

User Manuelle V2.5, 2014-04-17

Contents

I	LinuxCNC Introduction	1
1	User Foreword	3
2	LinuxCNC User Introduction	5
2.1	This Manual	5
2.2	How LinuxCNC Works	5
2.3	Graphical User Interfaces	6
2.4	Virtual Control Panels	12
2.5	Languages	14
2.6	Thinking Like a Machine Operator	14
2.7	Modes of Operation	14
3	Important User Concepts	16
3.1	Trajectory Control	16
3.1.1	Trajectory Planning	16
3.1.2	Path Following	16
3.1.3	Programming the Planner	16
3.1.4	Planning Moves	17
3.2	G Code	18
3.2.1	Defaults	18
3.2.2	Feed Rate	18
3.2.3	Tool Radius Offset	18
3.3	Homing	18
3.4	Tool Changes	18
3.5	Coordinate Systems	19
3.5.1	G53 Machine Coordinate	19
3.5.2	G54-59.3 User Coordinates	19
3.5.3	When You're Lost	19
3.6	Machine Configurations	19

II	User Interfaces	22
4	AXIS GUI	23
4.1	Introduction	23
4.2	Getting Started	24
4.2.1	A Typical Session	24
4.3	AXIS Display	25
4.3.1	Menu Items	25
4.3.2	Toolbar buttons	28
4.3.3	Graphical Display Area	29
4.3.4	Text Display Area	31
4.3.5	Manual Control	31
4.3.6	MDI	33
4.3.7	Feed Override	33
4.3.8	Spindle Speed Override	33
4.3.9	Jog Speed	34
4.3.10	Max Velocity	34
4.4	Keyboard Controls	34
4.5	Show LinuxCNC Status (linuxcnc_top)	35
4.6	MDI interface	35
4.7	axis-remote	36
4.8	Manual Tool Change	36
4.9	Python modules	36
4.10	Using AXIS in Lathe Mode	37
4.11	Advanced Configuration	37
4.11.1	Program Filters	37
4.11.2	The X Resource Database	38
4.11.3	Physical jog wheels	39
4.11.4	~/axisrc	39
4.11.5	External Editor	39
4.11.6	Virtual Control Panel	39
4.11.7	Axis Preview Control	39
5	NGCGUI	41
5.1	Overview	42
5.2	Demo Configs	42
5.3	Libraries	43
5.4	Embedding NGCGUI in Axis	43
5.4.1	INI File	44
5.4.2	Truetype Tracer	45
5.4.3	INI Examples	45
5.5	Subroutine Requirements	48
5.6	DB25 Example	49

6	Touchy GUI	53
6.1	Panel Configuration	54
6.1.1	HAL connections	54
6.1.1.1	Required controls	54
6.1.1.2	Optional controls	54
6.1.1.3	Optional panel lamps	54
6.1.2	Recommended for any setup	54
6.2	Setup	54
6.2.1	Enabling Touchy	54
6.2.2	Preferences	55
6.2.3	Macros	55
7	TkLinuxCNC GUI	56
7.1	Introduction	56
7.2	Getting Started	57
7.2.1	A typical session with TkLinuxCNC	57
7.3	Elements of the TkLinuxCNC window	57
7.3.1	Main buttons	58
7.3.2	Offset display status bar	58
7.3.3	Coordinate Display Area	58
7.3.3.1	Backplot	58
7.3.4	Automatic control	59
7.3.4.1	Buttons for control	59
7.3.4.2	Text Program Display Area	59
7.3.5	Manual Control	59
7.3.5.1	Implicit keys	59
7.3.5.2	The Spindle group	60
7.3.5.3	The Coolant group	60
7.3.6	Code Entry	60
7.3.6.1	MDI:	60
7.3.6.2	Active G-Codes	60
7.3.7	Jog Speed	61
7.3.8	Feed Override	61
7.3.9	Spindle speed Override	61
7.4	Keyboard Controls	61

8	MINI GUI	62
8.1	Introduction	62
8.2	Screen layout	63
8.3	Menu Bar	64
8.4	Control Button Bar	65
8.4.1	MANUAL	65
8.4.2	AUTO	66
8.4.3	MDI	67
8.4.4	[FEEDHOLD]—[CONTINUE]	67
8.4.5	[ABORT]	67
8.4.6	[ESTOP]	68
8.5	Left Column	68
8.5.1	Axis Position Displays	68
8.5.2	Feed rate Override	69
8.5.3	Messages	69
8.6	Right Column	69
8.6.1	Program Editor	70
8.6.2	Backplot Display	71
8.6.3	Tool Page	71
8.6.4	Offset Page	72
8.7	Keyboard Bindings	73
8.7.1	Common Keys	73
8.7.2	Manual Mode	74
8.7.3	Auto Mode	75
8.8	Misc	75
9	KEYSTICK GUI	76
9.1	Introduction	76
9.2	Installing	77
9.3	Using	77
III	Using LinuxCNC	78
10	CNC Machine Overview	79
10.1	Mechanical Components	79
10.1.1	Axes	79
10.1.2	Spindle	79
10.1.3	Coolant	79
10.1.4	Feed and Speed Override	80

10.1.5	Block Delete Switch	80
10.1.6	Optional Program Stop Switch	80
10.2	Control and Data Components	80
10.2.1	Linear Axes	80
10.2.2	Rotational Axes	80
10.2.3	Controlled Point	80
10.2.4	Coordinated Linear Motion	81
10.2.5	Feed Rate	81
10.2.6	Coolant	81
10.2.7	Dwell	81
10.2.8	Units	81
10.2.9	Current Position	81
10.2.10	Selected Plane	82
10.2.11	Tool Carousel	82
10.2.12	Tool Change	82
10.2.13	Pallet Shuttle	82
10.2.14	Path Control Mode	82
10.3	Interpreter Interaction with Switches	82
10.3.1	Feed and Speed Override Switches	82
10.3.2	Block Delete Switch	82
10.3.3	Optional Program Stop Switch	82
10.4	Tool Table	83
10.5	Parameters	83
11	Coordinate System	84
11.1	Introduction	84
11.2	The Machine Position Command (G53)	84
11.3	Fixture Offsets (G54-G59.3)	85
11.3.1	Default coordinate system	86
11.3.2	Setting coordinate (fixture) offsets from G code	86
11.4	G92 Offsets	86
11.4.1	The G92 commands	86
11.4.2	Setting G92 values	87
11.4.3	G92 Cautions	87
11.5	Sample Program Using Offsets	88

12 Tool Compensation	90
12.1 Tool Length Offsets	90
12.1.1 Touch Off	90
12.1.2 Using G10 L1/L10/L11	91
12.2 Tool Table	91
12.2.1 Tool Table Format	91
12.2.2 Tool Changers	92
12.3 Cutter Compensation	93
12.3.1 Overview	94
12.3.2 Examples	95
13 G Code Overview	97
13.1 Overview	97
13.2 Format of a line	97
13.3 Block Delete	98
13.4 Line Number	98
13.5 Word	98
13.6 Number	99
13.7 Parameters	99
13.7.1 Numbered Parameters	100
13.7.2 Subroutine Parameters	101
13.7.3 Named Parameters	101
13.7.4 System Parameters	102
13.8 Expressions	102
13.9 Binary Operators	102
13.10 Functions	102
13.11 Repeated Items	103
13.12 Item order	103
13.13 Commands and Machine Modes	104
13.14 Polar Coordinates	104
13.15 Modal Groups	106
13.16 Comments	107
13.17 Messages	108
13.18 Probe Logging	108
13.19 Logging	108
13.20 Debug Messages	108
13.21 Print Messages	108
13.22 Comment Parameters	108
13.23 File Requirements	109

13.24	File Size	109
13.25	G Code Order of Execution	109
13.26	G Code Best Practices	110
13.26.1	Use an appropriate decimal precision	110
13.26.2	Use consistent white space	110
13.26.3	Use Center-format arcs	110
13.26.4	Put important modal settings at the top of the file	110
13.26.5	Don't put too many things on one line	110
13.26.6	Don't set & use a parameter on the same line	110
13.26.7	Don't use line numbers	111
13.27	Linear and Rotary Axis	111
13.28	Common Error Messages	111
14	G Codes	112
14.1	Conventions	112
14.2	G Code Quick Reference Table	112
14.3	G0 Rapid Move	113
14.3.1	Rapid Velocity Rate	114
14.4	G1 Linear Move	114
14.5	G2, G3 Arc Move	115
14.5.1	Center Format Arcs	115
14.5.2	Center Format Examples	117
14.5.3	Radius Format Arcs	119
14.6	G4 Dwell	119
14.7	G5.1 Quadratic B-spline	120
14.8	G5.2 G5.3 NURBs Block	120
14.9	G7 Lathe Diameter Mode	121
14.10	G8 Lathe Radius Mode	121
14.11	G10 L1 Set Tool Table	122
14.12	G10 L2 Set Coordinate System	122
14.13	G10 L10 Set Tool Table	124
14.14	G10 L11 Set Tool Table	124
14.15	G10 L20 Set Coordinate System	125
14.16	G17 - G19.1 Plane Selection	125
14.17	G20, G21 Units	125
14.18	G28, G28.1 Go to Predefined Position	126
14.19	G30, G30.1 Go to Predefined Position	126
14.20	G33 Spindle Synchronized Motion	127
14.21	G33.1 Rigid Tapping	127

14.22G38.x Straight Probe	128
14.23G40 Compensation Off	129
14.24G41, G42 Cutter Compensation	130
14.25G41.1, G42.1 Dynamic Cutter Compensation	130
14.26G43 Tool Length Offset	131
14.27G43.1: Dynamic Tool Length Offset	131
14.28G49: Cancel Tool Length Compensation	132
14.29G53 Move in Machine Coordinates	132
14.30G54-G59.3 Select Coordinate System	132
14.31G61, G61.1 Exact Path Mode	133
14.32G64 Path Blending	133
14.33G73 Drilling Cycle with Chip Breaking	134
14.34G76 Threading Cycle	134
14.35Canned Cycles	137
14.35.1 Common Words	137
14.35.2 Sticky Words	137
14.35.3 Repeat Cycle	137
14.35.4 Retract Mode	138
14.35.5 Canned Cycle Errors	138
14.35.6 Preliminary and In-Between Motion	138
14.35.7 Why use a canned cycle?	139
14.36G80 Cancel Canned Cycle	140
14.37G81 Drilling Cycle	141
14.38G82 Drilling Cycle, Dwell	144
14.39G83 Peck Drilling Cycle	145
14.40G84 Right-Hand Tapping Cycle	145
14.41G85 Boring Cycle, Feed Out	145
14.42G86 Boring Cycle, Spindle Stop, Rapid Move Out	146
14.43G87 Back Boring Cycle	146
14.44G88 Boring Cycle, Spindle Stop, Manual Out	146
14.45G89 Boring Cycle, Dwell, Feed Out	146
14.46G90, G91 Distance Mode	146
14.47G90.1, G91.1 Arc Distance Mode	147
14.48G92 Coordinate System Offset	147
14.49G92.1, G92.2 Reset Coordinate System Offsets	147
14.50G92.3 Restore Axis Offsets	148
14.51G93, G94, G95: Feed Rate Mode	148
14.52G96, G97 Spindle Control Mode	148
14.53G98, G99 Canned Cycle Return Level	149

15 G Code Examples	150
15.1 Mill Examples	150
15.1.1 Helical Hole Milling	150
15.1.2 Slotting	150
15.1.3 Grid Probe	150
15.1.4 Smart Probe	150
15.1.5 Tool Length Probe	151
15.1.6 Hole Probe	151
15.1.7 Cutter Compensation	151
15.2 Lathe Examples	151
15.2.1 Threading	151
16 Lathe User Information	152
16.1 Lathe Mode	152
16.2 Lathe Tool Table	152
16.3 Lathe Tool Orientation	152
16.4 Tool Touch Off	156
16.4.1 X Touch Off	156
16.4.2 Z Touch Off	156
16.4.3 The Z Machine Offset	157
16.5 Spindle Synchronized Motion	157
16.6 Arcs	157
16.6.1 Arcs and Lathe Design	158
16.6.2 Radius & Diameter Mode	158
16.7 Tool Path	158
16.7.1 Control Point	158
16.7.2 Cutting Angles without Cutter Comp	159
16.7.3 Cutting a Radius	160
16.7.4 Using Cutter Comp	162
17 RS274/NGC Differences	163
17.1 Changes from RS274/NGC	163
17.2 Additions to RS274/NGC	163
18 M Codes	165
18.1 M Code Quick Reference Table	165
18.2 M0, M1 Program Pause	165
18.3 M2, M30 Program End	165
18.4 M60 Pallet Change Pause	166
18.5 M3, M4, M5 Spindle Control	166

18.6 M6 Tool Change	166
18.6.1 Manual Tool Change	166
18.6.2 Tool Changer	167
18.7 M7, M8, M9 Coolant Control	167
18.8 M48, M49 Speed and Feed Override Control	167
18.9 M50 Feed Override Control	167
18.10M51 Spindle Speed Override Control	168
18.11M52 Adaptive Feed Control	168
18.12M53 Feed Stop Control	168
18.13M61 Set Current Tool Number	168
18.14M62 to M65 Output Control	168
18.15M66 Wait on Input	169
18.16M67 Synchronized Analog Output	170
18.17M68 Analog Output	170
18.18M100 to M199 User Defined Commands	170
19 O Codes	173
19.1 Subroutines	173
19.2 Looping	174
19.3 Conditional	175
19.4 Repeat	175
19.5 Indirection	176
19.6 Calling Files	176
20 Other Codes	177
20.1 F: Set Feed Rate	177
20.2 S: Set Spindle Speed	177
20.3 T: Select Tool	177
21 Image to G Code	179
21.1 What is a depth map?	179
21.2 Integrating image-to-gcode with the AXIS user interface	179
21.3 Using image-to-gcode	180
21.4 Option Reference	180
21.4.1 Units	180
21.4.2 Invert Image	180
21.4.3 Normalize Image	180
21.4.4 Expand Image Border	180
21.4.5 Tolerance (units)	180
21.4.6 Pixel Size (units)	180

21.4.7 Plunge Feed Rate (units per minute)	180
21.4.8 Feed Rate (units per minute)	181
21.4.9 Spindle Speed (RPM)	181
21.4.10 Scan Pattern	181
21.4.11 Scan Direction	181
21.4.12 Depth (units)	181
21.4.13 Step Over (pixels)	181
21.4.14 Tool Diameter	181
21.4.15 Safety Height	181
21.4.16 Tool Type	182
21.4.17 Lace bounding	182
21.4.18 Contact angle	182
21.4.19 Roughing offset and depth per pass	182
22 Glossary	184
23 Legal Section	189
23.1 Copyright Terms	189
23.2 GNU Free Documentation License	189
24 Index	193

The LinuxCNC Team

Part I

LinuxCNC Introduction



This handbook is a work in progress. If you are able to help with writing, editing, or graphic preparation please contact any member of the writing team or join and send an email to emc-users@lists.sourceforge.net.

Copyright © 2000-2012 LinuxCNC.org

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and one Back-Cover Text: This LinuxCNC Handbook is the product of several authors writing for linuxCNC.org. As you find it to be of value in your work, we invite you to contribute to its revision and growth. A copy of the license is included in the section entitled GNU Free Documentation License. If you do not find the license you may order a copy from Free Software Foundation, Inc. 59 Temple Place, Suite 330 Boston, MA 02111-1307

LINUX® is the registered trademark of Linus Torvalds in the U.S. and other countries. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

HINWEIS

Aufgrund der in jüngster Zeit zunehmend in das Interesse an anderen Übersetzungen, die EMC2 Team hat vor kurzem diese Bemühungen begonnen, eine zu liefern Deutsch-Übersetzung der EMC2 Dokumentation.

Wenn Sie möchten, einen Freiwilligen-Editor für die sein Deutsch-Übersetzung von EMC2, kontaktieren Sie uns bitte.

NOTICE

Because of a recent increase in interest in other translations, the EMC2 team has recently begun this effort to deliver a German Translation of the EMC2 documentation.

If you would like to be a volunteer editor for the German translation of EMC2, please contact us.

Chapter 1

User Foreword

LinuxCNC is modular and flexible. These attributes lead many to see it as a confusing jumble of little things and wonder why it is the way it is. This page attempts to answer that question before you get into the thick of things.

LinuxCNC started at the National Institute of Standards and Technology in the USA. It grew up using Unix as its operating system. Unix made it different. Among early Unix developers there grew a set of code writing ideas that some call the Unix way. These early LinuxCNC authors followed those ways.

Eric S. Raymond, in his book *The Art of Unix Programming*, summarizes the Unix philosophy as the widely-used engineering philosophy, "Keep it Simple, Stupid" (KISS Principle). He then describes how he believes this overall philosophy is applied as a cultural Unix norm, although unsurprisingly it is not difficult to find severe violations of most of the following in actual Unix practice:

- Rule of Modularity: Write simple parts connected by clean interfaces.
- Rule of Clarity: Clarity is better than cleverness.
- Rule of Composition: Design programs to be connected to other programs.
- Rule of Separation: Separate policy from mechanism; separate interfaces from engines.¹

Mr. Raymond offered several more rules but these four describe essential characteristics of the LinuxCNC motion control system.

The **Modularity** rule is critical. Throughout these handbooks you will find talk of the interpreter or task planner or motion or HAL. Each of these is a module or collection of modules. It's modularity that allows you to connect together just the parts you need to run your machine.

The **Clarity** rule is essential. LinuxCNC is a work in progress — it is not finished nor will it ever be. It is complete enough to run most of the machines we want it to run. Much of that progress is achieved because many users and code developers are able to look at the work of others and build on what they have done.

The **Composition** rule allows us to build a predictable control system from the many modules available by making them connectable. We achieve connectability by setting up standard interfaces to sets of modules and following those standards.

The **Separation** rule requires that we make distinct parts that do little things. By separating functions debugging is much easier and replacement modules can be dropped into the system and comparisons easily made.

What does the Unix way mean for you as a user of LinuxCNC. It means that you are able to make choices about how you will use the system. Many of these choices are a part of machine integration, but many also affect the way you will use your machine. As you read you will find many places where you will need to make comparisons. Eventually you will make choices, "I'll use this interface rather than that" or, "I'll write part offsets this way rather than that way." Throughout these handbooks we describe the range of abilities currently available.

As you begin your journey into using LinuxCNC we offer two cautionary notes:²

¹Found at http://en.wikipedia.org/wiki/Unix_philosophy, 07/06/2008

²Found at http://en.wikipedia.org/wiki/Unix_philosophy, 07/06/2008

- Paraphrasing the words of Doug Gwyn on UNIX: "LinuxCNC was not designed to stop its users from doing stupid things, as that would also stop them from doing clever things."
- Likewise the words of Steven King: "LinuxCNC is user-friendly. It just isn't promiscuous about which users it's friendly with."

Chapter 2

LinuxCNC User Introduction

2.1 This Manual

The focus of this manual is on *using* LinuxCNC. It is intended to be used once LinuxCNC is installed and configured. For standard installations see the Getting Started Guide for step by step instructions to get you up and going. For detailed information on installation and configuration of LinuxCNC see the Integrator Manual.

2.2 How LinuxCNC Works

The Enhanced Machine Controller (LinuxCNC) is a lot more than just another CNC mill program. It can control machine tools, robots, or other automated devices. It can control servo motors, stepper motors, relays, and other devices related to machine tools.

There are four main components to the LinuxCNC software:

- a motion controller (EMCMOT)
- a discrete I/O controller (EMCIO)
- a task executor which coordinates them (EMCTASK)
- and one of several graphical user interfaces.

In addition there is a layer called HAL (Hardware Abstraction Layer) which allows configuration of LinuxCNC without the need of recompiling.

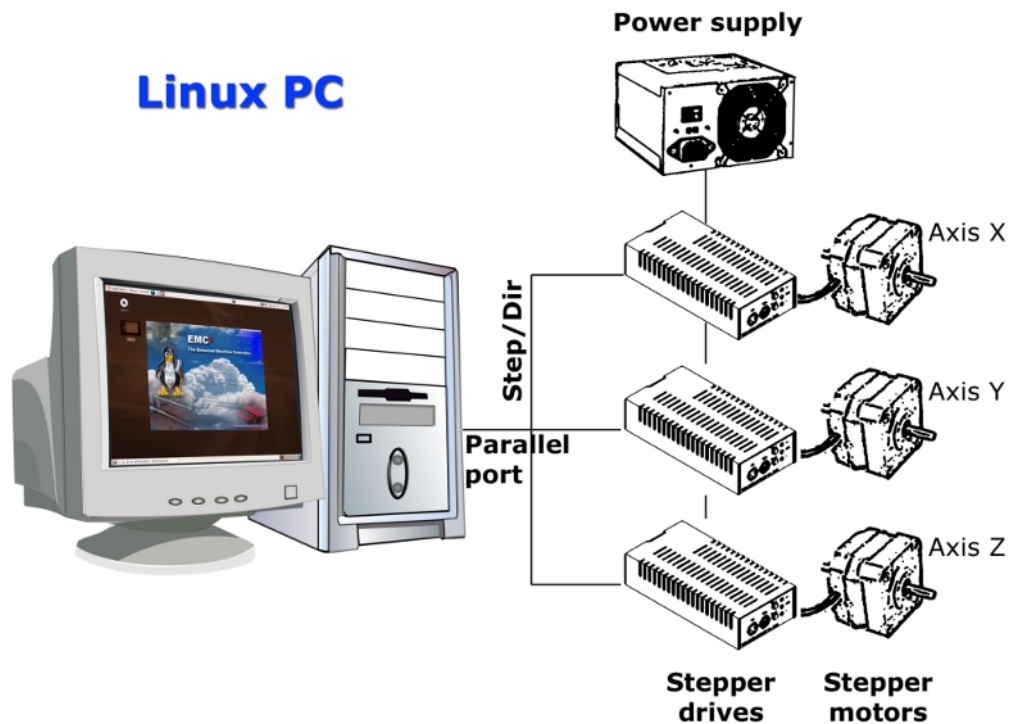


Figure 2.1: Simple LinuxCNC Controlled Machine

The above figure shows a simple block diagram showing what a typical 3-axis LinuxCNC system might look like. This diagram shows a stepper motor system. The PC, running Linux as its operating system, is actually controlling the stepper motor drives by sending signals through the printer port. These signals (pulses) make the stepper drives move the stepper motors. The LinuxCNC system can also run servo motors via servo interface cards or by using an extended parallel port to connect with external control boards. As we examine each of the components that make up an LinuxCNC system we will remind the reader of this typical machine.

2.3 Graphical User Interfaces

A user interface is the part of the LinuxCNC that the machine tool operator interacts with. The LinuxCNC comes with several types of user interfaces:

- [Axis](#), the standard GUI interface.

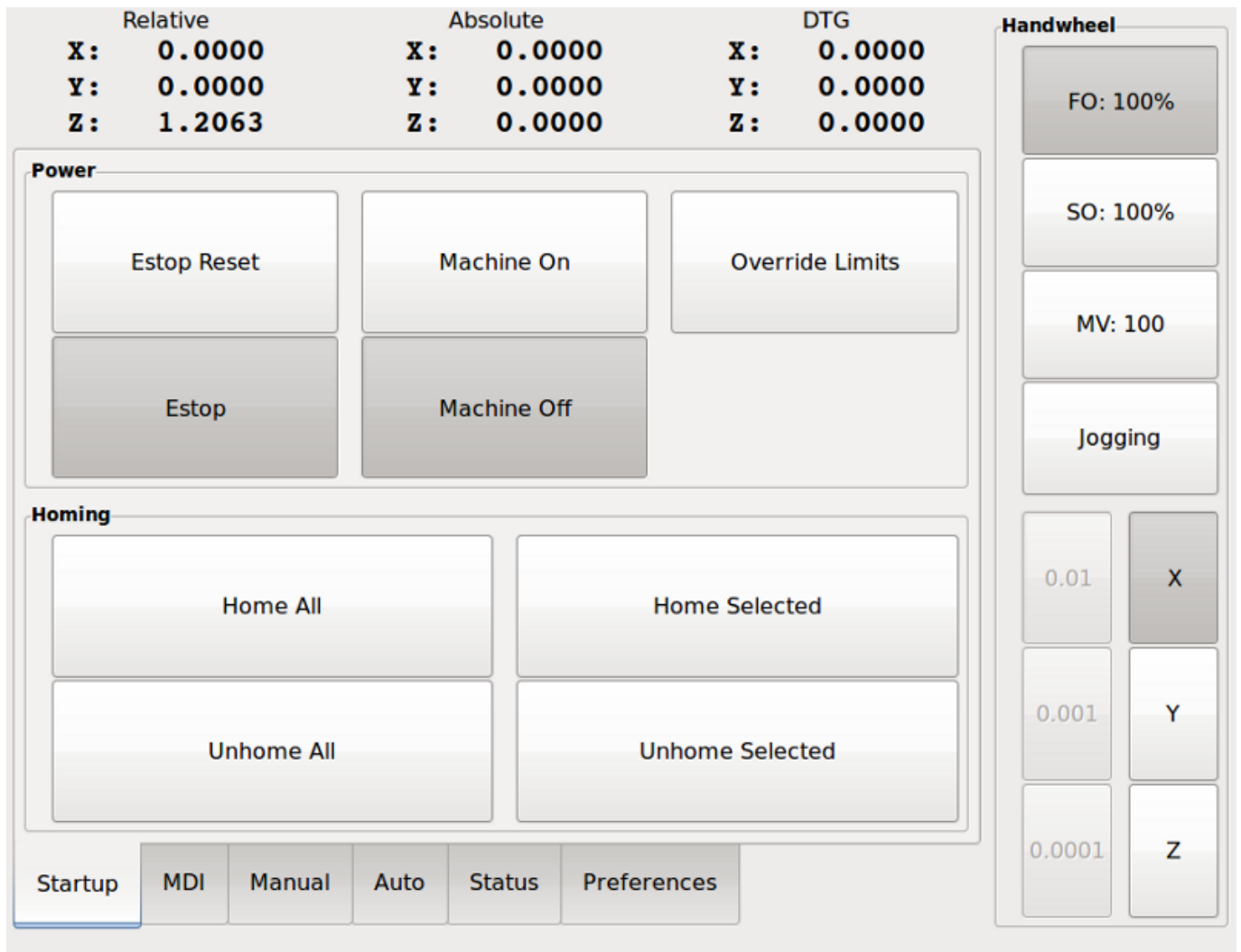


Figure 2.3: Touchy GUI

- [NGCGUI](#), a subroutine GUI that provides *fill in the blanks* programming of G code. It also supports concatenation of subroutine files to enable you to build a complete G code file without programming.

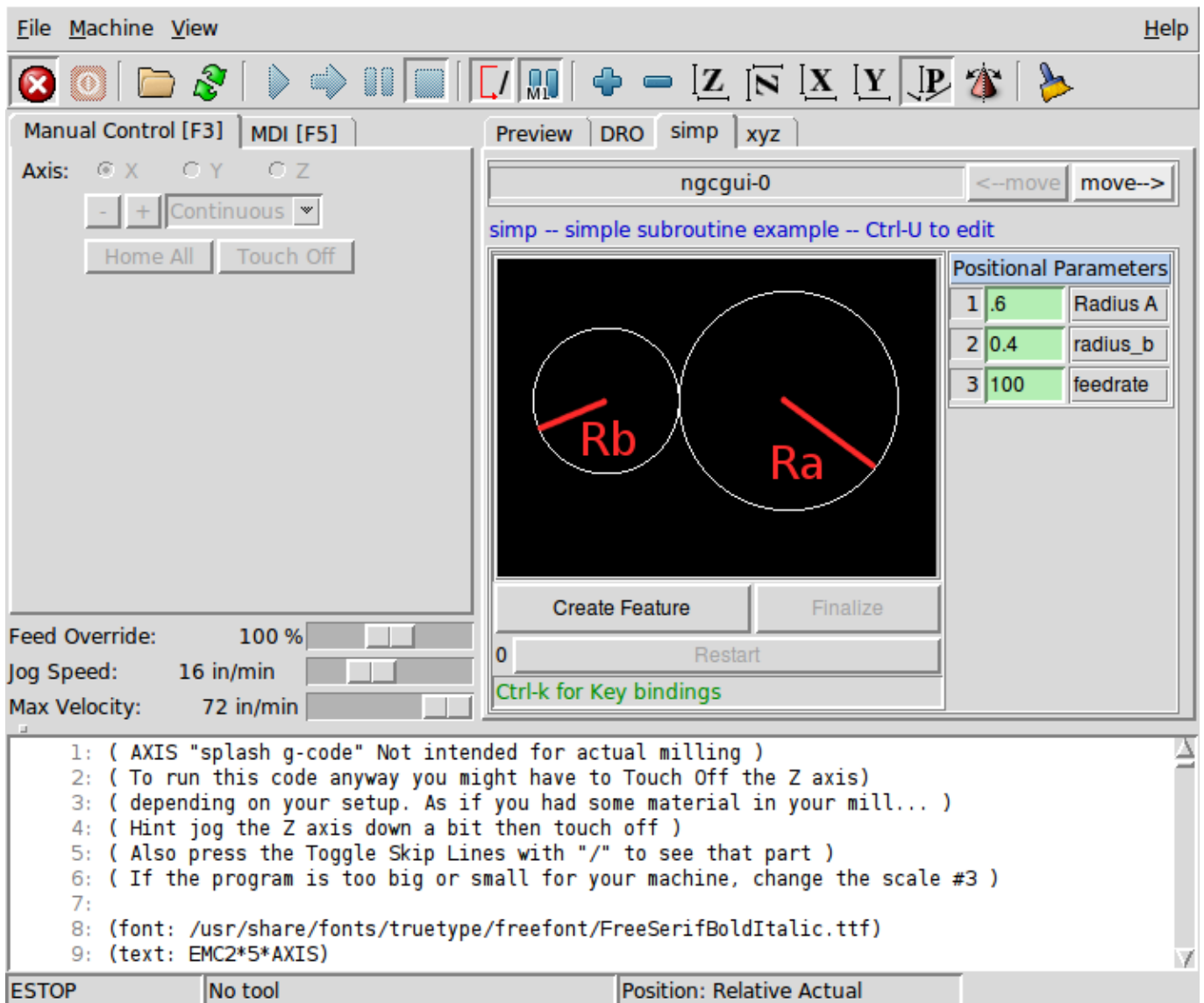


Figure 2.4: NGCGUI GUI imbedded into Axis

- [Mini](#), a Tcl/Tk-based GUI

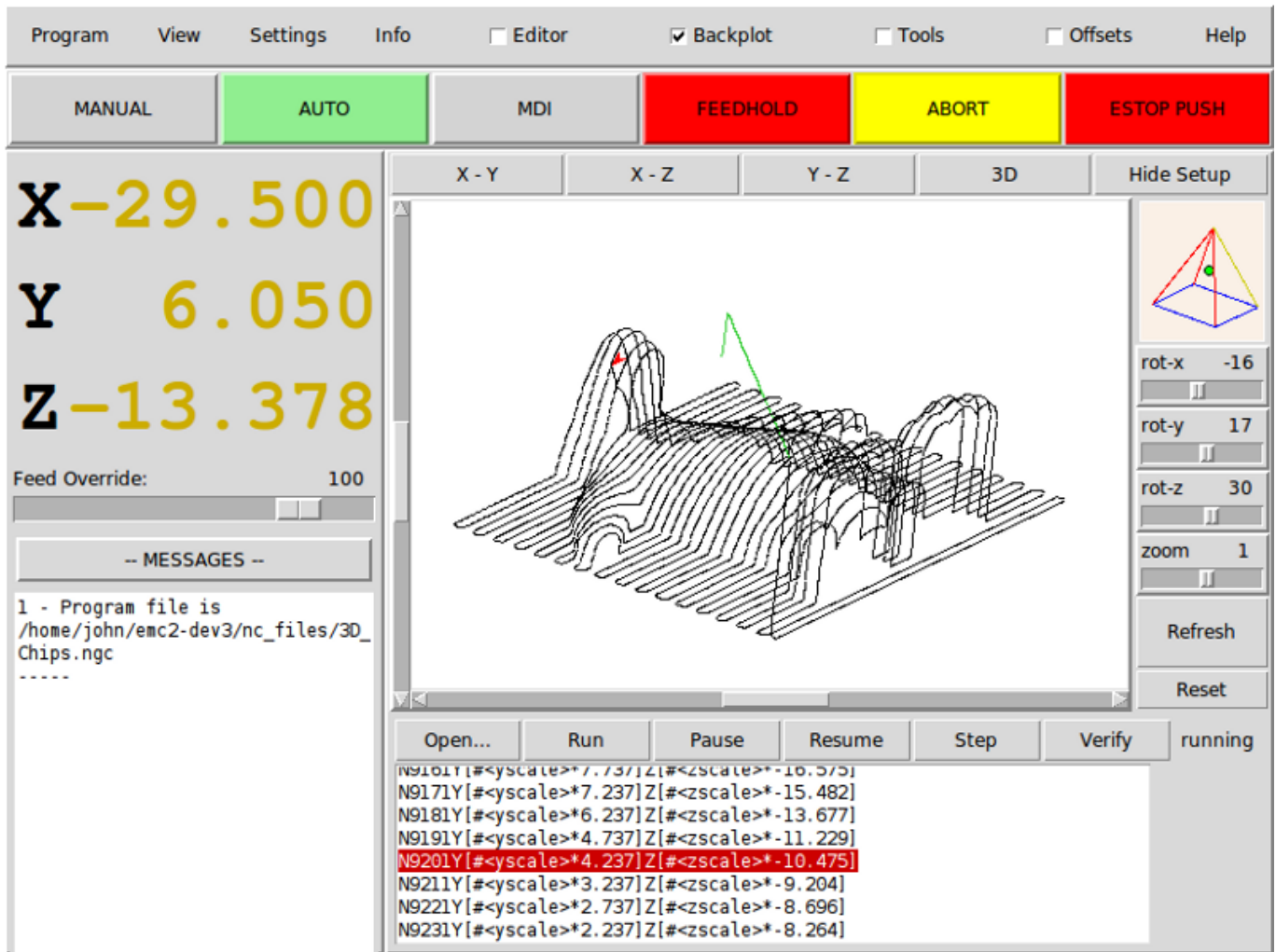


Figure 2.5: The Mini GUI

- [TkLinuxCNC](#), a Tcl/Tk-based GUI

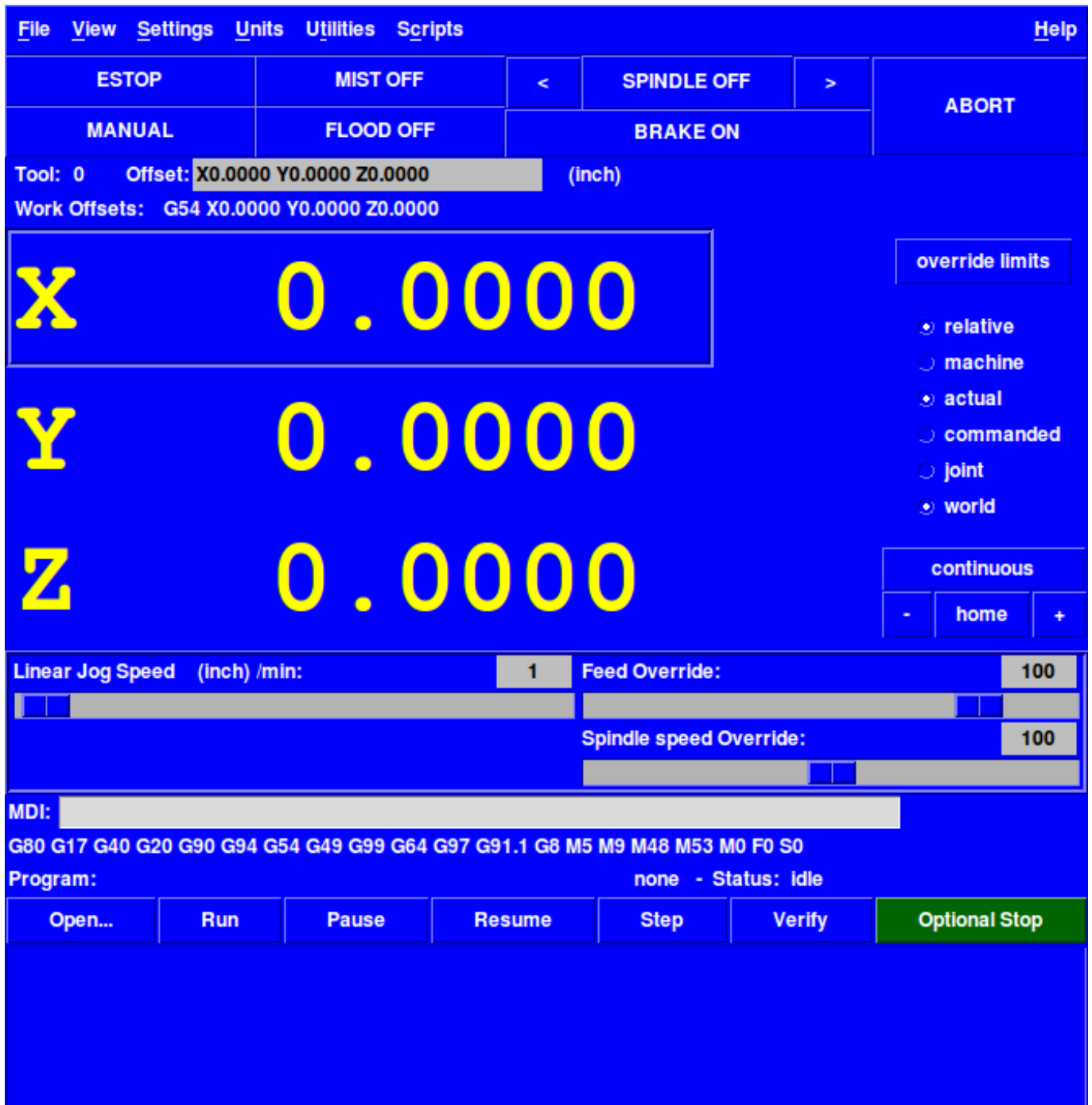


Figure 2.6: The TkLinuxCNC GUI

- [Keystick](#), a character-based screen graphics program suitable for minimal installations (without the X server running).

F1 Estop On/Off	F5 MDI Mode	F9 Spndl Fwd/Off	ESC Aborts Actions
F2 Machine On/Off	F6 Reset Interp	F10 Spndl Rev/Off	TAB Selects Params
F3 Manual Mode	F7 Mist On/Off	F11 Spndl Decrease	END Quits Display
F4 Auto Mode	F8 Flood On/Off	F12 Spndl Increase	? Toggles Help

ESTOP	MANUAL	SPINDLE STOPPED	--- HOMED
Override: <u>100%</u>	LUBE OFF	BRAKE ON	% SELECTED
Tool: <u>0</u>	LUBE OK	MIST OFF	Speed: <u>60.0</u>
Offset: <u>0.0000</u>		FLOOD OFF	Incr: <u>continuous</u>

	--X--	--Y--	--Z--
Relative Act Pos:	0.0000	0.0000	0.0000
	0.0000	0.0000	0.0000

```
-----EMC-HAL-SIM-KEYSTICK EMC Version -----
```

Figure 2.7: The Keystick GUI

- *Xemc*, an X-Windows program. A simulator configuration of Xemc can be ran from the configuration picker.
- *halui* - a HAL based user interface which allows to control LinuxCNC using knobs and switches. See the Integrators manual for more information on halui.
- *linuxcncrsh* - a telnet based user interface which allows commands to be sent to LinuxCNC from remote computers.

2.4 Virtual Control Panels

- *PyVCP* a python based virtual control panel that can be added to the Axis GUI or be stand alone.

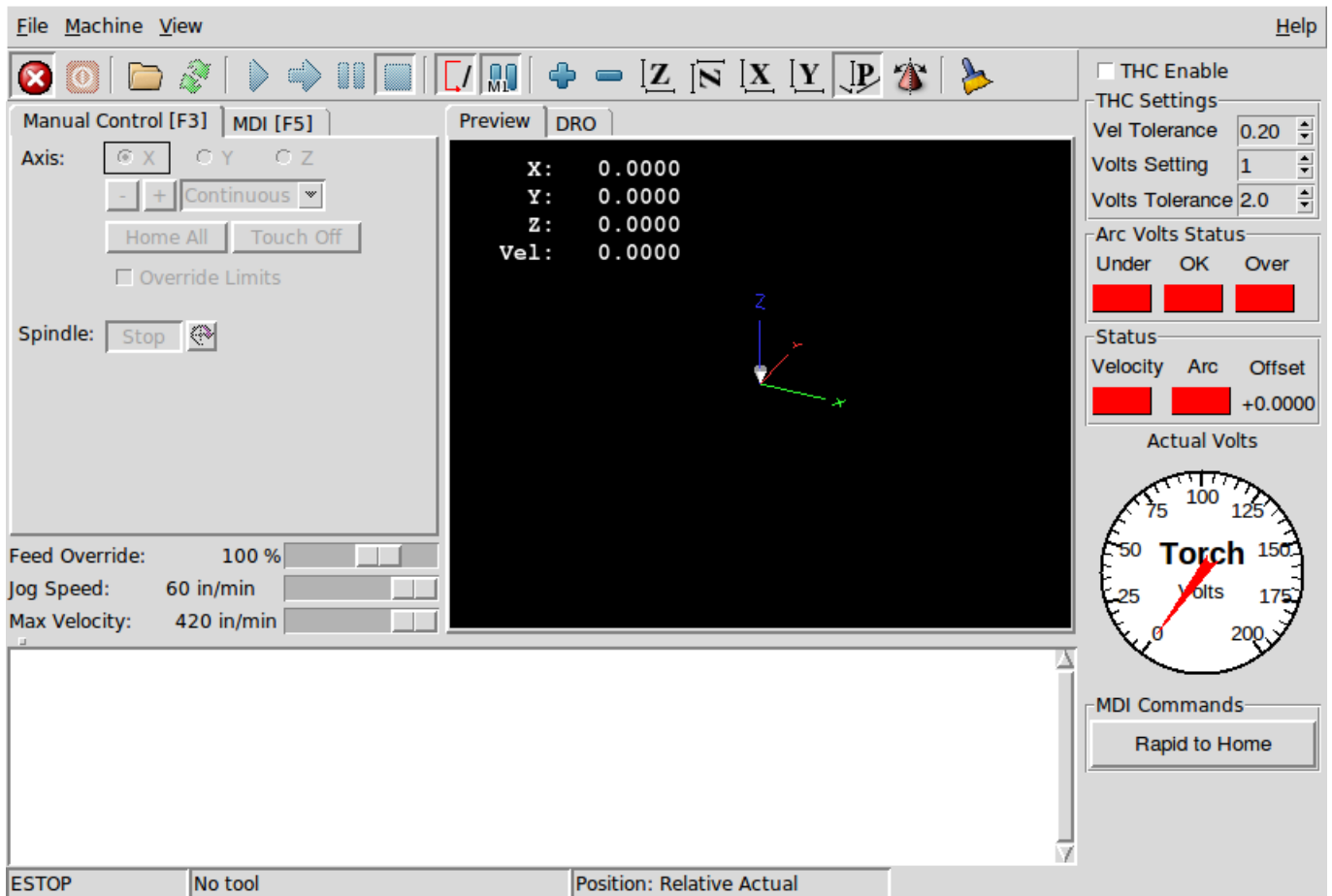


Figure 2.8: PyVCP with Axis

- *GladeVCP* - a glade based virtual control panel that can be added to the Axis GUI or be stand alone.

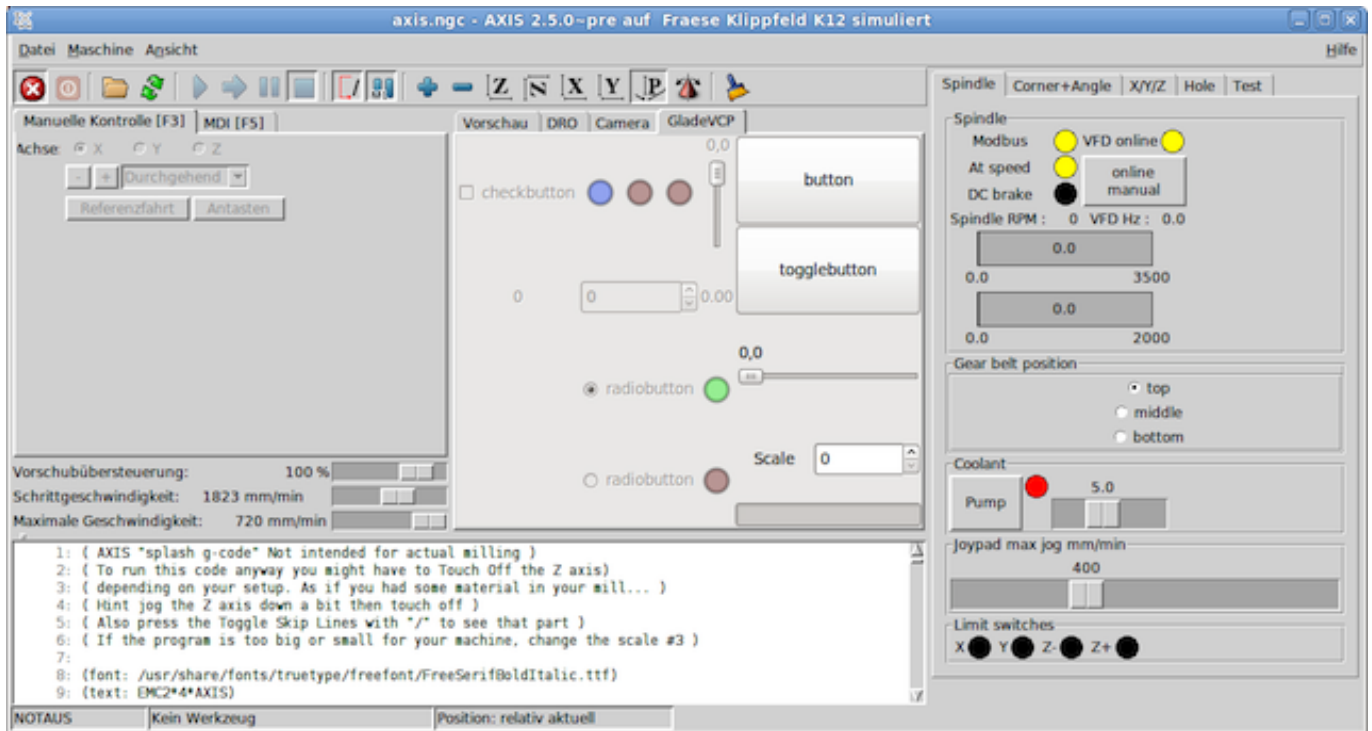


Figure 2.9: GladeVCP with Axis

See the Integrators manual for more information on Virtual Control Panels.

2.5 Languages

LinuxCNC uses translation files to translate LinuxCNC User Interfaces into many languages. You just need to log in with the language you intend to use and when you start up LinuxCNC it comes up in that language. If your language has not been translated contact a developer on the IRC or the mailing list if you can assist in the translation.

2.6 Thinking Like a Machine Operator

This book will not even pretend that it can teach you to run a mill or a lathe. Becoming a machinist takes time and hard work. An author once said, "We learn from experience, if at all." Broken tools, gouged vices, and scars are the evidence of lessons taught. Good part finish, close tolerances, and careful work are the evidence of lessons learned. No machine, no computer program, can take the place of human experience.

As you begin to work with the LinuxCNC program, you will need to place yourself in the position of operator. You need to think of yourself in the role of the one in charge of a machine. It is a machine that is either waiting for your command or executing the command that you have just given it. Throughout these pages we will give information that will help you become a good operator of the LinuxCNC system. You will need some information right up front here so that the following pages will make sense to you.

2.7 Modes of Operation

When LinuxCNC is running, there are three different major modes used for inputting commands. These are *Manual*, *Auto*, and *MDI*. Changing from one mode to another makes a big difference in the way that the LinuxCNC control behaves. There are

specific things that can be done in one mode that cannot be done in another. An operator can home an axis in manual mode but not in auto or MDI modes. An operator can cause the machine to execute a whole file full of G-codes in the auto mode but not in manual or MDI.

In manual mode, each command is entered separately. In human terms a manual command might be *turn on coolant* or *jog X at 25 inches per minute*. These are roughly equivalent to flipping a switch or turning the hand wheel for an axis. These commands are normally handled on one of the graphical interfaces by pressing a button with the mouse or holding down a key on the keyboard. In auto mode, a similar button or key press might be used to load or start the running of a whole program of G-code that is stored in a file. In the MDI mode the operator might type in a block of code and tell the machine to execute it by pressing the <return> or <enter> key on the keyboard.

Some motion control commands are available and will cause the same changes in motion in all modes. These include *abort*, *estop*, and *feed rate override*). Commands like these should be self explanatory.

The AXIS user interface hides some of the distinctions between Auto and the other modes by making Auto-commands available at most times. It also blurs the distinction between Manual and MDI because some Manual commands like Touch Off are actually implemented by sending MDI commands. It does this by automatically changing to the mode that is needed for the action the user has requested.

Chapter 3

Important User Concepts

This chapter covers important user concepts that should be understood before attempting to run a CNC machine with g code.

3.1 Trajectory Control

3.1.1 Trajectory Planning

Trajectory planning, in general, is the means by which LinuxCNC follows the path specified by your G Code program, while still operating within the limits of your machinery.

A G Code program can never be fully obeyed. For example, imagine you specify as a single-line program the following move:

```
G1 X1 F10 (G1 is linear move, X1 is the destination, F10 is the speed)
```

In reality, the whole move can't be made at F10, since the machine must accelerate from a stop, move toward X=1, and then decelerate to stop again. Sometimes part of the move is done at F10, but for many moves, especially short ones, the specified feed rate is never reached at all. Having short moves in your G Code can cause your machine to slow down and speed up for the longer moves if the *naive cam detector* is not employed with G64 Pn.

The basic acceleration and deceleration described above is not complex and there is no compromise to be made. In the INI file the specified machine constraints such as maximum axis velocity and axis acceleration must be obeyed by the trajectory planner.

3.1.2 Path Following

A less straightforward problem is that of path following. When you program a corner in G Code, the trajectory planner can do several things, all of which are right in some cases: it can decelerate to a stop exactly at the coordinates of the corner, and then accelerate in the new direction. It can also do what is called blending, which is to keep the feed rate up while going through the corner, making it necessary to round the corner off in order to obey machine constraints. You can see that there is a trade off here: you can slow down to get better path following, or keep the speed up and have worse path following. Depending on the particular cut, the material, the tooling, etc., the programmer may want to compromise differently.

Rapid moves also obey the current trajectory control. With moves long enough to reach maximum velocity on a machine with low acceleration and no path tolerance specified, you can get a fairly round corner.

3.1.3 Programming the Planner

The trajectory control commands are as follows:

- *G61* - (Exact Path Mode) visits the programmed point exactly, even though that means it might temporarily come to a complete stop in order to change direction to the next programmed point.

- *G61.1* - (Exact Stop Mode) tells the planner to come to an exact stop at every segment's end.
- *G64* - (Blend Without Tolerance Mode) *G64* is the default setting when you start LinuxCNC. *G64* is just blending and the naive cam detector is not enabled. *G64* and *G64 P0* tell the planner to sacrifice path following accuracy in order to keep the feed rate up. This is necessary for some types of material or tooling where exact stops are harmful, and can work great as long as the programmer is careful to keep in mind that the tool's path will be somewhat more curvy than the program specifies. When using *G0* (rapid) moves with *G64* use caution on clearance moves and allow enough distance to clear obstacles based on the acceleration capabilities of your machine.
- *G64 P- Q-* - (Blend With Tolerance Mode) This enables the *naive cam detector* and enables blending with a tolerance. If you program *G64 P0.05*, you tell the planner that you want continuous feed, but at programmed corners you want it to slow down enough so that the tool path can stay within 0.05 user units of the programmed path. The exact amount of slowdown depends on the geometry of the programmed corner and the machine constraints, but the only thing the programmer needs to worry about is the tolerance. This gives the programmer complete control over the path following compromise. The blend tolerance can be changed throughout the program as necessary. Beware that a specification of *G64 P0* has the same effect as *G64* alone (above), which is necessary for backward compatibility for old G Code programs. See the G Code Chapter for more information on *G64 P- Q-*.
- *Blending without tolerance* - The controlled point will touch each specified movement at at least one point. The machine will never move at such a speed that it cannot come to an exact stop at the end of the current movement (or next movement, if you pause when blending has already started). The distance from the end point of the move is as large as it needs to be to keep up the best contouring feed.
- *Naive Cam Detector* - Successive *G1* moves that involve only the XYZ axes that deviate less than *Q-* from a straight line are merged into a single straight line. This merged movement replaces the individual *G1* movements for the purposes of blending with tolerance. Between successive movements, the controlled point will pass no more than *P-* from the actual endpoints of the movements. The controlled point will touch at least one point on each movement. The machine will never move at such a speed that it cannot come to an exact stop at the end of the current movement (or next movement, if you pause when blending has already started) On *G2/3* moves in the *G17* (XY) plane when the maximum deviation of an arc from a straight line is less than the *G64 Q-* tolerance the arc is broken into two lines (from start of arc to midpoint, and from midpoint to end). those lines are then subject to the naive cam algorithm for lines. Thus, line-arc, arc-arc, and arc-line cases as well as line-line benefit from the *naive cam detector*. This improves contouring performance by simplifying the path.

In the following figure the blue line represents the actual machine velocity. The red lines are the acceleration capability of the machine. The horizontal lines below each plot is the planned move. The upper plot shows how the trajectory planner will slow the machine down when short moves are encountered to stay within the limits of the machines acceleration setting to be able to come to an exact stop at the end of the next move. The bottom plot shows the effect of the Naive Cam Detector to combine the moves and do a better job of keeping the velocity as planned.

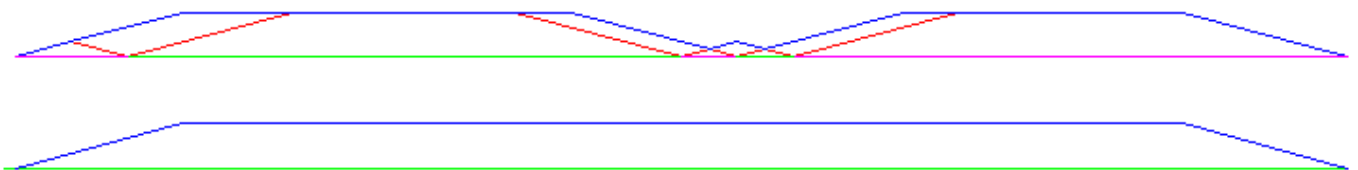


Figure 3.1: Naive Cam Detector

3.1.4 Planning Moves

Make sure moves are *long enough* to suit your machine/material. Principally because of the rule that the machine will never move at such a speed that it cannot come to a complete stop at the end of the current movement, there is a minimum movement length that will allow the machine to keep up a requested feed rate with a given acceleration setting.

The acceleration and deceleration phase each use half the ini file `MAX_ACCELERATION`. In a blend that is an exact reversal, this causes the total axis acceleration to equal the ini file `MAX_ACCELERATION`. In other cases, the actual machine acceleration is somewhat less than the ini file acceleration

To keep up the feed rate, the move must be longer than the distance it takes to accelerate from 0 to the desired feed rate and then stop again. Using A as $1/2$ the ini file MAX_ACCELERATION and F as the feed rate **in units per second**, the acceleration time is $t_a = F/A$ and the acceleration distance is $d_a = F \cdot t_a / 2$. The deceleration time and distance are the same, making the critical distance $d = d_a + d_d = 2 \cdot d_a = F^2/A$.

For example, for a feed rate of 1 inch per second and an acceleration of **10 inches/sec²**, the critical distance is $1^2/10 = 1/10 = 0.1$ inches.

For a feed rate of 0.5 inch per second, the critical distance is $5^2/100 = 25/100 = 0.025$ inches.

3.2 G Code

3.2.1 Defaults

When LinuxCNC first starts up many G and M codes are loaded by default. The current active G and M codes can be viewed on the MDI tab in the *Active G-Codes:* window in the AXIS interface. These G and M codes define the behavior of LinuxCNC and it is important that you understand what each one does before running LinuxCNC. The defaults can be changed when running a G-Code file and left in a different state than when you started your LinuxCNC session. The best practice is to set the defaults needed for the job in the preamble of your G-Code file and not assume that the defaults have not changed. Printing out the G-Code [Quick Reference](#) page can help you remember what each one is.

3.2.2 Feed Rate

How the feed rate is applied depends on if an axis involved with the move is a rotary axis. Read and understand the [Feed Rate](#) section if you have a rotary axis or a lathe.

3.2.3 Tool Radius Offset

Tool Radius Offset (G41/42) requires that the tool be able to touch somewhere along each programmed move without gouging the two adjacent moves. If that is not possible with the current tool diameter you will get an error. A smaller diameter tool may run without an error on the same path. This means you can program a cutter to pass down a path that is narrower than the cutter without any errors. See the [Cutter Compensation](#) Section for more information.

3.3 Homing

After starting LinuxCNC each axis must be homed prior to running a program or running a MDI command.

If your machine does not have home switches a match mark on each axis can aid in homing the machine coordinates to the same place each time.

Once homed your soft limits that are set in the ini file will be used.

If you want to deviate from the default behavior, or want to use the Mini interface you will need to set the option NO_FORCE_HOMING = 1 in the [TRAJ] section of your ini file. More information on homing can be found in the Integrator Manual.

3.4 Tool Changes

There are several options when doing manual tool changes. See the [EMCIO] section of the Integrator Manual for information on configuration of these options. Also see the G28 and G30 section of the User Manual.

3.5 Coordinate Systems

The Coordinate Systems can be confusing at first. Before running a CNC machine you must understand the basics of the coordinate systems used by LinuxCNC. In depth information on the LinuxCNC Coordinate Systems is in the [Coordinate System](#) Section of this manual.

3.5.1 G53 Machine Coordinate

When you home LinuxCNC you set the G53 Machine Coordinate System to 0 for each axis homed.

- No other coordinate systems or tool offsets are changed by homing.

The only time you move in the G53 machine coordinate system is when you program a G53 on the same line as a move. Normally you are in the G54 coordinate system.

3.5.2 G54-59.3 User Coordinates

Normally you use the G54 Coordinate System. When an offset is applied to a current user coordinate system a small blue ball with lines will be at the machine origin when your DRO is displaying *Position: Relative Actual* in Axis. If your offsets are temporary use the Zero Coordinate System from the Machine menu or program *G10 L2 P1 X0 Y0 Z0* at the end of your G Code file. Change the *P* number to suit the coordinate system you wish to clear the offset in.

- Offsets stored in a user coordinate system are retained when LinuxCNC is shut down.
- Using the *Touch Off* button in Axis sets an offset for the chosen User Coordinate System.

3.5.3 When You're Lost

If you're having trouble getting 0,0,0 on the DRO when you think you should, you may have some offsets programmed in and need to remove them.

- Move to the Machine origin with G53 G0 X0 Y0 Z0
- Clear any G92 offset with G92.1
- Use the G54 coordinate system with G54
- Set the G54 coordinate system to be the same as the machine coordinate system with G10 L2 P1 X0 Y0 Z0 R0
- Turn off tool offsets with G49
- Turn on the Relative Coordinate Display from the menu

Now you should be at the machine origin X0 Y0 Z0 and the relative coordinate system should be the same as the machine coordinate system.

3.6 Machine Configurations

The following diagram shows a typical mill showing direction of travel of the tool and the mill table and limit switches. Notice how the mill table moves in the opposite direction of the Cartesian coordinate system arrows shown by the *Tool Direction* image. This makes the *tool* move in the correct direction in relation to the material.

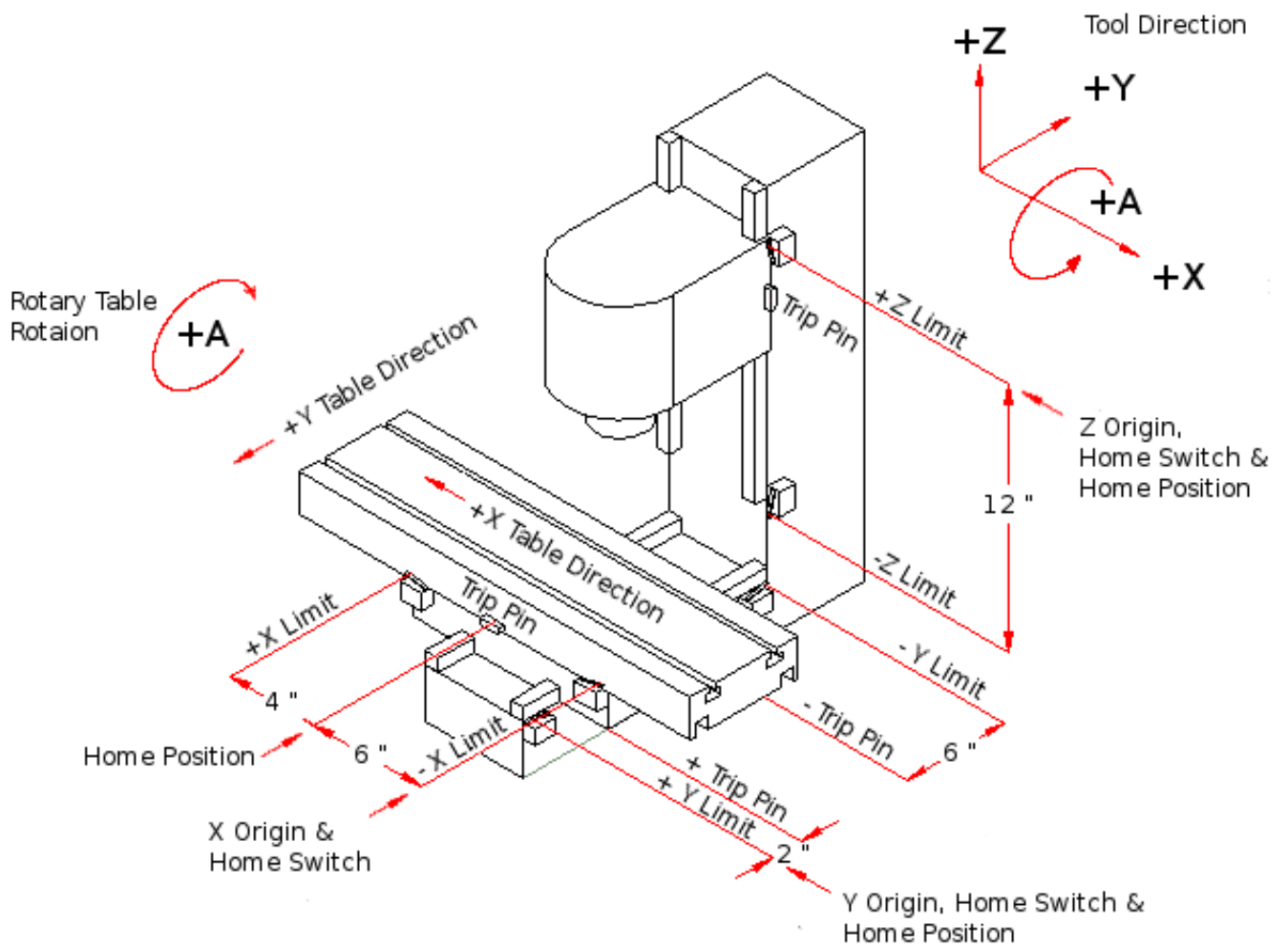


Figure 3.2: Mill Configuration

The following diagram shows a typical lathe showing direction of travel of the tool and limit switches.

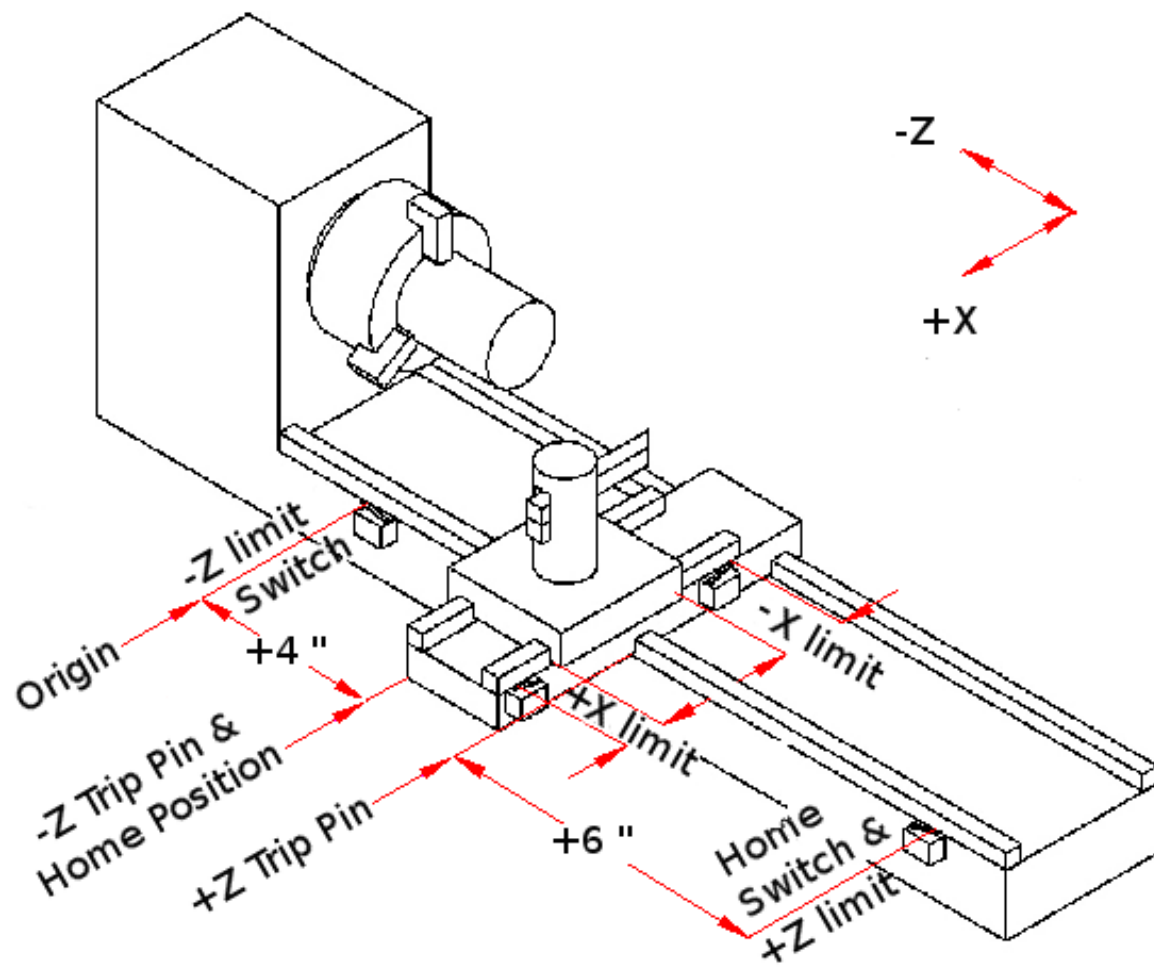


Figure 3.3: Lathe Configuration

Part II

User Interfaces

Chapter 4

AXIS GUI

4.1 Introduction

AXIS is a graphical front-end for LinuxCNC which features a live preview and backplot. It is written in Python and uses Tk and OpenGL to display its user interface.

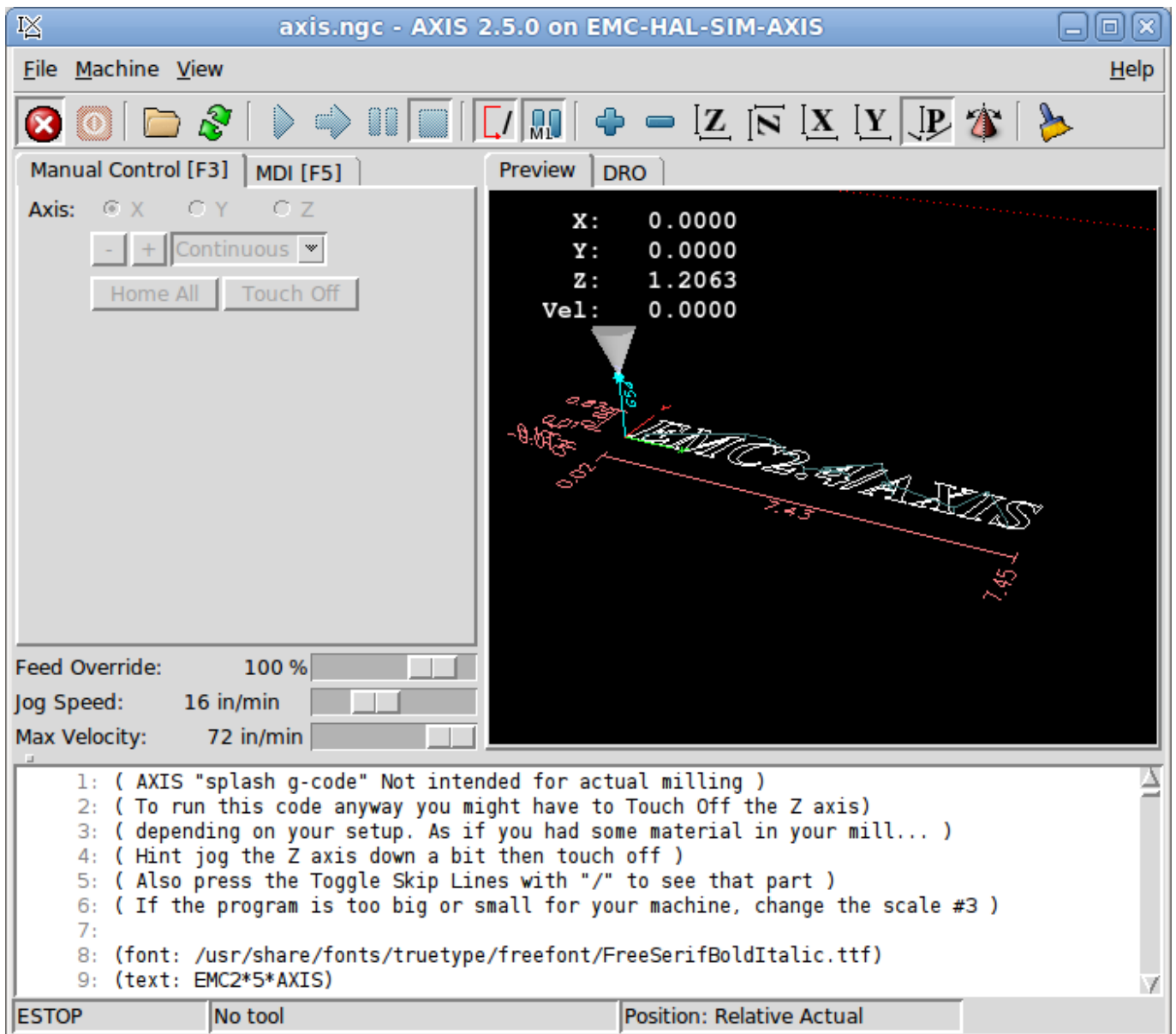


Figure 4.1: AXIS Window

4.2 Getting Started

If your configuration is not currently set up to use AXIS, you can change it by editing the .ini file. In the section `[DISPLAY]` change the `DISPLAY` line to read `DISPLAY = axis`.

The sample configuration `sim/axis.ini` is already configured to use AXIS as its front-end.

4.2.1 A Typical Session

1. Start LinuxCNC.
2. Reset E-STOP (F1) and turn the Machine Power (F2) on.
3. Home all axes.

4. Load the g-code file.
5. Use the preview plot to verify that the program is correct.
6. Load the material.
7. Set the proper offset for each axis by jogging and using the Touch Off button as needed.
8. Run the program.

Note

To run the same program again depends on your setup and requirements. You might need to load more material and set offsets or move over and set an offset then run the program again. If your material is fixtured then you might need to only run the program again. See the [Machine Menu](#) for more information on the run command.

4.3 AXIS Display

The AXIS window contains the following elements:

- A display area that shows one of the following:
 - a preview of the loaded file (in this case, *axis.ngc*), as well as the current location of the CNC machine's *controlled point*. Later, this area will display the path the CNC machine has moved through, called the *backplot*
 - a large readout showing the current position and all offsets.
- A menu bar and toolbar that allow you to perform various actions
- *Manual Control Tab* - which allows you to make the machine move, turn the spindle on or off, and turn the coolant on or off if included in the ini file.
- *MDI Tab* - where G-code programs can be entered manually, one line at a time. This also shows the *Active G Codes* which shows which modal G Codes are in effect.
- *Feed Override* - which allows you to scale the speed of programmed motions. The default maximum is 120% and can be set to a different value in the ini file. See the Integrator Manual for more information on this setting.
- *Spindle Override* - which allows you to scale the spindle speed up or down.
- *Jog Speed* - which allows you to set the jog speed within the limits set in the ini file. See the Integrator Manual for more information on the ini file.
- *Max Velocity* - which allows you to restrict the maximum velocity of all programmed motions (except spindle synchronized motion).
- A text display area that shows the loaded G-Code.
- A status bar which shows the state of the machine. In this screen shot, the machine is turned on, does not have a tool inserted, and the displayed position is *Relative* (showing all offsets), and *Actual* (showing feedback position).

4.3.1 Menu Items

Some menu items might be grayed out depending on how you have your .ini file configured. For more information on configuration see the Integrator Manual.

FILE MENU

- *Open...* - Opens a standard dialog box to open a g code file to load in AXIS. If you have configured LinuxCNC to use a filter program you can also open it up. See the Integrator manual for more information on filter programs.
-

- *Recent Files* - Displays a list of recently opened files.
- *Edit...* - Open the current g code file for editing if you have an editor configured in your ini file. See the Integrator Manual for more information on specifying an editor to use.
- *Reload* - Reload the current g code file. If you edited it you must reload it for the changes to take affect. If you stop a file and want to start from the beginning then reload the file. The toolbar reload is the same as the menu.
- *Save gcode as...* - Save the current file with a new name.
- *Properties* - The sum of the rapid and feed moves. Does not factor in acceleration, blending or path mode so time reported will never be less than the actual run time.
- *Edit tool table...* - Same as Edit if you have defined an editor you can open the tool table and edit it.
- *Reload tool table* - After editing the tool table you must reload it.
- *Ladder editor* - If you have loaded Classic Ladder you can edit it from here. See the Integrator Manual on setting up Classic Ladder
- *Quit* - Terminates the current LinuxCNC session.

MACHINE MENU

- *Toggle Emergency Stop F1* - Change the state of the Emergency Stop.
- *Toggle Machine Power F2* - Change the state of the Machine Power if the Emergency Stop is not on.
- *Run Program* - Run the currently loaded program from the beginning.
- *Run From Selected Line* - Select the line you want to start from first. Use with caution as this will move the tool to the expected position before the line first then it will execute the rest of the code.



Warning

Do not use *Run From Selected Line* if your g code program contains subroutines.

- *Step* - Single step through a program.
 - *Pause* - Pause a program.
 - *Resume* - Resume running from a pause.
 - *Stop* - Stop a running program. When run is selected after a stop the program will start from the beginning.
 - *Stop at M1* - If an M1 is reached, and this is checked, program execution will stop on the M1 line. Press Resume to continue.
 - *Skip lines with "/"* - If a line begins with / and this is checked, the line will be skipped.
 - *Clear MDI history* - Clears the MDI history window.
 - *Copy from MDI history* - Copies the MDI history to the clipboard
 - *Paste to MDI history* - Paste from the clipboard to the MDI history window
 - *Calibration* - Starts a PID tuning assistant, which is mainly for servo systems. Some things can be changed on a stepper system.
 - *Show HAL Configuration* - Opens the HAL Configuration window where you can monitor HAL Components, Pins, Parameters, Signals, Functions, and Threads.
 - *HAL Meter* - Opens a window where you can monitor a single HAL Pin, Signal, or Parameter.
-

- *HAL Scope* - Opens a virtual oscilloscope that allows plotting HAL values vs. time.
- *Show LinuxCNC Status* - Opens a window showing LinuxCNC's status.
- *Set Debug Level* - Opens a window where debug levels can be viewed and some can be set.
- *Homing* - Home one or all axes.
- *Unhoming* - Unhome one or all axes.
- *Zero Coordinate System* - Clear (set to zero) a chosen offset.
- *Tool touch off to workpiece* - When performing Touch Off, the value entered is relative to the current workpiece (*G5x*) coordinate system, as modified by the axis offset (*G92*). When the Touch Off is complete, the Relative coordinate for the chosen axis will become the value entered. See [G10 L10](#) in the G code chapter.
- *Tool touch off to fixture* - When performing Touch Off, the value entered is relative to the ninth (*G59.3*) coordinate system, with the axis offset (*G92*) ignored. This is useful when there is a tool touch-off fixture at a fixed location on the machine, with the ninth (*G59.3*) coordinate system set such that the tip of a zero-length tool is at the fixture's origin when the Relative coordinates are 0. See [G10 L11](#) in the G code chapter.

It's all in your point of view

The AXIS display pick menu *View* refers to *Top*, *Front*, and *Side* views. These terms are correct if the CNC machine has its Z axis vertical, with positive Z up. This is true for vertical mills, which is probably the most popular application, and also true for almost all EDM machines, and even vertical turret lathes, where the part is turning below the tool.

The terms *Top*, *Front*, and *Side* might be confusing however, in other CNC machines, such as a standard lathe, where the Z axis is horizontal, or a horizontal mill, again where the Z axis is horizontal, or even an inverted vertical turret lathe, where the part is turning above the tool, and the Z axis positive direction is down!

Just remember that positive Z axis is (almost) always away from the part. So be familiar with your machine's design and interpret the display as needed.

- *Top View* - The Top View (or Z view) displays the G code looking along the Z axis from positive to negative. This view is best for looking at X & Y.
- *Rotated Top View* - The Rotated Top View (or rotated Z view) also displays the G code looking along the Z axis from positive to negative. But sometimes it's convenient to display the X & Y axes rotated 90 degrees to fit the display better. This view is also best for looking at X & Y.
- *Side View* - The Side View (or X view) displays the G code looking along the X axis from positive to negative. This view is best for looking at Y & Z.
- *Front View* - The Front View (or Y view) displays the G code looking along the Y axis from negative to positive. This view is best for looking at X & Z.
- *Perspective View* - The Perspective View (or P view) displays the G code looking at the part from an adjustable point of view, defaulting to X+, Y-, Z+. The position is adjustable using the mouse and the drag/rotate selector. This view is a compromise view, and while it does do a good job of trying to show three (to nine!) axes on a two-dimensional display, there will often be some feature that is hard to see, requiring a change in viewpoint. This view is best when you would like to see all three (to nine) axes at once.
- *Display Inches* - Set the AXIS display scaling for inches.
- *Display MM* - Set the AXIS display scaling for millimeters.
- *Show Program* - The preview display of the loaded G code program can be entirely disabled if desired.
- *Show Program Rapids* - The preview display of the loaded G code program will always show the feedrate moves (*G1*, *G2*, *G3*) in white. But the display of rapid moves (*G0*) in cyan can be disabled if desired.





- *Alpha-blend Program* - This option makes the preview of complex programs easier to see, but may cause the preview to display more slowly.
- *Show Live Plot* - The highlighting of the feedrate paths (G1,G2,G3) as the tool moves can be disabled if desired.
- *Show Tool* - The display of the tool cone/cylinder can be disabled if desired.
- *Show Extents* - The display of the extents (maximum travel in each axis direction) of the loaded G code program can be disabled if desired.
- *Show Offsets* - The selected fixture offset (G54-G59.3) origin location can be shown as a set of three orthogonal lines, one each of red, blue, and green. This offset origin (or fixture zero) display can be disabled if desired.
- *Show Machine Limits* - The machine's maximum travel limits for each axis, as set in the ini file, are shown as a rectangular box drawn in red dashed lines. This is useful when loading a new G code program, or when checking for how much fixture offset would be needed to bring the G code program within the travel limits of your machine. It can be shut off if not needed.
- *Show Velocity* - A display of velocity is sometimes useful to see how close your machine is running to its design velocities. It can be disabled if desired.
- *Show Distance to Go* - Distance to go is a very handy item to know when running an unknown G code program for the first time. In combination with the rapid override and feedrate override controls, unwanted tool and machine damage can be avoided. Once the G code program has been debugged and is running smoothly, the Distance to Go display can be disabled if desired.
- *Clear Live Plot* - As the tool travels in the Axis display, the G code path is highlighted. To repeat the program, or to better see an area of interest, the previously highlighted paths can be cleared.
- *Show Commanded Position* - This is the position that LinuxCNC will try to go to. Once motion has stopped, this is the position LinuxCNC will try to hold.
- *Show Actual Position* - Actual Position is the measured position as read back from the system's encoders or simulated by step generators. This may differ slightly from the Commanded Position for many reasons including PID tuning, physical constraints, or position quantization.
- *Show Machine Position* - This is the position in unoffset coordinates, as established by Homing.
- *Show Relative Position* - This is the Machine Position modified by G5x, G92, and G43 offsets.
















HELP MENU

- *About Axis* - We all know what this is.
- *Quick Reference* - Shows the keyboard shortcut keys.

4.3.2 Toolbar buttons

From left to right in the Axis display, the toolbar buttons (keyboard shortcuts shown [in brackets]) are:

-  Toggle Emergency Stop [F1] (also called E-Stop)
-  Toggle Machine Power [F2]
-  Open G Code file [O]
-  Reload current file [Ctrl-R]

-  Begin executing the current file [R]
-  Execute next line [T]
-  Pause Execution [P] Resume Execution[S]
-  Stop Program Execution [ESC]
-  Toggle Skip lines with "/" [Alt-M-/]
-  Toggle Optional Pause [Alt-M-1]
-  Zoom In
-  Zoom Out
-  Top view
-  Rotated Top view
-  Side view
-  Front view
-  Perspective view
-  Toggle between Drag and Rotate Mode [D]
-  Clear live backplot [Ctrl-K]

4.3.3 Graphical Display Area

Coordinate Display In the upper-left corner of the program display is the coordinate display. It shows the position of the machine. To the left of the axis name, an origin symbol is shown if the axis has been homed.



A limit symbol is shown if the axis is on one of its limit switches.



To properly interpret these numbers, refer to the *Position:* indicator in the status bar. If the position is *Absolute*, then the displayed number is in the machine coordinate system. If it is *Relative*, then the displayed number is in the offset coordinate system. When the coordinates displayed are relative and an offset has been set, the display will include a cyan *machine origin* marker.



If the position is *Commanded*, then it is the ideal position --for instance, the exact coordinate given in a *G0* command. If it is *Actual*, then it is the position the machine has actually moved to. These values can differ for several reasons: Following error, dead band, encoder resolution, or step size. For instance, if you command a movement to X 0.0033 on your mill, but one step of your stepper motor or one encoder count is 0.00125, then the *Commanded* position might be 0.0033, but the *Actual* position will be 0.0025 (2 steps) or 0.00375 (3 steps).

Preview Plot

When a file is loaded, a preview of it is shown in the display area. Fast moves (such as those produced by the *G0* command) are shown as cyan lines. Moves at a feed rate (such as those produced by the *G1* command) are shown as solid white lines. Dwells (such as those produced by the *G4* command) are shown as small pink X marks.

G0 (Rapid) moves prior to a feed move will not show on the preview plot. Rapid moves after a T<n> (Tool Change) will not show on the preview until after the first feed move. To turn either of these features off program a G1 without any moves prior to the G0 moves.

Program Extents

The *extents* of the program in each axis are shown. At the ends, the least and greatest coordinate values are indicated. In the middle, the difference between the coordinates is shown.

When some coordinates exceed the *soft limits* in the .ini file, the relevant dimension is shown in a different color and enclosed by a box. In figure below the maximum soft limit is exceeded on the X axis as indicated by the box surrounding the coordinate value. The minimum X travel of the program is -1.95, the maximum X travel is 1.88, and the program requires 3.83 inches of X travel. To move the program so it's within the machine's travel in this case, jog to the left and Touch Off X again.

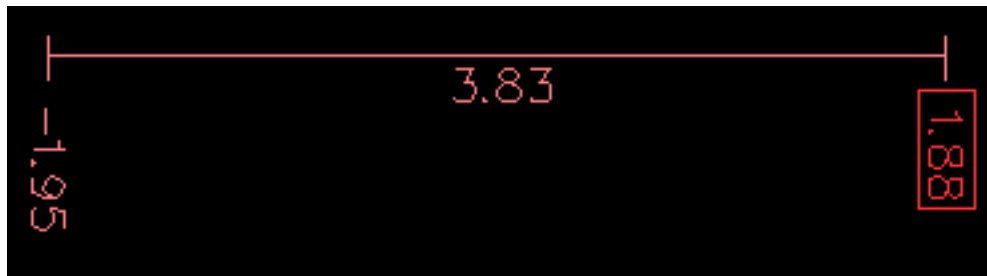


Figure 4.2: Soft Limit

Tool Cone When no tool is loaded, the location of the tip of the tool is indicated by the *tool cone*. The *tool cone* does not provide guidance on the form, length, or radius of the tool.

When a tool is loaded (for instance, with the MDI command *T1 M6*), the cone changes to a cylinder which shows the diameter of the tool given in the tool table file.

Backplot When the machine moves, it leaves a trail called the backplot. The color of the line indicates the type of motion: Yellow for jogs, faint green for rapid movements, red for straight moves at a feed rate, and magenta for circular moves at a feed rate.

Interacting By left-clicking on a portion of the preview plot, the line will be highlighted in both the graphical and text displays. By left-clicking on an empty area, the highlighting will be removed.

By dragging with the left mouse button pressed, the preview plot will be shifted (panned).

By dragging with shift and the left mouse button pressed, or by dragging with the mouse wheel pressed, the preview plot will be rotated. When a line is highlighted, the center of rotation is the center of the line. Otherwise, the center of rotation is the center of the entire program.

By rotating the mouse wheel, or by dragging with the right mouse button pressed, or by dragging with control and the left mouse button pressed, the preview plot will be zoomed in or out.

By clicking one of the *Preset View* icons, or by pressing V, several preset views may be selected.

4.3.4 Text Display Area

By left-clicking a line of the program, the line will be highlighted in both the graphical and text displays.

When the program is running, the line currently being executed is highlighted in red. If no line has been selected by the user, the text display will automatically scroll to show the current line.

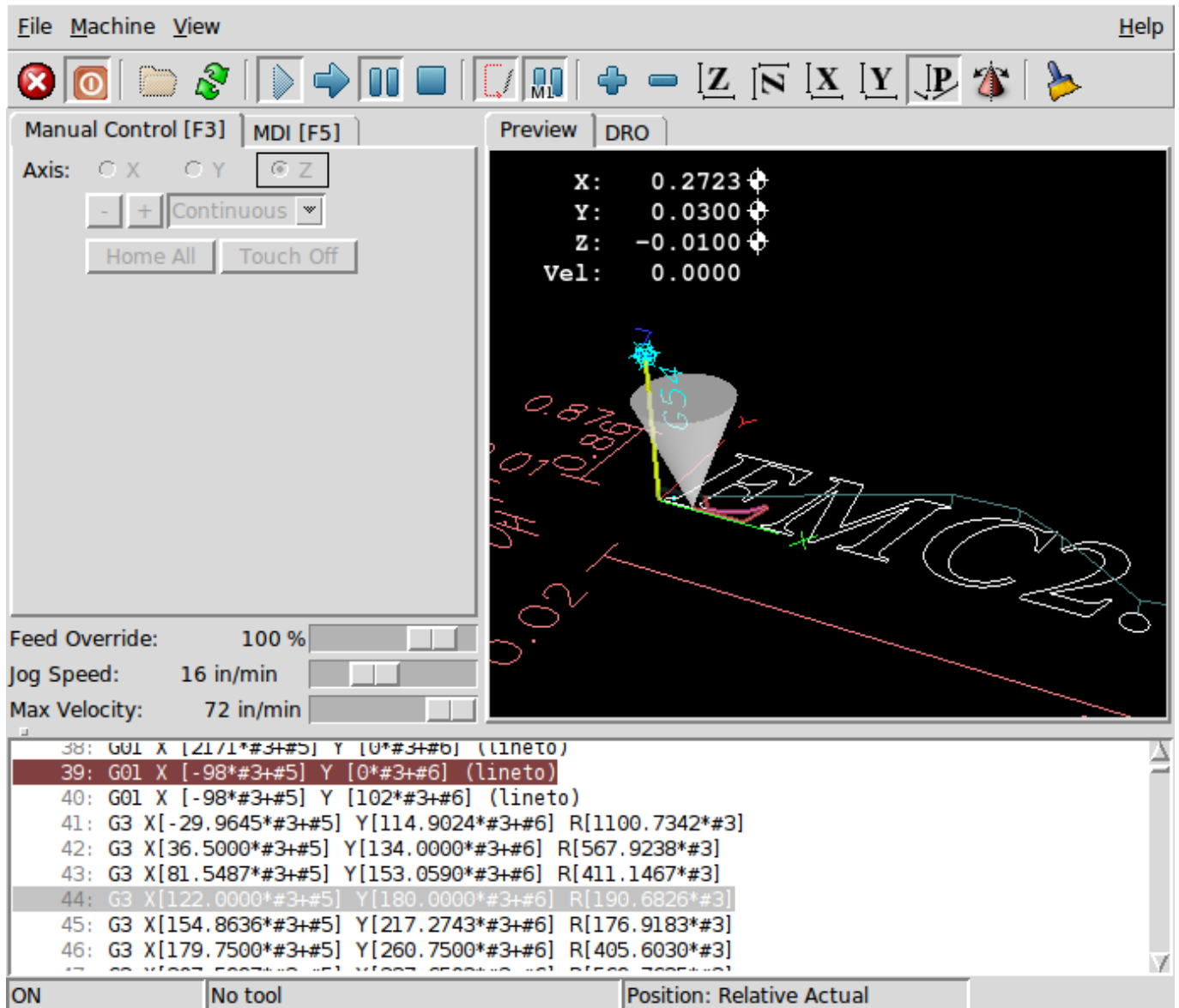


Figure 4.3: Current and Selected Lines

4.3.5 Manual Control

While the machine is turned on but not running a program, the items in the *Manual Control* tab can be used to move the machine or control its spindle and coolant.

When the machine is not turned on, or when a program is running, the manual controls are unavailable.

Many of the items described below are not useful on all machines. When AXIS detects that a particular pin is not connected in HAL, the corresponding item in the Manual Control tab is removed. For instance, if the HAL pin *motion.spindle-brake* is not

connected, then the *Brake* button will not appear on the screen. If the environment variable *AXIS_NO_AUTOCONFIGURE* is set, this behavior is disabled and all the items will appear.

The Axis group *Axis* allows you to manually move the machine. This action is known as *jogging*. First, select the axis to be moved by clicking it. Then, click and hold the + or - button depending on the desired direction of motion. The first four axes can also be moved by the arrow keys (X and Y), PAGE UP and PAGE DOWN keys (Z), and the [and] keys (A).

If *Continuous* is selected, the motion will continue as long as the button or key is pressed. If another value is selected, the machine will move exactly the displayed distance each time the button is clicked or the key is pressed. By default, the available values are 0.1000, 0.0100, 0.0010, 0.0001

See the Configure section of the Integrator Manual for more information on setting the increments.

Homing If your machine has home switches and a homing sequence defined for all axes the button will read *Home All*. The *Home All* button or the Ctrl-HOME key will home all axes using the homing sequence. Pressing the HOME key will home the current axis, even if a homing sequence is defined.

If your machine has home switches and no homing sequence is defined or not all axes have a homing sequence the button will read *Home* and will home the selected axis only. Each axis must be selected and homed separately.

If your machine does not have home switches defined in the configuration the *Home* button will set the current selected axis current position to be the absolute position 0 for that axis and will set the *is-homed* bit for that axis.

See the Integrator Manual for more information on homing.

Touch Off By pressing *Touch Off* or the END key, the *G54 offset* for the current axis is changed so that the current axis value will be the specified value. Expressions may be entered using the rules for rs274ngc programs, except that variables may not be referred to. The resulting value is shown as a number.

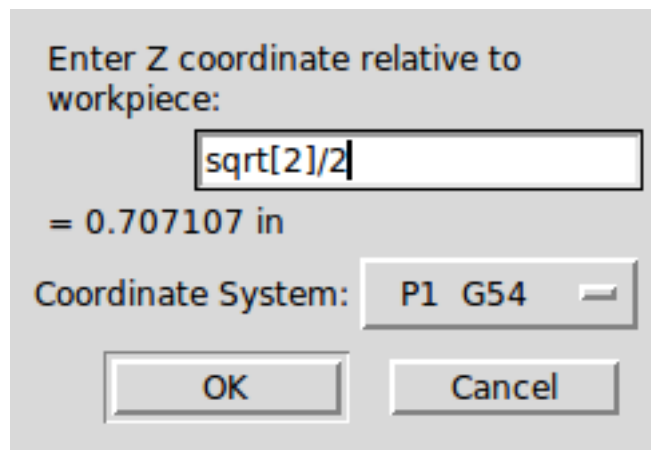


Figure 4.4: Touch Off

See also the *Tool touch off to workpiece* and *Tool touch off to fixture* options in the Machine menu.

Override Limits By pressing *Override Limits*, the machine will temporarily be allowed to jog off of a physical limit switch. This checkbox is only available when a limit switch is tripped. The override is reset after one jog. If the axis is configured with separate positive and negative limit switches, LinuxCNC will allow the jog only in the correct direction. *Override Limits will not allow a jog past a soft limit. The only way to disable a soft limit on an axis is to Unhome it.*

The Spindle group

The buttons on the first row select the direction for the spindle to rotate: Counterclockwise, Stopped, Clockwise. Counterclockwise will only show up if the pin *motion.spindle-reverse* is in the HAL file (it can be *net trick-axis motion.spindle-reverse*). The buttons on the next row increase or decrease the rotation speed. The checkbox on the third row allows the spindle brake to be engaged or released. Depending on your machine configuration, not all the items in this group may appear. Pressing the spindle start button sets the *S* speed to 1.

The Coolant group

The two buttons allow the *Mist* and *Flood* coolants to be turned on and off. Depending on your machine configuration, not all the items in this group may appear.

4.3.6 MDI

MDI allows G-code commands to be entered manually. When the machine is not turned on, or when a program is running, the MDI controls are unavailable.

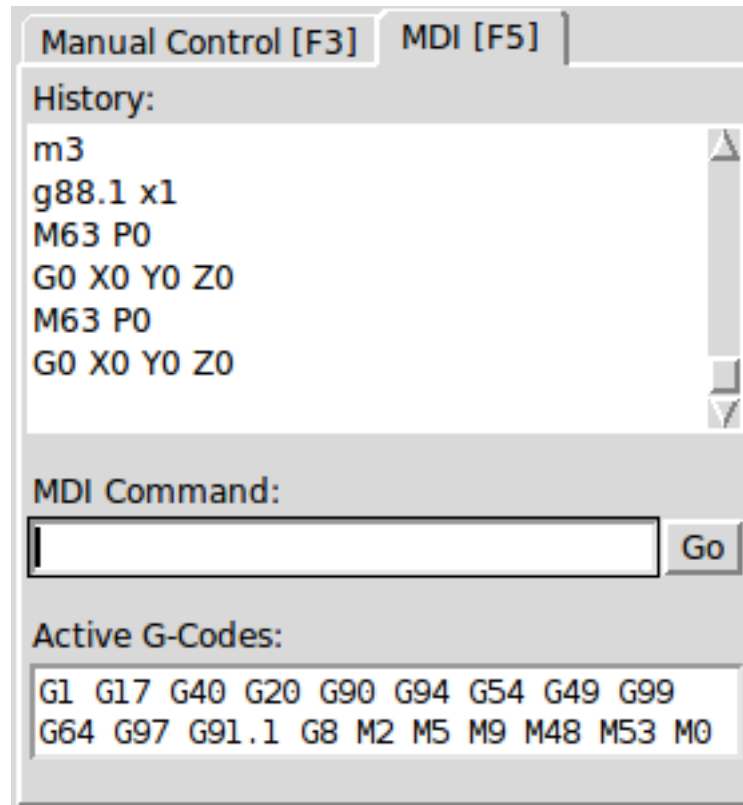


Figure 4.5: The MDI tab

- *History* - This shows MDI commands that have been typed earlier in this session.
- *MDI Command* - This allows you to enter a g-code command to be executed. Execute the command by pressing Enter or by clicking *Go*.
- *Active G-Codes* - This shows the *modal codes* that are active in the interpreter. For instance, *G54* indicates that the *G54 offset* is applied to all coordinates that are entered. When in Auto the Active G-Codes represent the codes after any read ahead by the interpreter.

4.3.7 Feed Override

By moving this slider, the programmed feed rate can be modified. For instance, if a program requests *F60* and the slider is set to 120%, then the resulting feed rate will be 72.

4.3.8 Spindle Speed Override

By moving this slider, the programmed spindle speed can be modified. For instance, if a program requests *S8000* and the slider is set to 80%, then the resulting spindle speed will be 6400. This item only appears when the HAL pin *motion.spindle-speed-out* is connected.

4.3.9 Jog Speed

By moving this slider, the speed of jogs can be modified. For instance, if the slider is set to 1 in/min, then a .01 inch jog will complete in about .6 seconds, or 1/100 of a minute. Near the left side (slow jogs) the values are spaced closely together, while near the right side (fast jogs) they are spaced much further apart, allowing a wide range of jog speeds with fine control when it is most important.

On machines with a rotary axis, a second jog speed slider is shown. This slider sets the jog rate for the rotary axes (A, B and C).

4.3.10 Max Velocity

By moving this slider, the maximum velocity can be set. This caps the maximum velocity for all programmed moves except spindle-synchronized moves.

4.4 Keyboard Controls

Almost all actions in AXIS can be accomplished with the keyboard. A full list of keyboard shortcuts can be found in the AXIS Quick Reference, which can be displayed by choosing Help > Quick Reference. Many of the shortcuts are unavailable when in MDI mode.

Feed Override Keys The Feed Override keys behave differently when in Manual Mode. The keys '12345678 will select an axis if it is programed. If you have 3 axis then ' will select axis 0, 1 will select axis 1, and 2 will select axis 2. The remainder of the number keys will still set the Feed Override. When running a program '1234567890 will set the Feed Override to 0% - 100%.

The most frequently used keyboard shortcuts are shown in the following Table

Table 4.1: Most Common Keyboard Shortcuts

Keystroke	Action Taken	Mode
F1	Toggle Emergency Stop	All
F2	Turn machine on/off	All
`, 1 .. 9, 0	Set feed override from 0% to 100%	Varies
X, `	Activate first axis	Manual
Y, 1	Activate second axis	Manual
Z, 2	Activate third axis	Manual
A, 3	Activate fourth axis	Manual
I	Select jog increment	Manual
C	Continuous jog	Manual
Control-Home	Perform homing sequence	Manual
End	Touch off: Set G54 offset for active axis	Manual
Left, Right	Jog first axis	Manual
Up, Down	Jog second axis	Manual
Pg Up, Pg Dn	Jog third axis	Manual
[,]	Jog fourth axis	Manual
O	Open File	Manual
Control-R	Reload File	Manual
R	Run file	Manual
P	Pause execution	Auto
S	Resume Execution	Auto
ESC	Stop execution	Auto
Control-K	Clear backplot	Auto/Manual
V	Cycle among preset views	Auto/Manual
Shift-Left,Right	Rapid X Axis	Manual
Shift-Up,Down	Rapid Y Axis	Manual
Shift-PgUp, PgDn	Rapid Z Axis	Manual


```
$ MDI ~/emc2/configs/sim/emc.nml
MDI>
(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI> G1 F5 X1
MDI>
(0.5928500000000374, 0.0, 0.0, 0.0, 0.0, 0.0)
MDI>
(1.0000000000000639, 0.0, 0.0, 0.0, 0.0, 0.0)
```

4.7 axis-remote

AXIS includes a program called *axis-remote* which can send certain commands to a running AXIS. The available commands are shown by running *axis-remote --help* and include checking whether AXIS is running (*--ping*), loading a file by name, reloading the currently loaded file (*--reload*), and making AXIS exit (*--quit*).

4.8 Manual Tool Change

LinuxCNC includes a userspace HAL component called *hal_manualtoolchange*, which shows a window prompt telling you what tool is expected when a *M6* command is issued. After the OK button is pressed, execution of the program will continue.

The HAL configuration file *configs/sim/axis_manualtoolchange.hal* shows the HAL commands necessary to use this component. *hal_manualtoolchange* can be used even when AXIS is not used as the GUI. This component is most useful if you have presettable tools and you use the tool table.

Note

Important Note: Rapids will not show on the preview after a *T<n>* is issued until the next feed move after the *M6*. This can be very confusing to most users. To turn this feature off for the current tool change program a *G1* with no move after the *T<n>*.

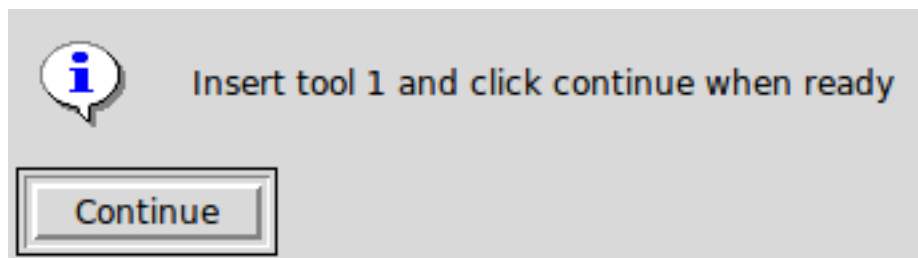


Figure 4.7: The Manual Toolchange Window

4.9 Python modules

AXIS includes several Python modules which may be useful to others. For more information on one of these modules, use *pydoc <module name>* or read the source code. These modules include:

- *emc* provides access to the LinuxCNC command, status, and error channels
- *gcode* provides access to the rs274ngc interpreter
- *rs274* provides additional tools for working with rs274ngc files

- *hal* allows the creation of userspace HAL components written in Python
- *_togl* provides an OpenGL widget that can be used in Tkinter applications
- *minigl* provides access to the subset of OpenGL used by AXIS

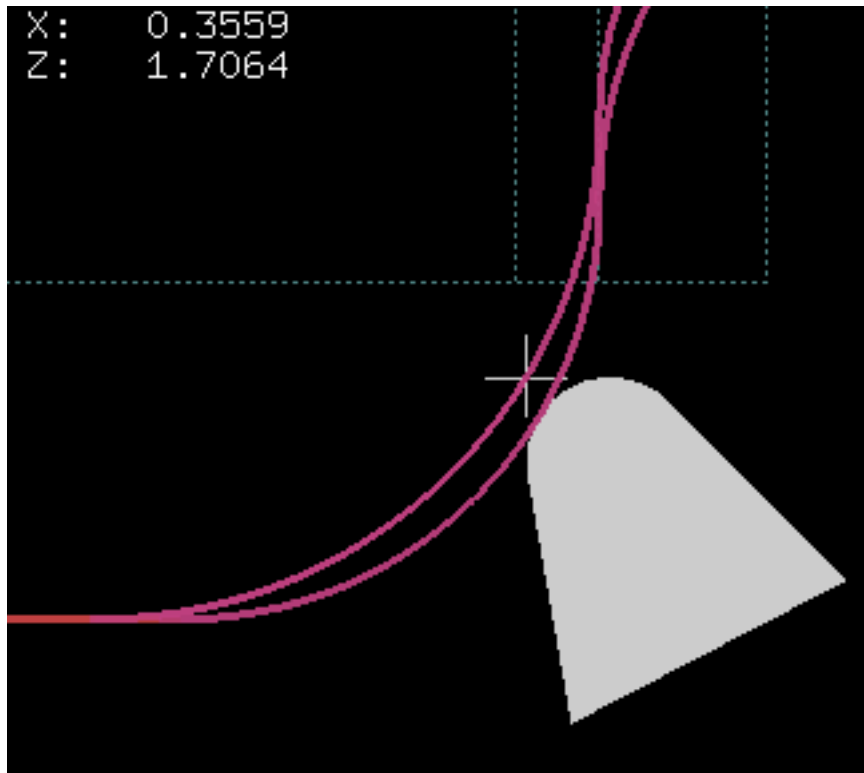
To use these modules in your own scripts, you must ensure that the directory where they reside is on Python's module path. When running an installed version of LinuxCNC, this should happen automatically. When running *in-place*, this can be done by using *scripts/rip-environment*.

4.10 Using AXIS in Lathe Mode

By including the line *LATHE = 1* in the [DISPLAY] section of the ini file, AXIS selects lathe mode. The *Y* axis is not shown in coordinate readouts, the view is changed to show the *Z* axis extending to the right and the *X* axis extending towards the bottom of the screen, and several controls (such as those for preset views) are removed. The coordinate readouts for *X* are replaced with diameter and radius.

Pressing *V* zooms out to show the entire file, if one is loaded.

When in lathe mode, the shape of the loaded tool (if any) is shown.



Lathe Tool Shape

4.11 Advanced Configuration

For more information on ini file settings that can change how AXIS works see the INI File/Sections/[DISPLAY] Section of Configuration chapter in the Integrator manual.

4.11.1 Program Filters

AXIS has the ability to send loaded files through a *filter program*. This filter can do any desired task: Something as simple as making sure the file ends with *M2*, or something as complicated as generating G-Code from an image.

The *[FILTER]* section of the ini file controls how filters work. First, for each type of file, write a *PROGRAM_EXTENSION* line. Then, specify the program to execute for each type of file. This program is given the name of the input file as its first argument, and must write rs274ngc code to standard output. This output is what will be displayed in the text area, previewed in the display area, and executed by LinuxCNC when *Run*. The following lines add support for the *image-to-gcode* converter included with LinuxCNC:

```
[FILTER]
PROGRAM_EXTENSION = .png,.gif Greyscale Depth Image
png = image-to-gcode
gif = image-to-gcode
```

It is also possible to specify an interpreter:

```
PROGRAM_EXTENSION = .py Python Script
py = python
```

In this way, any Python script can be opened, and its output is treated as g-code. One such example script is available at *nc_files/holecircle.py*. This script creates g-code for drilling a series of holes along the circumference of a circle.

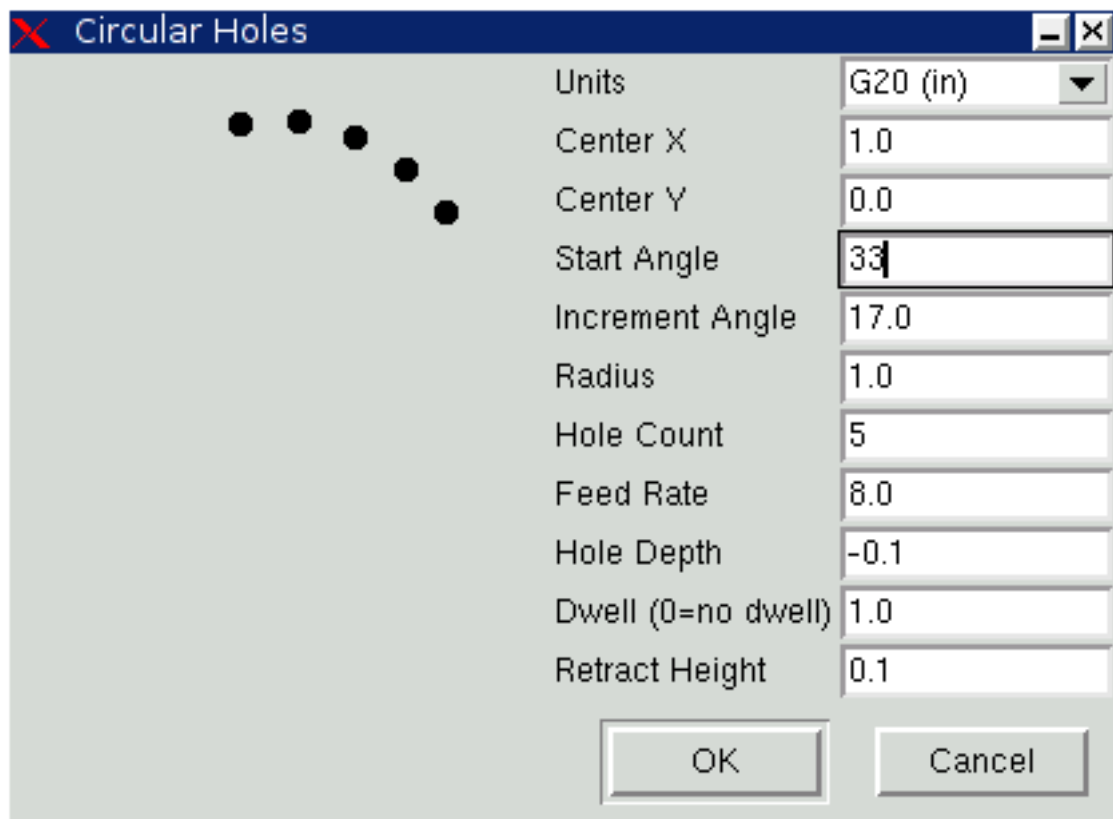


Figure 4.8: Circular Holes

If the environment variable *AXIS_PROGRESS_BAR* is set, then lines written to stderr of the form

```
FILTER_PROGRESS=%d
```

will set the *AXIS* progress bar to the given percentage. This feature should be used by any filter that runs for a long time.

4.11.2 The X Resource Database

The colors of most elements of the *AXIS* user interface can be customized through the X Resource Database. The sample file *axis_light_background* changes the colors of the backplot window to a *dark lines on white background* scheme, and also serves

as a reference for the configurable items in the display area. The sample file *axis_big_dro* changes the position readout to a larger size font. To use these files:

```
xrdb -merge /usr/share/doc/emc2/axis_light_background
xrdb -merge /usr/share/doc/emc2/axis_big_dro
```

For information about the other items which can be configured in Tk applications, see the Tk man pages.

Because modern desktop environments automatically make some settings in the X Resource Database that adversely affect AXIS, by default these settings are ignored. To make the X Resource Database items override AXIS defaults, include the following line in your X Resources:

```
*Axis*optionLevel: widgetDefault
```

this causes the built-in options to be created at the option level *widgetDefault*, so that X Resources (which are level *userDefault*) can override them.

4.11.3 Physical jog wheels

To improve the interaction of AXIS with physical jog wheels, the axis currently selected in the GUI is also reported on a pin with a name like *axisui.jog.x*. One of these pins is *TRUE* at one time, and the rest are *FALSE*. These are meant to control motion's jog-enable pins.

After AXIS has created these HAL pins, it executes the HAL file named in *[HAL]POSTGUI_HALFILE*. Unlike *[HAL]HALFILE*, only one such file may be used.

4.11.4 ~/.axisrc

If it exists, the contents of *~/.axisrc* are executed as Python source code just before the AXIS GUI is displayed. The details of what may be written in the *axisrc* are subject to change during the development cycle.

The following adds Control-Q as a keyboard shortcut for Quit.

```
root_window.bind("<Control-q>", "destroy .")
help2.append(("Control-Q", "Quit"))
```

4.11.5 External Editor

The menu options File > Edit... and File > Edit Tool Table... become available after defining the editor in the ini section *[DISPLAY]*. Useful values include *EDITOR=gedit* and *EDITOR=gnome-terminal -e vim*. For more information, see the *DISPLAY* section of the INI Configuration Chapter in the Integrator Manual.

4.11.6 Virtual Control Panel

AXIS can display a custom virtual control panel in the right-hand pane. You can program buttons, indicators, data displays and more. For more information, see the Integrator Manual.

4.11.7 Axis Preview Control

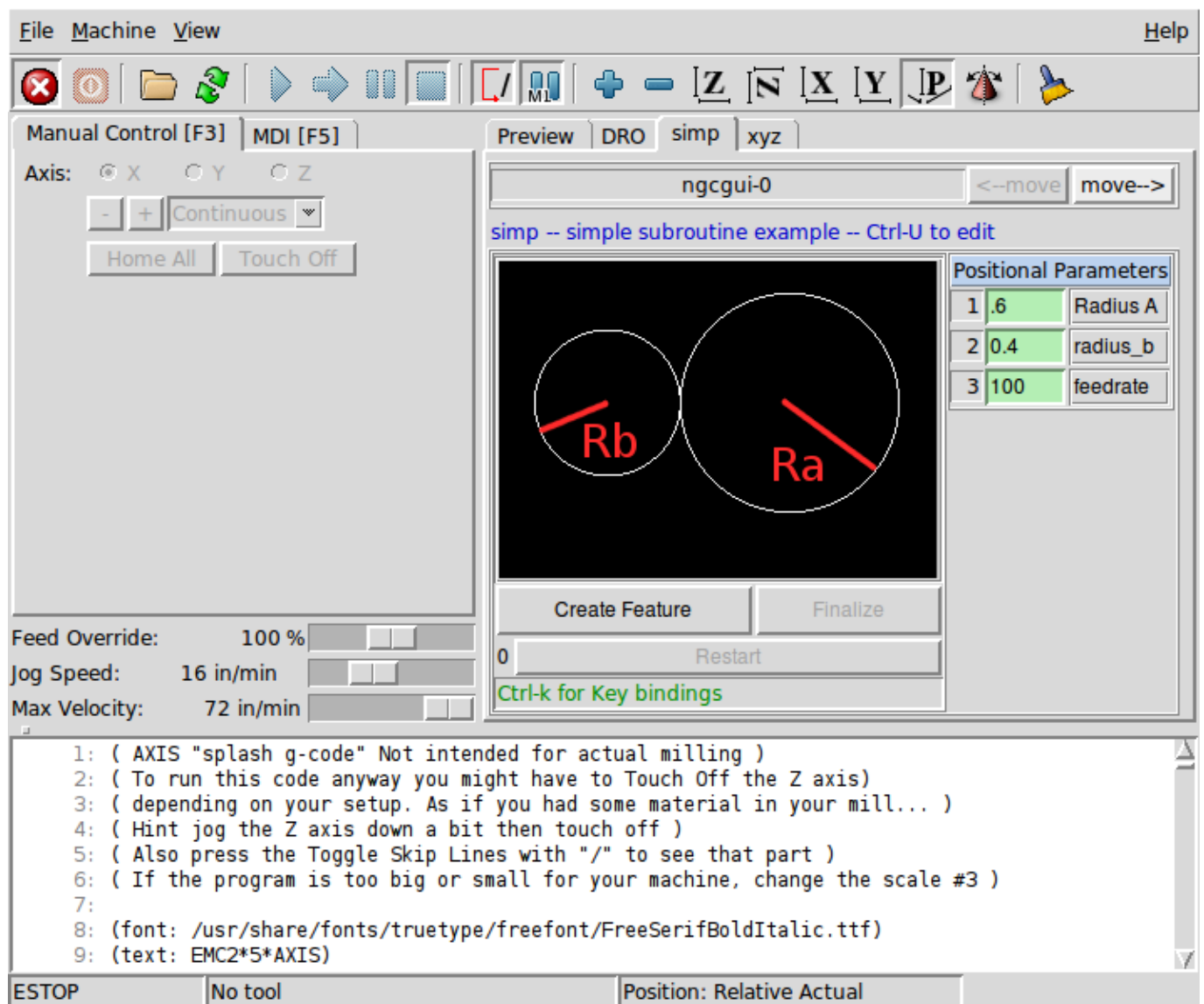
Special comments can be inserted into the G Code file to control how the preview of AXIS behaves. In the case where you want to limit the drawing of the preview use these special comments. Anything between the (AXIS,hide) and (AXIS,show) will not be drawn during the preview. The (AXIS,hide) and (AXIS,show) must be used in pairs with the (AXIS,hide) being first. Anything after a (AXIS,stop) will not be drawn during the preview.

These comments are useful to unclutter the preview display (for instance while debugging a larger g-code file, one can disable the preview on certain parts that are already working OK).

- (AXIS,hide) Stops the preview (must be first)
 - (AXIS,show) Resumes the preview (must follow a hide)
 - (AXIS,stop) Stops the preview from here to the end of the file.
 - (AXIS,notify,the_text) Displays the_text as an info display This display can be useful in the Axis preview when (debug,message) comments are not displayed.
-

Chapter 5

NGCGUI



5.1 Overview




- *NGCGUI* is a utility for using LinuxCNC subroutines.
- *NGCGUI* can run as a standalone application or be embedded in multiple tab pages in the axis gui
- Multiple copies of the same subroutine can be created
- Subroutines can be concatenated together to form a complete multiple step program
- New subroutines can be added on the fly

NGCGUI is a powerful tool for building g-code programs from subroutines on the fly. Subroutines can be concatenated to build a complete program. Multiple instances of a subroutine can be used to perform the same task in different locations on the part. Any valid g-code can be used in the subroutine.

5.2 Demo Configs

Three demo configurations are located in the sim directory of the LinuxCNC configuration picker. The configuration picker is on the main menu Applications > CNC > LinuxCNC.

- *ngcgui* - a comprehensive example that contains these subroutines
 - *simp* - a simple subroutine example that creates two circles
 - *xyz* - creates a box based on two opposite corners
 - *iquad* - creates an internal quadrilateral
 - *db25* - creates a DB25 plug cutout
 - *ihex* - creates an internal hexagon
 - *gosper* - a recursion demo
 - *Custom* - load other ngcgui-compatible subfiles
 - *ttt* - True Type Tracer creates text for engraving
- *ngcgui-lathe* - an example with lathe subroutines
 - *id* - bores the inside diameter
 - *od* - turns the outside diameter
 - *taper-od* - turns a taper on the outside diameter
 - *Custom* - creates custom tabs
- *ngcgui-simple* - a simple example
 - *simp* - a simple subroutine example that creates two circles
 - *xyz* - creates a box based on two opposite corners

To view the demonstration subroutines press the *E-Stop*  then *Machine Power*  then *Home All*. Pick a ngcgui tab and press *Create Feature* then *Finalize*. Now press the *Run*  button to watch it run.

Note

The demonstration subroutines should run on the simulated machine configurations included in the distribution. A user should always understand the behavior and purpose of a program before running on a real machine.

5.3 Libraries

The simulation configs for ngcgui use links to non-user-writable LinuxCNC libraries for:

- *ngcgui-compatible subfiles* - ngcgui_lib
- *Helper subroutines* - ngcgui_lib/utilitysubs
- *User M files* - ngcgui_lib/mfiles

These libraries are defined by the ini file items:

```
[RS274NGC]
SUBROUTINE_PATH = ../../../../nc_files/ngcgui_lib:../../../../nc_files/ngcgui_lib/utilitysubs
USER_M_PATH      = ../../../../nc_files/ngcgui_lib/mfiles
```

Note

These are long lines (not continued on multiple lines) that specify the directories used in a search patch. The directory names are separated by colons (:)

A user can create new directories for their own subroutines and M-files and add them to the search path(s).

For example, a user could create directories from the terminal.

```
mkdir /home/myusername/mysubs
mkdir /home/myusername/mymfiles
```

And then create or copy files to these user-writable directories. For instance, a user might create a ngcgui-compatible subfile named:

```
/home/myusername/mysubs/example.ngc
```

The ini file must be edited to include new subfiles and to augment the path(s). For this example:

```
[RS274NGC]
...
SUBROUTINE_PATH = /home/myusername/mysubs:../../../../nc_files/ngcgui_lib:../../../../nc_files/ ↔
                  ngcgui_lib/utilitysubs
USER_M_PATH      = /home/myusername/mymfiles:../../../../nc_files/ngcgui_lib/mfiles

[DISPLAY]
NGCGUI_SUBFILE = example.ngc
```

LinuxCNC and ngcgui use the first file found when searching directories in a search path. With this behavior, you can supersede an ngcgui_lib subfile by placing a subfile with an identical name in a directory that is found earlier in the path search. More information can be found in the INI chapter of the Integrators Manual.

5.4 Embedding NGCGUI in Axis

Several NGCGUI examples are included with LinuxCNC and are located in the sim/ngcgui directory.

5.4.1 INI File

The following INI file items for NGCGUI go in the [DISPLAY] section.

- *TKPKG = Ngcgui 1.0* - the main NGCGUI package (must precede Ngcguiittt)
- *TKPKG = Ngcguiittt 1.0* - the True Type Tracer package for generating text for engraving.
- *NGCGUI_FONT = Helvetica -12 normal* - specifies the font
- *NGCGUI_PREAMBLE = in_std.ngc* - the preamble file to be added in front of the subroutines. When concatenating several subroutines this is only added once.
- *NGCGUI_SUBFILE = simp.ngc* - creates a tab from the named subroutine
- *NGCGUI_SUBFILE = ""* - creates a custom tab
- *NGCGUI_OPTIONS = opt1 opt2 ...* - NGCGUI options
 - *nonew* - disallow making a new custom tab
 - *noremove* - disallow removing any tab page
 - *noauto* - no auto send (makeFile, then manually send)
 - *noiframe* - no internal image, image on separate top level
- *TTT = truetype-tracer* - the truetype tracer program
- *TTT_PREAMBLE = in_std.ngc* - Optional, specifies filename for preamble used for ttt created subfiles

This is an example of embedding NGCGUI into Axis. The subroutines need to be in a directory specified by the [RS274NGC]SUBROUTINE_PATH. Some example subroutines use other subroutines so check to be sure you have the dependences, if any, in a SUBROUTINE_PATH directory. Some subroutines may use custom Mfiles which must be in a directory specified by the [RS274NGC]USER_M_PATH.

Note

This is not a complete INI the items show are what is used by ngcgui. Many additional items are required by LinuxCNC to have a complete INI file.

Sample INI

```
[RS274NGC]
...
SUBROUTINE_PATH  = ../../../../nc_files/ngcgui_lib:../../../../ngcgui_lib/utilitysubs
USER_M_PATH      = ../../../../nc_files/ngcgui_lib/mfiles

[DISPLAY]
TKPKG            = Ngcgui      1.0
TKPKG            = Ngcguiittt 1.0
# Ngcgui must precede Ngcguiittt

NGCGUI_FONT      = Helvetica -12 normal
# specify filenames only, files must be in [RS274NGC]SUBROUTINE_PATH
NGCGUI_PREAMBLE  = in_std.ngc
NGCGUI_SUBFILE   = simp.ngc
NGCGUI_SUBFILE   = xyz.ngc
NGCGUI_SUBFILE   = iquad.ngc
NGCGUI_SUBFILE   = db25.ngc
NGCGUI_SUBFILE   = ihex.ngc
NGCGUI_SUBFILE   = gosper.ngc
# specify "" for a custom tab page
NGCGUI_SUBFILE   = ""
```

```
#NGCGUI_SUBFILE    = "" use when image frame is specified if
#                   opening other files is required
#                   images will be put in a top level window
NGCGUI_OPTIONS     =
#NGCGUI_OPTIONS    = opt1 opt2 ...
# opt items:
#   nonew          -- disallow making a new custom tab
#   noremove       -- disallow removing any tab page
#   noauto         -- no auto send (makeFile, then manually send)
#   noiframe       -- no internal image, image on separate top level

TTT                = truetype-tracer
TTT_PREAMBLE       = in_std.ngc

PROGRAM_PREFIX     = ../../nc_files
```

5.4.2 Truetype Tracer

Ngcgui_ttt provides support for truetype-tracer (v4). It creates an axis tab page which allows a user to create a new ngcgui tab page after entering text and selecting a font and other parameters. (Truetype-tracer must be installed independently).

To embed ngcgui_ttt in axis, specify the following items in addition to ngcgui items:

Item: [DISPLAY]TKPKG = Ngcgui_ttt version_number
 Example: [DISPLAY]TKPKG = Ngcgui_ttt 1.0
 Note: Mandatory, specifies loading of ngcgui_ttt in an axis tab page named ttt.
 Must follow the TKPKG = Ngcgui item.

Item: [DISPLAY]TTT = path_to_truetype-tracer
 Example: [DISPLAY]TTT = truetype-tracer
 Note: Optional, if not specified, attempt to use /usr/local/bin/truetype-tracer.
 Specify with absolute pathname or as a simple executable name
 in which case the user PATH environment will used to find the program.

Item: [DISPLAY]TTT_PREAMBLE = preamble_filename
 Example: [DISPLAY]TTT_PREAMBLE = in_std.ngc
 Note: Optional, specifies filename for preamble used for ttt created subfiles.

5.4.3 INI Examples

Ngcgui uses the EMC search path to find files.

The search path begins with the standard directory specified by:

```
[DISPLAY]PROGRAM_PREFIX
```

followed by multiple directories specified by:

```
[RS274NGC]SUBROUTINE_PATH
```

Directories Directories may be specified as absolute paths or relative paths.

Example: [DISPLAY]PROGRAM_PREFIX = /home/myname/emc2/nc_files
 Example: [DISPLAY]PROGRAM_PREFIX = ~/emc2/nc_files
 Example: [DISPLAY]PROGRAM_PREFIX = ../../../../nc_files

An absolute path beginning with a "/" specifies a complete filesystem location. A path beginning with a "~/ " specifies a path starting from the user's home directory. A path beginning with "~username/" specifies a path starting in username's home directory.

Relative Paths Relative paths are based on the startup directory which is the directory containing the ini file. Using relative paths can facilitate relocation of configurations but requires a good understanding of linux path specifiers.

```
./d0          is the same as d0, e.g., a directory named d0 in the startup directory
../d1         refers to a directory d1 in the parent directory
../../d2      refers to a directory d2 in the parent of the parent directory
../../../d3   etc.
```

Multiple directories can be specified with [RS274NGC]SUBROUTINE_PATH by separating them with colons. The following example illustrates the format for multiple directories and shows the use of relative and absolute paths.

Example: [RS274NGC]SUBROUTINE_PATH = ../../../../nc_files/ngcgui_lib:../../../../nc_files/ngcgui

This is one long line, do not continue on multiple lines. When emc and/or ngcgui searches for files, the first file found in the search is used.

EMC (and NGCGUI) must be able to find all subroutines including helper routines that are called from within NGCGUI subfiles. It is convenient to place utility subs in a separate directory as indicated in the example above.

The distribution includes the ngcgui_lib directory and demo files for preambles, subfiles, postambles and helper files. To modify the behavior of the files, you can copy any file and place it in an earlier part of the search path. The first directory searched is [DISPLAY]PROGRAM_PREFIX. You can use this directory but it is better practice to create dedicated directory(ies) and put them at the beginning of the [RS274NGC]SUBROUTINE_PATH.

In the following example, files in /home/myname/emc2/mysubs will be found before files in ../../../../nc_files/ngcgui_lib.

Example: [RS274NGC]SUBROUTINE_PATH = /home/myname/emc2/mysubs:../../../../nc_files/ngcgui_li

New users may inadvertently try to use files that are not structured to be compatible with ngcgui requirements. Ngcgui will likely report numerous errors if the files are not coded per its conventions. Good practice suggests that ngcgui-compatible subfiles should be placed in a directory dedicated to that purpose and that preamble, postamble, and helper files should be in separate directory(ies) to discourage attempts to use them as subfiles. Files not intended for use as subfiles can include a special comment: "(not_a_subfile)" so that ngcgui will reject them automatically with a relevant message.

To embed ngcgui in axis, specify the following items in the inifile:

```
Item:      [DISPLAY]PROGRAM_PREFIX = dirname
Example:   [DISPLAY]PROGRAM_PREFIX = ../../../../nc_files
Note:      Mandatory and needed for numerous emc functions
           It is the first directory used in the search for files
```

```
Item:      [RS274NGC]SUBROUTINE_PATH = dirname1:dirname2:dirname3 ...
Example:   [RS274NGC]SUBROUTINE_PATH = ../../../../nc_files/ngcgui_lib:../../../../nc_files/ngcgui
Note:      Optional, but very useful to organize subfiles and utility files
```

```
Item:      [DISPLAY]TKPKG=Ngcgui version_number
Example:   [DISPLAY]TKPKG=Ngcgui 1.0
Note:      Mandatory, specifies loading of ngcgui axis tab pages
```

```
Item:      [DISPLAY]NGCGUI_FONT = font_descriptor
Example:   [DISPLAY]NGCGUI_FONT = Helvetica -12 normal
Note:      Optional, font_descriptor is a tcl-compatible font specifier
           with items for fonttype -fontsize fontweight
           Default is: Helvetica -10 normal
```

```
Item:      [DISPLAY]NGCGUI_SUBFILE = subfile_filename
Example:   [DISPLAY]NGCGUI_SUBFILE = simp.ngc
```

Example: [DISPLAY]NGCGUI_SUBFILE = xyz.ngc

Example: [DISPLAY]NGCGUI_SUBFILE = ""

Note: Use one or more items to specify ngcgui-compatible subfiles that require an axis tab page on startup. A "Custom" tab will be created when the filename is "". A user can use a "Custom" tab to browse the file system and identify preamble, subfile, and postamble files.

Item: [DISPLAY]NGCGUI_PREAMBLE = preamble_filename

Example: [DISPLAY]NGCGUI_PREAMBLE = in_std.ngc

Note: Optional, when specified, the file is prepended to all subfiles. Files created with "Custom" tab pages use the preamble specified with the page.

Item: [DISPLAY]NGCGUI_POSTAMBLE = postamble_filename

Example: [DISPLAY]NGCGUI_POSTAMBLE = bye.ngc

Note: Optional, when specified, the file is appended to all subfiles. Files created with "Custom" tab pages use the postamble specified with the page.

Item: [DISPLAY]NGCGUI_OPTIONS = opt1 opt2 ...

Example: [DISPLAY]NGCGUI_OPTIONS = nonew noremove

Note: Multiple options are separated by blanks. By default, ngcgui configures tab pages so that:

- 1) a user can make new tabs
- 2) a user can remove tabs (except for the last remaining one)
- 3) finalized files are automatically sent to axis
- 4) an image frame (iframe) is made available to display an image for the subfile

The options nonew, noremove, noauto, noiframe respectively disable these default behaviors.

By default, if an image (.png,.gif,jpg,pgm) file is found in the same directory as the subfile, the image is displayed in the iframe. Specifying the noiframe option makes available additional buttons for selecting a preamble, subfile, and postamble and additional checkboxes. Selections of the checkboxes are always available with special keys:

Ctrl-R Toggle "Retain values on Subfile read"

Ctrl-E Toggle "Expand subroutine"

Ctrl-a Toggle "Autosend"

(Ctrl-k lists all keys and functions)

If noiframe is specified and an image file is found, the image is displayed in a separate window and all functions are available on the tab page.

The NGCGUI_OPTIONS apply to all ngcgui tabs except that the nonew, noremove, and noiframe options are not applicable for "Custom" tabs. Do not use "Custom" tabs if you want to limit the user's ability to select subfiles or create additional tab pages.

5.5 Subroutine Requirements

An NGCGUI-compatible subfile contains a single subroutine definition. The name of the subroutine must be the same as the filename (not including the .ngc suffix). LinuxCNC supports named or numbered subroutines, but only named subroutines are compatible with NGCGUI. For more information see the [O-Codes](#) Chapter.

The first non-comment line should be a sub statement. The last non-comment line should be a endsub statement.

examp.ngc:

```
o<examp> sub
    BODY_OF_SUBROUTINE
o<examp> endsub
```

The body of the subroutine should begin with a set of statements that define local named parameters for each positional parameter expected for the subroutine call. These definitions must be consecutive beginning with #1 and ending with the last used parameter number. Definitions must be provided for each of these parameters (no omissions).

Parameter Numbering

```
#<xparm> = #1
#<yparm> = #2
#<zparm> = #3
```

LinuxCNC considers all numbered parameters in the range #1 thru #30 to be calling parameters so ngcgui provides entry boxes for any occurrence of parameters in this range. It is good practice to avoid use of numbered parameters #1 through #30 anywhere else in the subroutine. Using local, named parameters is recommended for all internal variables.

Each defining statement may optionally include a special comment and a default value for the parameter.

Statement Prototype

```
#<vname> = #n (=default_value)
or
#<vname> = #n (comment_text)
or
#<vname> = #n (=default_value comment_text)
```

Parameter Examples

```
#<xparm> = #1 (=0.0)
#<yparm> = #2 (Ystart)
#<zparm> = #3 (=0.0 Z start setting)
```

If a default_value is provided, it will be entered in the entry box for the parameter on startup.

If comment_text is included, it will be used to identify the input instead of the parameter name.

Global Named Parameters

Notes on global named parameters (#<_globalname>) and ngcgui:

As in many programming languages, use of globals is powerful but can often lead to unexpected consequences. In LinuxCNC, existing global named parameters will be valid at subroutine execution and subroutines can modify or create global named parameters.

The use of global named parameters as inputs to subroutines is discouraged because such usage requires the establishment and maintenance of a well-defined global context that is problematic to maintain. Using numbered parameters #1 thru #30 as subroutine inputs should be sufficient to satisfy a wide range of design requirements.

Declining Feature: Ngcgui includes some support for global named input parameters but usage is deprecated and not documented here. Use of global named input parameters will not be supported with linuxCNC2.6.

While input global named parameters are discouraged, emc subroutines must use global named parameters for returning results. Since ngcgui-compatible subfiles are aimed at gui usage, return values are not a common requirement. However, ngcgui is useful

as a testing tool for subroutines which do return global named parameters and it is common for ngcgui-compatible subfiles to call utility subroutine files that return results with global named parameters.

To support these usages, ngcgui ignores global named parameters that include a colon (:) character in their name. Use of the colon (:) in the name prevents ngcgui from making entryboxes for these parameters.

Global Named Parameters

```
o<examp> sub
...
#<_examp:result> = #5410      (return the current tool diameter)
...
o<helper> call [#<x1>] [#<x2>] (call a subroutine)
#<xresult> = #<_helper:answer> (localize immediately the helper result)
#<_helper:answer> = 0.0      (nullify global named parameter used by subroutine)
...
o<examp> endsup
```

In the above example, the utility subroutine will be found in a separate file named helper.ngc. The helper routine returns a result in a global named parameter named #<_helper:answer.

For good practice, the calling subfile immediately localizes the result for use elsewhere in the subfile and the global named parameter used for returning the result is nullified in an attempt to mitigate its inadvertent use elsewhere in the global context. (A nullification value of 0.0 may not always be a good choice).

Ngcgui supports the creation and concatenation of multiple features for a subfile and for multiple subfiles. It is sometimes useful for subfiles to determine their order at runtime so ngcgui inserts a special global parameter that can be tested within subroutines. The parameter is named #<_feature:>. Its value begins with a value of 0 and is incremented for each added feature.

Additional Features A special *info* comment can be included anywhere in an ngcgui-compatible subfile. The format is:

```
(info: info_text)
```

The info_text is displayed near the top of the ngcgui tab page in axis.

Files not intended for use as subfiles can include a special comment so that ngcgui will reject them automatically with a relevant message.

```
(not_a_subfile)
```

An optional image file (.png,.gif,.jpg,.pgm) can accompany a subfile. The image file can help clarify the parameters used by the subfile. The image file should be in the same directory as the subfile and have the same name with an appropriate image suffix, e.g. the subfile examp.ngc could be accompanied by an image file examp.png. Ngcgui attempts to resize large images by subsampling to a size with maximum width of 320 and maximum height of 240 pixels.

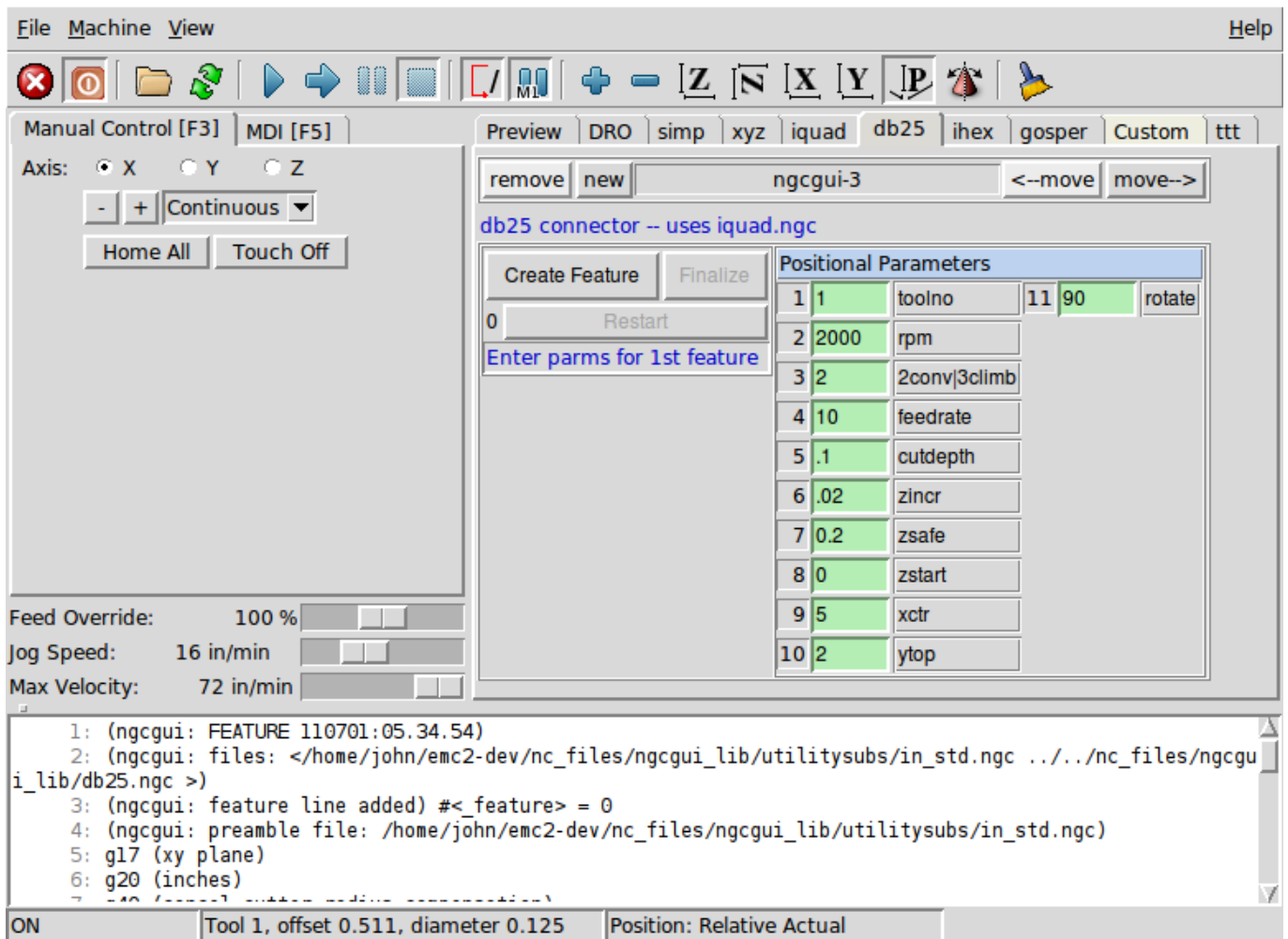
None of the conventions required for making an ngcgui-compatible subfile preclude its use as general purpose subroutine file for LinuxCNC.

The LinuxCNC distribution includes a library (ngcgui_lib directory) that includes both example ngcgui-compatible subfiles and utility files to illustrate the features of LinuxCNC subroutines and ngcgui usage.

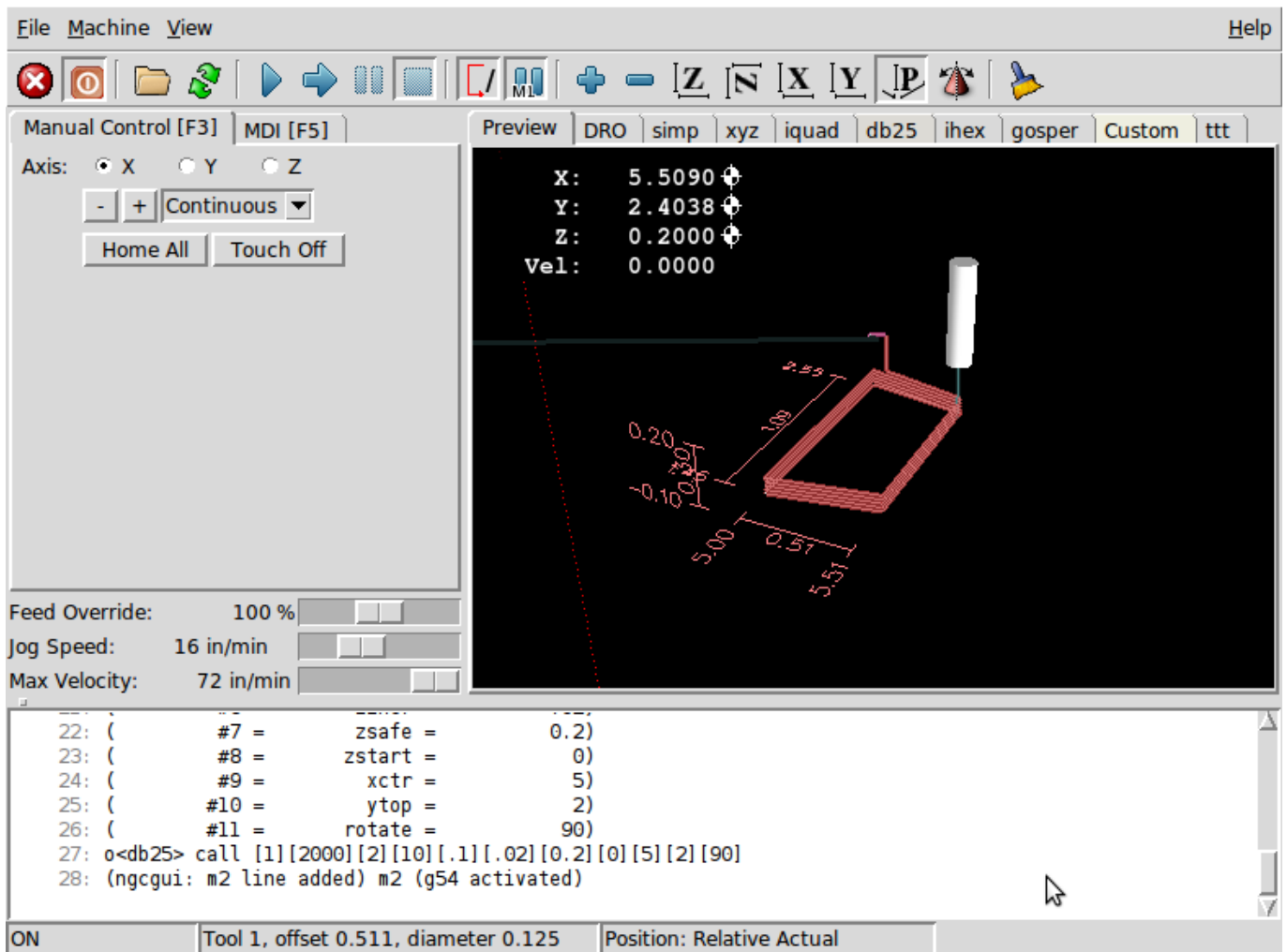
Additional user submitted subroutines can be found on the Forum in the Subroutines Section.

5.6 DB25 Example

The following shows the DB25 subroutine. In the first photo you see where you fill in the blanks for each variable.



This photo shows the backplot of the DB25 subroutine.



This photo shows the use of the new button and the custom tab to create three DB25 cutouts in one program.

Chapter 6

Touchy GUI

Touchy is a user interface for LinuxCNC meant for use on machine control panels, and therefore does not require keyboard or mouse.

It is meant to be used with a touch screen, and works in combination with a wheel/MPG and a few buttons and switches.

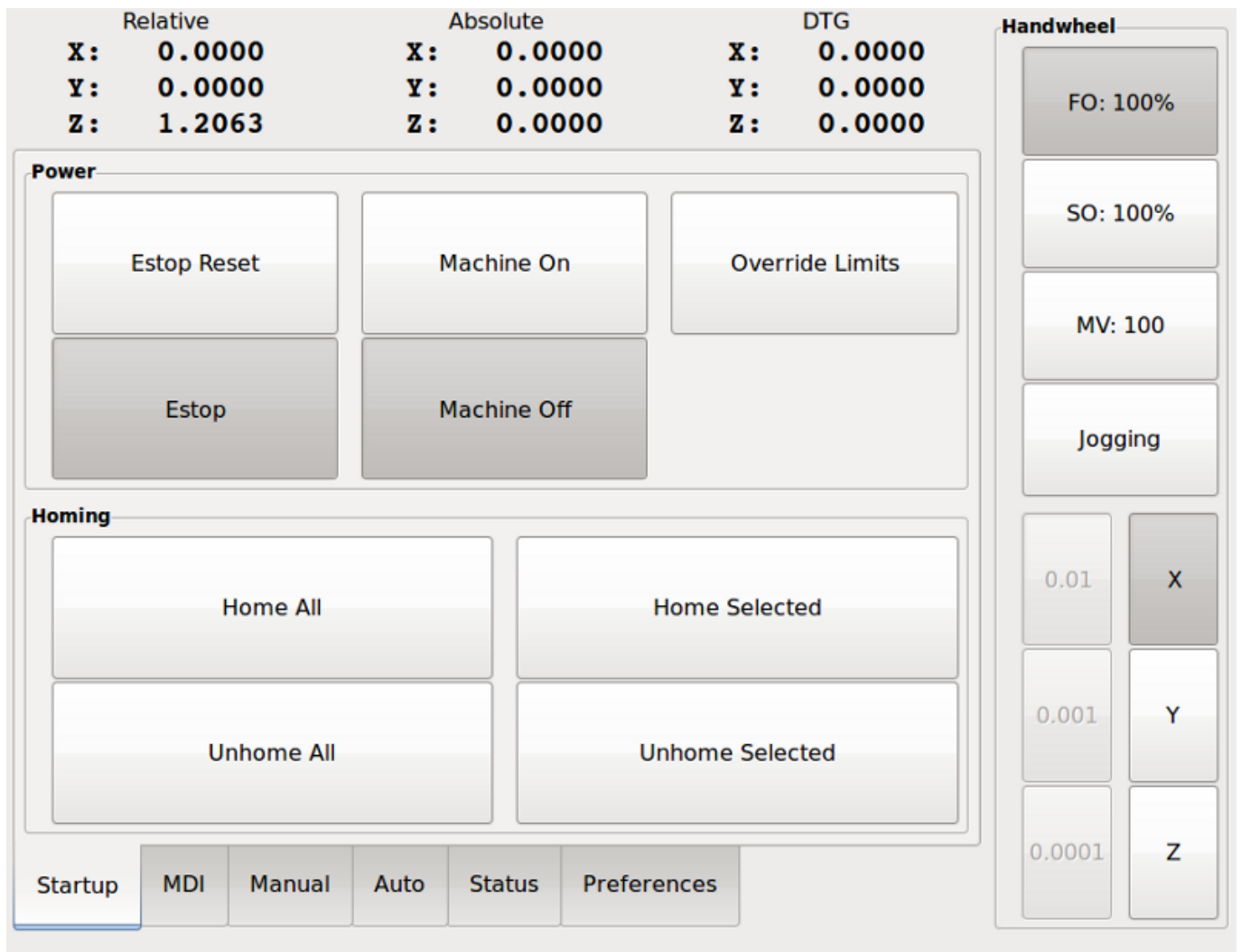


Figure 6.1: Touchy

6.1 Panel Configuration

6.1.1 HAL connections

Touchy requires that you create a file named *touchy.hal* in your configuration directory (the directory your ini file is in) to connect its controls. Touchy executes the HAL commands in this file after it has made its own pins available for connection.

Touchy has several output pins that are meant to be connected to the motion controller to control wheel jogging:

- *touchy.jog.wheel.increment*, which is to be connected to the *axis.N.jog-scale* pin of each axis N.
- *touchy.jog.wheel.N*, which is to be connected to *axis.N.jog-enable* for each axis N.
- In addition to being connected to *touchy.wheel-counts*, the wheel counts should also be connected to *axis.N.jog-counts* for each axis N. If you use HAL component *ilowpass* to smooth wheel jogging, be sure to smooth only *axis.N.jog-counts* and not *touchy.wheel-counts*.

6.1.1.1 Required controls

- Abort button (momentary contact) connected to the HAL pin *touchy.abort*
- Cycle start button (momentary contact) connected to *touchy.cycle-start*
- Wheel/MPG, connected to *touchy.wheel-counts* and motion pins as described above
- Single block (toggle switch) connected to *touchy.single-block*

6.1.1.2 Optional controls

- For continuous jog, one center-off bidirectional momentary toggle (or two momentary buttons) for each axis, hooked to *touchy.jog.continuous.x.negative*, *touchy.jog.continuous.x.positive*, etc.
- If a quill up button is wanted (to jog Z to the top of travel at top speed), a momentary button connected to *touchy.quill-up*.

6.1.1.3 Optional panel lamps

- *touchy.jog.active* shows when the panel jogging controls are live
- *touchy.status-indicator* is on when the machine is executing G-code, and flashes when the machine is executing but is in pause/feedhold.

6.1.2 Recommended for any setup

- Estop button hardwired in the estop chain

6.2 Setup

6.2.1 Enabling Touchy

To use Touchy, in the *[DISPLAY]* section of your ini file change the display selector line to *DISPLAY = touchy*

6.2.2 Preferences

When you start Touchy the first time, check the Preferences tab. If using a touchscreen, choose the option to hide the pointer for best results.

The Status Window is a fixed height, set by the size of a fixed font. This can be affected by the Gnome DPI, configured in System / Preferences / Appearance / Fonts / Details. If the bottom of the screen is cut off, reduce the DPI setting.

All other font sizes can be changed on the Preferences tab.

6.2.3 Macros

Touchy can invoke O-word macros using the MDI interface. To configure this, in the *[TOUCHY]* section of the ini file, add one or more *MACRO* lines. Each should be of the format

MACRO=increment xinc yinc

In this example, increment is the name of the macro, and it accepts two parameters, named xinc and yinc.

Now, place the macro in a file named *increment.ngc*, in the *PROGRAM_PREFIX* directory or any directory in the *SUBROUTINE_PATH*.

It should look like:

```
O<increment> sub
G91 G0 X#1 Y#2
G90
O<increment> endsub
```

Notice the name of the sub matches the file name and macro name exactly, including case.

When you invoke the macro by pressing the Macro button on the MDI tab in Touchy, you can enter values for xinc and yinc. These are passed to the macro as *#1* and *#2* respectively. Parameters you leave empty are passed as value 0.

If there are several different macros, press the Macro button repeatedly to cycle through them.

In this simple example, if you enter -1 for xinc and press cycle start, a rapid *G0* move will be invoked, moving one unit to the left.

This macro capability is useful for edge/hole probing and other setup tasks, as well as perhaps hole milling or other simple operations that can be done from the panel without requiring specially-written gcode programs.

Chapter 7

TkLinuxCNC GUI

7.1 Introduction

TkLinuxCNC is one of the first graphical front-ends for LinuxCNC. It is written in Tcl and uses the Tk toolkit for the display. Being written in Tcl makes it very portable (it runs on a multitude of platforms). A separate backplot window can be displayed as shown.

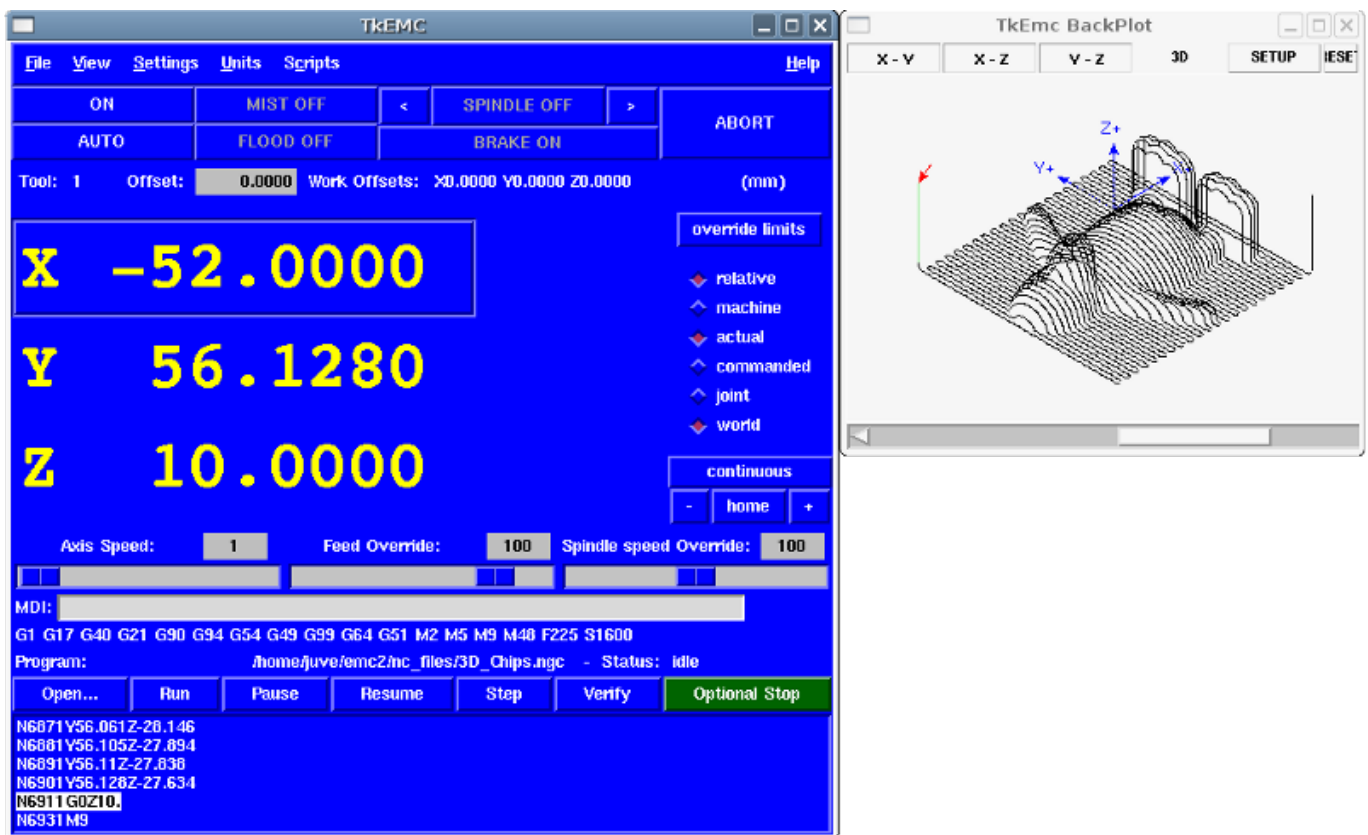


Figure 7.1: TkLinuxCNC Window

7.2 Getting Started

To select TkLinuxCNC as the front-end for LinuxCNC, edit the .ini file. In the section *[DISPLAY]* change the *DISPLAY* line to read

```
DISPLAY = tklinuxcnc
```

Then, start LinuxCNC and select that ini file. The sample configuration *sim/tklinuxcnc/tklinuxcnc.ini* is already configured to use TkLinuxCNC as its front-end.

7.2.1 A typical session with TkLinuxCNC

1. Start LinuxCNC and select a configuration file.
2. Clear the *E-STOP* condition and turn the machine on (by pressing F1 then F2).
3. *Home* each axis.
4. Load the file to be milled.
5. Put the stock to be milled on the table.
6. Set the proper offsets for each axis by jogging and either homing again or right-clicking an axis name and entering an offset value. ¹
7. Run the program.
8. To mill the same file again, return to step 6. To mill a different file, return to step 4. When you're done, exit LinuxCNC.

7.3 Elements of the TkLinuxCNC window

The TkLinuxCNC window contains the following elements:

- A menubar that allows you to perform various actions
- A set of buttons that allow you to change the current working mode, start/stop spindle and other relevant I/O
- Status bar for various offset related displays
- Coordinate display area
- A set of sliders which control *Jogging speed*, *Feed Override*, and *Spindle speed Override* which allow you to increase or decrease those settings
- Manual data input text box *MDI*
- Status bar display with active G-codes, M-codes, F- and S-words
- Interpreter related buttons
- A text display area that shows the G-code source of the loaded file

¹For some of these actions it might be necessary to change the mode LinuxCNC is currently running in.

7.3.1 Main buttons

From left to right, the buttons are:

- Machine enable: *ESTOP* > *ESTOP RESET* > *ON*
- Toggle mist coolant
- Decrease spindle speed
- Set spindle direction *SPINDLE OFF* > *SPINDLE FORWARD* . *SPINDLE REVERSE*
- Increase spindle speed
- Abort

then on the second line:

- Operation mode: *MANUAL* > *MDI* > *AUTO*
- Toggle flood coolant
- Toggle spindle brake control

7.3.2 Offset display status bar

The Offset display status bar displays the currently selected tool (selected with Txx M6), the tool length offset (if active), and the work offsets (set by right-clicking the coordinates).

7.3.3 Coordinate Display Area

The main part of the display shows the current position of the tool. The color of the position readout depends on the state of the axis. If the axis is unhomed the axis will be displayed in yellow letters. Once homed it will be displayed in green letters. If there is an error with the current axis TkLinuxCNC will use red letter to show that. (for example if an hardware limit switch is tripped).

To properly interpret these numbers, refer to the radio boxes on the right. If the position is *Machine*, then the displayed number is in the machine coordinate system. If it is *Relative*, then the displayed number is in the offset coordinate system. Further down the choices can be *actual* or *commanded*. Actual refers to the feedback coming from encoders (if you have a servo machine), and the *commanded* refers to the position command send out to the motors. These values can differ for several reasons: Following error, deadband, encoder resolution, or step size. For instance, if you command a movement to X 0.0033 on your mill, but one step of your stepper motor is 0.00125, then the *Commanded* position will be 0.0033 but the *Actual* position will be 0.0025 (2 steps) or 0.00375 (3 steps).

Another set of radio buttons allows you to choose between *joint* and *world* view. These make little sense on a normal type of machine (e.g. trivial kinematics), but help on machines with non-trivial kinematics like robots or stewart platforms. (you can read more about kinematics in the Integrator Manual).

7.3.3.1 Backplot

When the machine moves, it leaves a trail called the backplot. You can start the backplot window by selecting View→Backplot.

7.3.4 Automatic control

7.3.4.1 Buttons for control

The buttons in the lower part of TkLinuxCNC are used to control the execution of a program: *Open* to load a program, *Verify* to check it for errors, *Run* to start the actual cutting, *Pause* to stop it while running, *Resume* to resume an already paused program, *Step* to advance one line in the program and *Optional Stop* to toggle the optional stop switch (if the button is green the program execution will be stopped on any M1 encountered).

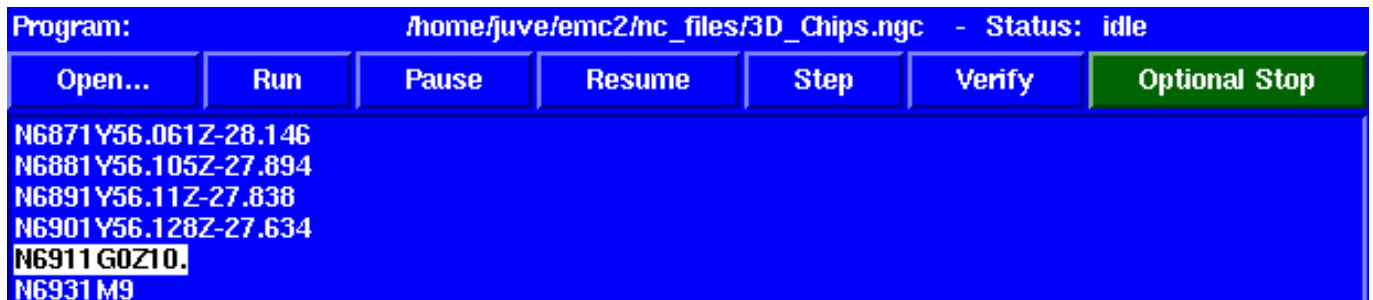


Figure 7.2: TkLinuxCNC Interpreter / program control

7.3.4.2 Text Program Display Area

When the program is running, the line currently being executed is highlighted in white. The text display will automatically scroll to show the current line.

7.3.5 Manual Control

7.3.5.1 Implicit keys

TkLinuxCNC allows you to manually move the machine. This action is known as *jogging*. First, select the axis to be moved by clicking it. Then, click and hold the + or - button depending on the desired direction of motion. The first four axes can also be moved by the keyboard arrow keys (X and Y), the PAGE UP and PAGE DOWN keys (Z) and the / and J keys (A/4th).

If *Continuous* is selected, the motion will continue as long as the button or key is pressed. If another value is selected, the machine will move exactly the displayed distance each time the button is clicked or the key is pressed. The available values are: 1.0000, 0.1000, 0.0100, 0.0010, 0.0001

By pressing *Home* or the HOME key, the selected axis will be homed. Depending on your configuration, this may just set the axis value to be the absolute position 0.0, or it may make the machine move to a specific home location through use of *home switches*. See the Integrator Manual for more information on homing.

By pressing *Override Limits*, the machine will temporarily be permitted to jog outside the limits defined in the .ini file. (Note: if *Override Limits* is active the button will be displayed using a red color).

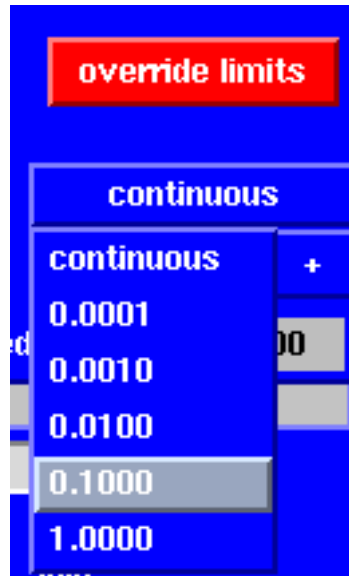


Figure 7.3: TkLinuxCNC Override Limits & Jogging increments example

7.3.5.2 The Spindle group

The button on the first row selects the direction for the spindle to rotate: Counterclockwise, Stopped, Clockwise. The buttons next to it allow the user to increase or decrease the rotation speed. The button on the second row allows the spindle brake to be engaged or released. Depending on your machine configuration, not all the items in this group may have an effect.

7.3.5.3 The Coolant group

The two buttons allow the *Mist* and *Flood* coolants to be turned on and off. Depending on your machine configuration, not all the items in this group may appear.

7.3.6 Code Entry

Manual Data Input (also called MDI), allows G-code programs to be entered manually, one line at a time. When the machine is not turned on, and not set to MDI mode, the code entry controls are unavailable.



Figure 7.4: The Code Entry tab

7.3.6.1 MDI:

This allows you to enter a g-code command to be executed. Execute the command by pressing Enter.

7.3.6.2 Active G-Codes

This shows the *modal codes* that are active in the interpreter. For instance, *G54* indicates that the *G54 offset* is applied to all coordinates that are entered.

7.3.7 Jog Speed

By moving this slider, the speed of jogs can be modified. The numbers above refer to axis units / second. The text box with the number is clickable. Once clicked a popup window will appear, allowing for a number to be entered.

7.3.8 Feed Override

By moving this slider, the programmed feed rate can be modified. For instance, if a program requests *F60* and the slider is set to 120%, then the resulting feed rate will be 72. The text box with the number is clickable. Once clicked a popup window will appear, allowing for a number to be entered.

7.3.9 Spindle speed Override

The spindle speed override slider works exactly like the feed override slider, but it controls to the spindle speed. If a program requested S500 (spindle speed 500 RPM), and the slider is set to 80%, then the resulting spindle speed will be 400 RPM. This slider has a minimum and maximum value defined in the ini file. If those are missing the slider is stuck at 100%. The text box with the number is clickable. Once clicked a popup window will appear, allowing for a number to be entered.

7.4 Keyboard Controls

Almost all actions in TkLinuxCNC can be accomplished with the keyboard. Many of the shortcuts are unavailable when in MDI mode.

The most frequently used keyboard shortcuts are shown in the following table.

Table 7.1: Most Common Keyboard Shortcuts

Keystroke	Action Taken
F1	Toggle Emergency Stop
F2	Turn machine on/off
`, 1 .. 9, 0	Set feed override from 0% to 100%
X, `	Activate first axis
Y, 1	Activate second axis
Z, 2	Activate third axis
A, 3	Activate fourth axis
Home	Send active axis Home
Left, Right	Jog first axis
Up, Down	Jog second axis
Pg Up, Pg Dn	Jog third axis
[,]	Jog fourth axis
ESC	Stop execution

Chapter 8

MINI GUI

8.1 Introduction

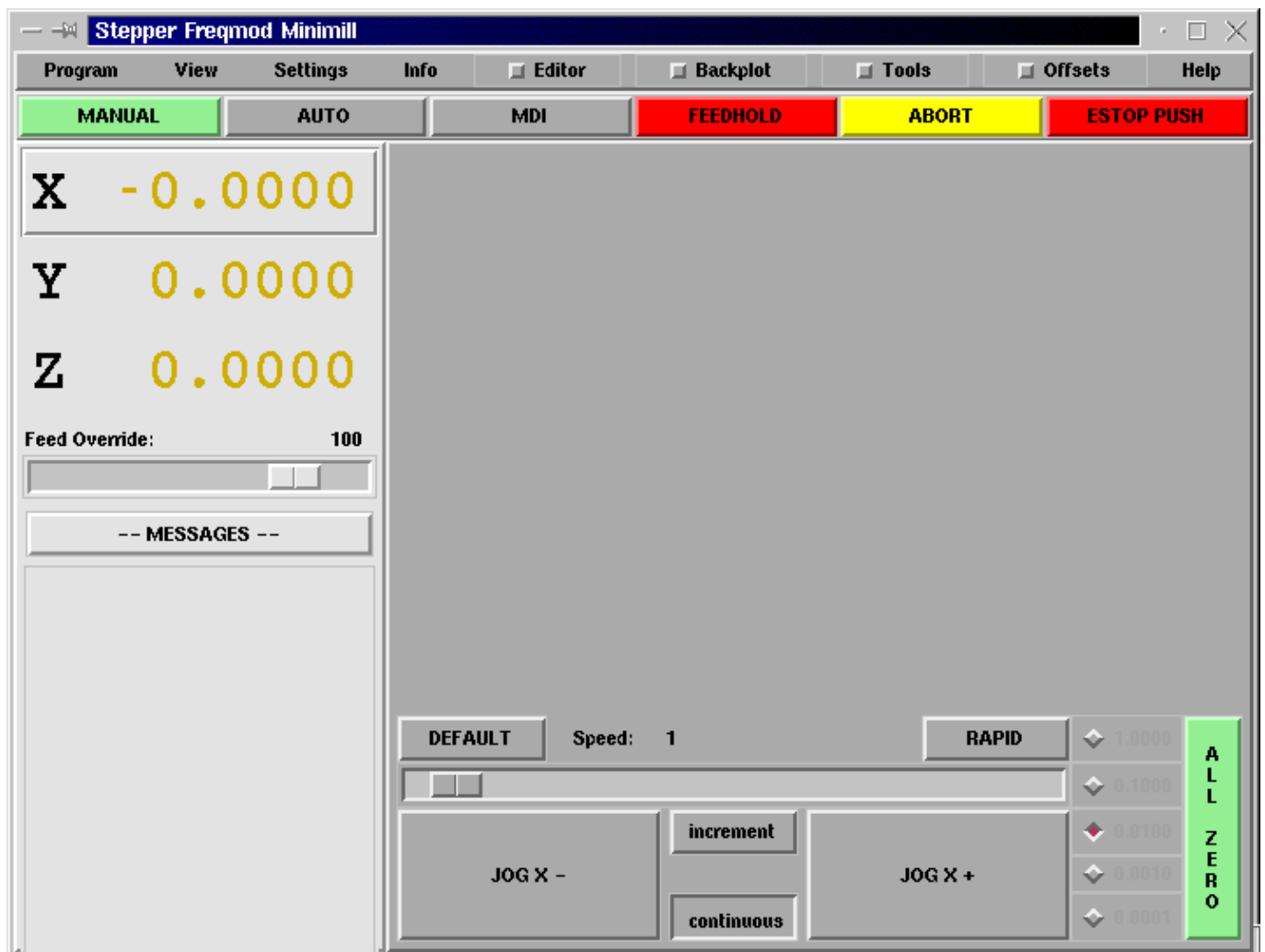


Figure 8.1: The Mini Graphical Interface (upon starting)

Mini was designed to be a full screen graphical interface. It was first written for the Sherline CNC but is available for anyone to use, copy, and distribute under the terms of the GPL copyright.

Rather than popup new windows for each thing that an operator might want to do, Mini allows you to display these within the regular screen. Parts of this chapter are copied from the instructions that were written for that mill by Joe Martin and Ray Henry.

¹

8.2 Screen layout

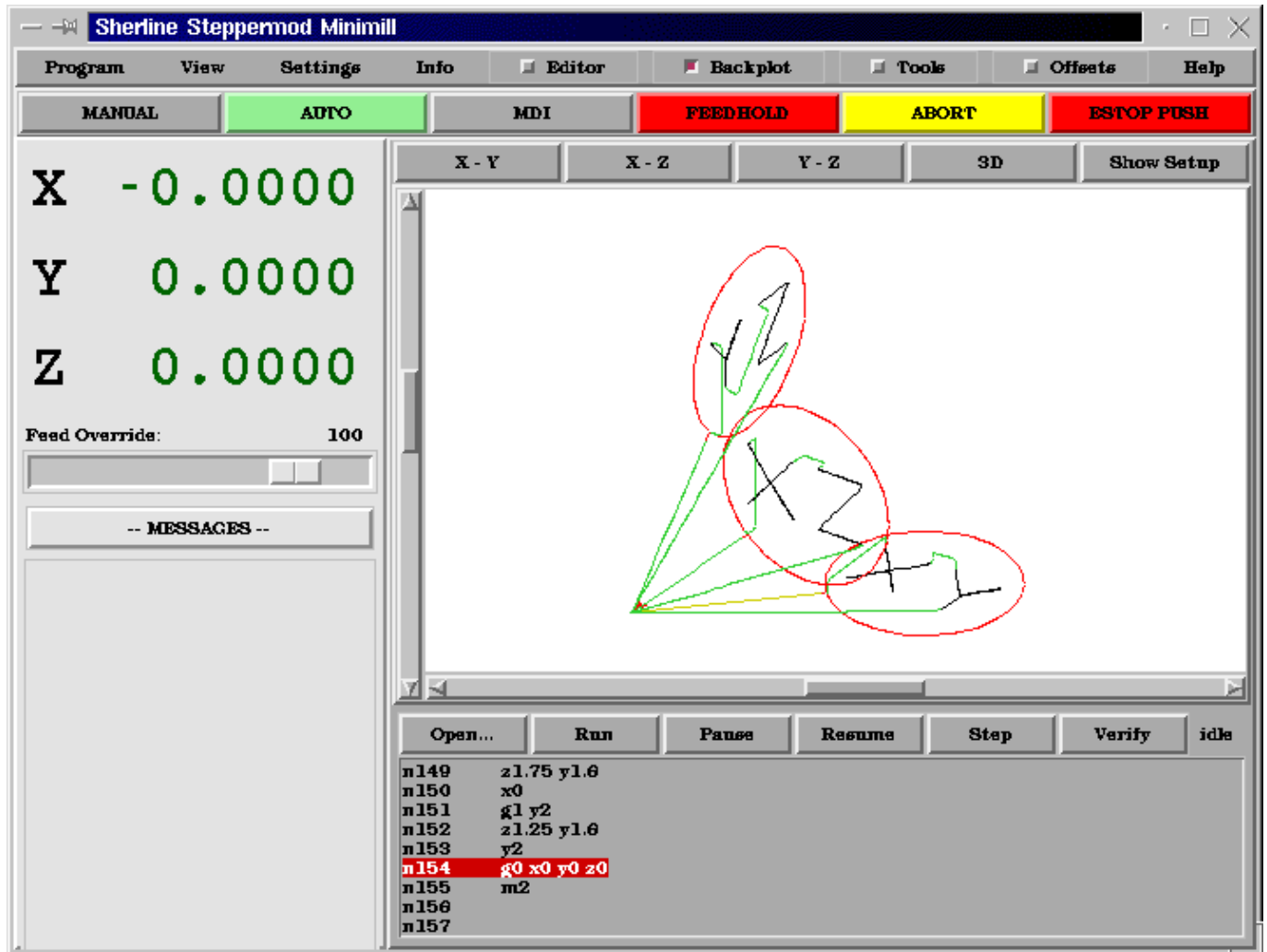


Figure 8.2: Mini Display for a Running LinuxCNC

The Mini screen is laid out in several sections. These include a menu across the top, a set of main control buttons just below the menu, and two rather large columns of information that show the state of your machine and allow you to enter commands or programs.

When you compare starting screen with run screen you will see many differences. In the second figure

- each axis has been homed — the display numbers are dark green
- the LinuxCNC mode is auto — the auto button has a light green background

¹Much of this chapter quotes from a chapter of the Sherline CNC Operators Manual.

- the backplotter has been turned on — backplot is visible in the pop-in window
- the tool path from the program is showing in the display.

Once you start working with Mini you will quickly discover how easily it shows the conditions of the LinuxCNC and allows you to make changes to it.

8.3 Menu Bar

The first row is the menu bar across the top. Here you can configure the screen to display additional information. Some of the items in this menu are very different from what you may be accustomed to with other programs. You should take a few minutes and look under each menu item in order to familiarize yourself with the features that are there.

The menu includes each of the following sections and subsections.

- *Program* - This menu includes both reset and exit functions. Reset will return the LinuxCNC to the condition that it was in when it started. Some startup configuration items like the normal program units can be specified in the ini file.
- *View* - This menu includes several screen elements that can be added so that you can see additional information during a run. These include
 - *Position_Type* - This menu item adds a line above the main position displays that shows whether the displays are in inches or metric and whether they are Machine or Relative location and if they are Actual positions or Commanded positions. These can be changed using the Settings menu described below.
 - *Tool_Info* - This adds a line immediately below the main position displays that shows which tool has been selected and the length of offset applied.
 - *Offset_Info* - adds a line immediately below the tool info that shows what offsets have been applied. This is a total distance for each axis from machine zero.
 - *Show_Restart* - adds a block of buttons to the right of the program display in auto mode. These allow the operator to restart a program after an abort or estop. These will pop in whenever estop or abort is pressed but can be shows by the operator anytime auto mode is active by selecting this menu item.
 - *Hide_Restart* - removes the block of buttons that control the restart of a program that has been aborted or estopped.
 - *Show_Split_Right* - changes the nature of the right hand column so that it shows both mode and pop-in information.
 - *Show_Mode_Full* - changes the right hand column so that the mode buttons or displays fill the entire right side of the screen. In manual mode, running with mode full you will see spindle and lube control buttons as well as the motion buttons.
 - *Show_Popin_Full* - changes the right hand column so that the popin fills the entire right side of the screen.
- *Settings* - These menu items allow the operator to control certain parameters during a run.
 - *Actual_Position* - sets the main position displays to actual(machine based) values.
 - *Commanded_Position* - sets the main position displays to the values that they were commanded to.
 - *Machine_Position* - sets the main position displays to the absolute distance from where the machine was homed.
 - *Relative_Position* - sets the main position displays to show the current position including any offsets like part zeros that are active. For more information on offsets see the chapter on coordinate systems.
- *Info* - lets you see a number of active things by writing their values into the MESSAGE pad.
 - *Program_File* - will write the currently active program file name.
 - *Editor_File* - will write the currently active file if the editor pop in is active and a file has been selected for editing.
 - *Parameter_File* - will write the name of the file being used for program parameters. You can find more on this in the chapters on offsets and using variables for programming.
 - *Tool_File* - will write the name of the tool file that is being used during this run.

- *Active_G-Codes* - will write a list of all of the modal program codes that are active whenever this item is selected. For more information about modal codes see the introductory part programming chapter.
- *Help* - opens a text window pop in that displays the contents of the help file.

You will notice between the info menu and the help menu there are a set of four buttons. These are called check buttons because they have a small box that shows red if they have been selected. These four buttons, Editor, Backplot, Tools, and Offsets pop in each of these screens. If more than one pop-in is active (button shown as red) you can toggle between these pop-ins by right clicking your mouse.

8.4 Control Button Bar

Below the menu line is a horizontal line of control buttons. These are the primary control buttons for the interface. Using these buttons you can change mode from [MANUAL] to [AUTO] to [MDI] (Manual Data Input). These buttons show a light green background whenever that mode is active.

You can also use the [FEEDHOLD], [ABORT], and [ESTOP] buttons to control a programmed move.

8.4.1 MANUAL

This button or pressing <F3> sets the LinuxCNC to Manual mode and displays an abbreviated set of buttons the operator can use to issue manual motion commands. The labels of the jog buttons change to match the active axis. Whenever Show_Mode_Full is active in in manual mode, you will see spindle and lube control buttons as well as the motion buttons. A keyboard <i> or <I> will switch from continuous jog to incremental jog. Pressing that key again will toggle the increment size through the available sizes.

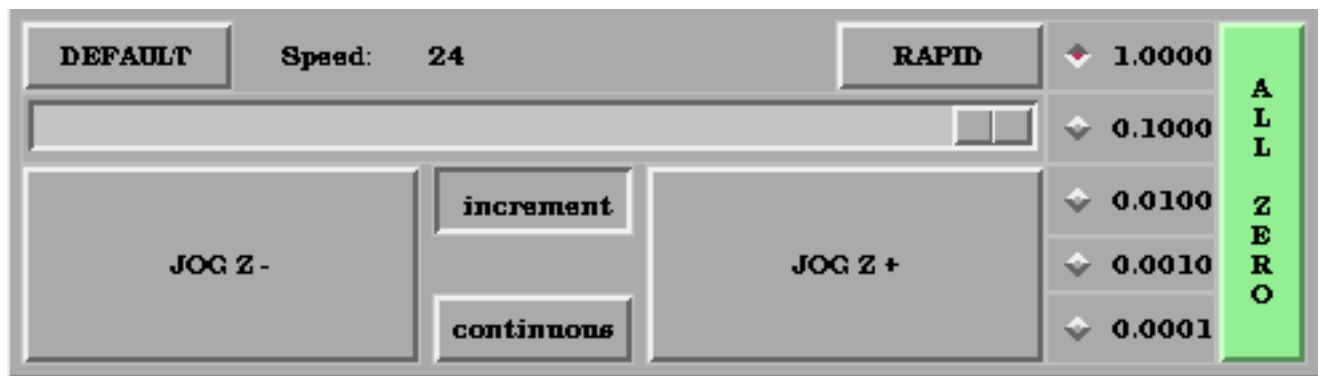


Figure 8.3: Manual Mode Buttons

From the Sherline CNC Operators Manual:

A button has been added to designate the present position as the home position. We felt that a machine of this type (Sherline 5400) would be simpler to operate if it didn't use a machine home position. This button will zero out any offsets and will home all axes right where they are.

Axis focus is important here. Notice in [startup figure](#) that in manual mode you see a line or *groove* around the X axis to highlight its position display. This groove says that X is the active axis. It will be the target for jog moves made with the *plus* and *minus* jog buttons. You can change axis focus by clicking on any other axis display. You can also change axis focus in manual mode if you press its name key on your keyboard. Case is not important here. [Y] or [y] will shift the focus to the Y axis. [A] or [a] will shift the focus to the A axis. To help you remember which axis will jog when you press the jog buttons, the active axis name is displayed on them.

LinuxCNC can jog (move a particular axis) as long as you hold the button down when it is set for *continuous*, or it can jog for a preset distance when it is set for *incremental*. You can also jog the active axis by pressing the plus [+] or minus [-] keys on the keyboard. Again, case is not important for keyboard jogs. The two small buttons between the large jog buttons let you set which kind of jog you want. When you are in incremental mode, the distance buttons come alive. You can set a distance by pressing it with the mouse. You can toggle between distances by pressing [i] or [I] on the keyboard. Incremental jog has an interesting and often unexpected effect. If you press the jog button while a jog is in progress, it will add the distance to the position it was at when the second jog command was issued. Two one-inch jog presses in close succession will not get you two inches of movement. You have to wait until the first one is complete before jogging again.

Jog speed is displayed above the slider. It can be set using the slider by clicking in the slider's open slot on the side you want it to move toward, or by clicking on the [Default] or [Rapid] buttons. This setting only affects the jog move while in manual mode. Once a jog move is initiated, jog speed has no effect on the jog. As an example of this, say you set jog mode to *incremental* and the increment to 1 inch. Once you press the [Jog] button it will travel that inch at the rate at which it started.

8.4.2 AUTO

When the Auto button is pressed, or <F4> on the keyboard, and LinuxCNC is set to that mode, a set of the traditional auto operation buttons is displayed, and a small text window opens to show a part program. During run the active line will be displayed as white lettering on a red background.

In the auto mode, many of the keyboard keys are bound to controls. For example, the numbers above the qwerty keys are bound to feed rate override. The 0 sets 100%, 9 sets 90% and such. Other keys work much the same as they do with the tkLinuxCNC graphical interface.



Figure 8.4: Auto Mode

Auto mode does not normally display the active or modal codes. If the operator wishes to check these, use menu Info→Active_G-Codes. This will write all modal codes onto the message scratch pad.

If abort or estop is pressed during a run, a set of buttons will display to the right of the text that allow the operator to shift the restart line forward or backward. If the restart line is not the last active line, it will be highlighted as white letters on a blue background. Caution, a very slow feed rate, and a finger poised over the pause button is advised during any program restart.

From the Sherline CNC Operators Manual:

The real heart of CNC machine tool work is the auto mode. Sherline's auto mode displays the typical functions that people have come to expect from LinuxCNC. Along the top are a set of buttons which control what is happening in auto mode. Below them is the window that shows the part of the program currently being executed. As the program runs, the active line shows in white letters on a red background. The first three buttons, [Open], [Run], and [Pause] do about what you'd expect. [Pause] will stop the run right where it is. The next button, [Resume], will restart motion. They are like feedhold if used this way. Once [Pause] is pressed and motion has stopped, [Step] will resume motion and continue it to the end of the current block. Press [Step] again to get the motion of the next block. Press [Resume] and the interpreter goes back to reading ahead and running the program. The combination of [Pause] and [Step] work a lot like single block mode on many controllers. The difference is that [Pause] does not let motion continue to the end of the current block. Feed rate Override ... can be very handy as you approach a first cut. Move in quickly at 100 percent, throttle back to 10% and toggle between [Feedhold] and 10% using the pause button. When you are satisfied that you've got it right, hit the zero to the right of nine (feedrate=100%) and go.

The [Verify] button runs the interpreter through the code without initiating any motion. If Verify finds a problem it will stop the read near the problem block and put up some sort of message. Most of the time you will be able to figure out the problem with your program by reading the message and looking in the program window at the highlighted line. Some of the messages are not very helpful. Sometimes you will need to read a line or two ahead of the highlight to see the problem. Occasionally the message will refer to something well ahead of the highlight line. This often happens if you forget to end your program with an acceptable code like %, M2, M30, or M60.

8.4.3 MDI

The MDI button or <F5> sets the Manual Data Input mode. This mode displays a single line of text for block entry and shows the currently active modal codes for the interpreter.

From the Sherline CNC Operators Manual:

MDI mode allows you to enter single blocks and have the interpreter execute them as if they were part of a program (kind of like a one-line program). You can execute circles, arcs, lines and such. You can even test sets of program lines by entering one block, waiting for that motion to end, and then enter the next block. Below the entry window, there is a listing of all of the current modal codes. This listing can be very handy. I often forget to enter a g00 before I command a motion. If nothing happens I look down there to see if g80 is in effect. G80 stops any motion. If it's there I remember to issue a block like g00 x0 y0 z0. In MDI you are entering text from the keyboard so none of the main keys work for commands to the running machine. [F1] will Estop the control.

Since many of the keyboard keys are needed for entry, most of the bindings that were available in auto mode are not available here.

8.4.4 [FEEDHOLD]—[CONTINUE]

Feedhold is a toggle. When the LinuxCNC is ready to handle or is handling a motion command this button shows the feedhold label on a red background. If feedhold has been pressed then it will show the continue label. Using it to pause motion has the advantage of being able to restart the program from where you stopped it. Feedhold will toggle between zero speed and whatever feed rate override was active before it was pressed. This button and the function that it activates is also bound to the pause button on most keyboards.

8.4.5 [ABORT]

The abort button stops any motion when it is pressed. It also removes the motion command from the LinuxCNC. No further motions are cued up after this button is pressed. If you are in auto mode, this button removes the rest of the program from the

motion cue. It also records the number of the line that was executing when it was pressed. You can use this line number to restart the program after you have cleared up the reasons for pressing it.

8.4.6 [ESTOP]

The estop button is also a toggle but it works in three possible settings.

- When Mini starts up it will show a raised button with red background with black letters that say *ESTOP PUSH*. This is the correct state of the machine when you want to run a program or jog an axis. Estop is ready to work for you when it looks like this.
- If you push the estop button while a motion is being executed, you will see a recessed gray button that says *ESTOPPED*. You will not be able to move an axis or do any work from the Mini gui when the estop button displays this way. Pressing it with your mouse will return Mini to normal ready condition.
- A third view is possible here. A recessed green button means that estop has been take off but the machine has not been turned on. Normally this only happens when <F1> estop has been pressed but <F2> has not been pressed.

Joe Martin says, "When all else fails press a software [ESTOP]." This does everything that abort does but adds in a reset so that the LinuxCNC returns to the standard settings that it wakes up on. If you have an external estop circuit that watches the relevant parallel port or DIO pin, a software estop can turn off power to the motors.

From the Sherline CNC Operators Manual:

Most of the time, when we abort or E-Stop it's because something went wrong. Perhaps we broke a tool and want to change it. We switch to manual mode and raise the spindle, change tools, and assuming that we got the length the same, get ready to go on. If we return the tool to the same place where the abort was issued, LinuxCNC will work perfectly. It is possible to move the restart line back or ahead of where the abort happened. If you press the [Back] or [Ahead] buttons you will see a blue highlight that shows the relationship between the abort line and the one on which LinuxCNC will start up again. By thinking through what is happening at the time of the restart you can place the tool tip where it will resume work in an acceptable manner. You will need to think through things like tool offsets, barriers to motion along a diagonal line, and such, before you press the [Restart] button.

8.5 Left Column

There are two columns below the control line. The left side of the screen displays information of interest to the operator. There are very few buttons to press here.

8.5.1 Axis Position Displays

The axis position displays work exactly like they do with tkLinuxCNC. The color of the letters is important.

- Red indicates that the machine is sitting on a limit switch or the polarity of a min or max limit is set wrong in the ini file.
- Yellow indicates that the machine is ready to be homed.
- Green indicates that the machine has been homed.

The position can be changed to display any one of several values by using the menu settings. The startup or default settings can be changed in the ini file so these displays wake up just the way that you want them.

8.5.2 Feed rate Override

Immediately below the axis position displays is the feed rate override slider. You can operate feed rate override and feedhold in any mode of operation. Override will change the speed of jogs or feed rate in manual or MDI modes. You can adjust feed rate override by grabbing the slider with your mouse and dragging it along the groove. You can also change feed rate a percent at a time by clicking in the slider's groove. In auto mode you can also set feed override in 10% increments by pressing the top row of numbers. This slider is a handy visual reference to how much override is being applied to programmed feed rate.

8.5.3 Messages

The message display located under the axis positions is a sort of scratch pad for LinuxCNC. If there are problems it will report them there. If you try to home or move an axis when the [ESTOP] button is pressed, you'll get a message that says something about commanding motion when LinuxCNC is not ready. If an axis faults out for something like falling behind, the message pad will show what happened. If you want to remind an operator to change a tool, for example, you can add a line of code to your program that will display in the message box. An example might be (msg, change to tool #3 and press resume). This line of code, included in a program, will display *change to tool #3 and press resume* in the message box. The word msg, (with comma included) is the command to make this happen; without *msg*, the message wouldn't be displayed. It will still show in the auto modes' display of the program file.

To erase messages simply click the message button at the top of the pad or, on the keyboard, hold down the [Alt] key and press the [m] key.

8.6 Right Column

The right column is a general purpose place to display and work. Here you can see the modal buttons and text entry or displays. Here you can view a plot of the tool path that will be commanded by your program. You can also write programs and control tools and offsets here. The modal screens have been described above. Each of the popin displays are described in detail below.

8.6.1 Program Editor

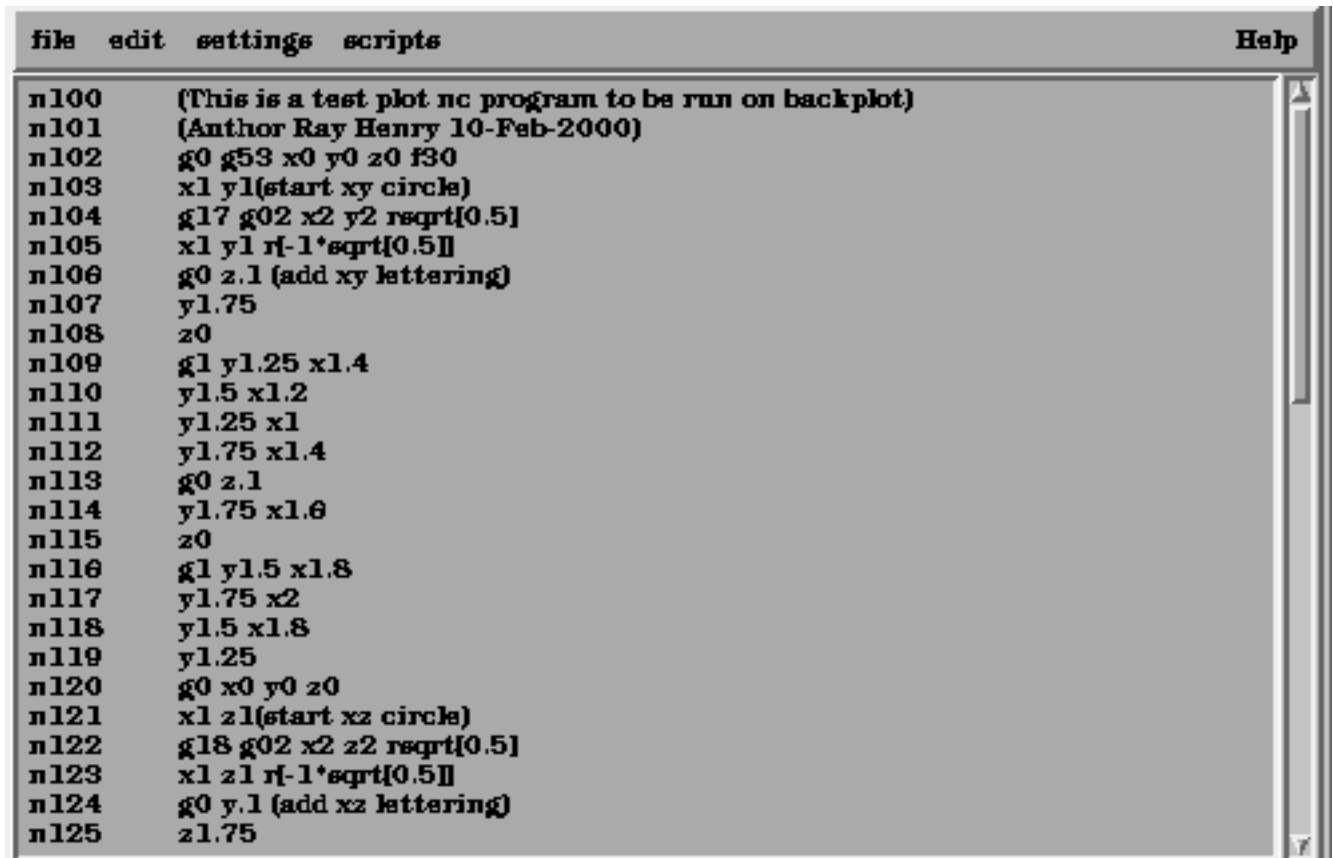


Figure 8.5: Mini Text Editor

The editor is rather limited compared to many modern text editors. It does not have *undo* nor *paste* between windows with the clipboard. These were eliminated because of interaction with a running program. Future releases will replace these functions so that it will work the way you've come to expect from a text editor. It is included because it has the rather nice feature of being able to number and renumber lines in the way that the interpreter expects of a file. It will also allow you to cut and paste from one part of a file to another. In addition, it will allow you to save your changes and submit them to the LinuxCNC interpreter with the same menu click. You can work on a file in here for a while and then save and load if the LinuxCNC is in Auto mode. If you have been running a file and find that you need to edit it, that file will be placed in the editor when you click on the editor button on the top menu.

8.6.2 Backplot Display

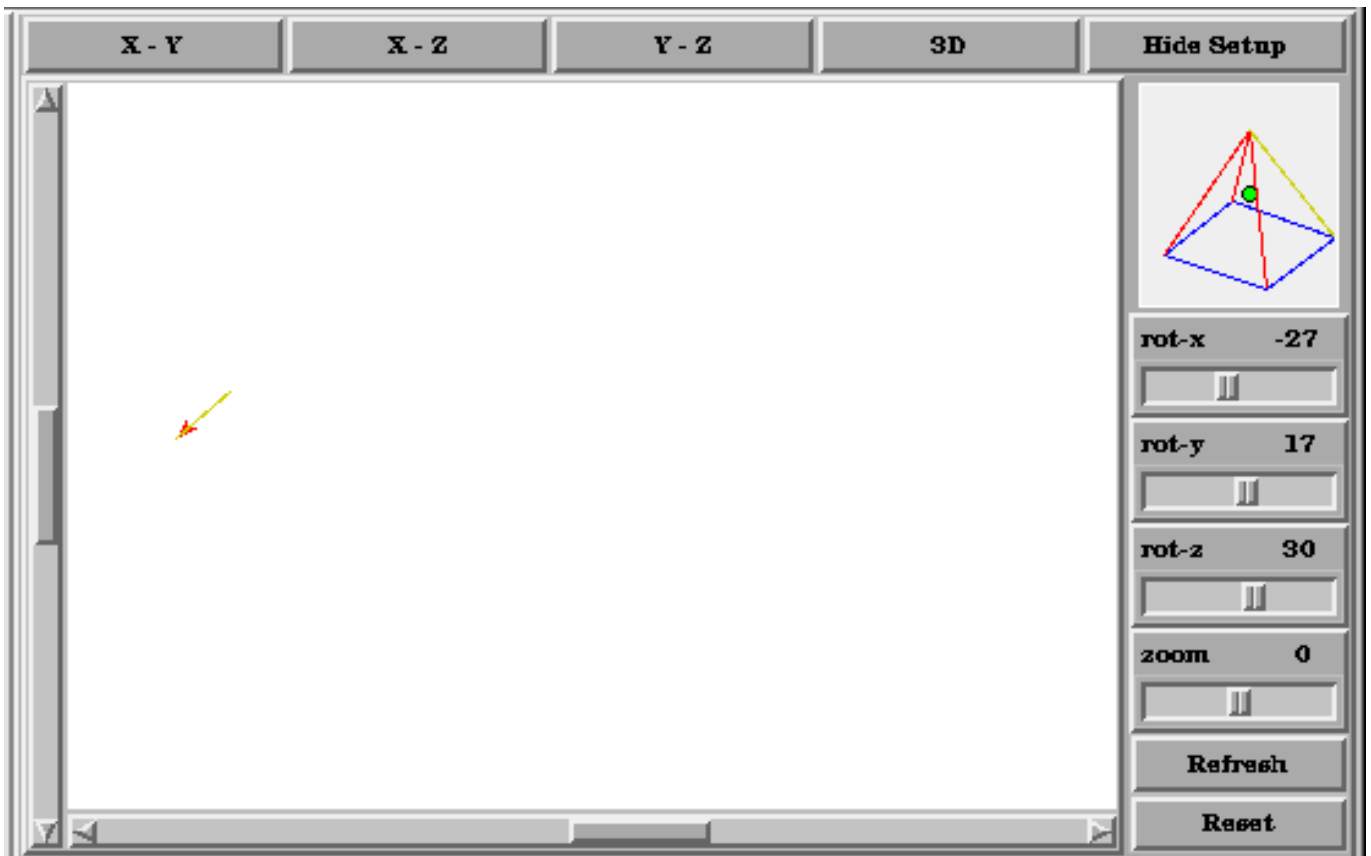


Figure 8.6: Minis Backplotter

Backplot [Backplot] will show the tool path that can be viewed from a chosen direction. *3-D* is the default. Other choices and controls are displayed along the top and right side of the pop-in. If you are in the middle of a cut when you press one of these control buttons the machine will pause long enough to re-compute the view.

Along the right side of the pop-in there is a small pyramid shaped graphic that tries to show the angle you are viewing the tool path from. Below it are a series of sliders that allow you to change the angle of view and the size of the plot. You can rotate the little position angle display with these. They take effect when you press the [Refresh] button. The [Reset] button removes all of the paths from the display and readies it for a new run of the program but retains your settings for that session.

If backplot is started before a program is started, it will try to use some color lines to indicate the kind of motion that was used to make it. A green line is a rapid move. A black line is a feed rate move. Blue and red indicate arcs in counterclockwise and clockwise directions.

The backplotter with Mini allows you to zoom and rotate views after you have run your program but it is not intended to store a tool path for a long period of time.

8.6.3 Tool Page

The tool page is pretty much like the others. You can set length and diameter values here and they become effective when you press the [Enter] key. You will need to set up your tool information before you begin to run a program. You can't change tool offsets while the program is running or when the program is paused.

TOOL SETUP
Click or tab to edit. Press enter to return to keyboard machine control.

TOOL NUMBER	LENGTH	DIAMETER	COMMENT
1	1.456	0.250	Drill
2	1.000	0.4968	End Mill
3	0.0	0.0	empty
4	0.0	0.0	empty
5	0.0	0.0	empty
6	0.0	0.0	empty

Add Extra Tool

Remove Last Tool

Figure 8.7: Mini Tool Display

The [Add Tools] and [Remove Tools] buttons work on the bottom of the tool list so you will want to fill in tool information in descending order. Once a new tool has been added, you can use it in a program with the usual G-code commands. There is a 32 tool limit in the current LinuxCNC configuration files but you will run out of display space in Mini long before you get there.

Tip

You can use Menu > View > Show Popin Full to see more tools if you need.

8.6.4 Offset Page

The offset page can be used to display and setup work offsets. The coordinate system is selected along the left hand side of the window. Once you have selected a coordinate system you can enter values or move an axis to a teach position.

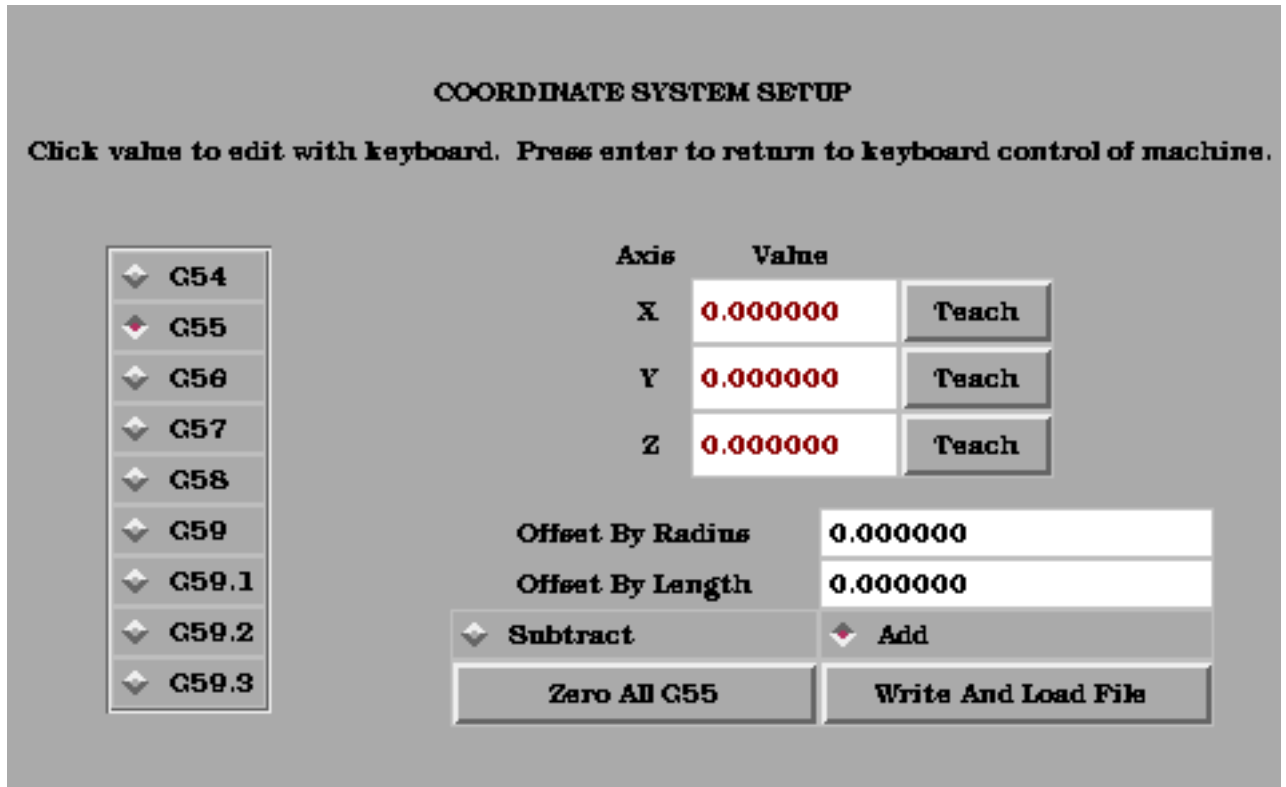


Figure 8.8: Mini Offset Display

You can also teach using an edgefinder by adding the radius and length to the offset_by widgets. When you do this you may need to add or subtract the radius depending upon which surface you choose to touch from. This is selected with the add or subtract radiobuttons below the offset windows.

The zero all for the active coordinate system button will remove any offsets that you have showing but they are not set to zero in the variable file until you press the write and load file button as well. This write and load file button is the one to use when you have set all of the axis values that you want for a coordinate system.

8.7 Keyboard Bindings

A number of the bindings used with tkLinuxCNC have been preserved with mini. A few of the bindings have been changed to extend that set or to ease the operation of a machine using this interface. Some keys operate the same regardless of the mode. Others change with the mode that LinuxCNC is operating in.

8.7.1 Common Keys

- *Pause* - Toggle feedhold
- *Escape* - abort motion
- *F1* - toggle estop/estop reset state
- *F2* - toggle machine off/machine on state
- *F3* - manual mode
- *F4* - auto mode

- *F5* - MDI mode
- *F6* - reset interpreter

The following only work for machines using auxiliary I/O

- *F7* - toggle mist on/mist off
- *F8* - toggle flood on/flood off
- *F9* - toggle spindle forward/off
- *F10* - toggle spindle reverse/off
- *F11* - decrease spindle speed
- *F12* - increase spindle speed

8.7.2 Manual Mode

- *I-9 0* - set feed override to 10%-90%, 0 is 100%
- *~* - set feed override to 0 or feedhold
- *x* - select X axis
- *y* - select Y axis
- *z* - select Z axis
- *a* - select A axis
- *b* - select B axis
- *c* - select C axis
- *Left Right Arrow* - jog X axis
- *Up Down Arrow* - jog Y axis
- *Page Up Down* - jog Z axis
- *- _* - jog the active axis in the minus direction
- *+ =* - jog the active axis in the plus direction.
- *Home* - home selected axis
- *i I* - toggle through jog increments

The following only work with a machine using auxiliary I/O

- *b* - take spindle brake off
 - *Alt-b* - put spindle brake on
-

8.7.3 Auto Mode

- *1-9,0* - set feed override to 10%-90%, 0 is 100%
- *~* - set feed override to 0 or feedhold
- *o/O* - open a program
- *r/R* - run an opened program
- *p/P* - pause an executing program
- *s/S* - resume a paused program
- *a/A* - step one line in a paused program

8.8 Misc

One of the features of Mini is that it displays any axis above number 2 as a rotary and will display degree units for it. It also converts to degree units for incremental jogs when a rotary axis has the focus.

Chapter 9

KEYSTICK GUI

9.1 Introduction

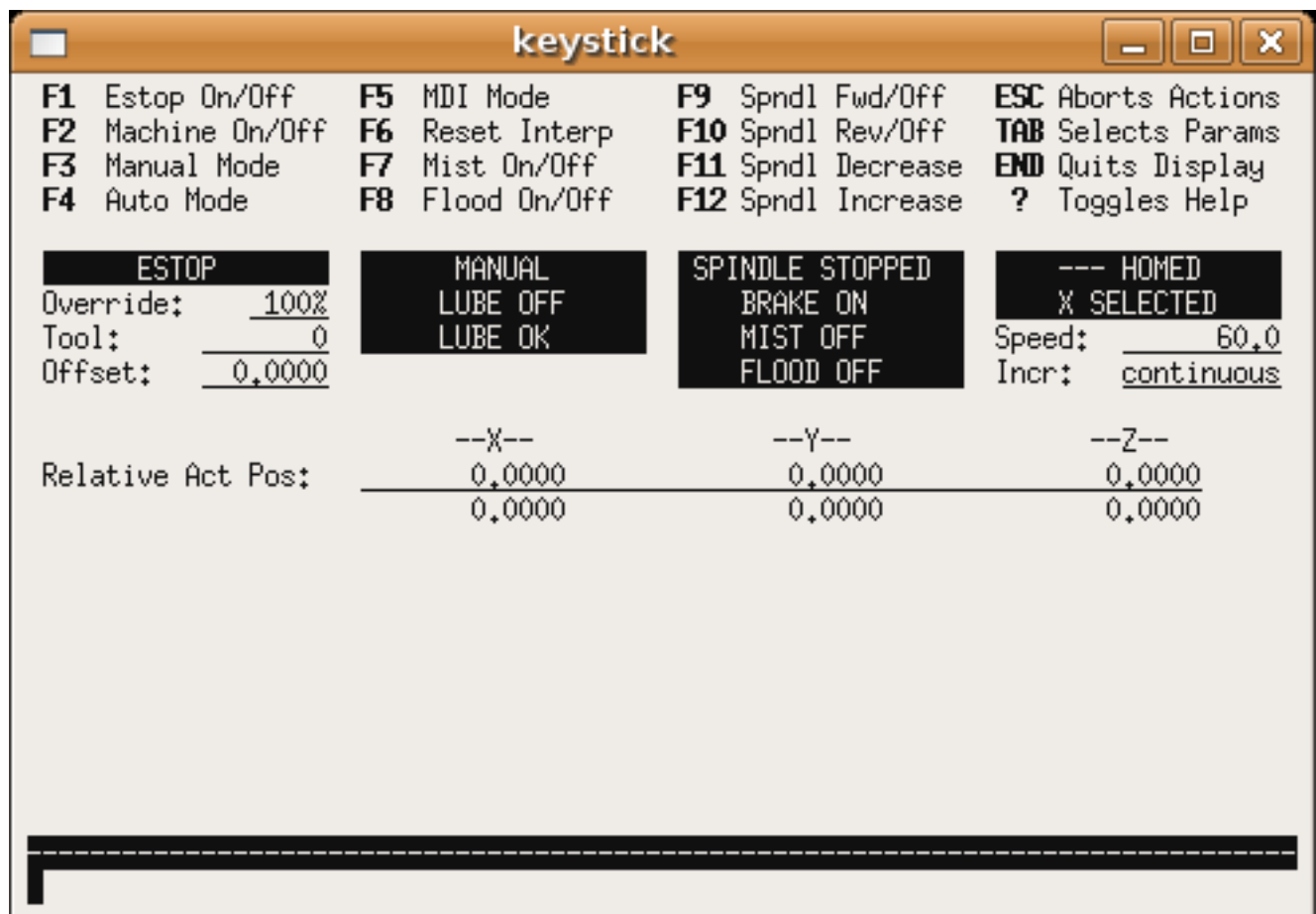


Figure 9.1: The Mini Graphical Interface

Keystick is a minimal text based interface.

9.2 Installing

To use keystick change the *DISPLAY* setting in the ini file setting to:

```
DISPLAY = keystick
```

9.3 Using

Keystick is very simple to use. In the MDI Mode you simply start typing the g code and it shows up in the bottom text area. The ? key toggles help.

Part III

Using LinuxCNC

Chapter 10

CNC Machine Overview

This section gives a brief description of how a CNC machine is viewed from the input and output ends of the Interpreter.

10.1 Mechanical Components

A CNC machine has many mechanical components that may be controlled or may affect the way in which control is exercised. This section describes the subset of those components that interact with the Interpreter. Mechanical components that do not interact directly with the Interpreter, such as the jog buttons, are not described here, even if they affect control.

10.1.1 Axes

Any CNC machine has one or more Axes. Different types of CNC machines have different combinations. For instance, a *4-axis milling machine* may have XYZA or XYZB axes. A lathe typically has XZ axes. A foam-cutting machine may have XYUV axes. In LinuxCNC, the case of a XYYZ *gantry* machine with two motors for one axis is better handled by kinematics rather than by a second linear axis.¹

Primary Linear Axes *axesprimary linear primary linear* The X, Y, and Z axes produce linear motion in three mutually orthogonal directions.

Secondary Linear Axes *axessecondary linear secondary linear* The U, V, and W axes produce linear motion in three mutually orthogonal directions. Typically, X and U are parallel, Y and V are parallel, and Z and W are parallel.

Rotational Axes *axesrotational rotational* The A, B and C axes produce angular motion (rotation). Typically, A rotates around a line parallel to X, B rotates around a line parallel to Y, and C rotates around a line parallel to Z.

10.1.2 Spindle

A CNC machine typically has a spindle which holds one cutting tool, probe, or the material in the case of a lathe. The spindle may or may not be controlled by the CNC software.

10.1.3 Coolant

If a CNC machine has components to provide mist coolant and/or flood coolant they can be controlled by G codes.

¹If the motion of mechanical components is not independent, as with hexapod machines, the RS274/NGC language and the canonical machining functions will still be usable, as long as the lower levels of control know how to control the actual mechanisms to produce the same relative motion of tool and workpiece as would be produced by independent axes. This is called *kinematics*.

10.1.4 Feed and Speed Override

A CNC machine can have separate feed and speed override controls, which let the operator specify that the actual feed rate or spindle speed used in machining at some percentage of the programmed rate.

10.1.5 Block Delete Switch

A CNC machine can have a block delete switch. See the [Block Delete](#) Section.

10.1.6 Optional Program Stop Switch

A CNC machine can have an optional program stop switch. See the [Optional Program Stop](#) Section.

10.2 Control and Data Components

10.2.1 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.

The U, V and W axes also form a standard right-handed coordinate system. X and U are parallel, Y and V are parallel, and Z and W are parallel (when A, B, and C are rotated to zero).

10.2.2 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.²

10.2.3 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. On a lathe, tool length offsets can be specified for X and Z axes, and the controlled point is either at the tool tip or slightly outside it (where the perpendicular, axis-aligned lines touched by the *front* and *side* of the tool intersect).

²If the parallelism requirement is violated, the system builder will have to say how to distinguish clockwise from counterclockwise.

10.2.4 Coordinated Linear Motion

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

Coordinated linear motion can be performed either at the prevailing feed rate, or at traverse rate, or it may be synchronized to the spindle rotation. If physical limits on axis speed make the desired rate unobtainable, all axes are slowed to maintain the desired path.

10.2.5 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* or *feed per revolution* modes are being used see Section [G93 G94 G95](#).

1. If any of XYZ are moving, F is in units per minute in the XYZ cartesian system, and all other axes (ABCUVW) move so as to start and stop in coordinated fashion.
2. Otherwise, if any of UVW are moving, F is in units per minute in the UVW cartesian system, and all other axes (ABC) move so as to start and stop in coordinated fashion.
3. Otherwise, the move is pure rotary motion and the F word is in rotary units in the ABC *pseudo-cartesian* system.

10.2.6 Coolant

Flood coolant and mist coolant may each be turned on independently. The RS274/NGC language turns them off together see Section [M7 M8 M9](#).

10.2.7 Dwell

A machining center 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, so the spindle is usually turning during a dwell. Regardless of the Path Control Mode (see Section [Path Control](#)) the machine will stop exactly at the end of the previous programmed move, as though it was in exact path mode.

10.2.8 Units

Units used for distances along the X, Y, and Z axes may be measured in millimeters 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 degrees per minute, or length units per spindle revolution, as described in Section [G93 G94 G95](#).

10.2.9 Current Position

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

1. Length units are changed.
 2. Tool length offset is changed.
 3. Coordinate system offsets are changed.
-

10.2.10 Selected Plane

There is always a *selected plane*, which must be the XY-plane, the YZ-plane, or the XZ-plane of the machining center. 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.

10.2.11 Tool Carousel

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

10.2.12 Tool Change

A machining center may be commanded to change tools.

10.2.13 Pallet Shuttle

The two pallets may be exchanged by command.

10.2.14 Path Control Mode

The machining center may be put into any one of three path control modes: (1) exact stop mode, (2) exact path mode, or (3) continuous mode with optional tolerance. In exact stop mode, the machine stops briefly at the end of each programmed move. In exact path mode, the machine follows the programmed path as exactly as possible, slowing or stopping if necessary at sharp corners of the path. In continuous mode, sharp corners of the path may be rounded slightly so that the feed rate may be kept up (but by no more than the tolerance, if specified). See Sections [G61/G61.1](#) and [G64](#).

10.3 Interpreter Interaction with Switches

The Interpreter interacts with several switches. This section describes the interactions in more detail. In no case does the Interpreter know what the setting of any of these switches is.

10.3.1 Feed and Speed Override Switches

The Interpreter will interpret RS274/NGC commands which enable *M48* or disable *M49* the feed and speed override switches. For certain moves, such as the traverse out of the end of a thread during a threading cycle, the switches are disabled automatically.

LinuxCNC reacts to the speed and feed override settings when these switches are enabled.

See the [M48 M49 Override](#) section for more information.

10.3.2 Block Delete Switch

If the block delete switch is on, lines of G code which start with a slash (the block delete character) are not interpreted. If the switch is off, such lines are interpreted. Normally the block delete switch should be set before starting the NGC program.

10.3.3 Optional Program Stop Switch

If this switch is on and an M1 code is encountered, program execution is paused.

10.4 Tool Table

A tool table is required to use the Interpreter. The file tells which tools are in which tool changer slots and what the size and type of each tool is. The name of the tool table is defined in the ini file:

```
[EMCIO]

# tool table file
TOOL_TABLE = tooltable.tbl
```

The default filename probably looks something like the above, but you may prefer to give your machine its own tool table, using the same name as your ini file, but with a tbl extension:

```
TOOL_TABLE = acme_300.tbl
```

or

```
TOOL_TABLE = EMC-AXIS-SIM.tbl
```

For more information on the specifics of the tool table format, see the [Tool Table Format](#) Section.

10.5 Parameters

In the RS274/NGC language view, a machining center maintains an array of numerical parameters defined by a system definition (RS274NGC_MAX_PARAMETERS). Many of them have specific uses especially in defining coordinate systems. The number of numerical parameters can increase as development adds support for new parameters. The parameter array persists over time, even if the machining center is powered down. LinuxCNC uses a parameter file to ensure persistence and gives the Interpreter the responsibility for maintaining the file. The Interpreter reads the file when it starts up, and writes the file when it exits.

All parameters are available for use in G code programs.

The format of a parameter file is shown in the following table. The file consists of any number of header lines, followed by one blank line, followed by any number of lines of data. The Interpreter skips over the header lines. It is important that there be exactly one blank line (with no spaces or tabs, even) before the data. The header line shown in the following table describes the data columns, so it is suggested (but not required) that that line always be included in the header.

The Interpreter reads only the first two columns of the table. The third column, *Comment*, is not read by the Interpreter.

Each line of the file contains the index number of a parameter in the first column and the value to which that parameter should be set in the second column. The value is represented as a double-precision floating point number inside the Interpreter, but a decimal point is not required in the file. All of the parameters shown in the following table are required parameters and must be included in any parameter file, except that any parameter representing a rotational axis value for an unused axis may be omitted. An error will be signaled if any required parameter is missing. A parameter file may include any other parameter, as long as its number is in the range 1 to 5400. The parameter numbers must be arranged in ascending order. An error will be signaled if not. Any parameter included in the file read by the Interpreter will be included in the file it writes as it exits. The original file is saved as a backup file when the new file is written. Comments are not preserved when the file is written.

Table 10.1: Parameter File Format

Parameter Number	Parameter Value	Comment
5161	0.0	G28 Home X
5162	0.0	G28 Home Y

See the [Parameters](#) section for more information.

Chapter 11

Coordinate System

11.1 Introduction

You have seen how handy a tool length offset can be. Having this allows the programmer to ignore the actual tool length when writing a part program. In the same way, it is really nice to be able to find a prominent part of a casting or block of material and work a program from that point rather than having to take account of the location at which the casting or block will be held during the machining.

This chapter introduces you to offsets as they are used by the LinuxCNC. These include;

- machine coordinates (G53)
- nine fixture offsets (G54-G59.3)
- global offsets (G92)

11.2 The Machine Position Command (G53)

Regardless of any offsets that may be in effect, putting a G53 in a block of code tells the interpreter to go to the real or absolute axis positions commanded in the block. For example

```
G53 G0 X0 Y0 Z0
```

will get you to the actual position where these three axes are zero. You might use a command like this if you have a favorite position for tool changes or if your machine has an auto tool changer. You might also use this command to get the tool out of the way so that you can rotate or change a part in a vice.

G53 is not a modal command. It must be used on each line where motion based upon absolute machine position is desired.

To display a graphic message and stop until the message window is closed use a graphic display program like Eye of Gnome to display the graphic file. When you close it the program will resume.

M110 Example file

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png
exit 0
```

To display a graphic message and continue processing the G code file suffix an ampersand to the command.

M110 Example display and keep going

```
#!/bin/bash
eog /home/john/linuxcnc/nc_files/message.png &
exit 0
```

Chapter 20

Other Codes

20.1 F: Set Feed Rate

Fx - set the feed rate to *x*. *x* is usually in machine units (inches or millimeters) per minute.

The application of the feed rate is as described in the [Feed Rate](#) Section, unless inverse time feed rate mode is in effect, in which case the feed rate is as described in the [G93 G94 G95](#) Section.

20.2 S: Set Spindle Speed

Sx - set the speed of the spindle to *x* revolutions per minute (RPM).

The spindle will turn at that speed when a *M3* or *M4* is in effect. 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.

As described in the [G84](#) Section, 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%.

20.3 T: Select Tool

Tx - prepare to change to tool *x*.

The tool is not changed until an *M6* is programmed (see Section [M6](#)). The *T* word may appear on the same line as the *M6* or on a previous line. It is OK if *T* words appear on two or more lines with no tool change. Only the the most recent *T* word will take effect at the next tool change.

Note

When LinuxCNC is configured for a nonrandom toolchanger (see the entry for `RANDOM_TOOLCHANGER` in the [EMCIO Section](#)), *T0* gets special handling: no tool will be selected. This is useful if you want the spindle to be empty after a tool change.

Note

T0 When LinuxCNC is configured for a random toolchanger (see the entry for RANDOM_TOOLCHANGER in the [EMCIO Section](#)), *T0* does not get any special treatment: *T0* is a valid tool like any other. It is customary to use *T0* on a random toolchanger machine to track an empty pocket, so that it behaves like a nonrandom toolchanger machine and unloads the spindle.

It is an error if:

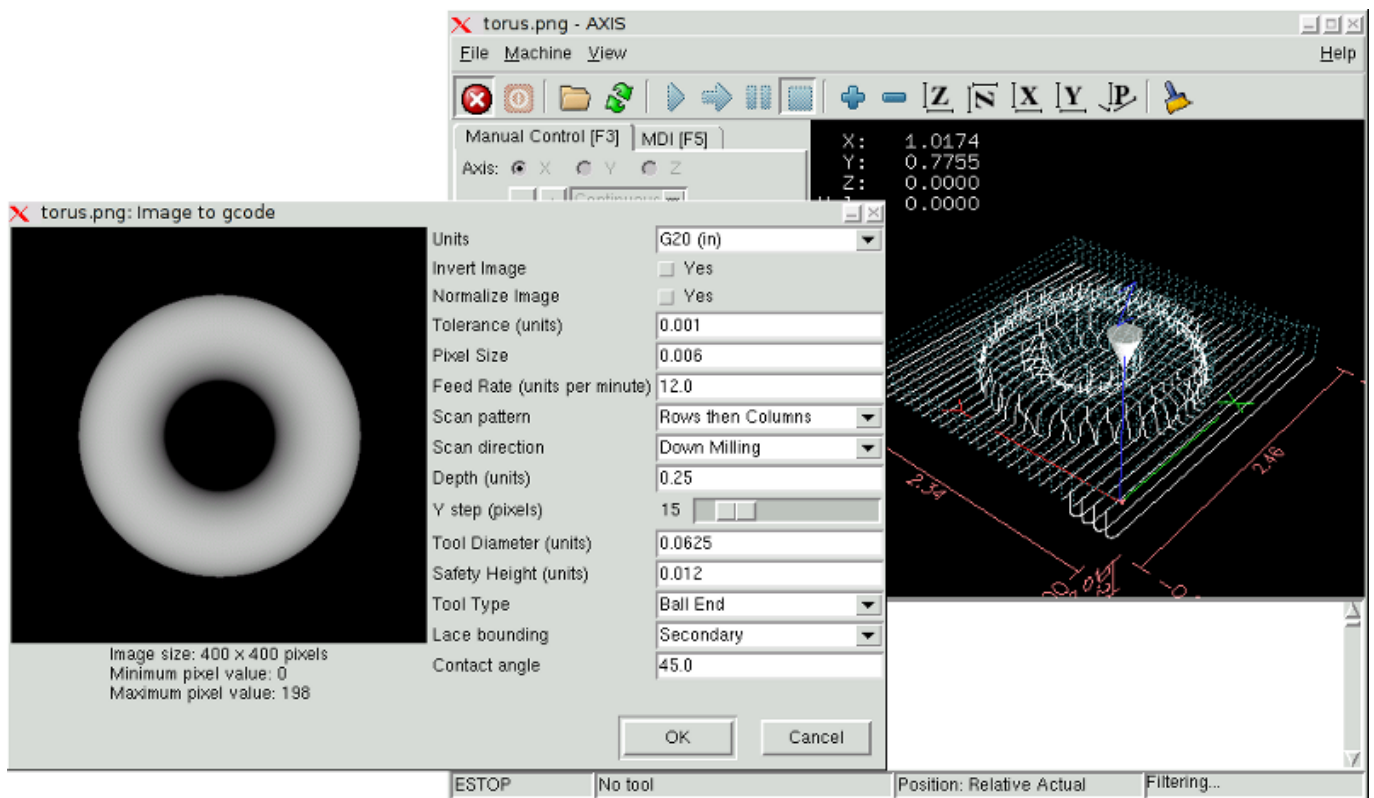
- a negative T number is used,
- T number is used that does not appear in the tool table file (with the exception that *T0* on nonrandom toolchangers **is** accepted, as noted above).

On some machines, the carousel will move when a T word is programmed, at the same time machining is occurring. On such machines, programming the T word several lines before a tool change will save time. A common programming practice for such machines is to put the T word for the next tool to be used on the line after a tool change. This maximizes the time available for the carousel to move.

Rapid moves after a *T<n>* will not show on the AXIS preview until after a feed move. This is for machines that travel long distances to change the tool like a lathe. This can be very confusing at first. To turn this feature off for the current tool program a G1 without any move after the *T<n>*.

Chapter 21

Image to G Code



21.1 What is a depth map?

A depth map is a greyscale image where the brightness of each pixel corresponds to the depth (or height) of the object at each point.

21.2 Integrating image-to-gcode with the AXIS user interface

Add the following lines to the *[FILTER]* section of your .ini file to make AXIS automatically invoke image-to-gcode when you open a .png, .gif, or .jpg image

```
PROGRAM_EXTENSION = .png,.gif,.jpg Grayscale Depth Image
png = image-to-gcode
```

```
gif = image-to-gcode  
jpg = image-to-gcode
```

The standard *sim/axis.ini* configuration file is already configured this way.

21.3 Using image-to-gcode

Start image-to-gcode either by opening an image file in AXIS, or by invoking image-to-gcode from the terminal, as follows:

```
image-to-gcode torus.png > torus.ngc
```

Verify all the settings in the right-hand column, then press OK to create the gcode. Depending on the image size and options chosen, this may take from a few seconds to a few minutes. If you are loading the image in AXIS, the gcode will automatically be loaded and previewed once image-to-gcode completes. In AXIS, hitting reload will show the image-to-gcode option screen again, allowing you to tweak them.

21.4 Option Reference

21.4.1 Units

Specifies whether to use G20 (inches) or G21 (mm) in the generated g-code and as the units for each option labeled (*units*).

21.4.2 Invert Image

If “no”, the black pixel is the lowest point and the white pixel is the highest point. If “yes”, the black pixel is the highest point and the white pixel is the lowest point.

21.4.3 Normalize Image

If yes, the darkest pixel is remapped to black, the lightest pixel is remapped to white.

21.4.4 Expand Image Border

If *None*, the input image is used as-is, and details which are at the very edges of the image may be cut off. If *White* or *Black*, then a border of pixels equal to the tool diameter is added on all sides, and details which are at the very edges of the images will not be cut off.

21.4.5 Tolerance (units)

When a series of points are within *tolerance* of being a straight line, they are output as a straight line. Increasing tolerance can lead to better contouring performance in LinuxCNC, but can also remove or blur small details in the image.

21.4.6 Pixel Size (units)

One pixel in the input image will be this many units—usually this number is much smaller than 1.0. For instance, to mill a 2.5x2.5-inch object from a 400x400 image file, use a pixel size of .00625, because $2.5 / 400 = .00625$.

21.4.7 Plunge Feed Rate (units per minute)

The feed rate for the initial plunge movement.

21.4.8 Feed Rate (units per minute)

The feed rate for other parts of the path.

21.4.9 Spindle Speed (RPM)

The spindle speed S code that should be put into the gcode file.

21.4.10 Scan Pattern

Possible scan patterns are:

- Rows
- Columns
- Rows, then Columns
- Columns, then Rows

21.4.11 Scan Direction

Possible scan directions are:

- Positive: Start milling at a low X or Y axis value, and move towards a high X or Y axis value
- Negative: Start milling at a high X or Y axis value, and move towards a low X or Y axis value
- Alternating: Start on the same end of the X or Y axis travel that the last move ended on. This reduces the amount of traverse movements
- Up Milling: Start milling at low points, moving towards high points
- Down Milling: Start milling at high points, moving towards low points

21.4.12 Depth (units)

The top of material is always at $Z=0$. The deepest cut into the material is $Z=-depth$.

21.4.13 Step Over (pixels)

The distance between adjacent rows or columns. To find the number of pixels for a given units distance, compute $distance/pixel\ size$ and round to the nearest whole number. For example, if $pixel\ size=.006$ and the desired step over $distance=.015$, then use a Step Over of 2 or 3 pixels, because $.015/.006=2.5$.

21.4.14 Tool Diameter

The diameter of the cutting part of the tool.

21.4.15 Safety Height

The height to move to for traverse movements. image-to-gcode always assumes the top of material is at $Z=0$.

21.4.16 Tool Type

The shape of the cutting part of the tool. Possible tool shapes are:

- Ball End
- Flat End
- 45 degree “vee”
- 60 degree “vee”

21.4.17 Lace bounding

This controls whether areas that are relatively flat along a row or column are skipped. This option only makes sense when both rows and columns are being milled. Possible bounding options are:

- None: Rows and columns are both fully milled.
- Secondary: When milling in the second direction, areas that do not strongly slope in that direction are skipped.
- Full: When milling in the first direction, areas that strongly slope in the second direction are skipped. When milling in the second direction, areas that do not strongly slope in that direction are skipped.

21.4.18 Contact angle

When *Lace bounding* is not *None*, slopes greater than *Contact angle* are considered to be *strong* slopes, and slopes less than that angle are considered to be weak slopes.

21.4.19 Roughing offset and depth per pass

Image-to-gcode can optionally perform rouging passes. The depth of successive roughing passes is given by *Roughing depth per pass*. For instance, entering 0.2 will perform the first roughing pass with a depth of 0.2, the second roughing pass with a depth of 0.4, and so on until the full Depth of the image is reached. No part of any roughing pass will cut closer than Roughing Offset to the final part. The following figure shows a tall vertical feature being milled. In this image, Roughing depth per pass is 0.2 inches and roughing offset is 0.1 inches.

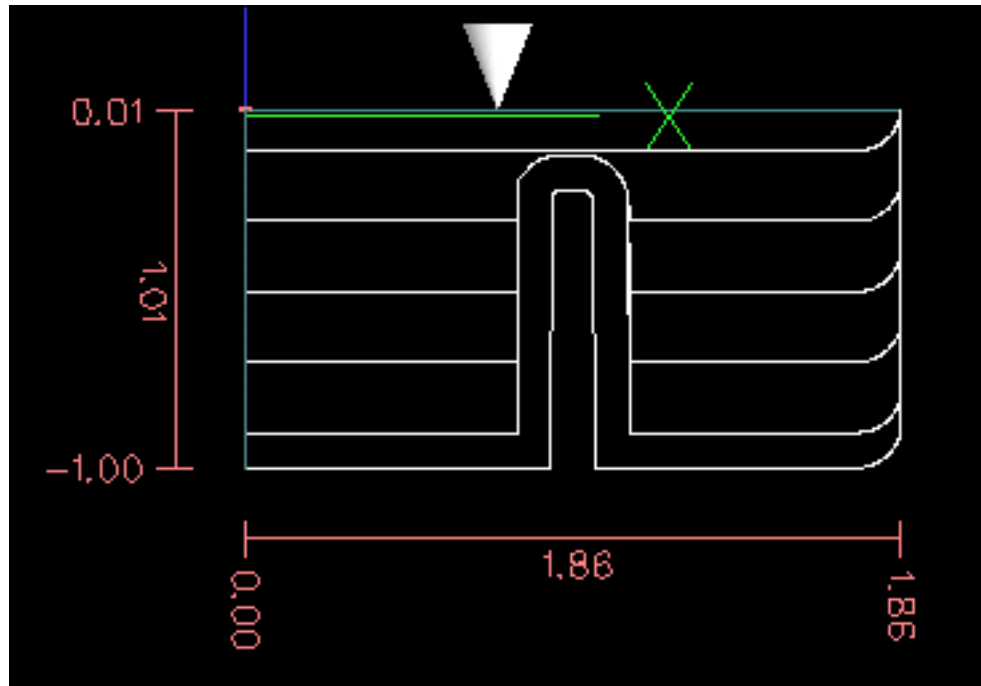


Figure 21.1: Roughing passes and final pass

Chapter 22

Glossary

A listing of terms and what they mean. Some terms have a general meaning and several additional meanings for users, installers, and developers.

Acme Screw

A type of lead-screw that uses an Acme thread form. Acme threads have somewhat lower friction and wear than simple triangular threads, but ball-screws are lower yet. Most manual machine tools use acme lead-screws.

Axis

One of the computer controlled movable parts of the machine. For a typical vertical mill, the table is the X axis, the saddle is the Y axis, and the quill or knee is the Z axis. Angular axes like rotary tables are referred to as A, B, and C. Additional linear axes relative to the tool are called U, V, and W respectively.

Axis(GUI)

One of the Graphical User Interfaces available to users of LinuxCNC. It features the modern use of menus and mouse buttons while automating and hiding some of the more traditional LinuxCNC controls. It is the only open-source interface that displays the entire tool path as soon as a file is opened.

Backlash

The amount of "play" or lost motion that occurs when direction is reversed in a lead screw. or other mechanical motion driving system. It can result from nuts that are loose on leadscrews, slippage in belts, cable slack, "wind-up" in rotary couplings, and other places where the mechanical system is not "tight". Backlash will result in inaccurate motion, or in the case of motion caused by external forces (think cutting tool pulling on the work piece) the result can be broken cutting tools. This can happen because of the sudden increase in chip load on the cutter as the work piece is pulled across the backlash distance by the cutting tool.

Backlash Compensation

Any technique that attempts to reduce the effect of backlash without actually removing it from the mechanical system. This is typically done in software in the controller. This can correct the final resting place of the part in motion but fails to solve problems related to direction changes while in motion (think circular interpolation) and motion that is caused when external forces (think cutting tool pulling on the work piece) are the source of the motion.

Ball Screw

A type of lead-screw that uses small hardened steel balls between the nut and screw to reduce friction. Ball-screws have very low friction and backlash, but are usually quite expensive.

Ball Nut

A special nut designed for use with a ball-screw. It contains an internal passage to re-circulate the balls from one end of the screw to the other.

CNC

Computer Numerical Control. The general term used to refer to computer control of machinery. Instead of a human operator turning cranks to move a cutting tool, CNC uses a computer and motors to move the tool, based on a part program.

Comp

A tool used to build, compile and install LinuxCNC HAL components.

Configuration(n)

A directory containing a set of configuration files. Custom configurations are normally saved in the users home/linuxcnc/-configs directory. These files include LinuxCNC's traditional INI file and HAL files. A configuration may also contain several general files that describe tools, parameters, and NML connections.

Configuration(v)

The task of setting up LinuxCNC so that it matches the hardware on a machine tool.

Coordinate Measuring Machine

A Coordinate Measuring Machine is used to make many accurate measurements on parts. These machines can be used to create CAD data for parts where no drawings can be found, when a hand-made prototype needs to be digitized for moldmaking, or to check the accuracy of machined or molded parts.

Display units

The linear and angular units used for onscreen display.

DRO

A Digital Read Out is a system of position-measuring devices attached to the slides of a machine tool, which are connected to a numeric display showing the current location of the tool with respect to some reference position. DROs are very popular on hand-operated machine tools because they measure the true tool position without backlash, even if the machine has very loose Acme screws. Some DROs use linear quadrature encoders to pick up position information from the machine, and some use methods similar to a resolver which keeps rolling over.

EDM

EDM is a method of removing metal in hard or difficult to machine or tough metals, or where rotating tools would not be able to produce the desired shape in a cost-effective manner. An excellent example is rectangular punch dies, where sharp internal corners are desired. Milling operations can not give sharp internal corners with finite diameter tools. A *wire* EDM machine can make internal corners with a radius only slightly larger than the wire's radius. A *sinker* EDM can make internal corners with a radius only slightly larger than the radius on the corner of the sinking electrode.

EMC

The Enhanced Machine Controller. Initially a NIST project. Renamed to LinuxCNC in 2012.

EMCIO

The module within LinuxCNC that handles general purpose I/O, unrelated to the actual motion of the axes.

EMCMOT

The module within LinuxCNC that handles the actual motion of the cutting tool. It runs as a real-time program and directly controls the motors.

Encoder

A device to measure position. Usually a mechanical-optical device, which outputs a quadrature signal. The signal can be counted by special hardware, or directly by the parport with LinuxCNC.

Feed

Relatively slow, controlled motion of the tool used when making a cut.

Feed rate

The speed at which a cutting motion occurs. In auto or mdi mode, feed rate is commanded using an F word. F10 would mean ten machine units per minute.

Feedback

A method (e.g., quadrature encoder signals) by which LinuxCNC receives information about the position of motors

Feedrate Override

A manual, operator controlled change in the rate at which the tool moves while cutting. Often used to allow the operator to adjust for tools that are a little dull, or anything else that requires the feed rate to be "tweaked".

Floating Point Number

A number that has a decimal point. (12.300) In HAL it is known as float.

G-Code

The generic term used to refer to the most common part programming language. There are several dialects of G-code, LinuxCNC uses RS274/NGC.

GUI

Graphical User Interface.

General

A type of interface that allows communications between a computer and a human (in most cases) via the manipulation of icons and other elements (widgets) on a computer screen.

LinuxCNC

An application that presents a graphical screen to the machine operator allowing manipulation of the machine and the corresponding controlling program.

HAL

Hardware Abstraction Layer. At the highest level, it is simply a way to allow a number of building blocks to be loaded and interconnected to assemble a complex system. Many of the building blocks are drivers for hardware devices. However, HAL can do more than just configure hardware drivers.

Home

A specific location in the machine's work envelope that is used to make sure the computer and the actual machine both agree on the tool position.

ini file

A text file that contains most of the information that configures LinuxCNC for a particular machine.

Instance

One can have an instance of a class or a particular object. The instance is the actual object created at runtime. In programmer jargon, the Lassie object is an instance of the Dog class.

Joint Coordinates

These specify the angles between the individual joints of the machine. See also Kinematics

Jog

Manually moving an axis of a machine. Jogging either moves the axis a fixed amount for each key-press, or moves the axis at a constant speed as long as you hold down the key. In manual mode, jog speed can be set from the graphical interface.

kernel-space

See real-time.

Kinematics

The position relationship between world coordinates and joint coordinates of a machine. There are two types of kinematics. Forward kinematics is used to calculate world coordinates from joint coordinates. Inverse kinematics is used for exactly the opposite purpose. Note that kinematics does not take into account, the forces, moments etc. on the machine. It is for positioning only.

Lead-screw

An screw that is rotated by a motor to move a table or other part of a machine. Lead-screws are usually either ball-screws or acme screws, although conventional triangular threaded screws may be used where accuracy and long life are not as important as low cost.

Machine units

The linear and angular units used for machine configuration. These units are specified and used in the ini file. HAL pins and parameters are also generally in machine units.

MDI

Manual Data Input. This is a mode of operation where the controller executes single lines of G-code as they are typed by the operator.

NIST

National Institute of Standards and Technology. An agency of the Department of Commerce in the United States.

NML

Neutral Message Language provides a mechanism for handling multiple types of messages in the same buffer as well as simplifying the interface for encoding and decoding buffers in neutral format and the configuration mechanism.

Offsets

An arbitrary amount, added to the value of something to make it equal to some desired value. For example, gcode programs are often written around some convenient point, such as X0, Y0. Fixture offsets can be used to shift the actual execution point of that gcode program to properly fit the true location of the vise and jaws. Tool offsets can be used to shift the "uncorrected" length of a tool to equal that tool's actual length.

Part Program

A description of a part, in a language that the controller can understand. For LinuxCNC, that language is RS-274/NGC, commonly known as G-code.

Program Units

The linear and angular units used in a part program. The linear program units do not have to be the same as the linear machine units. See G20 and G21 for more information. The angular program units are always measured in degrees.

Python

General-purpose, very high-level programming language. Used in LinuxCNC for the Axis GUI, the Stepconf configuration tool, and several G-code programming scripts.

Rapid

Fast, possibly less precise motion of the tool, commonly used to move between cuts. If the tool meets the workpiece or the fixturing during a rapid, it is probably a bad thing!

Rapid rate

The speed at which a rapid motion occurs. In auto or mdi mode, rapid rate is usually the maximum speed of the machine. It is often desirable to limit the rapid rate when testing a g-code program for the first time.

Real-time

Software that is intended to meet very strict timing deadlines. Under Linux, in order to meet these requirements it is necessary to install a realtime kernel such as RTAI and build the software to run in the special real-time environment. For this reason real-time software runs in kernel-space.

RTAI

Real Time Application Interface, see <https://www.rtai.org/>, the real-time extensions for Linux that LinuxCNC can use to achieve real-time performance.

RTLINUX

See <https://en.wikipedia.org/wiki/RTLinux>, an older real-time extension for Linux that LinuxCNC used to use to achieve real-time performance. Obsolete, replaced by RTAI.

RTAPI

A portable interface to real-time operating systems including RTAI and RTLINUX

RS-274/NGC

The formal name for the language used by LinuxCNC part programs.

Servo Motor

Generally, any motor that is used with error-sensing feedback to correct the position of an actuator. Also, a motor which is specially-designed to provide improved performance in such applications.

Servo Loop

A control loop used to control position or velocity of an motor equipped with a feedback device.

Signed Integer

A whole number that can have a positive or negative sign. In HAL it is known as s32. (A signed 32-bit integer has a usable range of -2,147,483,647 to +2,147,483,647.)

Spindle

The part of a machine tool that spins to do the cutting. On a mill or drill, the spindle holds the cutting tool. On a lathe, the spindle holds the workpiece.

Spindle Speed Override

A manual, operator controlled change in the rate at which the tool rotates while cutting. Often used to allow the operator to adjust for chatter caused by the cutter's teeth. Spindle Speed Override assumes that the LinuxCNC software has been configured to control spindle speed.

Stepconf

An LinuxCNC configuration wizard. It is able to handle many step-and-direction motion command based machines. It writes a full configuration after the user answers a few questions about the computer and machine that LinuxCNC is to run on.

Stepper Motor

A type of motor that turns in fixed steps. By counting steps, it is possible to determine how far the motor has turned. If the load exceeds the torque capability of the motor, it will skip one or more steps, causing position errors.

TASK

The module within LinuxCNC that coordinates the overall execution and interprets the part program.

Tcl/Tk

A scripting language and graphical widget toolkit with which several of LinuxCNCs GUIs and selection wizards were written.

Traverse Move

A move in a straight line from the start point to the end point.

Units

See "Machine Units", "Display Units", or "Program Units".

Unsigned Integer

A whole number that has no sign. In HAL it is known as u32. (An unsigned 32-bit integer has a usable range of zero to 4,294,967,296.)

World Coordinates

This is the absolute frame of reference. It gives coordinates in terms of a fixed reference frame that is attached to some point (generally the base) of the machine tool.

Chapter 23

Legal Section

23.1 Copyright Terms

Copyright (c) 2000-2013 LinuxCNC.org

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and one Back-Cover Text: "This LinuxCNC Handbook is the product of several authors writing for linuxCNC.org. As you find it to be of value in your work, we invite you to contribute to its revision and growth." A copy of the license is included in the section entitled "GNU Free Documentation License". If you do not find the license you may order a copy from Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307

23.2 GNU Free Documentation License

GNU Free Documentation License Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that

could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Chapter 24

Index

—
.axisrc, [39](#)

A

acme screw, [184](#)
Arc Distance Mode, [147](#)
Arc Move, [115](#)
Auto, [66](#), [75](#)
AXIS, [23](#), [36](#)
axis, [184](#)
Axis GUI, [23](#)
AXIS in lathe mode, [37](#)
Axis Menu, [25](#)
Axis Preview Control, [39](#)
AxisUI
 coolant, [32](#)
 feed override, [33](#)
 jog speed, [34](#)
 keyboard shortcuts, [34](#)
 Max Velocity, [34](#)
 MDI, [33](#)
 spindle, [32](#)
 spindle speed override, [33](#)

B

backlash, [184](#)
backlash compensation, [184](#)
backplot, [71](#)
ball nut, [184](#)
ball screw, [184](#)
Block Delete, [98](#)
block delete, [82](#)
break, [174](#)

C

call, [173](#)
Calling Files, [176](#)
CNC, [25](#), [184](#)
CNC Machine Overview, [79](#)
comp, [185](#)
Conditional: if
 elseif
 else, [175](#)

continue, [174](#)
controlled point, [80](#)
coolant, [32](#), [79](#), [81](#)
coordinate measuring machine, [185](#)
Coordinate System, [84](#)

D

Debug Messages, [108](#)
display units, [185](#)
do, [174](#)
DRO, [185](#)
Dwell
 Feed Out, [146](#)
dwell, [81](#)

E

EDM, [185](#)
else, [175](#)
elseif, [175](#)
 else, [175](#)
EMC, [185](#)
EMCIO, [185](#)
EMCMOT, [185](#)
encoder, [185](#)
endif, [175](#)
endsub, [173](#)
endwhile, [174](#)
ESTOP), [28](#)
External Editor, [39](#)

F

F: Set Feed Rate, [177](#)
feed, [185](#)
Feed Out, [145](#), [146](#)
feed override, [33](#), [69](#), [80](#)
feed rate, [81](#), [185](#)
feedback, [185](#)
feedrate override, [185](#)

G

G Code Best Practices, [110](#)
G Code Order of Execution, [109](#)
G Code Overview, [97](#)
G Code Table, [112](#)

G Codes, [112](#)
G-Code, [186](#)
G0 Rapid Move, [113](#)
G1 Linear Move, [114](#)
G10 L1 Tool Table, [122](#)
G10 L10 Set Tool Table, [124](#)
G10 L11 Set Tool Table, [124](#)
G10 L2 Coordinate System, [122](#)
G10 L20 Set Coordinate System, [125](#)
G17 G18 G19 Plane Selection, [125](#)
G2
 G3 Arc Move, [115](#)
G20 Inches, [125](#)
G21 Millimeters, [125](#)
G28, [126](#)
G3 Arc Move, [115](#)
G30, [126](#)
G33 Spindle Synchronized Motion, [127](#)
G33.1 Rigid Tapping, [127](#)
G38.x Probe, [128](#)
G4 Dwell, [119](#)
G40 Cutter Compensation Off, [129](#)
G41 G42 Cutter Compensation, [130](#)
G41.1 G42.1 Dynamic Compensation, [130](#)
G43 Tool Length Offset, [131](#)
G43.1 Dynamic Tool Length Offset, [131](#)
G49 Cancel Tool Length Offset, [132](#)
G5.1 Quadratic B-spline, [120](#)
G5.2 G5.3 NURBs Block, [120](#)
G53 Machine Coordinates, [132](#)
G54-G59.3 Select Coordinate System, [132](#)
G55, [85](#)
G61 G61.1 G64 Path Control, [133](#)
G64 Path Blending, [133](#)
G7 Lathe Diameter Mode, [121](#)
G73 Drilling Cycle Chip Break, [134](#)
G76 Threading, [134](#)
G8 Lathe Radius Mode, [121](#)
G80 Cancel Modal Motion, [140](#)
G80-G89 Canned Cycles, [137](#)
G81 Drilling Cycle, [141](#)
G82 Drilling Cycle Dwell, [144](#)
G83 Peck Drilling, [145](#)
G84 Right-Hand Tapping, [145](#)
G85 Boring
 Feed Out, [145](#)
G86 Boring
 Spindle Stop
 Rapid Move Out, [146](#)
G87 Back Boring, [146](#)
G88 Boring Cycle
 Spindle Stop
 Manual Out, [146](#)
G89 Boring
 Dwell
 Feed Out, [146](#)
G90

 G91 Distance Mode, [146](#)
G91 Distance Mode, [146](#)
G92 Coordinate System Offset, [147](#)
G93
 G94
 G95: Feed Rate Mode, [148](#)
G94
 G95: Feed Rate Mode, [148](#)
G95: Feed Rate Mode, [148](#)
G96
 G97 Spindle Control Mode, [148](#)
G97 Spindle Control Mode, [148](#)
G98
 G99 Canned Cycle Return, [149](#)
G99 Canned Cycle Return, [149](#)
GUI, [184](#), [186](#)

H

HAL, [186](#)
home, [186](#)

I

if, [175](#)
Image to G Code, [179](#)
Indirection, [176](#)
INI, [186](#)
Instance, [186](#)

J

jog, [186](#)
jog speed, [34](#)
joint coordinates, [186](#)

K

keyboard shortcuts, [34](#)
KEYSTICK, [76](#)
kinematics, [186](#)

L

Lathe User Information, [152](#)
lead screw, [186](#)
Line Number, [98](#)
Linear Move, [114](#)
Linux, [6](#)
LinuxCNC User Introduction, [5](#)
Logging, [108](#)
loop, [187](#)
Looping, [174](#)

M

M Codes, [165](#)
M0 Program Pause, [165](#)
M1 Program Optional Pause, [165](#)
M100 to M199 User Defined Commands, [170](#)
M2 Program End, [165](#)
M3 Spindle CW, [166](#)
M30 Program End, [165](#)
M4 Spindle CCW, [166](#)

M48

- M49 Override Control, [167](#)
- M49 Override Control, [167](#)
- M5 Spindle Stop, [166](#)
- M50 Feed Override Control, [167](#)
- M51 Spindle Speed Override, [168](#)
- M52 Adaptive Feed Control, [168](#)
- M53 Feed Stop Control, [168](#)
- M6-Tool-Change, [166](#)
- M60 Pallet Change Pause, [166](#)
- M61 Set Current Tool Number, [168](#)
- M62 to M65 Output Control, [168](#)
- M66 Input Control, [169](#)
- M67 Analog Motion Output Control, [170](#)
- M68 Analog Aux Output Control, [170](#)
- M7 Mist Coolant, [167](#)
- M8 Flood Coolant, [167](#)
- M9 Coolant Off, [167](#)
- machine on, [28](#)
- machine units, [186](#)
- Manual, [31](#), [65](#), [74](#)
- Manual Out, [146](#)
- Manual Tool Change, [36](#)
- Max Velocity, [34](#)
- MDI, [33](#), [186](#)
- Messages, [108](#)
- Mini GUI, [62](#)
- Modal Groups, [106](#)

N

- NGCGUI, [41](#)
- NIST, [186](#)
- NML, [187](#)

O

- O Codes, [173](#)
- offsets, [187](#)
- OpenGL, [23](#)
- operator precedence, [102](#)
- optional block delete, [80](#)
- optional program stop, [80](#), [82](#)
- Other Codes, [177](#)

P

- Parameters, [99](#)
- parameters, [83](#)
- part Program, [187](#)
- Path Control, [133](#)
- path control mode, [82](#)
- Plane Selection, [125](#)
- Polar Coordinates, [104](#)
- preview plot, [30](#)
- Print Messages, [108](#)
- Probe Logging, [108](#)
- program extents, [30](#)
- program units, [187](#)
- Programming the Planner, [16](#)

- Python, [23](#), [36](#)

R

- rapid, [187](#)
- Rapid Move, [113](#)
- Rapid Move Out, [146](#)
- rapid rate, [187](#)
- real-time, [187](#)
- Repeat, [175](#)
- return, [173](#)
- RS274/NGC Programs, [163](#)
- RS274NGC, [187](#)
- RTAI, [187](#)
- RTAPI, [187](#)
- RTLINUX, [187](#)

S

- S: Set Spindle Speed, [177](#)
- servo motor, [187](#)
- Sherline, [63](#)
- Signed Integer, [187](#)
- spindle, [32](#), [79](#), [187](#)
- spindle speed override, [33](#), [80](#)
- Spindle Stop
 - Manual Out, [146](#)
 - Rapid Move Out, [146](#)
- stepper motor, [188](#)
- sub, [173](#)
- Subroutines, [173](#)

T

- T: Select Tool, [177](#)
- TASK, [188](#)
- Tk, [23](#), [188](#)
- tkLinuxCNC, [56](#)
- TkLinuxCNC GUI, [56](#)
- Tool Compensation, [90](#)
- Tool Touch Off, [27](#)
- Tool-Table-Format, [91](#)
- Touch Off, [90](#)
- Touchy GUI, [53](#)
- Trajectory Control, [16](#), [133](#)
- Traverse Move, [188](#)

U

- units, [81](#), [188](#)
- Unsigned Integer, [188](#)
- User Concepts, [16](#)
- User Defined Commands M100-M199, [170](#)
- User Foreword, [3](#)

V

- Virtual Control Panel, [39](#)

W

- while, [174](#)
- Word, [98](#)
- world coordinates, [188](#)