout	
	15.2.6.1 Append to Principal Layout	
15.2.	7 Control Manipulation	
	15.2.7.1 Re-scale controls	15-5
	15.2.7.2 Re-order DROs	
15.2.	8 Screen captions and other workarounds	
	15.2.8.1 Screen captions	
	15.2.8.2 Buttons identify screens	15-6
15.3	Layout file format	
15.3.	1 Overall file format	
15.3.	2 ControlRec	
	15.3.2.1 Screen	
	15.3.2.2 Type	
	15.3.2.3 Function	
	15.3.2.4 OEMCode	
	15.3.2.5 lext	15-8
	15.5.2.0 GTeXt	
	15.3.2.7 BitiviapFatti	
	15.3.2.8 Holiz & Vet	
	15.3.2.9 Eucer	15-8
	15.3.2.11 HotKey	
	15.3.2.12 Flash Flag	
	15.3.2.13 RedGreen Flag	
	15.3.2.14 Tabbing Group	15-9
	15.3.2.15 PosX1, Y1, X2, Y2	
15.3.	3 ColorsRec	
16.	Appendix 4 - Sample schematic diagrams	
16.1	EStop and limits using relays	
16.2	Torch Height Control interface	
17.	Appendix 5 - Mach2 defined VB script functions	
10		
18.	Appendix 6 – General utility programs	
18.1	KeyGrabber	
18.1.	1 Overview	
18.1.	2 Installation	
	18.1.2.1 The files	
	18.1.2.2 Windows compatibility	
	10.1.2.5 Kulling Keyorabbel and then Mach2	
18 1	3 Configuring KeyGrabher	18-7
18.1.	4 Configuring Keyboard Keys	

21.	Index	
20.	Revision history	2
19.	Appendix 7 - Record of configuration used	1
	18.1.10 Testing and troubleshooting	
	18.1.9 Keyboard Emulator programming	
	18.1.8 Multiple machines and KeyGrabber Profiles (.GRAB files)	
	18.1.7 Axes as joysticks	
	18.1.6.5 Misc Settings	
	18.1.6.4 HID encoders	
	18.1.6.3 HID Keys	
	18.1.6.2 The HID Controllers tab	
	18.1.6.1 Preparation for HIDs	
	18.1.6 Configuring HIDs	
	18.1.5 Configuring Keyboard Encoders	

## 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 the performance of 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 (Mach2Turn) and supporting documentation is being developed to support the full functionality of a lathes etc.

You are strongly advised to join the online discussion forum for Mach2. This is currently hosted by Yahoo! A link to join it is on the *Company* page at <u>www.artofcnc.ca</u> You should be aware that, while the forum has many engineers with a vast range of experience as participants, it does not constitute a substitute for a machine tool manufacturer's support network. If your application requires this level of support then you should buy the system from a local distributor or an OEM with a distributor network. In that way you will get the benefits of Mach2 with the possibility of on-site support.

Certain portions of text in this manual are printed "greyed out". They generally describe features found in machine controllers but 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. Credits are given for individual utilities and features as these are described in the body of the manual.

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

Art Fenerty and John Prentice assert their right to be identified as the authors of this work. The right to make copies of this manual is granted solely for the purpose of evaluating and/or using licensed or demonstration copies of Mach2. It is not permitted, under this right, for third parties to charge for copies of this manual.

Every effort has been made to make this manual as complete and as accurate as possible but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this manual, Use of the manual is covered by the license conditions to which you must agree when installing Mach2 software.

#### Preface

Windows XP and Windows 2000 are registered trademarks of Microsoft Corporation. If other trademarks are used in this manual but not acknowledged please notify ArtSoft Corporation so this can be remedied in subsequent editions.

## 2. Introducing CNC machining systems

## 2.1 Parts of a machining system



Figure 1.1 - Typical NC 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.

The main parts of a system for numerically controlled mill 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, vertical borer etc.

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 (6) are not trying to move any axis beyond its limits.

The Machine Controller also has controls like buttons, a keyboard, potentiometer knobs, a manual pulse generator (MPG) wheel, 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.

Mach2 communicates principally via one (or optionally two) parallel (printer) ports and, if desired, a serial (COM) port.

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. An overview of 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 by ArtSoft Corp. via the Internet. You download the package as one self installing file (which, in the present release, is less than 3.5 megabytes). This will run for an indefinite period as a demonstration version with a few limitations on the speed, 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 Corporation website www.artofcnc.ca

## 3.1.1 Downloading

Download the package from <u>www.artofcnc.ca</u> using the right mouse button and *Save Target as*... to put the self-installing file in any convenient working directory (perhaps Windows\Temp). You should be logged in to Windows as and Administrator.

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 possibly required (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 reasonably quickly that it is not much hardship to do it every time.

## 3.1.4 Convenient desktop icons

So you **have** rebooted! The installation wizard will have created desktop icons for the main programs. Mach2.exe is the actual user interface code. If you run it, it will ask which Profile you wish to use. Mach2Mill, Mach2Turn etc. are shortcuts which run this with a Profile defined by a "/p" argument in the shortcut target. You will usually employ these to start the required system.

It is now worthwhile to setup some icons for desktop shortcuts to other Mach2 programs. Use Windows Explorer (right-click *Start*) and by right-clicking on the .EXE file for the screen designer create a shortcut to the file. Repeat this for the file OCXDriverTest.exe and KeyGrabber.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 systems due to many factors. For example, QuickTime's system monitor (qtask.exe) running in the background can kill it 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. Other possible sources of erratic operation are local area network connections which may be configured to automatically speed detect. You should configure these to the actual speed 10 Mbps or 100 Mbps of your network. Finally a machine that has been surfing the Internet may have gained one or more of a host of "robot" type programs which spy on what you are doing and send data over the 'net to their originators. This traffic can interfere with Mach2 and is not something you want anyway. Use a search engine for terms like "Spybot" to locate software to tidy up your machine.

Because of these factors, 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 Frequency. 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



Figure 3-1 - OCXTest screenshot

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  $\frac{1}{4}$  inch or so on most systems. Even if there are variations its possible they are below the threshold necessary to create timing jitters so when your machine tool is connected you should perform a movement test to see if jogging and G0/G1 moves are 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 see 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 <u>www.artofcnc.ca</u> 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 a Profile file. 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 .XML profile file or files.

## 3.2 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 differences then your system supplier should have given you a revised 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

RESET	Program Rua (F1) MDI	I (F2)	Positioning (F3)	ToolPath (F4)	Tables (FS)	Corrections (F6)	Diagnostics (F7)
						Mach2Mill	

Figure 3.2 - The screen selection buttons

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:

- Buttons (e.g. Reset, Stop Alt-S, etc.)
- DROs or Digital Readouts. 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)
- ♦ Jogging controls

There is one further important type of 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 can be 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 most buttons have a keyboard hotkey. 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 letters are shown as uppercase (for ease of reading) you do **not** use the shift key when using the shortcuts.

A keyboard emulator circuit with push switches connected to it or a wide range of custom keyboards, often fitted with the USB interface, can be connected. You can use these on a long lead as a pendant controller or to label dedicated keys. You can customise your system to respond to the codes generated by these special devices.

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, clicking its hotkey (or using the global hotkey to select DROs and moving to 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 your own data into a DRO. For example the display of your 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 by 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 3.3 gives a possible view of them.

You can often 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 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. You can use the jogging keys on any screen with the *Jog Mode* button on it.

In figure 3.3 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



Figure 3.3 - Jog controls (simulated view)

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. A rotary encoder (sometimes termed a digital potentiometer) can be interfaced to Mach2 and used to perform incremental jogging by turning its knob.

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. You can use multiple joysticks (strictly Axes on Human Interface Devices) by installing manufacturer's profiler software or, even better, the KeyGrabber utility supplier with Mach.

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 press Enter which will automatically select it. 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:

GO X1.6 Y2.3

Which will move the tool to coordinates 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 coordinates.

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 it easy to repeat a command 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 line (block). If in doubt about the order that will be used then type several separate 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  ${\tt spiral.txt}$ 

```
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
```

Don't forget to put press the *Enter* key after the 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*.

Note: You should run you program from a harddrive. Mach2 needs highspeed access to the file, which it maps into memory. The program file must not be read-only either.

## 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.

The display is like a wire model of the path the tool will follow placed inside a clear sphere. By dragging the mouse over the window you can rotate the "sphere" and so see the model from different angles. The set of axes in the top left hand corner how you what directions are X, Y and Z. So if you drag the mouse from the centre in an upwards direction the "sphere" will turn showing you the Z axis and you will be able to see that the circle is actually a spiral cut



Figure 3.4 Toolpath from Spiral.txt

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.

You can if you wish produce a display like the conventional isometric view of the toolpath.

A few minutes of "play" will soon give you confidence in what can be done. Your display may be a different colour to that shown in figure 3.4. The colors can be configured. See chapter 5.

## 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 in the window with the Right mouse button held.

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

## 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 coordinate 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.

This chapter tells you about the hardware aspects of connections. Chapter 5 gives details of configuring Mach2 to use the connected items.

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.

## 4.1 Safety - emphasised



Any machine tool is potentially dangerous. 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 the performance of 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 a very flexible program 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" 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
- Special functions.

Most 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. Connections can also be made through a "keyboard emulator" which generates pseudo key presses in response to input signals.

The control of special functions like an LCD display, a tool-changer, axis clamps or a swarf conveyor is by user defined M-code macros which can control a serial (COM) port.

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 coordinates 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 types 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 an 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. USB is taking over many of these functions and this conveniently leaves the parallel port free for Mach2.

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

On first reading, you may wish to skip to the next heading 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 your 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 (transistortransistor 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.<sup>1</sup> 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 in, 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, current flowing out of a "hi" will make the voltage be lower and nearer to the 2.4 volts lower limit. 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".

<sup>&</sup>lt;sup>1</sup> 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!

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 practical **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 (less than 40 microamps) for "hi" inputs and will supply some (less than 0.4 milliamps) for "lo" inputs.

Because modern computer motherboards combine many function, including the parallel port, into one chip we have experienced systems where the voltages only just obey the "hi" and "lo" rules. You might find that a machine tool that ran on and old system becomes temperemental when you upgrade the computer. Pins 2 to 9 are likely to have similar properties (they are the data pins when printing). Pin 1 is also vital in printing but the other output pins are little used and may be less powerful in a carefully "optimised" design. A good isolating breakout board (see next section) will protect you from these electrical compatibility problems.

## 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 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.

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



Figure 4.2 - Cased breakout board, drive electronics and power 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". This will provide you with terminals that are easy to connect to, a separate 0 volt (common) for the drives, home switches etc. and will avoid exceeding 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" then 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 ballnut), belts, chains or rack and pinion. The mechanical drive method will determine the speed and torque required and hence any gearing required between the motor and machine.

Properties of a bipolar 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 3000 ounce inches. (21 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. Provides open loop control which means it is possible to lose steps under high loading and this may not immediately be obvious to the machine user.



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

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 a practically unlimited torque (if your budget can stand it!)
- 5. Provides closed loop control so drive position is always known to be correct (or a fault condition will be raised)

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.

Fuller details of the calculations are given in chapter 5.

#### **Example 1 - MILL TABLE CROSS SLIDE**

We start with checking the minimum possible move distance. This is an absolute limit to the accuracy of work done on the machine. We will 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". This is  $1/1000}$  of a revolution of the motor shaft if it is coupled directly to the screw.

#### Slide 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 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 (in ounces) x 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.

#### Slide with servo motor

Again we look at the size of one step. A servo motor has an encoder to tell its drive electronics where it is. This consists of a slotted disc and will generate four "quadratuere" pulses for each slot in the disc. Thus a disc with 300 slots generates 300 cycles per revolution (CPR) This is fairly low for commercial encoders. The encoder electronics will output 1200 quadrature counts per revolution (QCPR) of the motor shaft.

The drive electronics for the servo will usually turn the motor by one quadrature count per input step pulse. Some high specification servo electronics can multiply and/or divide the step pulses by a constant (e.g. one step pulse moves by 5 quadrature 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 if you have high count encoders.

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 at least 60" on the gantry axis and a ballscrew for this length will be expensive and difficult to protect from dust. Many designers would 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 microsteps) gives 2000 steps per revolution so a 5:1 reduction



Figure 4.4 - Step pulse waveform

(belt or gear box) is needed between the motor and sprocket shaft.  $[0.0005" = 5"/(2000 \times 5)]$ 

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



Figure 4.5 - Wrongly configured output alters step waveform

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.

## 4.5.3 How the Step and Dir signals work

Mach2 puts one pulse (logic 1) 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.

## 4.6 Limit and Home 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 **Home switch**. Mach2 can be commanded to move one (or all) axes to the home 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 Home switch then you will have to jog the axes by eye to a reference position. The home switch for an axis can be at the any coordinate position and you define this location. Thus the home switches do not have to be at **Machine Zero**.



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

Thus each axis could need three switches (i.e.

limit switches at the two ends of travel and a home switch). So a basic mill would require nine parallel port inputs for them. This is not much good as a parallel port only has 5 inputs! The problem can be solved in three 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 switches can be interfaced by a keyboard emulator.

The first method is best and mandatory for a very large, expensive or fast machine where you cannot trust software and its configuration to prevent mechanical damage. Switches connected to the 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 but does rely on having a sophisticated drive.

On a small machine when you use 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.

The keyboard emulator has a much slower response time that the parallel port but is satisfactory for limit switches on a machine without highspeed feeds.

## 4.6.2 The switches

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

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"



Figure 4.7 - Two NC mechanical switches give logic OR

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.



Figure 4.8 - Optical switch on table with vane on bed of machine

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.

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 for home.

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 4.11). 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 4.8 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



Figure 4.10 – Mill with tool at X=0, Y=0 position (note the dog is on limit switch)



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 Master5/Mach1/Mach2 Yahoo! group for more ideas and techniques for switches.

## 4.6.4 How Mach2 uses shared switches

This section refers to the configuration for small machines where Mach2 rather than EStop is controlled by the switches.

For a full understanding of this you will also have to read the section in chapter 5 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 home switch.

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 toggle 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* button. 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 home switch operates. 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 coordinate. If you use zero then the home switch position is also the machine zero 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 home switch) and stop when you get off the switch. This is fine when you have a separate home 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 home point until it crashes. So the advice is: **always jog carefully off the limit switches, then reference.** 

## 4.6.6 Other Home and Limit options and hints

#### Home switch not near limit switch

It is sometimes not very convenient to have the home 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 home 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 home 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.

#### Separate high accuracy home switch

The X and Y axes on a high precision machine might have a separate home 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.

#### Home switches of multiple axes connected together

If you are **really** short of inputs to Mach2 then you can OR the home switches together and define all home inputs to be that signal. In this case you can only reference one axis at once – so you need to remove REF All buttons from your screens – and your home switches must all be at the end of travel on their respective axes.

#### 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 5 on Configuring Mach2 for details. Both axes should have limit and home 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 home 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.

Figures 4.12 and 4.13 show the pulse width at approxumately 20% of the cycle and 50% of the cycle.

In order for the PWM spindle speed signal to be turned into direct current (actually a direct voltage



Figure 4.12 – A 20% pulse width modulated signal

is generally used as the input to variable speed drives, but you know what we mean) the pulse signal it must transformed. In essence a circuit is used to find the average of the pulse

width modulated signal. The circuit can be a simple capacitor and resistor or be much more complex depending (a) on how linear you want the relationship between the width and the final output voltage and (b) on the speed of response



Figure 4.13 – A 50% pulse width modulated signal

you need to the changing pulse width. Further details can be found in the discussion and files area of the Mach2DN site and by using PWM converter as a search term to Google or your favorite search engine.

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 using the Motor Clockwise/Counterclockwise outputs.
### Hardware issues and connecting your machine tool

**Note:** Many users have found that PWM and other variable speed spindle drives are often a serious source of electrical noise which can cause problems with the machine axis drives, limit switch sensing etc. If you use such a spindle drive we strongly recommend you to use and optically isolated breakout board.

# 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 (THC)

Mach2 will control a plasma cutting torch and table. Mach2 uses signals from the plasma cutter controller to move the torch up and down in relation to the workpiece to give optimal cutting conditions. The THC measures the tip voltage, which is proportional to the gap distance, and outputs a logic lo signal to one of two pins (Designated Up and Down in the Input Pin Setup) to



Figure 4.14 - Sprung torch mounting showing reference switch

control the Z axis. It will also wait for a signal from the plasma controller to ensure that the arc has stabilised before making the cutting moves. Appendix 4 gives circuits, developed by a long standing Mach2 user, which can be used as they stand or as an example to allow you to develop your own electronics.

The THC is most conveniently set up by mounting the torch itself on a sprung bracket, with a special reference switch, on the Z axis. Figure 4.14 shows a possible arrangement for this.

A reference operation with the torch off will touch the tip on the work, continued downward movement will and operate the switch after a fixed distance. The spring accommodates this overtravel. The torch is the retracted by this fixed distance plus the desired gap for the piercing operation that will start each cut. This is conveniently set as Z = 0. Alternatively the top of the material can be set as zero and the initial pierce height and starting cut height be included in the part program code.

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**. This is because the high voltages and large currents being used by the torch can be dangerous and cause difficulties in the PC and axis drives because of electrical noise. The correct use of a star grounding strategy is vital for success. A web search will reveal a product or products specially designed for use with Mach2.

# 4.10 Digitise probe

Mach2 can be connected to a contact 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 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). If the probe 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.

A development feature is implemented to allow the use of a laser probe.

# 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 4.16. 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.

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



Figure 4.15 - Quadrature signals

phototransistor through the ruling would give a signal like A in figure 4.15. 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 analog 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



Figure 4.16 - Glass scale encoder (awaiting installation)

### Hardware issues and connecting your machine tool

supply (often 5 volts) for the lights and any driver chips in them.

### 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 display the actual speed of the spindle, 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, does an EStop or fails in some way. You can use this signal to charge a capacitor through a diode pump (hence the name) whose output enables your axis and spindle drives etc.

# 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.

There are three OEM trigger signals which can be used to call any of the OEM functions which are normally attached to a screen button.

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 and, although a keyboard emulator will help giving more inputs, these cannot be used for all functions. Full details are given in chapter 5.

# Hardware issues and connecting your machine tool

This page intentionally not used.

# 5. Configuring Mach2 for your machine 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 stores this information in a file which you can view. The Mach Developers Network has a set of "worksheets" that you can print which will take you step-by-step through configuration and give you a written record of the process.

# 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

Engine Configuration Ports & Pins		<b>X</b>
Engine Configuration Ports & Pins         Printer Port Input Pins       Input Emulation         Printer Port Setup and Axis Selection Page       Input Signals Selection         Printer Port # 1       Printer Port # 2         Image: Printer Port # 1       Printer Port # 2         Image: Printer Port # 1       Printer Port # 2         Image: Printer Port # 1       Printer Port # 2         Image: Printer Port # 1       Printer Port # 2         Image: Port Enabled       Image: Port Enabled         Image: Port Enabled       Image: Port Enabled         Image: Port Port # 2       Image: Port Enabled         Image: Port Enabled       Image: Port Enabled         Image: Port Enabled       Image: Port Address         Prints 2 · 9 and 1,14,16,17 are outputs while pins 10 · 15 are inputs on each port       Image: Port Address         Image: Port Address       Image: Port Address       Image: Port Address         Prints 2 · 9 and 1,14,16,17 are outputs while pins 10 · 15 are inputs on each port       Image: Port Address         Image: Port Port Port Port Port Port Port Port	Printer Port Output Pins   ction Page   Output	Encoder Pin inputs Signals Selection Page
Kernal Speed C 25000Hz C 45000Hz	οκ	Cancel Andro

Figure 5.1 - Ports and Axis selection tab

one is as shown in figure 5.1.

# 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 boxes for 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 your 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 Hz 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.

The demonstration version will only run at 25,000 Hz. If Mach2 is forcibly closed then on re-start it will automatically revert to 25,000 Hz operation. The actual frequency in the running system is displayed on the standard Diagnostics scree.

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.

Engine Configuration Port	s & Pins		
Printer Port Input Pins Printer Port Setup and Axis !	Input Emulation Selection Page   Input Signals	Printer Port Output Pins	Encoder Pin inputs Signals Selection Page
Output Options     Digitise Trigger Pulse     Enable Line 1     Enable Line 2     Enable Line 3     Enable Line 4     Enable Line 5     Enable Line 6     Check box to     16 Pins a	Relay activation signals External Activation 1 External Activation 2 External Activation 3 Charge Pump Safety Line b indicate signal is to be used re available after axis and port selection.		
		ок [	Cancel Apply

Figure 5.2 Output Signals selection

## 5.3.1 Output signals to be used

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

You will probably only want to use one Enable output (as all the axis drives can be connected to it). Indeed if you are using the charge pump/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.

5-3

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.

Printer F
-----------

Figure 5.3 - Input signal selection

## 5.3.2 Input signals to be used

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

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.

The *Activation input #4* can be used to connect an external pushbutton switch to implement the Single Step function.

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

Check *Index Pulse* if you have a sensor that gives 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.

Check the OEM Trigger inputs if you want electrical signals to be able to call OEM button functions without a screen button needing to be provided.

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 by not 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.

		-	200000	
Printer Port Setup and /	Axis Selection	n Page   Divide a Divide	Input S	Ignals Selection Page
Low Active	Port #	Pin #		1
Digitize Trigger Pulse		0 Low active	Port	Pin
Finable line 1	1	0 T X Axis Step	1	3
Enable line 2	0	0 TXAxis Direction	1	2
🔽 Enable line 3	0	0 TY Axis Step	1	5
厂 Enable line 4	0	0 T Y Axis Direction	1	4
📕 Enable line 5	0		1	7
📕 Enable line 6	0	0 E Z Avis Direction	1	6
📕 External Activation Pulse	#1 1		0	
External Activation Pulse	#2 1			
External Activation Pulse	#3 1	I spindle Dir	lo	lo
Low active Port	Pin	Low active Port	Pin	or 1 11 - r - 1
A Axis Step	0	C Axis Step	0	checked box indicates low active (IVDC)
A Axis Direction 1	0	CAxis Direction	0	Selection Greyed out when
□ B Axis Step	0	Charge Pump Safety 1	17	inactive
R Axis Direction	0			123456789141617
				Only
				( <b></b>

Figure 5.4 - Parallel Port Output Pins assignment

## 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.

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. The logic convention is determined by the designer of the drive or breakout board to which you are connecting. 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

You now define where input signals come from. There are two possibilities:

- Signals are connected to a keyboard emulator
- Signals are connected to one of the in of a parallel port

While the second case is more common and essential for some signals we will look at the emulator configuration first as its settings override those for the parallel port(s). A keyboard emulator can also be used to generate keycodes to "press" on-screen buttons and to be used as jogging hotkeys. Some inputs can be used to generate as signals and others on the same emulator can be hotkeys. These hotkey functions are described in chapter 8.

### If you are not using a keyboard emulator then you can skip the next section.

## 5.4.2.1 Emulator setup

You will first need to program the emulator itself to map its input connections into the virtual keystrokes which you want to use.

As the first thing that Mach2 does with incoming keystrokes is to filter out those whose codes are defined as emulator inputs and use them to set the internal state of the input you cannot use such key codes for any other function. As a rather extreme example, if you set up the code for the key digit 1



Figure 5.5 – A keyboard emulator card

as the X home switch then you could never enter a number with a "1" in it into any DRO or the MDI line!

Most emulators were originally designed for use with games software and have a default configuration conforming to the MAME standard. This explains the odd naming of terminals on emulators – like Coin 1 etc. It is probable the MAME codes will clash with keys you want to use to control Mach2 screens (e.g. as screen button hotkeys). So the first thing you will need to do is to assign your own keycodes to input terminals and program the micro-controller in your emulator. A good strategy is to use codes that the keyboard cannot produce.

If you are using an Ultimarc IPAC keyboard emulator the the KeyGrabber utility will automatically program the IPAC with a set of codes which will not conflict with your actual keyboard.

If you wish to program an emulator by hand then the utility program Mach2ScreenTweak has an option for exporting all the information in a screen layout to a CSV file. You can inspect this with Microsoft Access or Excel (at a pinch even Notepad) and discover the complete list of shortcut hotkeys that are used by the screen layout or layouts that you want to use. Armed with this list you can allocate unused codes to your emulator inputs and program the emulator.

It is **very important** to connect the signals and test this programming now. If you do not do so, if you find that Mach2 does not respond as expected, then you will have no idea if it is the hardware which is faulty or if the emulator or Mach2 configuration is incorrect.

Figure 5.6 shows the dialog for defining inputs via an emulator.

Check the entries inputs which you have connected to the emulator and define the hotkey Mach2 scancode that the emulator will generate. For details of how to calculate a code by hand see Scancodes in chapter 8.

Engine Configuration	on Ports	& Pins					×
Printer Port Setu Printer Port Input	up and Axis S Pins	election Page   Input Emulation	Input Signals	Selection Page   Printer Port Outp	Output S ut Pins	ignals Selection Page Encoder Pin inputs	
Emulate On Limit X Plus Limit X Minus Limit Y Plus Limit Y Minus Limit Z Plus Limit Z Plus Limit A Plus Limit A Plus Limit B Minus Limit B Minus Limit B Minus Limit C Plus Limit C Plus Home Y		Emulate On Home X Home Z Emergency Stop Limits OverRide Activation 1 Activation 2 Activation 3 Activation 4 Home A Home B Home C		Emulate On Torch heigi Torch up Torch Dow OEM Trigge OEM Trigge OEM Trigge Emulator Test Box Test bo Last Key Down Last Key Up	HotKey ht On   n   er1   er 2   er 3   x Active		
					ок	Cancel Appl	y .

Figure 5.6 – Hotkey definitions for keyboard emulator signals

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

When you have completed the configuration of signals input via the parallel port in the next section you can use the LEDs on the diagnostics page to check that your actual signals give the expected result on the LEDs. In particular you will need to make sure that the correct Active Hi or Lo logic convention applies to each the pin. See Testing below for details.

### 5.4.2.2 Parallel input setup

Now define the input pins on the *Printer Port Input Pins* tab. Any settings for signals defined as coming from the keyboard emulator will have no effect of this screen. This is illustrated in figure 5.7.

If you are sharing the home 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 are controlling a plasma torch then you need to define the pins used to interface to it. The *Torch Up* and *Torch Down* signals are used by Mach2, when in THC mode, to control the position of the Z axis. The signals are derived from the arc voltage of the plasma cutter and drive the Z axis as a "bang-bang" servo. The parameters of this are set in the Plasma Torch Height control family – see chapter 6 for details. The *Torch Height On* input should

Output Signals Selection Page       Printer Port Input Pins       Input Emulation       Printer Port Output Pins       Encoder Pin input Pins         Low Active       Port #       Pin #       Low Active       Port #       Pin #       Torch height On 1       10         Imput Signals Selection Page       Port #       Pin #       Low Active       Port #       Pin #       Torch height On 1       10         Imput Y Plus       1       0       Imput Signals OverRide       1       10       Torch up       1       11         Imput Y Plus       1       0       Imput Signals OverRide       1       10       Torch up       1       11         Imput Y Plus       1       12       Activation 1       1       10       OEM Trigger 2       1       10         Imit Z Plus       1       13       Activation 2       1       10       OEM Trigger 2       1       10         Imit A Plus       1       0       Activation 3       1       10       OEM Trigger 3       1       10         Imit A Minus       0       Digitize Point       1       10       OEM Trigger 5       1       10         Imit A Minus       0       Home A       1       10       OEM Trigger 5	Printer Port Setup a	nd Axis Selection Page		Input Signals Selecti	ion Page
Low Active         Port #         Pin #         Low Active         Port #         Pin #         Cow Active         Port #         Pin #         Torch height On         1         10           I Limit X Plus         1         0         I         I         0         I         10         I         10           I Limit X Plus         1         0         I         I         10         I         10         I         10           I Limit X Plus         1         0         I         I         10         I         10         I         12           I Limit Z Plus         1         12         Activation 2         1         10         OEM Trigger 2         1         10           I Limit A Plus         1         0         Activation 3         1         10         OEM Trigger 3         1         10           I Limit A Plus         1         0         Activation 4         1         10         OEM Trigger 5         1         10           I Limit B Plus         1         0         I Home A         1         10         OEM Trigger 5         1         10           I Limit B Minus         1         0         Home A         1         10	Dutput Signals Selection Page	Printer Port Input Pins Input I	mulation	Printer Port Output Pins	Encoder Pin input
Limit C Minus       1       0       Home C       1       10       Timing Mark       1       10         Home Y       1       10       Checking a box indicateds a signal is active when low. (0VDC)       Signals not selected are greyed out is active when low. (0VDC)       Signals not selected are greyed out prin numbers must be 10,11,12,13, and 15 or Port numbers must be 1 or 2 only!!!	Low Active       Pott #         Limit X Plus       1         Limit X Minus       1         Limit Y Plus       1         Limit Y Plus       1         Limit Z Plus       1         Limit B Plus       1         Limit B Plus       1         Limit C Plus       1         Limit C Minus       1         Limit C Minus       1         Home Y       1         Home Z       1	Check signal to signify low and Pin # Low Active Port Free Emergency Stop 1 Checking a box ind Free Emergency Stop 1 Checking a box ind Checking a box ind Free Emergency Stop 1 Checking a box ind Free Emergency Stop 1 Free Emergency Stop 1 Fr	tive # Pin # 10 10 10 10 10 10 10 10 10 10	Low Active Torch height On Torch up Torch Down OEM Trigger 1 OEM Trigger 2 OEM Trigger 3 OEM Trigger 3 OEM Trigger 5 OEM Trigger 5 OEM Trigger 5 Index Timing Mark signals not selected Pin numbers must be 10,1 Port numbers must be	Port # Pin # 1 10 1 11 1 12 1 10 1 10

Figure 5.7 - Parallel Port Input Pin assignment

be made active by the plasma unit when it has an arc established. It enables the actual operation of height control servo and allows the part program, which is held up at the M3 command which turns on the torch, to continue running.

The OEM Trigger signals allow physical switches to be connected via the parallel port that can issue the OEM control codes that are generally attached to buttons. The codes to be issued are defined, rather strangely as they are nothing to do with keyboard shortcuts, on the Configure>Set Axis Hotkeys dialog.

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.

## 5.4.3 Encoder inputs

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.** 

## 5.4.4 Assign External Activation Outputs

Display the dialog Configure>Output Devices. This will be like figure 5.8. Use the radio buttons to allocate the spindle and coolant options to the required outputs. With a plasma torch Activation Output 1 should be used to enable the plasma controller

Spindle				OK
	Ext Act 1	Ext Act 2	Ext Act 3	
CW Signal	æ	C	C	Lancel
CCW Signal	C	æ	C	
Not Used	Spinup/dow	n time 2	Seconds	
Not Used Flood Mist	Spinup/dow	n time 2	Seconds	
Not Used Flood Mist	Spinup/dow Ext Act 1	n time 2 Ext Act 2	Seconds Ext Act 3	]
Not Used Flood Mist	Spinup/dow Ext Act 1	n time 2 Ext Act 2	Seconds Ext Act 3	
Not Used Flood Mist Flood Control Mist Control	Spinup/dow Ext Act 1	n time 2 Ext Act 2	Seconds Ext Act 3	

Figure 5.8 - Output Devices dialog

(i.e. it should "operate" the switch on the torch – probably by a relay contact across the switch circuit).

## 5.4.5 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 manual switches such as Home 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 home 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 home 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 multimeter 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 (a) the signals in and out of the computer are incorrect (i.e. Mach2 is not doing what you want or expect) or (b) 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) or 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.

Set Default Units for Se 🔀
Units for Motor Setup Dialog MM's C Inches
Figure 5.9 - 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 5.9). Once you have made a choice 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 your breakout board/computer. You are about to mix high power and computing so it is better to be safe than smoky!

## 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 three points in turn then bring them together.

### 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 single start screw is  $1 \div 8 = 0.125$ ")

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**.

Now you can calculate the *screw revs per unit* 

screw revs per unit =  $1 \div$  effective screw pitch

If the screw is directly driven from the motor then this is the motor revs per unit. If the motor has a gear, chain or belt drive to the screw with  $N_m$  teeth on the motor gear and  $N_s$  teeth on the screw gear then:

motor revs per unit = screw revs per unit  $x N_s \div N_m$ 

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  $8 \times 48 \div 16 = 24$  (**Hint**: keep all the figures on your calculator at each stage of calculation to avoid rounding errors)

As a metric example, suppose a two start screw has 5 millimetres between thread crests (i.e. effective pitch is 10 millimetres) and it is connected to the motor with 24 tooth pulley on the motor shaft and a 48 tooth pulley on the screw. So the *screw revs per unit* = 0.1 and motor revs per unit would be  $0.1 \times 48 \div 24 = 0.2$ 

For a rack and pinion or 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. For a rack find its tooth pitch. This is best done by measuring the total distance spanning 50 or even 100 gaps between teeth. Note that, because standard gears are made to a diametral pitch, your length will not be a rational number as it includes the constant  $\pi$  (pi = 3.14152...).

For all drives we will call this tooth pitch.

If the number of teeth on the pinion/sprocket/pulley on the primary shaft which drives the rack/belt/chain is  $N_{\rm s}$  then:

*shaft revs per unit* =  $1 \div (tooth pitch \times N_s)$ 

So, for example with a  ${}^{3}/{}_{8}$ " chain and a 13 tooth sprocket which is on the motor shaft then 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 multiply the motor revs per unit by the reduction ratio of the gearbox.

*motor revs per unit* = *shaft revs per unit*  $x N_s \div N_m$ 

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 Calculating motor steps per revolution

The basic resolution of all modern stepper motors is 200 steps per revolution (i.e. 1.8° per step). Note some older steppers are 200 steps per rev.

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

## 5.6.1.3 Calculating 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) while 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 count from the motor encoder (thus giving 1200 steps per rev for a 300 CPR 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 5.10 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* **now**.



Figure 5.10 - Motor tuning dialog

## 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)

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<sup>2</sup> 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 trials. 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 then 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 incorrectly or are trying to drive it much too fast. The labelling of stepper wires (especially 8 wire motors) is sometimes very confusing. You will 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). If you have any troubles here then you will be glad of 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 maximum 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 Mach1/Mach2 Yahoo! 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 have suffered. 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 co-ordinated 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 co-ordinates 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. Strictly this should be mounted in the toolholder but for a conventional mill you can use the frame of the machine as the spindle does not move relative to the frame in the X-Y plane.

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 DTI probe touches it. Ensure you finish by a move in the minus X direction.

Rotate the bezel to zero the reading. This is illustrated in figure 5.11.

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

Move the table 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 5.11 - Establishing a zero 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 5.12 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.



Figure 5.12 - Gage block in position

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 L50 (run subroutine 50 times)
M30 (stop)
O1234
G1 X4
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 direction, 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.

If you do not have motor speed control the choose Pulley 4 with a high maximum speed like 10,0000 rpm and . This will prevent Mach2 complaining if you run a program with a S word asking for say 6000 rpm.



Figure 5.13 - Pulley spindle drive

G20,G21 Control G20,G21 Control Control Control Control Control G2 - Inhibit C2 - Inhibit C2 - Inhibit C3 - Inhibit C3 - Inhibit C4 -	Angular Properties Angular Properties Axis is Angular C-Axis is Angular C-Axis is Angular Unchecked for Linear Pgm End or M30 or Rewind Turn off all outputs E-Stop the system Perform G92.1 Remove Origin Offsets Remove Tool Offset	Program Safety Program Safety Lockout This disables program translation while the External Activation #1 input is activated. PWM Pulley Control Current Pulley Set Pulley Ratio #1 Pulley Ratio #2 Pulley Ratio #3 Pulley Ratio #3 Pulley Ratio #4 Pulley Ratio #4 Pulley Ratio #4	imum Speed 5 %
<ul> <li>Laser Mode. freq by Feedrate %</li> <li>Use Spindle Feedback for Feeds</li> <li>Persistent DROs</li> <li>Disable Gouge/Concavity Checks Input</li> <li>Debouce Interval: 100</li> <li>Index Debounce</li> </ul>	<ul> <li>Radius Comp Off</li> <li>Plasma Mode</li> <li>No Angular Descrimination</li> <li>FeedOverRide Persist</li> <li>Unchecked resets on Stop File</li> <li>Persistant Z-Inhibit</li> <li>Unchecked resets on rewind</li> </ul>	G04 Dwell param in MilliSeconds instea C:\WINDOWS\notepad.exe Serial Output ComPort # 1 BaudRate 9600 M01 Control ✓ Stop on M1 Command	d of Seconds Browse Editor OK

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 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 5.14). 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.

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  $720 \text{ rpm} (3600 \div 5)$ . 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 ( $3600 \times 4$ ). 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.

The *Minimum Speed* value applies equally to all pulleys and is expressed as a percentage of the maximum speed and is, of course, also the minimum percentage PWM signal ratio. If a speed lower than this is requested (by the S word etc.) then Mach2 will request you to change the pulley ratio give a lower speed range. For example, with a maximum speed of 10,000 rpm on pulley 4 and a minimum percentage of 5% then S499 would request a different pulley. This feature is to avoid operating the motor or its controller at a speed below its minimum rating

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 value is compared with the maximum speed for the currently selected pulley. If the requested speed is greater than the maximum then an error occurs.

• Otherwise the percentage of the maximum for the pulley that has been 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.

Define External Activation signals in Ports and Pins and Configure>Output Devices to switch the PWM controller on/off and, if required, to set the direction of rotation.

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 pins must be connected to your motor drive electronics. *Apply* the changes.

Define External Activation signals in Ports and Pins and Configure>Output Devices to switch the spindle motor controller on/off if you wish to take power off the motor when the spindle is stopped by M5. It will not be rotating anyway of course as Mach2 will not be sending step pulses but, depending on the driver design, may still be dissipating power.

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 position. 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 vou to define what happens when a reference operation (G28, G28.1 or a screen button) is performed. Figure 5.15 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 is 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)

#### **Reference Configuration** Referencing Speeds X 20 % 20 % 20 Y 20 % 20 7 % 20 9 Speed as a percentage of Max Auto-Zero DRO's when homed Use full speed when referenced. All referencing is to Home pin selected. Reference direction ΓX ΓZ E B F A ΓC E Y Negative Axis-- when checked 0K

Figure 5.15 - Referencing

## 5.7.1.2 Position of home switches

As described above the home 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 home switches are defined on the Corrections screen.

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

# 5.7.2 Configure Encoders

### 5.7.2.1 Encoders for position display

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 (generally as defined in Setup Units) of movement. This is the number of "lines" on the encoder scale per unit (not the resolution

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

Figure 5.16 - Encoder configuration

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 5.16.

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.

## 5.7.2.2 Encoder for jogging

An encoder, which must be Encoder 3, can be used to generate pulses (i.e. as an MPG) which will jog a chosen axis. For full details see the Jog Controls family.

In this application the shaft usually has a detent mechanism on it so you move in a series of "clicks". Each click should move by one jog increment so you should configure Encoder 3 so that its Counts per Unit is the counts per "click". For example a 1000 cycles per rev. encoder will generate 4000 "counts". If you have a 100 tooth detent wheel then Counts per Unit should be set to 40.

You can also interface an MPG encoder via the keyboard emulator if you use suitable emulator software such as KeyGrabber. This saves on the scarce parallel port input pins.

# 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>Backlash 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 is made. The speed at which this movement is to be made is also specified. See figure 5.17

**Note:** These settings are only used when backlash compensation is enabled on the Configure>Initial State dialog.

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

Bac V	klash Distance i	n units
Ŷ	0	
z	0	
А	0	
В	0	
С	0	
Bac	klash Speed %	of Max
	20	

Figure 5.17 - Backlash configuration

Slave Control	×
Primary Axis	Slave Axis
• None	C A - Axis
C X - Axis	C B - Axis
C Y - Axis	C C - Axis
C Z · Axis	
	OK

Figure 5.18 - Slaving configuration

with the speed and acceleration being determined by the "slower" of the two.

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 home switch positions and any racking which has occurred be eliminated.

Although Mach2 keeps the master and slaves axes in step, the DRO of the slave axis will not display offsets applied by the Tool table, fixture offsets etc. Its values may thus be confusing to the operator. We therefore recommend that you use the Screen Designer to remove the axis DRO and related controls from all the screens except Diagnostics. Save As the new design with a name other than the default and use the Layouts>Load Layout menu to load it into Mach2.

#### 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.

0	X Axis	15	×	1
0	Y Axis	15	Y	1
0	Z Axis	999999	Z	1
Min		Max	А	pproach safe



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 5.19.

The Approach Safety values are the distance from the limits at which the approach speed is automatically reduced. If this value is set too big you will reduce the effective working area of the machine. If they are set too small then you risk hitting the hardware limits.

The defined limits only apply when switched on using the Software Limits toggle button see Limits and Miscellaneous control family for details.

If a part program attempts to move outside a soft limit then it will raise an error.

#### 5.7.6 **Configure Initial State**

Configure>State opens a dialog which allows you to define the modes which are active when Mach2 is loaded. It is shown in figure 5.20.

Motion mode: Constant velocity sets G64, Exact Stop sets G61. For details of these option see Constant Velocity and Exact Stop in chapter 11.

Distance mode: Absolute sets G 90, Inc sets G91

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

I/J Mode: 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 always fail to display or to cut properly (especially obvious by them being too big if they are far from the origin) then the IJ mode is not compatible with you part program. An error in this setting is the most frequent cause of questions from users.

**Backlash** (compensation): can be set on of off here (you might have looked for it on Configure Logic!). The values used are on the Configure Backlash dialog.

**Initialisation String:** is a set of valid G-codes to set the desired initial state of Mach2 when it is started. These are applied after the values set in the radio buttons above so may override them. Use the radio buttons wherever possible to avoid confusion. If Init of ALL "Resets" is checked then these codes will be applied however Mach2 is rest –e.g. after an EStop condition.

**ToolPath** (colors): are defined here. The brightness of each of the colours Red Green Blue are set on a scale 0 to 1. **Hint:** Use a program like Photoshop to make a color which you like and divide its RGB values by 255 (it uses the scale 0 to 255) to get the values for Mach2. The *Virtual Compass* checkbox displays a "key" for the direction of the axes which you might as well always have on.

Modal States Motion Mode C Constant Velocity © Exact Stop	ToolPath Red Green Blue BackGround Color 0 0.15 0.45			
Distance Mode IJ Mode	Rapid Color Feedrate G1 color G2,G3 Color Enhance Color	1 0.89	0.68	0.2
Active Plane of Movement				
Backlash @ Off @ On		1	1	1
itialization String 「Init on ALL "Resets" 680	Colors range from 0	(none) t	o 1 (full)	
og Increments in Cycle Mode			r	

Figure 5.20 - Initial State configuration

# 5.7.7 Configure other Logic items

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

**G20/G21 Control**: If Lock DROs to set up units is checked then even 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:** Sets a limit to the lowest value of Z permitted. For example if Z inhibit is set to minus 2.1 (- 2.1) and enabled then G0 Z-2.5 will leave the controlled point at Z = -2.1.

**Tool change:** An M6 tool change request can be ignored or used to call the M6 macros (q.v.). If Auto Tool Changer is checked then the M6Start/M6End macros will be called but Cycle Start does not need to be pressed at any stage.

**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 M30**: defines action(s) to take place at end of your part program. Check the required functions. *Caution:* Before checking the items to remove offsets and to perform G92.1 you should be absolutely clear on how these features work of you may find that the current position has coordinates very different from what you expect at the end of a 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.

**Persistent DROs:** If checked then the axis DROs will have the same values as when Mach2 is closed down. Note that the positions of the physical axes are unlikely to be preserved if the machine tool is powered down, especially with micro-stepper drives.

Logic Configuration			
G20,G21 Control G20,G21 Control Control Lock DRO's to setup units Z - Inhibit G Z - Inhibit On Max Depth 0 Units Tool Change G Ignore Tool Change G Stop Spindle. Wait for Cycle Start. AutoTool Changer PWMBase Freq. 7 Laser Mode. freq by Feedrate % Use Spindle Feedback for Feeds	Angular Properties Angular Properties A-Axis is Angular C-Axis is Angular Unchecked for Linear Pgm End or M30 or Rewind Turn off all outputs E-Stop the system Perform G92.1 Remove Origin Offsets Remove Tool Offset Radius Comp Off	Program Safety Program Safety Lockout This disables program translation while the External Activation #1 input is activated. PWM Pulley Control Current Pulley Set Pulley Ratio #1 Pulley Ratio #1 Pulley Ratio #2 Pulley Ratio #3 O Pulley Ratio #3 O Culley Ratio #4 D Context Current Pulley Ratio #4 Current Pulley Ratio #4 Current Pulley Ratio #4 Current Pulley Ratio #4 Current Pulley Ratio #4 Pulley Ratio #4 Current Pulley	nimum Speed
Persistent DR0s     Disable Gouge/Concavity Checks     Input     Debouce Interval: 100     Index Debounce 0	<ul> <li>Persistent DROs</li> <li>Plasma Mode</li> <li>✓ Disable Gouge/Concavity Checks</li> <li>✓ No Angular Descrimination</li> <li>✓ PeedOverRide Persist</li> <li>Unchecked resets on Stop F</li> <li>Persistant Z-Inhibit</li> <li>Unchecked resets on rewind</li> </ul>	C:\WINDOWS\notepad.exe Serial Output ComPort # 1 BaudRate 9600 M01 Control I Stop on M1 Command	Editor

Figure 5.12 – (duplicate) Logic Configuration dialog

**Disable Gouge/Concavity checks:** If this option is off then, during cutter compensation (G41 and G42), Mach2 will check if the tool diameter is too large to cut "insider corners" without gouging the work. Check the box to disable the warning.

**Plasma Mode:** This controls Mach2's implementation of constant velocity moves to suit the characteristics of plasma cutters.

**No Angular Discrimination:** This is also only relevant to constant velocity working. When unchecked Mach2 treats changes of direction whose angle is greater than the value set in the *CV Angular Limit* DRO as exact stop (even if CV mode is set) to avoid excessive rounding of sharp corners. Full details of Constant Velocity mode are given in chapter 11.

**FeedOveride Persists:** If this option is checked then the selected feed override will be retained at the end of a part program run.

**Persistent Z-inhibit:** If this option is checked then the Z-inhibit value entered will be retained on a program rewind, otherwise it will be cleared. This setting us used in conjunction with the automatic roughing out of a part be Mach2 decrementing the Z-inhibit and repeating a program a give number of times.

**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  $(100 \div 35000 = .0029 \text{ secs})$ . The Index pulse and the other inputs have independent settings.

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

**G04 Dwell param in Milliseconds:** If you check this then the command G4 5000 will give a Dwell in running of 5 seconds. If the control is unchecked it gives a dwell of 1 hour 23 minutes 20 seconds!

**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.

**Serial output:** Defines the COM port number to be used for the serial output channel and the baud rate at which it should output. This port can be written to from VB script in a macro and can be used to control special functions of a machine (e.g. LCD display, tool-changers, axis clamps, swarf conveyor etc,)

# 5.8 How the Profile information is stored

When the Mach2.exe program is run it will prompt you for the Profile file to use. This will generally be in the Mach2 folder and will have the extension .XML. You can view and print the contents of Profile files with Internet Explorer (as XML is mark-up language used on web pages)

Shortcuts are set up by the system installer to run Mach2.exe with default Profiles for a Mill and for Turning (i.e. Mach2Mill and Mach2Turn. You can create your own shortcuts each with a different Profile so one computer can control a variety of machine tools.

This is very useful if you have more than one machine and they require different values for the motor tuning, or have different limit and home switch arrangements.

You can either run Mach2.exe and choose from the list of available profiles or you can set up extra shortcuts that specify the profile to use.

In a shortcut the profile to load is given in the "/p" argument in the Target of the shortcut properties. As an example you should inspect the Properties of the Mach2Mill shortcut. This can be done, for example, by right clicking the shortcut and choosing Properties from the menu.

An .XML file for a profile **can** be edited by a text editor like Notepad but you are **strongly** advised not to do this unless you are fully conversant with the meaning of each entry in the files. Notice that some tags (e.g. the screen layout) are only created when a built-in default value is overridden using Mach2 menus. It is much safer to use Mach2's configuration menus to update the XML profiles.

This page intentionally not used

# 6. Mach2 controls and running a part program

This chapter is intended for reference to explain the screen controls provided by Mach2 for setting up and running a job on the machine. It is of relevance to machine operators and for part-programmers who are going to prove their programs on Mach2.

# 6.1 How the controls are explained in this chapter

Although at first sight you may feel daunted by the range of options and data displayed by Mach2, this is actually organised into a few logical groups. We refer to these as Families of Controls. By way of explanation of the term "control", this covers both buttons and their associated keyboard shortcuts used to operate Mach2 **and** the information displayed by DROs (digital read-outs) or LEDs (light emitting diodes).

The elements of each control family are defined for reference in this chapter. You should, however, note that the actual screens of the standard Mach2 do not always include every control of a family when the family is used. This may be to increase readability of a particular screen or to avoid accidental changes to the part being machined in a production environment.

The families are explained in order of importance for most users.

Screens can be modified or be designed from scratch by you so you can add any controls to a particular screen if your application requires this. For details see chapter 8.



Figure 6.1 - Screen switching control family

# 6.1.1 Screen switching controls

These controls appear on each screen. They allow switching between screens and also display information about the current state of the system.

## 6.1.1.1 Reset

This is a toggle. When the system is Reset the LED glows steadily, the charge pump pulse monitor (if enabled) will output pulses and the Enable outputs chosen will be active.

## 6.1.1.2 Labels

The "intelligent labels" display the last "error" message, the current modes, the file name of the currently loaded part program (if any) and the Profile that is in use.

## 6.1.1.3 Screen selection buttons

These buttons switch the display from screen to screen. The keyboard shortcuts are given in brackets. For clarity they are in upper-case. You should not, however, use the shift key when pressing the shortcut.

# 6.1.2 Axis control family

This family is concerned with the current position of the tool (or more precisely, the controlled point).

The axes have the following controls:

### Mach2 controls and running a part program



Figure 6.2 - Axis control family

## 6.1.2.1 Coordinate value DRO

These are displayed in the current units (G20/G21) unless locked to the setup units on the Configure>Logic dialog. The value is the coordinate of the controlled point in the displayed coordinate system. This system (Incl G92, Fixture or Absolute Machine) is selected using the Coordinate System Display mode family.

You can type a new value into the coordinate DRO. This will apply an appropriate G92 offset to make the controlled point in the current coordinate system be the coordinate you have set. The Display Mode (q.v.) does not affect how the value is interpreted but after you have entered it the DRO shows the result of your implied G92 in the current display mode. **Because of the complexity of this process, caution is advised in typing coordinate data into axis DROs.** 

## 6.1.2.2 Referenced

The LED is green if the axis has been referenced (i.e. is in a known actual position)

Each axis can be referenced using the Ref button.

- If no home/reference switch is defined for the axis, then the axis will not actually be moved but, if *Auto Zero DRO when homed* is checked in Configure>Referencing, then the absolute machine coordinate of the current position of the axis will be set to the value defined in the Reference switch Location control family (q.v). This is most usefully zero.
- ◆ If there is a home/reference switch defined for the axis and it is not providing an active input, then the axis will be moved in the direction defined in Configure>Referencing until the input becomes active. It then backs off a short distance so that the input is inactive. If the input is already active then the axis just moves the same short distance into the inactive position. If *Auto Zero DRO when homed* is checked in Configure>Referencing then the absolute machine coordinate of the current position of the axis will be set to the value defined in the *Reference switch Location* control family (q.v).

The *Ref All* button is equivalent to referencing all the axes.

### Mach2 controls and running a part program

### 6.1.2.3 Scale

Scale factors for any axes can be set by G51 and can be cleared by G50. If a scale factor (other than 1.0) is set then it is applied to coordinates when they appear in G-code (e.g. as X words, Y words etc.). The Scale LED will flash as a reminder that a scale is set for an axis. The value defined by G51 will appear, and can be set, in the Scale DRO. Negative values mirror the coordinates about the relevant axis.

The G50 button executes a G50 command to set all scales to 1.0

## 6.1.2.4 Diameter/Radius correction

Rotary axes can have the approximate size of the workpiece defined using the Rotational Diameter control family. This size is used when making blended feedrate calculations for co-ordinated motion including rotational axes. The appropriate LED in this family indicates that a non-zero value is defined.

## 6.1.2.5 G92 button

The G92 button is used to set a G92 offset for the axis making its current coordinate zero. It is exactly equivalent to issuing an MDI G92 X0 (Y0 or Z0 as applicable) command

# 6.1.3 Coordinate system display mode family

The axis DROs display the coordinates of the controlled point. The coordinate system to be used for this display is selected using this family.

The *Coordinate System* button toggles between *Fixture* mode - which does not display any offset applied by G92 commands (including direct entry of a coordinate to an axis DRO) and *G92* mode which includes both Fixture and G92 offsets in the displayed values. The LEDs indicate the state of the toggle. For further details of offsets see chapter 7.

The *Machine* (sometimes *Mach*) button displays absolute machine coordinates. The LED warns that absolute coordinates are being displayed.

**Note:** This family only alters how the coordinates are displayed on the Axis DROs. The values given in X, Y, Z etc. words in G-code commands (unless G53 is used) and the value typed into an axis DRO **always relate to the current coordinate system -** that is with all fixture, tool and G92 offsets applied.

# 6.1.4 Jogging control family

Whenever the *Jog Mode* button is displayed on the current screen the axes of the machine can be jogged using (a) the jog hotkeys – including an MPG encoder connected via a keyboard emulator. The hotkeys are defined in Configure Axis hotkeys; (b) an MPG handwheel connected to an encoder; (c) joysticks interfaced as USB Human Interface Devices; (d) (if displayed) the simulated trackball icon (see figure 6.4). or (e) as a legacy feature a Windows compatible analog joystick.

If the *Jog Mode* button is not displayed jogging is not allowed for safety reasons.

Using USB HIDs allows very flexible and powerful



control of jogging. Full details are given in the KeyGrabber appendix. Some of the controls described here are redundant if you use KeyGrabber which submits data messages directly to Mach2

### 6.1.4.1 Hotkey jogging

There are two modes. Continuous and Incremental which are selected by the *Jog Mode* button and indicated by the LEDs.

Continuous mode moves the axis or axes at a defined feedrate while the hotkeys are depressed

The continuous jog speed is defined as shown below but this can be overridden by depressing *Shift* with the hotkey(s). An LED beside the Cont. LED indicates this full speed jogging is selected

Incremental mode moves the axis by one increment (as defined by the *Jog Increment* DRO) for each keypress (or Typematic repeated presses). The **current** feedrate is used for these moves. The size of increment can be set by typing it into the DRO or values can be set by cycling through a set of 8 user definable values using the Cycle button. Incremental mode is selected by the toggle button or temporarily selected by holding down Ctrl before pressing the hotkeys. The LEDs indicate the current jogging mode.

### 6.1.4.2 Parallel port encoder handwheel jogging

If Encoder 3 is configured then it can be enabled as a jog wheel by the Encoder Jog Toggle. When enabled it inhibits other jogging methods.

The axis it jogs is indicated by the LEDs and the installed axes are selected in turn by the Cycle button.

After suitable scaling set by Configure>Encoders, each "click" from the encoder requests one incremental jog step (with the distance set as for hotkey incremental jogging). The requests will be queued up so, for slow feedrates, rapid movement of the wheel may mean that the axis moves for some time after the wheel movement has stopped.

This feature is of particular use in making very fine controlled movements when setting up work on the machine.

**Note:** It is probably better to interface an MPG encoder via a keyboard emulator and KeyGrabber.

### 6.1.4.3 Joystick jogging

USB HID joysticks (axis controls) are probably better supported via KeyGrabber. This section describes the legacy feature in Mach2.

The Joystick button will enable a Windows compatible analog "joystick" if fitted. This can be used to control the two axes controlled by the trackball icon (usually X and Y for a mill).

A wireless joystick is a very convenient jogging control and they are often equipped with buttons which can be configured to be hotkeys for things like *Pause, Rewind and Stop.* 

This method of jogging is best for jogging long distances where precise control is not required.

### 6.1.4.4 Jog rate

The jogging speed used with hotkeys in Continuous mode is set as a percentage of the rapid traverse rate for the axis and for the Windows compatible joystick as a percentage of the feed for the given stick deflection by the *Slow Jog Percentage* DRO. This can be set (in the range 0.1% to 100%) by typing into the DRO. It can be nudged in 5% increments by the buttons or their hotkeys.

If a Windows compatible joystick is installed and enabled then the throttle control on it can be used to change the *Slow Jog Percentage* DRO. This option is chosen by the three way

### Mach2 controls and running a part program

*Joystick Throttle* button. The associated LEDs indicate the current throttle function. This is a legacy feature. The Throttle HID axis is best supported through KeyGrabber.

The value in the slow jog percentage DRO does not affect the trackball jogging rate.

### 6.1.4.5 Jogball

Clicking in a quadrant of the "jogball's" frame will give axis motion in the relevant direction(s) whose speed is proportional to the distance of the mouse from the centre. The axes used can be configured using Screen Designer (q.v.) but are usually X and Y for a mill.

## 6.1.4.6 Spindle Speed control family

Depending on the design of your machine, the machine spindle can be controlled in three ways: (a) Speed is fixed/set by hand, switched on and off by hand; (b) Speed fixed/set by hand, switched on and of by M-codes via external activation outputs, (c) Speed set by Mach2 using PWM or step/direction drive.



This control family is only important for case (c).

Figure 6.5 - Spindle speed control family

*Set Spindle RPM* has its value set when an S word is used in a part program. It is the desired spindle speed. It can also be set by typing into the DRO.

It is an error to try to set it (in either way) to a speed greater than that displayed in *Max RPM for this pulley*.

If the Index input is configured and a sensor which generates one pulse per revolution of the spindle is connected to its pin, then the current speed will be displayed in *True Spindle RPM*. This DRO cannot be set.

The *Pulley number* must range from 1 to 4. The maximum speed for each pulley (gear ratio) setting is defined in the Configure>Logic dialog and its value for the chosen pulley displayed in this family. You must tell Mach2 what pulley (gear) you have selected by entering its number into the *Pulley Number* DRO.

Mach2 uses the ratio of the Set RPM to the maximum RPM to determine the PWM markspace ratio to output or the percentage of maximum configured step speed to use depending

on the spindle control option selected in the Ports and Axis tab of Configure>Ports & Pins.

# 6.1.5 Feed control family

There are two feed modes which are chosen by the *Feed Mode* toggle button and indicated by the *Units/Rev* and *Units/Min* LEDs.

## 6.1.5.1 Feed Units per minute

The *Prog Feed* DRO gives the feed rate in current units (inches/millimetres per minute). It is set by the F word in a part program or by typing into the DRO. Mach2 will aim to use this speed as the actual rate of the co-ordinated movement of the tool through the material. If this rate is not possible because of the



Figure 6.6 Feed control family

maximum permitted speed of any axis then the actual feed rate will be the highest achievable.

### 6.1.5.2 Feed Units per rev.

As modern cutters are often specified by the permitted cut per "tip" it may be convenient to specify the feed per revolution (i.e. feed per tip x number of tips on tool). The *Prog Feed* DRO gives the feed rate in current units (inches/millimetres) per rev of the spindle. It is set by the F word in a part program or by typing into the DRO.

A revolution of the spindle can either be determined by the *Set spindle speed* DRO or from the measured speed by counting index pulses. Configure>Logic has a checkbox to define which to adopt.

To employ Feed units/rev, Mach2 must know the value of the chosen measure of the speed of the spindle (i.e. it must have been (a) defined in an S word or by data entered to Set Spindle RPM DRO in the Spindle speed control family or (b) the Index must be connected up to measure actual spindle speed).

Notice that the numeric values in the control will be very different unless spindle speed is near to 1 rpm! So using a feed per minute figure with feed per rev mode will probably produce a disastrous crash.

## 6.1.5.3 Feed display

The actual feed in operation allowing for the co-ordinated motion of all axes is displayed in *Units/min* and *Units/rev*. If the spindle speed is not set and the actual spindle speed is not measured then the *Feed per rev* value will be meaningless.

### 6.1.5.4 Feed override

Unless M49 (Disable feedrate override) is in use, the feedrate can be manually overridden, in the range 20% to 299%, by entering a percentage in the *Feedrate Override* DRO. This value can be nudged (in steps of 10%) with the buttons or their keyboard shortcuts and be reset to 100%. Alternatively if a Windows compatible joystick is enabled then its throttle can be configured by the three way *Joystick Throttle* toggle. The *Feed Rate* LED indicates if it is controlling the *Feedrate Override* DRO.

The *Over Feedrate* DRO displays the calculated result of applying the percentage override to the set feedrate.

## 6.1.6 Program Running control family

These controls handle the execution of a loaded part program or the commands on an MDI line.

## 6.1.6.1 Cycle Start

**Safety warning:** Note that the *Cycle Start* button will, in general, start the spindle and axis movement. It should always be configured to require "two hand" operation and **not** be a single easy to hit hotkey.

### 6.1.6.2 Pause

*Pause* brings the current move to a controlled stop applying decelereation etc. Resumption by Cycle Start is always safe.

### 6.1.6.3 Stop

*Stop* halts axis motion as quickly as possible. It may result in lost steps (especially on stepper motor driven axes) and restarting may not be valid.

### Mach2 controls and running a part program



Figure 6.7 - Program running family

## 6.1.6.4 Single

*Single* when pressed in the stopped state this executes the next line of the part program. When executed in the paused state it is equivalent to Cycle Start.

## 6.1.6.5 Line Number

Line number is the ordinal number of the current line in the G-code display window (starting from 0). Note that this is not related to the "N word" line number.

You can type into this DRO to set the current line.

## 6.1.6.6 Run from here

*Run from here* performs a dummy run of the part program to establish what the modal state (G20/G21, G90/G91 etc.) should be and then prompts for a move to put the controlled point in the correct position to for the start of the line in *Line Number*. You should not attempt to *Run from here* in the middle of a subroutine.

## 6.1.6.7 Set next line

Like Run from here but without the preparatory mode setting or move.

## 6.1.6.8 Tool change request

Unless tool change requests are being ignored (Configure>Logic), on encountering an M6 Mach2 will move to Safe Z and stop, flashing the *Tool Change Req* LED. You continue (after changing the tool) by clicking *Cycle Start*.

## 6.1.6.9 Dwell active

The *Dwell Active* LED lights if a dwell in the run of the part program is taking place as a result of starting the spindle or the program executing a G4 (Dwell)

## 6.1.6.10 M1 Optional Stop

This LED and the button to toggle it is an on screen version of the checkbox in Configure>Logic. If the LED is lit then an M1 command will cause the part program to stop on the following line. It can be resumed by pressing Cycle Start.

## 6.1.6.11 Block Delete

If this LED is lit then any line (also termed a block) of the part program which begins with the character "/" (i.e. a forward slash) will net be displayed in the G-code window and not be executed when the program is run.

## 6.1.6.12 Ignore tool change

If this LED is lit then any M6 request for a tool change will be ignored. If it is not lit then the M6Start and M6End macros will be called to implement the tool change process. This is the on-screen version of a checkbox on the Configure>Logic dialog.

# 6.1.7 Auxiliary (Spindle & Coolant) control family



This family allows manual control of the spindle and coolant and displays the current state of the outputs controlling these machine functions.

# 6.1.8 G-code and Toolpath control family

The currently loaded part program is displayed in the G-code window. The current line is

itimiig · ·	000 O I I 040 I	
·	3	
ELOD.		
gD x1 y0 zD 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		
Simulate Execute Progra	From Here m Extrema ni c1 000	
The second sec	0 + 1.0000 0 + 0.0000	
Stop AINS A Flarge +0.000	<pre>iii +0.0000</pre>	- 「へいたち」 ディア・アイ・アイ
DiBance (999) 000	0.000999.000	
21-cs +0.0000 C Ranko 1.4450 100	0 000.0000	
11/2s +0.0000 Absolute	Coordinates	
ZTUS +0.0000		
4P13 +0.0000	Jog Lollow	
B = (2 +0.0000	Пара	
+0.0000		

Figure 6.9 - Toolpath and G-code display family

highlighted and can be moved using the scroll bar on the window.

The Toolpath display shows the path that the controlled point will follow in the X, Y, Z planes. When a part program is executing the path is overpainted in green. This overpainting is dynamic and is not preserved when you change screens or indeed alter views of the toolpath.

The *Program extrema* data allow you to check the maximum excursion of the controlled point to be reasonable (e.g. not milling the top off the table).

The toolpath display can be rotated by left clicking and dragging the mouse in it. It can be zoomed by shift-left clicking and dragging and can be panned by dragging a right click.

The *Jog Follow* toggle and its indicator LED will move the toolpath display so that as the controlled point is moved by jogging or MDI it will always be the centre of the display.

The *Regen* button will regenerate the toolpath display from the G-code with the currently enabled fixture and G92 offsets.



Figure 6.10 – File control family
# 6.1.9 File control family

These controls are involved with the file of your part program. They should be self-evident in operation.

# 6.1.10 Fixture and tool table control family

Fixture and tool tables can be accessed from the Configure menu and within a part program but it is often most convenient to manipulate them through this family. Refer to chapter 7 for details of the tables and techniques like "Touching".

Fixture Off 1						
Part X Offse	t <u>1.0000</u>	Touch Part A Offset	0.0000 Touch			
Part Y Offse	t 3.2000	Touch Part B Offset	0.0000 Touch			
Part Z Offse	t 0.0000	Touch Part C Offset	0.0000 Touch			
Offset On/OffStock Diamter, Guage Block thickness, etcTool # 3ToolTool # 3Correction						
X Offset	0.000 🌆	uch On/Off Al	t T			
Z Offset	0.000 🚥	uch Save Tables I	here to			
Diameter	12.000	make them per Fixture Ta	manent able			
Tip Radius	0.000	Tool Tab	le .			

Figure 6.11 - Fixture and Tool table control family

Because of the underlying G-code definitions Fixtures and Tool tables work in slightly different ways.

## 6.1.10.1 Fixtures

Mach 2 defines Fixture 0 to mean no offsets are to be applied. Choosing any value from 1 to 255, and entering it in the *Fixture Number* DRO, will turn the offsets of that fixture on. You can change its offsets by typing into the relevant DROs. Values can also be set by moving the axes and clicking *Touch*.

Touching can include "diameter" and gage block corrections from the *Touch Correction* DRO. This is controlled by the *On/Off* toggle with associated indicator LED.

If a fixture is to be persistent in effect from one run of Mach2 to another then the data can be saved using the *Save Fixture Table* button to open the required dialog.

## 6.1.10.2 Tools

Tools are numbered from 0 to 255. The tool number is selected by the T word in a part program or entering the number in the *Tool* # DRO. Its offsets are only applied if they are switched On by the *Offset On/Off* toggle button (G43 and G49)

In Mach2Mill only the *Z offset* and *Diameter* are used. *X Offset*, *Z Offset* and *Tip Radius* are for Mach2Turn which can share a tool table with Mach2Mill.

If a tool's data is to be persistent in effect from one run of Mach2 to another then the data can be saved using the *Save Fixture Table* button to open the required dialog.

## 6.1.10.3 Rotational Diameter control family

As described in the Feedrate control family, it is possible to define the approximate size of a rotated workpiece so the rotational axis speed can be correctly included in the blended feedrate. The relevant diameters are entered in the DROs of this family.

The Axis control Family has warning LEDs to indicated the setting of non-zero values here.

Values are not required if rotary movement is

not to be coordinated with linear axes. In this case a suitable F word for degrees per minute or degrees per rev should be programmed.

# 6.1.11 Plasma Torch Height control family

If you are controlling a plasma torch using the torch height control feature then this family controls its operation.

## 6.1.11.1 The controls

The *Calibrate to Zero* button will move the torch to the reference position defined by the switch on its sprung holder. A correction to this zero value will be needed to set the desired height for piercing. This can easily be done by a G0 input using MDI and then setting the Z axis to zero by a *G92Z0* button, typing zero into the Z axis DRO or MDI of G92 Z0. You will need to establish the correct gap for clean piercing, with minimum cutting tip wear, based on

Rotation DiametersA Axis+ 5.6000ZeruB Axis+ 0.0000ZeruC Axis+ 0.0000Zeru

Figure 6.12- Rotational diameters



Figure 6.13 - THC control family

experience and the documentation of your plasma cutter. Another way to accomplish the same goal is to allow the axis to correct the switch offset value and to zero the axis so Z = 0 at the top of the material. Using this approach, all heights will reference from the nominal top of the material and the pierce height and initial cut height can be controlled programmatically.

The *Enable Toggle* switches on the automatic servo control of the Z axis. Its movement is then based on height information from the plasma unit. The appropriate input pins must have been defined in the Ports and Pins dialog and been connected to the plasma cutter's THC controller.

The Anti-Dive toggle enables use of the Anti-Dive limit. Details of this are given below.

It is easiest to understand the operation of the system by looking at the sequence for a complete cut. We assume that Z = 0 is a suitable height for the start of the piercing operation and the *Enable Toggle* is ON.

## 6.1.11.2 THC in operation

The part program moves the torch at Safe Z to the start of the cut then moves to Z = 0. It then executes M3 ("spindle start") to start the arc. Mach2Mill stops executing the part program and completes the torch switch circuit. Piercing starts. When the THC interface detects that the arc is good it signals this to Mach2Mill which switches on the torch height servo and resumes execution of the part program. This can have a short Dwell (G4) or immediately commence the first move for the cut.

Note: Different brands of plasma cutters have different methods of sensing when the arc has started and to turn on the start circuit. It is not in the scope of this manual to explain how to tap into a specific plasma unit's signal to generate the *Torch On* signal.

Whenever the height servo is on it will sample the Torch Up and Torch Down inputs. If Torch Up is active then the Z axis will move in the plus direction at the rapid feed rate

multiplied by the Correction Speed percentage DRO. Similarly Torch Down will move in the minus Z direction. The up and down signals are derived from the arc voltage. Too low a voltage and the torch must be moved up, too high a voltage and it must be moved down. The actual threshold values are determined by the calibration of the hardware interface between the plasma controller and the PC. The correction from the nominal (usually Z = 0) position is shown in the *Current Correction* DRO. The Z axis DRO will update to reflect the actual position of the axis.

If the cutting speed becomes low, perhaps due to acceleration limitations while turning a sharp corner then the height servo will see an increase in arc voltage and think that the height needs to be reduced. This causes the torch to dive at sharp corners. You can prevent this by defining a minimum actual feed rate in *Anti-Dive limit* and toggling *Anti-Dive* on. Choose an appropriate feedrate by experiment to suit the acceleration of your X and Y axes and plasma unit characteristics.

The *Highest* and *Lowest* value DROs are to minimise the risk of damage to the torch if a cut is made over the edge of the material or into a hole or if the arc goes out due to dirty material or the like. If the Current Correction value attempts to go outside these limits then the Torch Up and Torch Down signals are ignored and the program execution continues.

When setting up your system, the Correction Speed should be configured starting from a low value and increasing it until the height is not controlled properly because of hunting in the servo loop. Operate at a lower speed to give a margin of safety. The Maximum Correction values (particularly the Lowest) should be set to minimise the risk of damage to the torch tip by crashing into the work.

At the end of the cut an M5 switches off the arc, small dwell (G4) can be used be used to allow the arc to extinguish. The plasma controller will deactivate the arc good signal and so the THC servo will switch off. At this point the actual height of the torch (whether above material zero or pierce height) will be loaded into the Z axis. It can then be moved by the part program to Safe Z and rapid moved to the start of the next cut when the process can be repeated.

## 6.1.11.3 Hints on G-code and running the system

The following points should help you get the best out of the THC system:

- A pierce is unlikely to be as clean as an established cut. It is therefore best, when you can, to pierce in scrap and use a tangential lead-in cut. If you use a CAD/CAM package to create your part program this should be very easy to program or be automatic. Most torch manufacturers concur that the best approach is to start the pierce higher above the work than the cut height. This is due to the fact that as the plasma arc is piercing the metal prior to breakthrough, the molten metal has no place to go but up! If the tip is too close then fouling the tip and secondary arcing are possible, greatly reducing the life of the tip and electrode. If you control the pierce height from your part program then you can move from the safe height to the initial pierce height, start the arc, wait for penetration then quickly lower the tip into the cut. The THC then will take over the fine adjustment as the torch moves along the cut.
- The sequence of M3 and the Z move to the "cutting" position is reversed to that used in milling. Any CAM post-processor needs to take account of this.
- Best results are usually obtained by running in Constant Velocity rather than Exact Stop modes. The best feedrate will have to be determined experimentally for each plasma system and material being cut. Some manufacturers' websites have tables of recommended cutting speeds for different materials. Use these as a guide to start but adjust the rate to generate the minimum amount of bottom side dross (slag) that sticks to the work. The flame of a plasma is "floppy" in that the bottom of the arc will trail the top as it moves and create issues if the move is too rapid or changes direction too quickly. Thus the recommended cut rates may work on straight cuts but prove unsuitable for smaller curves and corners.
- The direction and sequence of cutting can affect the final quality of work. The plasma swirls in the torch and as it comes out so that it is much like a rotating bit,

This implies an optimum direction for circular cuts. In most cases this is clockwise. It is best to distribute cuts over a sheet to minimise distortion due to localised heating of one part. It is helpful to allow the work to cool after doing an initial series of short cuts before embarking on the more difficult long ones. Experience has shown that cutting from the center out causes smaller stresses and less buckling. Also, obviously, it is necessary to cut holes out of the middle of objects before their outlines are cut.

- Unless the material is very distorted it is usually satisfactory to calibrate the Z position once rather than between cuts. The G28.1 command can be used to reference (home) an individual axis so Z can be referenced at any point but on jobs where there are many small cuts this can take up a lot of time
- When you get your system running you will find that you are able to make a very large number of cuts with a considerable total length during a day's work. In these circumstances using a plasma unit which has a good supply of discounted quality consumables is more important than initial purchase price.

# 6.1.12 Limits and miscellaneous control family

## 6.1.12.1 Input Activation 4

Input activation signal 4 can be configured to give a hard wired Single Step function equivalent to the *Single* button in the Program Running control family.

## 6.1.12.2 Override limits

Mach2 can use software to override limit switches connected to its inputs.

This can be automatic i.e. the jogging performed immediately after a reset will not be subject to limits until the axis is jogged off the limit switches. The *Toggle* button and warning LED for *Auto Limit Override* controls this.

As an alternative limits may be locked out using the OverRide Limits toggle. Its use is indicated by the LED.

**Notice** that these controls do not apply if limit switches are wired to the drive electronics or to activate EStop. In this case an external electrical override switch will be needed to disable the switch circuit while you jog off them.

## 6.1.12.3 Soft limits enable

The *Soft Limits* button is a toggle which will enable the soft limits values defined in Configure>Soft Limits. When enabled the soft limits will not allow the controlled point to

move outside the declared limits. An attempt to jog outside them will be indicated as an error. A programmed move will clip at the limits.

# 6.1.13 Reference switch location control family

As described elsewhere it is not necessary for reference switches to be at the end of an axis. The performance and usability of a machine may be enhanced if they are positioned elsewhere. The absolute coordinates of each reference switch are defined in this family of controls.



Figure 6.15 - Reference switch locations

Sofware Limits Figure 6.14 - Limits control family

Single Step on input Activation4

Auto Limit OverRide

OverRide Limits

Toggle

Toggle

# 6.1.14 System Settings control family

*Note:* The controls in this family are not in one place on the screens released with Mach2. You will need to hunt for them on Program Run, Corrections and Diagnostics screens.

## 6.1.14.1 Units

This toggle implements the G20 and G21 codes to change the current measurement units. You are strongly advised not to do this except in small fragments of part program on account of the fact that Fixtures and Tool tables are in one fixed set of units.





# 6.1.14.2 Safe Z

This family allows you to define the Z value which is clear of clamps and parts of the workpiece. It will be used for moves to zero and changing the tool.

## 6.1.14.3 CV Mode/Angular Limit

This LED is lit when the system is running in "Constant Velocity" mode. This will give smoother and faster operation than "Exact stop" mode but may cause some rounding at sharp corners depending on the speed of the axis drives. Even when the system is in CV mode a corner with a change of direction more acute than the value given in the *Angular Limit* DRO will be performed as if Exact Stop was selected. Full details of this are given under *Constant Velocity* in chapter 11.

# 6.1.14.4 Offline

This toggle and warning LED "disconnects" all the output signals of Mach2. This is intended for machine setup and testing. Its use during a part program will cause you all sorts of positioning

problems.

## 6.1.15 Encoder control family

This family displays the values from the axis encoders and allows them to be transferred to and from the main axis DROs





The Zero button will reset the corresponding encoder DRO to zero.

The *To DRO* button copies the value into the main axis DRO (i.e. applies this values as a G92 offset).

The Load DRO button loads the encoder DRO from the corresponding main axis DRO.

## 6.1.16 Automatic Z control family

Mach2 has the facility to set a lower limit for moves in the Z axis. See Configure>Logic dialog for the static setting of this Inhibit-Z value.

There is also a control family which allows this Inhibit Z value to be set



Figure 6.18 – Automatic Z control

while preparing and before running a G-code program. This is shown in figure 6.18.

Code the program, which might often be a DXF or HPGL import, so that it makes a single cut or set of cuts at the finally desired Z depth (perhaps Z = -0.6 inch assuming top of workpiece is Z = 0). The last command should be an M30 (Rewind)

Using the Automatic Z Control controls (a) set the Z-inhibit value to the Z for depth for the first roughing cut (perhaps Z= -0.05) (b) the Lower Z-Inhibit to the successive cut depths (we might allow 0.1 as the tool has some side support). The whole job will need seven passes to get to Z = -0.6, so (c) enter 7 in Repetitions and toggle On/Off to ON. On Cycle Start the machine will automatically make the series of cuts at increasing Z depth. The DROs track the progress decrementing Repetions as they are performed and updating the Z-inhibit value. If the given number of Repetitions does not reach the part program's requested Z depth then you can update the Repetitions DRO and restart the program.

# 6.1.17 Laser Trigger output family

Mach2 will output a pulse on the Digitise Trigger Out Pin (if defined) when the X or Y axes pass through trigger points.

The Laser Trigger group of controls allows you to define the grid points in the current units and relative to an arbitrary datum.

Click *Laser Grid Zero* when the controlled point is at the desired grid origin. Define the positions of the grid lines in X and Y axes and click *Toggle* to enable the output of pulses whenever an axis crosses a grid line.

This feature is experimental and subject to change in later releases.



# 6.1.18 Custom controls families

Mach2 allows a machine builder, which could be you or your supplier, to add a whole range of features by custom screens which can have DROs, LEDs and buttons which are used by VB Script programs (either attached to the buttons or run from macro files). Examples of such facilities are given in chapter 8. These example also show how different Mach2 screens can look to suit different applications even though they perform essentially the same function required by a milling machine or router.

# 6.2 Loading a G-code part program

If you have an existing part program then you load it into Mach2 using the File>Load G-code menu. You choose the file from a standard Windows file open dialog. Alternatively you can choose from a list of recently used files which is displayed by the *Recent* screen button.

When the file is chosen, Mach2 will load and analyse the code. This will generate a toolpath for it, which will be displayed, and will establish the program extrema. Errors in lines of code are generally ignored. You should not rely on being given a detailed syntax check.

The loaded program code will be displayed in the G-code list window. You can scroll through this moving the highlighted current line using the scroll bar.

# 6.3 Editing a part program

Provided you have defined a program to be used as the G-code editor (in Configure>Logic), you can edit the code by clicking the *Edit* button. Your nominated editor will open in a new window with the code loaded into it.

When you have finished editing you should save the file and exit the editor. This is probably most easily done by using the close box and replying Yes to the "Do you want to save the changes?" dialog.

While editing, Mach2 is suspended. If you click it its window it will appear to be locked up. You can easily recover by returning to the editor and closing it.

After editing the revised code will again be analysed and used to regenerate the toolpath and extrema. You can regenerate the toolpath at any time using the *Regen* button.

# 6.4 Manual preparation and running a part program

## 6.4.1 Inputting a hand-written program

If you want to write a program "from scratch" then you can either do so running the editor outside Mach2 and saving the file or you can use the *Edit* button with no part program loaded. In this case you will have to Save As the completed file and exit the editor.

In both cases you will have to use File>Load G-code to load your new program into Mach2.

# 6.4.2 Before you run a part program

It is good practice for a part program to make no assumptions about the state of the machine when it starts. It should therefore include G17/G18/G19, G20/G21, G40, G49, G61/G62, G90/G91, G93/G94.

You should ensure that the axes are in a known reference position - probably by using the *Ref All* button.

You need to decide whether the program starts with an S word or if you need to set the spindle speed by hand or by entering a value in the *Set Spindle Speed* DRO.

You will need to ensure that a suitable feedrate is set before any G01/G02/G03 commands are executed. This may be done by an F word or entering data into the *Feedrate* DRO.

Next you may need to select a tool and/or fixture and to apply their offsets.

Finally, unless the program has been proved to be valid you should attempt a dry run, cutting "air" to see that nothing terrible happens.

# 6.4.3 Running your program

You should monitor the first run of any program with great care. You may find that you need to override the feed rate or, perhaps, spindle speed to minimise chattering or to optimise production. When you want to make changes you should either do this on the "fly" or use the *Pause* button, make your changes and the click *Cycle Start*.

# 6.5 Importing other files

Mach2 will convert files in DXF, HPGL or JPEG format into G-code which will cut a representation of them.

This is done using the File>Import Data menu. Having chosen a file type you have to load the original file. You are prompted for parameters to define the conversion and feed and coolant commands to be included in the part program. You the import the data. Mach2 has to create a .TAP working file which contains the generated G-code, so you will be prompted by a file save dialog for a name and folder for this.

The .TAP file is then loaded into Mach2 and you can run it as with any other part program.

Full details of the conversion processes and their parameters are given in chapter 9.

This page is intentionally not used

This chapter explains how Mach2 works out where exactly you mean when you ask the tool to move to a given position. It describes the idea of a coordinate system, defines the Machine Coordinate 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 coordinate 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.





This is easiest to explain looking at a machine. We have chosen an imaginary machine that make it easier to visualise how the coordinate 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 Coordinate 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 position is often defined by the position of Home switches which the machine moves to when it is switched on. At any event there will be a zero position for each axis called the **absolute machine zero**. 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 coordinates of the

**CNC Machining with Mach2Mill** 



Controlled Point relative to some coordinate system. The reason you are having to read this chapter is that it is not always convenient to have the zeros of the measuring coordinate 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 machine zero position.

# 7.2 Fixture offsets

Mach2, like all machine controllers, allows you to move the origin of the coordinate 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 coordinate system.

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



Figure 7.3 - Coordinate system origin offset to corner of paper

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)

# 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 coordinate of the Touched axis is put into the Offset DRO and the axis DRO reads zero. Subsequent Touches on other axes copy the Current Coordinate to the offset and zero that axis DRO.

If you wonder what has happened then 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 coordinates of the controlled point) to give the absolute machine coordinates of the controlled point. Mach2 will display the absolute coordinates 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 coordinates 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 coordinates 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 Coordinate 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 home switches installed, this will move each linear axes (or chosen axis) until its switch is operated then move slightly off it. The absolute machine coordinate system origin (i.e. machine zero) is then set to given X, Y, Z etc. values - frequently 0.0. You can define a non-zero value for the home switches if you want but ignore this for now!

The Z home 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 coordinate Z=0.0 then all the working positions are lower and will be negative Z values in machine coordinates.

Again if this is not totally clear at present do not worry. Having the Controlled Point (tool) out of the way when homed is obviously practically convenient and it is easy to use the Fixture offset to set a convenient coordinate 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 Coordinate 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).

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



Figure 7.4 - Now we want another color



Figure 7.5 - Disaster at 0,0,0!

Setting the Tool Z (and for turning 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 well known to lathe users, 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.

- 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 0.0 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 Coordinate 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 (T word, 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 of 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:

• In standard 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. Tools number from 0.

- 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 Coordinate 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 coordinates 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 coordinate 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 Coordinate 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



Figure 7.9 - A double fixture

allow you to look at all the entries in your tables and to Save them on disc. These facilities can also be accessed from the Configure>Tooltable and Configure>Fixtures menus.

# 7.7 Practicalities of "Touching"

## 7.7.1 End mills

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.

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* 

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

*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.

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 coordinates of the Controlled Point to be. This does **not** move the tool it just adds another set



Figure 7.11 - Edge-finder in use on a mill

of offsets to the origin of the Current Coordinate 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 coordinate 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 coordinates. If the *G92* LED is lit then axis DROs are showing the Controlled Point in the Current Coordinate system with all offsets (Tool, Fixture and G92 applied). This is the most useful display as all moves are in this coordinate 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 coordinate 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 Coordinate System values (G92 LED lit) and can check things relative to the absolute machine coordinates using the *Mach/Machine* button.

# 7.9 Tool diameter and tool tip radius

These are two additional values in the Tool Table entries. 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 applies to the Mach2Turn program. 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.

This page intentionally not used.

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: special keyboards, macro writing, Screen Designer, 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 Communication "routes"

Before we can look at customisation in detail you will have to know how the various parts of your machine communicate with Mach2 and, indeed, how you communicate with it.

Figure 8.1 shows the important communication routes and with the following description should allow you to understand how they work. The diagram looks complex and this is a fair reflection of the fact that Mach2 is complex and can be configured to work with many types of machine. Your system will, almost certainly, not have all the features on it but, before you start customising, now is a good time to understand the options. Notice the key which explains what the different type of lines represent.

# 8.1.1 Electrical connections

If you have set up your own system you will have been through the process of connecting switches like the home switches to pins of the parallel port(s) and using the dialogs to tell Mach2 which pins are used for which function. This is shown at the bottom left of figure 8.1. Notice we have shown the limit and EStop switches directly controlling the drive electronics because this is the safest technique and recommended in this manual. Your switches might, of course, control Mach2 via the parallel port(s). This would change the arrows from the switches on the diagram to go into the port(s) instead of the motor drives box.

Mach2 makes extensive use of these signals during running. This is shown by the arrow into the "brain".

Some switches (currently 3) can be assigned to input pins as OEM Trigger signals. These can be configured to do what you wish by issuing an "OEM code" command directly to Mach2 - just like pressing a screen button does. They do not, however, need anything on the screen to do this. The OEM Button codes that can be issued by each are defined to Mach2 using, rather unexpectedly as they are nothing to do with Hotkeys, the Configure>Set Axis Hotkeys dialog.

# 8.1.2 Keystroke connections

## 8.1.2.1 Keystrokes

A "keystroke" produces a scancode that is sent to your computer using the PS/2 or USB connectors. They can be generated in three different ways:

- By typing on the computer keyboard
- By closing/opening a switch connected to a keyboard emulator. Such devices may be connected via the Universal Serial Bus (USB) and/or be daisy-chained to the main keyboard. They were originally developed by the gaming community to allow computer games to put in and be controlled by arcade-like cabinets but are very convenient for machine control applications. It is also possible to connect an encoder (typically low resolution like those used as "digital potentiometers" in hi-

fi and video equipment) and to generate a given scancode for each "click" in the clockwise direction and another scancode for each counterclockwise click.

 By pressing a button or key on a custom keyboard (perhaps like a numeric pad on a long lead, buttons on a joystick, or a "console"). These options are really just some switches and an emulator in a standard box, probably with some neat way of labelling the switches. Windows refers to them as Human Interface Devices (HIDs) Figure 8.2 shows an example with a Mach2 function overlay sheet in place.



## Figure 8.1 – Flow of control in a Mach2 system

A keystroke when Mach2 is running can, of course, do many things: be part of a command entered in the MDI line, be part of a value entered into a DRO or be a *hotkey* shortcut. A

real keyboard will be used for all these things but the other sources of keystrokes will generally only be used for shortcuts.

## 8.1.2.2 Keystrokes and Shortcuts (Hotkeys)

A keystroke generates a Scancode which is processed by Windows and then Mach2. If it is not used directly as input to MDI or a DRO then it can perform **three** functions:

- if the code defined as a Button Shortcut (Hotkey) for a button currently displayed on screen then it is equivalent to clicking the mouse on that button
- if the code defined in the Configure>Set Axis Hotkeys it will jog, load a file or select MDI or a DRO as appropriate

if the code defined in the

٠



Figure 8.2 – A customisable keyboard

Configure>Ports and Pins Input Pins dialog it will turn On (key down) or Off (key up) the signal to which it is attached. This, used with a keyboard emulator, gives a mechanism for extending the number of input pins far beyond those available on the two parallel ports. Not all signals can be made "virtual" like this and you are advised not to use the mechanism for safety critical operations.

You may consider using having real push buttons (or perhaps a membrane switch pad) beside the screen which can be labelled for different functions on each screen (making soft keys like in a bank ATM) or be used for a common function like screen changing. This is illustrated top right of figure 8.1.

A rotary encoder (MPG) connected by keyboard emulator can be used to send jog axis hotkeys to jog axes or to send the codes for hotkeys for buttons like Raise/Lower spindle speed or override feedrate.

This keystroke processing mechanism is shown by the blue lines on figure 8.1 from the "keyboard devices" to the screen buttons and the Input Pins dialog.

# 8.1.3 The KeyGrabber and profilers

As mentioned above, keystokes from the actual keyboard and keyboard emulators have their scancodes processed by Windows before they are given to Mach2. In particular, of course, Windows has to route the keycodes to the application whose window has the focus.

Custom keyboards, game pads etc. (HIDs) generally need a piece of software to define which keycodes each button is to generate. This software is often called a *profiler* and is supplied by the custom keyboard manufacturer. Profilers usually have a picture of the device and its buttons and when you click on these you are prompted to input the keycode and any <Shift>, <Ctrl> or <Alt> modifies to be used with it.

A utility, KeyGrabber, has been developed, by Les Newell, for use in conjunction with Mach2 (and its Screen Designer) to make it easy to configure Keyboard Emulators and USB connected HIDs including custom buttons, joy sticks, throttle controls etc. It has the additional advantage that it will pass selected keycodes to Mach2 whatever program has the focus. This Mach2 can continue to be controlled even if you need to briefly run another application like Windows Explorer, the Calculator, indeed, you accidentally swap to a program by clicking in the task bar. Although at first sight translation of a few key codes sounds rather trivial if you look into it we think you will discover how powerful a technique it is for making Mach2 wasy to use with a variety of standard and therefore reasonably priced input devices.

Details of KeyGrabber and its installation and configuration are given in Appendix 6.

# 8.1.4 VB Script connections

A part program is written using G and M codes. It is, however, also possible to program Mach2 in another language called VB Script. This is a subset of the Microsoft Visual Basic language which is available in many applications besides Mach2, including Microsoft Office. While it would be possible to totally control a machine tool using Mach2 with a VB Script program, the facility is really intended to help customise the environment in which a G-code part program runs.

## 8.1.4.1 VB Script program

A VB Script program can be used in two ways:

- By being "attached" to a screen button and so be run when the button is clicked or its hotkey is pressed
- By being put in a macro and called from within a part program, the MDI line of the G-code attached to a button. The last of these methods being suitable if the program is too big to attach directly to the button or is required to be used as a macro from a part program as well as a button.

The VB Script program can:

- declare and use VB variables,
- execute conditional statements (If-Then-Else),
- perform loops, and
- call VB the functions and subroutines defined by the macro writer or provided by Mach2 as the interface between it and the script.

The predefined interface functions and subroutines are defined in detail in a section below. The main ones allow the script to issue G-code commands, to read and write the values of DROs, to inspect LEDs (and for User LEDs, to set them), and to send Function and OEM codes to Mach2 thus simulating what a user does when s/he clicks a screen button. Additionally a macro can test the values of input signals, set output signals, access internal data such as motor tuning parameters and use parallel and serial ports not used by Mach2 itself.

# 8.1.4.2 Mach2 macro

As already mentioned, a macro is a piece of VB Script. Each macro has a name like M134. The M is used at the start of every macro name and the number can be any integral value up to 99999 that is not used to define a built-in M-code. These built-in numbers are listed in chapter 11. Thus for example, M12, M50, M16543 are all valid macro names; while M3, M-56, M0234, M567.4 are not valid names.

Standard macros will use the number range up to M999, Original Equipment Manufacturers (OEMs) are advised to use M1000 to M89999 and end-users can avoid naming conflicts by using M90000 to M99999.

Each macro is stored as text in its own file whose name is the macro name with file extension ".m1s". Thus one would find files: M12.m1s, M50.m1s and M16543.m1s on a system with the above macros in it. The macro files are collected in a folder called "Macros" within the Mach2 folder.

A macro is called just like an ordinary M-code command by:

- Execution of a line (block) of a part program containing its name
- Including its name on commands entered on the MDI line
- Including its name in the code to be executed on clicking a screen button.

All these features are shown by the arrows into and out of the Macro box in figure 8.1.

## 8.1.5 Windows' control

The final elements shown in figure 8.1 are the standard devices that control Windows. Common examples are the mouse, and trackball. Windows also supports a gaming joystick and throttle. This can be used for jogging and overriding jog and feed rates.

# 8.2 Coding VB Script programs

This section aims to help you write button or macro code of your own which can 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 to call in part programs if you want them to be portable to other CNC controllers.

If you are considering writing VB Script then you will need to have some experience of simple computer programming (e.g. in Visual Basic, C or even BASIC).

All the usual VB Script data and control structures can be used in the program. For details consult *Windows Script 5.6 Documentation* available for download by following some obvious links from:

http://msdn.microsoft.com/scripting/

The Microsoft presentation is rather oriented to scripts of web pages but the documentation that you can download summarises the data types, control structures, operators etc. very clearly. You will not need to use advanced features like Classes. Regrettably this information is not presented in a form which can conveniently be printed out so you might consider purchase of one on the many books on VB Script.

You will find that there are many operators for manipulating strings, doing trigonometry etc. VB Script is a complete and very capable language. The only rstrictions within Mach2 scripts are on the use of input/output functions and subroutines.

# 8.2.1 A simple button script

Many machine tools allow you to control the feedrate, spindle speed in a geometric progression (i.e. each step is so-much times the previous one).

O JoyStick Togg	gle 🔿 SoftLimits Toggle	© Radius Tracking Toggle	🔿 Jog Toggle
O OemCode 🛛		ScanCode 0	Set Hot Key
C Execute G-Code Execute VB Script	Rem Geometric 20% increa Dim val val = GetDro(18) ' Feedrate if val = 0 then val = 1 else val = val * 1.2 end if	se of Feedrate	
Button Text	Geometric Up		



Figure 8.3 shows the dialog box setting up a button that will increment the feedrate in this way. This operation is performed using the Screen Designer program. Full details of this are given below. For the present we concentrate on the VB Script code. Most of it is shown in the edit box of the dialog. The complete code is as follows:

```
Rem Geometric 20% increase of Feedrate
Dim val
val = GetDro(18) ' Feedrate
If val = 0 then
val = 1
Else
val = val * 1.2
End If
Call SetDro(18,val)
```

The first line is a Remark, ignored by Mach2 but useful to remind one what the button does. The Dim statement declares a variable to use again not essential. GetDRO (18) reads DRO function code 18 (which is the commanded feedrate) and assigns it to the variable *val*. The quote ' character introduces a comment on the end of a line of code. If *val* is zero then it is set to 1 otherwise it is incremented by 20%. Finally the new *val* is stored in the Feedrate DRO so setting the required rate.

As you can see from the dialog the caption on the button will be "Geometric up". A similar button might be provided with code for Geometric Down.

## 8.2.2 Sample macros

#### 8.2.2.1 A simple macro

The following is code to form a very simple Mach2 macro. Suppose it is stored in a file M99990.m1s (the extension is letter-m, figure-one, letter-s). The call M99991 could be included in the file at the beginning of each part program you want to run. If you set a feedrate by the F-word before running the program then it will use that rate otherwise the macro will ask the operator for a value.

#### 8.2.2.2 More complex macro

The following is more complicated example of a Mach2 macro. Suppose it is stored in the file M99992.m1s It might be useful to edit M99992 into the start of a part program if the post-processor that created the program has not taken care to set all the modal values at the start of the program. It would also be handy macro to call by M99992 Q100 S2000 from manual data input (MDI) to set up a feed and speed and the modes before running a program.

```
Rem This macro sets modals to defined values. Useful at start of
         a part program if you are not certain about the default
.
         state of you controller
Rem The P-word can be used to choose options:
         Px1 gives inch units (Px0 gives mm)
.
         P1x gives constant velocity (P0x gives exact stop)
.
           e.g. P10 gives CV and mm
Rem The Q word can set a feed rate. The S word can set a
          spindle speed
Rem The code is written to illustrate some features of VB Scripting
         so includes declarations that are not strictly needed etc.
         The logic can be done in other ways. You might like to
         try some!
Option Explicit ' declarations are mandatory
Dim TxtChoice, PValue, QValue, SValue
PValue = Param1()
 Select Case PValue
    Case -1
      TxtChoice = "G21G61"
    Case 1
      TxtChoice = "G20G61"
```

```
Case 10
   TxtChoice = "G21G64"
  Case 11
   TxtChoice = "G20G64"
  Case Else
   Code "(MSG, P word is not valid - defaults are set)"
End Select
  Call Code (TxtChoice & "G17G40G49G50G90G94") ' set the modes
  QValue = Param2() : SValue = Param3() ' get values
  If OValue >0 Then
                              ' a value has been given
      SetFeedRate (QValue) ' using a Mach2 function
  End If
  If SValue > 0 Then
        code "S" & SVal ' using G-code where there is no function
  End If
```

#### **Commentary on program**

As mentioned before the comments, lines starting with Rem or the text following an apostrophe ' are not interpreted by the system.

Unlike the C and C++ languages, VB Script is not case sensitive so NEWRATE, NeWRate, newrate are all the same variable. You can, however, use case to make your programs easier to read.

Variables can be used in the VB Script code to store numbers or character strings. They do not have to be declared but using a Dim declaration helps other people to follow your program.

The program next accesses the value of the P word in the macro call using the Mach2 standard function Param1(). The brackets show that a function is being called. The value, in this case a number, is assigned to the variable PValue.

Next the Select Case control structure is used to assign a different string to TxtChoice depending on the value of the P word. Other control structures available include If Then Else End If and various forms of loop.

The VB Script program can interact with the operator by asking for a value by using the Mach2 function Question. In the example above no value is used. If a value is required then the form x =Question ("What is the new X value") is used.

Next the program uses the & operator to concatenate (join) the TxtChoice string which is already set up with the standard initial G values and uses the Mach2 function Code to send the string to Mach2 to be interpreted.

Finally the R and S words are accessed by Param2() and Param3() and, if used, are sent to Mach2 by the function SetFeedRate and another use of Code.

## 8.2.3 A common confusion with VB Script and a hint

You may have noticed two different ways of calling the Code subroutine in the second example.

```
Call Code (TxtChoice & "G17G40G49G50G90G94") and Code "S" & SVal
```

Could have been written

Code TxtChoice & "G17G40G49G50G90G94" and Call Code ("S" & SVal)

Either is correct and both do the same thing. If you like to group the arguments with brackets then you need to use the keyword Call as well.

A function can be called and have its result thrown away as in this example:

```
Question ("Did you know P word was not valid - defaults are set")
```

Using brackets with no Call show that it is a Function which is expected.

Sometimes you can get away with breaking these rules but you will find that you get very hard to find bugs!

If you include the statement Option Explicit at the start of your code the VB Script will require you to declare all variables (e.g. with Dim statements). This is very helpful in trapping bugs caused by misspelled variable names.

## 8.2.4 The Mach2 VB Script functions and subroutines

This section describes the most important standard routines (i.e. functions and subroutines) that van be called from VB Script on buttons or in macros.

#### 8.2.4.1 To execute G or M-codes from a script

The simplest and most powerful routine is:

Subroutine Code (text as String)

The text argument is a string expression (including, or course a constant or simple variable) which is any line of G or M codes that you could enter into MDI. It will be passed to Mach2 for execution. The only restriction is that you are advised not to call another script from within a script.

Examples, using both versions of the syntax for subroutine calls:

```
Code "G0X0" ' X to zero in current coords
Code "G1X10" & Feed ' variable Feed has been set to something like
' "F150"
```

#### 8.2.4.2 For accessing the screen controls

As you have seen in the earlier examples, a macro read and change the data in a DRO. It can also read the state of any LED and simulate the action of clicking a screen button. To access these operations on Mach2 controls you use the codes used internally by Mach2 and its Screen Designer for the DRO, LED or button operation you want to use.

There are, for historical reasons, two different code lists for each type of control. The original built in controls are described by their Function Code. Later features have controls described by so called "OEM" Codes. You will have seen the reason for this if you have already tried Screen Designer. If you have not used Screen Designer have a look at the figure showing the Properties of the Standard Cycle Start button in this chapter. The buttons that are defined by checking one of the Radio Buttons in the dialog are referred to by Function Codes. Buttons defined by checking OEM Code and entering a Code value in the box are referred to by OEM Codes.

DROs and LEDs can be defined that have no meaning to Mach2 being solely for you use. There are 255 of each denoted by "OEM" codes 1000 to 1254. You must refer to them using special functions with "User" in the name to make it obvious that they are not controlling Mach2 itself.

The Function code numbers start at zero in the top-left and count up across the screen. Thus in the Set Button screen, Run (i.e. the Cycle Start button) is Function = 0 and Zero Y is Function = 9. Rather than looking up codes using Screen Designer, you may find it more convenient to refer to the complete table of codes (ControlDefs.csv) in the Layouts folder of the Mach Developers' Network (see title page for link to MachDN).

Although we use literal values (like 14) in the examples you are strongly advised to assign the values you want to use to variables at the beginning of your macro and then use the variables in calls to the routines. This will make your program much easier to read. Thus the first LED example in a complete script would be:

```
JoyStickLEDFn = 14
:
:
bJoy = GetLed (JoyStickLEDFn)
```

#### LEDs

```
Function GetLED (ledFun as Integer) as Boolean
Function GetOEMLED (ledOEMCode as Integer) as Boolean
Function GetUserLED (ledUserCode as Integer) as Boolean
```

Choose the appropriate routine depending on whether you want to access a built-in, OEM or User LED. ledUserCode must be in the range 1000 to 1244. The result is True (i.e. non-zero if converted to an integer) if the LED referred to is alight.

User LEDs, only, can be set on or off by:

```
Sub SetUserLED (ledUserCode as Integer, cond as Integer)
```

If cond = 1 the LED will be on. If cond = 0 then it will be Off

#### Examples:

bJoy = GetLed (14)	'	set variable bJoy if Joystick is enabled
If GetOEMLed (29) Then	'	see if a Fixture is in use
SetUserLED (1002, 1)	'	turn on user LED

#### DROs

Function GetDRO (droFun as Integer) as Double
Function GetOEMDRO (droOEMCode as Integer) as Double
Function GetUserDRO (droUserCode as Integer) as Double

Choose the appropriate routine depending on whether you want to access a built-in, OEM or user DRO. droUserCode will be in the range 1000 to 1254. The result is the current value displayed by the DRO.

```
Sub SetDRO (droFun as Integer, newValue as Double)
Sub SetOEMDRO (droOEMCode as Integer, newValue as Double)
Sub SetUserDRO (droUserCode as Integer, newValue as Double)
```

Choose the appropriate routine depending on whether you want to access a built-in, OEM or user DRO. droUserCode will be in the range 1000 to 1254. The routine sets the expression provided for newValue into the DRO. Not all DROs can be written. If you cannot type a value into the DRO on the screen (e.g. X Velocity = function 6) then you cannot set it in a script.

#### *Example:*

```
Call SetDRO (18, GetDRO (18) * 1.1) ' increase feedrate by 10%
```

#### **Button Commands**

```
Sub DoButton (buttFun as Integer)
Sub DoOEMButton (buttOEMCode as Integer)
```

Choose the appropriate routine depending on whether you want to use a built-in or OEM command. Mach2 is instructed by the script to perform the function specified.

There is no provision for the trapping or reporting of errors but as most functions have an LED associated with them this can be inspected by the script code to check that the required action has been performed..

Very many "buttons" are toggles or cycle through a range of possible states or values. A loop containing inspection of an associated LED can be used to set a particular state. This example would be particularly suitable to be attached to a button.

#### Example:

Rem This sets the MPG jog to be enabled and the wheel to jog the Y axis  $% \left[ {\left[ {{{\rm{T}}_{\rm{T}}} \right]_{\rm{T}}} \right]$ 

```
JogTogButton = 174
JogMPGEn = 175
MPGJogOnLED = 57
MGPJogsY = 60
OK = False
For I = 1 to 2
  If Not GetOEMLED (MPGJogOnLED) Then
         Call DoOEMButton (JogMPGEn) ' try to enable
  Else
         OK = True
                     ' MPG is enabled
         Exit For
  End If
Next T
Rem Could test of OK true here
OK = False
For I = 1 to 6 \cdot must get there after six axis tries
  If Not GetOEMLED (MPGJogsY) Then
         Call DoOEMButton (JogTogButton) ' try next one
  Else
         OK = True ' got right axis selected
         Exit For
  End If
Next I
Rem Could test OK here as well
```

#### 8.2.4.3 Interrogating Mach2 internal variable

The current value of Mach2 internal variables can be read using the GetParam function.

Function GetParam (name as String) as Double

This returns a numeric value corresponding to the name of the given variable which is provided as a string (constant or variable)

The corresponding routine SetParam sets the value of the variable to newVal.

Sub SetParam (name as String, newVal as Double)

Examples:

```
Rem interrogate drive arrangements
mechProp1 = GetParam ("StepsPerAxisX")
Rem make C acceleration be same as X for slaving
Call SetParam("AccelerationC", GetParam ("AccelerationX"))
```

Notice that the word "Param" is used here in a different sense to the Machine Parameters accessed by the # operator from within a part program and in accessing the Q, R & S word "parameters" to a macro call.

#### 8.2.4.4 Access to the machine G-code parameter block

Mach2 has a block of variables which can be used in part programs. They are identified by # followed by a number (the parameter address). The contents of the Tool and Fixture tables are in these parameters but there are many values that can be used by the writer of a part program.

These machine variables can be accessed within macros by GetVar and SetVar.

```
Function GetVar (PVarNumber as Integer) as Double
```

Sub SetVar (PVarNumber as Integer, newVal as Double)

The predefined parameter variables are defined in chapter 11.

Examples:

FixNumb = GetVar (5220) ' get current fixture number

Rem set X offset of fixture 2 to be same as fixture 1 Call SetVar (5241, GetVar (5221)) Rem increment a counter, say in a multiple part layout Call SetVar (200, GetVar (200) + 1))

#### 8.2.4.5 Arguments of macro call

When a macro is called from the MDI line or within a part program then data can be passed to it by P, Q, and S words on the line. The values of these words are "read" in the macro using the Param functions.

Function Param1 () as Double ' gets P word
Function Param2 () as Double ' gets Q word
Function Param3 () as Double ' gets S word

#### 8.2.4.6 Information from and to the user

Scripts can communicate with the operator by displaying a dialog box with a prompt into which the user can type numeric data. The Question function prompts for one item. The GetCoord routine prompts for the values of X, Y, Z and A coordinates.

The other strategy, probably more suited to scripts attached to buttons, is to provide DROs of a screen into which data is set before running the macro. These can of course also display results from the script.

User intelligent labels and Tickers enable messages to be displyed.

#### Question

Function Question (prompt as String) as Double

The string in prompt is displayed in a modal dialog titled "Answer this. The dialog contains an edit box. The value of the function is set to the number in this when OK is clicked.

#### GetCoord

Sub GetCoord (prompt as String)

As with Question, a modal dialog titled "Enter Coordinates" displays prompt. This has four edit boxes labelled X, Y, Z and A into which values can be typed. GetCoord itself does not return the values to the macro code. These must be fetched by GetXCoor, GetYCoor etc.

```
Function GetXCoor () as Double
Function GetYCoor () as Double
Function GetZCoor () as Double
Function GetACoor () as Double
```

#### **User defined DROs**

This technique is mainly applicable to scripts which are run from a user defined screen button.

A block of DRO OEM codes is allocated to 255 DROs which are not used by Mach2 itself. These DROs, suitably labelled, can be placed on a screen.

The operator enters data into the DRO(s) before pressing a button or series of buttons to run the macro or macros. The macro(s) access the data using GetUserDRO as explained above. The macro can also use SetUserDRO to update the data or return a result in another DRO.

In addition there are 255 user LEDs which can be read and (unlike normal LEDs) written using GetUserLED and SetUserLED.

This technique can, for example, be used to implement a totally personal scheme to extend the Mach2 offset setting by Touch with Correction. Suppose you have a probe with a 5 mm tip diameter which only trips in sideways movement (i.e. for X and Y) then you might use a 1 mm slip or piece of shim-stock to manually feel the Z touch. You could define a pair of macros attached to two buttons to apply the fixed, 5 mm, X and Y correction and a third button that uses a Z-correction DRO to set the thickness of the shim or slip which is in use.

Such features can be made to appear to the operator to be exactly like built-in Mach2 functionality.

## **User Labels and Tickers**

If the current screen has a label whose, case sensitive, text is in the range *UserLabel1* to *UserLabel255* then the actual text displayed can be set in a Script by calling

Sub SetUserLabel (number as Int, text as String)

This will display the given text in the label corresponding to the number given.

e.g. SetUserLabel 12, "You must enter a whole number of holes"

Sub SetTicker (number as Int, text as String)

Accesses the 255 tickers *Ticker1* to *Ticker255*. In a ticker the text of the message scrolls through the box so a very long message can be given in a small area of screen at the expense of some inconvenience for the user.

e.g. rather frivilously: SetTicker 205, "This is a very long error message because you seem to have done something very silly"

#### 8.2.4.7 Input/Output signals, a serial port and "foreign" ports

Scripts can access the input signals (both on parallel ports and defined virtually in response to keycodes) such as the state of home and limit switches and can control output signals.

```
Function IsActive (sigNo as Integer) as Boolean
Sub ActivateSignal (sigNo as Integer)
Sub DeActivateSignal (sigNo as Integer)
```

IsActive tests input signals. It will return True if the signal is **active** (i.e. its LED would be lit on the Diagnostics screen). In other words this test is after the application of the Active Hi/Active Lo configuration of the signal hot a test of "0 volts" or "5 volts" on the signal's pin.

ActivateSignal and DeActivateSignal similarly control the logical state of output pins. Mach2 will apply the Active Hi/Active Lo configuration to establish the electrical state required.

The required signal is coded using the values defined in the table SignalDefs.csv in the ForMacroWriters folder of Mach Developers' Network. It is best to import this data into Microsoft Excel or Access to make it easy to read.

## 8.2.4.8 Serial port

You can send bytes of raw data to a serial port. The port number (i.e. *n* in COM*n*) to be used and the baud rate for transmission is set in Configure>Logic. RTS/CTS hardware flow control protocol will be used to control large volumes of data but this will not be normally required. Data is transmitted 8 data bits, 1 stop bit No Parity by a call of SendSerial.

Sub SendSerial (chars as String)

*Example:* to write the value of X DRO to an LCD display connected to the serial (RS232) port.

Call SendSerial ("X-Axis = " & GetDRO (0))

#### 8.2.4.9 Foreign ports

Scripts can access ports on the PC which are additional to the one (or perhaps two) parallel port(s) defined in Configure>Ports and Pins. These are accessed at the basic hardware port address level and you will have to be aware of the details of the individual port addresses, allocation of data and status bits etc.

Function GetPortByte (pAddr as Integer) as Byte
Sub PutPortByte (pAddr as Integer, bData as Byte)

This feature should be used with great care as, if misused, it can interfere with any peripheral on your system, including the hard-drive.

#### 8.2.4.10 Waiting features

As described above the script code and Mach2 itself run in two separate processes. You can test to see in Mach2 is busy or idle by calling:

Function IsMoving () as Boolean

This will return True if Mach2 is busy. You should call it in a loop after commanding an axis move or other function which could take a significant time and before reading DROs or LEDs that could be affected by the move.

Example:

```
Call Code ("GOX12Z100")
While IsMoving ()
WEnd
x = GetDRO (2) ' get Z value in case it has been Z inhibited
```

#### 8.2.4.11 Legacy routines

Scripts may also call a range of "legacy routines" whose functions can now be performed in a more general and sometimes efficient way. These are listed in the Macros section of the Mach Developers' Network.

## 8.2.5 A more complicated macro example

The next example is used to show the features available in more detail and to point out some of the difficulties in using macros. The macro is designed to perform a number of roughing cuts to reduce the size of a piece of bar. Figure 8.4 shows the operation partially completed. The macro asks the operator for the X values of the ends of the piece to be cut, the initial Z position of the top and the final Z coordinate of the top when the roughing has been done.



Figure 8.4 – Roughing a bar

For the sake of illustration we suppose it is to be called by M62 and so will be stored in the file M62.m1s

Rem This macro M62 makes a series of roughing cuts to reduce a
' non-standard piece of material to a desired initial size.
Rem The P-word imust be provided to give depth of cut to be used
Rem The tool diameter and area to be machined are input in response to
' prompts from the macro. The units are the currently selected ones.
' The macro leaves the machine in Absolute distance mode
Rem The code is written to illustrate some features of VB Scripting so
' includes declarations that are not strictly needed etc. The logic

```
can be done in other ways. You might like to try some!
Option Explicit
Dim TxtChoice, CutDepth, StartX, EndX, StartZ, EndZ
    CutDepth = Param1()
                         ' the P word is not given
   While CutDepth <= 0
        CutDepth = Question ("What depth of cut do you want each pass?")
    Wend
      StartX = Question ("What is X of left end?")
      EndX = Question ("What is X of right end?")
      While EndX <= StartX
         EndX = Question ("X right must be > left. What is it?")
      Wend
      StartZ = Question ("What is Z of top of material?")
      EndZ = Question ("What is desired final Z?")
      While EndZ >= StartZ
         EndZ = Question ("Final Z must be below initial Z. What is it?")
      Wend
      code "GOZ" & StartZ
     CurrZ = StartZ ' top of work
code "GOX" & EndX ' cutter to right
      code "G1Z" & CurrZ 'feed down for first cut
      While CurrZ > EndZ
                                       ' loop for the z planes cut
          CurrZ = CurrZ - CutDepth ' Update Z
          If CurrZ < EndZ Then
                                      ' last cut dead to size
              CurrZ = EndZ
          End If
          code "G1Z" & CurrZ ' feed down
code "G1X" & StartX ' cut pass
          code "G1Z" & CurrZ
          code "GOZ" & (CurrZ + 0.1) ' avoid scratching surface
          code "GOX" & EndX
                                       ' back to right end
      Wend
      code "GOZ" & StartZ
      code "GOX" & StartX
  Rem End of M62
```

In this macro, which will probably be used from MDI the P word is used to define the depth of cut in each pass. So a typical call using inch units might be M62 P0.05

The macro then asks a series of questions to establish the X coordinates of the ends of the bar and the original and desired Z values. These values are validated and re-prompted for if they are invalid.

A While loop then performs cutting moves with ever decreasing Z until the desired size is reached.

# 8.2.6 Script Snags and Hints

#### 8.2.6.1 What Windows/Mach2 does with your macro

In order to understand some of the limitations of macros and to help you debug them, you need to understand what Windows and Mach2 do when you run a script from a button or macro from a part program or MDI.

There are three stages in running a script. The last two proceed in parallel:

**Analysis:** The lines of text in your script or macro file are read by the Windows VB Script engine and converted into a more compact internal symbolic form.

**Macro run-time:** The VB Script program is run. At this stage the user will be asked and will answer questions. Calls to the Mach2 functions will be made. These will return values or state of an input pin (IsActive()) and write requests for G-code to be executed (Code etc.).

The function ismoving() can be called to see if the G-code runtime thread (see below) is executing commands. It returns a non-zero value if commands are being executed or moves are buffered.

G-code run-time: Mach2 which is running as a separate "thread" executes the G-code.

This scheme has several implications for you which are described below.

## 8.2.6.2 Script error reporting

Errors discovered in the analysis stage (e.g. mis-spelling while as whyle) are reported with the offending line displayed. This will make them fairly easy to correct.

Unless you have a Microsoft debugger from Visual Studio installed on your computer, errors at macro run-time do not give any useful diagnostics. Errors can range from the wrong number of arguments passed to a function, division by zero etc. You may be able to work out where they are from the Questions asked, by putting in extra "dummy" Questions or by the use of On Error Resume Next followed by a call to the Mach2 Question function to display the error message stored in the global variable err.description. Note that, as any G-code issued is not running in synchronism with the script, you cannot tell where you are by seeing the axes move or the DROs change.

Errors when the G-code runs can be very difficult to trace as Mach2 tends to ignore requests it does not understand rather than flag them up to you. You will just have to "dry run" the macro code to see exactly what is being requested. It is sometimes helpful to store a line of G-code that you have constructed in a variable so that you can print it in a "debugging" Question to check that it looks right.

#### 8.2.6.3 Stuck in a rut?

As you have some very powerful control structures it is quite easy to get a script stuck in a loop. For example in this code fragment EndX is mistyped EndZ

```
While EndX <= StartX
    EndZ = Question ("X right must be > left. What is it?")
Wend
```

There is nothing that the operator can type that will correct the invalid EndX value for the code loops for ever. You can only get out of this by using Control-Alt-Delete and getting Windows to end the Mach2 program. In this case you must run OCXTest to reset the driver or re-boot Windows.

If the line in grey in this code

While CurrZ > EndZ	' loop for the z planes cut
CurrZ = CurrZ - CutDepth	' Update Z
If CurrZ < EndZ Then	
CurrZ = EndZ	' last cut dead to size
End If	
code "G1Z" & CurrZ	' feed down
code "G1X" & StartX	' cut pass
code "G0Z" & (CurrZ + 0.1)	' avoid scratching surface
code "GOX" & EndX	' back to right end
Wend	

is omitted or if CutDepth could be zero or negative then the While will run for ever. This is worse than the loop above because each time round it requests the execution of some G-code. Eventually the buffer will overflow and may well crash Mach2 in totally unpredictable ways.

#### 8.2.6.4 Reporting errors to users

If you accept data from users by Question() or by user DROs in a conversational programming screen then you need to validate this data with logic in the VB Script. For example, few mills will drill holes with negative depth successfully!

You can write messages to the error line by exploiting the MSG, form of GCode comment. The example in 8.2.2.2 illustrates this in the line:

Code "(MSG, P word is not valid - defaults are set)"

After a user error you may not be able to take a default action and so may wish to exit from your script. The VB Script language does not have a suitable command that can be used in the main program (which is what most scripts will be) or a GOTO so you may end up with very deeply nested *if* commands.

A way of avoiding this is to make the code of your script be a subroutine called, say, MainProg. The actual main program just calls this. Now, because Exit Sub is valid in a subroutine and immediately exits from it you can easily abort the run of your script. A skeleton of the whole cone might look like:

```
Sub MainProg
' get data...from DROs..
'
Rem now check the user data
If UserFooVal > MaxVal Then
        Code "(MSG,You are not allowed values of "
        & "Foo greater than" & MaxVal & ")"
        Exit Sub ' bail out because of error
End If
Rem script continues to use good value.....
'
End Sub ' MainProg
MainProg ' call the actual code when button/macro used
```

# 8.3 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 can be 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 800 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.

The individual screens of a layout can be moved, deleted, imported etc. using the ScreenTweak utility (q.v.) It is primarily concerned with whole screens while Screen Designer deals with individual controls, their functions and positioning.

## 8.3.1 Screen Editor basics

The Screen Editor allows you to create and edit a set of 15 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 an image rather than text giving function
- 4) Bitmap display an area of color or 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 15 and screen P which is used to place controls that will appear on every screen (P is for Persistent). This is used, for example, for the buttons used to change from screen to screen.

# 8.3.2 Try out the Designer

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

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 and toolbar 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 or sometimes off the bottom of your finished screen.

Run Screen Designer and do not open a layout. You will 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 pallet 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 that they actually appear on all screens 1 to 15.

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. You can also move selected objects by "nudging" them with the left/right/up/down cursor keys. Each keypress moves the control by one pixel.

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 fifteen 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, other than a button, 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.

When you have created a test screen, use Save As to save it with a new file name. Run Mach2 and load your screen using the Layouts>Load Layout menu. You should never overwrite the default screens with your designs as these files are likely to be overwritten by the installer each time you upgrade Mach2 to a new version.

# 8.3.3 Making the controls work

It is now time to see how to see how controls "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 (F1)" 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 8.5.

## 8.3.3.1 Key scan codes

Buttons and Bitmap buttons can of course be "pressed" by clicking the mouse. They can also have a "hotkey" assigned to them. Pressing that key on the keyboard when the button is visible is equivalent to clicking it with the mouse.

The "hotkey" is defined by its "Mach2 scancode". Screen Designer allows you to type the key you want to use and it will store and display (in decimal notation) the associated scancode.

If you need to check these codes at any time then you can calculate the value by hand as follows:

- ♦ Lookup the ASCII value of the code for the key (For letters it is the uppercase version of the character for other keys it is the code of the unshifted character i.e. 5 rather than %). Thus, for example, spacebar = 32, A = 65, / = 47 etc. If you do not have a printed table of these codes then entering "ASCII code table" into your favorite search engine should find one.
- If your hotkey is to be a given key together with Shift, Ctrl or Alt depressed (or indeed a combination of these) then add, to the basic code, the following values (in decimal)

Shift	1,024
Ctrl	32,768
Alt	2,048

Thus Shift-A would be 1,089 (65 + 1,024), Alt-/ would be 2,095 (47 + 2,048) and Shift-Ctrl-Alt-spacebar would be 35,872 (32 + 1,024 + 32,768 + 2,048)

If you have worked with binary numbers then you will recognise that these rather odd looking numbers are actually powers of two.

## 8.3.3.2 Defining function by G-code or VB Script

This is the most direct way of making a button work. You can provide some G-code or VB Script to be executed when it is clicked. You need to check the appropriate radio button to say what type of code you have written in the edit box. As an example look at figure 8.3 and the G92X0 button on the MDI screen of 1024.set. You will see that the latter (not unexpectedly!) issues the command "G92X0".

You can define buttons using G-code to do whatever you want that can be done by a single MDI line. The VB Script program, indeed almost always will, have multiple lines and can be up to 64k characters long!
**Beware:** If you use G-code like this, then any scaling which is active on axes at the time you click the button will be applied to the coordinates built into your button. The problem does not, of course, apply to zero values, which is a common case, whatever scaling is applied.

**Hint:** Remember that a button will only be operated by its hotkey if it is on the persistent or currently displayed screen. A useful trick for VB Script or G-code buttons, that you want to work at all times, is to draw then on the persistent screen but off the bottom of the viewable area. You can "design" here by hiding the toolbar and status bar using the View menu or by designing on a higher resolution display than you are going to use with Mach2.

If you are only interested in adding hotkeys, G-code and VB Script then you probably need to skip to the section on "Properties of other controls" and then read no further about Screen Designer. Exit **not saving** any changes to the 1024.set.

### 8.3.3.3 Defining function by name

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 scancode. Alt-R is code 2162. If you click *Set Hot Key* and type any other key (or Shift/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

#### Set Button

This button will call the Function:	
C Run C Pause C Rewind C Stop C Single C Resume C Edit File C Zero All Axis C Zero X C Zero Y C Zero Z C Zero A C Zero B C Zero C C Reset FeedOV C Estimate Job C SetNextExec C Goto Zero's C Show Machine Coord. C Show current pos. C Verify Postion C Reset C Home X C Home Y C Home Z C Home A C Home B C Home C G JoyStick Toggle C SoftLimits Toggle C Radius Tracking Toggle C Jog Toggle	
C DemCode 0 ScanCode 0 Set Hot Key	
C Execute G-Code C Execute VB Script	
Browse       Path relative to C:\Mach2\Bitmaps:         Position/Size       b021.bmp         TopLeft-Y:       279         Width:       60         Height:       20	
Locked for mouse	

Figure 8.5 - Properties of standard Cycle Start button

to Alt-R.

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

Close the dialog by OK

### 8.3.3.4 Defining buttons 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 online in the Mach2 Developers' Network.

If you look at the properties on the *MDI* (*M*) button you will see that its OEM code is 2. Codes 1 to 15 are used to select the fifteen 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.

# 8.3.4 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 8.6)

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.

### 8.3.4.1 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 8.7 shows the DROs just after drawing. They are then selected with the



Figure 8.6 - Screen Designer alignment tools

top one chosen last and Align Left Edge clicked. Figure 8.8 shows the result.

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.

### 8.3.4.2 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 8.9 and 8.10 show this done in two stages with the two "Make same.." icons.

The size of a control, other than a button, will determine the size of any text in it.

#### 8.3.4.3 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 8.11.

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.

1234.5678	UNC Discussion
1234.5678	anna iain ann ann ann ann ann ann
1234.5678	

Figure 8.11 - Equally spaced controls

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

You are strongly advised to save any changes in a layout with a name of your own



Figure 8.9 - Same width DROs

Figure 8.10 - Same size DROs

choosing. If you just change 1024 set then, as this is replaced at each Mach2 upgrade, you will lose your changes.

#### 8.3.5 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.

User defined LEDs and DROs are setup with values 1000 to 1254 in the OEM code field.

You can, for example, set the number of lines to be displayed in the G-code window and that, together with the size of the window, will define the text size.

Notice that DROs have a hotkey associated with them. When this key is pressed when Mach2 is running the appropriate DRO will be selected for data input. This is very useful in systems that are run without a mouse.

Some controls are very simple (e.g. the joystick ball) while others are complex (LEDs are different colors, can flash etc.).

# 8.3.6 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:

### 8.3.6.1 System labels

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)

### 8.3.6.2 User Labels/Tickers

#### UserLabel1, UserLabel10, up to UserLabel255

#### Ticker1 up to Ticker255

These display the text set be VB Script calls to SetTicker and SetUserLabel. The difference is that a Ticker scrolls the text so that a small control can display a long message.

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

# 8.3.7 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.

# 8.3.8 Use of Bitmaps

The location of the file containing a Windows bitmap is given in the setup dialog for bitmap buttons and for background bitmaps. All files must be in the Mach2 Bitmaps folder or in folders below it in the tree. It is convenient to group all the bitmaps for a custom layout in a folder within Mach2\Bitmaps to make them easy to maintain and to distribute to other users.

Screen Designer will always store the path relative to Mach2\Bitmaps in the layout file and it is this path that is displayed in the dialog after you have used Browse to locate a bitmap file. This can be seen in figure 8.5.

### 8.3.8.1 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 8.12

Bitmap buttons when created are empty and transparent. In Screen Designer a black border is drawn but such a button will be invisible on the screen when Mach2 is running. Such a button can be though of as a "hotspot" on the screen with its function being indicated on a background bitmap below it. An example is given in the next section.



Figure 8.12 -Bitmap buttons

# 8.3.8.2 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 8.13 You should complete the placing of the controls before drawing the bitmap as it is more difficult to select the handles to resize the controls when the bitmap is present.

Screen designer provides two options on the Edit menu to assist with bitmaps as backgrounds. Edit>Paste Exact pastes an object in exactly the same place on a screen as the place from which it was Cut or Copied.



Figure 8.13 - Bitmap frame for DROs



Figure 8.14 – Screen with all graphics on bitmap and transparent buttons

If you wish to change the size or position of controls on a bitmap then you can cut it and paste it temporarily on an unused screen. Now move the controls and when finished you can Cut from the temporary screen and Paste Exact to restore the position of the bitmap.

If you want to make big changes to a screen then you might cut and paste all the bitmaps to the temporary screen at one time.

After re-pasting bitmaps back to their proper place they will of course be on top of the layers of objects. Edit>Bitmaps to Back allows you to put all the bitmaps on the current screen behind the other controls.

You can use one bitmap file for many backgrounds or bezels. A basic size of 100 pixels square works well for coloured backgrounds.

Stretching a square bezel 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.

### 8.3.8.3 Identifying controls by the background bitmap

Although the controls in Screen Designer are quite flexible, you can achieve an even more interesting presentation by putting all the graphic detail in a large background bitmap. Figure 8.14 illustrates a prototype of such a screen (courtesy of Ken Bell). In such a design the DROs, labels, LEDs etc. are real Mach2 controls. The buttons are in the bitmap but have empty bitmaps drawn over them to define the "hotspots".

Another advantage of this type of screen design is that it is fully scalable to different screen resolutions using ScreenTweak as it does not have the problem of the fixed size text of conventional buttons.

### 8.3.8.4 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 path relative to 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 8.15 shows this done to the set of DROs shown in figure 8.13

### 8.3.9 Advanced features for setting up controls

Figure 8.5 shows several other features of the dialogs for setting up buttons. Similar features exist on the dialogs for other controls.

**Positioning:** The position of the top corner of the control and its size is given in the dialog. If you click the *Enable Input* button then this data can be altered and the new size and position will be applied to



Figure 8.15 - An alternative frame link in dynamically

the control when OK is used to close the dialog.

**Original size:** The *Button will be bitmap's original size* checkbox on the dialog for bitmap buttons and a similar one for bitmaps allows the size of the control to be adjusted t the number of pixels in your bitmap as it was originally created. This option can be used with the option to specify a new position. If this is not used then the top lefthand corner is not altered.

As you can only change the size and position of a control that is on the screen's visible area, you need to be careful with the values that you enter. In an emergency you can move any controls that lie partly off a 1024 x 768 screen to a visible position using Edit>Retrieve Offscreens. This will also reposition controls intentionally placed just off the page such as labels that start with § used to identify screens to ScreenTweak.

**Locking:** The *Locked for Mouse* checkbox allows you to lock the control so that it cannot be moved or resized by the mouse or be nudged. Double-click will still be available to reset its properties.

You can also lock all the selected items on a given screen using Edit>Mouse-lock all selected. You can unlock all the locked controls on a screen using Edit>Clear Mouse-locks.

### 8.3.10 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.

# 8.4 Conversational programming

The techniques of VB Script and custom screens can be combined to implement conversational programming of parts using Mach2. Figure 8.16 shows one of a series of screens designed by Brian Barker at the University of Maine to simplify machining of standard features without writing explicit G-code part programs.

Further examples of his screens as the code used to run them can be found in the Layouts section of the MachDN website. All users are invited to contribute their ideas to the pages of this site.



Figure 8.16 – Sample conversational Programming screen

# 8.5 Other customisation

### 8.5.1 Global hotkeys

Configure>Set Axis Hotkeys allows you to set up the keys which will jog the axes when the Jog Mode button 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 to

CNC Machining with Mach2Mill

make sure they are unique and do not clash with other hotkeys you might want to use. See appendix 2 or use Screen Tweak to prepare a list of standard hotkeys.

These hotkeys can very usefully be generated by a profiler such as KeyGrabber to provide jogging support from Human Interface Devices or buttons and encoders attached to a keyboard emulator.

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.

This page is intentionally not used.

This chapter covers importing files and their conversion to part programs by Mach2

It assumes a limited understanding of simple G-codes and their function.

# 9.1 Introduction

As you will have seen Mach2Mill uses a part program to control the tool movement in your machine tool. You may have written part programs by hand (spiral.txt is such an example) or generated them using a CAD/CAM (Computer Aided Design/Computer Aided Manufacturing) system.

Importing files which define "graphics" in DXF, HPGL, BMP or JPEG formats provides and intermediate level of programming. It is easier than coding by hand but provides much less control of the machine than a program output by a CAD/CAM package.

The Automatic Z control feature (q.v) and repetitive execution decrementing the Inhibit-Z value is a powerful tool for making a series of roughing cuts based on imported DXF and HPGL files.

# 9.2 DXF import

Most CAD programs will allow you to output a file in DXF format even though they do not offer any CAM features. A file will contain the description of the start and finish of lines and arcs in the drawing together with the layer that they are drawn on. Mach2 will import such a file and allow you to assign a particular tool, feed rate and "depth of cut" to each layer. The DXF file must be in **text** format, not binary, and Mach2 will only import **lines**, **polylines**, **circles and arcs (not text)**.

During import you can (a) optimise the order of the lines to minimise non-cutting moves. (b) use the actual coordinates of the drawing or offset them so that the bottom leftmost point is 0,0, (c) optionally insert codes to control the arc/beam on a plasma/laser cutter and (d) make the plane of the drawing be interpreted as Z/X for turning operations.



#### Figure 9.1 - DXF import dialog

The DXF import is in the file menu. The dialog in figure 9.1 is displayed.

# 9.2.1 File loading

This shows the four stages of importing the file. Step 1 is to load the DXF file. Clicking the *Load File* button displays an open file dialog for this. Figure 9.2 shows a file with two rectangles and a circle.



Figure 9.2 - a drawing of eight lines and one circle

# 9.2.2 Defining action for layers

The next stage is to define how the lines on each layer of the drawing are to be treated. Click the *Layer Control* button to display the dialog shown in figure 9.3.

Turn on the layer or layers which have lines on them that you want to cut, choose the tool to use, the depth of cut, the feedrate to use, the plung rate, the spindle speed (only used if you have a step/direction or PWM spindle controller) and the order in which you want the layers cutting. Notice that the "Depth of cut" value is the Z value to be used in the cut so, if the

ntrol			$\overline{)}$	1) 2) 3)	Load File Layer Control Factors	La Entitie Max 40 Min	yers 4 × MaxY 46.0555 × MinY
Layer Name	On/Off	Tool #	Cut Z Coord.	Feed Rate	Plunge Rate	Spindle Speed	Cutting Order
LAYER1	🖌 On	0	+0.0000	+1.00	+0.00	1000	1
TEMPLATE	🗶 Off	EMPTY	EMPTY	EMPTY	EMPTY	EMPTY	2
LAYER2	🚽 On	0	+0.0000	+1.00	+0.00	1000	3
LAYER3	🔏 On	0	+0.0000	+1.00	+0.00	1000	4
		0	+0.0000	0.0000 / Min	0.0000 / Min	0 RPM	0
		0	+0.0000	0.0000 / Min	0.0000 / Min	0 RPM	0
		0	+0.0000	0.0000 / Min	0.0000 / Min	0 RPM	0
		0	+0.0000	0.0000 / Min	0.0000 / Min	0 RPM	0
		0	+0.0000	0.0000 / Min	0.0000 / Min	0 RPM	0
		A 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	and the second				

Figure 9.3 - Options for each layer

surface of the work is Z = 0, will be a negative value. The order may be important for issues like cutting holes out of a piece before it is cut from the surrounding material.

### 9.2.3 Conversion options

Next you choose the options for the conversion process (see step 3 on figure 9.2).

**DXF Information:** Gives general details of your file which are useful for diagnostic purposes.

**Optimise:** If Optimise is not checked then the entities (lines etc.) will be cut in the order in which they appear in the DXF file. If it is checked then they will be re-ordered to minimise the amount of rapid traverse movement required. Note that the cuts are always optimised to minimise the number of tool changes required.

*As Drawn:* If As Drawn is not checked then the zero coordinates of the G-code will be the "bottom left corner" of the drawing. If it is checked then the coordinates of the drawing will be the coordinates of the G-code produced.

**Plasma Mode:** If *Plasma Mode* is checked then M3 and M5 commands will be produced to turn the arc/laser on and off between cuts. If it is not checked then the spindle will be started at the beginning of the part program, stopped for tool changes and finally stopped at the end of the program.

*Connection Tol.* Two lines on the same layer will be considered to join if the distance between their ends is less than the value of this control. This means that they will be cut without a move to the "Rapid Plane" being inserted between them. If the original drawing was drawn with some sort of "snap" enabled then this feature is probably not required.

*Rapid plane:* This control defines the Z value to be adopted during rapid moves between entities in the drawing.

*Lathe mode:* If Lathe Mode is checked then the horizontal (plus X) direction of the drawing will be coded as Z in the G-code and the vertical (plus Y) will be coded as minus X so that a part outline drawn with the horizontal axis of the drawing as its centerline is displayed and cut correctly in Mach2Turn.

# 9.2.4 Generation of G-code

Finally click *Generate G-code* to perform step 4. It is conventional to save the generated G-code file with a .TAP extension but this is not required and Mach2 will not insert the extension automatically.

You can repeat steps 2 to 4, or indeed 1 to 4 and when you have finished these click Done.

Mach2 will load the last G-code file which you have generated. Notice the comments identifying its name and date of creation.

#### Notes:

- The generated G-code has feedrates depending on the layers imported. Unless your spindle responds to the S word, you will have to manually set up the spindle speed and change speeds during tool changes.
- DXF input is good for simple shapes as it only requires a basic CAD program to generate the input file and it works to the full accuracy of your original drawing
- DXF is good for defining parts for laser or plasma cutting where the "tool" diameter is very small
- For milling you will have to make your own manual allowances for the diameter of the cutter. The DXF lines will be the path of the centreline of the cutter. This is not straightforward when you are cutting complex shapes.
- The program generated from a DXF file does not have multiple passes to rough out a part or clear the centre of a pocket. To achieve these automatically you will need to use a CAM program

• If your DXF file contains "text" then this can be in two forms depending on the program which generated it. The letters may be a series of lines. These will be imported into Mach2. The letters may be DXF Text objects. In this case they will be ignored. Neither of these situations will give you G-code which will engrave letters in the font used in the original drawing although the lines of an outline font may be satisfactory with a small v-point or bullnose cutter. A plasma or laser cutter will have a narrow enough cut to follow the outline of the letters and cut them out although you have to be sure that the centre of letters like "o" or "a" is cut before the outline!

# 9.3 HPGL import

HPGL files contain lines drawn with one or more pens. Mach2Mill makes the same cuts for all pens. HPGL files can be created by most CAD software and often have the filename extension .HPL or .PLT.

Filename: C:\Documents and Settings\John	Cancel
Browse	Scale 40 Hpgl Units
Translations	
Pen Up 5 units	
Pen Down 1	
□ Mist □ Flood □ Spindle	
Width 124.725 Set Size before or	lext
Height 133.425 after Browsing file	
Check only for laser table	
FeedRate 60	

Figure 9.4 – HPGL import filter

# 9.3.1 About HPGL

An HPGL file represents objects to a lower precision than DXF and uses straight line segments to represent all curves even if they are circles.

The import process for HPGL is similar to DXF in that a .TAP file is produced which contains the G-code produced from the HPGL

# 9.3.2 Choosing file to import

The import filter is accessed from File>Import HPGL/BMP/JPG and the HPGL button on the dialog. Figure 9.4 shows the import dialog itself.

First choose the *Scale* corresponding to that at which the HPGL file was produced. This is usually 40 HPGL units per millimetre (1016 units per inch). You can change this to suit different HPGL formats or to scale your g-code file. For example, choosing 20 (rather than 40) would double the size of the objects defined.

Now enter the name of the file containing the HPGL data or "Browse" for it. The default extension for browsing is .HPG so it is convenient to create you files named like this.

### 9.3.3 Import parameters

When the file is opened, its maximum *Width* and *Height* in millimetres will be displayed. These sizes are after allowing for the defined HPGL units per millimetre. Although the label for Width and Height suggests that you can change these values and scale the resultant G-code, you will find it easier to alter the size of your original drawing or perhaps the HPGL units scale.

The *Pen Up* and *Pen Down* controls are the Z values (in the current unit in which Mach2 is working) to be used when making moves. Pen Up will typically need to position the tool just above the work and Pen Down will give the desired depth of cut.

If *Check only for laser table* is checked then the G-code will include an M3 (Spindle Start Clockwise) before the move to the Pen Down Z level and an M5 (Spindle Stop) before the move to the Pen Up level.

The given Feedrate is inserted as an F word at the start of the generated G-code.

# 9.3.4 Writing the G-code file

Finally, having defined the import translations, click *Import File* to actually import the data to Mach2Mill. You will be prompted for the name to use for the file which will store the generated code. You should type the full name including the extension which you wish to use or select an existing file to overwrite. Conventionally this extension will be .TAP.

#### Notes:

- The import filter is run by suspending Mach2 and running the filter program. If you switch to the Mach2Mill screen (for example by accidentally clicking on it) then it will appear to have locked up. You can easily continue by using the Windows task bar to return to the filter and completing the import process. This is similar to the way the Editor for part programs is run.
- If your .TAP file already exists and is open in Mach2, then the import filter will not be able to write to it. Suppose you have tested an import and want to change the translations by importing again, then you need to make sure that you close the .TAP file in Mach2Mill before repeating the import.
- It is generally easiest to work in metric units throughout when importing HPGL files.
- If you use the "Laser Table" option with a laser or plasma cutter then you need to check if the sequence of M3/M5 and the moves in the Z direction is compatible with initiating and finishing a cut correctly.
- For milling you will have to make your own manual allowances for the diameter of the cutter. The HPGL lines will be the path of the centreline of the cutter. This allowance is not straightforward to calculate when you are cutting complex shapes.
- The program generated from a HPGL file does not have multiple passes to rough out a part or clear the centre of a pocket. To achieve these automatically you will need to use a CAM program

# 9.4 Bitmap import (BMP & JPEG)

This option allows you to import a photograph and generate a G-code program which will render different shades of grey a different depths of cut. The result is a photo-realistic engraving.

# 9.4.1 Choosing file to import

The import filter is accessed from File>Import HPGL/BMP/JPG and the JPG/BMP button on the dialog.

The first step is to define the file containing the image using the *Load Image File* button. When the file is loaded a dialog prompts you for the area on the workpiece into which the image is to be fitted. You can use inch or metric units as you wish depending on the G20/21



Figure 9.5 – Size of photographic import

mode in which you will run the generated part program. Figure 9.5 shows this dialog. The *Maintain Perspective* checkbox automatically computes the Y-size if a given X-size is specified and vice versa so as to preserve the aspect ratio of the original photograph. I the image is in colour it will be converted to monochrome as it is imported.

# 9.4.2 Choose type of rendering

Next you select the method of rendering the image. This is defining the path of the tool as it "rasterises" the image. Raster X/Y cuts along the X axis moving the Y axis at the end of each X-line. Raster Y/X makes the raster lines be in the Y direction incrementing X for each line. Spiral starts at the outside of a circle bounding the image and moves in to the centre. Each raster line is made up of s series of straight

Load Photograph FileBMP ,JPO	Cut Style Cut Style Raster X/Y C Box Cut from outside Raster Y/X Box Cut from inside Spiral Dot Diffusion KSize 110.96 Depth 0.49 Y-Size 80. Safe Z 4 Invert (White is down)
X - Step 0 Y - Step 0	ver 1 Ver 1 Ver 1 OK

Figure 9.6 – Defining the Step-over

lines with the Z coordinates of the ends of them depending on the shade of grey of that part of the picture.

# 9.4.3 Raster and spiral rendering

As you select one of these raster methods you will be prompted by a dialog for the *step-over* values. See figure 9.6. These define the distance between raster lines and the length of the short segments making up each line. The total number of moves is the *XSize*  $\div$  *X-Step-Over* and, of course, increases as the square of the size of the object and the inverse square of the size of step-over. You should start with a modest resolution to avoid impossibly big files and long cutting times.

# 9.4.4 Dot diffusion rendering

If you choose the Dot Diffusion rendering method then you will be asked for a different set of details. Dot diffusion "drills" a series of dots, in a regular grid, in the work. Typically these will be formed by a Vpointed or bull-nosed tool. The depth of each dot is determined by the shade of grey at the point on the image. The number of dots required to cover the area is computed by the filter on the basis of the shape of the tool and the depth (relief) of engraving vou select. Figure 9.7 illustrates the data

Y 1143	Max Hadius: JU
1 1.10	Min Radius: 0
Note: a depth of 1 is a Z of1 a depth of -1 is a Z of ++1	Dots in X: 0
Angle of V-Bit: 90 degrees	Dots in Y: 0
Black Depth: 0.01 in default units (Inch or mm's)	Quality % 0
White Depth: 0.5 in default units (Inch or mm's)	Check Stats
Boundry% 5	
Output Size X: 120 Output Size X: 186.6666666	

Figure 9.7 – Dot diffusion parameters

required. Each dot consists of a move to its location, a Z move to its depth and a Z move to above the work. You must prepare your image with a suitable photo editor to have a reasonable number of pixels to control the computation load when diffusing the dots. The statistics obtained by the *Check Stats* button will give you an idea of how sensible your choice of parameters has been.

Now having defined the rendering technique you set the *Safe Z* at which moves over the work will be done and choose if black or white is to be the deepest cut.

# 9.4.5 Writing the G-code file

Finally click *Convert* to actually import the data into Mach2Mill. You will be prompted for the name to use for the file which will store the generated code. You should type the full name including the extension which you wish to use or select an existing file to overwrite. Conventionally this extension will be .TAP.

### Notes:

• The import filter is run by suspending Mach2 and running the filter program. If you switch to the Mach2Mill screen (for example by accidentally clicking on it) then it will appear to have locked up. You can easily continue by using the

Windows task bar to return to the filter and completing the import process. This is similar to the way the Editor for part programs is run.

- If your .TAP file already exists and is open in Mach2, then the import filter will not be able to write to it. Suppose you have tested an import and want to change the translations by importing again, then you need to make sure that you close the .TAP file in Mach2Mill before repeating the import.
- You will need to define the feedrate to be used using MDI or by editing the part program before it is run.
- Dot Diffusion places big demands on the performance of your Z axis. You must set the *Safe Z* as low as possible to minimise the distance travelled and have the Z axis motor tuning very carefully set. Lost steps part of the way through an engraving will ruin the job!

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



Figure 10.1 - Two possible toolpaths to cut triangle

than requiring the production of another part program.

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 "preentry 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 coordinates 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



Figure 10.2 - First cutter compensation move - Straight

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.

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



Figure 10.3 - First cutter compensation move - Arc

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.

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. 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



Figure 10.5 - Simple entry move

want to go along the contour from D. In 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.

This page is intentionally not used

# 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 coordinate system of orthogonal linear axes. Positions of the three linear motion mechanisms are expressed using coordinates 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 decreases 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 the 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 Correction 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<sup>2</sup> + dB<sup>2</sup> + dC<sup>2</sup>). 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.

# 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.

#### G and M-code reference

### 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. The units in which you specify Dwell are either seconds or Milliseconds depending on the setting on Configure>Logic

### 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
- Coordinate 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 XZplane 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. These modes are to allow the user to control the compromise involved in turning corners because a real machine has a finite acceleration due to the inertia of its mechanism.

*Exact stop* does what it says. The machine will come to rest at each change of direction and the toll will therefore precisely follow the commanded path.

*Constant velocity* will overlap acceleration in the new direction with deceleration in the current one in order to keep the commanded feedrate. This implies a rounding of any corner but faster and smoother cutting. This is particularly important in routing and plasma cutting.

#### G and M-code Reference

The lower the acceleration of the machine axes, the greater will be the radius of the rounded corner.

In Plasma mode (set on *Configure Logic* dialog) the system attempts to optimise corner motion for plasma cutting by a proprietary algorithm.

It is also possible to define an limiting angle so that changes in direction of more than this angle will always be treated as Exact Stop even though Constant Velocity is selected. This allows gentle corners to be smoother but avoids excessive rounding of sharp corners even on machines with low acceleration on one or more axes. This feature is enabled in the *Configure Logic* dialog and the limiting angle is set by a DRO. This setting will probably need to be chosen experimentally depending on the characteristics of the machine tool an, perhaps, the toolpath of an individual job.

# **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 milling) and tool tip radius (for turning)

# **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 in the normal flow so will generally be parts of subroutines.

#### G and M-code reference

Parameter	Meaning	Parameter	Meaning
5161	C29 home V	5261	Einture 2 V
5162	G28 home V	5262	Fixture 3 X
5162	C28 home 7	5262	Fixture 2 7
5165	C28 home A	5263	Fixture $3 \Sigma$
5104	G28 home A	5204	Fixture 3 A
5165	G28 home B	5265	Fixture 3 B
5166	G28 home C	5266	Fixture 3 C
5181	G30 home X	5281	Fixture 4 X
5182	G30 home Y	5282	Fixture 4 Y
5183	G30 home Z	5283	Fixture 4 Z
5184	G30 home A	5284	Fixture 4 A
5185	G30 home B	5285	Fixture 4 B
5186	G30 home C	5286	Fixture 4 C
5191	Scale X	5301	Fixture 5 X
5192	Scale Y	5302	Fixture 5 Y
5193	Scale Z	5303	Fixture 5 Z
5194	Scale A	5304	Fixture 5 A
5195	Scale B	5305	Fixture 5 B
5196	Scale 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
5222	Fixture 1 Y		values until
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
		10304	Fixture 255 A
		10305	Fixture 255 B
		10306	Fixture 255 C

Figure 11.1 - Sys	tem defined	parameters
-------------------	-------------	------------

# 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 Coordinate Systems

The machining system has an absolute coordinate system and 255 program coordinate (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 coordinates for the origin of for the axes in terms of the absolute coordinate 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  $P\sim$  (e.g. G59 P23 would select fixture 23). The absolute coordinate system can be selected by G59 P0.

You can offset the current coordinate system using G92 or G92.3. This offset will then

Letter	Meaning
Α	A-axis of machine
В	B-axis of machine
С	C-axis of machine
D	tool radius compensation number
F	feedrate
G	general function (see Table 5)
Н	tool length offset index
Ι	X-axis offset for arcs
	X offset in G87 canned cycle
J	Y-axis offset for arcs
	Y offset in G87 canned cycle
Κ	Z-axis offset for arcs
	Z offset in G87 canned cycle
L	number of repetitions in canned
	cycles/subroutines
	key used with G10
М	miscellaneous function (see Table 7)
N	line number
0	Subroutine label number
Р	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
Т	tool selection
U	Synonymous with A
V	Synonymous with B
W	Synonymous with C
Χ	X-axis of machine
Y	Y-axis of machine
Ζ	Z-axis of machine

Figure 11.2 - Word initial letters

#### G and M-code reference

apply fixture coordinate systems. This offset may be cancelled with G92.1 or G92.2.

You can make straight moves in the absolute machine coordinate 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.

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  $g_{0x} + 0.12 \ 34y \ 7$  is equivalent to  $g_{0x} + 0.1234 \ y_{7}$ 

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.

#### G and M-code Reference

- 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:

 $[1+a\cos[0]-[#3**[4.0/2]]]$ 

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. Note the comma which is required. The rest of the characters before the right parenthesis are considered to be a message to the operator. Messages are displayed on screen in the "Error" intelligent label.

### 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.

### 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 controls

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
```

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 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

#### G and M-code reference

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 ( $\sim$ ) 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  $X^{\sim}$ ,  $Y^{\sim}$ ,  $Z^{\sim}$ ,  $A^{\sim}$ ,  $B^{\sim}$ ,  $C^{\sim}$ ,  $U^{\sim}$ ,  $V^{\sim}$ ,  $W^{\sim}$ ) are given, they specify a destination point. Axis numbers relate to the currently active coordinate system, unless explicitly described as being in the absolute coordinate 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.

U, V and W are synonyms for A, B and C. Use of A with U, B with V etc. is erroneous (like using A twice on a line). In the detailed descriptions of codes U, V and W are not explicitly mentioned each time but are implied by A, B or C.

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  $L \sim is$  written in a prototype the " $\sim$ " will often be referred to as the "L number". Similarly the " $\sim$ " in  $H \sim$  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/U, B/V, C/W word and to the relevant I, J, K or R words when they are used.

# 11.7.1 Rapid Linear Motion - G0

(a) For rapid linear motion, program G0  $X \sim Y \sim Z \sim A \sim B \sim C \sim$ , 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.

#### G and M-code Reference

(b) If G16 has been executed to set a Polar Origin then for rapid linear motion to a point described by a radius and angle G0  $X \sim Y \sim$  can be used.  $X \sim$  is the radius of the line from

Summary of G-codes			
G0	Rapid positioning		
G1	Linear interpolation		
G2	Clockwise circular/helical interpolation		
G3	Counterclockwise circular/Helical interpolation		
G4	Dwell		
G10	Coordinate system origin setting		
G12	Clockwise circular pocket		
G13	Counterclockwise circular pocket		
G15/G16	Polar Coordinate moves in G0 and G1		
G17	XY Plane select		
G18	XZ plane select		
G19	YZ plane select		
G20/G21	Inch/Millimetre unit		
G28	Return home		
G28.1	Reference axes		
G30	Return home		
G31	Straight probe		
G40	Cancel cutter radius compensation		
G41/G42	Start cutter radius compensation left/right		
G43	Apply tool length offset (plus)		
G49	Cancel tool length offset		
G50	Reset all scale factors to 1.0		
G51	Set axis data input scale factors		
G53	Move in absolute machine coordinate 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/G64	Exact stop/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 coordinates and set parameters		
G92.x	Cancel G92 etc.		
G93	Inverse time feed mode		
G94	Feed per minute mode		
G95	Feed per rev mode		
G98	Initial level return after canned cycles		
G99	R-point level return after canned cycles		

Figure 11.4 - Table of G codes
the G16 polar origin and  $Y \sim$  is the angle in degrees measured with increasing values counterclockwise from the 3 o'clock direction (i.e. the conventional four quadrant conventions).

Coordinates of the current point at the time of executing the G16 are the polar origin.

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 Coordinates.

## 11.7.2 Linear Motion at Feed Rate - G1

(a) For linear motion at feed rate (for cutting or not), program G1  $X \sim Y \sim Z \sim A \sim B \sim C \sim$ , 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).

(b) If G16 has been executed to set a polar origin then linear motion at feed rate to a point described by a radius and angle G0  $X \sim Y \sim$  can be used.  $X \sim$  is the radius of the line from the G16 polar origin and  $Y \sim$  is the angle in degrees measured with increasing values counterclockwise from the 3 o'clock direction (i.e. the conventional four quadrant conventions).

Coordinates of the current point at the time of executing the G16 are the polar origin.

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 Coordinates.

## 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 coordinate 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.

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 coordinates 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 coordinate 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 coordinates 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 coordinates 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  $X \sim Y \sim Z \sim A \sim B \sim C \sim I \sim J \sim$  (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  $X \sim Y \sim Z \sim A \sim B \sim C \sim I \sim K \sim$  (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  $X \sim Y \sim Z \sim A \sim B \sim C \sim J \sim K \sim$  (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  $P\sim$ . This will keep the axes unmoving for the period of time in seconds or milliseconds specified by the P number. The time unit to be used is set up on the Configure>Logic dialog. For example, with units set to Seconds, G4 P0.5 will dwell for half a second. It is an error if:

• the P number is negative.

## 11.7.5 Set Coordinate System Data Tool and Fixture tables - G10

See details of tool and fixture offsets for further information on coordinate 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 coordinate values for the origin of a fixture coordinate 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 coordinates of the origin of the coordinate system specified by the P number are reset to the coordinate values given (in terms of the absolute coordinate system). Only those coordinates 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 coordinate 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  $\times$ 3.5  $\vee$ 17.2 sets the origin of the first coordinate system (the one selected by G54) to a point where X is 3.5 and Y is 17.2 (in absolute coordinates). The Z coordinate of the origin (and the coordinates for any rotational axes) are whatever those coordinates 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 coordinates as the centre. The tool is returned to the centre.

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

#### 11.7.7 Exit and Enter Polar mode - G15 and G16

It is possible for G0 and G1 moves in the X/Y plane only to specify coordinates as a radius and angle relative to a temporary center point. Program G16 to enter this mode. The current coordinates of the controlled point are the temporary center.

Program G15 to revert to normal Cartesian coordinates.

```
G0 X10 Y10 // normal G0 move to 10,10
G16 //start of polar mode.
G10X10Y45
( this will move to X 17.xxx, Y 17.xxx which is a
spot on a circle) (of radius 10 at 45 degrees from
the initial coordinates of 10,10.)
```

This can be very useful, for example, for drilling a circle of holes. The code below moves to a circle of holes every 10 degrees on a circle of radius 50 mm centre X = 10, Y = 5.5 and peck drills to Z = -0.6

```
G21
               // metric
G0 X10Y5.5
G16
G1 X50 Y0
              //polar move to a radius of 50 angle 0deg
G83 Z-0.6
              // peck drill
              // ten degrees from original center...
G1 Y10
G83 Z-0.6
              // 20 degrees....etc...
G1 Y20
G1 Y30
G1 Y40
> ....etc....
G15
               //back to normal cartesian
```

#### Notes:

(1) you must not make X or Y moves other than by using G0 or G1 when G16 is active

(2) This G16 is different to a Fanuc implementation in that it uses the current point as the polar center. The Fanuc version requires a lot of origin shifting to get the desired result for any circle not centred on 0,0

#### 11.7.8 Plane Selection - G17, G18, and G19

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

#### 11.7.9 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.

## 11.7.10 Return to Home - G28 and G30

A home position is defined (by parameters 5161-5166). The parameter values are in terms of the absolute coordinate 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.

## 11.7.11 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).

## 11.7.12 Straight Probe – G31

#### 11.7.12.1 The Straight Probe Command

Program G31 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.
- G31 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 2000 to 2005 will be set to the coordinates of the location of the controlled point at the time the probe tripped and a triplet giving X, Y and Z at the trip will be written to the triplet file if it has been opened by the M40 macro/OpenDigFile() function (q.v.)

#### 11.7.12.2 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.

#### 11.7.12.3 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 wellaligned 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.

```
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 G31 X[#1001 + #1005] (probe +X side of hole)
N140 #1011=#2000 (save results)
N150 G0 X#1001 Y#1002 (back to center of hole)
N160 G31 X[#1001 - #1005] (probe -X side of hole)
N170 #1021=[[#1011 + #2000] / 2.0] (find pretty good X-value of hole center)
N180 G0 X#1021 Y#1002 (back to center of hole)
N190 G31 Y[#1002 + #1005] (probe +Y side of hole)
N200 #1012=#2001 (save results)
N210 G0 X#1021 Y#1002 (back to center of hole)
N220 G31 Y[#1002 - #1005] (probe -Y side of hole)
N230 #1022=[[#1012 + #2001] / 2.0] (find very good Y-value of hole center)
N240 #1014=[#1012 - #2001 + [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=#2000 (save results)
N280 G0 X#1021 Y#1022 (back to center of hole)
N290 G31 X[#1021 - #1005] (probe -X side of hole)
N300 #1041=[[#1031 + #2000] / 2.0] (find very good X-value of hole center)
N310 #1024=[#1031 - #2000 + [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

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.

## 11.7.13 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  $G41 \quad D \sim$  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  $G42 \quad D \sim$  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.

G41 and G42 can be qualified by a P-word. This will override the value of the diameter of the tool (if any) given in the current tool table entry.

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.

## 11.7.14 Tool Length Offsets - G43 and G49

To use a tool length offset, program G43  $H\sim$ , 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.

## 11.7.15 Scale factors G50 and G51

To define a scale factor which will be applied to an X, Y, Z, A, B, C, I & J word before it is used program G51 X~ Y~ Z~ A~ B~ C~ where the X, Y, Z etc. words are the scale factors for the given axes. These values are, of course, never themselves scaled.

It is not permitted to use unequal scale factors to produce elliptical arcs with G2 or G3.

To reset the scale factors of all axes to 1.0 program G50

## 11.7.16 Move in Absolute Coordinates - G53

For linear motion to a point expressed in absolute coordinates, 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 coordinate systems.

#### 11.7.17 Select Fixture Coordinate System - G54 to G59 & G59 P~

To select fixture coordinate 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 coordinate systems.

#### 11.7.18 Set Path Control Mode - G61, and G64

Program G61 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.

#### 11.7.19 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.

#### 11.7.20 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 coordinate 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.

#### 11.7.20.1 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.

#### 11.7.20.2 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)

#### 11.7.20.3 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.

#### 11.7.20.4 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.

#### 11.7.20.5 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  $150 \ge 2 = 300$ .

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.

## 11.7.20.6 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.

## 11.7.20.7 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.

#### 11.7.20.8 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 coordinate 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.
- 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.



Figure 11.6 - G87 back boring sequence

- 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.

#### 11.7.20.9 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  $X \sim Y \sim Z \sim A - B \sim C \sim R \sim L \sim P \sim$ 

- 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.

#### 11.7.20.10 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  $X \sim Y \sim Z \sim A \sim B \sim C \sim R \sim L \sim P \sim$ 

- 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.

## 11.7.21 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 coordinate 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.

# 11.7.22 Coordinate System or Workpiece Offsets - G92, G92.1, G92.2, G92.3

See the chapter on coordinate systems for full details.

To make the current point have the coordinates 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 coordinate 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 coordinate system moves. To do this, origin offsets are calculated so that the coordinates 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 coordinate 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 coordinate 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-coordinate 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 coordinate systems. Thus all fixture coordinate 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.

## 11.7.23 Set Feed Rate Mode - G93, G94 and G95

Three feed rate modes are recognized: inverse time, units per minute and units per revolution of spindle. Program G93 to start the inverse time mode (this is very infrequently employed). Program G94 to start the units per minute mode. Program G95 to start the units per rev mode.

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.

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 units per rev 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 spindle revolution, millimetres per spindle revolution, or degrees per spindle revolution, depending upon what length units are being used and which axis or axes are moving.

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.

## 11.7.24 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
M3/4	Rotate spindle clockwise/counterclckwise
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
M47	Repeat program from first line
M48	Enable speed and feed override
M49	Disable speed and feed override
M98	Call subroutine
M99	Return from subroutine/repeat

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. M2 leaves the next line to be executed as the M2 line. M30 "rewinds" the G-code file. These commands can have the following effects depending on the options chosen on the Configure>Logic dialog:

- 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 resume the program (M2) or start the program back at the beginning of the file (M30).

## 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.

For a PWM or Step/Dir spindle the speed is programmed by the S word. For an on/off spindle control it will be set by the gearing/pulleys on the machine.

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 permitted to use M3 or M4 when the spindle is already turning or to use M5 when the spindle is already stopped but see the discussion on safety interlocks in configuration for the implications of a sequence which would reverse an already running spindle..

## 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 and to move the axes to a convenient location to tool changing if you wish.

If tool change requests are set to be ignored (in Configure>Logic) 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 Re-run from first line - M47

On encountering an M47 the part program will continue running from its first line. It is an error if:

• M47 is executed in a subroutine

The run can be stopped by the Pause or Stop buttons

See also the use of M99 outside a subroutine to achieve the same effect.

#### 11.8.6 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.

#### 11.8.7 Call subroutine - M98

To call a subroutine program M98  $P \sim L \sim$  or M98  $\sim P \sim Q$  The program must contain an O line line with the number given by the P word of the Call . This O line is a sort of "label" which indicates the start of the subroutine. It will normally be written with other subroutines and follow either an M2, M30 or M99 so it is not reached directly by the flow of the program.

The L word (or optionally 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 L (Q) word is omitted then its value defaults to 1.

By using parameters values 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.

#### 11.8.8 Return from subroutine

To return from a subroutine program M99 Execution will continue after the M98 which called the subroutine.

If M99 is written in the main program, i.e. not in a subroutine, then the program will start execution from the first line again. See also M47 to achieve the same effect.

## 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.

New macros can be written using an external editor 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.

## **11.10 Other Input Codes**

#### 11.10.1 Set Feed Rate - F

To set the feed rate, program F~

Depending on the setting of the Feed Mode toggle the rate may be in units-per-minute or units-per-rev of the spindle.

The units are those defined by the G20/G21 mode.

Depending on the setting in Configure>Logic a revolution of the spindle may be defined as a pulse appearing on the Index input or be derived from the speed requested by the S word or *Set Spindle speed* DRO.

The feed rate may sometimes be overidden as described in M48 and M49 above.

## 11.10.2 Set Spindle Speed - S

To set the speed in revolutions per minute (rpm) of the spindle, program  $S \sim$  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  $\mathbb{T}$ ~ 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.

Order	Item
1	Comment (including message)
2	Set feed rate mode (G93, G94, G95)
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 coordinate 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 or repeat (M0, M1, M2, M30, M47, M99)

Table 11.9 - Order of execution on a line

## **11.11 Error Handling**

This section describes error handling in Mach2.

Mach2 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 GO 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.

## **11.12 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 11.9 if they occur on the same line.

## 12. Supplied macros and other extensions

## 12.1 Contact probe digitising

Mach2Mill will construct a part program which will scan in a grid over a surface and construct a point cloud of Z values at the grid intersections. This "cloud" can be used as input to a solid modelling CAD package.

The Wizards>Digitise dialog is used to define the rectangle to be gridded on the X-Y plane, the X and Y grid step sizes and anticipated range of Z values so that the probe is clear of the object being digitised.

When the parameters are defined the Create and Load button will prompt for a filename to store the generated G-code (conventionally with the .TAP extension) and when this is given will load the digitising program into the G-code window. An example is shown in figure 12.2.

You must edit an F word into the start of the program to define the

ignining boarie	Rear Corner —	Z-Start Height	1	
	× 5	Z-Max Depth	-5	
	Y 5	X-Step Size	0.5	
		Y-Step Size	0.5	
'h = et				
. 0				

Figure 12.1 – Digitise wizard dialog

approach rate required for the probe or use MDI to issue a suitable F word.

The part program starts with the M40 macro which prompts for the name of a file to hold the digitised points (triplets for X, Y and Z at each point on the grid). This file is closed at the end of the part program by M41.



Figure 12.2 - sample digitising path and G-code

## 12.2 Laser probe digitising

This feature is experimental in the current release of the software for use with laser digitisers.

You use it by setting an output for "Digitize Trigger" and then enabling the Grid on the corrections screen. Zero the Laser Grid at your X,Y 0,0 point after offsetting. From that

#### Supplied macros and other extensions

point forward, any G1,G0,G2 or G3 which passes a grid coordinate will trigger a Digitize pulse output as the axis passes that coordinate. This pulse will be approx. 20-40microseconds in width. If you code a grid shaped movement, you will trigger on the intervals specified on the corrections tab.

This is a test development so that digitizer designers can play with their triggering and data to see if possibilities exist for this method. A wizard like the Touch digitising one will be provided when appropriate.

For example, if you set the grid to 5mm X and Y and then execute a G-code program which sweeps a 25 x 25 area, you will end up with a trigger at the 5mm division points of the movement. Movement does not need to stop to trigger, so a line in the above situation like G0X100 will generate a trigger at 0, 5, 10, 15, 20 etc., but you must start the gridding with a move to, say,  $G0 \times -1$  so that the move will cross the 0,0 point to trigger there as well.

You need to make your own arrangements for logging the output value from the laser probe. A serial port with a large buffer in its UART chip may be satisfactiory for this if the data is characters sent by RS-232.

## 12.3 Standard Macro definitions

To be supplied in later revision

## 13. Appendix 1 - Mach2 screenshot pullout



Mill Program Run screen



Mill MDI screen



**Mill Toolpath screen** 



**Mill Positioning screen** 

Re Configure Layours Words Lock Heb           Milling -> G3 G17 G40 G21 G90 G94 G53 G49 G99 G64           Feature Off           DRO Display Type         Fixture Number         O           Alt-C         Part X Offset         0.0000         Touch         Part A Offset         0.0000         Touch           Period         Ref All Cirt-A         Part X Offset         0.0000         Touch         Part B Offset         0.0000         Touch           Y Axis         +1.0000         Offset         0.0000         Touch         Part C Offset         0.0000         Touch           Y Axis         +0.0000         Correction         0.0000         Touch         Stock Diamter, Ouage Block thickness, etc.           Tool         Ø         Tool         Stock Diamter, Ouage Block thickness, etc.           Tool         Ø         Tool         Stock Diamter, Ouage Block thickness, etc.           Touch         +0.0000         X Offset         0.000         Teuch         Alt T         Save Tables her moke then perms           B Axis         +0.0000         Tip Radius         0.000         Jog Increment         Tool Tool	Mach 2Mill				
DRO Display Type At-C       Fixture Off         Coordinate System       Fixture Number         Mach       G92         Perid       Ref All CurlA         X Axis       +1.0000         Y Axis       +0.0000         Axis       -1.2000         Axis       +0.0000         B Axis       +0.0000         C Axis       +0.0000         C Axis       +0.0000         Total #       0.000         Tool       Stock Diamter, Guage Block thickness, etc.         Tool #       0.0000         Touch       +0.0000         X Axis       +0.0000         X Offset       0.000         Touch       Att T         Swe Tables her         make then perma         Diameter       0.000         Jog Increment	File Configure Layouts Wizards Look Help	Million > C2 C17 C40 C2	1 000 004 053 (	C40 C00 C64	
Z Axis       -1.2000       Tool # 0 Tool       Touch Correction       +0.0000         A Axis       +0.0000       Z Offset       0.000       Touch Correction       -0.0000         B Axis       +0.0000       Diameter       0.000       Touch Correction       -1.2000         C Axis       +0.0000       Diameter       0.000       Touch Correction       -1.2000         Diameter       0.000       Touch Correction       -1.2000       -1.2000       -1.2000         Diameter       0.000       Touch Correction       -1.2000       -1.2000       -1.2000         C Axis       +0.0000       Tip Radius       0.000       Jog Increment       -1.2000	DRO Display Type All-C Ceordinate System Ficture G92 Rend Ref All Otri-A X Axis +1.000	Milling -> G3 G17 G40 G2           Fixture 0ff           Fixture Number           Part X Offset           Part Y Offset           O           Offset 0n/0ff	1 G90 G94 G53 ( 0 0.0000 Touch Pa 0.0000 Touch Pa 0.0000 Touch Pa	art A Offset 0. art B Offset 0. art C Offset 0.	0000 Teach 0000 Teach 0000 Teach
	Z Axis -1.200 A Axis +0.000 B Axis +0.000 C Axis +0.000	Tool #         O         Tool           IO         X Offset         O           IO         Z Offset         O           IO         Z Offset         O           IO         Diameter         O           IO         Tip Radius         O	.000 Teuch .000 Teuch .000	Alt T	Save Tables here to make them permanent Fibture Table
Go Home Alt H	Go Home Alt H	n Run (F1) MDI (F2) Positioning (F3)	Mach2%Spiral bt ToolPath (F4) Tab	Les (F5) Corrections Mach21	Slow Jog Percentage 100.0 % Dw [ Up ] Shit Jog to overnide Jeg Mede Alt-J Continuous Incremental (F6) Diagnostics (F7)
Ready CAP 7	Ready				KAP NUM

#### Mill Tables screen

R	Mill otation Diameter	ing ->G3 S ι	G17 C aser Trig	540 G21 Iger On	Reference	Switch Locati	9 G64 ons Torch	Height Control
A Axis	+0.0000	1	Toggle aser Gri	d Zero	in machine	ABS coordina	ates Cur V	ent Correction
B Axis	+4.0000	Y	Laser	Grid	Y Axis	+0.0000	Maximu	m Correction + or
C Axis	+0.0000	Ŷ	+0.0	0000	Z Axis	+0.0000	) Highs	st +1.0000 st +0.2000
-	0.0000				A Axis	+0.0000	) Correction	Speed 20
					B Axis	+0.0000	Enable	Toggle 🛛
	Encoder Position			ř	C Axis	+0.0000	) _0	alibrate to Zero
X Axis	+0.0000	Load DRO	Zera	Alt X	Single Step on	input Activation4	Auto Limi	tOverRide 🔄
Y Axis	+0.0000	Te DRO	Zera	Alt Y	Sofware Li	nits 🛛 🗖	OverRi	de Limits 🛛
Z Axis	+0.0000	Te DRO	Zere	Alt Z	Throttle Co Stow log Dat	introl		
UNITS AR	U				Feed Rate			
MM's I	Pulley Number	Ignore T	ool Chan	9e_ 🔲		Sah	Safe_Z	+0.0000
	-		_	- C.1	Mach2\Spiral.txt			
RES	Program Run (F	MDI (F2)	Positio	ning (F3)	ToolPath (F4)	Tables (F5)	Corrections (F6)	Diagnostics (F7
	Clear						Mach2Mill	

#### **Mill Corrections screen**

Zero All         Milling -> G3 G17 G40 G21 G90 G94 G53 G49 G99 G64           × Pos         -0.9425         ×vei         -99.54         Zero X         × Referenced         Ref X         Atbet 1         Inputs           2 Pos         -1.1055         Yvei         -99.54         Zero X         × Referenced         Ref X         Atbet 1         Atbet 2         Y+Linit         Y	
X Pros       -0.9425       X:V9       -2.94       Zere X       X Referenced       Ret X       Active 1       Active 1 </th <th>0</th>	0
Y Pos       -0.1785       Y-Vel       -99.54       Zena Y       Y Referenced       Ref Y         Z Pos       -1.1055       Z-Vel       -3.36       Zere Z       Z Referenced       Ref Z       Active3       Ac	0
Z Pos       -1.1055       Z Vel       -3.36       Zero Z       Z Referenced       Ref Z       Arthvels       ZLimit       Arthvels       Art	X-Home X-Home
A Pos +0.0000 A Vel 0.00 Zero A A Referenced Ref A Referenced Ref A B-Lint B -Lint C +Lint C +	Z-Home
B Pos       +0.0000       E-Vel       0.00       Zere B       B Referenced       BeLB       Indicv       C+Linit       C+Linit         C Pos       +0.0000       C-Vel       0.00       Zere C       C Referenced       BeLC       Emergency       Torch OF       Torch OF         Edit       Biended       99,64       Program Extrema       OutPut       Enable 1       Enable 2       Enable 2       Enable 2       Enable 3       Enable 3       Enable 3       Enable 4	A-Home 8-Home
C Pos +0.0000 C-Vel 0.00 Zere C C Referenced Ret C Frequency for the first of the f	C-Home
Edia       Biended       99,64       Program Extrema       OutPut       Enable 1 Enable 5       Enable 2 Enable 5         Edia       Biended       99,64       X Range       -1.0000       +1.0000       Enable 5       Enable 5       Enable 6         View Case       Y Range       -1.0000       +1.0000       +1.0000       0.0000 <t< td=""><td>I forch Do</td></t<>	I forch Do
A Range       -1.0000       +1.0000       -1.0000       +1.0000       -1.00000       -1.00000       -1.00000       -1.00000       -1.00000       -1.00000<	Enable 4 Enable 7 Extern 3
Biggraf MD       Finglish       Finglish       Findinge       Findinge       Findinge       Findinge       Findinge       CPU Load       Inter Scale         0 0 vit V0 20       0 0 vit V0 20       Findinge       A Range       +0.0000       -9999.0000       -9999.0000       Inter Scale       Jog Intr         0 0 vit V0 20       Findinge       C Range       +9999.0000       -9999.0000       -9999.0000       Inter Scale       Jog Intr         0 0 vit V0 20 0 vit V0 0 0 vit V0 0 vit V	Lucon o
Method         2         Raining         -1,2000         +0,0000         100 %         17 %         Time Scale           00 st +0 20         2 + 1 0         0         00 st +0 20         -1 0         0         100 %         17 %         Time Scale           00 st +0 20         2 + 1 0         0         0 st +0 20         -1 0         -1 0         100 %         17 %         Time Scale           0 st +0 20 2 + 1 0         0 st +0 20 2 + 1 0         0 st +0 20 2 + 1 0         -1 0         Reget + 9999.0000         -9999.0000         -9999.0000         -9999.0000         -1 0         Method         17 %         Time Scale         Jog Incl	m +1
a) of V(25)       Run       A Radige       V0.0000	1 0000
Basit V0 2008 11 00 Basit V0 2008 11 00 Bas	ement
Bear W0200000000000000000000000000000000000	100>
Pause       Flood Teggle       Order Offset Y       ±0.0000 Ares offset Y       ±0.0000 Ares offset Y       ±0.0000 Ares offset Z       ±0.00000 Ares offset Z       ±0.0000 Are	0.010
Rewind Stop Single       Mist Toggle       Ordan Offset Z       +0.0000 Ares offset Z	de Alt-J
Stop Single Resume       Stop Single Resume       Dwell Active FeedRate       Ontain Offset A       +0.0000 Acts offset B       +0.0000 +0.0000 Acts offset B       +0.0000 +0.0000 Acts offset B       +0.0000 +0.0000 Acts offset B       +0.0000 +0.0000 Acts offset C       +0.0000	Incrementa
Single Resume         FeedRate Resume         Onicin Office18         ±0.0000 Auscoffset 8	
Resume       100.00       onion onserc       +0.0000 Avis orsetc       +0.0000       Bitting         CPU Speed       +1001.0000 Mhz       Pulse Frequency       Elacsed       00:00:0         Beach       Jog Followi       36124       Estimate       00:00:0         C: Mach2/Spiral b4       Program Run (F1)       MDI (F2)       Positioning (F3)       ToolPath (F4)       Tables (F5)       Corrections (F6)       Diag         Clear       Mach2Mil       Mach2Mil       Mach2Mil       Mach2Mil       Mach2Mil	Percentage
CPU Speed +1001.0000 Mhz Pulse Frequency Elapsed 00:00:0  Reach Jog Follow    RESET Program Run (F1) MDI (F2) Positioning (F3) ToolPath (F4) Tables (F5) Corrections (F6) Diag Clear Mach2Mil	to override
Reach     Jog Follow     36124     Estimate 00:00:0       C: Wach2/Spiral bt     C: Wach2/Spiral bt       Program Run (F1)     MDI (F2)     Positioning (F3)     ToolPath (F4)     Tables (F5)     Corrections (F6)     Diag       Clear     Mach2Mil	14
C: Wach2/Spiral bt  Program Run (F1) MDI (F2) Positioning (F3) ToolPath (F4) Tables (F5) Corrections (F6) Diag  Clear Mach2Mil	/ 1 IO Estima
RESET     Program Run (F1) MDI (F2) Positioning (F3) ToolPath (F4) Tables (F5) Corrections (F6) Diag     Glear     Mach2Mil	in the second
RESET     Program Run (F1) MDI (F2) Positioning (F3) ToolPath (F4) Tables (F5) Corrections (F6) Diag     Clear     Mach2Mil	
Clear Mach2Mil	nostics (Fi
	CIP MP

Mill Diagnostics screen





## 14. Appendix 2 - Standard shortcut codes

Shortcut	Function	Command/Op/OEMCode	Screen
-	Lower spindle speed	164	1,2
(	Slow Jog Dn	112	4,5,7
)	Slow Jog Up	111	4,5,7
/	Feedrate reset	Reset Feed Ov	1,2,7
{	Feed lower	109	1,2,7
}	Feed raise	108	1,2,7
+	Raise spindle speed	163	1,2
<	Inc Inc Down	101	4,5,7
	Stop	Stop	2
	Pause	Pause program	1
>	Inc Inc Up	100	4,5,7
A	Corrections	6	1,2,3,4,5,6,7
Alt-C	Coord System	Show Machine Coord	1,4
Alt-H	Go Home	138	5
Alt-I	Reset Interp	102	6
Alt-J	Jog mode toggle	103	4,5,7
Alt-N	Single	Single step program	1
Alt-R	Cycle start	Run program	1
Alt-R	Cycle start	Run program	3
Alt-S	Stop	Stop program	1,3
Alt-T	Touch Corr Toggle	148	5
Alt-U	Units	106	6
Alt-W	Rewind	Rewind program	1
Ctrl-A	Ref all	105	2,5
Ctrl-F	Flood	113	1
Ctrl-J	Joystick Toggle		4,5
Ctrl-M	Mist	114	1
Ctrl-O	GotoZs		1
Ctrl-S	Spindle CW	110 "M3"	1
Ctrl-V	Verify	Verify position	2
Ctrl-Z	Goto Safe Z		1
D	Diagnostics	5	1,2,3,4,5,6,7
М	MDI	2	1,2,3,4,5,6,7
0	Tables	7	1,2,3,4,5,6,7
Р	Program Run	1	1,2,3,4,5,6,7
R	Positioning	4	1,2,3,4,5,6,7
Т	Toolpath	3	1,2,3,4,5,6,7

**Note:** The shortcuts for any given layout can be found in the CSV file which can be exported using Mach2ScreenTweak

#### Standard shortcut codes

This page is intentionally not used

## 15. Appendix 3 - Screen Layout files (.SET & .LSET)

## 15.1 Roles of Screen Designer and Mach2ScreenTweak

The screen layouts used by Mach2 are stored in files (.SET for Mill and .LSET for Turn). The file to use is set by the Layout menu in Mach2 and is persistent when the program is reloaded.

Standard Layout files are distributed with Mach2. You can modify the details of individual DROs, buttons etc. (i.e. Controls) displayed by these using Screen Designer (see chapter 8).

You can make your own layouts by combining screens from the standard layouts and ones of your own (created with Screen Designer) using the utility Mach2ScreenTweak. This utility also allows you to export a layout to a comma separated variables (CSV) file so that it can be imported into Microsoft Excel or Microsoft Access for analysis. Mach2ScreenTweak will also import a layout in CSV format so allowing you to make alterations using the Microsoft utilities or your own programs.

**Warning:** If you create an invalid layout then Mach2 may behave in unstable ways. Support cannot be given to problems encountered when using customised screens so you should always retain the standard layouts so that you can use them to demonstrate and report difficulties to Support.

## 15.2 Using Mach2ScreenTweak

## 15.2.1 Introduction

Mach2ScreenTweak is a utility program aimed at OEMs, dealers and expert users of Mach2Mill and Mach2Turn who make use of the "Screen Designer". It is designed to complement rather that replace that program.

Mach2ScreenTweak is really concerned with manipulating the individual screens within a Mach2 Layout. In other words it processes .SET files.

The following is a summary of its main features:

- Import screen(s) from another layout (perhaps locally designed screens imported into the standard ones)
- Delete screen(s) from a layout
- Promote/Demote a screen in a layout so its screen number as used in Screen Designer corresponds to its "position" in normal use
- Allocate F-key numbers to screen selection buttons and input new caption text for these buttons after screens have been moved or added
- Scale down or up the size of a screen or all screens in a layout to work with a different resolution of display
- Re-order the DROs in a group so that the arrow keys traverse them logically (in columns or rows) rather than in the order in which they were created in Screen Designer
- Export a comma-separated-variable file with a record for each control on the screens of a layout to analyse and document a layout.

The code is written in Visual Basic (Version 6.0) and the source is available on request for not-for-profit use. Hopefully this will encourage others to develop and distribute screens, utilities, filters, wizards etc. in the open way in which Mach2 and its predecessors evolved.

#### 15.2.2 Installation

The utility is distributed as a .ZIP file including the executable Mach2ScreenTweak.exe, this manual and the standard Microsoft Common Dialog ActiveX controls in COMDLG32.OCX

The .ZIP also includes a sample layout (TweakSamp.set) that contains 1024 resolution screens with Caption Labels that can be used to try out appending screens from an Additional Layout.

Mach2ScreenTweak.exe can be copied into any convenient folder and run from there via a desktop or quick-start area shortcut.

If your system does not already have it, then you will need to copy COMDLG32.OCX into the folder C:\Windows\System32.

The utility can be uninstalled by deleting the .EXE file and, if specially installed, the COMDLG32.OCX

#### 15.2.3 The main screen and its buttons

Figure 15.1 shows the main screen after loading the standard 1024.SET from Mach2Mill Release 1.00.

You will find it worthwhile "playing" before you read on in this manual. Just avoid the *Save As* operation!

The **Principal Layout** table shows the contents of the persistent screen and the 15 user screens of the layout. The buttons control what is to be done with the screens in the Principal Layout.

rincipa						
	al Layou	÷				
Oper	1 SET	CSV Export	Save As All Screens in I	_ayout		- Selected Screen
oper	1.001			. (		I
:\Mac	:h2\102	4.set	Edit All Con	trols		Control Edit
s# []	Ctrle	Screen label	Shown bu	and Shown bu		Move up list
0 m   1	17	Paraistant	Showin by	and onlowin by		
	113	Program Bup	Program Burn (E1)			Maria dana Bat
2 1	88	MDI	MDL(E2)			Move down list
3	44	ToolPath	ToolPath (E4)			
1 1	56	Positioning	Positioning (E3)			Undate
5	259	Diagnostics	Diagnostics (EZ)			Buttons
	104	Corrections	Corrections (F6)			
7 .	102	Tables	Tables (F5)			
3 (	0					0-1-1-
9 (	0				1	Caraan
10 (	0				×	
ddition Oper none>	nal Layo n .SET	CSV Import				- Selected Scree Append to Principal Lavout
S# (	Ctrls.	Screen label	Shown by	and Shown by		

Figure 15.1 – ScreenTweak main screen

## 15.2.4 Manipulate all screens in layout

#### 15.2.4.1 Open .SET

This prompts for a Layout file to open as the Principal Layout. You can select Mill (.SET) or Turn (.LSET) files to be displayed in the dialog.

When opened a copy is held within the program and the .SET file is closed so it can be used in Mach2 or Screen Designer while Mach2Tweak is in use. When a file is opened then the other buttons will be enabled.

All changes are performed on the Principal Layout. The Additional Layout (see below) is used as a source to import screens.

The Principal Layout contains information on colors to be used for controls which will be saved by Mach2ScreenTweak but is not otherwise processed by it.

This button is also implemented in the File menu.

#### 15.2.4.2 CSV Save

This will prompt for the name of a .CSV file to contain information about each control in the Layout. The format is self-evident when studied in conjunction with the definition of Layout file to be provided in the Mach2Mill/Turn manuals.

The .CSV file can be imported into Microsoft Excel or, probably better, Microsoft Access for analysis and documenting the implemented controls in the standard or custom screens.

This button is also implemented in the File menu.

#### 15.2.4.3 Save As

The Save As button prompts for a file in which to save the modified Layout. You may of course save into the originally opened file but this is discouraged by the absence of a Save button.

This button is also implemented in the File menu.

#### 15.2.4.4 Edit All Controls

The *Edit All Controls* button opens the dialog for manipulation of individual controls in the Layout. For full detail see section 4.

Not all operations are available, or meaningful, across the whole Layout and have to be performed on individual screens.

#### 15.2.4.5 Edit Undo

The Edit>Undo allows you to "undo" operations performed on the Principal Layout. Undo will cause the Principal Layout to revert to its condition before the last operation or, if a sequence of several identical ones have been performed one after the other, to its condition before the first one of the sequence.

For example, suppose an MDI screen is moved up by three consecutive clicks on *Move Up* from being screen number 6. Undo would move it back from being screen 3 to screen 6. If, however, it was moved up 4 steps by *Move Up* the moved down by one *Move Down* Undo would make it move from screen 3 to screen 2, i.e. where it was before the *Move Down* which is the last operation and so is what is undone.

## 15.2.5 Manipulate selected screen

An individual screen can be selected by clicking its entry in the table. Screens are identified by their number (as used in Screen Designer), their Screen Caption, the captions on button(s) that display them (typically on the Persistent screen) and as a last resort the number of controls on the screen. Full details of this identification are given in section 5 of this document.

#### 15.2.5.1 Delete

Deletes the selected screen and moves all those below it in the list up to close the gap. *Undo* will of course undo a sequence of consecutive *Deletes*.

#### 15.2.5.2 Move Up/Down list

Moves the selected screen up or down the list as appropriate.

This has no real significance in normal use with Mach2, but it is very convenient that the screens have the same "position" in use as when they are displayed in Screen Designer. If this is not so it is difficult to remember which screen number to select in Screen Designer. Well I can never remember the number for Mach2Mill Diagnostics!

#### 15.2.5.3 Update Buttons

When screens are moved up and down the list then the buttons (typically on the Persistent screen but they can be anywhere) are updated to display the correct screen by=ut their shortcut keys (probably F-keys) and captions are not changed.

*Update Buttons* will alter any button that had a Function key shortcut in the original layout to have the F-key number corresponding to its screen number. For example screen 2 would be displayed in Mach2 on pressing F2. This is not done if the existing shortcut is not a function key. So the MDI (M) screen from older Layouts would continue to be selected by letter-M.

In addition *Update Buttons* will prompt for the new text to be on the button caption so that, for example, if it is being changed to F9 then you can enter "Short Diags (F9)"

This function is more obvious in use than it appears when explained (I hope!)

#### 15.2.6 The Additional Layout

An additional layout (which can be the same file as the Principal one) is opened and summarised on the lower part of the screen. Its only role is to be a source of screens to be appended to the Principal Layout.

A layout stored in a CSV file, such as would be exported (see above) can also be loaded. This allows mechanically processed layouts to be added to the Principal Layout.

#### 15.2.6.1 Append to Principal Layout

An entry in the Additional Layout table is selected. When the Append to Principal Layout button is clicked then its controls are placed on the first empty screen of the Principal Layout. This appears like "appending" the screen from the Additional to the Principal Layout.

There are some point to note:

- The screens in the Additional Layout are "identified" by number, label and captions of buttons that display them in Mach2. When imported to the Principal Layout, as "new" screens, the buttons are not likely to exist and of course the screen number changes. You can use the count of controls as a guide to the correct screen having been appended. If you always use Screen Caption labels (i.e those starting with a §) in your Additional Layouts then you will know what is happening.
- It is possible that the screen appended contains buttons that select other screens in the Additional Layout (or indeed unusually the screen itself). These would not have any meaning when in the context of the Principal Layout so they are not appended with all the other controls. This "disappearance" of controls is noted in a confirmation dialog after the append.
- Sequences of appends can be undone like any other operation on the Principal Layout.

## 15.2.7 Control Manipulation

Although ScreenTweak is not a replacement for Screen Designer it does have some limited features which manipulate controls rather than whole screens. These are displayed on the dialog accessed by the *Edit all Controls* and *Control Edit* buttons on the main screen. The screen is shown in figure 15.2

riginal resolution 640 x 480 800 x 600 1024 x 768 1152 x 864 1280 x 1024	New resolution 640 x 480 800 x 600 1024 x 768 1152 x 864 1280 x 1024	Re-scale controls to suit new screen resolution. Note: due to rounding of font sizes some hand tuning will be needed Re-scale
DRO Group number:	DROs are in Columns C Rows	Re-order DROs by rows or columns so arrow keys move focus logically Re-order DROs

Figure 15.2 – Editing controls

#### 15.2.7.1 Re-scale controls

The controls on an individual screen or all screens of a layout can be re-scaled to fit different resolutions of display.

Mach2ScreenTweak will try to guess the current resolution by looking at the "bottom right" control in the selected screen or the layout. If this is wrong then you can override it by clicking the appropriate radio button. Now choose the resolution of the display on which you want to use the screen or layout and click *Re-scale*.

The following should be noticed:

- The dialog automatically closes to avoid you being tempted to repeatedly Rescale (getting smaller and smaller or bigger and bigger!)
- Because the font size of captions on buttons is fixed in Mach2 you will probably find that the caption is too big at a new resolution. Screen Designer has to be used to abbreviate captions but you have the controls reasonably placed. You can also export to a CSV, use Word, Excel etc. to edit the captions and import the amended CSV.

Screens with bitmap buttons do not suffer from this problem.

• Scaling down and then back up again will, of course, cause you loss of resolution due to rounding effects.

#### 15.2.7.2 Re-order DROs

In Mach2 the arrow keys can be used to cycle through a group of DROs (e.g. all the axis DROs). The order of this cycling is the order of original creation of the DROs in Screen Designer and may not be logical.

#### Screen layout files

You can re-order the DROs on a selected screen to run down a column or across the screen as a row using the radio buttons to choose the logic. Multiple columns/rows can be ordered and small errors in the placing of the DROs are ignored.

#### 15.2.8 Screen captions and other workarounds

Screen Designer uses an admirably simple structure for Layout data. This causes some difficulties when mechanically processing Layouts. For example there is no link, other than the visual one, between labels like "Dwell" and the LED indicating it. Screens are only identified by their overall appearance rather than a "caption". This can cause problems in Mach2ScreenTweak.

#### 15.2.8.1 Screen captions

The caption problem is overcome by use of a special "Caption Label" on screens. When you design a screen to be "tweaked" you should place somewhere on it a label that starts with the "curly-S" section symbol. For example you might have **§Compact Diagnostics.** The § is easily typed by the sequence Alt-0167 on the numeric pad (Num Lock being ON). You can also paste it from the Windows Character Map utility (in Accessories>System Tools).

The Caption Label can, of course, be visible if you have the space or can be off the bottom of the screen. **Hint:** you can work on a bit of screen not normally seen in Mach2 by temporarily switching off View>Toolbar and View>Status Bar in Screen Designer.

The main screens issued with Mach2 have Caption Labels which are "hidden" in normal use below the Reset button.

#### 15.2.8.2 Buttons identify screens

As many existing screens will not have Caption Labels, Mach2ScreenTweak looks to see what buttons are programmed to display a screen and will use the button captions of the first two found to help identify a screen. Notice, as mentioned above, that this is little help on screens appended from the Additional Layout.

## 15.3 Layout file format

Mill (.SET) and Turn (.LSET) layouts are stored in binary files and must not be opened with Microsoft Word, Notepad or the like. To view a layout in "clear text" export it using Mach2ScreenTweak.

For reference, the file structure of .SET and .LSET files is given here.

## 15.3.1 Overall file format

The file consists of (a) a count of controls across all screens of the layout, (b) a record defining each control and (c) a record giving the colours to be used for displaying controls.

This is illustrated in figure 15.3

*Int32* is a four byte signed integer. As usual it is stored in

the file with the least significant byte first.

Int32 Cour	t of controls in	Layout = n		
ControlRec	ControlRec Information record for Control 1			
ControlRec		for Control 2		
ControlRec				
ControlRec				
ControlRec				
etc.				
ControlRec		for Control n		
ColorsRec = 10 x Int32 Colors for Layout				

Figure 15.3 – Layout file format

#### Screen layout files

The structure of *ControlRec* and *ColorsRec* is defined below. Notice that *ControlRecs* are variable in length (they contain several variable length strings) and *ColorsRec* consists of ten *Int32s* (i.e. 40 bytes)

## 15.3.2 ControlRec

Figure 15.4 shows the structure of each *ControlRec*. The values are integers or strings.

Integers are as defined already, some of which indicate True/False boolean values by being Zero = True, nonZero = False.

Strings follow the C Language structure of a one byte character count (unsigned so allowing 0 to 255 characters) followed by the characters, one per byte, using the Windows character set. This is illustrated in figure 15.5. Care should be taken if you try to put a " (doublequote) in a string as this character is used as the delimiter in CSV files.

Note: that although strings and hence *ControlRecs* are variable length they can be decoded by knowing the format and moving serially through the file interpreting the character counts of the strings. Direct access to records in the layout file are not possible.

	,				
Int32	Cont	rol Fu	ncti	on	
Int32	Cont	Control Type			
Int32	Disp	layed	on S	Scree	en number
C-String				Tex	t on control
C-String			G-	code	text
C-String					Path of bitmap file
Int32	Horiz	zCode			
Int32	Vert	Code			
C-String Label			abel		
Int32	Color				
Int32	OEM Code value				
Int32	HotKey code				
Int32	Flash flag				
Int32	Red/Green flag				
C-String Forma			at c	ode	
Int32	Tabbing group				
Int32	PosX1				
Int32	Pos	(1			
Int32	Pos	<2			
Int32	Pos	(2			



Most of the fields of the ControlRec are dedicated to data for a particular type of



#### Figure 15.5 – Storage of character strings

control. The values for other controls will be "undefined". The following explains the purpose of each value. The detailed coding is to be found at the Mach Developers' Network (MachDN) site. (See Frontispiece for current link)

#### 15.3.2.1 Screen

The number of the screen (1 to 15) on which control will appear. Value 0 will appear on all screens (i.e. persistent control)

#### 15.3.2.2 Type

The coded type of each control as follows:

Type code	Control
1	DRO
2	Scrollbar
3	Bitmap
4	Conventional Button
5	Jogball
6	LED
7	Label

#### Screen layout files

8	Bitmap Button
9	Manual Data Entry (MDI)
10	G-code window
11	Toolpath display

#### 15.3.2.3 Function

The function is a code starting from 0 for each type of control defining what it displays or does. The code numbers can be obtained by counting (from 0) the radio buttons in the Screen Designer dialog box that sets up the relevant type of control. The following functions are extended by OEMcodes (see below):

Control type	Function code that is extended by OEMcode
DRO	12
Conventional button	32
LED	56
Bitmap button	32

The full list of functions is listed in Layouts on the MachDN site.

#### 15.3.2.4 OEMCode

DROs, LEDs and conventional and bitmap buttons which have special purposes and are not included on the radio buttons in Screen Designer are accessed via the "OEM" radio button and an "OEM" code. This is really an extension to the Function. Public OEM codes are listed in Layouts on the MachDN site. **Hint:** To manipulate these codes in a database it is often convenient to combine Function and OEMCode by a formula such as (Function x 1000 + OEMCode) where OEMCode is set to zero if the Function does not use it.

#### 15.3.2.5 Text

For conventional Buttons, this is the caption on the button.

#### 15.3.2.6 GText

For Conventional and Bitmap buttons whose function is 33 this string is the G-code line to issue.

#### 15.3.2.7 BitMapPath

For Bitmap buttons this is the filename of the .BMP file on the user's machine. (e.g. C:\Mach2\MyStart.bmp)

#### 15.3.2.8 Horiz & Vert

For Jogball controls the code gives the axes to be jogged for horizontal and vertical "movement" of the "ball". Axis coding is X = 0, Y = 1, Z = 2, A = 3, B = 4, C = 5

#### 15.3.2.9 Label

The string displayed in a Label control. Some values like File, Error are "intelligent" and are replaced by what they describe at runtime. Labels starting with the S mark (§) are used as screen captions.

#### 15.3.2.10 Color

Used for LED controls. Green = 0, Red = 1, Yellow = 2

#### 15.3.2.11 HotKey

For Conventional and Bitmap buttons this gives the scan code of the HotKey to activate the button (or 0 for no HotKey).
## 15.3.2.12 Flash Flag

LED will flash if this flag is non-zero

## 15.3.2.13 RedGreen Flag

LED will change colour from Red to Green when signal is active. Color value is ignored

## 15.3.2.14 Tabbing Group

For DROs this defines a group of DROs The DROs in a group are selected cyclically by the up/down arrow cursor keys. The left/right arrow keys move between different DRO groups.

### 15.3.2.15 PosX1, Y1, X2, Y2

These values define the top left and bottom right coordinates of the control (in pixels) on the screen. The top left corner of the screen is X = 0, Y = 0. Controls may be drawn partially or wholly off the visible screen at the current resolution without raising an error.

## 15.3.3 ColorsRec

The format of the *ColorsRec* is given in figure 15.6. A Color32 value has its most significan byte = 0 and the following three bytes the unsigned intensity (i.e. range 0 to 255) of the colors Red, Green and Blue in decreasing significance. Thus Yellow (i.e. Red and Green) would be coded using hexadecimal notation as 0x00FFFF00 i.e. decimal 16776960.

The color chooser dialog in Screen Designer displays the individual RGB values for a control.

	-
Color32	Background
Color32	DRO
Color32	DROSelected
Color32	MDI
Color32	MDISelected
Color32	Label
Color32	DRONumerals
Color32	MDIText
Color32	G-code
Color32	G-codeText

#### Figure 15.6 Colors record

#### Screen layout files

This page is intentionally not used

# 16. Appendix 4 - Sample schematic diagrams

# 16.1 EStop and limits using relays



Figure 16.1 - Sample EStop and Limit switch connections

### Notes:

- 1. This circuit is only illustrative of one possible solution to externally connected limit switches. If you require reference switches then these should be separate and connected to Mach2 inputs.
- 2. Relay contacts are shown in the de-energised position. Limit switches and push buttons are not operated.
- 3. Holding Interface Reset pressed will allow the Mach2 Reset button to be pressed and the axes to be jogged off the limit switches. The Interface Reset will then latch.

#### Sample schematic diagrams

- 4. Relay A needs one NO contact. It must have a 5 volt coil that is at least 150 ohms (i.e. not require more than 33 milliamps to operate). Omron G6H-2100-5 is suitable with contacts rated at 1 amp 30 volts DC
- 5. Relay B needs 1 NC and 2 NO contacts. It can have any convenient coil voltage to suit an available supply. The common of this should, ideally, not be the PC 0 volt rail to avoid the long wiring of the limit and EStop switches inducing noise. The Omron MY4 series is suitable having four contacts rated at 5 amps 220 volts AC.
- 6. The LEDs are optional but useful as an indication of what is happening. The current limiting resistor for the Interface OK LED needs to be 1.8 kilohms if a 24 volt supply is used.
- 7. If the coil voltages are suitable then the contactors can use the "Control" positive and common supply.
- 8. The arrangement of contactors (Coils shown as C1, C2, C3) depends on your drive power supply arrangements and the wiring of the motors in the machine tool. You should aim to switch the DC supply to the steppers and/or servos after the smoothing capacitor to ensure a prompt stop. You may wish to rewire the spindle and coolant motors so that the control contactor does not trip the no-volt release circuitry (i.e. you may wish to switch the motor leads after the main machine contactors. Do not share contacts on a given contactor between AC mains and the stepper/servo DC supply on account of the greatly increased risk of a short circuit between these supplies. Seek advice if you are unsure, especially before working with 230/415 volt 3-phase circuits.
- 9. The catching diodes across the relay and contactor coils are needed to absorb the back emf when switching the current off in the coils. Contactors may come with suitable coil suppression circuits built in.



# **16.2 Torch Height Control interface**

Figure 16.2 – Sample interface for Miller plasma cutter unit

Sample schematic diagrams



Figure 16.3 – Interface from modified plasma cutter to Mach2 parallel port

#### Notes on THC circuits:

- The diagrams in this section were developed by Tom Caudle to whom many thanks are due. They are included to illustrate how Mach2 interfaces with a plasma system. You are strongly advised to purchase a fully engineered system or custom retro-fitting kit. Such a system will come with complete set-up instructions and quickly save its cost in economical use of consumables and quality of finished work during the time you would be commissioning your own circuits.
- 2. The circuits are designed to give isolation between the THC and the computer parallel port(s) The internal plasma card (figure 16.2) located inside the plasma torch controller uses the plasma internal low voltage DC to operate relays to provide isolation of signals. The THC controller circuit (figure 16.3) has opto isolators to keep dangerous torch voltage and ground currents away from the computer.
- 3. The functional blocks on figure 16.2 are: (a) a potential divider and filter network for sensing the arc voltage; (b) a relay to simulate the operation of the torch trigger switch and (c) a relay to provide a contact closure when the arc is proved good by the plasma controller.
- 4. The functional blocks on figure 16.3 are: (d) Comparators to establish the highest and lowest acceptable arc voltages. These drive the up and down outputs which are Active Lo logic. The deadband between these voltages is adjustable. (e) Comparators to raise a fault condition if the arc voltage strays beyond limits wider than the control limits; (f) Optical isolation and front panel LED indicators for these signals. (g) A regulated DC supply fed from an isolated external transformer winding. (h) Connections for a panel mounted potentiometer (ideally 10 turn) to set the ideal torch height arc voltage and a panel mounted voltmeter to monitor the actual arc voltage.
- 5. Tom's conditions for release/use of the details in this manual and directly from him are:
  - If you have a registered copy of Mach2 you may make a THC controller for your **own personal use** or to **give** to another registered Mach2 user for their personal use.
  - The THC Mach2 Controller Circuit (fig 16.3) and the THC Internal Plasma Card schematics and Printed Circuit Board artwork are copyrighted original works. You may **not** make copies beyond what is required to accomplish the goals stated above. You may **not** make additional printed circuit boards to **resell**, or **sell kits** or **finished units** based on the designs.
  - It is Tom's intent to offer this low cost THC solution to Mach2 users. If you would like a copy of the printed circuit layouts for the circuits please contact Tom directly. He is available at <u>info@tcaudle.com</u>
  - If you want to distribute the THC along with Mach2 for commercial resale or have any questions about the THC and/or it's operation then please contact Tom directly. He is available at <u>info@tcaudle.com</u>
  - Tom's Disclaimer: "I am not responsible for any damage or injuries caused by the THC MACH2 Controller Circuit. The user assumes all responsibility for any use of the circuit or the printed circuit board layout. Plasma Torches and CNC equipment are dangerous and should be worked on by experienced persons."

# 17. Appendix 5 - Mach2 defined VB script functions

```
Running code
  Subroutine Code (text as String)
LEDs
  Function GetLED (ledFun as Integer) as Boolean
  Function GetOEMLED (ledOEMCode as Integer) as Boolean
DROs
  Function GetDRO (droFun as Integer) as Double
  Function GetOEMDRO (droOEMCode as Integer) as Double
  Sub SetDRO (droFun as Integer, newValue as Double)
  Sub SetOEMDRO (droOEMCode as Integer, newValue as Double)
Button Commands
  Sub DoButton (buttFun as Integer)
  Sub DoOEMButton (buttOEMCode as Integer)
Named variables
  Function GetParam (name as String) as Double
  Sub SetParam (name as String, newVal as Double)
Access to the machine G-code parameter block
  Function GetVar (PVarNumber as Integer) as Double
  Sub SetVar (PVarNumber as Integer, newVal as Double)
Arguments of macro call
  Function Param1 () as Double ' gets P word
  Function Param2 () as Double ' gets Q word
  Function Param3 () as Double ' gets S word
Information from and to the user
  Sub SetUserLabel (number as Int, text as String)
  Sub SetTicker (number as Int, text as String)
  Function Question (prompt as String) as Double
  Sub GetCoord (prompt as String)
  Function GetXCoor () as Double
  Function GetYCoor () as Double
  Function GetZCoor () as Double
  Function GetACoor () as Double
Input/Output signals, a serial port and "foreign" ports
  Function IsActive (sigNo as Integer) as Boolean
  Sub ActivateSignal (sigNo as Integer)
  Sub DeActivateSignal (sigNo as Integer)
Serial port
  Sub SendSerial (chars as String)
Foreign ports
```

#### Mach2 defined VB Script functions

Function GetPortByte (pAddr as Integer) as Byte

Sub PutPortByte (pAddr as Integer, bData as Byte)

#### Waiting/Control features

Function IsMoving () as Boolean

Returns true if there are any commands queued in Mach2 or the trajectory planner has any movements in it. It will return false when drawing a toolpath or simulating a run.

Function IsLoading () as Boolean

Returns true if the part program is loading (e.g. so the toolpath is being generated). This can be used to inhibit script actions like Question ().

#### **Other routines**

There are other special purpose and "legacy" Mach2 functions that can be called from the VB Script of a macro are given in a file in the MachDN>Macros>ForMacroWriters folder on the Mach2 Developers' Network site. (See title page for current link)

# 18. Appendix 6 – General utility programs

## 18.1 KeyGrabber

## 18.1.1 Overview

Chapter 8 gives a general description of the control of Mach2 including Human Interface Devices (HIDs) and keyboard emulators.

KeyGrabber is a utility written by Les Newell, to whom very many thanks are due, which translates signals from the keyboard, a keyboard emulator or the buttons, Point of View control or axes control of one or more HIDs into keycodes in Mach2's input buffer. These keycodes can then be interpreted by Mach2 as jogging hotkeys, screen button/DRO hotkeys or to activate/deactivate simulated signals. They could be entered into DROs or the MDI box if you want to implement a numeric keypad or the like..

KeyGrabber, unlike some general profiler programs, knows that it is running with Mach2 (actually it assumes responsibility for starting Mach2 or ScreenDesigner) and so will direct the HID buttons and the specially defined keycodes from the keyboard emulator or, indeed, the actual keyboard to Mach2 even if another program has the focus. Hence its name – it grabs keycodes for Mach2's use.

Many keyboard emulators were originally developed for use by the computer gaming community to interface the controls of an arcade style cabinet to the PC using the MAME standard codes. Some, however, can be configured to generate other codesets. KeyGrabber has a facility for programming the Ultimarc IPAC/2 and IPAC/4 with a set of codes different from any that can be produced by the PC keyboard. It also identifies these codes using the signal names of the IPAC.

KeyGrabber will implement Typematic repeating of keycodes from any of its possible inputs. This is useful for operating screen buttons for spindle speed control, and feed and jograte override. This Typematic hat three settings for each input. It can be set to be off, to be at one speed or to change (typically increase) speed after a defined number of repeats.

HIDs can be configured to have up to four separate meanings to each button. This is particularly useful on devices where the buttons are easy to label such as a membrane keyboard like the Saitek P8000 (aka Dash/2). The definitions are defined as pages 1 to 4. You can define arbitrary buttons on the HID to be the page selectors. Buttons such as "Fire" and "Shift" are often convenient choices.

When Mach2 is running without KeyGrabber it not only recognises the keycodes but interprets them according to whether the *Shift Ctrl* or *Alt* keys on the keyboard are simultaneously depressed. These keys are called **modifiers**. For example the standard layout uses Alt-R for the Cycle Start hotkey, the *Ctrl* key will switch Jog modes between Inc and Cont while it is depressed and *Shift* will cancel a jograte override. KeyGrabber can be used to define arbitrary HID buttons to produce the *Shift Ctrl* and *Alt* modifiers

Finally in this overview, low resolution encoders are a useful human interface to Mach2 for implementing an MPG style axis jog dial and for rotary panel controls for speed and feed override. KeyGrabber will interpret a pair of inputs to a keyboard emulator as quadrature A and B signals and translate these into a keydown/keyup of a user defined keycode for each clockwise "click" and a similar sequence for another keycode for each counterclockwise "click". These codes can operate the incremental jog hotkeys or the hotkeys for up/down buttons within Mach2.

Although, in principle re-mapping codes and the other functions of KeyGrabber are straightforward, the detail can be confusing. You may find that this appendix is easier to understand if you experiment with the software and devices as you read it.

## 18.1.2 Installation

#### 18.1.2.1 The files

The KeyGrabber software is automatically installed by the Mach2 installer. You may wish to setup a desktop or quickstart bar shortcut to it. If you do not start the KeyGrabber program then its installation will have no effect on your system.

#### 18.1.2.2 Windows compatibility

You will get the best results out of KeyGrabber by making sure that your Windows has been upgraded to include DirectX version 9. See the Microsoft website for full details of upgrades.

### 18.1.2.3 Running KeyGrabber and then Mach2

KeyGrabber can be run by double-clicking the KeyGrabber.exe file, in the Mach2 folder, or a shortcut to it. You will need to do this to set up how keys and buttons are to be interpreted. You can then run ScreenDesigner or Mach2 and choose the Mach2 profile by hand.

For everyday use, most Mach2 users will either use the standard Mach2Mill or Mach2Turn shortcuts that are setup when Mach2 is installed. These can easily be adapted to run KeyGrabber first and then automatically start Mach2 with the originally specified profile (i.e. Mill, Turn or indeed your own custom Mach2 profile).

Right click on the shortcut, choose Properties from the menu and modify its Target property by replacing "Mach2.exe" with "KeyGrabber.exe". Leave the text starting /p alone – this is what specifies the Mach2 XML profile to use. See chapter 5 for further details of Mach2 profiles. You may want to use a different KeyGrabber configuration for each Mach2 profile. Details of how to do this are given below.

### 18.1.2.4 Shortcut Icons

If you setup a shortcut on the desktop as described above then you will find that its Icon is the KeyGrabber smiley face. This is a useful reminder that you are using keygrabber. If you wish you can replace the Icon in the shortcut with one from Mach2 or indeed from any other place.

Right-click on the shortcut and choose *Properties* from the menu. Click *Change Icon*... Use *Browse* to choose the program file containing the icon you wish use for the shortcut. This is probably Mach2.exe. Then choose the icon out of the possible set in the display. Some programs may have several others only offer one.

If you have a graphics program which will create Windows Icons then you can design your own and use it by browsing to its .ico file rather than a .exe file.

## 18.1.3 Configuring KeyGrabber

When KeyGrabber runs for the first time it will create a file called Default.grab and this name is displayed in the caption of its window.

You can use File>Save As to save your current configuration under a different name and File>Open (or information in the shortcut Target described later) to specify that you want to use a non-default configuration.

There are four area to configure. You may not wish to use all of them and may, of course, not have the hardware to do so.

- Keyboard Keys
- ♦ Keyboard Encoders
- Human Interface Devices (HID)
- Misc Settings

The KeyGrabber window has a tab for each. Additional tabs are displayed when HID devices are enabled.

The tabs are described in the following sections:

## 18.1.4 Configuring Keyboard Keys

😬 KeyGrabber - Default.grab	2		
File IPac Testing Help			
Keyboard Keys Keyboard Encoders HID control	llers   HID keys   HID encoders   Misc Settings		
Source key	Translate to	Options	^
Undefined	Undefined		
Undefined	Undefined		~
Start Screen designer	Start Mach2	11	1

Figure 18.1 – Configure Keyboard Keys table

This tab shows a table of keycodes which may be received from the Windows keyboard port and the corresponding codes to be sent to Mach2. These codes can, of course, come from an actual keyboard or a keyboard emulator. See figure 18.1.

Either double-click or right-click and choose Define in an entry in the *Source key* column. This allows you to define the code you want to be processed. You will be presented with a dialog asking you to Press a Key. On an emulator this corresponds to making the circuit connected to the emulator input pin. If the code corresponds to one programmed into an IPAC then you will be told the terminal label and pin for this input. If the code corresponds

HeyGrabber - Default					>
Keyboard Keys Keyboard En Press	a key D keys   H	ID encoders   Misc 9	ettings		
Source keyC	ancel anslate to			Options	^
IPac BD1 PL1 SW 1	ClaiKeys Shift Control	User key 1 User key 2	User key 17 User key 18		
Undefined Undefined		User key 3 User key 4	User key 19 User key 20	-	
Undefined Undefined		User key 5 User key 6	User key 21 User key 22		
Undefined		User key 7 User key 8	User key 23 User key 24	-	
Undefined		User key 9 User key 10	User key 25 User key 26		
Start Screen designer		User key 11 User key 12 User key 13	User key 27 User key 28 User key 29		1
	-	User key 14 User key 15	User key 30 User key 31		•
		User key 16	User key 32		

Figure 18.2 – Defining special Keys

to a key on the keyboard then you will be shown the character on the key (e.g. Q) or a description of the key (e.g. LEFT for the left cursor arrow).

Where meaningful, a default value of the code to be sent to Mach2 will have been put into the *Translate to* column. A double-click or right-click in this entry allows you to redefine it to any translated value. Good keys are the ones on the numeric keypad, function keys or multimedia special function keys. Any keyboard emulator code that does not have a default but which you do not want to translate can be entered by making the circuit connected to the emulator input pin again with the *Translate To* Press a Key box showing.

An alternative to pressing a "key" is to choose *Special Keys*. This displays the options shown in figure 18.2. You can say that the "key" which you are defining is to be the *Shift* or *Ctrl* modifier or one of 32 unique user key values. These are chosen so as not conflict with other actual codes. The *Alt* modifier is not available from the keyboard as Windows makes priority use of it.

When Mach2 (or Screen Designer) is running then any keycode that is **not** in the first column, which will be most of them on you system, will be ignored by KeyGrabber and sent by Windows to the application which has the focus when the keycode arrives. The corollary of this is that any key whose code **is** in the list will not be seen by any application other than Mach2 while Mach2 is running with the KeyGrabber. You will be warned when you try to setup keys on the standard keyboard that this will happen. It does become rather annoying but it might help you being too confused when you first "grab" A, Q, Z and M and find that Microsoft Word cannot input Monday (ondy) or Zebra (ebr) or worse you cannot get a password with any of these letters in it accepted!

The final Options column allows you to choose if typematic is to apply to the keycode and if modifiers will be sent with it. Double-click or right-click and define the cell you want to set. Figure 18.3 shows the dialog.

The parameters for Typematic which apply to all codes are defined on the Misc. Settings tab. If you select *Two stage typematic* then the *Translate to* keycode will be sent to Mach2 at the specified *First rate* until *Number at first rate* codes have been sent. Codes will then be sent at *Second rate*.

Keyboard Keys   Keyboard Encoders   HID controllers	HID keys	Key options	1		
Saitek P8000		No typematic	💌 Pag	e 1	•
Source button	Translate	C Two stage typematic		Options	^
Button 1	1 Unknov	Single stage typematic			
Button 2	Undefine	🗖 Block shift modifier			
Button 3	16 SHIFT	Block control modifier			
Button 4	Undefine	Block alt modifier			
Button 5	Undefine	[			
Button 6	Undefine	<u>i</u>	2		
Button 7	Undefine	d			
Button 8	1243 <sh< td=""><td>HFT&gt;[</td><td></td><td></td><td>~</td></sh<>	HFT>[			~

#### Figure 8.3 – Key translation options

The Block modifiers check boxes are provided to allow you to specify that the state of the corresponding modifier button/key is to be ignored (i.e. not sent to Mach2) for this *Source button*. This might be used for a signal coming from a switch like a Home switch where the same code is to be recognised irrespective of the *Shift*, *Ctrl* or *Alt* keys being depressed at the time.

**Note:** If you have a keyboard emulator which provides hardware key repeat (typematic) on its inputs then KeyGrabber will ignore this.

## 18.1.5 Configuring Keyboard Encoders

This tab shows the configuration of up to 16 encoders connected to keyboard emulator pins. These can be any device that produces a quadrature output, such as MPGs, digital pots etc.

Double-click or right-click and Define in either an A channel or B channel entry in the table. KeyGrabber will display a dialog showing an encoder count and two code values. See figure 18.4.

Rotate the encoder clockwise slowly. KeyGrabber will notice the two inputs that are changing and put their codes against the Phase A code and Phase B code labels. If the count value decreases then click the *Reverse Direction* button. If your encoder has click detents then you might find that it counts two for each click. If so check the *Half Speed* box.

Because KeyGrabber is looking for two related signals it will get very confused if more than two change during this setup process so make sure that you do not type on the keyboard or have signals or noise on other pins of the keyboard emulator.

ay pressiau
urn the encoder. the counter goes the rrong way then press he reverse direction utton
Count = 7
'hase A code = 265 'hase B code = 266
📕 Half speed
Reverse direction

Figure 18.4 – Finding the encoder

When you are happy with the way your encoder counts then click OK.

You now have to configure the entries in the *Up Key* and *Down Key* columns of the table. Double-click (or right-click Define) the relevant cell and enter the required code (see figure 18.5).These will typically be the hotkeys that are setup in Configure>Axis Hotkeys and used to jog the Mach2 axis involved or be the hotkey codes on the buttons to increment/decrement the feedrate or spindle speed.

**Note 1:** Some USB keyboard emulators are limited to a maximum of 8 keys simultaneously pressed. Each encoder can press up to 2 keys so do not use more than 4 encoders on this type of emulator

(3 if you are using it for other functions as well). You cannot use an emulator which has repeats on these inputs.

Note 2: The response speed of the keyboard system is limited so don't expect to connect up a 2000 cpr encoder and spin it fast! About 25 Hz (40 msecs for a cycle of the A or B inputs) seems to

Keyboard Key	is Keyb	pard Encoders	HID devi	ices   HID keys   HI	D encoders	
A channel		B channel		Up key	Down key	
IPac BD1 PL	2 SW 7	IPac BD1 PL1 SW 8		LEFT	RIGHT	
Undefined	-	Undefined		Undefined	Undefined	
Undefined	Key p	ressed		Undefined	Undefined	
Undefined	Due	S a key Cancel		Undefined	Undefined	
Undefined	Pres			Undefined	Undefined	
Undefined				Undefined	Undefined	
Undefined	-			Undefined	Undefined	
Undefined		Undefined		Undefined	Undefined	
Undefined		Undefined		Undefined	Undefined	
		t.		A contraction of the		

Figure 18.5 – Defining Up and Down keys

work reliably on a typical Mach2 configuration. This is 10 rpm on a 32 cpr digital potentiometer. For jogging or the like a few steps lost at speed would not really matter as you will probably be watching the work or DROs.

## 18.1.6 Configuring HIDs

#### 18.1.6.1 Preparation for HIDs

Human Interface Devices (HIDs) support many different types of control. Example include an "analogue" or proportional joystick or throttle (termed by Microsoft, slightly confusingly for us, Axes), a Point of View (POV) pad or hat (like an 8 way digital joystick) and buttons.

If your HID has more than 32 buttons then you must have DirectX version 9 installed in Windows.

Plug in your HID's USB interface. Windows should recognise it with a chime or, if it is the first time you have connected, it should recognise the new device (by name) and install drivers for it. You **should not** install or run any profiler that was supplied with your HID.

You should then be able to check the basic operation of your HID using the Game Controllers option of Windows' Control Panel.

#### 18.1.6.2 The HID Controllers tab

When you run KeyGrabber, the HID Controllers tab should show a list of all the devices you have currently plugged in. If you double-click in the *Use device?* Column this will toggle the clicked device in and out of use. Right-click or the Properties button will confirm the configuration of the device. Each device that is "in-use" has a flashing watchdog confirming (or otherwise) communication with it. A red signal indicates a problem.

When at least one HID is in-use, additional tabs are displayed.

- ♦ HID Keys
- HID Encoders

#### 18.1.6.3 HID Keys

On this tab, see figure 18.6, use the drop-down list to choose the HID you wish to set up. The table on the HIDs tab is unlike the *Keyboard Keys* tab as Windows knows how many buttons a HID has (see Properties) and the table has one line for each HID key/button. You

😬 KeyGrabber - Default.grab			X
File IPac Testing Help			
Keyboard Keys Keyboard Encoders HID controllers	HID keys HID encoders Misc Settings		
Saitek P8000	▼ Pag	e1	•
Source button	Translate to	Options	^
Button 2	Undefined		
Button 3	16 SHIFT		
Button 4	Undefined		
Button 5	Undefined		
Button 6	Undefined		
Button 7	Undefined		
Button 8	1243 <shift>[</shift>		
Button 9	1245 <shift>]</shift>		V
Start Screen designer	Start Mach2	1	1

#### Figure 18.6 – After a HID button is pressed

select the Page you are defining by the drop-down. When Mach2 is running then Page 1 will be used for translation if no Page Select button is pressed. The other pages are selected by the buttons configured as described below. This selection is always instantaneous (i.e. it only applied while the button is depressed).

Now press a button on the HID. Its line is selected. By double-clicking or right-click and Define the *Translate to* column then the keycode that this button is to *Translate to* can be defined by pressing the appropriate key.

A key can be defined to select an alternative *Gain* for the Joystick axes. See Misc Settings for more details.

Special keys like *Shift* and *Ctrl* and *User defined* keys can be set up in exactly the same way as with the *Keyboard Keys* dialog but with the additional option to define a button as a HID Page Select code. By default when you apply a page select code this will be put on all pages. This can be overidden if required but if you do this you may make it difficult for the user to access.

Exactly the same procedure is followed to assign codes to POV controls and HID analog axis controls. They can be used for page selection. For details of how an axis position is converted to the relevant entry in the table see the section on Misc Settings below.

#### 18.1.6.4 HID encoders

This feature is not currently implemented

board Keys   Keyboard Encoders   HID contro	ollers   HID keys   HID encoders   Joyst	icks Misc Settings Diags
Typematic Initial delay 300 millisecono First rate 3.33 per secono Number at first rate 5 Second rate 10 per secono	Axis parameters Threshold 1 1000 Threshold 2 1000 Delay 100 ms	Joystick sensitivity button Button up 100 % Button down 100 %
Axis number 2 Axis speed	0 ThreadID 0	

#### 18.1.6.5 Misc Settings



Typematic settings are described in the section on Keyboard Keys Options. Notice that the same delay, rates and wait count apply to all keys with Typematic enabled whether they be on a keyboard, keyboard emulator or HID.

An Axis is a direction of analog control like a joystick or position of a throttle or steering wheel of a game controller. Each axis generates a value in the range -1000 to +1000 depending on its position. Some controllers generate axis signals from switches (rather like POV controls or indeed which double as POV controls). The switches will generally only produce the maximum positive and negative values.

The magnitude of values which trigger KeyGrabber to output a keycode are set in the dialog under Axis parameters. When the axis value numerically exceeds Threshold1 then the code in *Axisn Plus* or *Axisn Minus* will be selected. When the value exceeds Threshold2 then the code in *Axisn Plus2* or *Axisn Minus2* will be selected the other values remaining in the keydown state. This could happen very quickly and confuse Mach2 with what appear to be simultaneous key presses. The Delay time allows the two events to be separated. A time of 300 milliseconds is a useful general setting.

Axes can be joysticks and send "analog" values to Mach2. The sensitivity of these is set by their Gain. This can be altered for each individual axis at setup time (see below) but "high"

and "low" gain can be chosen dynamically for all axes using a HID key. When the configured key is pressed the Joystick Sensitivity *Button Down* value is used. If the key is not pressed then the *Button Up* value is used. This can be used to have high speed jogging for fast positioning with a button pressed and a very fine control when it is released.

## 18.1.7 Axes as joysticks

Axes which generate a range of values can very usefully be configured to jog axes and override feedrate, jogging speeds and spindle speed. These parameters are set up on the Joysticks tab (figure 18.8)

The dropdown selects which HID to configure. Move the control in the direction you want to configure to highlight the appropriate row of the table.

😬 KeyGrabber - Default.grab				
File IPac Testing Help				
Keyboard Keys Keyboard Encoders HID	) controllers   HID keys	HID encoders Joys	ticks Misc Setting	s Diags
WingMan Action Pad				•
Axis	Controls	Gain	Response curve	Reverse
Joystick Axis 1	Keyboard jog speed	70 %	Linear	No
Joystick Axis 2	Y axis	100 %	Linear	No
Joystick Axis 3	Z axis	100 %	Exponential2	No
Start Screen designer	Start	Mach2	Watchdogs	

Figure 18.8 – Configuration of joysticks

The *Controls* column allows you to define the function for the axis. These should be self explanatory except for Axis Gain. This allows you to use a axis on the HID to control how sensitive the other axes will be.

The *Gain* column is used to set the relative sensitivity of each axis on the HID (e.g. how much displacement of a joystick gives a given jogging speed). This is overidden by the gain defined in Misc Settings if you have defined a *Joystick Sensitivity* button.

The *Response Curve* is the relationship between the displacement of the controld and the signal sent to Mach2. The Exponential values give a small output for positions near the center (zero) value but large values next the extremes. This can allow, for example, very sensitive control of jogging at low speeds but the possibility of very fast jogging when the joystick is held fully over.

*Reverse* allows positive values to be swapped for negative ones to cater for the way in which the HID is designed or mounted.

## 18.1.8 Multiple machines and KeyGrabber Profiles (.GRAB files)

If you use more than one machine controlled by a computer the you will probably want to configure your HID and keyboard emulator differently for each. This is done but having several .GRAB files. As mentioned above, you can Save As a configuration in a file other than Default.grab and can use File>Open to open a non-default configuration.

It would however be more convenient to automatically associate the correct .GRAB with its Mach2.

This is done by including the name of the .GRAB file to be used in the Target field of the properties of the shortcut used to run KeyGrabber. The .GRAB file name comes before /p

which specifies the Mach2 profile to use. Thus for example if the original shortcut to run Mach2 to control your lathe was:

C:\Mach2\Mach2 /p Mach2Turn

Then replacing it with a shortcut with a Target of:

C:\Mach2\KeyGrabber.exe Turn.grab /p Mach2Turn

Would run Mach2 with the Mach2 profile Mach2Turn.xml after setting up KeyGrabber with the Turn.grab configuration.

**Hint:** If you have several shortcuts for different KeyGrabber and Mach2 profiles then you are very likely to want to setup different icons for each of them as described above.

## 18.1.9 Keyboard Emulator programming

Your keyboard emulator will probably come programmed with the MAME standard gaming keycodes and may also allow you to program a set of your own codes. In this case you can of course use any values but given that KeyGrabber is going to translate them it is probably best to program in a set of codes that do not clash with those that the actual keyboard can generate. Values decimal 264 upwards are probably suitable.

KeyGrabber will automatically program the Ultimarc IPAC range of emulators with such a range of unused keycodes using the IPAC menu. You choose the IPAC configuration and the speed at which you wish to program. Start fast and slow down if the programming fails.

**Beware:** To program an IPAC you need to set the MAME/Alt jumper correctly. Programming an IPAC will loose any existing codes stored in its "Alt" memory. The MAME codes are not overwritten.



Figure 18.9 – Ultimarc IPAC/2 keyboard emulator

## 18.1.10 Testing and troubleshooting

The Testing menu allows you to display a dialog which will give you the code and mnemonic for any key seen by KeyGrabber. This will test the actual keyboard, switches on a keyboard emulator, encoders on an emulator, and the buttons, POV and axes on an enabled HID. You can use this to check that your wiring is correct and to discover the code that the keyboard or emulator is actually sending to KeyGrabber.

**Note:** For IPAC users. Notice that the IPAC supports LED outputs indicating the state of the Num Lock, Caps Lock and Scroll Lock signals of the PC keyboard. These outputs are

shared with three input terminals. These effectively cannot be used, because they are pulled lo, if the corresponding LED is on.

The following are limitations of version 3.4 of KeyGrabber.

Windows 2000 command prompt windows intercept keyboard messages before they reach KeyGrabber. If you have a command prompt window or a DOS application in the foreground then KeyGrabber will not work from the keyboard.

# **19.** Appendix 7 - Record of configuration used

This allows you to record which pins on the parallel ports you use for which functions. If you use it, it could save a lot of time when your memory has grown dim!

If you have used the setup worksheets then you may wish to keep these in a safe place in case you need to repeat the process.

A complete Mach2 configuration includes a lot of detailed information. You will not wish to repeat the process step by step when you update your computer.

Mach2 profiles are .XML files and you will probably keep them in the Mach2 folder. Use Windows Explorer to find the profile you wish to copy and drag it to another folder **while holding down the Control key**. You can of course use any other file copying technique if you prefer.

If you double-click the file name then your web browser (probably Internet Explorer) will open the .xml file and display it



Figure 18.1 – Internet Explorer displaying Profile

The XML file can be edited by a text editor such as Notepad but this is not recommended,

The profile file can be useful information to attach to an e-mail when asking for suooprt from ArtSoft Corp

# 20. Revision history

Rev A4.5	2 March 2004	UserLabels and Tickers added G15/G16 added THC Anti-dive added Angle Discriminator CV added Joysticks in KeyGrabber added
Rev A4.1	14 December	Page numbering scheme revised KeyGrabber documentation extensively updated for KG Release 3.4
Rev A3.3	30 November	Inclusion of new Screen Designer features and application
Rev A3.2	18 November	Customisation chapter 8 restructure to reflect likely use of VB Script attached to buttons rather than macros and that setting such script and hotkeys for buttons and DROs is the most likely use of Screen Designer.
Rev A3.1	13 November 2003	Revision of description of keyboard emulation and documentation of KeyGrabber
Rev A2.7	21 October 2003 for Rel A1.7	Customisation chapter 8 totally rewritten to reflect new macro/Mach2 interface functionality, custom keyboards, signals derived from keyboard emulation and OEM function input pins. Terminology of Home, Reference and Zeros brought into line with agreed usage.
Rev A2.6	23 September 2003 for Rel 1.2	MSG, comment syntax added Units and Save Z control group redefined and has CV Mode and Offline function added to it Macro VB functions available updated Toolchanger option in Configure>Logic MPG jogging configuration added
Rev A2.5	18 September 2003	MPG/Encoder jogging added Preset increments for Inc Jog added. Ctrl key documented to select Inc Jog Optional Stop and Block Delete and Spindle CCW controls added Setup screenshots revised to reflect new and clearer information now displayed Various typos corrected Derivation of values of hotkey scan codes added Addition DXF import functionality added
Rev A2.4	8 September 2003	G61.1 removed Action of program hitting softlimits is error not clipping Use of L word for repeat count in M98 (subroutines) documented Use of output signals with PWM and Step/Dir spindles clarified Differences between M2 and M30 documented Changes to items on Configure>Logic Correction of chapter headers
Rev A1-8	22 July 2003	First complete release

# 21. 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 tried to look something up and the index entry was missing, please take a moment to e-mail <u>support@artofcnc.ca</u> 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!

#### .LSET file

layout	 15-6
.SET file	
layout	 15-6

## Α

.

Absolute distance mode
G9011-25
Absolute IJ mode 11-14
Absolute machine coordinates
G53 - move in
Acceleration
configuring 5-13
importance of configuration for accuracy 5-13
Acknowledgements1-1
ActivateSignal
subroutine
Activation outputs
assignimng to coolant and spindle control 5-8
Activation outputs - defining 5-3
Activation signals
input
interfacing 4-15
output
interfacing 4-15
Active Hi convention
Active Lo convention
Angular axes
configure
Arc - center format 11-14
Arc - radius format 11-13
Arc at feed rate
G02/G03 defined 11-13
Arc motion
defined 11-2
Arcs display or cut incorrectly
wrong IJ Mode setting 5-20
Automatic Z control
Auxiliary controls family 6-8
Axes - defining those in use 5-2
Axis controls family
described6-1
Axis coordinate DRO
described6-2
Axis drive design

design calculation	4-6
limitation by Mach2 step rate	4-7
minimum possible move	4-6
rapid speed	4-6
Axis drives	
role in system	2-2
Axis DROs	
what they show	
Axis jogging	3-5
ball	3-6, 6-5
by Manual Pulse Generator (MPG) or	encoder 6-
4	
by Manual Pulse Generatselecting axis	s to be
moved by MPG/encoder	6-4
control family	
described	6-3
Incremental selected by Ctrl key	6-4
joystick	3-6, 6-4
keyboard	3-5
Continuous	3-6, 6-4
Incremental	3-6, 6-4
rate override	3-6, 6-4
rate override by joystick throttle	
Axis slaving	
purpose and referencing	4-11
-	

## В

Back boring canned cycle	
G87	. 11-24
Backgrounds	
bitmap	8-23
Backlash	
configure	5-19
try to avoid	5-19
Backlash compensation	
switched on and off in Configure Init Stat	e. 5-21
Belt or chain pitch	5-10
Binary operations	
defined	11-8
Bitmap backgrounds	8-23
Bitmap file import	
choosing rasterising toolpath	9-6
dot diffustion rendering	9-7
for photo engraving	9-6
need to define feedrate before running	9-8
rectangular raster	9-6
selecting file to import	9-6

## С

Cancel modal motion	
G80 explained	11-20
Canned cycle return level	
G98/G99	11-27
Canned cycles	11-20
in-between motion	11-21
preliminary motion	11-21
repeats by L word	11-21
retract defined by R word	11-21
sticky numbers	11-20
Center format arc	11-14
Charge Pump Safety output - defining	5-3
Circles display or cut incorrectly	
wrong IJ Mode setting	5-20
Circular pocket	
G12/G13	11-15
Code definition syntax explained	11-11
Coding	
VB Script	8-5
Coding macros	8-5
Color scheme	
setting in ScreenDesigner	8-25
Comments	
defined	11 <b>-</b> 9
Compensation - tool diameter	
alignment move	10-3
Configure	
acceleration	5-13
Angular axes	5-21
backlash	5-19
belt or chain pitch	5-10
configure	5-22
DROs locked to initial units	5-21
encoder for jogging input	5-19
encoders	5-18
initial state	5-20
M30 - action at	5-21
Mach2 steps per revolution	5-11
Mach2 steps per unit	5-11
max motor speed	5-12
motor revs per unit	5-10
motor steps per revolution	5-11
motor tuning	5-10
optional stop	5-22
permanent DROs	5-22
program end - action at	5-21
pulse widths	5-13

referencing	. 5-18
rotary axis units	. 5-11
screw revs per unit	. 5-10
serial output	. 5-23
slaving	. 5-19
spindle speed feedback used for feeds	. 5-22
steps per unit	. 5-10
tool change action	. 5-21
Z-inhibit	. 5-21
Configure - Ports & Pins	5-1
Constant velocity mode	
G64 - setting	11-20
purpose of explained	. 11-3
Control	
alignment of	. 8-20
Bitmap	. 8-16
Bitmap button	. 8-16
bitmap buttons	. 8-22
color scheme	. 8-25
DRO	. 8-16
DRO groups	. 8-22
G-code list	8-16
Jovstick	. 8-16
Label	8-16
label - intelligent	8-22
LED	8-16
locking	8-25
MDI	8-16
nudging of	8-17
OEM codes	8-20
positioning by pixel numbers	8-24
sizing be pixel count.	8-24
sizing of	8-20
sizing to hitman	0 21
	. 0-24
spacing of	. 8-24
spacing of	. 8-24
spacing of Toolpath Controlled point	. 8-24 . 8-21 . 8-16
spacing of	. 8-24 . 8-21 . 8-16
spacing of	. 8-24 . 8-21 . 8-16 , 11-1
spacing of	. 8-24 . 8-21 . 8-16 , 11-1
spacing to ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25
spacing to ortinap spacing of	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2
spacing of ortinap spacing of	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28
spacing of	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28
spacing of	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28 11-28
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28 11-28 . 4-13
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28 11-28 11-28 . 4-13
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28 11-28 . 4-13 6-3
spacing of orinnap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28 11-28 . 4-13 6-3
spacing of orinnap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28 11-28 . 4-13 6-3 . 11-2
spacing of orinnap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-21 . 8-25 . 11-2 11-28 11-28 11-28 11-28 . 4-13 6-3 . 11-2 . 5-18
spacing of ortinap spacing of	. 8-24 . 8-21 . 8-21 . 8-25 . 11-2 11-28 11-28 11-28 . 4-13 6-3 . 11-2 . 5-18 1-1
spacing of ortinap spacing of	. 8-24 . 8-21 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28 11-28 . 4-13 6-3 . 11-2 . 5-18 1-1
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-26 . 11-2 11-28 11-28 11-28 11-28 . 4-13 6-3 . 11-2 . 5-18 1-1 . 11-3
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-26 . 11-2 11-28 11-28 11-28 11-28 . 4-13 6-3 . 11-2 . 5-18 1-1 . 11-3 . 6-14
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-16 , 11-1 . 8-25 . 11-2 11-28 11-28 11-28 . 11-2 . 4-13 6-3 11-2 5-18 1-1 11-3 6-14
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-26 . 11-2 11-28 11-28 11-28 11-28 . 11-2 . 4-13 6-3 11-2 5-18 1-1 11-3 6-14 10-1
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-26 . 11-2 11-28 11-28 11-28 11-28 . 4-13 6-3 . 11-2 . 5-18 11-3 . 6-14 . 10-1 . 10-2
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-26 . 11-2 11-28 11-28 11-28 11-28 . 11-2 6-3 11-2 5-18 1-1 11-3 6-14 10-1 10-2
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-26 . 11-2 11-28 11-28 11-28 . 11-2 . 11-28 6-3 11-2 5-18 1-1 11-3 6-14 10-1 10-2
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-25 . 11-2 11-28 11-28 11-28 11-28 . 11-2 . 4-13 6-3 11-2 5-18 11-3 6-14 . 10-1 10-4
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-25 . 11-2 11-28 11-28 11-28 11-28 . 11-2 . 12-28 . 11-2 . 12-3 . 4-13 6-3 11-2 5-18 11-1 10-1 10-4 10-3
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-25 . 11-2 11-28 11-28 11-28 11-28 . 11-2 . 11-2 6-3 11-2 5-18 1-1 11-3 6-14 10-1 10-4 10-4 10-4
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-25 . 11-2 11-28 11-28 11-28 11-28 . 11-2 . 11-2 6-3 11-2 5-18 1-1 11-3 6-14 10-1 10-4 10-4 10-2
spacing of ortinap spacing of Toolpath Controlled point defined	. 8-24 . 8-21 . 8-25 . 11-2 11-28 11-28 11-28 11-28 . 11-2 . 6-3 . 11-2 . 5-18 1-1 . 11-3 . 6-14 . 10-1 . 10-2 . 10-4 . 10-2

#### CNC Machining with Mach2Mill

Cutting time	
estimate	
Cycle Start button	6-6

## D

DeActivateSignal	
subroutine	8-12
Debounce	
configure	5-22
Developers Network	
Mach2 - link to	i
Diameter compensation LED	
described	6-3
Digital Readout	e DRO
Digitise	
defining input pin	5-4
laser trigger grid definition control family	6-14
laser trigger interface	12-1
probe interfaces	4-14
with contact probe	12-1
Direction & Step interface See Step & D	irection
Disclaimer of liability	1-1
DoButton	
subroutine	8-9
DoOEMButton	
subroutine	8-9
Downloading Mach2	3-1
Drilling canned cycle	
G81	11-21
Drilling with dwell canned cycle	
G82	. 11-22
DRO	
cancelling entry in	3-5
caution when changing axis	3-5
control on screens	3-5
entering data to	3-5
hotkeys	8-21
DROs	
user defined for script I/O	8-11
DROs locked to initial units	5-21
Dwell	11-3
G04 - defined	11-15
Dwell active LED	6-7
DXF file import	9-1
action for layers	9-2
connecting lines that nearly touch	9-3
generation and filing G-code	9-3
optimise tool movement	9-3
position of origin	9-3
z level for rapid moves	9-3
•	

## Ε

Editing G-code program	6-14
Editor program	
configure filename for	5-23
Enable outputs - defining	5-3
Encoder	
configure for jogging input	5-19
controls for	6-13
defining input pins for	5-5
input via keyboard emulator	18-1
Encoders	
configure	5-18
Engine frequency	
choice of suitable value	5-2

defining	5-2
Errors in data	
In conversational programming	8-15
reporting to user using MSG,	8-15
EStop	
lockout until reset	4-2
EStop button	
function	4-2
not involving software	4-2
Exact stop mode	
G61 - setting	11-20
purpose of explained	11-3
Excecution of words	
order of	11-31
Expressions	
defined	11-8
-	

## F

F word -feed rate	11-29
Fault finding	
port addressing and connections	5-9
Feed and speed override	
controlled by M48/M49	11-29
Feed rate	
controls family	
described	6-5
defined	. 11-2
F word to set	11-29
inverse time - G93	11-26
units per minute - G94	11-26
units per rev	6-6
units per rev - G95	11-26
Feed rate units per rev - G95	11-26
Feedrate	
display DROs	
described	6-6
File format	
layout	. 15-6
Fixture coordinate select	
G53-G59 defined	11-20
Fixture coordinate systems - setting - G10	11-15
Fixture offsets	
explained	7-2
saved in Fixture table	7-4
setting	7-3
Fixture table	
controls family	6-9
vs Tool table	7-5
Flood coolant	. 4-13
Function codes	
of screen controls	8-8
Functions – VB script functions available in	
macros	. 17-2

## G

G00 - rapid linear motion	11-11
G01 - linear feed rate move	11-13
G02 - clockwise arc	11-13
G03 - counterclockwise arc	11-13
G04 - dwell	11-15
units of P word in	5-22
G10 - set coordinate systems	11-15
G12 - circular pocket	11-15
G13 - circular pocket	11-15
G15 - exit Polar mode	11-16

G16 - enter Polar mode	11-16
GI7 - select XY plane	11-16
G18 - select XZ plane	11-16
G19 - select YZ plane	11-16
G20 - inch units - setting	11-16
G21 - millimetre units - setting	11-16
G28 - return to home	11-17
G28.1 - reference axes	11-17
G30 - return to home	11-17
G31 - straight probe	11-17
G40 - cutter radius compensation - Off	11-19
G41 - cutter radius compensation - Left	11-19
G42 - cutter radius compensation - Right	11-19
G43 - enable tool length offset	11-19
G49 - disable tool length offset	11-19
G50 - clear axis scale factors	11-19
G51 - set axis scale factors	11-19
G53 - move in absolute machine coordinates	11-19
G53 - select fixture 1	11-20
G54 - select fixture 2	11-20
G55 - select fixture 3	11-20
G56 - select fixture 4.	11-20
G57 - select fixture 5	11-20
G58 - select fixture 6	11-20
G59 - select any fixture	11_20
G61 - set exact stop mode	11_20
G64 - set constant velocity mode	11-20
G80 - cancel modal motion	11-20
G81 - drilling canned cycle	11-20
G82 drilling with dwell canned cycle	11 21
G82 - unning with dwell called cycle	11-22
G84 tanning canned cycle	11-22
064 - tapping canned cycle	11-23
C05 having and reaming sonned avala	11 22
G85 - boring and reaming canned cycle	11-23
G85 - boring and reaming canned cycle G86 - boring with dwell canned cycle	11-23 11-23
G85 - boring and reaming canned cycle G86 - boring with dwell canned cycle G87 - back boring canned cycle	11-23 11-23 11-24
<ul> <li>G85 - boring and reaming canned cycle</li> <li>G86 - boring with dwell canned cycle</li> <li>G87 - back boring canned cycle</li> <li>G88 - boring manual retract canned cycle</li> </ul>	11-23 11-23 11-24 11-25
<ul> <li>G85 - boring and reaming canned cycle</li> <li>G86 - boring with dwell canned cycle</li> <li>G87 - back boring canned cycle</li> <li>G88 - boring manual retract canned cycle</li> <li>G89 - boring with dwell and retract canned cy</li> </ul>	11-23 11-23 11-24 11-25 cle
<ul> <li>G85 - boring and reaming canned cycle</li></ul>	11-23 11-23 11-24 11-25 cle 11-25
<ul> <li>G85 - boring and reaming canned cycle</li></ul>	11-23 11-23 11-24 11-25 cle 11-25 11-25
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cy         G90 - absolute distance mode         G91 - incremental distance mode	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 6-3 11-25
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         G92 - workpiece offsets	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 6-3 11-25 11-26
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         See Workpiece offsets	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         G93 - feed rate inverse time	11-23 11-23 11-24 11-25 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         See Workpiece of         G93 - feed rate inverse time         G94 - feed rate units per minute	11-23 11-23 11-24 11-25 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 11-26 11-26
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26 11-27
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         S93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word	11-23 11-23 11-24 11-25 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26 11-27 11-27 6-8
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26 11-27 11-27 6-8
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         S93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code execution         in script	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26 11-27 11-27 6-8 8-8
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         See Workpiece of         G93 - feed rate inverse time         G94 - feed rate units per minute         G95 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26 11-27 11-27 11-27 6-8 8-8
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing	11-23 11-23 11-24 11-25 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26 11-27 11-27 11-27 11-27 6-8 8-8 8-8
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         See Workpiece of         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing         inputting	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26 11-27 11-27 11-27 6-8 8-8 8-8
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         See Workpiece of         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing         inputting         loading	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-26 11-27 11-27 6-8 8-8 8-8 8-8
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         See Workpiece of         G93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing         inputting         loading         running	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 11-26 11-26 11-27 11-27 11-27 6-8 8-8 6-14 6-15 6-14 6-15
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         S92 offsets         S93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing         inputting         loading         running	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 11-26 11-26 11-27 11-27 11-27 6-8 8-8 6-14 6-15 6-14 6-15
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G87 - back boring manual retract canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         S93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing         inputting         loading         running         G-code window         control on screens	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 11-26 11-26 11-27 11-27 11-27 6-8 8-8 6-14 6-15 6-14 3-5
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G87 - back boring manual retract canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         S94 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing         inputting         loading         running         G-code window         control on screens         G-codes	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-27 11-27 11-27 6-8 8-8 6-14 6-15 6-14 6-15 3-5
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         S93 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing         inputting         loading         running         G-code window         control on screens         G-codes         summary table	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-27 11-27 11-27 11-27 6-8 8-8 6-14 6-15 6-14 6-15 3-5
G85 - boring and reaming canned cycle         G86 - boring with dwell canned cycle         G87 - back boring canned cycle         G88 - boring manual retract canned cycle         G89 - boring with dwell and retract canned cycle         G90 - absolute distance mode         G91 - incremental distance mode         G92         button         described         G92 - workpiece offsets         interaction with parameters         G92 offsets         S94 - feed rate inverse time         G94 - feed rate units per minute         G98 - canned cycle return level         to old Z         G99 - canned cycle return level         to R word         G-code display control         G-code program         editing         inputting         loading         running         G-code window         control on screens         G-codes         summary table         G-codedisplay	11-23 11-23 11-24 11-25 cle 11-25 11-25 11-25 11-25 11-25 11-26 ffsets 11-26 11-27 11-27 11-27 11-27 6-8 8-8 6-14 6-15 6-14 6-15 3-5

GetDRO
function
GetLED
function
GetOEMDRO
function
GetOEMLED
function
GetParam
function
GetPortByte
function
GetVar
function
GetXCoor
function (also for Y, Z, A, B, C) 8-11
Greved out text - meaning 1-1
Ground
signal 4-4

## Η

Hardware single step button	6-12
HID	
as source of keycodes	8-3
Home	
using location in practical machine	
Home - return to G28/G30	11-17
Home switch See also Lim	it switches
not near axis limit:	4-11
purpose	4-8
Hotkey codes	
how calculated	8-17
Hotkeys	
DRO	8-21
Hotkeys - global	8-25
HPGL file import	
choosing scale	
limitations	
production and filing of G-code	
Human Interface Device	See HID

## I

lcons	
on desktop shortcuts for Mach2	18-2
Ignore Tool Change switch	6-8
IJ Mode	
defines how G02/G03 are interpreted	5-20
IJ mode - "Absolute"	11-14
IJ mode - Increments	11-14
Import	
DXF file	
Importing foreign data files	6-15
Inch units	
G20 - setting	11-16
Incremental distance mode	
G91	11-25
Incremental IJ mode	11-14
Index	
defining pin for pulse	5-4
interface for spindle	4-15
Initial state	
configure	5-20
Input pins	
shortage of	5-5
Inputting G-code program	6-15
-	

J
Jerky motion with short lines
Constant velofity mode to avoid 11-3
Jog follow button
Jog increment
setting
Jogging
JPEG file import See Bitmap file import
JPG file import See Bitmap file import

# Κ

Keyboard
shortcuts
Keyboard emulator
setup for generating signals 5-6
Keyboard emulators
uses for in controlling Mach218-1
Keyboard shortcut - Screen Designer 14-1
Keycodes
processing of
Keygrabber
configuring for keyboard or keyboard emulator
keys 18-3
KeyGrabber
analog axis mapping to keycodes 18-7
configuring encoder inputs 18-5
configuring HID inputs18-6
files used18-2
generation of keycodes by 8-3
HID Page selection 18-6
installation of 18-2
overview
profiles (.GRAB files) 18-8
sending modifer keys 18-4
special keys18-4
troubleshooting
Keystrokes
actions of
and their shortcuts

## L

Label	
Intelligent, Ticker formatted	8-22
Intelligent, User	
Lathe operation using Mach2	1-1
LayoutSET file	8-16
LED	

· 1	2.5
control on screens	3-5
LEDS	0 1 1
usei defined	0-11
License statement	I-I
Light Emitting Diode	e LED
	5 1
defining	5-4
Limit switches	3-3
auto and manual override controls	6-12
cabling	4-9
defining override switch input	5-5
external implementation	4-8
magnetic	
applications for	4-9
manual override	4-10
microswitches	
accuracy of	4-9
overtravel	4-9
mounting	4-9
OR for electronic switches	4-9
purpose	4-8
sharing Mach2 inputs	4-10
sharing Mach2 inputs for	4-8
Limits	
soft - enabling and disabling	6-12
Limits - soft	5-20
Limits and miscellaneous control family des	cribed
-	6-12
Line	
format of code	11-7
Line number	
format of	11-7
Linear axes	
defined	11-1
Linear feed rate move	
G01 defined	11-13
Linear glass scale	
not part of servo loop	
quadrature interface	4-14
Loading G-code program	6-14
Locking	
controls	8-25

## М

M00 program stop	11 27
	11-27
M01 - optional program stop	11-27
M02 - program end	11-27
M03 - spindle clockwise	11-28
M04 - spindle counterclockwise	11-28
M05 - stop spindle	11-28
M07 - mist coolant on	11-28
M08 - flood coolant on	11-28
M09 - all coolant off	11-28
M30 - action at	
configure	5-21
M30 - program end	11-27
M48 - feed and speed override on	11-29
M49 - feed and speed override off	11-29
M98 - subroutine call	11-29
M99 - subroutine return	11-29
Mach Developers NetworkDN	
link to	i
Mach2	
charge pump monitorSee Cha	rge pump
Components of	3-1
computer requirements	2-2

how distributed
on laptop
pulse monitor
what macines it can control
Mach2 scancodes
how calculated
Mach2ScreenTweak
MachDN
developers network link i
Machine controller
role in system 2-2
Machine coordinates
diamlacing on axis DBOs
C52 manual in
G53 - move in
Machine modes
defined 11-10
MacroSee alse VB Script
naming and calling8-4
simple example code
Macro M-codes
Macros
coding 8-5
detailed description of an example 8-6
generating g-code sequences within 8-7
interaction with machine operator 8.7
interaction with machine operator
overview on writing 11-29
passing parameters to
Manual Data Input See MDI, See MDI
Manual Pulse Generator
Maximum spindle speed 5-16
M-code
macros 11-29
M-codes - built in
M-codes - built in summary table 11-27
M-codes - built in summary table 11-27 MDI
M-codes - built in summary table 11-27 MDI control on screens
M-codes - built in summary table

string introduces	an operator message	11-9
-------------------	---------------------	------

## Ν

NC machine	
parts of	
Noise	
on signal ground	4-4
Nudging	
controls into position	8-17
Number	
format of	11-7

## 0

OEM codes	
of screen controls	
OEM signal inputs	4-15
Offset	
fixture	See Fixture offsets
tool	See Tool offsets
Operators - binary	
defined	
Operators - unary	
defined	11-9
Optional program stop	
M01	11-27
Optional stop	
configure	5-22
Order of items on line	11-10
Override	
for feed and speed - disabling	
Override feed and speed	
controlled by M48/M49	11-29

## Ρ

Parallel port	
D25 connector pinout	4-3
general background	4-3
Param	
functions	8-11
Parameter	
setting value of	11-9
using value of	11-8
Parameters	
predefined	11-4
Part program	
editing	6-14
inputting	6-15
loading	6-14
repeating indefinitely - M47	11-28
repeating indefinitely -M99	11-29
running	6-15
running controls family	
described	6-6
Part Program	
running a sample	3-7
Pause button	6-6
PC	
configuration required	2-2
Peck drilling canned cycle	
G83	11-22
Permanent DROs	
configure	5-22
Persistent screen - controls on	8-16
Plane selection	
G17/G18/G19 defined	11-16

avoiding distortion
calibrate to zero
Correction speed 6-11
defining input signals
DXF converted for use with
Enable toggle
experience needed to build 4-13
height control
height servo
Highest and Lowest correction limits
need for systematic grounding
running economically
sequence of operations
spring mounting with reference switch
support of by Macriz
torch up signal 5.7
Plasma torch controller
acknowledgement 16-4
interface circuits license 16-4
interface functional blocks 16-4
Plasma torch controller interface
disclaimer 16-4
Plasma torch controls family 6-10
Polar mode 11-16
Port addresses - finding with Windows Control
Panel
Ports
foreign - access to
Probe
Probe - design requirements
Profile
copying and viewing
······································
display name of profile in use

PWM speed control	See also	Spindle
-------------------	----------	---------

## Q

Quadrature encoder	
as MPG	See MPG
Question	
function	

# R

Radius format arc 11-13
Rapid motion
G00 definrd 11-11
Reaming and boring canned cycle
G85
Re-boot during installation
how to manually uninstall driver if you fail to do
it
reason for
Recording your configuration
Reference - G28.1
Reference switch
defining
location controls
Referenced LED
described
Referencing
configure
details of Mach2 actions
Regen button
Relay activation outputs See Activation Outputs
Repeating part program indefinitely - M47 11-28
Repeating part program indefinitely - M99 11-29
Reset button
described
Retrofitting old CNC machines
caution
Return level after canned cycle
G98/G9911-27
Rotational axes
defined
Rotational diameter correction
controls family 6-10
Roughing
automatic using Inhibit-Z 6-13
taxhnique for use with DVF and HDGL import
texinique for use with DAT and Th OL import
files
files
files

# S

S word - spindle speed 11-30
Safe Z
control 6-13
Safety warning1-1, 4-1
professional advice 1-1, 4-1
Scale factor - on axis data - G50, G51 11-19
Scale factor DRO
described
Scaling coordinates
Scaling parts 11-1
Scancodes
how calculated
Screen
LED - example of

Screen captions
Screen controls
Screen Designer
standard shortcuts 14-1
Screen Designer - explained
Screen Designer Controls See Controls - Screen
Screen layouts
sample
Screen switching buttons
Screen switching controls
described
ScreenTweak
Additional layout 15-4
appending screen to Principal layout
delete screen
functions of 15-1
installation
move screen up or down list 15-4
Principal layout
promote and demote screen
re-ordering DRO in groups
Sscreen captions 15-6
Screw revs per unit
ScripSetUserLabel subroutine
Script
avoid infinite oops in
debugging
$0 \ge 0 \le 0 \le 10 \le 0 $
stages of execution of
stages of execution of8-14Script function8-9GetDRO8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12
stages of execution of8-14Script function8-9GetDRO8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Ouestion8-11
stages of execution of8-14Script functionGetDROGetDRO8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-11Script subroutine
stages of execution of8-14Script function6etDROGetDRO8-9GetCEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12Script subroutine8-12
stages of execution of8-14Script function8-9GetDRO8-9GetLED8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DeActivateSignal8-12
stages of execution of8-14Script functionGetDROGetDRO8-9GetOEMDRO8-9GetOEMLED8-9GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9DoButton8-9
stages of execution of8-14Script functionGetDROGetDRO8-9GetCEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9DoOEMButton8-9CatCoard8-11
stages of execution of8-14Script function8-9GetDRO8-9GetLED8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-11GetVar8-12
stages of execution of8-14Script functionGetDROGetDRO8-9GetCEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9DoOEMButton8-9GetCoord8-11GetVarSetVar8-10
stages of execution of8-14Script function6etDROGetDRO8-9GetCEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-12DoSutton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12SetDRO8-9
stages of execution of8-14Script function6etDROGetDRO8-9GetLED8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-12DoButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12SetDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9
stages of execution of8-14Script functionGetDROGetDRO8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12DoCEMButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12SetDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetParam8-10SetParam8-10SetParam8-10SetParam8-10SetParam8-10SetParam8-10SetParat8-10SetParat8-10SetParat8-10SetParat8-10SetParat8-10SetParat8-10
stages of execution of8-14Script function6etDROGetDRO8-9GetCEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12DoCEMButton8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-13SetTicker8-13SetTicker8-13
stages of execution of8-14Script function6etDROGetDRO8-9GetLED8-9GetOEMDRO8-9GetOEMLED8-9GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-12DoButton8-9SendSerial8-12SetDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-13SetTicker8-12SetUserLabel8-12
stages of execution of8-14Script function8-9GetDRO8-9GetLED8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12DoEMButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12SetDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-13SetTicker8-13SetTicker8-12SetUserLabel8-12Secondhand equipment8-12
stages of execution of
stages of execution of8-14Script functionGetDROGetDRO8-9GetLED8-9GetOEMDRO8-9GetOEMLED8-9GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-12DoButton8-9SendSerial8-12SetDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-13SetTicker8-13SetTicker8-12SetOentByte8-13SetTicker8-12SetUserLabel8-12Secondhand equipmenta cautiona caution4-5Selected plane44-5
stages of execution of8-14Script functionGetDRO8-9GetLED8-9GetOEMDRO8-9GetOEMLED8-9GetParam8-10GetPortByte8-13GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-12DoButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12SetDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-9SetOEMDRO8-13SetTicker8-10SetOEMDRO8-13SetTicker8-12SetOend8-13SetTicker8-12SetOend8-13SetTicker8-12SetOend8-13SetTicker8-12SetOend8-13SetTicker8-12SetOend8-13SetTicker8-12SetOend8-13SetTicker8-12SetOend8-13SetTicker8-12SetOend8-13SetTicker8-13SetTicker8-13SetTicker8-13SetTicker8-13SetTicker8-13SetTicker8-13SetTicker8-13SetTicker8-13SetO
stages of execution of8-14Script function8-9GetDRO8-9GetLED8-9GetOEMDRO8-9GetOEMLED8-9GetPortByte8-13GetVar8-10GetVar8-10IsActive8-12Param1, Param2, Param38-11Question8-12DeActivateSignal8-12DoButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12DoEMButton8-9GetCoord8-11GetVarSetVar8-10SendSerial8-12SetOEMDRO8-9SetOEMDRO8-9SetOemdrom8-13SetTicker8-10SetPortByte8-13SetTicker8-12SetOend8-13SetTicker8-12SetOend8-12SetOend8-13SetTicker8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12SetOend8-12

<b>c</b> oning <b>ai c</b>
Servo motor drives
properties
SET file - contains screen layouts
Set fixture coordinate systems - G10 11-15
Set next line button
SetDRO
subroutine
SetOEMDRO
subroutine 8-9
SetParam
subroutine 8-10
SetPortByte
subroutine 8.12
Subloutine
Set i lekel subioutile
setup units
choosing between inch and millimetres
SetVar
subroutine
Shortcut - Keyboard in Screen Designer 14-1
Shortcut keys
Shortcuts See Keyboard shortcuts
Signal
ground 4-4
Signals
in Mach2 communication paths
Single button
Slave axis
Slaving
configure 5-19
Soft limits See Limits
Special Mach? sys driver
installation of 3-3
need for 2.2
Speed and feed override
speed and reed override
11 20
controlled by M48/M49 11-29
controlled by M48/M49 11-29 Spindle
controlled by M48/M49 11-29 Spindle M03 - clockwise 11-28
controlled by M48/M49
controlled by M48/M49.11-29SpindleM03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5maximum, defined for pulleys5-16S word to set11-30Spindle speed feedback used for feeds5-22Step & Direction5-22Step & Direction4-12Step & Direction4-12Step & Direction4-7waveforms4-7
controlled by M48/M49.11-29SpindleM03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5maximum, defined for pulleys5-16minimum, defined for pulley5-16S word to set11-30Spindle speed feedback used for feeds5-22Step & Direction5-22Step & Direction interface4-12Caution about active hi/lo4-7waveforms4-7Stepper motor drive4-7
controlled by M48/M49.11-29SpindleM03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5control family described6-5maximum, defined for pulleys5-16S word to set11-30Spindle speed feedback used for feeds5-22Step & Direction5-22Step & Direction4-12Step & Direction interface4-7waveforms4-7Stepper motor drive4-5
controlled by M48/M49.11-29SpindleM03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5control family described6-5maximum, defined for pulleys5-16minimum, defined for pulley5-16Sword to set11-30Spindle speed feedback used for feeds5-22configure5-22Step & Direction4-12Step & Direction interface4-7waveforms4-7Stepper motor drive4-5Stepper motor drives4-5
controlled by M48/M49.11-29Spindle11-28M03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5control family described6-5maximum, defined for pulleys5-16minimum, defined for pulley5-16Sword to set11-30Spindle speed feedback used for feeds5-22configure5-22Step & Direction4-12Step & Direction interface4-7waveforms4-7Stepper motor drive4-5Stepper motor drives4-5Imit of canability4-5
controlled by M48/M49.11-29SpindleM03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5control family described6-5maximum, defined for pulleys5-16minimum, defined for pulley5-16Sword to set11-30Spindle speed feedback used for feeds5-22configure5-22Step & Direction4-12Step & Direction interface4-7waveforms4-7Stepper motor drive4-5Stepper motor drives4-5Iimit of capability4-5
controlled by M48/M49.11-29SpindleM03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5control family described6-5maximum, defined for pulleys5-16minimum, defined for pulley5-16Sword to set11-30Spindle speed feedback used for feeds5-22configure5-22Step & Direction4-12Step & Direction interface4-12caution about active hi/lo4-7waveforms4-7Stepper motor drive4-5properties4-5Step per unit5-10
controlled by M48/M49.11-29Spindle11-28M03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5control family described6-5maximum, defined for pulleys5-16S word to set11-30Spindle speed feedback used for feeds5-22configure5-22Step & Directionspindle drivespindle drive4-12Step & Direction interface4-7waveforms4-7Stepper motor drive4-7properties4-5Stepper motor drives111-30Stepper motor drives5-10Stop button6-6
controlled by M48/M49.11-29Spindle11-28M03 - clockwise11-28M04 - counterclockwise11-28M05 - stop11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5control family described6-5maximum, defined for pulleys5-16minimum, defined for pulleys5-16S word to set11-30Spindle speed feedback used for feeds5-22configure5-22Step & Direction4-12Step & Direction interface4-7waveforms4-7Stepper motor drive4-7properties4-5Stepper motor drives11-10limit of capability4-5Steps per unit5-10Stop button6-6Straight probe414.10
controlled by M48/M49.11-29Spindle11-28M03 - clockwise11-28M04 - counterclockwise11-28PWM speed control5-17Step & Direction speed control5-17Spindle control5-17clockwise/counterclockwise4-11interlocking requirements4-12Spindle speed6-5control family described6-5maximum, defined for pulleys5-16minimum, defined for pulleys5-16S word to set11-30Spindle speed feedback used for feeds5-22configure5-22Step & Direction4-7spindle drive4-12Step & Direction interface4-7caution about active hi/lo4-7waveforms4-7Stepper motor drive4-5Stepper motor drives11-18limit of capability4-5Steps per unit5-10Stop button6-6Straight probe6-9example program11-18

#### CNC Machining with Mach2Mill

11-29
11-29
11-7
11-29
11-11

# T

T word - tool select	11-30
Tapping canned cycle	
G84	11-23
Testing	
axis calibration	5-14
configuration settings	5-9
for lost steps	5-15
Mach2 installation	
OCXDiverTest	
spindle drive	5-17
Ticker	
User defined	8-22
Tool change	
supplied M6 macros	11-28
Tool change action	
configure	5-21
Tool length	
allowing for by offsets	7-4
Tool length offset	
G43 - enable	11-19
G49 - disable	11-19
Tool offsets	
reason for	7-4
Tool select	
T word	
Tool table	
controls family	6-9
saving offsets in	7-5
vs Fixture table	7-5
Toolpath display	
colors to be used for	5-21
control on screens	
panning	
rotating	
zooming	
Tools	
presettable	7-5
Tooltip radius	
compensation overview	
Touching	
role of edge finder in	7-7

role of slip gage in	
Trademarks	
Transparent bitmap buttons	8-22
TTL	
current sourcing and sinking	4-3
signal levels	4-3
Turning using Mach2	1-1

## U

## V

VB Script	
accessing screen controls from	8-8
coding	8-5
confusion with brackets in calls	8-7
example code	8-5
executing G-code from	8-8
on buttons	8-18
ways to use	8-4
VB Script subroutine	
Code	

## W

Waiting for Mach2	8-13
Word	
format of	
intial letters	11-7
Workpiece offsets	
G92	
692	11-25

## Y

yGrabber	
Typematic settings	-7

# Ζ

Z-inhibit	
configure	 5-21