

PORLAND STATE UNIVERSITY

CAPSTONE PROJECT REPORT

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

3D Metal Printer

Members:

Cameron Tribe
Branden Driver
Brian Andrews
Ahmad Qazi

Faculty Supervisor:

Dr. Marek Perkowski

Industry Sponsor:

Aram Kasparov

May 23, 2015



Contents

1 Project Overview	2
2 Project Proposal	2
2.1 Sponsor Proposal	2
2.2 The Goal	3
2.3 Our Starting Point	3
2.4 Requirements	4
3 Schedule	6
4 Hardware	7
4.1 Welder Control	7
4.2 Wire Speed	11
5 Software	12
5.1 G-Code	12
5.2 Control Program	13
5.3 Register Descriptions	15
5.4 GUI Description	16
5.5 The Graphical User Interface (GUI)	16
5.6 Reasons for using GTK+	16
5.7 Examples of GUIs created using GTK+	18
6 Testing	20
7 Photos and Videos of Progress	30
8 Bill of Materials (BOM)	38
Appendix A	39
Appendix B	49
Appendix C	50
Appendix D	50
Appendix E	50
Appendix F	50

1 Project Overview

The team interfaced a CNC machine with a MIG welder to create a 3D metal printer.

2 Project Proposal

2.1 Sponsor Proposal

The company is in the process of constructing an innovative 3D metal printer controlled by CNC (Computer Numerical Control). The project was a combination of two machines:

- CNC mill – (3, 4 or 5 axis CNC mill)
- MIG/TIG welding machine.

The purpose for the CNC motion control (CNC mill) is to program and control motion of the machine, and in this case, the metal deposition process. The purpose for the MIG welder is to deposit liquid metal. Many kinds of wire can be used by the welder to form the parts; carbon steel, titanium, stainless steel or aluminum. The idea for metal deposition and an example that uses a laser can be found at:

https://www.youtube.com/watch?feature=player_embedded&v=s9IdZ2pI5dA

A problem with laser use is its high cost. In this project, the welding machine used cost \$400. Another example can be found here:

<http://www.wired.co.uk/magazine/archive/2014/08/play/steel-sketch>

AKTechnology's plan is to manufacture parts for pump and compressors, and research and develop parts for all sorts of use. The goal is to fabricate low cost and highly usable machines.

The company has CNC PC based CNC mill-motion controller.

<https://www.youtube.com/watch?v=Plf3t7o951U&list=UUlGufPQeEKdN1-50F89Ejig>

<https://www.youtube.com/watch?v=G-jokU7v92E&list=UUlGufPQeEKdN1-50F89Ejig>

<https://www.youtube.com/watch?v=bPQ5UNiGA4c&list=UUlGufPQeEKdN1-50F89Ejig>

The project was to upgrade this CNC motion controller – mill into 3D metal deposition printer by adding a MIG welder instead of a cutting tool spindle. The CNC motion control was reprogrammed. The MIG welder was operational.

The project will also build control to integrate CNC mill and MIG welding machine. The Welder has 2 adjustments - feed of wire and current. A stepper motor is planned to be used to control those analog data for wire feed and power current. The PLC, programmable logic controller, will join the CNC motion controller and the MIG welding machine.

2.2 The Goal

The end goal of this project is to fully integrate the MIG welder with the LinuxCNC system. Integration will include a way to control all of the functions of the welder, i.e. wire speed, maximum current output, engaging and disengaging the welder at appropriate times. In order for this to be done, electromechanical devices must be used to manipulate the knobs on the MIG welder. At the very least, the machine must be able to deposit material, reproducing a simple single object from a CAD drawing. Our aim is to produce a 1" cube. However, it is desired that the machine will be able to create complex structures on a single base. Precision of the deposition is not the primary concern, however it will be a requirement that the total amount of material deposited is more than the minimum tolerance of the part being created. This will allow for material to be machined away to a more precise tolerance.

2.3 Our Starting Point

The groundwork of this project has been completed by Aram Kasparov, the project sponsor. The project at its current state consists of a PC controlled CNC machine, a MIG welder, an infrared temperature sensor and a current measuring sensor. The PC controlling the CNC machine is running a Linux operating system. LinuxCNC an open-source software is used for programing and interfacing with the physical machine. Additional hardware is installed onto the PC, consisting of Mesa Electronics 5I20 FPGA based PCI Anything I/O card, 7i33 analog servo interface card and two 7i37-COM isolated I/O cards. The LinuxCNC software communicates the control signals and receives feedback through these cards. The CNC machine is a 3-axis machine—that is it can move in the X, Y and Z directions. Each axis is moved by a servo-motor and each servo motor is driven by a driver which receives its control commands from the PC. The machine is functional, though the motors will require some tuning and limit switches need to be programmed in (they are physically installed on the machine but not included in the program). The MIG/Flux cored welder is rated at 180 Amp-DC, 240 Volt with a duty cycle of 20% at 140 amps. The welder has current and wire feed adjustment capabilities for controlling the weld. These two knobs will be controlled by two stepper motors which have been installed onto the welder already. The current sensor has the ability to measure up to 225A. It has been demonstrated to be functional and will be used to monitor the current of the weld. The infrared non-contact temperature sensor is rated to measure temperatures up to 1800 degrees Celsius, though no tests have been performed yet.

2.4 Requirements

- Must use a wire feed welder
- Welder must have a Control System
- Must measure weld temperature
- Must measure weld current
- Must use both previous parameters to estimate current quality of weld
- Must use “G code” as inputs
- Must control when material is being deposited
- Must have user interface
- Should allow for welder thermal shutdown
- Should Measure Wire Speed from welder

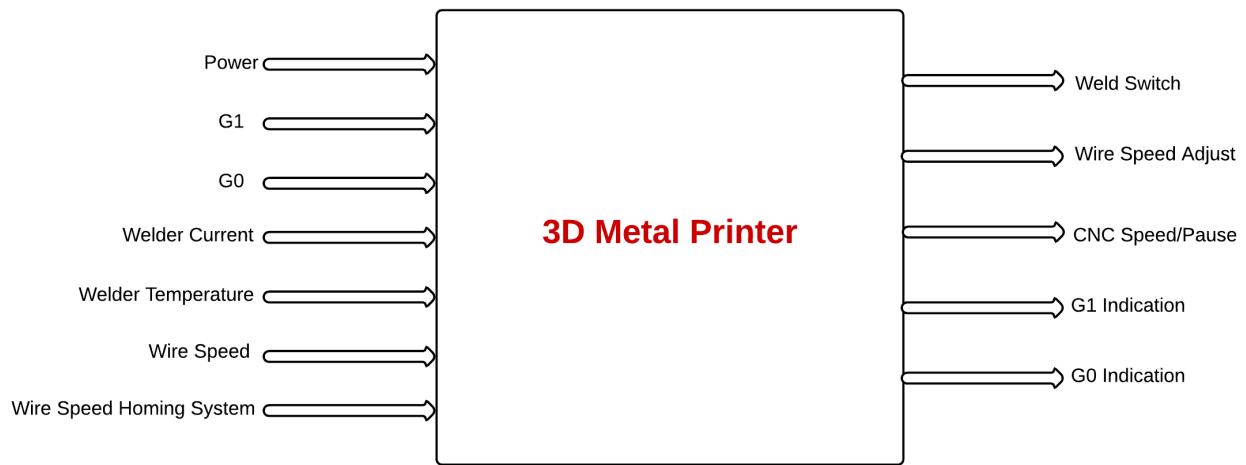


Figure 1: General Black Box Diagram of the 3D Metal Printer

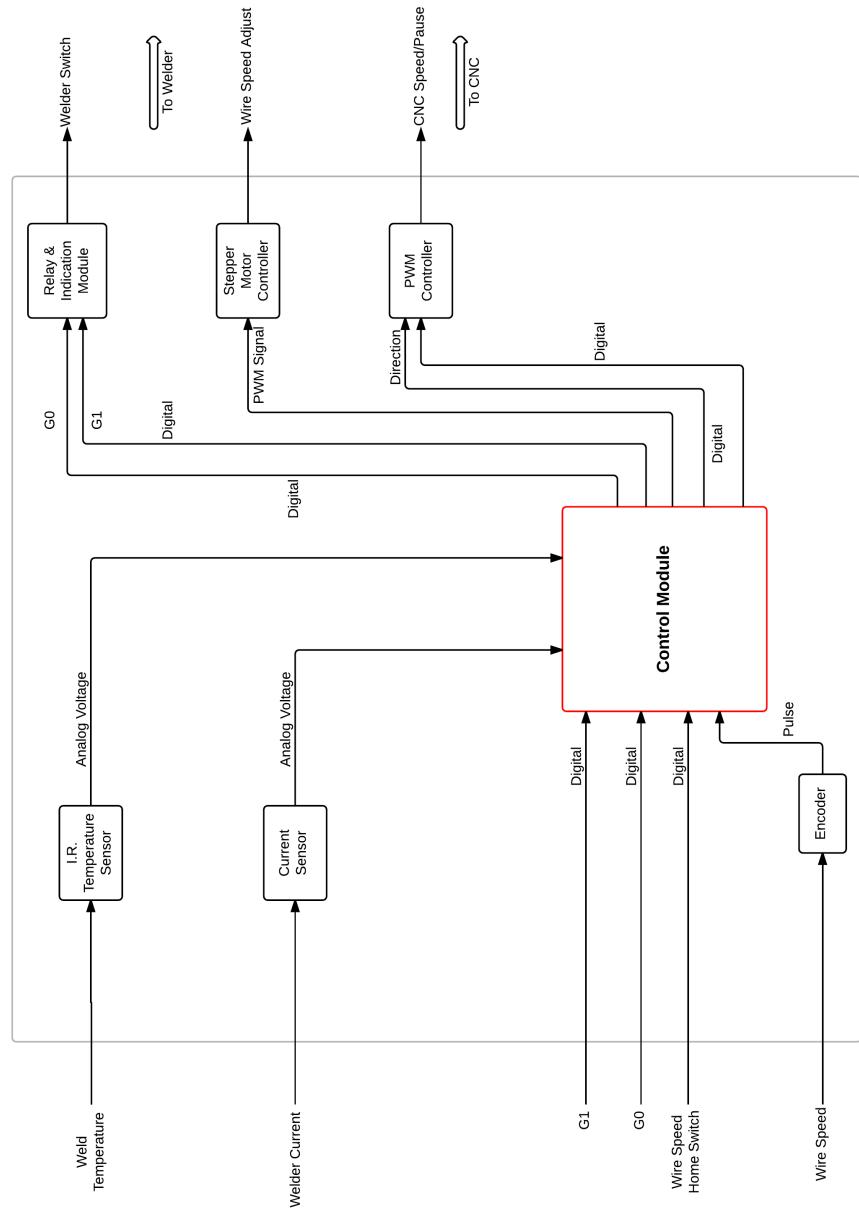


Figure 2: Detailed Black Box Diagram of the 3D Metal Printer

3 Schedule

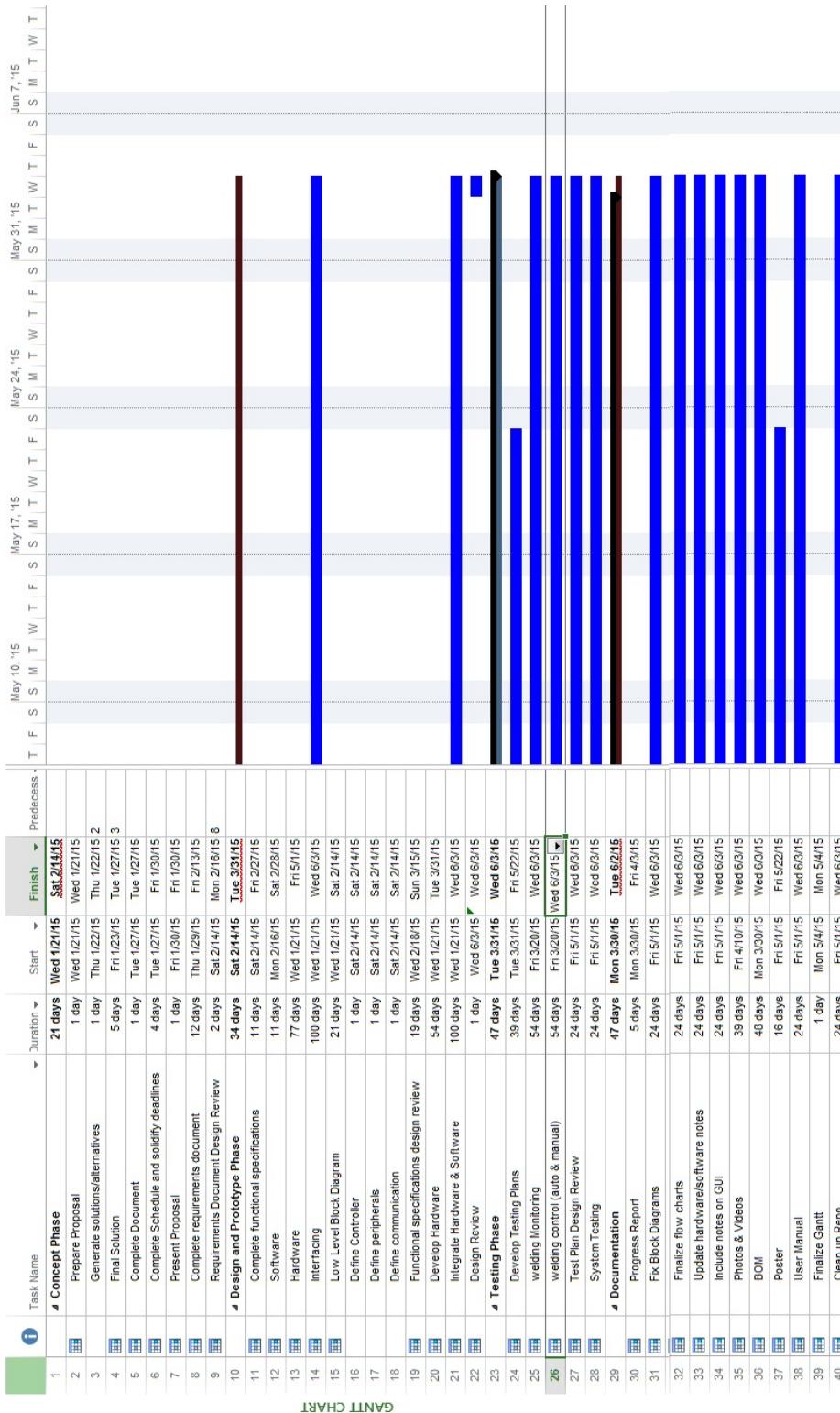


Figure 3: Detailed Black Box Diagram of the 3D Metal Printer

4 Hardware

4.1 Welder Control

To control the welder, a central control module will be used. This was a hot topic of debate for several weeks, as the number of choices available for this project are very high. The sponsor's requirements for the project was that all of the control work was done by a separate computer from the one used by Linux CNC, this only narrowed it down to a choice between a PCIe DAC board and a single board computer. Based on the need for both analog and digital control pins and the need for future expansion, we researched and came up with several options.

Single Board Computers	DAC
Raspberry-Pi	Sensoray 826
Intel Galileo	MCC DAS1602/16
SBC 8600B	
Wander Board Solo	
Beagle Bone Black	

In the end we chose the Sensoray 826 board because for the price it outperforms all other boards on the market by having 16 analog inputs, 8 analog outputs, and 48 digital I/O pins. This board was chosen for the high level of future expandability that it has, and because it is a PCIe card which can be packaged into its own desktop as per request from the sponsor. To control the current to the weld and the wire speed of the welder, two stepper motors have been fitted to the manual control knobs, and are connected to a motor driver module. The Sensoray board will be controlling the motor drivers using a sequence of rising and falling edges. To allow the controller board to control at what time the welder is depositing and when it is not depositing, a relay with a transistor driver will be used.

The signal that tells the controller will be coming from the CNC machine's I/O card. It is a switch type signal which means that when the signal is sent, an internal switch will be closed, causing whatever is on the input to be shown on the output. The CNC machine uses G-Code (described below), and Linux CNC allows outputs to be asserted when a particular G-Code instruction is executed. G1 and G0 are going to be used to tell the I/O card to close and open the switch, which will assert 5V DC to the input to the control module. The control module will assert an output high or low which will open or close the welder switch, turning the welder on and off.

The Sensoray 826 I/O card has three 50 pin connectors and two 26 pin connectors. To allow easy access to these pins, a breakout board with screw terminals has been made so that wires can easily be disconnected and switched.

Power Supply for IC

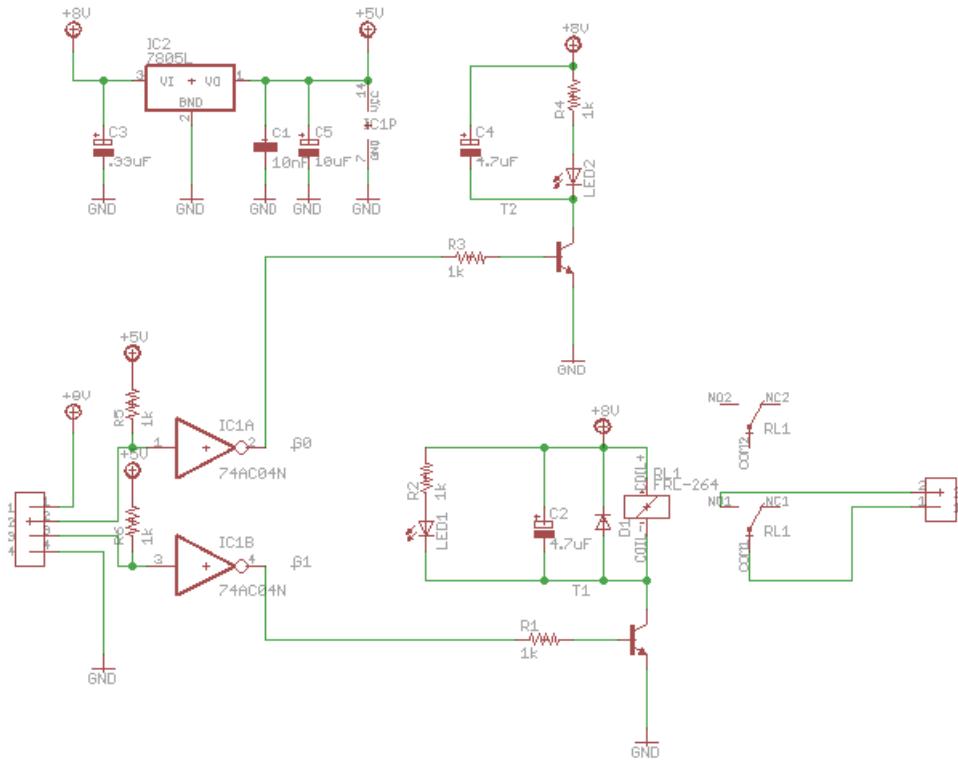


Figure 4: Schematic of Relay and Indication Module

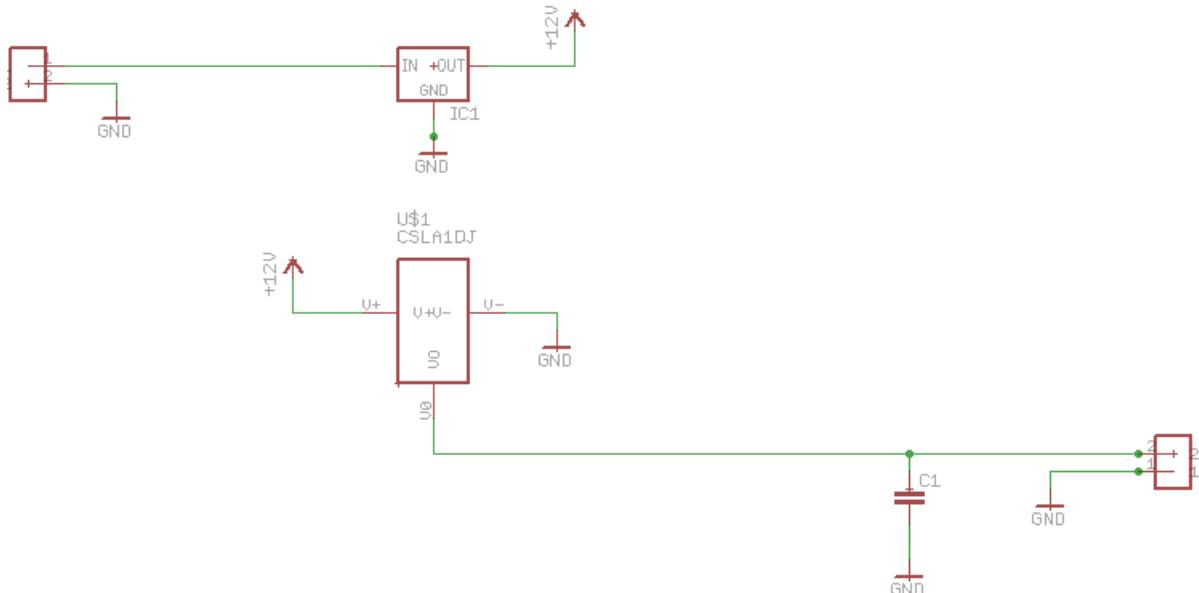


Figure 5: Schematic of the Current Sensor

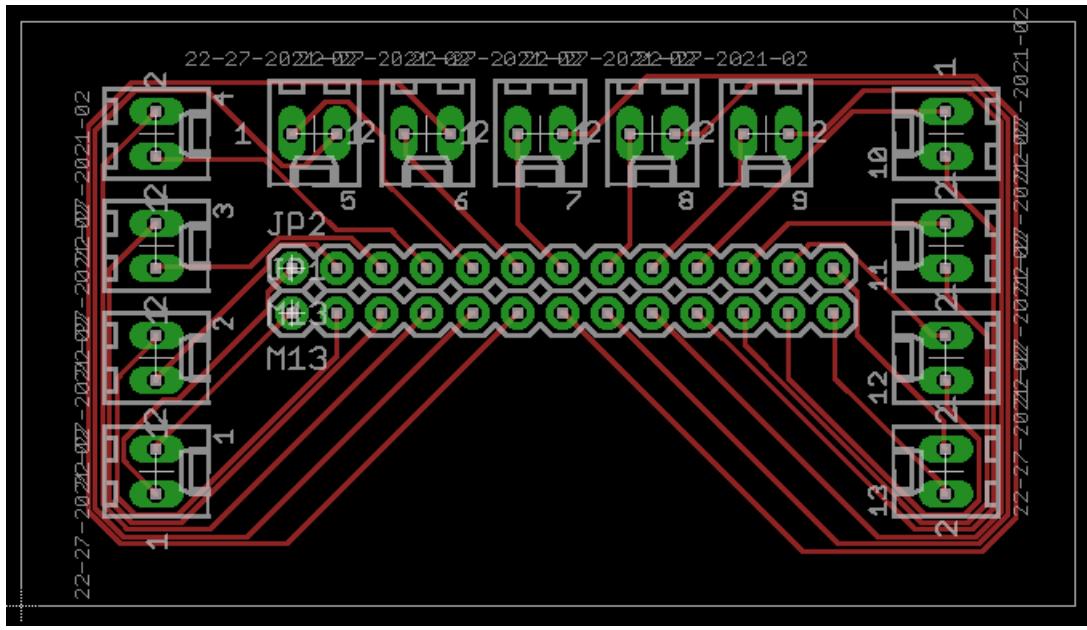


Figure 6: Board Layout of the 26 pin Breakout Board for the Sensoray 826 I/O card

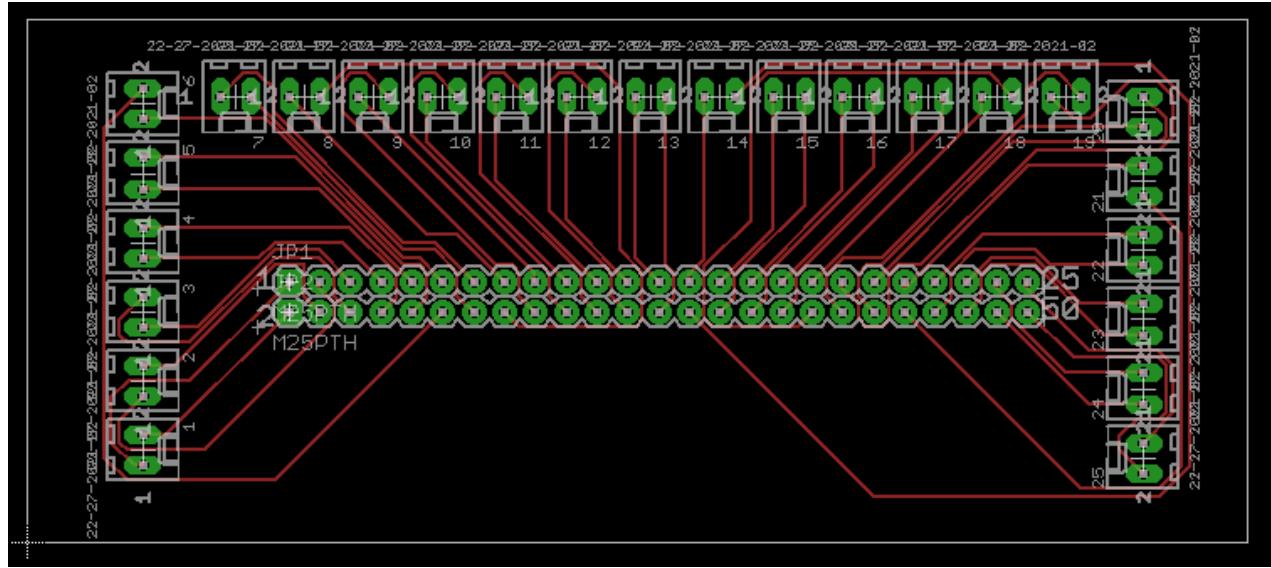


Figure 7: Board Layout of the 50 pin Breakout Board for the Sensoray 826 I/O card

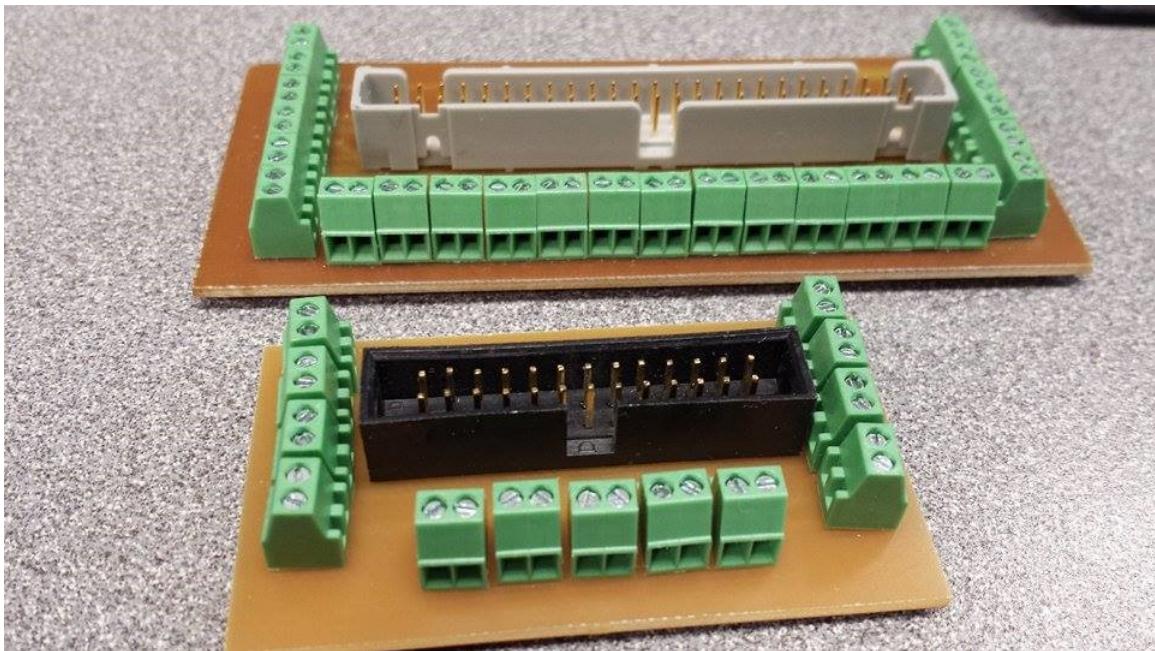
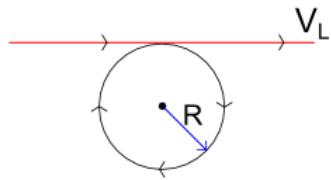


Figure 8: The 26 and 50 pin Breakout Board with connectors for The Sensoray 826 I/O card

4.2 Wire Speed

For calculating Wire Speed with Rotary Encoder



So Angular Velocity is,

$$f = \frac{n}{Nt}$$

n = number of up/down counts, N = number of up/down counts/rev, T = sampling time.

And Linear Velocity is,

$$v_L = \omega r$$
$$\omega = 2\pi f$$

So,

$$V = \frac{\pi n d}{NT}$$

Connections to 826

J ₄	Pin 1	+A0
	Pin 2	-A0
	Pin 3	GND
	Pin 4	+B0
	Pin 5	-B0

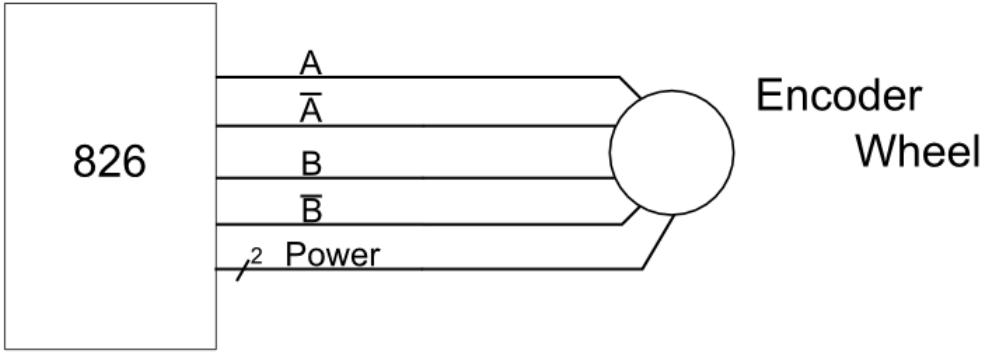


Figure 9: Encoder Connection

5 Software

5.1 G-Code

G-code is the commonly used name to refer to a numerical programming language. As is the case with all languages, G-code has its own syntax and semantics. A line of code is referred to as a block, and a program is defined as multiple blocks. All programs start and end with the percent symbol (%). While writing G-code, the backslash (/) can be used to comment out an entire line, whereas if you want to make comments about a block, parentheses are to be used. Anything inside of a set of parentheses will be ignored by the compiler. As is the case with most other languages, G-code ignores white space, so spacing is used to clarify the code for the writer as well as future users.

All points of G code are comprised of words and numbers. A word is simply a letter. For example, the block “X0” is simply the word X and the value 0. The words X, Y, and Z refer to the three axes, while the G words refer to the movement, motion, and location. When first started up, any blocks of code will use the point (0,0,0) as home, however G54 establishes a new temporary “home” point and G52 establishes the point where the temporary reference point is to be set. In other words, the block of code “G54 G52 X100 Y100 Z0” changes the reference point from (0,0,0) to (100,100,0) and the remainder of the code will run from this point, unless a new reference point is defined.

Similarly, other G-code words can be used to define how the space between two points should be interpolated. In some instances you may want two points to be connected via a straight line, while at other times it would be better to have them connected via a circular pattern. When dealing with circular interpolation, you can set it to be done in either a clockwise or counterclockwise manner. Beyond just linear and circular interpolation, there are dozens of G-code words that determine how the code is to be run (Appendix C).

5.2 Control Program

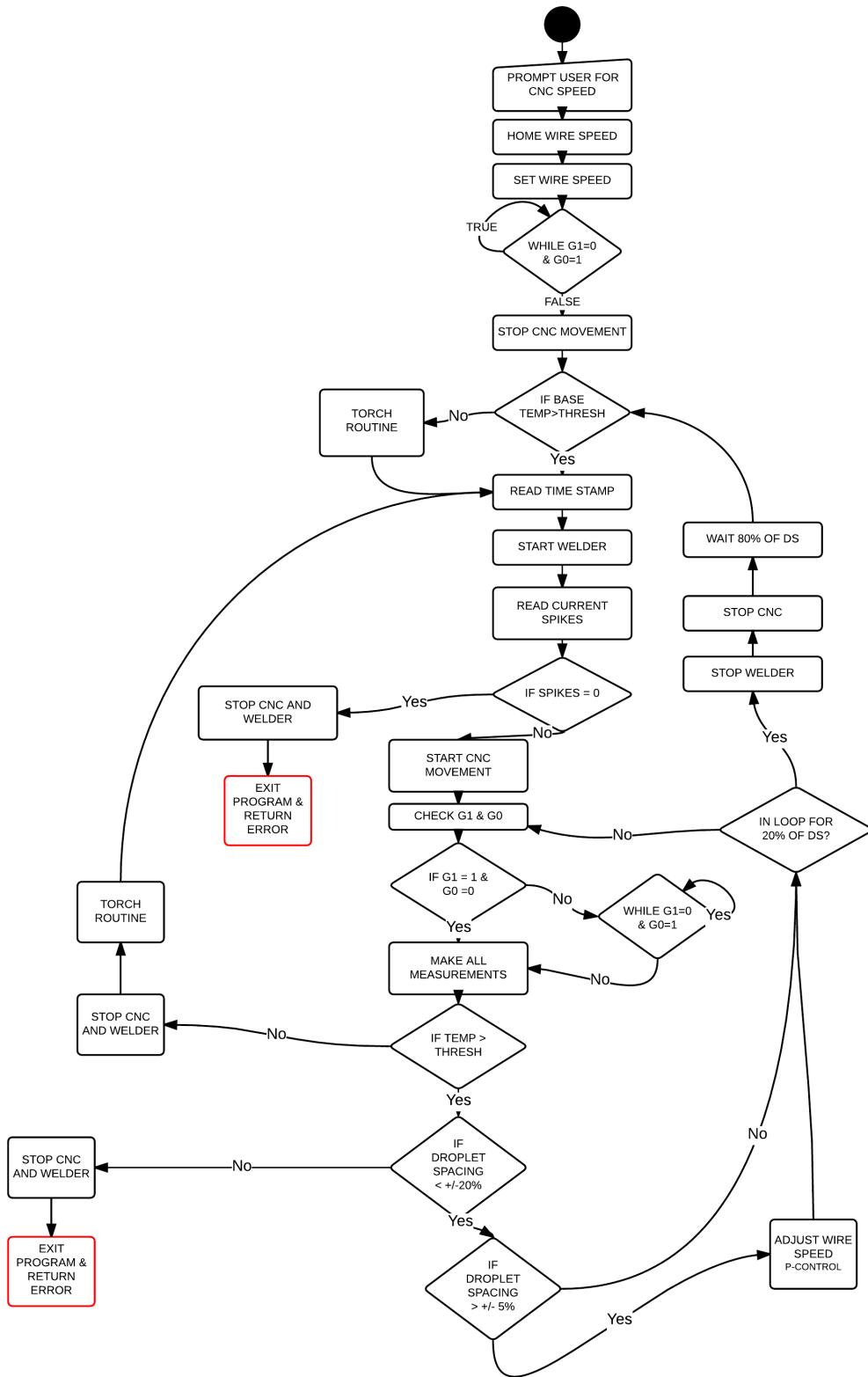


Figure 10: Control Program Flowchart

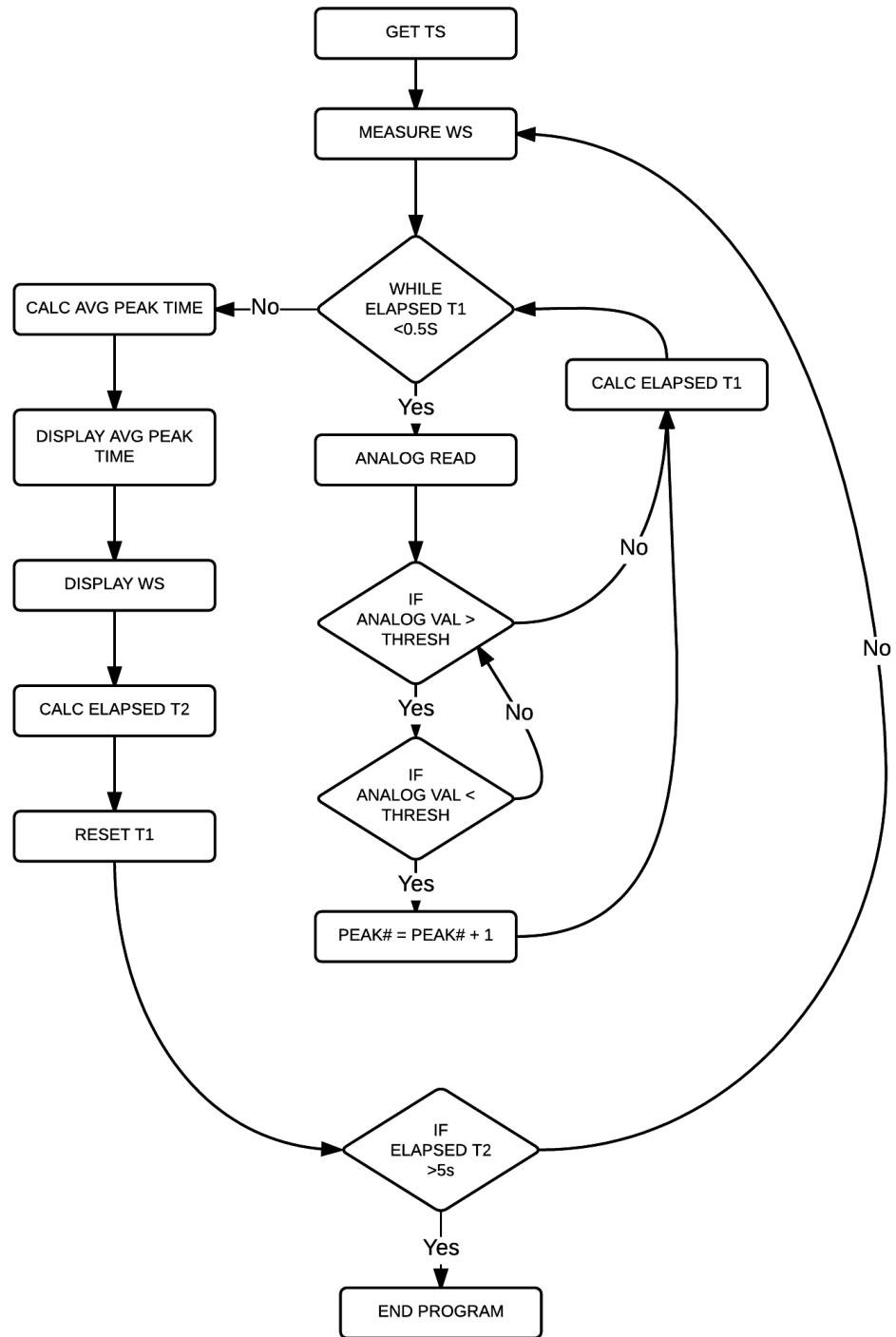


Figure 11: Wire Feed and Droplet Spacing Test Flowchart

5.3 Register Descriptions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0	IP	IM	0	0	0	TP	NR	UD	BP	OM	OP	TP	TE	TD	K	XS																	
0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	0		1		6		8		2		0		2		0		2		0														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	
	0		0		0		0		0		8		0		0		8		0		0		0		0		0		0		A		

Table 1: Counter Mode Register Description

where,

- Row 3 - BIN VAL For PWM Mode
- Row 4 - HEX VAL
- Row 5 - BIN VAL For Frequency Measurement Mode
- Row 6 - HEX VAL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47
								x		C		x		x		x		x		x		x		x		x		x		x	
								x		3		x		x		x		x		x		x		x		x		x		x	
								x 1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
								x 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
								x x 0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
								1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Table 2: DIO[0] Register Description

where,

- Row 1 - Bit
- Row 2 - Channel
- Row 3 - Pin
- Row 4 - G1/G0 Read Mask
- Row 5 - G1/G0 Write Mask
- Row 6 - Motor CCW
- Row 7 - Motor CW
- Row 8 - Wire Speed Home Switch
- Row 9 - Weld and CNC stop

5.4 GUI Description

Our software will have a GUI in which the user can see real-time graphed data coming from the current and temperature sensors as well as see the current wire speed. The system should use this feedback to automatically adjust the weld and keep it in a state that can be considered a "good weld". Also included here will be manual overrides for the user to adjust the current and wire speed to their own desired result. Fig. 10 shows an initial GUI layout that we will be aiming for.

5.5 The Graphical User Interface (GUI)

We are using GTK+ in C programming language to generate the Graphical User Interface in order to view the different data and also control different settings.

The GTK+ is a library for creating graphical user interfaces. The library is created in C programming language. The GTK+ library is also called the GIMP toolkit. Originally, the library was created while developing the GIMP image manipulation program. Since then, the GTK+ became one of the most popular toolkits under Linux and BSD Unix. Today, most of the GUI software in the open source world is created in Qt or in GTK+. The GTK+ is an object oriented application programming interface. The object oriented system is created with the Glib object system, which is a base for the GTK+ library. The GObject also enables to create language bindings for various other programming languages. Language bindings exist for C++, Python, Perl, Java, C#, and other programming languages.

The GTK+ itself depends on the following libraries.

- Glib
- Pango
- ATK
- GDK
- GdkPixbuf
- Cairo

*Note: For detailed functions of each library refer to Appendix A

5.6 Reasons for using GTK+

- Language Bindings

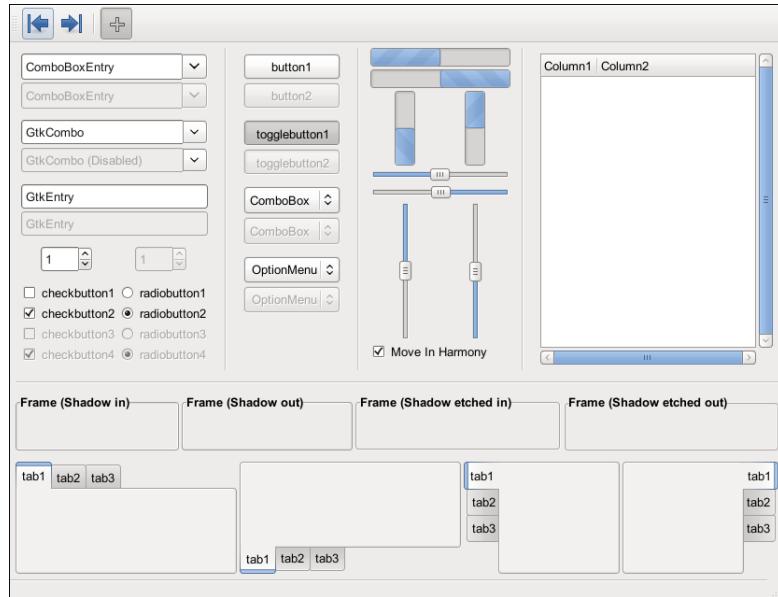
GTK+ is available in many other programming languages thanks to the language bindings available. This makes GTK+ quite an attractive toolkit for application development.

- Interfaces

GTK+ has a comprehensive collection of core widgets and interfaces for use in your application.

- Windows (normal window or dialog, about and assistant dialogs)
- Displays (label, image, progress bar, status bar)
- Buttons and toggles (check buttons, radio buttons, toggle buttons and link buttons)
- Numerical (horizontal or vertical scales and spin buttons) and text data entry (with or without completion)
- Multi-line text editor
- Tree, list and icon grid viewer (with customizable renderers and model/view separation)
- Combo box (with or without an entry)
- Menus (with images, radio buttons and check items)
- Toolbars (with radio buttons, toggle buttons and menu buttons)
- GtkBuilder (creates your user interface from XML)
- Selectors (color selection, file chooser, font selection)
- Layouts (tabulated widget, table widget, expander widget, frames, separators and more)
- Status icon (notification area on Linux, tray icon on Windows)
- Printing widgets
- Recently used documents (menu, dialog and manager)

5.7 Examples of GUIs created using GTK+



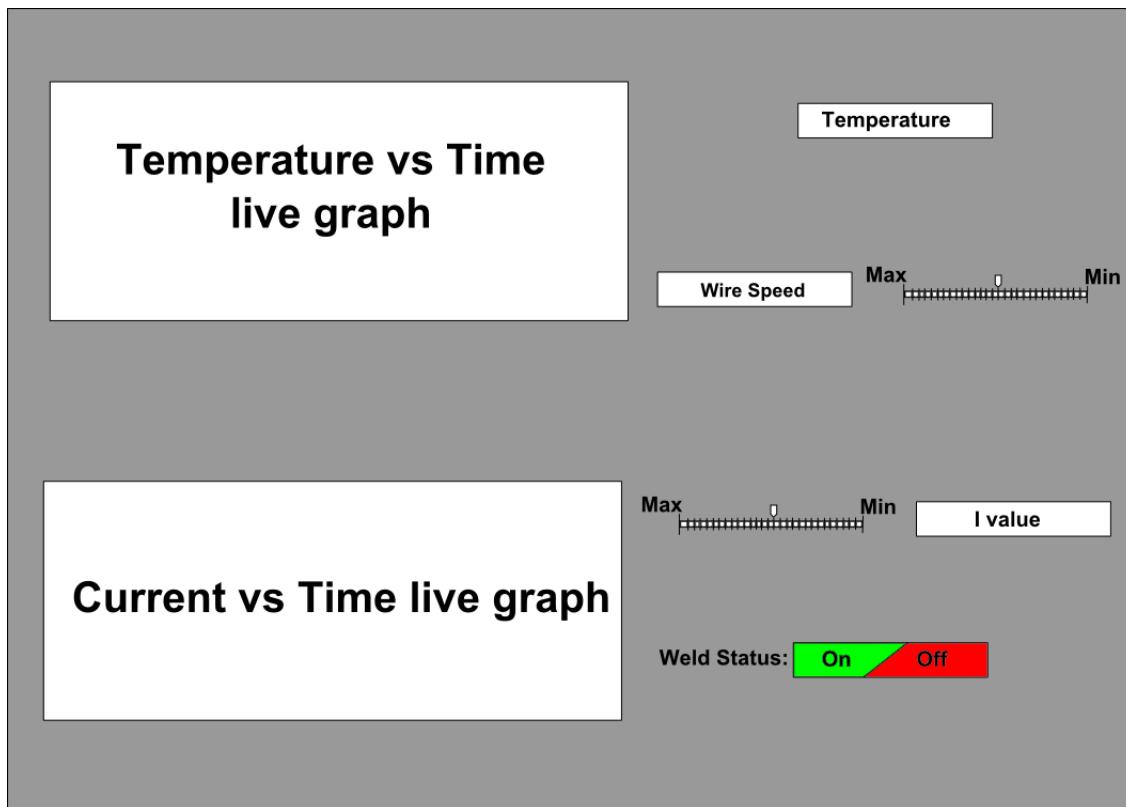


Figure 12: Proposed Rough GUI Layout

6 Testing

aaaaaaaaaa

Test Case # 1: CNC Confirmation Test

Test Writers: Cameron Tribe					
Test Case Name:		CNC Confirmation Test #1		Test ID #:	3MP-CNC-01
Description:		This test is to simply confirm that the CNC is operational in all three dimensions.		Type:	
Tester Information					
Name of Tester:				Date:	05/30/2015
Hardware Ver:				Time:	
Setup:		Only Welder will be used, not CNC. Remove Cover of welder to ensure wire feed is properly set up. Welding will not be taking place, so be sure ground wire is not connected.			
Step	Action	Expected Result	Pass	Fail	N/A
1	Load G-Code into Linux CNC and run initial homing procedure				
2	Place paper on base				
3	Fix felt marker onto CNC machine at an appropriate height so that it will draw on the paper.				
4	Run CNC Machine	This test verified that the CNC machine was working properly. Fig 14 shows the reproduced image that was created by the CNC Machine.			
Overall Test Result:					

Test Case # 4: Wire Speed Test

Test Writers: Branden Driver					
Test Case Name:		Wire Speed Encoder Test #1		Test ID #:	3MP-Wire-01
Description:		Measures the actual wire speed in correlation with the wire speed encoder. This test will allow for the construction of the lookup table to be used with the main program.		Type:	
Tester Information					
Name of Tester:				Date:	
Hardware Ver:		Display V1.0, Main Board V1.5, Sensor V1.0		Time:	
Setup:		Only Welder will be used, not CNC. Remove Cover of welder to ensure wire feed is properly set up. Welding will not be taking place, so be sure ground wire is not connected.			
Step	Action	Expected Result	Pass	Fail	N/A
1	Open wire feed program	Program should open			
2	Set program to run for one second	program should only run for one second			
3	Cut wire extruded from nozzle as short as possible	Very little wire should be showing			
4	Set wire feed speed nozzle to home setting	Wire feed should be at lowest setting			
5	Turn on welder	Welder should turn on			
6	Run test program	Welder should feed wire for exactly one second			
7	Turn off welder	Welder should turn off			
8	Cut wire extruded from nozzle as short as possible	Very little wire should be showing			
9	Measure length of cut wire and record value	Value determined for specified wire feed setting			
10	Repeat steps 1-9 for various values of wire speed	Determine if wire feed is linear. If so, interpolate for all wire feed speeds			
Overall Test Result:					

Wire Speed (in/s)	Time (sec)	Expected Length (in)	Measured Length (in)
0.88	3	2.64	2.56
0.90	3	2.70	2.46
2.00	3	6.00	6.06
3.20	3	9.60	10.10
5.50	3	16.50	16.80
7.60	3	22.80	23.20
9.50	3	28.50	28.06

Table 3: Linear Velocity Program Test

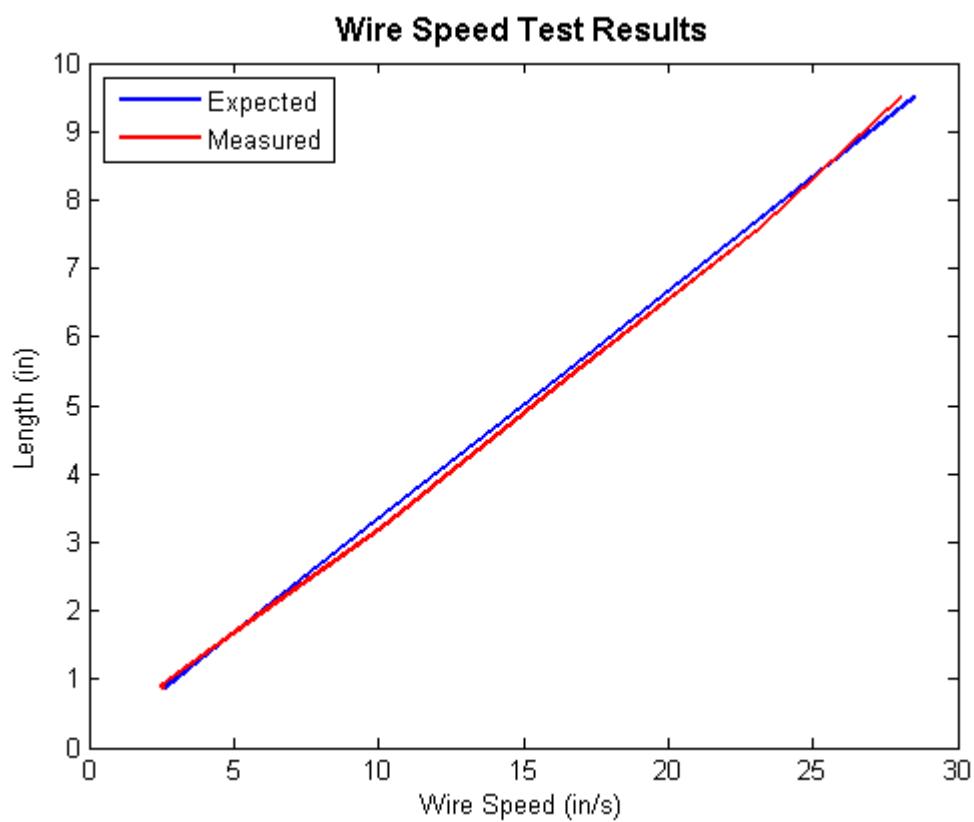


Figure 13: Proposed Rough GUI Layout

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
4.21	2	19047
4.04	3	20619
3.88	4	22857
3.98	5	21978
4.14	6	20833
3.88	7	20943
3.95	8	18872
6.02	2	13119
5.81	3	15037
6.19	4	14084
6.06	5	12539
5.71	6	15504
5.6	7	10257
6.54	8	17621
7.5	2	12820
7.2	3	14035
7.61	4	14760
7.67	5	15267
7.61	6	17167
7.45	7	11594
6.95	8	10790
8.99	2	11364
8.82	3	13114
9.58	4	11940
10	5	11765
8.5	6	10106
9.47	7	12751
9.84	8	n/a
11	2	10582
10.86	3	9557
10.65	4	10811
10.03	5	12270
11.05	6	11007
11.49	7	12012
11.85	8	11695

Table 4: Plate 1

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
3.48	2	20014
3.47	3	23121
3.3	4	23530
3.3	5	27398
2.96	6	26845
3.39	7	29646
3.42	8	30534
4.07	2	24342
4.19	3	19802
3.95	4	17467
3.86	5	18868
4.14	6	14545
4.09	7	13333
3.99	8	19900
6.55	2	11267
6.8	3	11173
7.23	4	12012
6.49	5	9788
6.38	6	9389
6.3	7	9216
6.68	8	9009
9.8	2	9442
9.57	3	8928
10.28	4	9456
10.89	5	9280
9.89	6	9204
9.83	7	8050
10.01	8	9456
11.49	2	8677
11.46	3	8340
11.43	4	8752
11.75	5	8798
11.6	6	9302
11.53	7	9029
11.69	8	9132

Table 5: Plate 2

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
5.34	2	12658
4.9	3	14053
4.97	4	12317
5.18	5	16461
4.53	6	11019
4.83	7	79756
n/a	8	n/a
7.12	2	9876
6.93	3	9959
7.22	4	10315
6.82	5	9527
6.74	6	9195
6.8	7	9227
6.87	8	9852
9.3	2	8948
9.19	3	9091
9.42	4	9599
9.26	5	9466
9.31	6	9204
n/a	7	n/a
9.16	8	8180
12.58	2	8421
12.41	3	8445
12.77	4	8465
12.74	5	8792
12.56	6	8620
12.66	7	8602
12.82	8	8714
14.67	2	8477
15.19	3	8585
14.87	4	8756
14.69	5	8547
14.89	6	8677
14.32	7	8639
14.65	8	8403

Table 6: Plate 3

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
3.83	4	9013
3.85	4.5	9828
3.89	5	9780
4	5.5	11662
3.96	6	9552
3.97	6.5	10447
3.8	7	10371
4.66	4	9860
5.07	4.5	9780
4.61	5	9569
4.64	5.5	9238
4.55	6	9501
4.59	6.5	9099
4.75	7	9287
4.77	4	9909
4.63	4.5	10032
4.6	5	9029
5.19	5.5	9784
4.56	6	n/a
4.67	6.5	9756
4.65	7	9434
5.35	4	9645
5.37	4.5	10025
5.34	5	9758
5.35	5.5	10554
5.61	6	9546
5.17	6.5	9961
5.36	7	9307
6.1	4	8877
5.98	4.5	9412
6.09	5	9615
6.03	5.5	9350
6.76	6	9863
6.35	6.5	10474
6.15	7	9198

Table 7: Plate 4

Wire Speed (in/s)	CNC Speed (in/min)	Avg Droplet Spacing (us)
4.09	4.8	9195
4.47	5	9737
4.28	5.2	10126
3.39	5.4	9623
4.25	5.6	9740
4.09	5.8	9876
3.98	6	9262
4.61	4.8	9953
4.38	5	9761
4.56	5.2	9389
4.77	5.4	10582
4.56	5.6	9661
4.53	5.8	9625
4.2	6	9183
4.71	4.8	9266
4.61	5	9569
4.94	5.2	9479
4.62	5.4	9376
4.92	5.6	9184
4.89	5.8	9324
4.86	6	9546
5.21	4.8	8994
4.97	5	9111
5.31	5.2	9595
5.21	5.4	9324
5.02	5.6	9117
5.1	5.8	9153
5.26	6	9456
5.37	4.8	9111
5.56	5	9376
5.51	5.2	9456
	5.4	
	5.6	
	5.8	
	6	

Table 8: Plate 5

Run #	Initial Wire Speed (in/s)	Final Wire Speed (in/s)	Turn Amount (deg)	Ratio (deg/(in/s))	Time/deg (us)	Pulse On Time (us)
Run 1	0.3	1.1	30	37.5	22222	666660
Run 2	1.1	1.8	30	42.85714286	22222	666660
Run 3	1.8	2.5	30	42.85714286	22222	666660
Run 4	2.5	3.3	30	37.5	22222	666660
Run 5	1.9	3	45	40.90909091	22222	999990
Run 6	3	4.2	45	37.5	22222	999990
Run 7	1.2	2.3	45	40.90909091	22222	999990
Run 8	2.3	3.4	45	40.90909091	22222	999990
Run 9	0.9	2.4	60	40	22222	1333320
Run 10	2.4	3.9	60	40	22222	1333320
Run 11	3.9	5.5	60	37.5	22222	1333320
Run 12	3.4	5	60	37.5	22222	1333320
Run 13	1	7	235	39.16666667	22222	5222170
			Average	39.66179654		

Table 9: Degree to Turn Measurements

7 Photos and Videos of Progress

Demo of the Printer

<https://www.youtube.com/watch?v=Ypetogtn1Iw>

One of the first things done in this project was to confirm the operation of the CNC machine. To do this, G-Code of a 2D image was uploaded to the machine. A felt marker was used to draw the image below.

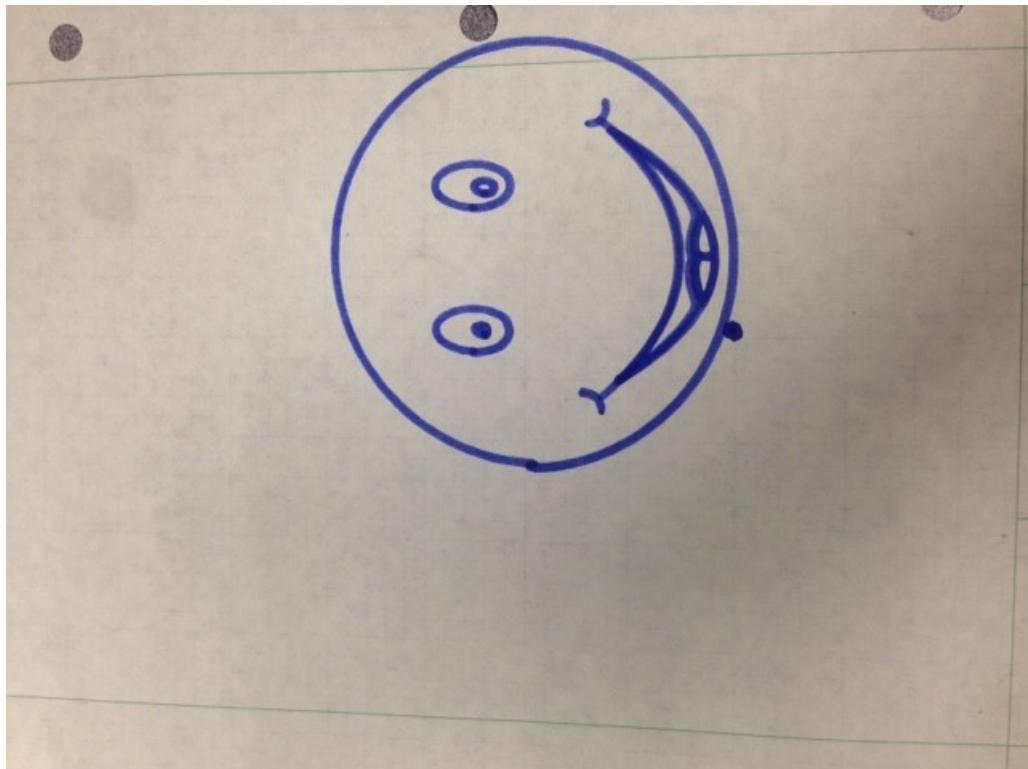


Figure 14: aaaaa

Next, we fitted the welder to the machine, and placed a metal base plate to weld on. To control the welder we just used our hand to activate the weld while the machine was moving.



After this, we connected the relay switch circuit in parallel with the manual welder switch, using G-Code to activate the switch. Shown Below is the result of letting the CNC Machine control the weld.



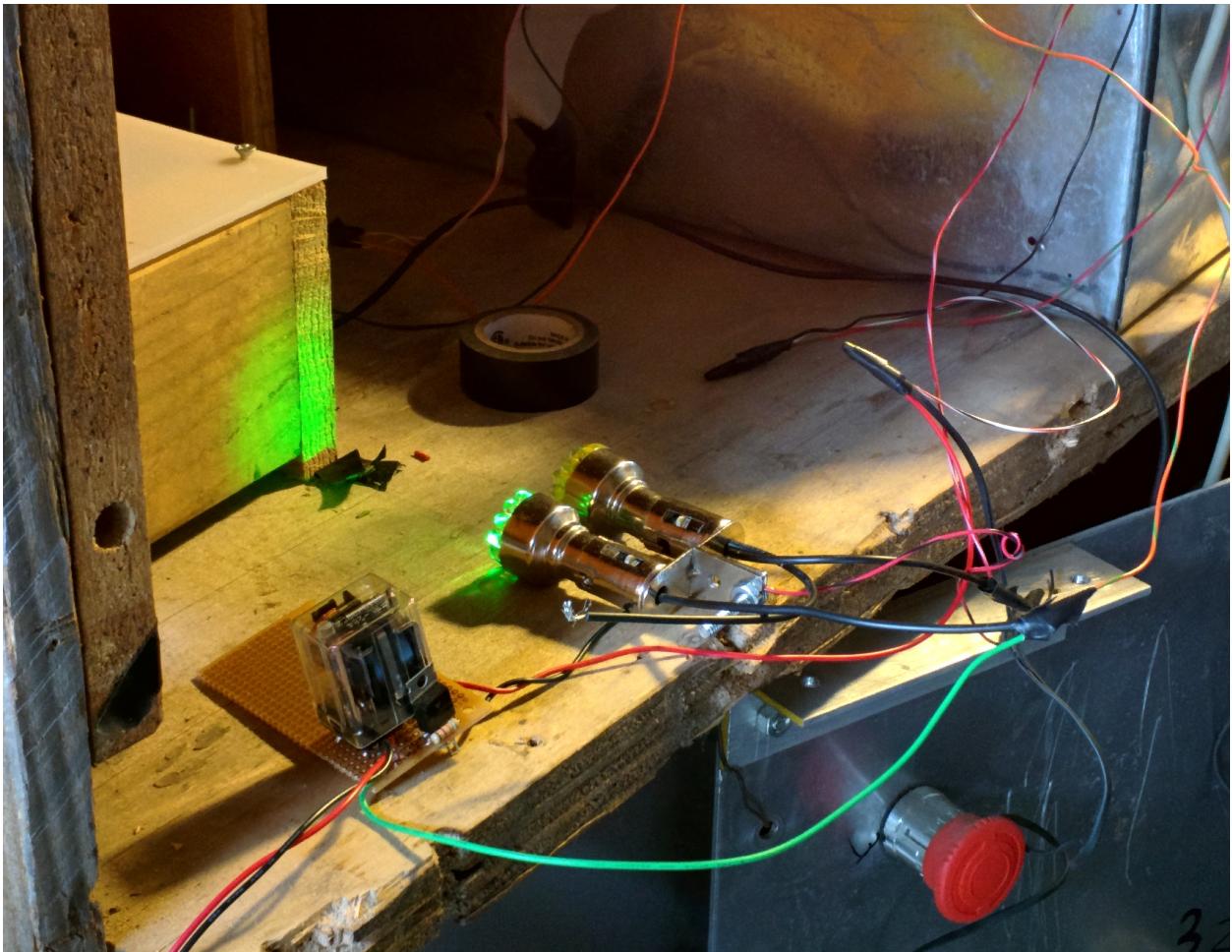


Figure 15: Proposed Rough GUI Layout

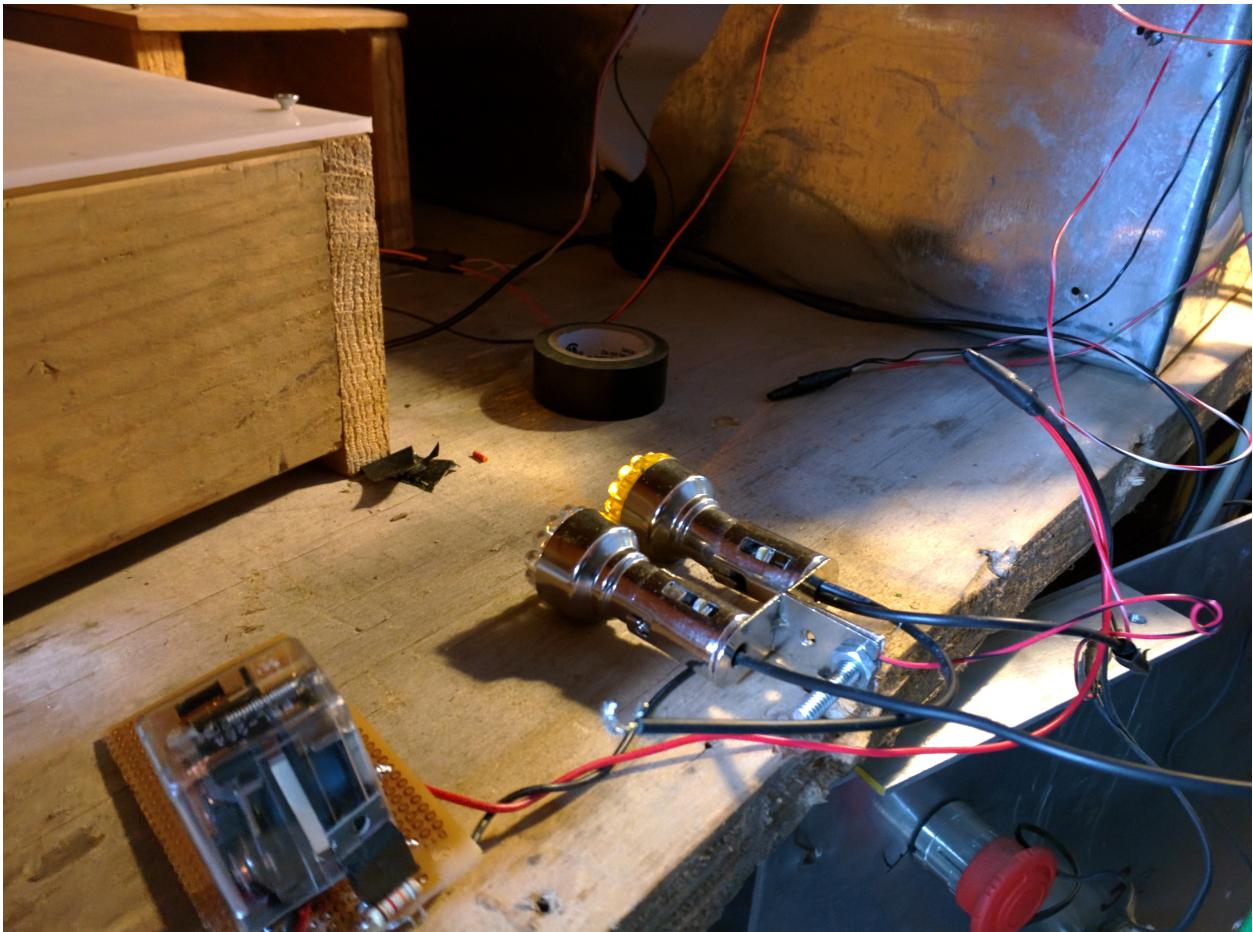


Figure 16: Proposed Rough GUI Layout

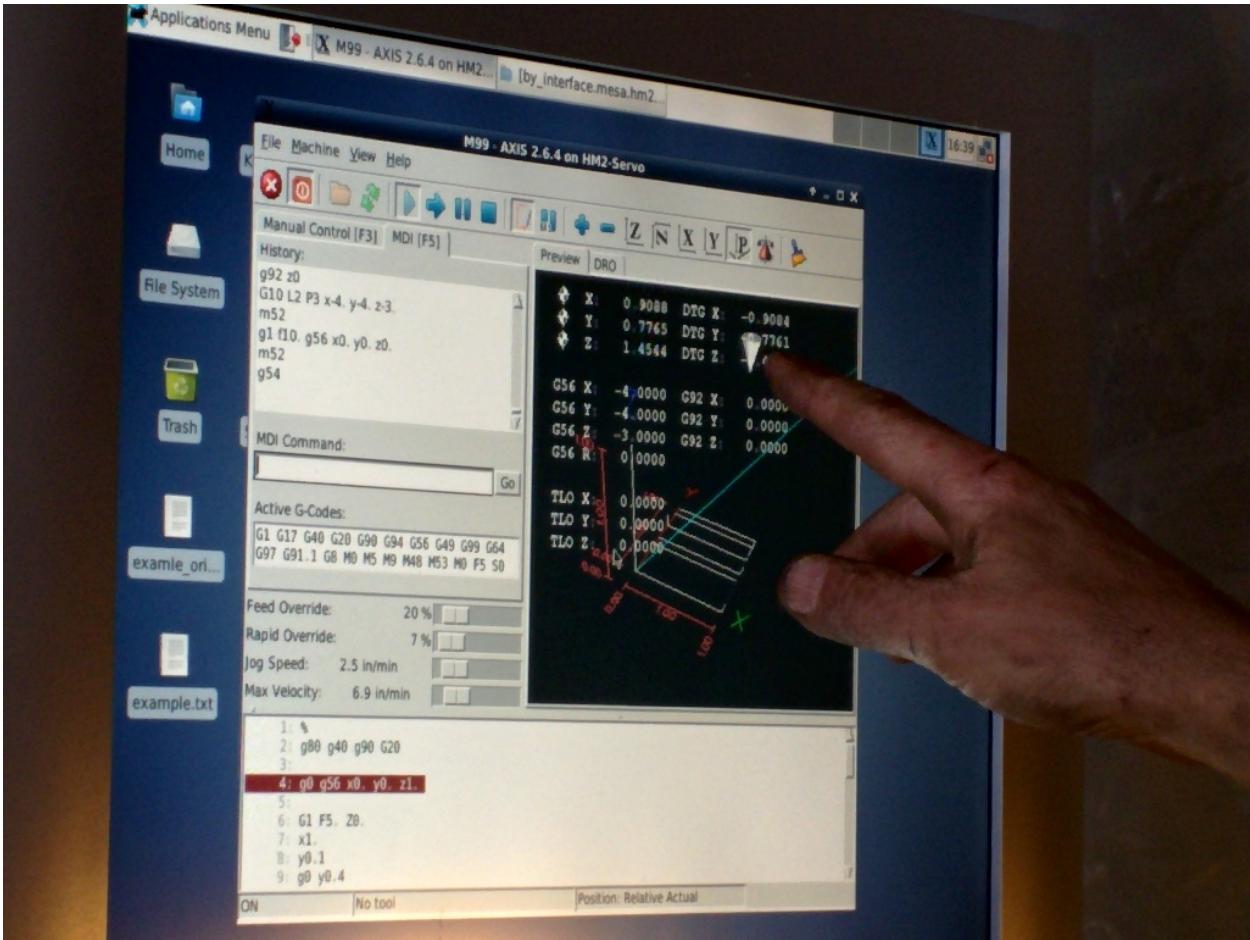


Figure 17: Proposed Rough GUI Layout



Figure 18: Proposed Rough GUI Layout

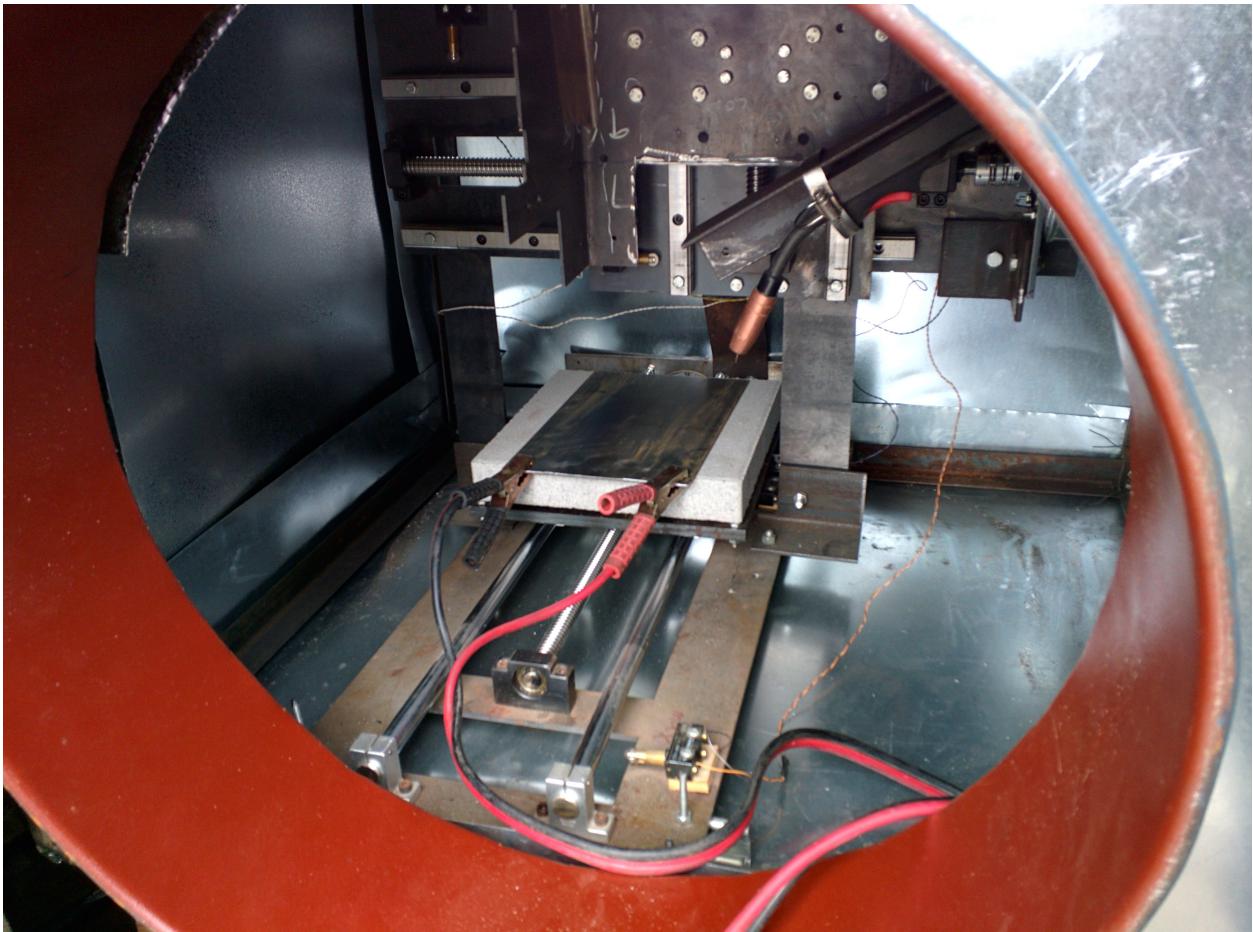


Figure 19: Proposed Rough GUI Layout

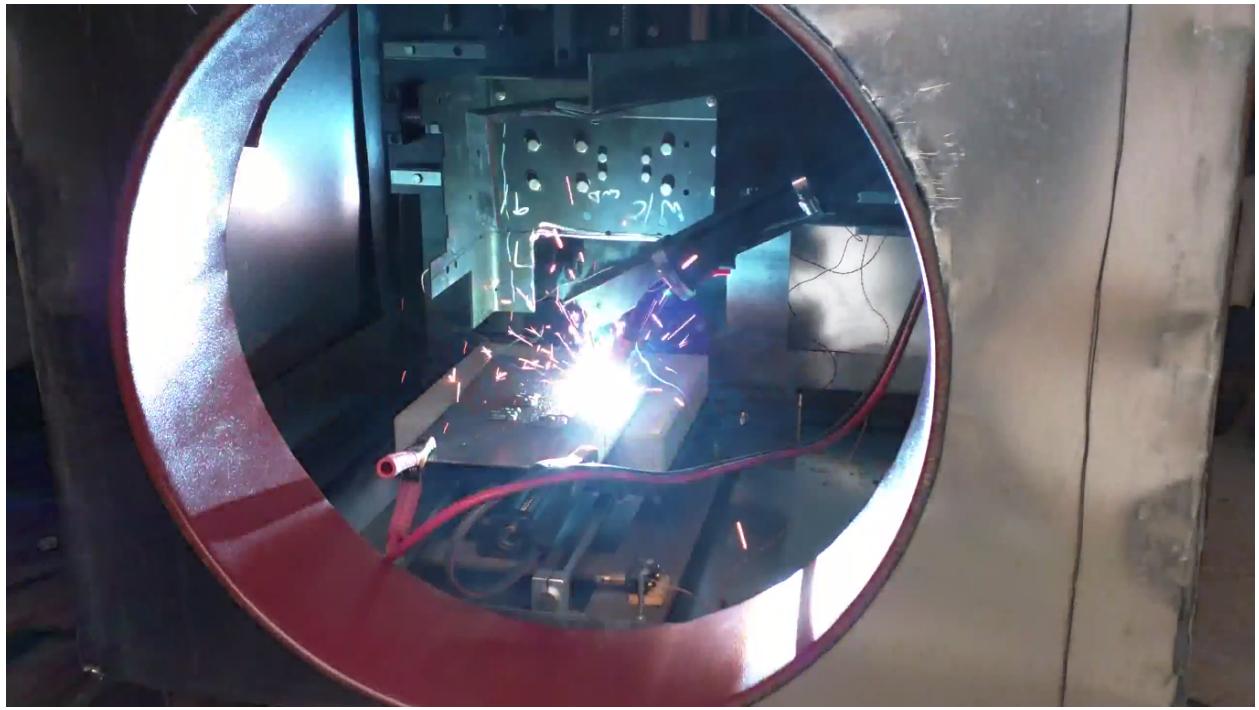


Figure 20: Proposed Rough GUI Layout

8 Bill of Materials (BOM)

Item	Quantity	Part Number	Mfg	Price	Description
CNC Machine					
- X-Stepper Motor	1				Stepper Motor for the X axis
- Y-Stepper Motor	1				Stepper Motor for the Y axis
- Z-Stepper Motor	1				Stepper Motor for the Z axis
Limit and Home Switches	9				
PCI I/O Card	1	5I20	Mesa Electronics		Part of the CNC system
Servo Interface Card	1	7i33	Mesa Electronics		Interface Card for communicating with servo motors
Isolated I/O Card	2	7i37	Mesa Electronics		I/O card used for limit and home switches
Temperature Sensor					
Current Sensor					
MIG Welder			Chicago Electric		
Motor Controller					

Appendix A: GTK+

- **Glib**

GLib provides the core application building blocks for libraries and applications written in C. It provides the core object system used in GNOME, the main loop implementation, and a large set of utility functions for strings and common data structures.

Description

- New types which are not part of standard C (but are defined in various C standard library header files) - gboolean, gsize, gssize, goffset, gintptr, guintptr.
- Integer types which are guaranteed to be the same size across all platforms - gint8, guint8, gint16, guint16, gint32, guint32, gint64, guint64.
- Types which are easier to use than their standard C counterparts - gpointer, gconstpointer, guchar, guint, gushort, gulong.

GLib also defines macros for the limits of some of the standard integer and floating point types, as well as macros for suitableprintf() formats for these types.

Standard Macros — commonly-used macros

Type Conversion Macros — portably storing integers in pointer variables

Byte Order Macros — a portable way to convert between different byte orders

Numerical Definitions — mathematical constants, and floating point decomposition

Miscellaneous Macros — specialized macros which are not used often

Atomic Operations — basic atomic integer and pointer operations

GLib Core Application Support

- **The Main Event Loop** — manages all available sources of events
- **Threads** — portable support for threads, mutexes, locks, conditions and thread private data
- **Thread Pools** — pools of threads to execute work concurrently
- **Asynchronous Queues** — asynchronous communication between threads
- **Dynamic Loading of Modules** — portable method for dynamically loading 'plug-ins'
- **Memory Allocation** — general memory-handling
- **Memory Slices** — efficient way to allocate groups of equal-sized chunks of memory
- **IO Channels** — portable support for using files, pipes and sockets
- **Error Reporting** — a system for reporting errors
- **Message Output and Debugging Functions** — functions to output messages and help debug applications
- **Message Logging** — versatile support for logging messages with different levels of importance

GLib Utilities

- **String Utility Functions** — various string-related functions
- **Character Set Conversion** — convert strings between different character sets
- **Unicode Manipulation** — functions operating on Unicode characters and UTF-8 strings
- **Base64 Encoding** — encodes and decodes data in Base64 format
- **Data Checksums** — computes the checksum for data
- **Secure HMAC Digests** — computes the HMAC for data
- **Internationalization** — gettext support macros
- **Date and Time Functions** — calendrical calculations and miscellaneous time stuff
- **GTimeZone** — a structure representing a time zone
- **GDateTime** — a structure representing Date and Time
- **Random Numbers** — pseudo-random number generator
- **Hook Functions** — support for manipulating lists of hook functions
- **Miscellaneous Utility Functions** — a selection of portable utility functions
- **Lexical Scanner** — a general purpose lexical scanner
- **Timers** — keep track of elapsed time
- **Spawning Processes** — process launching
- **File Utilities** — various file-related functions
- **URI Functions** — manipulating URIs
- **Hostname Utilities** — Internet hostname utilities
- **Shell-related Utilities** — shell-like cmdline handling
- **Commandline option parser** — parses commandline options
- **Glob-style pattern matching** — matches strings against patterns containing '*' (wildcard) and '?' (joker)
- **Perl-compatible regular expressions** — matches strings against regular expressions
- **Regular expression syntax** — syntax and semantics of regular expressions supported by GRegex
- **Simple XML Subset Parser** — parses a subset of XML
- **Key-value file parser** — parses .ini-like config files
- **Bookmark file parser** — parses files containing bookmarks
- **Testing** — a test framework
- **UNIX-specific utilities and integration** — pipes, signal handling

- **Windows Compatibility Functions** — UNIX emulation on Windows

GLib Data Types

- **Doubly-Linked Lists** — linked lists that can be iterated over in both directions
- **Singly-Linked Lists** — linked lists that can be iterated in one direction
- **Double-ended Queues** — double-ended queue data structure
- **Sequences** — scalable lists
- **Trash Stacks** — maintain a stack of unused allocated memory chunks
- **Hash Tables** — associations between keys and values so that given a key the value can be found quickly
- **Strings** — text buffers which grow automatically as text is added
- **String Chunks** — efficient storage of groups of strings
- **Arrays** — arrays of arbitrary elements which grow automatically as elements are added
- **Pointer Arrays** — arrays of pointers to any type of data, which grow automatically as new elements are added
- **Byte Arrays** — arrays of bytes
- **Balanced Binary Trees** — a sorted collection of key/value pairs optimized for searching and traversing in order
- **N-ary Trees** — trees of data with any number of branches
- **Quarks** — a 2-way association between a string and a unique integer identifier
- **Keyed Data Lists** — lists of data elements which are accessible by a string or GQuark identifier
- **Datasets** — associate groups of data elements with particular memory locations
- **GVariantType** — introduction to the GVariant type system
- **GVariant** — strongly typed value datatype
- **GVariant Format Strings** — varargs conversion of GVariants
- **GVariant Text Format** — textual representation of GVariants

Deprecated APIs

- **Deprecated thread API** — old thread APIs (for reference only)
- **Caches** — caches allow sharing of complex data structures to save resources
- **Relations and Tuples** — tables of data which can be indexed on any number of fields
- **Automatic String Completion** — support for automatic completion using a group of target strings

GLib Tools

- **glib-gettextize** — gettext internationalization utility
- **gtester** — test running utility
- **gtester-report** — test report formatting utility

• Pango

Pango is a library for laying out and rendering of text, with an emphasis on internationalization. Pango can be used anywhere that text layout is needed, though most of the work on Pango so far has been done in the context of the GTK+ widget toolkit. Pango forms the core of text and font handling for GTK+-2.x.

Pango is designed to be modular; the core Pango layout engine can be used with different font backends. There are three basic backends, with multiple options for rendering with each.

- Client side fonts using the FreeType and fontconfig libraries, using HarfBuzz for complex-text handling. Rendering can be with Cairo or Xft libraries, or directly to an in-memory buffer with no additional libraries.
- Native fonts on Microsoft Windows using Uniscribe for complex-text handling. Rendering can be done via Cairo or directly using the native Win32 API.
- Native fonts on MacOS X using CoreText for complex-text handling, rendering via Cairo.

The integration of Pango with Cairo provides a complete solution with high quality text handling and graphics rendering.

Dynamically loaded modules then handle text layout for particular combinations of script and font backend. Pango ships with a wide selection of modules, including modules for Hebrew, Arabic, Hangul, Thai, and a number of Indic scripts. Virtually all of the world’s major scripts are supported. As well as the low level layout rendering routines, Pango includes PangoLayout, a high level driver for laying out entire blocks of text, and routines to assist in editing internationalized text. Pango depends on 2.x series of the GLib library.

• ATK

ATK provides the set of accessibility interfaces that are implemented by other toolkits and applications. Using the ATK interfaces, accessibility tools have full access to view and control running applications.

Base accessibility object

- **AtkObject** — The base object class for the Accessibility Toolkit API.

Event and Toolkit Support

- **AtkUtil** — A set of ATK utility functions for event and toolkit support

ATK Interfaces

- **AtkAction** — The ATK interface provided by UI components which the user can activate/interact with.
- **AtkComponent** — The ATK interface provided by UI components which occupy a physical area on the screen. which the user can activate/interact with.
- **AtkDocument** — The ATK interface which represents the toplevel container for document content.
- **AtkEditableText** — The ATK interface implemented by components containing user-editable text content.
- **AtkHyperlinImpl** — An interface from which the AtkHyperlink associated with an AtkObject may be obtained.
- **AtkHypertext** — The ATK interface which provides standard mechanism for manipulating hyperlinks.
- **AtkImage** — The ATK Interface implemented by components which expose image or pixmap content on-screen.
- **AtkSelection** - The ATK interface implemented by container objects whose AtkObject children can be selected.
- **AtkStreamableContent** — The ATK interface which provides access to streamable content.
- **AtkTable** — The ATK interface implemented for UI components which contain tabular or row/column information.
- **AtkTableCell** - The ATK interface implemented for a cell inside a two-dimentional AtkTable
- **AtkText** — The ATK interface implemented by components with text content.
- **AtkValue** — The ATK interface implemented by valiators and components which display or select a value from a bounded range of values.
- **AtkWindow** — The ATK Interface provided by UI components that represent a top-level window.

Basic accessible data types

- **AtkRange** - A given range or subrange, to be used with AtkValue
- **AtkRelation** — An object used to describe a relation between a object and one or more other objects.
- **AtkRelationSet** — A set of AtkRelations, normally the set of AtkRelations which an AtkObject has.
- **AtkState** — An AtkState describes a single state of an object.
- **AtkStateSet** — An AtkStateSet contains the states of an object.

Custom accessible objects

- **AtkGObjectAccessible** — This object class is derived from AtkObject and can be used as a basis implementing accessible objects.
- **AtkHyperlink** — An ATK object which encapsulates a link or set of links in a hypertext document.
- **AtkNoOpObject** — An AtkObject which purports to implement all ATK interfaces.
- **AtkPlug** — Toplevel for embedding into other processes
- **AtkSocket** — Container for AtkPlug objects from other processes

Utilities

- **AtkNoOpObjectFactory** — The AtkObjectFactory which creates an AtkNoOpObject.
- **AtkObjectFactory** — The base object class for a factory used to create accessible objects for objects of a specific GType.
- **AtkRegistry** — An object used to store the GType of the factories used to create an accessible object for an object of a particular GType.
- **Versioning macros** — Variables and functions to check the ATK version

Deprecated Interfaces

- **AtkMisc** — A set of ATK utility functions for thread locking
- **GDK**

I API Reference

- **General** — Library initialization and miscellaneous functions
- **GdkDisplayManager** — Maintains a list of all open GdkDisplays
- **GdkDisplay** — Controls a set of GdkScreens and their associated input devices
- **GdkScreen** — Object representing a physical screen
- **GdkDeviceManager** — Functions for handling input devices
- **GdkDevice** — Object representing an input device
- **Points and Rectangles** — Simple graphical data types
- **Pixbufs** — Functions for obtaining pixbufs
- **RGBA Colors** — RGBA colors
- **Visuals** — Low-level display hardware information
- **Cursors** — Standard and pixmap cursors
- **Windows** — Onscreen display areas in the target window system
- **Frame clock** — Frame clock syncs painting to a window or display

- **Frame timings** — Object holding timing information for a single frame
- **GdkGLContext** — OpenGL context
- **Events** — Functions for handling events from the window system
- **Event Structures** — Data structures specific to each type of event
- **Key Values** — Functions for manipulating keyboard codes
- **Selections** — Functions for transferring data via the X selection mechanism
- **Drag And Drop** — Functions for controlling drag and drop handling
- **Properties and Atoms** — Functions to manipulate properties on windows
- **Threads** — Functions for using GDK in multi-threaded programs
- **Pango Interaction** — Using Pango in GDK
- **Cairo Interaction** — Functions to support using cairo
- **X Window System Interaction** — X backend-specific functions
- **Wayland Interaction** — Wayland backend-specific functions
- **Application launching** — Startup notification for applications
- **Testing** — Test utilities

II Deprecated

- **Colors** — Manipulation of colors
- **GdkPixbuf**

I API Reference

- **Initialization and Versions** — Library version numbers.
- **The GdkPixbuf Structure** — Information that describes an image.
- **Reference Counting and Memory Management** — Functions for reference counting and memory management on pixbufs.
- **File Loading** — Loading a pixbuf from a file.
- **File saving** — Saving a pixbuf to a file.
- **Image Data in Memory** — Creating a pixbuf from image data that is already in memory.
- **Inline data** — Functions for inlined pixbuf handling.
- **Scaling** — Scaling pixbufs and scaling and compositing pixbufs
- **Rendering** — Rendering a pixbuf to a GDK drawable.
- **Drawables to Pixbufs** — Getting parts of a GDK drawable's image data into a pixbuf.
- **Utilities** — Utility and miscellaneous convenience functions.
- **Animations** — Animated images.
- **GdkPixbufLoader** — Application-driven progressive image loading.
- **Module Interface** — Extending GdkPixBuf
- **gdk-pixbuf Xlib initialization** — Initializing the gdk-pixbuf Xlib library.

- **Xlib Rendering** — Rendering a pixbuf to an X drawable.
- **X Drawables to Pixbufs** — Getting parts of an X drawable’s image data into a pixbuf.
- **XlibRGB** — Rendering RGB buffers to X drawables.

II Tools Reference

- **gdk-pixbuf-csource** — C code generation utility for GdkPixbuf images
- **gdk-pixbuf-query-loaders** — GdkPixbuf loader registration utility

• Cairo

A Vector Graphics Library.

Drawing

- **cairo_t** — The cairo drawing context
- **Paths** — Creating paths and manipulating path data
- **cairo_pattern_t** — Sources for drawing
- **Regions** — Representing a pixel-aligned area
- **Transformations** — Manipulating the current transformation matrix
- **text** — Rendering text and glyphs
- **Raster Sources** — Supplying arbitrary image data

Fonts

- **cairo_font_face_t** — Base class for font faces
- **cairo_scaled_font_t** — Font face at particular size and options
- **cairo_font_options_t** — How a font should be rendered
- **FreeType Fonts** — Font support for FreeType
- **Win32 Fonts** — Font support for Microsoft Windows
- **Quartz (CGFont) Fonts** — Font support via CGFont on OS X
- **User Fonts** — Font support with font data provided by the user

Surfaces

- **cairo_device_t** — interface to underlying rendering system
- **cairo_surface_t** — Base class for surfaces
- **Image Surfaces** — Rendering to memory buffers
- **PDF Surfaces** — Rendering PDF documents
- **PNG Support** — Reading and writing PNG images

- **PostScript Surfaces** — Rendering PostScript documents
- **Recording Surfaces** — Records all drawing operations
- **Win32 Surfaces** — Microsoft Windows surface support
- **SVG Surfaces** — Rendering SVG documents
- **Quartz Surfaces** — Rendering to Quartz surfaces
- **XCB Surfaces** — X Window System rendering using the XCB library
- **XLib Surfaces** — X Window System rendering using XLib
- **XLib-XRender Backend** — X Window System rendering using XLib and the X Render extension
- **Script Surfaces** — Rendering to replayable scripts

Utilities

- **cairo_matrix_t** — Generic matrix operations
- **Error handling** — Decoding cairo’s status
- **Version Information** — Compile-time and run-time version checks.
- **Types** — Generic data types

Appendix B: Sensoray 826i Manual

PCI Express Multifunction I/O Board Instruction Manual

Model 826 | Rev.3.0.5 | November 2013

SENSORAY | embedded electronics | 

Designed and manufactured in the U.S.A.

SENSORAY | p.503.684.8005 | email:info@SENSORAY.com | wwwSENSORAY.com

7313 SW Tech Center Drive | Portland, OR 97203

Table of Contents

Chapter 1: Preliminary.....	1	5.3.4 S826_AdcSlotlistRead	17
1.1 Limited Warranty	1	5.3.5 S826_AdcTrigModeWrite	18
1.2 Handling Instructions	1	5.3.6 S826_AdcTrigModeRead	19
Chapter 2: Introduction.....	2	5.3.7 S826_AdcEnableWrite	19
2.1 Overview	2	5.3.8 S826_AdcEnableRead	19
2.1.1 Timestamp Generator	3	5.3.9 S826_AdcStatusRead	20
2.1.2 Board Reset	3	5.3.10 S826_AdcRead	20
2.2 Hardware Configuration	3	5.3.11 S826_AdcWaitCancel	22
2.3 Board Layout	4	Chapter 6: Analog Outputs.....	23
2.4 Cable Installation	4	6.1 Introduction	23
Chapter 3: Programming.....	5	6.1.1 Safemode	23
3.1 Thread Safety	5	6.1.2 Reset State	24
3.1.1 Atomic Read-Modify-Write	5	6.2 Connector J1	24
3.2 Event-Driven Applications	5	6.3 Programming	24
3.3 Error Codes	6	6.3.1 S826_DacRangeWrite	24
3.4 Open/Close Functions	6	6.3.2 S826_DacDataWrite	25
3.4.1 S826_SystemOpen	6	6.3.3 S826_DacRead	25
3.4.2 S826_SystemClose	6	Chapter 7: Counters.....	27
3.5 Status Functions	7	7.1 Introduction	27
3.5.1 S826_VersionRead	7	7.1.1 ClkA, ClkB and IX Signals	27
3.5.2 S826_TimestampRead	7	7.1.2 Quadrature Decoder	28
Chapter 4: Virtual Outputs.....	9	7.1.3 ExtIn Signal	28
4.1 Introduction	9	7.1.4 ExtOut Signal	28
4.1.1 Safemode	9	7.1.5 Snapshots	29
4.2 Programming	9	7.1.6 Preloading	29
4.2.1 S826_VirtualWrite	9	7.1.7 Tick Generator	30
4.2.2 S826_VirtualRead	10	7.1.8 Cascading	30
4.2.3 S826_VirtualSafeWrite	10	7.1.9 Status LEDs	30
4.2.4 S826_VirtualSafeRead	11	7.1.10 Reset State	30
4.2.5 S826_VirtualSafeEnablesWrite	11	7.2 Connectors J4/J5	31
4.2.6 S826_VirtualSafeEnablesRead	12	7.2.1 Counter Signals	31
Chapter 5: Analog Inputs.....	13	7.2.2 Application Connections	32
5.1 Introduction	13	7.3 Programming	32
5.1.1 Triggering	14	7.3.1 S826_CounterSnapshotRead	32
5.1.2 Burst Counter	14	7.3.2 S826_CounterWaitCancel	34
5.1.3 Result Registers	14	7.3.3 S826_CounterCompareWrite	34
5.2 Connector J1	15	7.3.4 S826_CounterCompareRead	35
5.3 Programming	15	7.3.5 S826_CounterSnapshot	35
5.3.1 S826_AdcSlotConfigWrite	15	7.3.6 S826_CounterRead	36
5.3.2 S826_AdcSlotConfigRead	16	7.3.7 S826_CounterPreloadWrite	36
5.3.3 S826_AdcSlotlistWrite	17	7.3.8 S826_CounterPreloadRead	37

7.3.14	<code>S826_CounterSnapshotConfigWrite</code>	40	8.3.13	<code>S826_DioOutputSourceRead</code>	58
7.3.15	<code>S826_CounterSnapshotConfigRead</code>	41	8.3.14	<code>S826_DioFilterWrite</code>	58
7.3.16	<code>S826_CounterFilterWrite</code>	42	8.3.15	<code>S826_DioFilterRead</code>	59
7.3.17	<code>S826_CounterFilterRead</code>	42			
7.3.18	<code>S826_CounterModeWrite</code>	43			
7.3.19	<code>S826_CounterModeRead</code>	45			
7.3.20	Common Applications	45			
	Chapter 8: Digital I/O	48			
8.1	Introduction	48	9.1	Introduction	60
8.1.1	<code>Signal Routing Matrix</code>	48	9.1.1	Operation	60
8.1.2	<code>Safemode</code>	49	9.1.2	Reset Out Circuit	61
8.1.3	<code>Edge Capture</code>	49	9.1.3	Initialization	61
8.1.4	<code>Pin Timing</code>	49	9.2	<code>Connector P2</code>	61
8.1.4.1	<code>Noise Filter</code>	50	9.3	Programming	61
8.1.5	<code>Reset State</code>	50	9.3.1	<code>S826_WatchdogConfigWrite</code>	61
8.2	<code>Connectors J2/J3</code>	50	9.3.2	<code>S826_WatchdogConfigRead</code>	62
8.3	Programming	50	9.3.3	<code>S826_WatchdogEnableWrite</code>	62
8.3.1	<code>S826_DioOutputWrite</code>	51	9.3.4	<code>S826_WatchdogEnableRead</code>	63
8.3.2	<code>S826_DioOutputRead</code>	51	9.3.5	<code>S826_WatchdogStatusRead</code>	63
8.3.3	<code>S826_DioInputRead</code>	52	9.3.6	<code>S826_WatchdogKick</code>	64
8.3.4	<code>S826_DioSafeWrite</code>	52	9.3.7	<code>S826_WatchdogEventWait</code>	64
8.3.5	<code>S826_DioSafeRead</code>	53	9.3.8	<code>S826_WatchdogWaitCancel</code>	65
8.3.6	<code>S826_DioSafeEnablesWrite</code>	53			
8.3.7	<code>S826_DioSafeEnablesRead</code>	53			
8.3.8	<code>S826_DioCapEnablesWrite</code>	54			
8.3.9	<code>S826_DioCapEnablesRead</code>	55			
8.3.10	<code>S826_DioCapRead</code>	55			
8.3.11	<code>S826_DioWaitCancel</code>	56			
8.3.12	<code>S826_DioOutputSourceWrite</code>	57			
	Chapter 9: Watchdog Timer	60			
	10.1	Introduction	66		
	10.1.1	<code>Write Protection</code>	66		
	10.2	Programming	67		
	10.2.1	<code>S826_SafeControlWrite</code>	67		
	10.2.2	<code>S826_SafeControlRead</code>	67		
	10.2.3	<code>S826_SafeWrenWrite</code>	68		
	10.2.4	<code>S826_SafeWrenRead</code>	68		
	Chapter 10: Safemode Controller	66			
	Chapter 11: Specifications	70			

Chapter 1: Preliminary

1.1 Limited Warranty

Sensoray Company, Incorporated (Sensoray) warrants the Model 826 hardware to be free from defects in material and workmanship and perform to applicable published Sensoray specifications for two years from the date of shipment to purchaser. Sensoray will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The warranty provided herein does not cover equipment subjected to abuse, misuse, accident, alteration, neglect, or unauthorized repair or installation. Sensoray shall have the right of final determination as to the existence and cause of defect.

As for items repaired or replaced under warranty, the warranty shall continue in effect for the remainder of the original warranty period, or for ninety days following date of shipment by Sensoray of the repaired or replaced part, whichever period is longer.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. Sensoray will pay the shipping costs of returning to the owner parts that are covered by warranty. A restocking charge of 25% of the product purchase price will be charged for returning a product to stock.

Sensoray believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, Sensoray reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition.

The reader should consult Sensoray if errors are suspected. In no event shall Sensoray be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, SENSORAY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF SENSORAY SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. SENSORAY WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

Third party brands, names and trademarks are the property of their respective owners.

1.2 Handling Instructions

The Model 826 circuit board contains electronic circuitry that is sensitive to Electrostatic Discharge (ESD).

Special care should be taken in handling, transporting, and installing circuit board to prevent ESD damage to the board. In particular:

- Do not remove the circuit board from its protective packaging until you are ready to install it.
- Handle the circuit board only at grounded, ESD protected stations.
- Remove power from the equipment before installing or removing the circuit board. In particular, disconnect all field wiring from the board before installing or removing the board from the backplane.

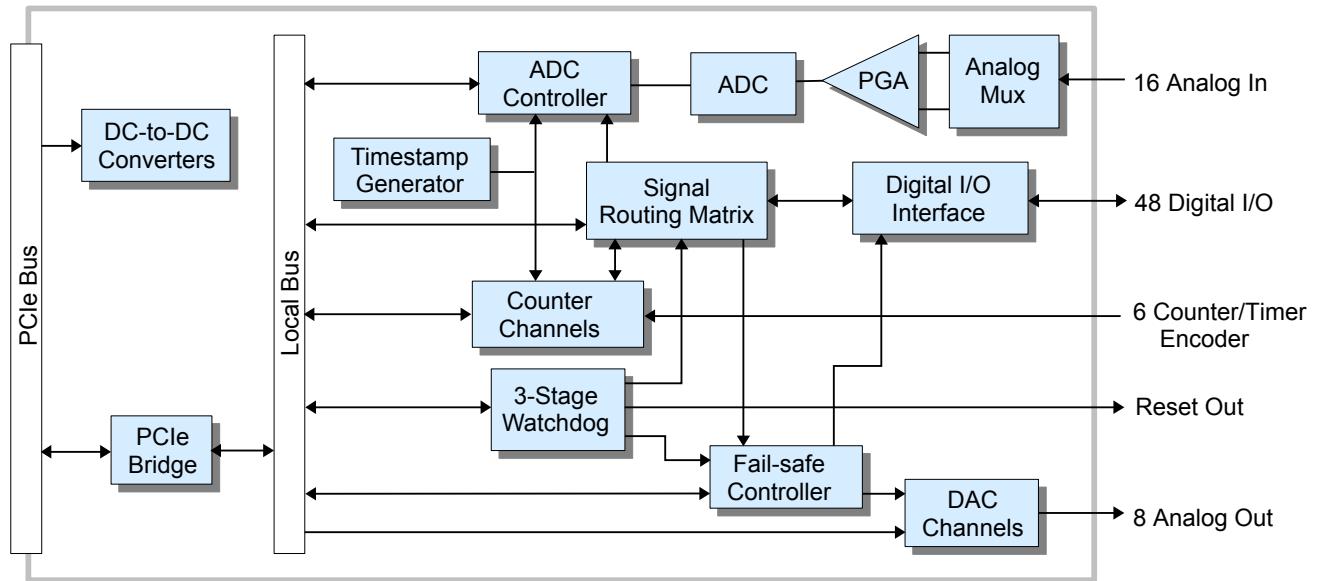
Chapter 2: Introduction

2.1 Overview

Model 826 is a PCI Express board that features an assortment of I/O interfaces commonly used in measurement and control applications:

- **48 bidirectional digital I/O channels** with edge detection and fail-safe output control.
- **Eight 16-bit analog outputs** ($\pm 10V$, $\pm 5V$, 0-10V, 0-5V) with fail-safe output control.
- **Sixteen 16-bit differential analog inputs** ($\pm 10V$, $\pm 5V$, $\pm 2V$, $\pm 1V$) with hardware and software triggering.
- **Six 32-bit counters**. Clock inputs may be driven from external quadrature-encoded (e.g., incremental encoder) or single-phase sources. Inputs accept differential RS-422 or standard TTL/CMOS single-ended signals. Auxiliary inputs and outputs can be internally routed to digital I/O channels.
- **Multistage watchdog timer** that can activate fail-safe outputs and generate service requests.
- **Fail-safe controller** switches outputs to safe states upon watchdog timeout or external trigger.
- **Signal routing matrix** allows software to interconnect interfaces without external wiring.

Figure 1: Model 826 block diagram



Sensoray provides a free, comprehensive API (application programming interface) to facilitate the rapid development of polled, event-driven, or mixed-mode programs for Model 826. The API works in concert with the board's advanced architecture to support complex, high performance applications.

Standard headers are provided for connecting on-board peripherals to external circuitry. The board's low-profile headers allow it to fit comfortably into high-density systems, and an integral cable clamp keeps cables secure and organized.

On-board LEDs provide visual indications of the board's condition. The PWR indicator is lit when the on-board power supplies are operating. Six LEDs are used to indicate counter clock activity as explained in Status LEDs.

2.1.1 Timestamp Generator

A timestamp generator is shared by the analog input system and counter channels. The timestamp generator is a free-running 32-bit counter that increments every microsecond. The generator starts at zero counts upon board reset and overflows every 2^{32} microseconds (approximately 71.6 minutes). The timestamp counter is not writable, but the current timestamp counts can be read by calling the S826_TimestampRead function.

2.1.2 Board Reset

Upon power-up, or in response to a hardware reset, all interfaces and outputs are forced to default initial states. The initial conditions of the interfaces are discussed in the interface chapters.

2.2 Hardware Configuration

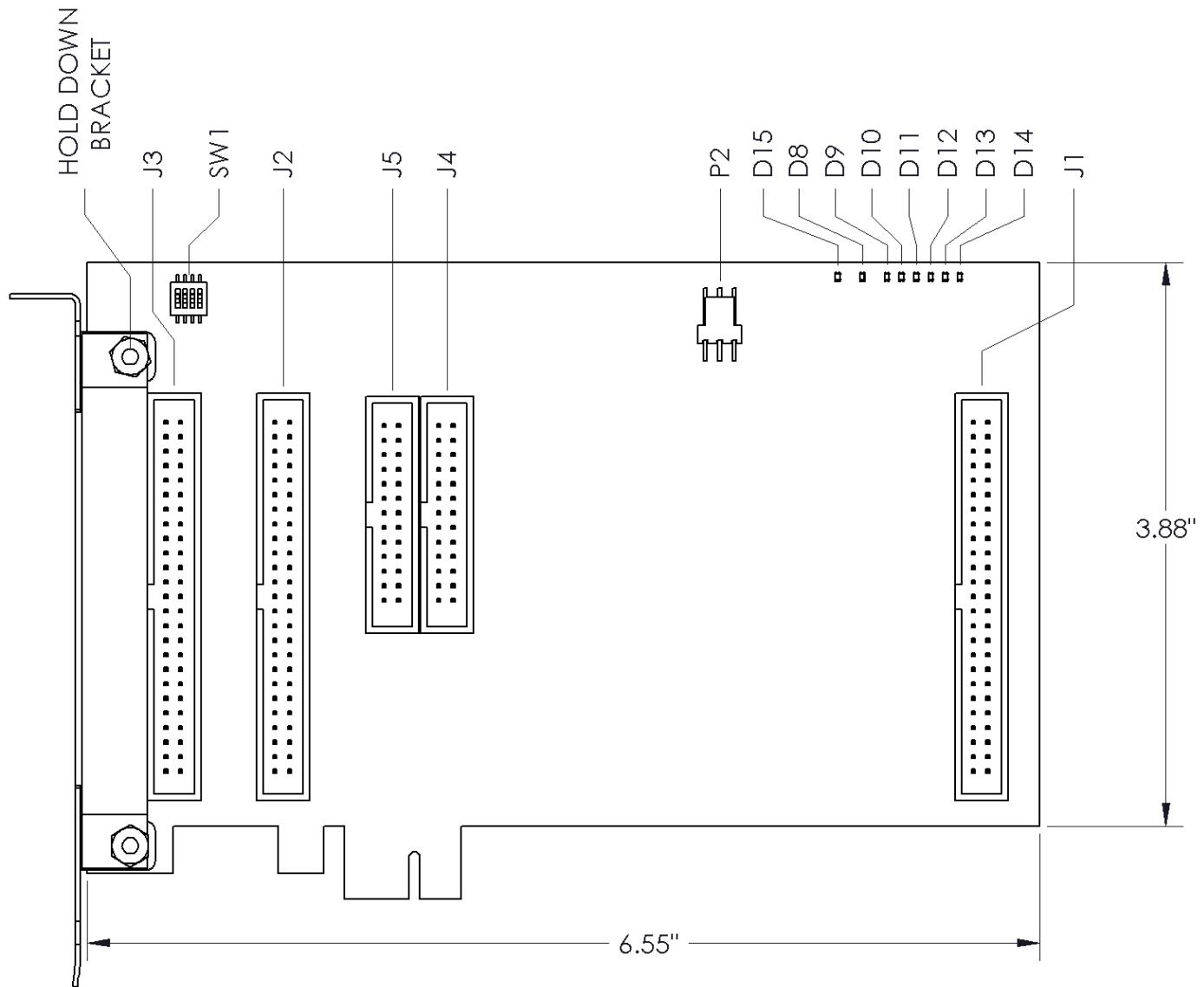
Up to sixteen model 826 boards can coexist in a single host computer. When more than one board is used in a system, each board must be configured before installation; this is done by setting quad dip switch SW1 to assign a unique identifier (board number) in the range 0 to 15 to the board.

By default, all model 826 boards are factory configured as board number 0. No reconfiguration is necessary if there is only one 826 board in the host computer; simply leave it configured at its default board number.

If more than one 826 board will be plugged into a common backplane, the boards must be configured so that each board has a unique board number. Board numbers are typically contiguous, though this is not required. For example, board numbers 0 and 3 could be assigned in a two-board system, though it would be more common to assign board numbers 0 and 1. This table shows the relationship between board number and switch settings:

Board Number	SW1-4	SW1-3	SW1-2	SW1-1	Notes
0	OFF	OFF	OFF	OFF	Factory default
1	OFF	OFF	OFF	ON	
2	OFF	OFF	ON	OFF	
3	OFF	OFF	ON	ON	
4	OFF	ON	OFF	OFF	
5	OFF	ON	OFF	ON	
6	OFF	ON	ON	OFF	
7	OFF	ON	ON	ON	
8	ON	OFF	OFF	OFF	
9	ON	OFF	OFF	ON	
10	ON	OFF	ON	OFF	
11	ON	OFF	ON	ON	
12	ON	ON	OFF	OFF	
13	ON	ON	OFF	ON	
14	ON	ON	ON	OFF	
15	ON	ON	ON	ON	

2.3 Board Layout



2.4 Cable Installation

The 826 board should be connected to external circuitry with Sensoray cables, model 826C1 (26 conductor, for counters) and 826C2 (50 conductor, for analog and digital I/O), which are specifically designed for this purpose. These cables feature thin, flat cable and low profile headers that allow the board to fit into high-density systems when loaded with a complete complement of five cables.

To install the cables (see above diagram):

- Loosen and remove the board's cable clamp (part of the hold-down bracket assembly).
- Pass each cable's low-profile end through the hold-down bracket (from left of bracket) and plug it into its connector.
- Install and tighten the cable clamp.

Chapter 3: Programming

A software developers kit (SDK) for Model 826 is available for download from Sensoray's web site. The SDK includes Linux and Windows device drivers, sample application programs, and an application programming interface (API) that is supplied as both a Windows dynamic link library (DLL) and a Linux static library. The API core is available in source code form to OEMs who need to port the API to other operating systems.

This chapter provides an overview of the API and discusses functions that are used in all applications. Later chapters discuss API functions that are specific to the board's I/O systems.

3.1 Thread Safety

API functions are thread and process safe when they are not used to access shared hardware resources. For example, consider the case of an API function that is used to control general-purpose digital I/O (DIO) outputs where two threads or processes can call the function simultaneously. The function is guaranteed to be thread and process safe if those threads or processes interact with mutually exclusive DIOs. However, if the threads or processes attempt to manipulate the same DIO, that DIO will be a shared resource and the function is not guaranteed to be thread/process safe.

3.1.1 Atomic Read-Modify-Write

To facilitate high-performance thread-safe behavior, many of the API functions include a “mode” argument that specifies how a register write operation is to be performed. The mode argument works in conjunction with a bit-set and bit-clear hardware function that is implemented on many of the board's interface control registers.

mode	Operation
0	Write all data bits unconditionally. All register bits are programmed to explicit values.
1	Clear (program to '0') all register bits that have '1' in the corresponding data bit. All other register bits are unaffected.
2	Set (program to '1') all register bits that have '1' in the corresponding data bit. All other register bits are unaffected.

For example, if mode=2 and the data value is 0x00000003, register bits 0 and 1 will be set to '1' and all other register bits will retain their previous values.

3.2 Event-Driven Applications

Event-driven applications can be implemented by calling the API's blocking functions: S826_CounterSnapshotRead, S826_AdcRead, S826_DioCapRead, and S826_WatchdogEventWait. These functions can be used to block the calling thread while it waits for hardware signal events. Blocking functions will return immediately (without blocking) if the event occurs before the function is called.

All of the blocking functions include a “tmax” argument that specifies how long, in microseconds, to wait for events:

tmax	Blocking function behavior
0	Return immediately even if event has not occurred (never blocks).
1 to 4294967294	Block until event occurs or tmax microseconds elapse, whichever comes first. This corresponds to times ranging from 1 microsecond to approximately 71.6 minutes.
S826_WAIT_INFINITE	Block until event occurs, with no time limit.

Note that non-realtime operating systems (such as Windows) may not respond to events with deterministic timing. The responsiveness of such systems can depend on a number of factors including CPU loading, thread and process priorities, memory capacity and architecture, core architecture and count, and clock frequency.

3.3 Error Codes

Most of the API functions return an error code. These functions return zero if no errors are detected, otherwise a negative value will be returned that indicates the type of error that occurred.

Error Code	Value	Description
S826_ERR_OK	0	No errors.
S826_ERR_BOARD	-1	Invalid board number.
S826_ERR_VALUE	-2	Illegal argument value.
S826_ERR_NOTREADY	-3	Device was busy or unavailable, or wait timed out.
S826_ERR_CANCELLED	-4	Blocking function was canceled.
S826_ERR_DRIVER	-5	Driver call failed.
S826_ERR_MISSEDTRIG	-6	ADC trigger occurred while previous conversion burst was in progress.
S826_ERR_DUPADDR	-9	Two 826 boards are set to the same board number. Change DIP switch settings.
S826_ERR_BOARDCLOSED	-10	Addressed board is not open.
S826_ERR_CREATEMUTEX	-11	Failed to create internal mutex.
S826_ERR_MEMORYMAP	-12	Failed to map board into memory space.
S826_ERR_MALLOC	-13	Failed to allocate memory.
S826_ERR_FIFO_OVERFLOW	-15	Counter channel's snapshot FIFO overflowed.
S826_ERR_OSSPECIFIC	-1xx	Error specific to the operating system. Contact Sensoray.

3.4 Open/Close Functions

3.4.1 S826_SystemOpen

The S826_SystemOpen function detects all Model 826 boards and enables communication with the boards.

```
int S826_SystemOpen(void);
```

Return Values

If the function succeeds, the return value is a set of bit flags indicating all detected 826 boards. Each bit position corresponds to the board number programmed onto a board's switches (see “Hardware Configuration“). For example, bit 0 will be set if an 826 with board number 0 is detected. The return value will be zero if no boards are detected, or positive if one or more boards are detected without error.

If the function fails, the return value is an error code (always a negative value). S826_ERR_DUPADDR will be returned if two boards are detected that have the same board number.

Remarks

This function must be called by a process before interacting with any 826 boards. The function allocates system resources that are used internally by other API functions. S826_SystemClose must be called once, for each call to S826_SystemOpen, to free the resources when they are no longer needed (e.g., when the 826 application program terminates). The board's circuitry is not reset by this function; all registers and I/O states are preserved.

S826_SystemOpen should be called once by every process that will use the 826 API. In a multi-threaded process, call the function once before any other API functions are called; all threads belonging to the process will then have access to all 826 boards in the system.

3.4.2 S826_SystemClose

The S826_SystemClose function frees all of the system resources that were allocated by S826_SystemOpen and disables communication with all 826 boards.

```
int S826_SystemClose();
```

Return Values

The return value is always zero.

Remarks

This function should be called when a process (e.g., an application program) has finished interacting with the board's I/O interfaces, to free system resources that are no longer needed. Any API functions that are blocking will return immediately, with return value S826_ERR_SYSCLOSED. The board's circuitry is not reset by this function; all registers and I/O states are preserved.

3.5 Status Functions

3.5.1 S826_VersionRead

The S826_VersionRead function returns API, driver, and board version information.

```
int S826_VersionRead(
    uint board,          // board identifier
    uint *api,           // API version
    uint *driver,         // driver version
    uint *bdrev,          // circuit board version
    uint *fpgarev        // FPGA version
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

api

Pointer to a buffer that will receive the API version info. The hexadecimal value is formatted as 0xXXYYZZZZ, where XX = major version, YY = minor version, and ZZZZ = build.

driver

Pointer to a buffer that will receive the driver version info. The hexadecimal info is formatted as 0xXXYYZZZZ, where XX = major version, YY = minor version, and ZZZZ = build.

boardrev

Pointer to a buffer that will receive the version number of the 826's circuit board.

fpgarev

Pointer to a buffer that will receive the board's FPGA version info. The hexadecimal info is formatted as 0xXXYYZZZZ, where XX = major version, YY = minor version, and ZZZZ = build.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

3.5.2 S826_TimestampRead

The S826_TimestampRead function returns the current value of the timestamp generator.

```
int S826_TimestampRead(
    uint board,          // board identifier
    uint *timestamp      // current timestamp
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

timestamp

Pointer to buffer that will receive the timestamp. The returned value is the contents of the timestamp generator at the moment the function executes.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function can be used to monitor elapsed times with microsecond level resolution, independent of the type of operating system being used.

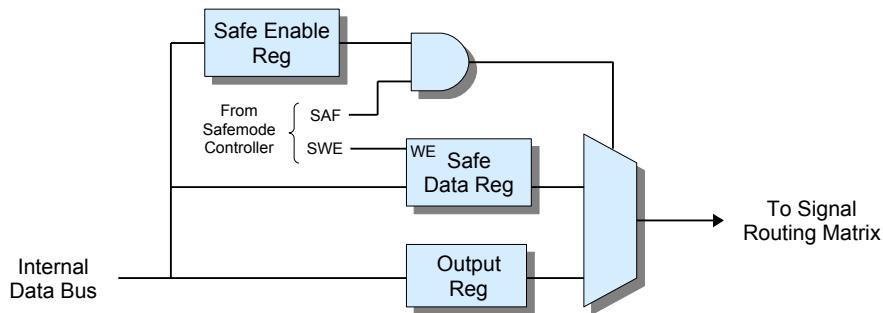
Chapter 4: Virtual Outputs

4.1 Introduction

The board has six virtual digital output channels, numbered 0 to 5, that can be used by software to signal various interfaces on the board. The virtual output channels are physical circuits that are architecturally similar to the board's general-purpose digital output channels, but they cannot be routed to the board's headers or connected to external circuitry.

Each virtual output channel is connected to the board's internal signal routing matrix, which in turn routes the channel's output signal to other on-board interfaces under program control. A virtual output channel can be routed to the ExtIn input of one or more counter channels, or to the ADC trigger input, or to combinations of these.

Figure 2: Virtual output channel (1 of 6)



4.1.1 Safemode

Safemode is activated when the SAF signal (see Figure 2) is asserted. When operating in safemode, the virtual output state is determined by the Safe Enable and Safe Data registers: when Safe Enable equals '1' the pin will be driven to the fail-safe value stored in Safe Data; when Safe Enable equals '0' the pin will output the normal runmode signal from its output register.

Upon board reset, the Safe Enable register is set to '1' so that the virtual output will exhibit fail-safe behavior by default (i.e., it will output the contents of the Safe Data register when SAF is asserted). Fail-safe operation can be disabled for a virtual output channel by programming its Safe Enable register to '0'.

The Safe Data register is typically programmed once (or left at its default value) by the application when it starts, though it may be changed at any time if required by the application. It can be written to only when the SWE bit is set to '1'. See "Safemode Controller" for more information about safemode.

4.2 Programming

The virtual output API functions employ individual bits to convey information about the virtual output channels, wherein each bit represents the information for one channel. The information is organized into a single quadlet as follows (e.g., "v5" = virtual output channel 5):

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	v5	v4	v3	v2	v1	v0	

4.2.1 S826_VirtualWrite

The S826_VirtualWrite function programs the virtual output registers.

```

int S826_VirtualWrite(
    uint board,      // board identifier
    uint data,       // data to write
    uint mode        // 0=write, 1=clear bits, 2=set bits
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Normal (runmode) output data for the six virtual channels (see Section 4.2).

mode

Write mode for data: 0 = write, 1 = clear bits, 2 = set bits (see “Atomic Read-Modify-Write”).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

In mode zero, this function will unconditionally write new values to all virtual output registers. In modes one and two, the function will selectively clear or set any arbitrary combination of output registers; data bits that contain logic '1' indicate virtual output registers that are to be modified, while all other output registers will remain unchanged. Modes one and two can be used to ensure thread-safe operation as they atomically set or clear the specified bits.

4.2.2 S826_VirtualRead

The S826_VirtualRead function reads the programmed states of all virtual output registers.

```

int S826_VirtualRead(
    uint board,      // board identifier
    uint *data       // pointer to data buffer
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to a buffer (see Section 4.2) that will receive the output register states.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function returns the output register states.

4.2.3 S826_VirtualSafeWrite

The S826_VirtualSafeWrite function programs the virtual output channel Safe registers.

```

int S826_VirtualSafeWrite(
    uint board,      // board identifier
    uint data,       // safemode data
    uint mode        // 0=write, 1=clear bits, 2=set bits
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to data array (see Section 4.2) to be programmed into the Safe registers.

mode

Write mode for data: 0 = write, 1 = clear bits, 2 = set bits (see “Atomic Read-Modify-Write”).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function should only be called when the SWE bit is set (see S826_SafeWrenWrite). The function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1).

4.2.4 S826_VirtualSafeRead

The S826_VirtualSafeRead function returns the contents of the virtual output channel Safe registers.

```

int S826_VirtualSafeRead(
    uint board,      // board identifier
    uint *data       // pointer to data buffer
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to a buffer (see Section 4.2) that will receive the Safe register states.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

4.2.5 S826_VirtualSafeEnablesWrite

The S826_VirtualSafeEnablesWrite function programs the virtual output channel Safe Enable registers.

```

int S826_VirtualSafeEnablesWrite(
    uint board,      // board identifier
    uint enables     // safemode enables
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

enables

Pointer to array of values (see Section 4.2) to be programmed into the Safe Enable registers.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function should only be called when the SWE bit is set (see S826_SafeWrenWrite). The function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1).

4.2.6 S826_VirtualSafeEnablesRead

The S826_VirtualSafeEnablesRead function returns the contents of the virtual output channel Safe Enable registers.

```
int S826_VirtualSafeEnablesRead(
    uint board,          // board identifier
    uint *enables        // pointer to data buffer
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

enables

Pointer to a buffer (see Section 4.2) that will receive the Safe Enable register contents.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

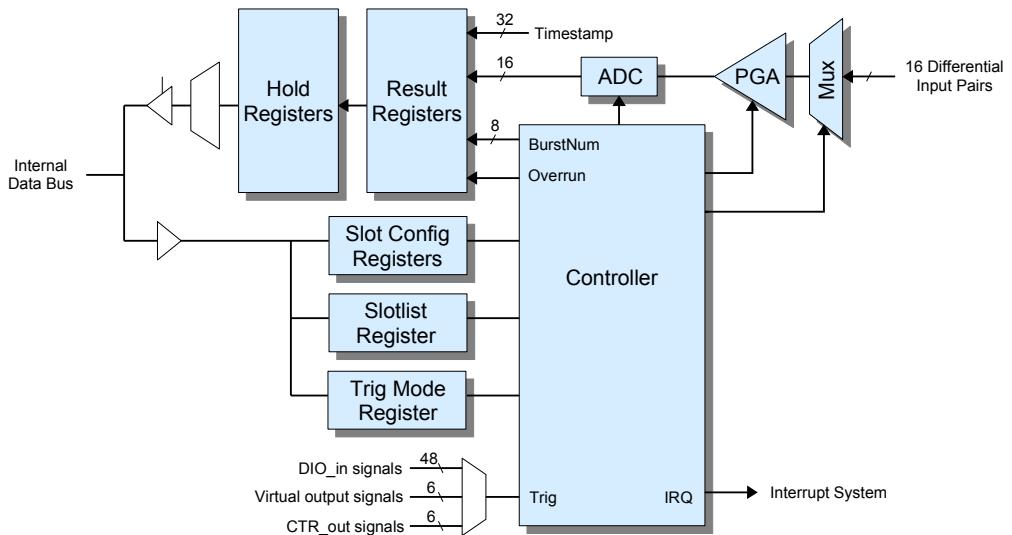
Chapter 5: Analog Inputs

5.1 Introduction

The board's analog input system consists of the following major elements:

- **Analog multiplexer (Mux)** - selects one of 16 differential input pairs (channels) for conversion.
- **Programmable gain amplifier (PGA)** - applies voltage gain of 1, 2, 5, or 10 to the selected input.
- **Analog-to-digital converter (ADC)** - performs an analog-to-digital conversion in three microseconds or less.
- **Controller** - manages operation of the analog input system.

Figure 3: Analog input system



Analog-to-digital (A/D) conversions are performed in bursts of up to 16 conversions. Each conversion takes place in a timeslot (called a *slot*), which is a reserved time interval within a burst. During a burst, slots are processed in numerical order beginning with slot 0 and ending with slot 15.

Every slot has three attributes that are programmed into its slot configuration register via the S826_AdcSlotConfigWrite function:

- **Analog channel number:** designates the analog input channel that will be digitized when the slot is processed. Any of the 16 channels may be assigned to any slot. A channel may be assigned to two or more slots if desired.
- **PGA gain:** specifies the analog gain to apply to the analog input signal.
- **Settling time:** specifies the delay from analog multiplexer switching to start of digitization.

The slotlist register designates each slot as either active or inactive; this is programmed via the S826_AdcSlotlistWrite function. When a burst occurs, each slot is processed according to the slotlist. An active slot is processed by performing one A/D conversion; its total time is the sum of its settling time plus a fixed conversion time. Inactive slots are skipped and consume no time during a burst (the slot configuration register is ignored for an inactive slot). The total processing time of each slot is configurable, from zero (inactive slot) to approximately 335 milliseconds, in one-microsecond increments.

ADC conversions are disabled upon board reset. Typically, an application program will configure the ADC system (by programming slotlist and slot configuration registers) before enabling conversions, though this is not required. The slotlist and slot configuration registers may be reprogrammed at any time, even when conversions are enabled.

5.1.1 Triggering

The trigger mode register determines which of two triggering modes, *triggered* or *continuous*, will be used to initiate conversion bursts. In triggered mode each burst must be initiated by a trigger signal, whereas in continuous mode the trigger signal is ignored and conversion bursts will automatically execute one after another with no idle time between bursts. The trigger mode and trigger signal source are programmed by calling S826_AdcTrigModeWrite.

When operating in triggered mode, S826_AdcTrigModeWrite selects one of the following signals to serve as the trigger:

- Any of the 48 general purpose digital I/O (DIO) channels. This enables an external digital signal to trigger bursts. When a DIO channel is used to trigger A/D conversions, the timing of the trigger signal is constrained by the DIO subsystem. See DIO “Pin Timing” for more information.
- Any of the six counter ExtOut signals. This enables a counter to periodically trigger ADC bursts. The selected counter output is internally routed to the ADC trigger input; no external wiring is required.
- Any of the six virtual digital outputs. This enables software to trigger ADC bursts by writing to a virtual output. See Virtual Outputs for more information. The selected virtual output is internally routed to the ADC trigger input; no external wiring is required.

5.1.2 Burst Counter

The controller maintains an 8-bit burst counter that indicates the number of conversion bursts since the ADC was enabled. The burst counter is zeroed upon board reset and whenever the ADC is disabled. The counter increments at the end of each burst and overflows to zero when incrementing from 255. The burst count is passed to the host along with every ADC sample so that the program can determine which burst a sample belongs to.

In triggered mode, the burst count can be used to keep track of the number of received triggers. If a trigger occurs while a burst is in progress, a MissedTrigger flag is set and the trigger will be ignored. After this happens, the burst count will no longer accurately indicate the number of received triggers (accuracy can be restored by disabling and then re-enabling ADC conversions).

5.1.3 Result Registers

Each slot has a result register that stores the slot's most recently acquired result, which is a set of four values: ADC output data, timestamp (which indicates the time the result was acquired), burst count, and overrun flag. Each slot also has a hold register that caches a result while it is being read by the host computer. The hold register ensures that the result's four component values will remain correlated if a new result is captured while the previous result is being read.

During a burst, the controller processes each slot by performing a sequence of operations. First it switches the analog multiplexer to the desired differential input pair and programs the gain and then, if the slot has a non-zero settling time, it waits for the settling time to elapse. When the settling time has elapsed, the controller starts an analog-to-digital conversion. At the moment the conversion completes, the four component values that comprise the result are simultaneously sampled and copied to the result register.

A new result will always overwrite the previous result, even if the previous result has not been read. If the previous result has not yet been read when a new result is written, the overrun flag will be set to '1'; otherwise it will be set to '0'.

Sixteen hardware status flags (one per slot) indicate unread results. A status flag is set when the controller writes a new result to the associated result register, and reset when the program reads the result. Results are read by calling the S826_AdcRead function. The S826_AdcStatusRead function can be called to check the status flags without returning results or altering status flags.

5.2 Connector J1

All analog input and output signals are available at connector J1.

J1 Pinout

Pin	Name	Function
1	GND	Power supply common
2	GND	Power supply common
3	-AIN0	Analog input (-) channel 0
4	+AIN0	Analog input (+) channel 0
5	-AIN1	Analog input (-) channel 1
6	+AIN1	Analog input (+) channel 1
7	-AIN2	Analog input (-) channel 2
8	+AIN2	Analog input (+) channel 2
9	-AIN3	Analog input (-) channel 3
10	+AIN3	Analog input (+) channel 3
11	-AIN4	Analog input (-) channel 4
12	+AIN4	Analog input (+) channel 4
13	-AIN5	Analog input (-) channel 5
14	+AIN5	Analog input (+) channel 5
15	-AIN6	Analog input (-) channel 6
16	+AIN6	Analog input (+) channel 6
17	-AIN7	Analog input (-) channel 7
18	+AIN7	Analog input (+) channel 7
19	GND	Power supply common
20	GND	Power supply common
21	-AIN8	Analog input (-) channel 8
22	+AIN8	Analog input (+) channel 8
23	-AIN9	Analog input (-) channel 9
24	+AIN9	Analog input (+) channel 9
25	-AIN10	Analog input (-) channel 10
Pin	Name	Function
26	+AIN10	Analog input (+) channel 10
27	-AIN11	Analog input (-) channel 11
28	+AIN11	Analog input (+) channel 11
29	-AIN12	Analog input (-) channel 12
30	+AIN12	Analog input (+) channel 12
31	-AIN13	Analog input (-) channel 13
32	+AIN13	Analog input (+) channel 13
33	-AIN14	Analog input (-) channel 14
34	+AIN14	Analog input (+) channel 14
35	-AIN15	Analog input (-) channel 15
36	SH15	Analog input (+) channel 15
37	GND	Power supply common
38	GND	Power supply common
39	GND	Power supply common
40	GND	Power supply common
41	AOUT4	Analog output channel 4
42	AOUT0	Analog output channel 0
43	AOUT5	Analog output channel 5
44	AOUT1	Analog output channel 1
45	AOUT6	Analog output channel 6
46	AOUT2	Analog output channel 2
47	AOUT7	Analog output channel 7
48	AOUT3	Analog output channel 3
49	GND	Power supply common
50	GND	Power supply common

5.3 Programming

5.3.1 S826_AdcSlotConfigWrite

The S826_AdcSlotConfigWrite function configures a timeslot.

```
int S826_AdcSlotConfigWrite(
    uint board,      // board identifier
    uint slot,       // timeslot number
    uint chan,       // analog input channel number
    uint tsettle,    // settling time in microseconds
    uint range       // input range code
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

slot

Timeslot number in the range 0 to 15.

chan

Analog input channel number in the range 0 to 15.

tsettle

Microseconds to allow the analog input to settle before conversion, in the range 0 to 335,544.

range

Enumerated value that specifies the analog input voltage range. This determines the analog gain that will be applied to the input signal before digitization.

range	PGA Gain	Analog Input Range	Notes
0	x1	-10V to +10V	Default upon reset
1	x2	-5V to +5V	
2	x5	-2V to +2V	
3	x10	-1V to +1V	

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

The settings established by this function will be used the next time the slot is converted and remain in effect until changed by another call to this function or a board reset.

The maximum total time for each slot is $t_{settle}+3\ \mu s$. Sufficient settling time must be allowed when different channels are being digitized so that each input signal has time to stabilize before digitization. If a single channel is being digitized in multiple, consecutive slots, only the first of its slots must allow for settling time; subsequent slots may have zero settling time because the channel will already be settled. Similarly, if only one channel is being digitized (even if it is being digitized multiple times in different slots), all settling times may be set to zero because no channel switching will occur.

The ADC data latency is greater than the slot time because an additional $1\ \mu s$ is needed to fetch the digitized data after each conversion. The data latency is only incurred once per burst, because the ADC controller always fetches data from the previous conversion while the next conversion is in progress. Consequently, the elapsed time from hardware trigger to burst completion (in microseconds) is

$$t_{burst} \leq 1 + 3 \cdot \text{NumTimeslots} + \sum t_{settle}$$

5.3.2 S826_AdcSlotConfigRead

The S826_AdcSlotConfigRead function returns the configuration of a timeslot.

```
int S826_AdcSlotConfigRead(
    uint board,          // board identifier
    uint slot,           // timeslot number
    uint *chan,           // analog input channel number
    uint *tsettle,        // settling time in microseconds
    uint *range           // input range code
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

slot

Timeslot number in the range 0 to 15.

chan

Buffer that will receive the analog input channel number.

tsettle

Buffer that will receive the analog settling time in microseconds.

range

Buffer that will receive the analog input voltage range code, as specified in S826_AdcSlotConfigWrite.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function returns the settings established by a board reset or previous call to S826_AdcSlotConfigWrite.

5.3.3 S826_AdcSlotlistWrite

The S826_AdcSlotlistWrite function programs the conversion slot list.

```
int S826_AdcSlotlistWrite(
    uint board,          // board identifier
    uint slotlist,       // conversion slot list
    uint mode            // write mode
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

slotlist

List of slots to be digitized during subsequent conversion bursts, one bit per slot. Bits 0 to 15 correspond to slots 0 to 15, respectively. Each bit that is set to logic '1' will cause the corresponding slot to be digitized, while '0' will cause the slot to be skipped.

mode

Write mode for slotlist: 0 = write, 1 = clear bits, 2 = set bits (see "Atomic Read-Modify-Write").

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

The settings established by this function will be used the next time a slot is digitized and remain in effect until changed by another call to this function or a board reset.

In mode zero, this function will unconditionally write a new slotlist. In modes one and two, the function will selectively set or clear any arbitrary combination of slots; slotlist bits that contain logic '1' indicate slots that are to be modified, while all other slots will remain unchanged. Modes one and two can be used to guarantee thread-safe operation as they atomically enable or disable the specified slots without affecting any other slots.

5.3.4 S826_AdcSlotlistRead

The S826_AdcSlotlistRead function returns the conversion slot list.

```

int S826_AdcSlotlistRead(
    uint board,          // board identifier
    uint *slotlist      // conversion slot list
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

slotlist

Pointer to a buffer that will receive the slotlist. In the received value, bits 0 to 15 correspond to slots 0 to 15. For each bit: '1' = active (slot will be digitized during bursts), '0' = inactive (slot will be skipped during bursts).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function returns the current slotlist, which was previously programmed by S826_AdcSlotlistWrite or cleared by a board reset.

5.3.5 S826_AdcTrigModeWrite

The S826_AdcTrigModeWrite function programs the ADC triggering mode.

```

int S826_AdcTrigModeWrite(
    uint board,          // board identifier
    uint trigmode       // hardware trigger mode
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

trigmode

Hardware trigger configuration:

Bit	Function	Description
31-8	Reserved	Set all bits to 0.
7	Trigger enable	Set to 1 to enable hardware triggering (triggered mode), or 0 to disable hardware triggering (continuous mode).
6	Trigger polarity	1 selects rising edge; 0 selects falling edge. This is ignored if hardware triggering is disabled.
5-0	Trigger source	Hardware trigger signal source (ignored if hardware triggering is disabled): 0-47 = DIO channel 0-47. 48-53 = ExtOut signal from counter channel 0-5. 54-59 = Virtual digital output channel 0-5.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

The settings established by this function take effect upon the next conversion burst and remain in effect until changed by another call to this function or a board reset. Hardware triggering is disabled upon board reset.

5.3.6 S826_AdcTrigModeRead

The S826_AdcTrigModeRead function reads the ADC triggering mode.

```
int S826_AdcTrigModeRead(  
    uint board,          // board identifier  
    uint *trigmode     // hardware trigger mode  
) ;
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

trigmode

Pointer to a buffer that will receive the hardware trigger configuration. See S826_AdcTrigModeWrite for the format of this value.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

5.3.7 S826_AdcEnableWrite

The S826_AdcEnableWrite function enables or disables ADC conversions.

```
int S826_AdcEnableWrite(  
    uint board,          // board identifier  
    uint enable         // enable conversions when true  
) ;
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

enable

Set to 1 to enable, or 0 to disable ADC conversion bursts. Conversion bursts are disabled by default upon board reset.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

When enable is false, any conversion burst that is currently in progress will be terminated and all results and overrun status flags are reset. When enable is true, the resulting behavior depends on the trigger mode:

Continuous mode

The first conversion burst will begin immediately, followed by additional bursts. Each subsequent burst begins immediately after the preceding burst ends. Back-to-back bursts will continue until ADC conversions are disabled.

Triggered mode

A single conversion burst will begin in response to the next trigger and conversions will cease upon completion of that burst. Each subsequent trigger will cause another single burst. Triggers have no effect when ADC conversions are disabled.

5.3.8 S826_AdcEnableRead

The S826_AdcEnableRead function returns the enable status of the ADC system.

```
int S826_AdcEnableRead(
    uint board,      // board identifier
    uint *enable    // enable status
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

enable

Buffer that will receive the ADC system enable status: 1 = enabled, 0 = disabled.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function returns the ADC system's enable status that was previously configured by a board reset or a call to S826_AdcEnableWrite.

5.3.9 S826_AdcStatusRead

The S826_AdcStatusRead function reads the ADC conversion status.

```
int S826_AdcStatusRead(
    uint board,      // board identifier
    uint *status     // conversion status
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

status

Pointer to a buffer that will receive the conversion status for all slots. Each bit corresponds to one slot; bits 0 to 15 correspond to slots 0 to 15. A bit will be set to '1' if new (unread) data is available in the slot's result register, or '0' when the result register is empty (or has been read).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function returns status information without altering the state of the ADC system.

5.3.10 S826_AdcRead

The S826_AdcRead function fetches ADC data from one or more timeslots.

```
int S826_AdcRead(
    uint board,      // board identifier
    int buf[16],     // pointer to adc result buffer
    uint tstamp[16], // pointer to timestamps buffer
    uint *slotlist,  // pointer to slotlist
    uint tmax        // maximum time to wait
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

buf

Pointer to a buffer that will receive ADC results for the sixteen possible slots. The buffer must be large enough to accommodate sixteen values regardless of the number of active slots. Each slot is represented by a 32-bit value, which is stored at buf[SlotNumber]:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSTNUM				V	000000								ADCVAL																		

Field	Description
BSTNUM	Burst number. This indicates the ADC burst number corresponding to the ADC data. It can be used to time-correlate the ADC data if V = '1'. The burst number is incremented at the end of each burst; it restarts at zero at the end of burst number 255.
V	Data Overwritten flag. When set to '1' this indicates the previous ADC result was overwritten by a new result before it was read.
ADCVAL	ADC data, expressed as 16-bit signed integer:
Analog Voltage	ADCVAL
-10V to +10V	0x8000 to 0xFFFF
-5V to +5V	0x8000 to 0x7FFF
-2V to +2V	0x8000 to 0x7FFF
-1V to +1V	0x8000 to 0x7FFF

tstamp

Pointer to a buffer that will receive the timestamps corresponding to ADC results. The buffer must be large enough to accommodate sixteen values regardless of the number of active slots. Each timestamp indicates the moment in time that its corresponding ADCVAL was acquired. For any given slot, the slot's timestamp will be stored in tstamp[SlotNumber]. The application may set this to NULL if timestamps are not needed.

slotlist

Pointer to a buffer containing bit flags, one bit per slot, that indicate slots of interest. Bits 0-15 correspond to slots 0-15. Before calling the function, set to '1' all bits that correspond to slots of interest. When the function returns, the buffer contents will have been changed so that '1' indicates a slot has new data in buf and '0' indicates no new data.

tmax

Maximum time, in microseconds, to wait for ADC data. See “Event-Driven Applications” for details.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function reads the result registers of an arbitrary set of slots and copies the results to buf. As many as sixteen results (one per slot) may be copied to buf each time the function is called. Over-range and under-range conditions are indicated by maximum or minimum data values for the selected input range, respectively; there are no special status flags that indicate these conditions.

Before calling the function, one or more slotlist bits must be set to indicate the slots of interest. Only new, unread results are copied to buf. If a slot is not of interest or has not been converted since it was last read, its buf value will not change. In all cases (even when the function returns an error), when the function returns, slotlist will indicate the slots of interest that have new buf values.

The function will operate in either blocking or non-blocking mode depending on the value of tmax. If tmax is zero, the function will return immediately. If tmax is greater than zero, the calling thread will block until all requested data is available or tmax elapses. In either case, the function will copy to buf all data from slots of interest that have a new result. The function will return S826_ERR_NOTREADY if any of the requested slot data is unavailable.

This function will return S826_ERR_CANCELLED if, while it is blocking, another thread calls S826_AdcWaitCancel to cancel waits on any of the slots of interest. It will return immediately if the wait criteria is completely satisfied due to the wait cancellations, otherwise it will continue to block and return when all remaining wait criteria is satisfied.

S826_ERR_BOARDCLOSED will be returned immediately if S826_SystemClose executes while this function is blocking. In either case, no result data will be copied to buf.

In triggered mode, the board's internal MissedTrigger error flag will be set if a trigger occurs while an ADC burst is in progress. When the MissedTrigger flag is active, S826_AdcRead will wait for the requested slot data to arrive and then it will clear the MissedTrigger flag and return S826_ERR_MISSEDTRIG. If multiple threads are waiting in S826_AdcRead, only the first thread to return will receive S826_ERR_MISSEDTRIG; the other waiting threads will not receive this error.

Thread-safe operation is guaranteed only if the slots of interest for any given thread do not coincide with those of another thread. For example, thread safety would be assured if one thread designates slots 1 and 3-5 as slots of interest while another thread designates slots 2 and 9. Thread safety would be compromised, though, if both threads shared a common slot of interest.

5.3.11 S826_AdcWaitCancel

The S826_AdcWaitCancel function cancels a blocking wait on one or more slots.

```
int S826_AdcWaitCancel(
    uint board,          // board identifier
    uint slotlist        // slots to cancel
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

slotlist

Set of bit flags, one bit per slot, that indicate slots for which waiting is to be canceled. Bits 0-15 correspond to slots 0-15.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function cancels blocking for an arbitrary set of slots so that another thread, which is blocked by S826_AdcRead while waiting for adc data to arrive, will return immediately with S826_ERR_CANCELLED.

Chapter 6: Analog Outputs

6.1 Introduction

The 826 board has eight 16-bit, single-ended digital-to-analog converter (DAC) channels. Each channel can be independently configured to generate output voltages across one of four output voltage ranges: 0 to +5V (default upon reset), 0 to +10V, -5 to +5V, and -10 to +10V.

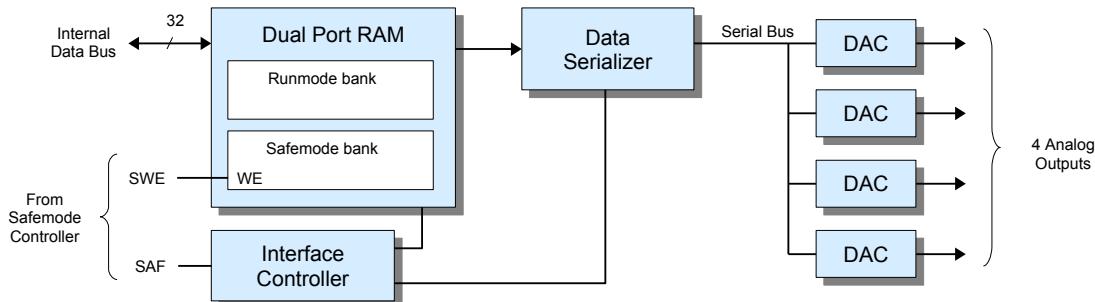
Each DAC channel has a setpoint and configuration register. A channel's output voltage is programmed by writing to its setpoint register. Before programming the setpoint, the DAC must be configured for the desired output range by writing to its configuration register.

Setpoint and configuration values are serially transmitted to the DAC devices over a high speed serial bus (Figure 4). The interface includes a dual-port RAM that allows the host to write setpoint and configuration values to the board while the serial bus is busy. If multiple untransmitted values are pending in the RAM, the controller will employ round-robin arbitration to ensure all pending values are transmitted in a fair and timely fashion. The data serializer requires 1.04 μ s to transmit each setpoint or configuration value.

The board has two identical DAC interfaces as shown in Figure 4. Each interface manages a group of four DAC channels. One interface manages channels 0 to 3 and the other interface manages channels 4 to 7. The two interfaces operate concurrently, thus making it possible to simultaneously write to two DACs that reside in different four-channel groups. For example, DAC channels 0 and 4 can be written to simultaneously.

The host may write setpoint and configuration values to the board at any time. If a new value is written to the RAM for a particular channel before that channel's previously written value has been transmitted, the previous RAM value will be overwritten and only the new value will be transmitted. Consequently, the host is allowed to write setpoint data at a rate that exceeds the serial bus bandwidth, though doing so will result in dropped samples.

Figure 4: Analog output interface (1 of 2)



6.1.1 Safemode

The dual-port RAM has two memory banks, one for normal operation (“runmode”) and another for “safemode” operation. The runmode bank stores the setpoint and configuration values that are used during normal operation; these values may be changed at any time as required by the application. The safemode bank contains alternate fail-safe settings that are typically programmed once during program initialization (or left at their default settings). Setpoint and configuration values can be written to the runmode bank at any time, but the safemode bank can only be written when SWE = '1'

The safemode signal (SAF) determines which RAM bank is being used by the interface controller ('1' = safemode, '0' = runmode). When SAF changes state, the appropriate RAM bank is selected and all of the DAC channels are reprogrammed to the settings stored in that bank. See “Safemode Controller” for more information about the fail-safe system.

6.1.2 Reset State

Upon reset, all DAC outputs and all channels in both RAM banks are programmed to 0V with the output range set to 0 to +5V.

6.2 Connector J1

DAC output signals are available on the analog I/O connector J1, which is shared with the ADC system. See Section 5.2 for the connector pinout.

Each DAC channel has a single-ended output signal that is referenced to the board's power supply common. The output and common signals are available on the analog I/O connector. DAC output signals are sensed at the connector; no external sense inputs are available on the connector.

6.3 Programming

6.3.1 S826_DacRangeWrite

The S826_DacRangeWrite function programs the voltage range of an analog output channel.

```
int S826_DacRangeWrite(
    uint board,      // board identifier
    uint chan,       // channel number
    uint range,      // output range
    uint safemode   // RAM bank select
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

DAC channel number in the range 0 to 7.

range

Enumerated value that specifies the output voltage range:

range	Analog Output Range	Notes
0	0 to +5V	Default upon reset
1	0 to +10V	
2	-5 to +5V	
3	-10 to +10V	

safemode

Specifies the RAM bank to be written: '0' = runmode, '1' = safemode.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function configures a channel's output voltage range and programs the setpoint to zero volts. It can be called at any time, though it is typically called once per channel during program initialization for the runmode bank and, if necessary, once per channel for the safemode bank as well (if default safemode values are not suitable).

The SWE bit must be set (see S826_SafeWrenWrite) to allow writing to the safemode bank. This function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1) when writing to the safemode bank.

6.3.2 S826_DacDataWrite

The S826_DacDataWrite function programs the output voltage of a DAC channel.

```
int S826_DacDataWrite(
    uint board,      // board identifier
    uint chan,       // channel number
    uint setpoint,   // output level
    uint safemode   // RAM bank select
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

DAC channel number in the range 0 to 7.

setpoint

Analog output level. This is a value ranging from 0x0000 to 0xFFFF. The resulting output voltage depends on the channel's previously programmed output range.

Analog output range	setpoint range
0 to +5V	0x0000 to 0xFFFF
0 to +10V	0x0000 to 0xFFFF
-5V to +5V	0x0000 to 0xFFFF
-10V to +10V	0x0000 to 0xFFFF

safemode

Specifies the RAM bank to be written: '0' = runmode, '1' = safemode.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function programs a channel's runmode or safemode output voltage level. If the output range will also be changed, the range should be changed first, before calling this function. This function is frequently used to change a runmode setpoint whenever an output level change is required. It can also be used to change a safemode setpoint; this is typically done once per channel when the application starts, after programming the channel's safemode output range.

The SWE bit must be set (see S826_SafeWrenWrite) to allow writing to the safemode bank. This function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1) when writing to the safemode bank.

6.3.3 S826_DacRead

The S826_DacRead function returns the output range and setpoint of an analog output channel.

```
int S826_DacRead(
    uint board,      // board identifier
    uint chan,       // channel number
    uint *range,     // output range
    uint *setpoint,  // output level
    uint safemode   // RAM bank select
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

DAC channel number in the range 0 to 7.

range

Pointer to a buffer that will receive the output range code as described in Section 5.3.1.

setpoint

Pointer to a buffer that will receive the output level as described in Section 5.3.2.

safemode

Specifies the RAM bank to be written: '0' = runmode, '1' = safemode.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function can be used to determine whether previously written configuration data has been sent to the DAC device. The value S826_ERR_NOTREADY will be returned if the range or setpoint is not yet active (i.e., one or both values have not yet been transmitted to the DAC device).

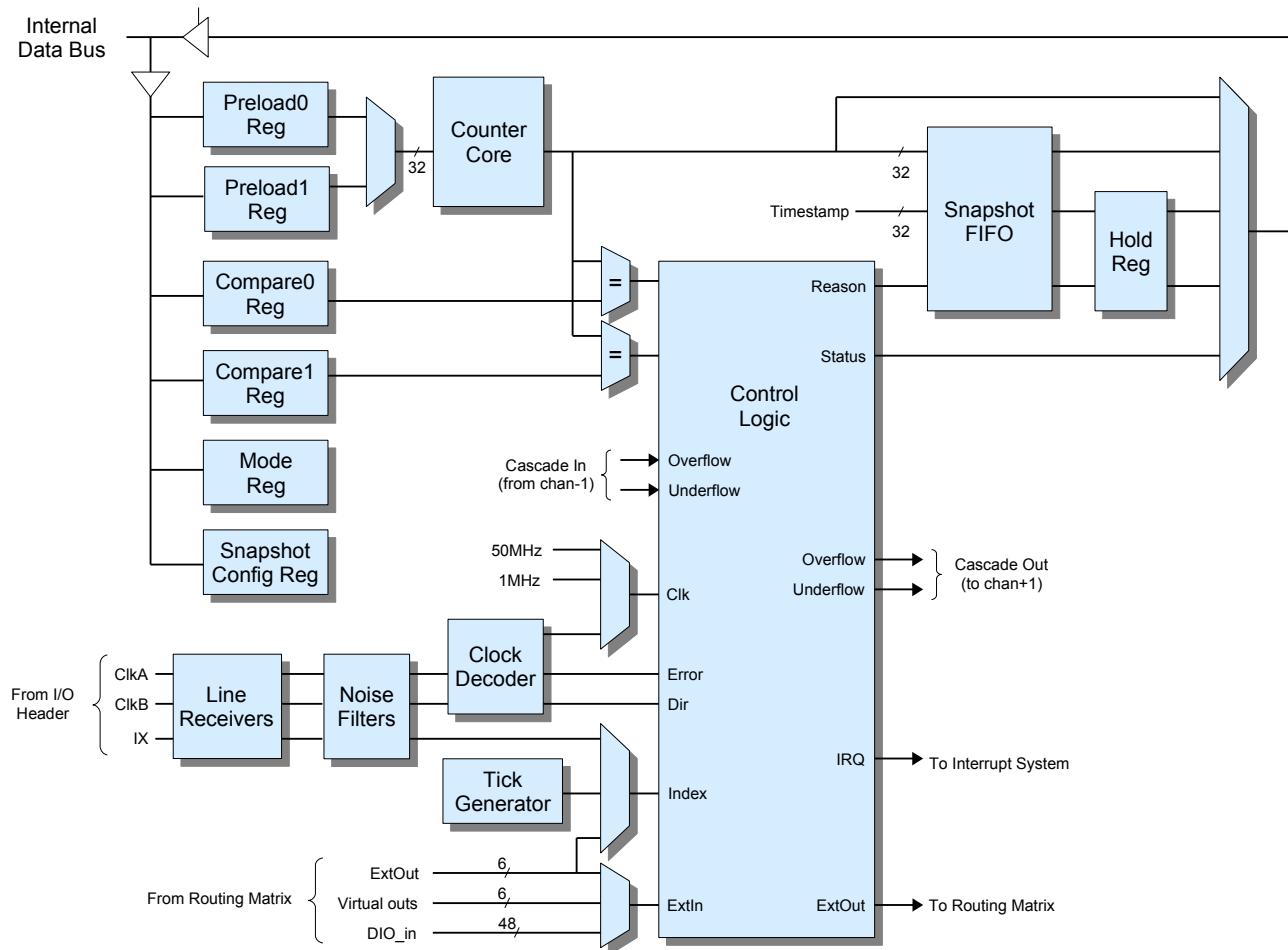
Chapter 7: Counters

7.1 Introduction

The model 826 board has six identical 32-bit programmable counter channels, numbered 0 to 5. Each counter channel can implement a complete solution for a variety of common applications, including incremental encoder interface, event counter, timer, pulse generator, PWM generator, pulse width measurement, period measurement, and frequency measurement. See “Application Connections” for tips on connecting external signals to counter channels, and “Common Applications” for programming strategies.

As shown in Figure 5, each channel can connect to as many as five external signals. Three input signals (ClkA, ClkB, and IX) are accessible through dedicated header pins. Two additional signals (input ExtIn and output ExtOut) can be routed to general-purpose digital I/O pins if physical access to these signals is needed.

Figure 5: Counter channel (1 of 6)



7.1.1 ClkA, ClkB and IX Signals

A channel can accept external signals on its ClkA and ClkB inputs consisting of either a single-phase or quadrature-encoded clocks. The maximum count rate is 25MHz regardless of external clock type. Single-phase clocks having exactly 50 percent duty rate can be counted at up to 25MHz; the maximum count frequency must be derated for other duty rates (see Specifications for details). Quadrature clocks up to 25 MHz (x1 multiplier), 12.5 MHz (x2), and 6.25 MHz (x4) are supported, with suitable derating for deviations from 90 degree clock phasing.

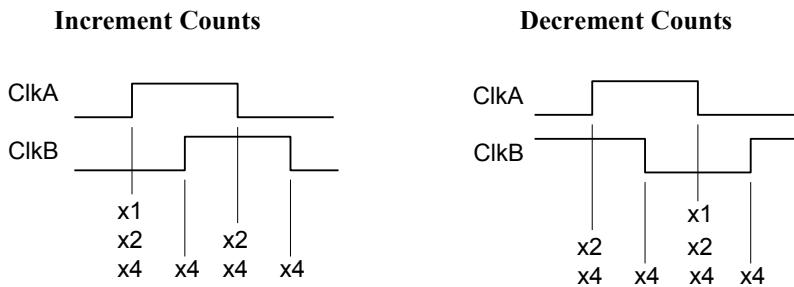
The ClkA, ClkB, and IX inputs employ differential RS-422 line receivers for buffering and noise immunity. All line receivers have built-in termination resistors. The clock and IX inputs are compatible with differential RS-422 signal pairs as well as single-ended TTL and 5V CMOS signals.

Each line receiver is followed by a noise filter that can be used to remove glitches. See [S826_CounterFilterWrite](#) for details.

7.1.2 Quadrature Decoder

When connected to quadrature-encoded clock signals, the ClkA and ClkB signals are processed by a quadrature decoder circuit to produce count direction, count enable, and quadrature error signals. Figure 6 shows the clock events that affect counting for each of the possible clock multipliers.

Figure 6: Quadrature Decoding



For example, with a x1 (“times 1”) multiplier, the counts will only change when ClkB is low; the core will count up on the rising edge of ClkA and down on the falling edge of ClkA.

If the decoder detects an encoding error, it will generate a special error snapshot (see [S826_Snapshots](#) below) and set an internal error flag. This will happen when ClkA and ClkB transition within 20 ns of each other (which should never occur in normal operation). This can be caused by various conditions, including signal noise on ClkA/ClkB or excessively high input clock frequency. The error snapshot serves as a warning that the counter core may no longer contain an accurate counts value.

7.1.3 ExtIn Signal

Some applications may require an additional external input (ExtIn) signal. If needed, this signal can be routed from any general-purpose digital I/O (DIO) channel, any virtual digital output channel, or the ExtOut signal of any counter channel (see [Section 7.3.12](#) for details). When ExtIn is routed from a DIO channel, the signal appearing on the DIO channel's connector pin must conform to the timing constraints of the DIO subsystem as explained in “Pin Timing”.

The ExtIn signal can be configured to behave as either a count or preload permissive by programming the IM field in the Mode register. See [S826_CounterModeWrite](#) for details.

7.1.4 ExtOut Signal

Some counter applications (e.g., pulse or PWM generator) may require a physical output from the counter. In such cases, the counter's ExtOut signal may be routed to a general-purpose DIO channel (see [Section 8.3.12](#) for details). When so routed, the DIO channel will act as a dedicated counter output, while the DIO input and edge detection functions continue to operate normally.

When routed to a DIO, the ExtOut signal timing is constrained by the DIO subsystem as explained in “Pin Timing”. In particular, the ExtOut signal may be delayed up to 20 ns before it appears on the DIO connector pin. Note also that short, positive output pulses may require the addition of external, supplemental pull-up resistance on the DIO pin to attain sufficiently fast rise time.

A channel's ExtOut signal will be asserted only when the channel is running. When the channel is halted, ExtOut is held at the inactive state, as defined by its configured polarity.

7.1.5 Snapshots

A “snapshot” consists of three values that are simultaneously sampled in response to a trigger: the counts contained in the counter core, the timestamp (which indicates the time the snapshot was captured), and a set of “reason” flags that indicate the type of event (or types of events, if multiple events occurred at the same time) that triggered the snapshot.

Snapshots are stored in the Snapshot FIFO. Snapshots are written to the FIFO as they occur, whereas the host may read them from the FIFO at any convenient time. The FIFO stores up to 16 snapshots. When the FIFO is full, a subsequent trigger will cause a new snapshot to be stored in the FIFO and the oldest snapshot in the FIFO will be deleted.

The Hold register caches a snapshot while it is being read by the host. The snapshot timestamp and reason code are copied to the Hold register when the snapshot counts are read. The host will then read the cached timestamp and reason code from the Hold register, thus ensuring the three snapshot values will remain correlated if a FIFO overflow occurs while the snapshot is being read.

A snapshot may be captured in response to various types of hardware and software triggers. All of the hardware trigger types can be individually enabled through the snapshot configuration register. Snapshots may be triggered when:

- The S826_CounterSnapshot function is called (i.e., a “soft” snapshot).
- The counter matches a compare register.
- Transitions occur on the Index input.
- Transitions occur on the ExtIn input.
- The counter reaches zero counts.
- A quadrature clock encoding error is detected. Once this happens, no further error-triggered snapshots can be captured until the error is cleared. The resulting error snapshot enables the application to determine the counts at the moment the error occurred. See S826_CounterSnapshotRead for further information.

Multiple counters can use the same snapshot trigger source or sources. For example, two or more counters could be configured to capture snapshots in response to a signal from a common DIO channel, thus enabling an external signal to simultaneously trigger snapshots on all affected counters. Similarly, a virtual digital output channel could be used as a common trigger, thereby allowing software to simultaneously trigger snapshots on the affected counters.

7.1.6 Preloading

The counter core can be “preloaded” (parallel-loaded) from the Preload0 and Preload1 registers in response to various hardware and software preload triggers. All of the hardware triggers can be individually enabled through the mode register. Preloads may be triggered:

- When the S826_CounterPreload function is called (i.e., a “soft” preload).
- When the channel state switches from halted to running.
- When the counter reaches zero counts.
- When the counter matches a compare register.
- When transitions occur on the Index input.
- While the Index input is held at its active level. This has the effect of holding the counter core at the preload value while Index is asserted.

The mode register's BP bit specifies whether both preload registers will be used or only Preload0. Preload0 is active (selected) by default when the channel state switches from halted to running. When a preload occurs, the core is first preloaded from the preload register and then the active register selector will change.

The preload mechanism behaves as shown in the following table. For example, if both preload registers are being used (BP=1) and a preload occurs because the counts reached zero, the core will be preloaded from the active preload register and then the other preload register will be activated.

Preload Behavior

Mode Register BP bit	Preload Trigger Type	Counter Core Loads From	Activated After Preload
0	Any	Preload0	Preload0
1	Zero Counts Reached	Active preload	Alternate preload
	Any except Zero Counts Reached	Preload0	Preload1

Preload triggers are prioritized. If two simultaneous preload triggers occur, the one with the highest priority is considered to be the cause of the preload and the preload mechanism will behave accordingly.

Some types of preload triggers are enabled and disabled by ExtIn when ExtIn is configured as a preload permissive (see “ExtIn Signal”). These triggers are disabled when ExtIn is negated and enabled with ExtIn is asserted.

Preload Trigger Type	Priority	Enabled/Disabled by ExtIn (when ExtIn is configured as preload permissive)
Channel switched to running state	7 (highest)	No
Zero counts reached	6	No
Compare1 match	5	Yes
Compare0 match	4	Yes
Index rising edge	3	Yes
Index falling edge	2	Yes
Index active level	1	Yes
Soft preload	0 (lowest)	Yes

7.1.7 Tick Generator

Each counter channel has an independent tick generator that can be used to create counting gates; this is useful for frequency measuring applications. The generator can produce periodic pulses at intervals ranging from one microsecond to ten seconds in decade steps. The channel's external IX input or the ExtOut output from any counter channel can be used in lieu of the internal tick generator if a custom gate time is needed.

7.1.8 Cascading

Limited cascading of counter channels is supported. Each channel receives overflow and underflow signals from the adjacent lower channel (channel 0 receives from channel 5). A channel may be cascaded onto the adjacent lower channel by setting K=4 in its mode register (see S826_CounterModeWrite). Note that when two channels are cascaded, only the count function is extended; the snapshot and preload mechanisms will continue to operate independently. Each cascade is limited to two counters.

7.1.9 Status LEDs

Each counter channel is associated with a status LED that indicates clock pulses are detected on that channel. The LED flashes at a constant rate while the clock signal is toggling. The LEDs are labeled E0-E5, corresponding to counter channels 0-5 respectively.

7.1.10 Reset State

Upon board reset, all counter channels are set to the “halted” state (see S826_CounterStateWrite for details). In addition, a board reset will also zero the Mode, Preload, and Compare registers.

7.2 Connectors J4/J5

Two 26-pin headers, J4 and J5, bring out connections from the board's six counter channels to external field wiring. J4 is used for counter channels 0-2 and J5 for channels 3-5.

J4 and J5 Pinouts

J4 - Encoder channels 0-2				J5 - Encoder channels 3-5					
Pin	Name	Function	Chan	Pin	Name	Function	Chan		
1	+A0	Differential A clock inputs	0	1	+A3	Differential A clock inputs	3		
2	-A0			2	-A3				
3	GND	Power supply return		3	GND	Power supply return			
4	+B0	4		+B3					
5	-B0	Differential B clock inputs		5	-B3	Differential B clock inputs			
6	+5V	+5V power output		6	+5V	+5V power output			
7	+I0	Differential IX inputs		7	+I3	Differential IX inputs			
8	-I0			8	-I3				
9	GND	Power supply return		9	GND	Power supply return			
10	+A1	1	10	+A4	Differential A clock inputs	4			
11	-A1		Differential A clock inputs	11			-A4		
12	+5V		+5V power output	12	+5V		+5V power output		
13	+B1		Differential B clock inputs		13		+B4	Differential B clock inputs	
14	-B1				14		-B4		
15	GND		Power supply return	15	GND		Power supply return		
16	+I1		Differential IX inputs		16		+I4	Differential IX inputs	
17	-I1				17		-I4		
18	+5V		+5V power output	18	+5V		+5V power output		
19	+A2	Differential A clock inputs	2	19	+A5	Differential A clock inputs	5		
20	-A2			20	-A5				
21	GND	Power supply return		21	GND	Power supply return			
22	+B2	22		+B5					
23	-B2	Differential B clock inputs		23	-B5	Differential B clock inputs			
24	+5V	+5V power output		24	+5V	+5V power output			
25	+I2	Differential IX inputs		25	+I5	Differential IX inputs			
26	-I2			26	-I5				

7.2.1 Counter Signals

Each counter channel has six input signals on its header connector: A+, A-, B+, B-, X+, and X-. The header also has power and ground pins that can be used to supply operating power to external devices such as incremental encoders. The following table details the header pins that are available for each channel and their recommended usage.

External signal connections to a counter channel

Function	Pin Name	Signal Type	Clock Source							
			Quadrature	Single-phase	Internal					
ClkA	A+	RS-422	ClockA+	Clock+	NC					
		TTL/CMOS	ClockA	Clock	NC					
	A-	RS-422	ClockA-	Clock-	NC					
		TTL/CMOS	NC	NC	NC					
ClkB	B+	RS-422	ClockB+	NC	NC					
		TTL/CMOS	ClockB	NC	NC					
	B-	RS-422	ClockB-	NC	NC					
		TTL/CMOS	NC	NC	NC					
			NC							
			NC	Internal / None						
IX	X+	RS-422	NC	NC						
		TTL/CMOS	IX	NC						
	X-	RS-422	IX-	NC						
		TTL/CMOS	NC	NC						
ExtIn	Note 1		Auxiliary counter input. The behavior of this signal is configurable.							
ExtOut	Note 1		Counter output. The behavior of this signal is programmable.							
Device Power	GND	POWER	5V power supply return and ground reference for all logic signals.							
	+5V	POWER	+5VDC power. This can be used to power external devices such as incremental encoders. The total 5V current for all six counter channels is limited to 500mA.							
Notes										
1. If used, this signal must connect to a DIO channel through the board's DIO signal routing matrix. Refer to the DIO documentation for details of the routing matrix and electrical characteristics of the DIO channels.										

7.2.2 Application Connections

Typical counter connections for common applications

Application	Signals				
	ClkA	ClkB	IX	ExtIn	ExtOut
Encoder Interface	Phase A clock	Phase B clock	Snapshot trigger Preload trigger NC	Count enable NC	NC
Event Counter	Event clock	NC	Snapshot trigger Preload trigger NC	Count enable NC	NC
Pulse Generator	NC	NC	Pulse trigger NC	Pulse trigger NC	Pulse output
PWM Generator	NC	NC	Output enable NC	NC	PWM output
Pulse Width Measurement	NC	NC	Signal to measure	NC	NC
Period Measurement	NC	NC	Signal to measure	NC	NC
Frequency Measurement	NC	NC	External time gate NC	NC	NC

7.3 Programming

7.3.1 S826_CounterSnapshotRead

The S826_CounterSnapshotRead function reads a snapshot from a counter channel.

```

int S826_CounterSnapshotRead(
    uint board,      // board identifier
    uint chan,       // channel number
    uint *counts,    // pointer to counts buffer
    uint *tstamp,    // pointer to timestamp buffer
    uint *reason,    // pointer to reason buffer
    uint tmax        // maximum time to wait
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

counts

Pointer to a buffer that will receive the latched counts value. Set to NULL to ignore counts.

tstamp

Pointer to a buffer that will receive the timestamp. Set to NULL to ignore timestamp.

reason

Pointer to a buffer that will receive the reason flags, which indicate what caused the snapshot. When a flag bit = '1', the snapshot was caused by that event type. More than one event may have occurred at the same time, so multiple bits may be set.

bit	Description
8	Quadrature error
7	Soft snapshot (caused by calling S826_CounterSnapshot)
6	ExtIn rising edge
5	ExtIn falling edge
4	Index rising edge
3	Index falling edge
2	Zero counts reached
1	Compare1 match
0	Compare0 match

Set to NULL to ignore the reason.

tmax

Maximum time, in microseconds, to wait for data. See Event-Driven Applications for details.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function reads a previously acquired snapshot from the snapshot FIFO, consisting of the counts at a particular moment in time and the associated timestamp and reason flags. Each snapshot can be read only one time; when a snapshot is read, it is removed from the FIFO and cannot be read again.

Snapshots may be acquired in response to asynchronous events. In such cases, snapshots may occur at any time. However, if snapshots are not automatically acquired, or for some other reason it is necessary to invoke a snapshot under program control, the application program must call S826_CounterSnapshot to capture a new snapshot before calling this function to read the snapshot.

The reason flags indicate the type of event that caused the snapshot. If two or more simultaneous events would each cause a snapshot, all of the associated reason flags will be set.

Quadrature-encoded clocks are monitored for encoding errors as described in Quadrature Decoder. When an encoding error is detected, a snapshot is captured (with reason bit 8 set) and an internal error flag is set. While the error flag remains set, subsequent encoding errors will be ignored and will not trigger new snapshots (though other types of triggers will continue to cause snapshots). The error flag is automatically cleared when this function reads the error-triggered snapshot, or when the channel is halted, or when the snapshot FIFO becomes empty. The latter case ensures that the error flag will be cleared if the error-triggered snapshot is lost due to FIFO overflow.

This function can operate in either blocking or non-blocking mode, depending on the value of *tmax*. If *tmax* is zero, the function will return immediately. If *tmax* is greater than zero, the calling thread will block until a snapshot is available or *tmax* elapses. The function will return either S826_ERR_OK or S826_ERR_FIFOOVERFLOW if a snapshot was read, or S826_ERR_NOTREADY if no snapshot is available. S826_ERR_FIFOOVERFLOW indicates that a snapshot was successfully read, but the channel's snapshot FIFO overflowed (one or more snapshots were lost); the FIFO overflow status will be automatically cleared when the function returns.

When this function is blocking, it will immediately return S826_ERR_CANCELLED if S826_CounterWaitCancel is called by another thread, or S826_ERR_BOARDCLOSED if S826_SystemClose is called. In either case, the counts, *tstamp* and *reason* values will be invalid.

7.3.2 S826_CounterWaitCancel

The S826_CounterWaitCancel function cancels a blocking wait on a counter channel.

```
int S826_CounterWaitCancel(
    uint board,      // board identifier
    uint chan       // channel number
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5, for which waiting is to be canceled.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function cancels blocking on a counter channel so that another thread, which is blocked by S826_CounterSnapshotRead while waiting for a snapshot, will return immediately with S826_ERR_CANCELLED.

7.3.3 S826_CounterCompareWrite

The S826_CounterCompareWrite function writes to a compare register.

```
int S826_CounterCompareWrite(
    uint board,      // board identifier
    uint chan,       // channel number
    uint regid,     // register identifier
    uint counts     // counts value
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan
Counter channel number in the range 0 to 5.

regid
Selects Compare register: 1 = Compare1 register, 0 = Compare0 register.

counts
Value to be written to the Compare register.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.4 S826_CounterCompareRead

The S826_CounterCompareRead function returns the contents of a compare register.

```
int S826_CounterCompareRead(
    uint board,      // board identifier
    uint chan,       // channel number
    uint regid,      // register identifier
    uint *counts     // counts value
);
```

Parameters

board
826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan
Counter channel number in the range 0 to 5.

regid
Selects Compare register: 1 = Compare1 register, 0 = Compare0 register.

counts
Pointer to a buffer that will receive the contents of the selected Compare register.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.5 S826_CounterSnapshot

The S826_CounterSnapshot function invokes an immediate snapshot.

```
int S826_CounterSnapshot(
    uint board,      // board identifier
    uint chan,       // channel number
);
```

Parameters

board
826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan
Counter channel number in the range 0 to 5.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function forces a snapshot to be captured immediately. It is typically used to capture a snapshot under program control prior to reading a counter. This is one of two methods of reading the core's instantaneous counts; the other method is S826_CounterRead. This function may be called at any time when the counter channel is running.

7.3.6 S826_CounterRead

The S826_CounterRead function reads the counter core's instantaneous counts.

```
int S826_CounterRead(
    uint board,      // board identifier
    uint chan,      // channel number
    uint *counts    // counts value
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

counts

Pointer to a buffer that will receive the instantaneous counts from the counter core.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function directly reads and returns the value currently stored in the counter core. If a timestamp is also needed, the application can call this function and also call S826_TimestampRead, or it may trigger a soft snapshot and then read the snapshot.

7.3.7 S826_CounterPreloadWrite

The S826_CounterPreloadWrite function writes to a preload register.

```
int S826_CounterPreloadWrite(
    uint board,      // board identifier
    uint chan,      // channel number
    uint reg,       // register identifier
    uint counts    // counts value
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

reg

Selects the preload register to write to: 0 = Preload0 register, 1 = Preload1 register.

counts

The 32-bit value to be written to the selected Preload register.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

The counts value is immediately written to the selected preload register when this function executes. The counter core is not affected until the next core preload occurs.

7.3.8 S826_CounterPreloadRead

The S826_CounterPreloadRead function reads a preload register.

```
int S826_CounterPreloadRead(
    uint board,      // board identifier
    uint chan,       // channel number
    uint reg,        // register identifier
    uint *counts    // counts value
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

reg

Selects the preload register to write to: 0 = Preload0 register, 1 = Preload1 register.

counts

Pointer to a buffer that will receive the counts from the selected Preload register.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.9 S826_CounterPreload

The S826_CounterPreload function manually invokes a core preload from the Preload0 register.

```
int S826_CounterPreload(
    uint board,      // board identifier
    uint chan,       // channel number
    uint level,      // preload signal level
    uint sticky      // make level "sticky"
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

level

Effective preload signal level: 1 = invoke preload, 0 = don't invoke preload. This is ignored if sticky=0.

sticky

Persistence: 1 = maintain level until reprogrammed, 0 = strobe trigger (level is ignored).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

If sticky is false, the level will be ignored. In this case, the channel's software preload command will be strobed active and then immediately return to its inactive state, thus causing Preload0 to be copied to the counter core.

If sticky is true, the specified level will be continuously applied to the channel's software preload command until changed by a subsequent call to this function. Typically, sticky will only be asserted if the counter is operating as a Pulse Generator, and only when output retrigerring is enabled. When sticky and level are both '1', the counter will continuously preload from Preload0, thus causing the pulse output to go active and remain active until the function is called again to set the level to '0'; at that time, the counter will be allowed to resume counting.

7.3.10 S826_CounterStateWrite

The S826_CounterStateWrite function starts or stops channel operation.

```
int S826_CounterStateWrite(
    uint board,      // board identifier
    uint chan,       // channel number
    uint state       // channel state
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

state

Channel state: 0 = halted, 1 = running.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function is used to start and stop counter channel operation, thus placing the channel in either the "running" or "halted" state. A channel's operating mode should be configured before switching it to the running state; this is done by calling S826_CounterModeWrite. A channel will operate as specified by the mode register while it is in the running state.

In the halted state:

- Counting and preloading are not allowed.
- Counter core is reset to zero counts.
- Snapshot register is marked empty.
- Output is held inactive.
- Quadrature error is reset.

The mode, preload, compare, and snapshot configuration registers are not affected when a channel transitions between halted and running states. This function has no effect if used to start a channel that is already running, or to stop a channel that is already halted.

This function can be used to zero the counter core by calling it once to halt the channel (and zero the counts) and again to start the channel running. Another way to zero the counts is to zero the Preload0 register (by calling S826_CounterPreloadWrite) and then transfer the preload value to the core (via S826_CounterPreload).

7.3.11 S826_CounterStatusRead

The S826_CounterStatusRead function returns information about a counter channel's status.

```
int S826_CounterStatusRead(
    uint board,      // board identifier
    uint chan,       // channel number
    uint *status     // channel status info
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

status

Pointer to a buffer that will receive the status:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	RUN	0	ST	0	0	0	0	0	0	0	0	0	0	PLS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit Description

- RUN Channel enable: '1' = running, '0' = halted. This is controlled by S826_CounterStateWrite.
- ST Sticky preload command. This is controlled by S826_CounterPreload.
- PLS Preload selector. This indicates the active preload register: '1' = Preload1, '0' = Preload0.
PLS can be '1' only if mode register bit BP=1 (see S826_CounterModeWrite).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.12 S826_CounterExtInRoutingWrite

The S826_CounterExtInRoutingWrite function selects the signal source for a counter channel's ExtIn input.

```
int S826_CounterExtInRoutingWrite(
    uint board,      // board identifier
    uint chan,       // counter channel number
    uint route       // routing configuration
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

route

Signal to be connected to ExtIn (this is ignored if mode register field IM=0):
0 to 47 = DIO channel 0 to 47;
48 to 53 = counter channel 0 to 5 ExtOut;
54 to 59 = virtual digital output channel 0 to 5.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function will route the counter channel's ExtIn input to a general-purpose digital I/O (DIO) channel so that an external signal (applied to the DIO channel's I/O pin) can control the channel's ExtIn input. Alternatively, the ExtIn input may internally routed to any counter channel's ExtOut output, or to any virtual digital output channel.

7.3.13 S826_CounterExtInRoutingRead

The S826_CounterExtInRoutingRead function returns a counter channel's ExtIn signal routing configuration.

```
int S826_CounterExtInRoutingRead(
    uint board,      // board identifier
    uint chan,       // counter channel number
    uint *route      // routing configuration
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

route

Pointer to buffer that will receive the ExtIn routing configuration. The format of the returned value is identical to the *route* argument used in the S826_CounterExtInRoutingWrite function.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.14 S826_CounterSnapshotConfigWrite

The S826_CounterSnapshotConfigWrite function programs the snapshot configuration register.

```
int S826_CounterSnapshotConfigWrite(
    uint board,      // board identifier
    uint chan,       // counter channel number
    uint cfg,        // snapshot configuration flags
    uint mode        // 0=write, 1=clear bits, 2=set bits
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

cfg

Flags with 'E' prefix determine the types of events that will capture snapshots: '1' = enable capturing, '0' = disable capturing. Each flag with 'R' prefix determines whether the corresponding 'E' flag will be automatically cleared upon snapshot capture, thus preventing subsequent events of that type from invoking snapshots.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	RER	REF	RXR	RXF	RZ	RMI	RMO	0	0	0	0	0	0	0	0	0	EER	EEF	EXR	EXF	EZ	EM1	EM0		

Flag	Snapshot trigger event
ER	ExtIn leading edge
EF	ExtIn falling edge
XR	Index leading edge
XF	Index falling edge
Z	Zero counts reached
M1	Compare1 register matches counter core
M0	Compare0 register matches counter core

mode

Write mode for cfg: 0 = write, 1 = clear bits, 2 = set bits (see “Atomic Read-Modify-Write”).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function programs the snapshot configuration register. A snapshot will be captured when an 'E' flag is programmed to '1' and the corresponding event occurs. Upon snapshot capture, the associated 'E' flag will be automatically cleared if the corresponding 'R' flag is set.

Independent of these flags, snapshots may also be captured upon quadrature clock error or in response to calls to S826_CounterSnapshot.

7.3.15 S826_CounterSnapshotConfigRead

The S826_CounterSnapshotConfigRead function reads the snapshot configuration register.

```
int S826_CounterSnapshotConfigRead(
    uint board,      // board identifier
    uint chan,       // counter channel number
    uint *cfg        // configuration flags
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

cfg

Pointer to buffer that will receive the configuration flags. The format of the returned value is identical to the events value used in the S826_CounterSnapshotConfigWrite function. Note that 'E' flags (flags with 'E' name prefix) may be automatically reset in response to captured snapshots.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.16 S826_CounterFilterWrite

The S826_CounterFilterWrite function configures the IX, CLKA, and CLKKB noise filters.

```
int S826_CounterFilterWrite(
    uint board,      // board identifier
    uint chan,      // counter channel number
    uint cfg        // filter configuration
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

cfg

Filter configuration:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IX	CK																													T	

Field Description

IX Enable IX filter: '1' = enable, '0' = disable.

CK Enable CLKA/CLKB filters: '1' = enable, '0' = disable.

T Filter time interval specified as multiple of 20ns, common to IX, CLKA and CLKB input filters.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

Each of the IX, CLKA, and CLKB input signals has a dedicated noise filter. This function enables and disables the filters and programs the filter time interval. The CLKA and CLKB filters are enabled or disabled as a group, whereas the IX filter is independently enabled or disabled. The filter time interval, T, is common to all enabled filters.

A filter's output will not change state until its input has held a constant state for $T * 20\text{ns}$. The maximum T value is 65535, corresponding to a filter time of 1.3107ms. When a filter is enabled, it will delay its input signal by $T * 20\text{ns}$. Note that when the noise filter is enabled for the CLKA/CLKB inputs (CK = '1'), the counter's maximum clock frequency is reduced; the maximum frequency reduction is inversely proportional to T.

7.3.17 S826_CounterFilterRead

The S826_CounterFilterRead function reads the configuration of the IX, CLKA, and CLKB noise filters.

```
int S826_CounterFilterRead(
    uint board,      // board identifier
    uint chan,      // counter channel number
    uint *cfg        // filter configuration
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

cfg

Pointer to buffer that will receive the configuration. The format of the returned value is identical to the cfg value used in the S826_CounterFilterWrite function.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.18 S826_CounterModeWrite

The S826_CounterModeWrite function configures a counter channel's operating mode.

```
int S826_CounterModeWrite(
    uint board,      // board identifier
    uint chan,       // channel number
    uint mode        // operating mode
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

mode

Channel operating mode (see “Common Applications” for examples):

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	IP	IM	0	0	0	TP	NR	UD	BP	OM	OP				TP		TE		TD		K		XS									

IP ExtIn signal polarity. This is ignored if IM=0.

- 0 Normal polarity.
- 1 Inverted polarity.

IM ExtIn mode. If IM>0 then ExtIn is routed to a DIO pin as specified by S826_CounterExtInRoutingWrite.

- 0 ExtIn input is not used. The counter's ExtIn input will effectively be hardwired to logic '1'.
- 1 ExtIn input is a count enable permissive.
This is typically used with incremental encoders and event counting, with ExtIn acting as a count enable.
- 2 ExtIn input is a preload enable permissive.
This is typically used when operating as a pulse generator, with ExtIn used as a pulse trigger.

**TP bit Preload enables. Determines event(s) that will cause a preload register to be copied to the counter.
Each bit enables preloads for one type of event: '1' = enable, '0' = disable.**

- 24 Preload upon start (when channel switches from halted to running)
- 16 Preload upon Index active level.
- 15 Preload upon Index leading edge.
- 14 Preload upon Index falling edge.
- 13 Preload upon zero counts reached.
- 12 Preload when Compare1 matches the counter core.
- 11 Preload when Compare0 matches the counter core.

NR	Preload permissive. Typically used for “one-shot” pulse generator applications.
0	Allow preloading any time. This can be used for applications such as a retriggerable one-shot.
1	Allow preloading only when counts equal zero. This can be used for applications such as a non-retriggerable one-shot. This applies to both hardware- and software-induced (via S826_CounterPreload function) preloads.
UD	Count direction
0	Normal count direction.
1	Reverse count direction.
BP	Preload toggle enable.
0	Use only Preload0. The Preload1 register will not be used.
1	Use both preload registers. This is typically used for PWM output.
OM	ExtOut mode. Determines when the channel's ExtOut output signal is active. The ExtOut signal may be routed to a DIO pin by calling S826_DioOutputSourceWrite.
0	Output always inactive.
1	Output pulses active upon Compare register snapshot.
2	Output active when Preload1 register is selected. This is typically used for PWM output.
3	Output active when the counts are not zero; useful for pulse generator applications.
4	Output active when the counts are zero; useful for watchdog timer applications.
OP	ExtOut signal polarity.
0	Normal polarity
1	Inverted
TE	Count enable trigger. Determines event(s) that will cause counting to be enabled.
0	Enable counting at start-up (i.e., when S826_CounterModeWrite is called to switch to running mode).
1	Enable counting upon Index leading edge.
2	Enable counting upon preload.
3	reserved -- do not use.
TD	Count disable trigger. Determines event(s) that will cause counting to be disabled.
0	Never disable counting.
1	Disable counting upon Index falling edge.
2	Disable counting upon zero counts reached.
3	reserved -- do not use
K	Clock mode.
0	External single-phase clock, increment on ClkA rising edge.
1	External single-phase clock, increment on ClkA falling edge.
2	Internal 1 MHz clock, increment every microsecond.
3	Internal 50 MHz clock, increment every 40 nanoseconds.
4	Link to adjacent channel's overflow/underflow outputs to this channel's overflow/underflow inputs (see “Cascading”). The adjacent channel's overflow/underflow output signals will cause this channel to increment/decrement. For channel 0, channel 5 is the adjacent channel; for all other channels N, the adjacent channel is N-1.
5	External quadrature clock, x1 multiplier.
6	External quadrature clock, x2 multiplier.
7	External quadrature clock, x4 multiplier.

XS	Index source.
0	External IX input, normal polarity.
1	External IX input, inverted.
2-7	ExtOut from counter channel 0-5 (e.g., 3 = ExtOut from channel 1).
8-15	Internal tick generator: 8 = 0.1 Hz, 9 = 1 Hz, 10 = 10 Hz, 11 = 100 Hz, 12 = 1 KHz, 13 = 10 KHz, 14 = 100 KHz, 15 = 1 MHz.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.19 S826_CounterModeRead

The S826_CounterModeRead function returns a counter channel's operating mode.

```
int S826_CounterModeRead(
    uint board,      // board identifier
    uint chan,       // channel number
    uint *mode       // operating mode
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chan

Counter channel number in the range 0 to 5.

mode

Buffer that will receive the channel operating mode, as defined in “S826_CounterModeWrite”.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

7.3.20 Common Applications

This section shows simple examples of how to configure and operate counter channels for common applications. In most cases there are other ways to configure the counters to achieve similar functionality, and numerous variations are possible that are not discussed here.

Encoder Interface

Monitor a quadrature-encoded device (e.g., incremental encoder) using x4 clock multiplier. Capture a snapshot at any time by calling S826_CounterSnapshot.

Register	Value
Mode	0x00000070

Event Counter

Count pulses applied to the ClkA input. Capture a snapshot at any time by calling S826_CounterSnapshot. Reset the counts to zero by calling S826_CounterPreload.

Register	Value
Mode	0x00000000
Preload0	0x00000000

Periodic Timer

Periodically capture snapshots.

Register	Value
Mode	0x00402020
Snapshot Configuration	0x00000004
Preload0	Period in μ s

Delay Timer

Call S826_CounterPreload to start the timer, then S826_CounterSnapshotRead to wait for the delay time to elapse.

Register	Value
Mode	0x00400520
Snapshot Configuration	0x00000004
Preload0	Delay in μ s

Frequency Measurement

Measure frequency by counting clocks applied to the ClkA input for one second intervals. At the end of each interval, a snapshot is captured and the next measurement begins automatically. The snapshot counts indicate measured frequency in Hz.

Register	Value
Mode	0x00008009
Snapshot Configuration	0x00000010
Preload0	0x00000000

Period Measurement

Measure the period of a signal applied to the IX input by counting internal clocks during one input cycle. At the end of each cycle, a snapshot is captured and the next measurement begins automatically. The snapshot counts indicate measured period in μ s.

Register	Value
Mode	0x00008020
Snapshot Configuration	0x00000010
Preload0	0x00000000

Pulse Width Measurement

Measure the width of pulses applied to the IX input by counting internal clocks during one pulse. At the end of each pulse, a snapshot is captured and the next measurement begins automatically. The snapshot counts will indicate the measured pulse width in μ s.

Register	Value
Mode	0x00008020
Snapshot Configuration	0x00000009
Preload0	0x00000000

Pulse Generator

Generate an output pulse on ExtOut in response to a trigger signal applied to the IX input. Also, a software trigger can be invoked by calling S826_CounterPreload. ExtOut must be routed to a DIO channel (see S826_DioOutputSourceWrite).

Register	Value
Mode	0x004C0520 (retriggerable), or 0x00CC0520 (non-retriggerable)
Preload0	Pulse duration in μ s

PWM Generator

Output a pulse width modulation (PWM) signal on ExtOut, which must be routed to a DIO channel through the DIO_out signal routing matrix (see S826_DioOutputSourceWrite).

Register	Value
Mode	0x01682020
Preload0	PWM “on” time in μ s
Preload1	PWM “off” time in μ s

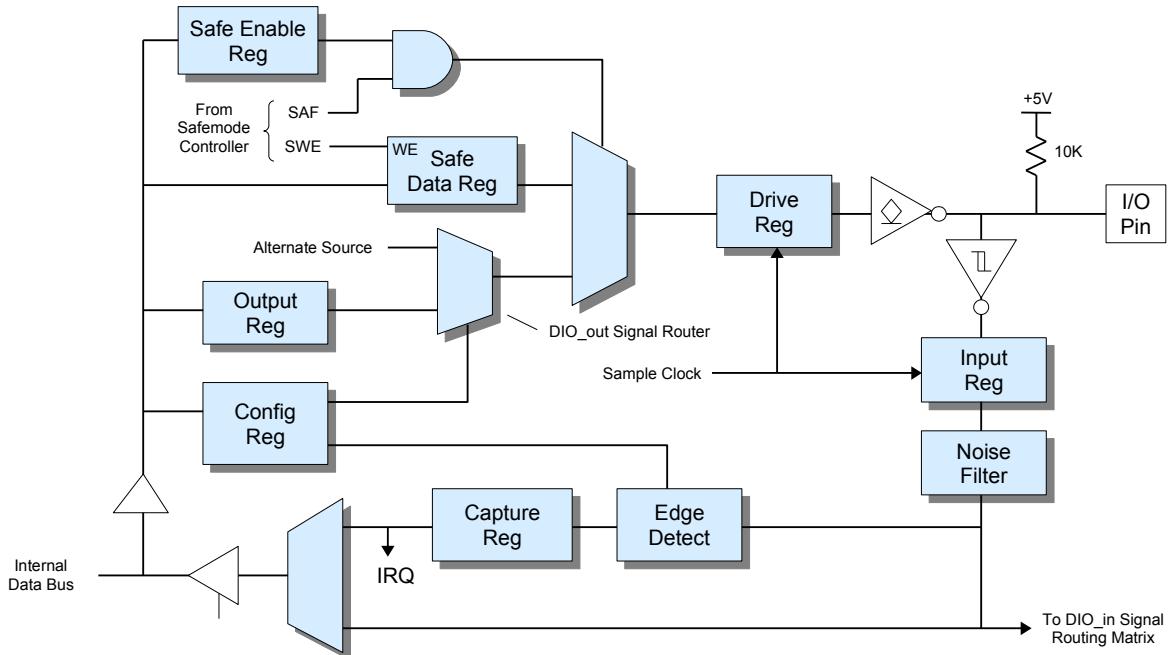
Synchronized PWM generators can be implemented by configuring the PWM channels as hardware triggered one-shots (pulse generators). Use an additional channel to generate a common trigger for the PWM channels; this channel should be configured to generate periodic output pulses at the desired PWM frequency. The duty cycle of each PWM channel is controlled by adjusting its output pulse width.

Chapter 8: Digital I/O

8.1 Introduction

The 826 board has 48 general-purpose digital I/O (DIO) channels. On the output side, each channel has a one-bit, writable output register, a synchronous drive register, and an inverting open-drain buffer that drives the channel's I/O pin. The open-drain buffer enables the pin to be driven low by the channel's output buffer or by an external signal. The pin, when not driven, is pulled up to +5V by a 10 kohm resistor. The output is high impedance when the board is unpowered so as to prevent unintended activation of an external solid state relay, if one is connected. The pin's physical state is sampled by an input register and then processed by a noise/debounce filter. The filter output is monitored by an edge detector and can also be directly read by the host computer.

Figure 7: DIO channel (1 of 48)



DIO signals are active-high on the local bus and active-low on the I/O pin. Writing a '1' to the output register causes the I/O pin to be driven low, whereas writing '0' allows the pin to be internally pulled up or driven high or low by an external circuit. Logic '0' must be written to the output register if the pin will be driven high by an external circuit; this will prevent high currents that could potentially damage the pin's output buffer.

The value read from a DIO channel indicates the filtered, sampled physical state of the I/O pin. If no external signals are driving the pin then the read value will equal the value stored in the drive register. The read value will differ from the drive register value if the drive register contains '0' while an external circuit drives the pin low.

Most of the host-accessible DIO registers support masked write operations so as to implement the atomic bit set and clear functions required for high performance, thread-safe operation.

8.1.1 Signal Routing Matrix

Each DIO channel connects to the board's internal signal routing matrix as shown in Figure 7. The matrix can be programmed to route the DIO connector pin to or from another interface (e.g., counter, watchdog, analog input system) so that the pin will act as a physical input or output for that interface.

The channel's DIO_out signal router consists of a data selector that can route either the DIO output register or an alternate source to the I/O pin. The pin will function as a general-purpose digital output when the DIO output register is selected. If the alternate source is selected, the pin state will be controlled by the alternate signal, but all DIO input functions (read, edge detection) will continue to operate normally. Each DIO is associated with a specific alternate source as explained in S826_DioOutputSourceWrite.

The DIO_in routing matrix connects the sampled DIO pin signal to other interfaces. The ADC trigger input and the six counter ExtIn inputs are connected to the DIO_in matrix so that any of these signals can be sourced from a DIO pin. When a DIO signal is routed to another interface via the DIO_in matrix, all of the DIO channel's input and output functions will continue to operate normally.

8.1.2 Safemode

Safemode is activated when the SAF signal (see Figure 7) is asserted. When operating in safemode, the DIO pin state is determined by the Safe Enable and Safe Data registers: when Safe Enable equals '1', the pin will be driven to the fail-safe value stored in Safe Data; when Safe Enable equals '0' the pin will output its normal runmode signal.

Upon board reset, the Safe Enable register is set to '1' so that the DIO pin will exhibit fail-safe behavior by default (i.e., it will output the contents of the Safe Data register when SAF is asserted). Fail-safe operation can be disabled for a DIO pin by programming its Safe Enable register to '0'.

The Safe Data register is typically programmed once (or left at its default value) by the application when it starts, though it may be changed at any time if required by the application. It can be written to only when the SWE bit is set to '1'. See "Safemode Controller" for more information about safemode.

8.1.3 Edge Capture

Every DIO channel includes an edge detection circuit and a capture flag register. When edge capturing is enabled, a channel's capture flag will be set when an edge is detected on its I/O pin. Each channel may be programmed to capture rising edges, falling edges, or both edges, or capturing may be disabled.

The API allows capture flags to be monitored by polling or, if the application is event driven, the calling thread can block while it waits for captured events. When blocking on edge capture events, the calling thread can specify a set of capture flags to wait for, and it can wait for either all of the events or any one event in the set.

When read by the host, capture flags are reset but remain enabled to capture future events. Edge events that occur on a channel while its capture flag is set will be lost. An input signal must hold for at least 20 ns after a transition for the transition to be reliably detected.

8.1.4 Pin Timing

The DIO subsystem is a fully synchronous system that is controlled by a 50 MHz sampling clock. The DIO pin drivers are updated and pin receivers are sampled once per cycle. As a result, outputs cannot change faster than the cycle time and inputs cannot be sampled faster than the cycle time.

Output registers are organized as two 24-channel groups. When these registers are written (via S826_DioOutputWrite), channels 0-23 will change simultaneously and channels 24-47 will also change simultaneously, but these two 24-channel groups are not guaranteed to change output states at the same time. Also, a DIO pin does not change output state immediately when its signal source (DIO output register or counter ExtOut) changes; it will be delayed for 20 ns due to the sampling clock.

When used as inputs, all 48 DIO channels are sampled simultaneously every 20 ns. As a result, the received signal on a DIO pin may be delayed up to 20 ns en route to its destination (e.g., DIO edge detector or read data, counter ExtIn input, or ADC trigger input), and input signal pulses shorter than 20 ns may not be recognized. The host reads pin states (via S826_DioInputRead) as two 24-channel groups (channels 0-23 and 24-47) in two separate read cycles. Consequently, channels within each group are guaranteed to be sampled simultaneously, but the two groups are not guaranteed to be associated with the same sample clock.

8.1.4.1 Noise Filter

Each DIO channel input circuit includes a noise filter that can be used to filter glitches (see S826_DioFilterWrite). A filter's output will change state only when its input has held constant for time T, the filter time interval. Consequently, when a DIO channel's filter is enabled, the sampled input signal will be delayed for an additional $T * 20$ ns en route to its destination, and input signal pulses shorter than $T * 20$ ns will not be recognized.

8.1.5 Reset State

DIO channels are forced to the following condition upon board reset:

- Output and Safe Data registers programmed to zero.
- Safe Enable registers programmed to all 1's.
- Output register is selected as I/O pin data source.
- Event capture disabled.

8.2 Connectors J2/J3

J2 Pinout - DIO channels 24-47

Pin	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	Even
DIO Channel	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	30	30	29	28	27	26	25	24	+5V	GND

J3 Pinout - DIO channels 0-23

Pin	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	Even
DIO Channel	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	+5V	GND

All even pin numbers on J2 and J3 connect to the power supply return. Odd pin numbers 1-47 are the active-low DIO channel I/O pins. Pin 49 is a +5V power output for low power loads such as solid state relay racks.

8.3 Programming

The DIO functions use individual bits to convey information about the DIO channels, wherein each bit represents the information for one channel. In such cases, the information for the 48 DIO channels is organized as an array of two bit quadlets (32-bit values), with each quadlet containing the information for 24 DIO channels:

The quadlet at array[0] is associated with DIO channels 0 to 23:

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIO	-	-	-	-	-	-	-	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

The quadlet at array[1] is associated with DIO channels 24 to 47:

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIO	-	-	-	-	-	-	-	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	

Typically, the DIO functions expect a pointer to a two-quadlet array, which must have been previously allocated by the program.

Although the DIO functions read or write values for all 48 DIO channels, the physical read or write operation is performed in steps; channels 0 to 23 first, as a group, and then channels 24 to 47 as another group. As a result, reads and writes do not sample or update all channels simultaneously. In general, channels 0 to 23 are sampled or updated concurrently as a group, and channels 24 to 47 are sampled or updated concurrently as a separate group.

8.3.1 S826_DioOutputWrite

The S826_DioOutputWrite function programs the DIO output registers.

```
int S826_DioOutputWrite(
    uint board,          // board identifier
    uint data[2],        // pointer to DIO data
    uint mode            // 0=write, 1=clear bits, 2=set bits
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to data array (see Section 8.3).

mode

Write mode for data: 0 = write, 1 = clear bits, 2 = set bits (see “Atomic Read-Modify-Write”).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

In mode zero, this function will unconditionally write new values to all DIO output registers. In modes one and two, the function will selectively clear or set any arbitrary combination of output registers; data bits that contain logic '1' indicate DIO output registers that are to be modified, while all other output registers will remain unchanged. Modes one and two can be used to ensure thread-safe operation as they atomically set or clear the specified bits.

8.3.2 S826_DioOutputRead

The S826_DioOutputRead function reads the programmed states of all DIO output registers.

```
int S826_DioOutputRead(
    uint board,          // board identifier
    uint data[2]         // pointer to data buffer
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to a buffer (see Section 8.3) that will receive the output register states.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function returns the output register states. Note that the returned values may not be the same as the physical I/O pin states in the case of pins that are externally driven or routed to a counter channel's output signal.

8.3.3 S826_DioInputRead

The S826_DioInputRead function reads the physical states of all DIO channel I/O pins.

```
int S826_DioInputRead(
    uint board,      // board identifier
    uint data[2]     // pointer to data buffer
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to a buffer (see Section 8.3) that will receive the physical I/O pin states.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function returns the sampled physical states of all DIO pins in the data buffer.

If this function is called immediately after calling S826_DioOutputWrite, the read data (sampled pin states) may not accurately reflect the previously written data. This happens because the DIO pins are driven by open-drain buffers with pull-up resistors. A pin will change state quickly when driven low, but when it is switched high, the state cannot change as quickly because additional time is required for the circuit capacitance to be charged through the pull-up resistor. The amount of time required for this depends on circuit capacitance; it can be shortened by decreasing the capacitance or by decreasing the pull-up resistance (by adding an external pull-up resistor).

In some applications, it may be desirable to have a DIO write function that will not return until all pins are stable. This can be implemented by calling S826_DioOutputWrite, and then polling with S826_DioInputRead until the expected states are read.

8.3.4 S826_DioSafeWrite

The S826_DioSafeWrite function programs the DIO Safe registers.

```
int S826_DioSafeWrite(
    uint board,      // board identifier
    uint data[2],    // pointer to safemode data
    uint mode        // 0=write, 1=clear bits, 2=set bits
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to data array (see Section 8.3) to be programmed into the Safe registers.

mode

Write mode for data: 0 = write, 1 = clear bits, 2 = set bits (see “Atomic Read-Modify-Write”).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function should only be called when the SWE bit is set (see S826_SafeWrenWrite). The function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1).

8.3.5 S826_DioSafeRead

The S826_DioSafeRead function returns the contents of the DIO Safe registers.

```
int S826_DioSafeRead(
    uint board,      // board identifier
    uint data[2]     // pointer to data buffer
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to a buffer (see Section 8.3) that will receive the Safe register states.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

8.3.6 S826_DioSafeEnablesWrite

The S826_DioSafeEnablesWrite function programs the DIO Safe Enable registers.

```
int S826_DioSafeEnablesWrite(
    uint board,        // board identifier
    uint enables[2]   // pointer to safemode enables
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

enables

Pointer to array of values (see Section 8.3) to be programmed into the Safe Enable registers.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function should only be called when the SWE bit is set (see S826_SafeWrenWrite). The function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1).

8.3.7 S826_DioSafeEnablesRead

The S826_DioSafeEnablesRead function returns the contents of the DIO Safe Enable registers.

```

int S826_DioSafeEnablesRead(
    uint board,          // board identifier
    uint enables[2]      // pointer to data buffer
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

enables

Pointer to a buffer (see Section 8.3) that will receive the Safe Enable register contents.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

8.3.8 S826_DioCapEnablesWrite

The S826_DioCapEnablesWrite function programs the edge sensitivity for DIO edge capturing.

```

int S826_DioCapEnablesWrite(
    uint board,          // board identifier
    uint rising[2],      // pointer to data buffer
    uint falling[2],     // pointer to data buffer
    uint mode            // write mode
);

```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

rising

Pointer to a buffer (see Section 8.3) that specifies rising edge sensitivity: '1'=capture rising edges, '0'=don't capture rising edges.

falling

Pointer to a buffer (see Section 8.3) that specifies falling edge sensitivity: '1'=capture falling edges, '0'=don't capture falling edges.

mode

Write mode for rising/falling: 0 = write, 1 = clear bits, 2 = set bits (see “Atomic Read-Modify-Write”).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function specifies the edges to be captured on DIO channels. It defines the edge sensitivity for each DIO channel in terms of the I/O pin voltage. For example, if falling edge sensitivity is enabled, edge capturing will occur when the I/O pin voltage transitions from 5V to 0V.

When mode is zero, all data flags are directly written to the capture enable registers. In modes one and two, the data flags determine which of the 48 DIO channels are to be affected; any flag that contains a logic '1' will cause the associated channel to be affected, while '0' will leave the channel unmodified.

After capturing has been enabled, it will remain enabled until disabled by this function or a board reset. Capturing is disabled on all channels following a board reset.

8.3.9 S826_DioCapEnablesRead

The S826_DioCapEnablesRead function returns the programmed edge sensitivity for DIO edge capturing.

```
int S826_DioCapEnablesRead(
    uint board,          // board identifier
    uint rising[2],      // pointer to data buffer
    uint falling[2]      // pointer to data buffer
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

rising

Pointer to a buffer (see Section 8.3) that receives rising edge sensitivity: '1'=capture rising edges, '0'=don't capture rising edges.

falling

Pointer to a buffer (see Section 8.3) that receives falling edge sensitivity: '1'=capture falling edges, '0'=don't capture falling edges.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function receives information about the types of edge events that will be captured, which were previously programmed by calling This function returns the sampled physical states of all DIO pins in the data buffer..

8.3.10 S826_DioCapRead

The S826_DioCapRead function waits for edge events on one or more DIO channels.

```
int S826_DioCapRead(
    uint board,          // board identifier
    uint chanlist[2],    // pointer to channel flags
    uint waitall,        // logic operator: 1=and (all), 0=or (any)
    uint tmax            // maximum time to wait
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chanlist

Pointer to channel flag bits (see Section 8.3). Upon entry, set flags indicate channels of interest. Upon exit, set flags indicate channels with captured edges.

waitall

Logic operator to apply to channels of interest: 1 = AND (return when all channels have events), 0 = OR (return when any channel has an event). This is ignored if tmax = 0.

tmax

Maximum time, in microseconds, to wait for the events of interest. See "Event-Driven Applications" for details.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function waits for edge events to be captured on an arbitrary set of DIO channels (the “channels of interest”) and then returns information about the events. Event capturing must have been previously enabled for channels of interest by calling S826_DioCapEnablesWrite.

Before calling the function, one or more chanlist bits must be set to identify the channels of interest. The function will modify chanlist to indicate channels of interest that have captured events, and it will reset the capture flag registers for all such indicated channels, thus re-enabling event capturing on those channels.

The function operates in either blocking or non-blocking mode depending on the value of tmax. If tmax is zero (non-blocking mode), the function will return immediately. If tmax is greater than zero (blocking mode), the calling thread will block until the capture criteria is satisfied or tmax elapses. In either case, some channels of interest may have captured events while others may not have; these will be indicated by chanlist when the function returns.

In blocking mode, the capture criteria is specified by waitall. Depending on waitall, the function will wait for events to be captured on either any, or all channels of interest. When waitall is true, the function will return when all channels of interest have captured events. When waitall is false, the function will return when any channel of interest has captured an event. The function will return S826_ERR_NOTREADY if tmax elapses before the capture criteria is satisfied.

In non-blocking mode, waitall is ignored and the function will never return S826_ERR_NOTREADY.

This function will return S826_ERR_CANCELLED if, while it is blocking, another thread calls S826_DioWaitCancel to cancel waits on any of the blocking channels. It will return immediately if the wait criteria is completely satisfied due to the wait cancellations, otherwise it will continue to block and return when all remaining wait criteria is satisfied. S826_ERR_BOARDCLOSED will be returned immediately if S826_SystemClose executes while this function is blocking. In either case, chanlist will be invalid when the function returns.

Thread-safe operation is guaranteed if the channels of interest for any given thread do not coincide with those of another thread. For example, thread safety is assured if a thread designates channels 1 and 3-5 as channels of interest while another thread designates channels 2 and 9.

8.3.11 S826_DioWaitCancel

The S826_DioWaitCancel function cancels a blocking wait on one or more DIO channels.

```
int S826_DioWaitCancel(
    uint board,           // board identifier
    uint chanlist[2]     // pointer to channel flags
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

chanlist

Pointer to DIO channel flag bits (see Section 8.3) that indicate channels for which waiting is to be cancelled.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function cancels blocking for an arbitrary set of DIO channels so that another thread, which is blocked by S826_DioCapRead while waiting for DIO edge events to be captured, will return immediately with S826_ERR_CANCELLED.

8.3.12 S826_DioOutputSourceWrite

The S826_DioOutputSourceWrite function assigns the signal sources for all DIO pins.

```
int S826_DioOutputSourceWrite(
    uint board,          // board identifier
    uint data[2]         // pointer to data buffer
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to a buffer (see Section 8.3) that specifies signal sources: 0=DIO output register, 1=alternate source.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function specifies the signal source that will be used to drive each DIO pin. Each DIO pin may be driven by its output register or by its alternate signal source. Upon board reset, all DIO channels assume their default configuration so that all DIO pins are driven by the DIO output registers (vs. alternate sources).

A DIO pin's signal source is determined by the corresponding bit in the data buffer. When the bit is set to '1' the pin will be driven by the channel's alternate signal source; when set to '0' (default) the pin will behave as a standard digital output, driven by the channel's output register.

The data[0] quadlet selects the signal sources for DIO channels 0 to 23:

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIO	-	-	-	-	-	-	-	-	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Src	-	-	-	-	-	-	-	-	NMI RST	C5	C4	C3	C2	C1	C0	NMI RST	C5	C4	C3	C2	C1	C0	NMI RST	C5	C4	C3	C2	C1	C0			

The data[1] quadlet selects the signal sources for DIO channels 24 to 47:

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIO	-	-	-	-	-	-	-	-	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24
Src	-	-	-	-	-	-	-	-	NMI RST	C5	C4	C3	C2	C1	C0	NMI RST	C5	C4	C3	C2	C1	C0	NMI RST	C5	C4	C3	C2	C1	C0			

Every DIO channel is associated with one of eight alternate signal sources as shown in the above tables. The tables use the following abbreviations for alternate signal sources:

Symbol	Signal Source
C0	Counter 0 ExtOut
C1	Counter 1 ExtOut
C2	Counter 2 ExtOut
C3	Counter 3 ExtOut
C4	Counter 4 ExtOut
C5	Counter 5 ExtOut
RST	Watchdog RST (Timer2) output
NMI	Watchdog NMI (Timer1) output

Examples:

- When data[1] bit 2 is set, DIO26 will be driven by the ExtOut signal from counter channel 2.
- When data[0] bit 14 is set, DIO14 will be driven by the Watchdog RST output signal.

Each of the eight alternate signal sources is associated with six DIO channels. For example, the watchdog RST signal is associated with DIO channels, 6, 14, 22, 30, 38, and 46. Typically, an alternate source is either not used or it is routed to one of its associated DIO pins, though it may be simultaneously routed to any combination of its associated pins.

This function should only be called when the SWE bit is set (see S826_SafeWrenWrite). The function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1).

8.3.13 S826_DioOutputSourceRead

The S826_DioOutputSourceRead function reads the signal sources assigned to all DIO channels.

```
int S826_DioOutputSourceRead(
    uint board,      // board identifier
    uint data[2]     // pointer to data buffer
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Pointer to a buffer (see Section 8.3) that will receive the signal source assignments for all DIO channels.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

8.3.14 S826_DioFilterWrite

The S826_DioFilterWrite function configures the DIO input noise filters.

```
int S826_DioFilterWrite(
    uint board,      // board identifier
    uint interval,   // filter interval (T)
    uint enables[2]  // filter enable flags
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

interval

Filter time interval in range 1 to 65535, specified as a multiple of 20ns. This is common to all DIO channels.

enables

Pointer to a buffer (see Section 8.3) that specifies filter enables for the DIO channels: '1'=enable, '0'=disable.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

Each DIO has an input noise filter that can be independently enabled or disabled. This function selectively enables and disables the individual filters and programs the filter time interval T, which is common to all DIO filters.

A filter's output will not change state until its input has held a constant state for $T * 20\text{ns}$. The maximum T value is 65535, corresponding to a filter time of 1.3107ms. When a DIO channel's noise filter is enabled, its input signal will be delayed by $T * 20\text{ns}$ and input pulses shorter than $T * 20\text{ns}$ will not be recognized.

8.3.15 S826_DioFilterRead

The S826_DioFilterRead function reads the configuration of the DIO input noise filters.

```
int S826_DioFilterRead(
    uint board,          // board identifier
    uint *interval,     // filter interval (T)
    uint enables[2]      // filter enable flags
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

interval

Pointer to a buffer that will receive the filter time interval as described in S826_DioFilterWrite.

enables

Pointer to a buffer (see Section 8.3) that will receive filter enable flags for the DIO channels: '1'=enable, '0'=disable.

Return Values

If the function succeeds, the return value is zero.

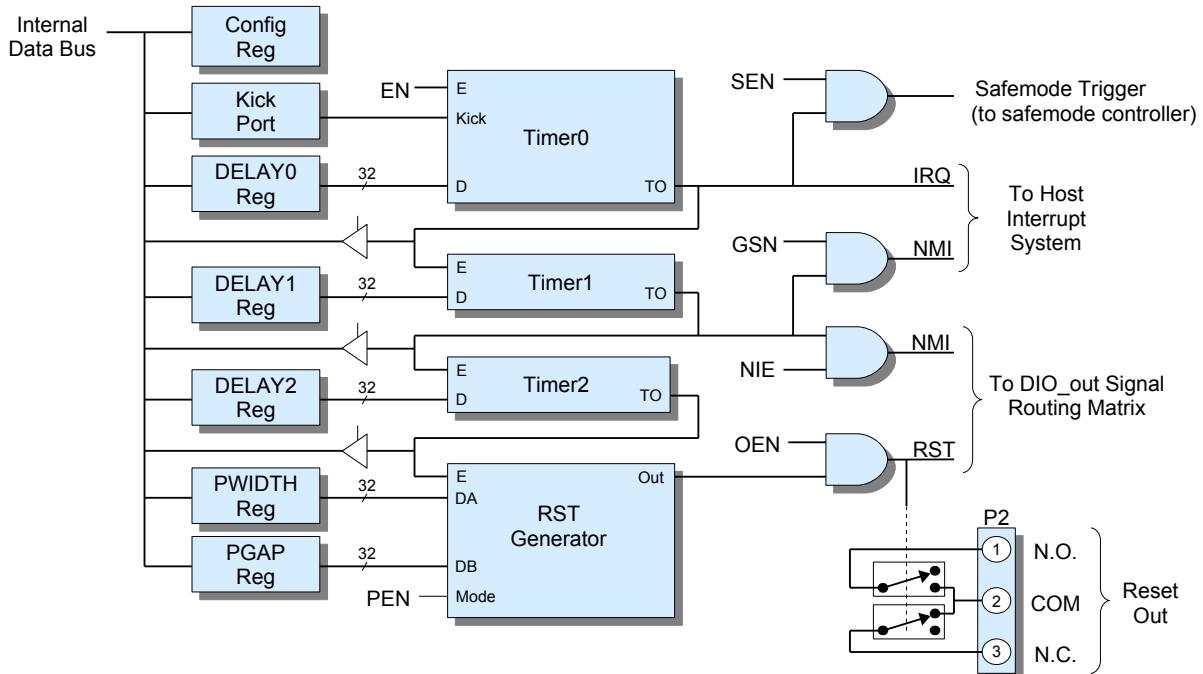
If the function fails, the return value is an error code.

Chapter 9: Watchdog Timer

9.1 Introduction

The model 826 board has a multistage watchdog timer that can activate the board's safemode system and generate service requests. The watchdog has three timer stages that generate timeout events in sequence according to user-defined timing. Each event is associated with an output signal. Typically, the event signals are utilized in such a way that successive events will generate service requests of progressively higher priority.

Figure 8: Watchdog system



9.1.1 Operation

When the watchdog system is disabled (default upon board reset), the three timers are halted and loaded from their associated DELAY registers. When the system becomes enabled ($EN=1$) by calling `S826_WatchdogEnableWrite`, initially only Timer0 is enabled so that it will count down towards zero while the other timers remain idle. During normal operation, the program regularly “kicks” the watchdog by writing to the Kick port, thus reloading Timer0 from DELAY0 before it can count down to zero.

If a fault condition prevents the program from kicking the watchdog, Timer0 will count down to zero and assert its timeout (TO) signal. After this event occurs, all subsequent kicks will be ignored. This event enables Timer1 and generates an interrupt request, and if $SEN=1$, it switches all control outputs to fail-safe states by triggering the safemode system. The program can wait for the interrupt by calling `S826_WatchdogEventWait`.

Timer1 asserts its TO signal upon counting down to zero. This event enables Timer2 and, if $NIE=1$, it asserts the NMI net of the DIO_out signal routing matrix, which in turn may route the net to a DIO pin (see `S826_DioOutputSourceWrite`). When $GSN=1$, a Timer1 event will cause the board to issue a PCI Express fatal error message; this can be used to generate a system non-maskable interrupt request if the system has been appropriately configured. Refer to your system documentation for information about generating NMI in response to a PCI Express fatal error message.

Timer2 asserts its TO signal upon counting down to zero, thus activating the RST signal generator. If $OEN=1$, the RST generator's output is routed to the RST net of the DIO_out signal routing matrix and to the board's Reset Out circuit. The RST generator can produce a continuous (non-pulsed) output or pulsed output. When generating a pulsed output, PWIDTH

determines the pulse duration and PGAP determines the gap time between pulses. The RST signal is not internally connected to the host computer's system reset input; if desired, this must be implemented by externally routing the selected DIO pin to the computer's reset input.

9.1.2 Reset Out Circuit

Two solid state relays (SSRs) are provided for controlling external reset circuits. Both SSRs are energized when the watchdog RST generator is asserting its output signal. One SSR has normally open contacts and the other normally closed contacts. The SSRs are galvanically isolated from other board circuitry.

9.1.3 Initialization

Before enabling the watchdog, the program must initialize it by writing to the configuration register and the five timing control registers (DELAY0-DELAY2, PWIDTH, and PGAP). This is done by calling S826_WatchdogConfigWrite. The DELAY registers determine the time intervals of their three associated timers, whereas the PWIDTH and PGAP registers determine the timing of the RST output signal.

9.2 Connector P2

Connector P2 interfaces external circuitry to the Reset Out solid state relay. Refer to the block diagram in section 9.1 for connector pinout.

9.3 Programming

9.3.1 S826_WatchdogConfigWrite

The S826_WatchdogConfigWrite function configures the watchdog system.

```
int S826_WatchdogConfigWrite(
    uint board,          // board identifier
    uint cfg,           // configuration flags
    uint timing[5]       // time intervals
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

cfg

Configuration flags: '1' = enable feature, '0' = disable feature.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GSN	0	SEN	NIE	PEN	0	OEN

Flag Function

GSN	Generate host system NMI upon Timer1 event.
SEN	Activate safemode upon Timer0 event.
NIE	Connect Timer1 event signal to the DIO_out routing matrix NMI net.
PEN	Enable RST output to pulse (vs. continuous active level).
OEN	Connect RST generator to the DIO_out routing matrix RST net.

timing

Pointer to array of five quadlets that define the watchdog's time intervals. Each quadlet is written to one of the watchdog timing control registers as shown below. All times are specified as multiples of 20 nanoseconds. For example, use the value 50,000,000 for a one-second time interval.

Quadlet	Register	Function
timing[0]	DELAY0	Timer0 interval. The program must kick the watchdog within this interval to prevent a watchdog timeout. This must be set to a non-zero value; set to 1 for shortest possible delay.
timing[1]	DELAY1	Timer1 interval. This specifies the elapsed time from Timer1 timeout to Timer2 timeout. This must be set to a non-zero value; set to 1 for shortest possible delay.
timing[2]	DELAY2	Timer2 interval. This specifies the elapsed time from Timer2 timeout to RST generator enable. This must be set to a non-zero value; set to 1 for shortest possible delay.
timing[3]	PWIDTH	RST pulse width. This is ignored if PEN='0'.
timing[4]	PGAP	Time gap between RST pulses. This is ignored if PEN='0'.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function programs the watchdog configuration register and timing control registers. To ensure reliable operation, it should be called only when the watchdog is disabled.

The function should only be called when the SWE bit is set (see [S826_SafeWrenWrite](#)). The function will fail without notification (return [S826_ERR_OK](#)) if SWE=0 (see Section 10.1.1).

9.3.2 S826_WatchdogConfigRead

The [S826_WatchdogConfigRead](#) function reads the watchdog configuration.

```
int S826_WatchdogConfigRead(
    uint board,          // board identifier
    uint *cfg,           // configuration flags
    uint timing[5]        // time intervals
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

cfg

Pointer to buffer that will receive the watchdog configuration flags.

timing

Pointer to array of five quadlets that will receive the watchdog time intervals.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

See Section 9.3.1 for details about the returned cfg and timing values.

9.3.3 S826_WatchdogEnableWrite

The [S826_WatchdogEnableWrite](#) function enables or disables the watchdog system.

```
int S826_WatchdogEnableWrite(
    uint board,          // board identifier
    uint enable           // enable watchdog when true
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

enable

Set to '1' to enable, or '0' to disable the watchdog. The watchdog is disabled by default upon board reset.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function should only be called when the SWE bit is set (see S826_SafeWrenWrite). The function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1).

9.3.4 S826_WatchdogEnableRead

The S826_WatchdogEnableRead function returns the enable status of the watchdog system.

```
int S826_WatchdogEnableRead(
    uint board,      // board identifier
    uint *enable    // enable status
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

enable

Buffer that will receive the watchdog system enable status: '1' = enabled, '0' = disabled.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

9.3.5 S826_WatchdogStatusRead

The S826_WatchdogStatusRead function reads the watchdog timeout status.

```
int S826_WatchdogStatusRead(
    uint board,      // board identifier
    uint *status     // watchdog status
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

status

Pointer to a quadlet buffer that will receive the watchdog status. Each status bit indicates the timeout status of one watchdog timer stage ('1' = timed out):

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TO2	TO1	TO0			

Flag	Function
TO2	Timer2 timeout
TO1	Timer1 timeout
TO0	Timer0 timeout

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

9.3.6 S826_WatchdogKick

The S826_WatchdogKick function reads the watchdog timeout status.

```
int S826_WatchdogKick(
    uint board,          // board identifier
    uint data            // valid signature
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

data

Valid signature. This must be 0x5A55AA5A to kick the watchdog; any other value will fail to kick the watchdog, although no error will be returned.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

When the watchdog system is running, the program should call this function to prevent a watchdog timeout. The DELAY0 value determines how often this function must be called to prevent a timeout.

9.3.7 S826_WatchdogEventWait

The S826_WatchdogEventWait function waits for a watchdog event (timeout) on Timer1.

```
int S826_WatchdogEventWait(
    uint board,          // board identifier
    uint tmax            // maximum time to wait
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

tmax

Maximum time, in microseconds, to wait for data. See “Event-Driven Applications” for details.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function can operate in either blocking or non-blocking mode. If *tmax* is zero, the function will return immediately. If *tmax* is greater than zero, the calling thread will block until a watchdog event or *tmax* elapses. The function will return zero if a watchdog timeout event occurred, or S826_ERR_NOTREADY if the watchdog has not timed out.

When this function is blocking, it will return immediately with return code S826_ERR_CANCELLED if S826_WatchdogWaitCancel is called by another thread, or with S826_ERR_BOARDCLOSED if S826_SystemClose is called by another thread.

9.3.8 S826_WatchdogWaitCancel

The S826_WatchdogWaitCancel function cancels a blocking wait on watchdog Timer1.

```
int S826_WatchdogWaitCancel(
    uint board      // board identifier
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

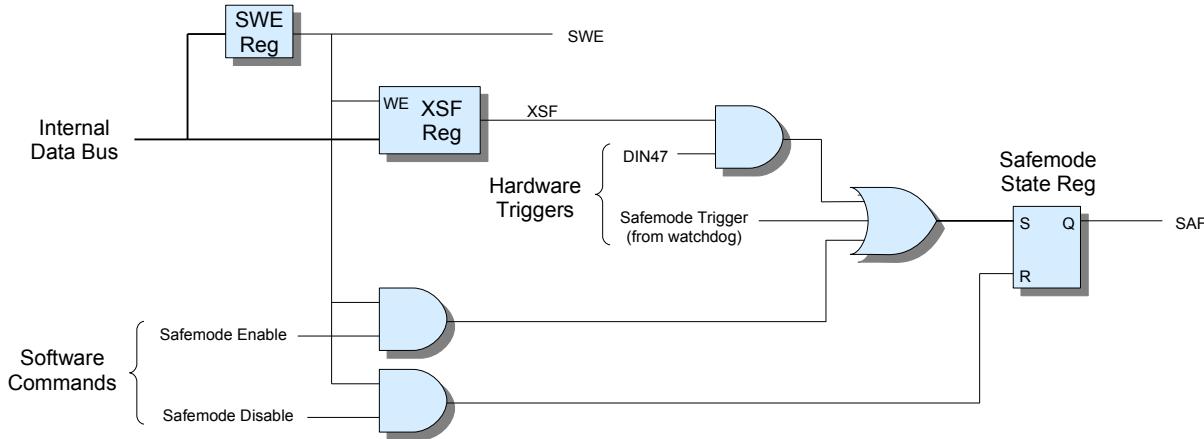
This function cancels blocking on the watchdog so that another thread, which is blocked by S826_WatchdogEventWait while waiting for a watchdog timeout event, will return immediately with S826_ERR_CANCELLED.

Chapter 10: Safemode Controller

10.1 Introduction

The 826 board features a fail-safe controller that forces analog and digital outputs to predetermined levels in response to hardware triggers. The controller works in concert with the watchdog timer and external devices such as emergency shutdown contacts to switch the board's outputs to fail-safe levels without software intervention.

Figure 9: Safemode Controller



The controller consists of configuration (XSF) and write protection control (SWE) registers, triggering logic, and a state register. When safemode is active ($SAF = '1'$), the board's analog and digital outputs are automatically switched to their fail-safe states. SAF can be set by the program and in response to hardware triggers, but only the program can reset SAF to turn off safemode. Upon power-up or board reset, SAF is reset.

If the watchdog is allowed to activate safemode (see `S826_WatchdogConfigWrite`), it will assert the Safemode Trigger signal upon Timer0 event, thus setting SAF . Once asserted, the trigger will remain asserted until the watchdog is disabled. Consequently, the program cannot reset SAF until the watchdog is disabled.

When $XSF = '1'$, safemode can be triggered by an active-low signal applied to the DIO channel 47 connector pin (DIO47). When this happens, the program cannot reset SAF until DIO47 is negated or XSF is cleared.

Additional information about safemode can be found in Section 6.1.1 (analog outputs) and Section 8.1.2 (DIO outputs).

10.1.1 Write Protection

The SWE register controls write protection for registers associated with the watchdog and safemode controller. All affected registers are write-protected when $SWE = '0'$; this is the default state of SWE at power-up and upon system reset. The SWE register state does not change when the board is opened or closed.

Before writing to protected registers, the program must set SWE (by calling `S826_SafeWrenWrite`) to allow writes to the registers. During initialization, the program will typically disable write protection, write all fail-safe states as required by the application, and then re-enable write protection to prevent modification of the registers due to subsequent wayward software execution.

Several of the API functions write to SWE protected registers. These functions can fail without notification if called while $SWE = '0'$ (they will return `S826_ERR_OK` if no other errors are detected, but the protected register will not be written). If it is necessary to detect a failed write to a write-protected register, the program should read the register after writing to it and compare the read and written values; a failed write is indicated when the read and written values are not equal. Each of the write functions has a corresponding read function that can be used to read back the programmed register state; these are not affected by the state of the SWE register.

These API functions write to SWE protected registers:

- S826_DacRangeWrite and S826_DacDataWrite (when safemode argument = '1')
- S826_DioOutputSourceWrite, S826_DioSafeWrite, and S826_DioSafeEnablesWrite
- S826_WatchdogConfigWrite, S826_WatchdogEnableWrite
- S826_SafeControlWrite
- S826_VirtualSafeWrite and S826_VirtualSafeEnablesWrite

10.2 Programming

10.2.1 S826_SafeControlWrite

The S826_SafeControlWrite function programs the board's fail-safe configuration and state.

```
int S826_SafeControlWrite(
    uint board,      // board identifier
    uint settings,  // safemode settings
    uint mode        // write mode
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

settings

Safemode configuration and state bits:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XSF	0	SAF	0	

Bit	Function
XSF	DIO47 trigger enable. Programmed to '0' upon reset. When '1', a low level (0 volts) on the DIO channel 47 header pin will set the SAF bit. When '0', DIO47 will not affect the SAF bit. When XSF=1, a thread can block on DIO47 falling edge events to receive notification when safemode is triggered. Alternatively, the program can poll SAF.
SAF	Safemode state. Programmed to '0' upon reset. '1' = safemode active, '0' = runmode active. This can be written by the application program. This bit can also be set by DIO47 when XSF=1, or by a timeout event on watchdog Timer0.

mode

Write mode: 0 = write, 1 = clear bits, 2 = set bits (see “Atomic Read-Modify-Write”).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

This function should only be called when the SWE bit is set (see S826_SafeWrenWrite). The function will fail without notification (return S826_ERR_OK) if SWE=0 (see Section 10.1.1).

10.2.2 S826_SafeControlRead

The S826_SafeControlRead function returns the board's fail-safe configuration and state.

```
int S826_SafeControlRead(
    uint board,      // board identifier
    uint *settings  // safemode settings
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

settings

Pointer to buffer that will receive the safemode configuration and state as detailed in S826_SafeControlWrite.

mode

Write mode: 0 = write, 1 = clear bits, 2 = set bits (see “Atomic Read-Modify-Write”).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

10.2.3 S826_SafeWrenWrite

The S826_SafeWrenWrite function enables or disables write protection for safemode-related registers.

```
int S826_SafeWrenWrite(
    uint board,      // board identifier
    uint wren        // write enable/disable
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

wren

1 = write protect (default upon board reset), 2 = write enable, other values have no effect. When writes are disabled (write protected), attempts to write to protected registers will without notification (return S826_ERR_OK).

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Remarks

See “Write Protection” for a list of registers that are write protected/enabled by this function.

10.2.4 S826_SafeWrenRead

The S826_SafeWrenRead function returns the board's fail-safe configuration and control settings.

```
int S826_SafeWrenRead(
    uint board,      // board identifier
    uint *wren       // write protection status
);
```

Parameters

board

826 board number. This must match the settings of the board's dip switches as described in section 2.2.

wren

Pointer to buffer that will receive the write protection status: 0 = write protected, 2 = write enabled.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is an error code.

Chapter 11: Specifications

Digital I/O		
Channels	Number/type	48 bi-directional
	Signal levels	5 V TTL/CMOS
	Sampling rate	50 Ms/s
Output	Driver type	Open drain, active-low sink driver
	Internal pull-up impedance	10 kΩ, 5%
	On-state sink current (maximum)	24 mA
	Sink current when board unpowered	< 20 μA
	Fail-safe mode	Yes
Input	Receiver type	Schmitt trigger
	Noise/debounce filter	Interval: 0 to 1.3107 ms in 20 ns steps, common to all channels. Enable/disable: per channel.

Counters		
Channels	Number/type	6 multifunction
	Resolution	32 bits
	Count rate (maximum)	25 MHz (external clock), 50 MHz (internal clock)
	Operating modes	Incremental encoder, event counter, frequency counter, pulse width measure, PWM generator, pulse generator, custom
Clock frequency	Internal	50 MHz, 50 ppm
	External (maximum)	Derate for deviations from 50% duty cycle: 6.25 MHz @ quadrature x4 12.5 MHz @ quadrature/mono x2 25 MHz @ quadrature/mono x1
Inputs (CLK, IX)	Receiver type	RS-422 differential
	Signal levels	Differential: RS-422, ±7 V CMV maximum Single ended: 5 V TTL/CMOS
	Noise/debounce filter	Interval: 0 to 1.3107 ms in 20 ns steps, common per channel. Enable/disable: independent IX, CLK pair.

Analog Inputs		
Channels	Number/type	16 differential
ADC	Resolution	16 bits
	Conversion time	≤ 3 μs
Input	Differential voltage measurement ranges	±1 V, ±2 V, ±5 V, ±10 V
	Absolute input voltage (signal + CMV, max)	±11 V off GND
	CMRR	> 80 dB @ 1 kHz, > 65 dB @ 10 kHz
	Input impedance	>10 MΩ in parallel with 100 pF
	Settling time for multichannel measurements	4μs max. @ < 1 kΩ input Z with no gain change
Triggering	Modes	Hardware: 48 external digital inputs, 6 internal counter outputs. Software: 6 virtual digital outputs. Untriggered (free-running).

Analog Outputs		
Channels	Number/type	8 single ended, with local (on-board) sense
DAC	Resolution	16 bits
	Conversion time (serializer transmission + analog conversion)	1.04 μs
Output	Voltage ranges	0 to +5 V, 0 to +10 V, ±5 V, ±10 V
	Load current (maximum)	2 mA
	Fail-safe mode	Yes

Watchdog Timer		
Timer stages	Number	3
	Interval (per stage)	Programmable from 1 to $2^{32}-1 * 20$ ns (20 ns to ~85.9 s)
	Output events	Stage 0: Fail-safe trigger, IRQ Stage 1: NMI out via digital output, PCIe Fatal Error Stage 2: Reset via digital output or onboard solid state relay (SSR)
Solid state relay (Reset out)	On-state resistance ($I_L = 10$ mA)	20 ohms typical, 30Ω max.
	Off-state leakage current	0.03 μ A typical, 1.0 μ A max.
	Applied voltage (maximum)	200 V
	Load current (maximum)	100 mA

Power and Environmental		
Power	Input power	350 mA (+12V) and 450 mA (+3.3V), nominal, with no loads
	Encoder power out	5 VDC $\pm 5\%$, 400 mA max. (total for all encoders)
Temperature	Operating	0 to 70°C
Mating Connectors (not included)	Counters	Sullins SFH210-PPPC-D13-ID-BK or equivalent (qty. 2)
	Analog I/O	Sullins SFH210-PPPC-D25-ID-BK or equivalent (qty. 1)
	Digital I/O	Sullins SFH210-PPPC-D25-ID-BK or equivalent (qty. 2)

Appendix C: Linux CNC

Getting Started V2.6.8-5-g8676c92, 2015-05-14

Contents

I Getting Started	1
1 System Requirements	2
1.1 Minimum Requirements	2
1.2 Problematic Hardware	2
1.2.1 Laptops	2
1.2.2 Video Cards	2
2 About LinuxCNC	3
2.1 The Software	3
2.2 The Operating System	3
2.3 Getting Help	4
2.3.1 IRC	4
2.3.2 Mailing List	4
2.3.3 LinuxCNC Wiki	4
2.4 Getting LinuxCNC	5
2.4.1 Normal Download	5
2.4.2 Multi-session Download	5
2.4.3 Burning the CD	6
2.4.4 Testing LinuxCNC	6
2.4.5 Installing LinuxCNC	7
2.4.6 Updates to LinuxCNC	7
2.4.7 Install Problems	7
3 Updating LinuxCNC	8
3.1 Updating from 2.5.x to 2.6.x	8
3.2 Updating from 2.4.x to 2.5.x	9
3.2.1 On Ubuntu Lucid 10.04	10
3.2.2 On Ubuntu Hardy 8.04	10
3.3 Config changes	11

3.4	Upgrading from 2.3.x to 2.4.x	11
3.5	Changes between 2.3.x and 2.4.x	12
3.5.1	emc.nml changes (2.3.x to 2.4.x)	12
3.5.2	tool table changes (2.3.x to 2.4.x)	12
3.5.3	hostmot2 firmware images (2.3.x to 2.4.x)	12
4	Stepper Quickstart	13
4.1	Latency Test	13
4.2	Sherline	13
4.3	Xylotex	13
4.4	Machine Information	13
4.5	Pinout Information	14
4.6	Mechanical Information	14
5	Stepper Configuration Wizard	16
5.1	Entry Page	17
5.2	Basic Information	18
5.3	Latency Test	19
5.4	Parallel Port Setup	21
5.5	Axis Configuration	22
5.5.1	Test This Axis	23
5.5.1.1	Finding Maximum Velocity	24
5.5.1.2	Finding Maximum Acceleration	24
5.6	Spindle Configuration	25
5.6.1	Spindle Speed Control	25
5.6.2	Spindle-synchronized motion	26
5.6.3	Determining Spindle Calibration	26
5.7	Advanced Configuration Options	27
5.8	Machine Configuration Complete	27
5.9	Axis Travel, Home Location, and Home Switch Location	28
5.9.1	Operating without Limit Switches	28
5.9.2	Operating without Home Switches	28
5.9.3	Home and Limit Switch wiring options	28
6	Mesa Configuration Wizard	30
6.1	Create or Edit	31
6.2	Basic Machine Information	32
6.3	External Configuration	34
6.4	GUI Configuration	36
6.5	Mesa Configuration	39

6.6	Mesa I/O Setup	40
6.7	Parport configuration	44
6.8	Axis Configuration	45
6.9	Spindle Configuration	52
6.10	Advanced Options	54
6.11	HAL Components	55
6.12	Advanced Usage Of PNCconf	56
7	Running LinuxCNC	58
7.1	Invoking LinuxCNC	58
7.2	Configuration Selector	58
7.3	Next steps in configuration	59
8	Linux FAQ	60
8.1	Automatic Login	60
8.2	Automatic Startup	60
8.3	Man Pages	60
8.4	List Modules	61
8.5	Editing a Root File	61
8.5.1	The Command Line Way	61
8.5.2	The GUI Way	61
8.5.3	Root Access	61
8.6	Terminal Commands	61
8.6.1	Working Directory	61
8.6.2	Changing Directories	62
8.6.3	Listing files in a directory	62
8.6.4	Finding a File	62
8.6.5	Searching for Text	62
8.6.6	Bootup Messages	63
8.7	Convenience Items	63
8.7.1	Terminal Launcher	63
8.8	Hardware Problems	63
8.8.1	Hardware Info	63
8.8.2	Monitor Resolution	63
8.9	Paths	63
9	Legal Section	64
9.1	Copyright Terms	64
9.2	GNU Free Documentation License	64
10	Index	68

The LinuxCNC Team



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.

Part I

Getting Started

Chapter 1

System Requirements

1.1 Minimum Requirements

The minimum system to run LinuxCNC and Ubuntu may vary depending on the exact usage. Stepper systems in general require faster threads to generate step pulses than servo systems. Using the Live-CD you can test the software before committing a computer. Keep in mind that the Latency Test numbers are more important than the processor speed for software step generation. More information on the Latency Test is [here](#).

Additional information is on the LinuxCNC Wiki site:

[Wiki.LinuxCNC.org, Hardware_Requirements](#)

LinuxCNC and Ubuntu should run reasonably well on a computer with the following minimum hardware specification. These numbers are not the absolute minimum but will give reasonable performance for most stepper systems.

- 700 MHz x86 processor (1.2 GHz x86 processor recommended)
- 384 MB of RAM (512 MB up to 1 GB recommended)
- 8 GB hard disk
- Graphics card capable of at least 1024x768 resolution, which is not using the NVidia or ATI fglrx proprietary drivers, and which is not an onboard video chipset that shares main memory with the CPU
- A network or Internet connection (not strictly needed, but very useful for updates and for communicating with the LinuxCNC community)

Minimum hardware requirements change as Ubuntu evolves so check the [Ubuntu](#) web site for details on the LiveCD you're using. Older hardware may benefit from selecting an older version of the LiveCD when available.

1.2 Problematic Hardware

1.2.1 Laptops

Laptops are not generally suited to real time software step generation. Again a Latency Test run for an extended time will give you the info you need to determine suitability.

1.2.2 Video Cards

If your installation pops up with 800 x 600 screen resolution then most likely Ubuntu does not recognize your video card or monitor. Onboard video many times causes bad real time performance.

Chapter 2

About LinuxCNC

2.1 The Software

- LinuxCNC (the Enhanced Machine Control) is a software system for computer control of machine tools such as milling machines and lathes, robots such as puma and scara and other computer controlled machines up to 9 axes.
- LinuxCNC is free software with open source code. Current versions of LinuxCNC are entirely licensed under the GNU General Public License and Lesser GNU General Public License (GPL and LGPL)
- LinuxCNC provides:
 - a graphical user interface (actually several interfaces to choose from)
 - an interpreter for *G-code* (the RS-274 machine tool programming language)
 - a realtime motion planning system with look-ahead
 - operation of low-level machine electronics such as sensors and motor drives
 - an easy to use *breadboard* layer for quickly creating a unique configuration for your machine
 - a software PLC programmable with ladder diagrams
 - easy installation with a Live-CD
- It does not provide drawing (CAD - Computer Aided Design) or G-code generation from the drawing (CAM - Computer Automated Manufacturing) functions.
- It can simultaneously move up to 9 axes and supports a variety of interfaces.
- The control can operate true servos (analog or PWM) with the feedback loop closed by the LinuxCNC software at the computer, or open loop with step-servos or stepper motors.
- Motion control features include: cutter radius and length compensation, path deviation limited to a specified tolerance, lathe threading, synchronized axis motion, adaptive feedrate, operator feed override, and constant velocity control.
- Support for non-Cartesian motion systems is provided via custom kinematics modules. Available architectures include hexapods (Stewart platforms and similar concepts) and systems with rotary joints to provide motion such as PUMA or SCARA robots.
- LinuxCNC runs on Linux using real time extensions.

2.2 The Operating System

Ubuntu has been chosen because it fits perfectly into the Open Source views of LinuxCNC:

- Ubuntu will always be free of charge, and there is no extra fee for the *enterprise edition*, we make our very best work available to everyone on the same Free terms.
- LinuxCNC is paired with the LTS versions of Ubuntu which provide support and security fixes from the Ubuntu team for 3 - 5 years.
- Ubuntu uses the very best in translations and accessibility infrastructure that the Free Software community has to offer, to make Ubuntu usable for as many people as possible.
- The Ubuntu community is entirely committed to the principles of free software development; we encourage people to use open source software, improve it and pass it on.

2.3 Getting Help

2.3.1 IRC

IRC stands for Internet Relay Chat. It is a live connection to other LinuxCNC users. The LinuxCNC IRC channel is #linuxcnc on freenode.

The simplest way to get on the IRC is to use the embedded java client on this [page](#).

Some IRC etiquette

- Ask specific questions... Avoid questions like "Can someone help me?".
- If you're really new to all this, think a bit about your question before typing it. Make sure you give enough information so someone can solve your question.
- Have some patience when waiting for an answer, sometimes it takes a while to formulate an answer or everyone might be busy working or something.
- Set up your IRC account with your unique name so people will know who you are. If you use the java client, use the same name every time you log in. This helps people remember who you are and if you have been on before many will remember the past discussions which saves time on both ends.

Sharing Files

The most common way to share files on the IRC is to upload the file to one of the following or a similar service and paste the link:

- For text - <http://pastebin.com/>, <http://pastie.org/>, <https://gist.github.com/>
- For pictures - <http://imagebin.org/>, <http://imgur.com/>, <http://bayimg.com/>
- For files - <https://filedropper.com/>, <http://filefactory.com/>, <http://1fichier.com/>

2.3.2 Mailing List

An Internet Mailing List is a way to put questions out for everyone on that list to see and answer at their convenience. You get better exposure to your questions on a mailing list than on the IRC but answers take longer. In a nutshell you e-mail a message to the list and either get daily digests or individual replies back depending on how you set up your account.

The [emc-users mailing list](#)

2.3.3 LinuxCNC Wiki

A Wiki site is a user maintained web site that anyone can add to or edit.

The user maintained LinuxCNC Wiki site contains a wealth of information and tips at:

<http://wiki.linuxcnc.org>

2.4 Getting LinuxCNC

2.4.1 Normal Download

Download the Live CD from:

[the LinuxCNC homepage at www.linuxcnc.org](http://www.linuxcnc.org)

and follow the Download link.

2.4.2 Multi-session Download

If the file is too large to download in one session because of a bad or slow Internet connection, use `wget` to allow resuming after an interrupted download.

Wget Linux

Open a terminal window. In Ubuntu it is Applications/Accessories/Terminal. Use `cd` to change to the directory where you would like to store the ISO. Use `mkdir` to create a new directory if needed.

Note that actual file names may change so you might have to go to <http://www.linuxcnc.org/> and follow the Download link to get the actual file name. In most browsers you can right click on the link and select Copy Link Location or similar, then paste the link into the terminal window with a right mouse click and select Paste.

Debian Wheezy and LinuxCNC fresh install

The Debian image is a "hybrid" iso, which means you can use the same iso file for a USB stick or a DVD.

To get the Debian Wheezy LinuxCNC Live CD using wget copy one of this in a terminal window and press enter:

[wget http://linuxcnc.org/binary.hybrid.iso](http://linuxcnc.org/binary.hybrid.iso)

The md5sum of the above file is: *b515c872335336ccfc96471d66b687d8*

To continue a partial download that was interrupted add the -c option to wget:

`wget -c http://linuxcnc.org/binary.hybrid.iso`

To stop a download use Ctrl-C or close the terminal window.

Ubuntu 8.04 Hardy Heron and LinuxCNC (older)

For more information on other versions of LinuxCNC visit the linuxcnc.org download page.

<http://linuxcnc.org/index.php/english/download>

After the download is complete you will find the ISO file in the directory that you selected. Next we will burn the CD.

Wget Windows

The wget program is also available for Windows from:

<http://gnuwin32.sourceforge.net/packages/wget.htm>

Follow the instructions on the web page for downloading and installing the windows version of the wget program.

To run wget open a command prompt window.

In most Windows it is Programs/Accessories/Command Prompt

First you have to change to the directory where wget is installed in.

Typically it is in C:\Program Files\GnuWin32\bin so in the Command Prompt window type:

```
cd C:\Program Files\GnuWin32\bin
```

and the prompt should change to: *C:\Program Files\GnuWin32\bin>*

Type the wget command into the window and press enter as above.

2.4.3 Burning the CD

LinuxCNC is distributed as CD image files, called ISOs. To install LinuxCNC, you first need to burn the ISO file onto a CD. You need a working CD/DVD burner and an 80 minute (700 Mb) CD for this. If the CD writing fails, try writing at a slower burn speed.

Verify md5sum in Linux

Before burning a CD, it is highly recommended that you verify the md5sum (hash) of the .iso file.

Open a terminal window. In Ubuntu it is Applications/Accessories/Terminal.

Change to the directory where the ISO was downloaded to.

```
cd download_directory
```

Then run the md5sum command with the file name you saved.

```
md5sum -b binary.hybrid.iso
```

The md5sum should print out a single line after calculating the hash. On slower computers this might take a minute or two.

```
b515c872335336ccfc96471d66b687d8 *binary.hybrid.iso
```

Now compare it to the md5sum value that it should be.

Burning the ISO in Linux

1. Insert a blank CD into your burner. A *CD/DVD Creator* or *Choose Disc Type* window will pop up. Close this, as we will not be using it.
2. Browse to the downloaded ISO image in the file browser.
3. Right click on the ISO image file and choose Write to Disc.
4. Select the write speed. If you are burning a Ubuntu Live CD, it is recommended that you write at the lowest possible speed.
5. Start the burning process.
6. If a *choose a file name for the disc image* window pops up, just pick OK.

Verify md5sum with Windows

Before burning a CD, it is highly recommended that you verify the md5 sum (hash) of the .iso file, to ensure that you got a good download.

Windows does not come with a md5sum program. You will have to download and install one to check the md5sum. More information can be found at:

<https://help.ubuntu.com/community/HowToMD5SUM>

Burning the ISO in Windows

1. Download and install Infra Recorder, a free and open source image burning program: <http://infrarecorder.org/>
2. Insert a blank CD in the drive and select Do nothing or Cancel if an auto-run dialog pops up.
3. Open Infra Recorder, and select the *Actions* menu, then *Burn image*.

2.4.4 Testing LinuxCNC

With the Live CD in the CD/DVD drive shut down the computer then turn the computer back on. This will boot the computer from the Live CD. Once the computer has booted up you can try out LinuxCNC without installing it. You can not create custom configurations or modify most system settings like screen resolution unless you install LinuxCNC.

To try out LinuxCNC from the Applications/CNC menu pick LinuxCNC. Then select a sim configuration to try out.

To see if your computer is suitable for software step pulse generation run the Latency Test as shown [here](#).

2.4.5 Installing LinuxCNC

To install LinuxCNC from the LiveCD select 'Install (Graphical) at bootup.

2.4.6 Updates to LinuxCNC

With the normal install the Update Manager will notify you of updates to LinuxCNC when you go on line and allow you to easily upgrade with no Linux knowledge needed. If you want to upgrade to Debian Wheezy from 10.04 or 8.04 a clean install from the Live-CD is recommended. It is OK to upgrade everything except the operating system when asked to.

Warning: Do not upgrade the operating system if prompted to do so.

2.4.7 Install Problems

In rare cases you might have to reset the BIOS to default settings if during the Live CD install it cannot recognize the hard drive during the boot up.

Chapter 3

Updating LinuxCNC

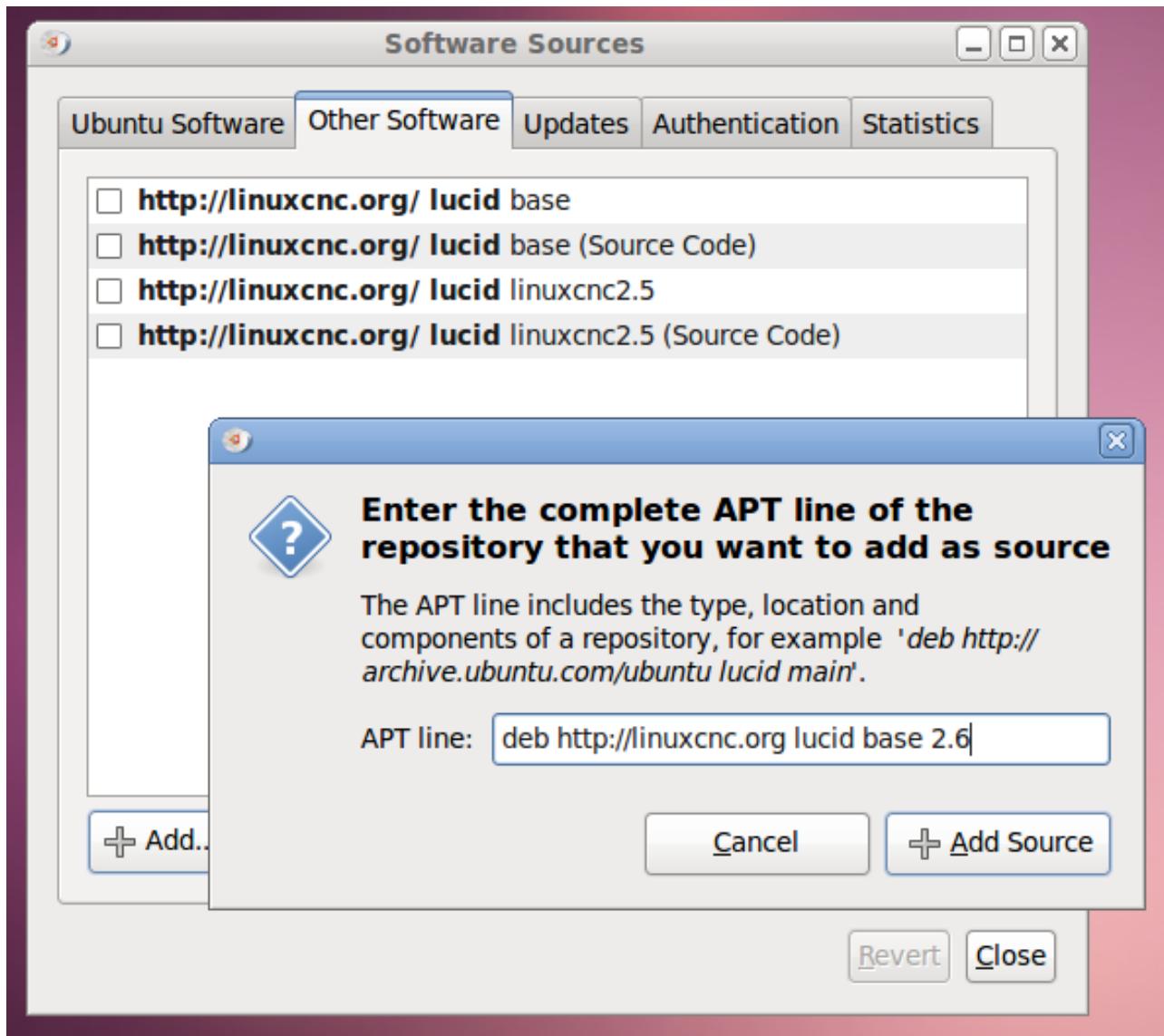
3.1 Updating from 2.5.x to 2.6.x

First you need to tell your computer where to find the new LinuxCNC software:

Click on the System menu in the top panel and select Administration->Software Sources.
Select the Other Software tab.
Disable or delete all the old linuxcnc.org entries.

Add a new Apt line that looks like this:

```
deb http://linuxcnc.org/ lucid base 2.6
```



When you add that line, Software Sources will pop up a window informing you that the information about available software is out-of-date. Click the Reload button.

Now your computer knows about the new software, next we need to tell it to install it:

- Click on the System menu in the top panel and select Administration->Synaptic Package Manager
- In the Quick Search bar at the top, type linuxcnc.
- Click the check box to mark the new linuxcnc package for installation.
- Click the Apply button, and let your computer install the new package. The old linuxcnc 2.5 package will be automatically upgraded to the new linuxcnc 2.6 package.

3.2 Updating from 2.4.x to 2.5.x

As of version 2.5.0, the name of the project has changed from EMC2 to LinuxCNC. All programs with "emc" in the name have been changed to "linuxcnc" instead. All documentation has been updated.

Additionally, the name of the debian package containing the software has changed. Unfortunately this breaks automatic upgrades. To upgrade from emc2 2.4.X to linuxcnc 2.5.X, do the following:

3.2.1 On Ubuntu Lucid 10.04

First you need to tell your computer where to find the new LinuxCNC software:

- Click on the System menu in the top panel and select Administration->Software Sources.
- Select the Other Software tab.
- Select the entry that says

```
http://linuxcnc.org/lucid lucid base emc2.4
```

or

```
http://linuxcnc.org/lucid lucid base emc2.4-sim
```

and click the Edit button.

- In the Components field, change emc2.4 to linuxcnc2.5, or change emc2.4-sim to linuxcnc2.5-sim.
- Click the OK button.
- Back in the Software Sources window, Other Software tab, click the Close button.
- It will pop up a window informing you that the information about available software is out-of-date. Click the Reload button.

Now your computer knows about the new software, next we need to tell it to install it:

- Click on the System menu in the top panel and select Administration->Synaptic Package Manager
- In the Quick Search bar at the top, type linuxcnc.
- Click the check box to mark the new linuxcnc package for installation.
- Click the Apply button, and let your computer install the new package. The old emc 2.4 package will be automatically removed to make room for the new LinuxCNC 2.5 package.

3.2.2 On Ubuntu Hardy 8.04

First you need to tell your computer where to find the new LinuxCNC software:

- Click on the System menu in the top panel and select Administration->Synaptic Package Manager
- Go to Settings->Repositories.
- Select the "Third-Party Software" tab.
- Select the entry that says

```
http://linuxcnc.org/hardy hardy emc2.4
```

or

```
http://linuxcnc.org/hardy hardy emc2.4-sim
```

and click the Edit button.

- In the Components field, change emc2.4 to linuxcnc2.5 or emc2.4-sim to linuxcnc2.5-sim.
- Click the OK button.
- Back in the Software Sources window, click the Close button.
- Back in the Synaptic Package Manager window, click the Reload button.

Now your computer knows about the new software, next we need to tell it to install it:

- In the Synaptic Package Manager, click the Search button.
- In the Find dialog window that pops up, type linuxcnc and click the Search button.
- Click the check-box to mark the linuxcnc package for installation.
- Click the Apply button, and let your computer install the new package. The old emc 2.4 package will be automatically removed to make room for the new LinuxCNC 2.5 package.

3.3 Config changes

The user configs moved from \$HOME/emc2 to \$HOME/linuxcnc, so you will need to rename your directory, or move your files to the new place.

The hostmot2 watchdog in LinuxCNC 2.5 does not start running until the HAL threads start running. This means it now tolerates a timeout on the order of the servo thread period, instead of requiring a timeout that's on the order of the time between loading the driver and starting the HAL threads. This typically means a few milliseconds (a few times the servo thread period) instead of many hundreds of milliseconds. The default has been lowered from 1 second to 5 milliseconds. You generally don't need to set the hm2 watchdog timeout any more, unless you've changed your servo thread period.

The old driver for the Mesa 5i20, hal_m5i20, has been removed after being deprecated in favor of hostmot2 since early 2009 (version 2.3.) If you are still using this driver, you will need to build a new configuration using the hostmot2 driver. Pncconf may help you do this, and we have some sample configs (hm2-servo and hm2-stepper) that act as examples.

3.4 Upgrading from 2.3.x to 2.4.x

The following instructions only apply to Ubuntu 8.04 "Hardy Heron". LinuxCNC 2.4 is not available for older releases of Ubuntu. Because there are several minor incompatibilities between 2.3.5 and 2.4.x, your existing install will not automatically be updated to 2.4.x. If you want to run 2.4.x, change to the LinuxCNC-2.4 repository by following these instructions:

run System/Administration/Synaptic Package Manager

go to Settings/Repositories

In the list of Third-Party software there should be at least two lines for linuxcnc.org.

For each of them:

- Select the line and click Edit
- On the Components line, change emc2.3 to emc2.4
- Click OK
- Close the *Software Preferences* window

- Click *Reload* as instructed
- Click *Mark All Upgrades*

Mesa card and hostmot2 users:

If you use a mesa card, find the proper hostmot2-firmware package for your card and mark it for installation. Hint: do a search for *hostmot2-firmware* in the synaptic package manager.

- Click *Apply*

3.5 Changes between 2.3.x and 2.4.x

Once you have done the upgrade, update any custom configurations by following these instructions:

3.5.1 emc.nml changes (2.3.x to 2.4.x)

For configurations that have not customized emc.nml, remove or comment out the infile line NML_FILE = emc.nml. This will cause the most up to date version of emc.nml to be used.

For configurations that have customized emc.nml, a change similar to this one is required.

Failure to do this can cause an error like this one:

```
libnml/buffer/physmem.cc 143: PHYSMEM_HANDLE:  
Can't write 10748 bytes at offset 60 from buffer of size 10208.
```

3.5.2 tool table changes (2.3.x to 2.4.x)

The format of the tool table has been changed incompatibly. The documentation shows the new format. The tool table will automatically be converted to the new format.

3.5.3 hostmot2 firmware images (2.3.x to 2.4.x)

The hostmot2 firmware images are now a separate package. You can:

- Continue using an already-installed *emc2-firmware-mesa-** 2.3.x package
- Install the new packages from the synaptic package manager. The new packages are named *hostmot2-firmware-**
- Download the firmware images as tar files from <http://emergent.unpy.net/01267622561> and install them manually

Chapter 4

Stepper Quickstart

This section assumes you have done a standard install from the Live CD. After installation it is recommended that you connect the computer to the Internet and wait for the update manager to pop up and get the latest updates for LinuxCNC and Ubuntu before continuing. For more complex installations see the Integrator Manual.

4.1 Latency Test

The Latency Test determines how late your computer processor is in responding to a request. Some hardware can interrupt the processing which could cause missed steps when running a CNC machine. This is the first thing you need to do. Follow the instructions [here](#) to run the latency test.

4.2 Sherline

If you have a Sherline several predefined configurations are provided. This is on the main menu CNC/EMC then pick the Sherline configuration that matches yours and save a copy.

4.3 Xylotex

If you have a Xylotex you can skip the following sections and go straight to the [Stepper Config Wizard](#). LinuxCNC has provided quick setup for the Xylotex machines.

4.4 Machine Information

Gather the information about each axis of your machine.

Drive timing is in nano seconds. If you're unsure about the timing many popular drives are included in the stepper configuration wizard. Note some newer Gecko drives have different timing than the original one. A [list](#) is also on the user maintained LinuxCNC wiki site of more drives.

Axis	Drive Type	Step Time ns	Step Space ns	Dir. Hold ns	Dir. Setup ns
X					
Y					
Z					

4.5 Pinout Information

Gather the information about the connections from your machine to the PC parallel port.

Output Pin	Typ. Function	If Different	Input Pin	Typ. Function	If Different
1	E-Stop Out		10	X Limit/Home	
2	X Step		11	Y Limit/Home	
3	X Direction		12	Z Limit/Home	
4	Y Step		13	A Limit/Home	
5	Y Direction		15	Probe In	
6	Z Step				
7	Z Direction				
8	A Step				
9	A Direction				
14	Spindle CW				
16	Spindle PWM				
17	Amplifier Enable				

Note any pins not used should be set to Unused in the drop down box. These can always be changed later by running Stepconf again.

4.6 Mechanical Information

Gather information on steps and gearing. The result of this is steps per user unit which is used for SCALE in the .ini file.

Axis	Steps/Rev.	Micro Steps	Motor Teeth	Leadscrew Teeth	Leadscrew Pitch
X					
Y					
Z					

- Steps per revolution* - is how many stepper-motor-steps it takes to turn the stepper motor one revolution. Typical is 200.
- Micro Steps* - is how many steps the drive needs to move the stepper motor one full step. If microstepping is not used, this number will be 1. If microstepping is used the value will depend on the stepper drive hardware.
- Motor Teeth and Leadscrew Teeth* - is if you have some reduction (gears, chain, timing belt, etc.) between the motor and the leadscrew. If not, then set these both to 1.
- Leadscrew Pitch* - is how much movement occurs (in user units) in one leadscrew turn. If you're setting up in inches then it is inches per turn. If you're setting up in millimeters then it is millimeters per turn.

The net result you're looking for is how many CNC-output-steps it takes to move one user unit (inches or mm).

Example 4.1 Units inches

Stepper	= 200 steps per revolution
Drive	= 10 micro steps per step
Motor Teeth	= 20
Leadscrew Teeth	= 40
Leadscrew Pitch	= 0.2000 inches per turn

From the above information, the leadscrew moves 0.200 inches per turn. - The motor turns 2.000 times per 1 leadscrew turn. - The drive takes 10 microstep inputs to make the stepper step once. - The drive needs 2000 steps to turn the stepper one revolution. So the scale needed is:

$$\frac{200\text{motor steps}}{1\text{motor rev}} \times \frac{10\text{microsteps}}{1\text{motor step}} \times \frac{2\text{motor revs}}{1\text{leadscrew rev}} \times \frac{1\text{leadscrew revs}}{0.2000\text{inch}} = \frac{20,000\text{microsteps}}{\text{inch}}$$

Example 4.2 Units mm

Stepper = 200 steps per revolution
Drive = 8 micro steps per step
Motor Teeth = 30
Leadscrew Teeth = 90
Leadscrew Pitch = 5.00 mm per turn

From the above information: - The leadscrew moves 5.00 mm per turn. - The motor turns 3.000 times per 1 leadscrew turn. - The drive takes 8 microstep inputs to make the stepper step once. - The drive needs 1600 steps to turn the stepper one revolution. So the scale needed is:

$$\frac{200 \text{ full steps}}{1 \text{ rev}} \times \frac{8 \text{ microsteps}}{1 \text{ step}} \times \frac{3 \text{ revs}}{1 \text{ leadscrew rev}} \times \frac{1 \text{ leadscrew rev}}{5.00 \text{ mm}} = \frac{960 \text{ steps}}{1 \text{ mm}}$$

Chapter 5

Stepper Configuration Wizard

LinuxCNC is capable of controlling a wide range of machinery using many different hardware interfaces.

Stepconf is a program that generates configuration files for LinuxCNC for a specific class of CNC machine: those that are controlled via a *standard parallel port*, and controlled by signals of type *step & direction*.

Stepconf is installed when you install LinuxCNC and is in the CNC menu.

Stepconf places a file in the linuxcnc/config directory to store the choices for each configuration you create. When you change something, you need to pick the file that matches your configuration name. The file extension is .stepconf.

The Stepconf Wizard needs at least 800 x 600 screen resolution to see the buttons on the bottom of the pages.

Step by Step Instructions

5.1 Entry Page

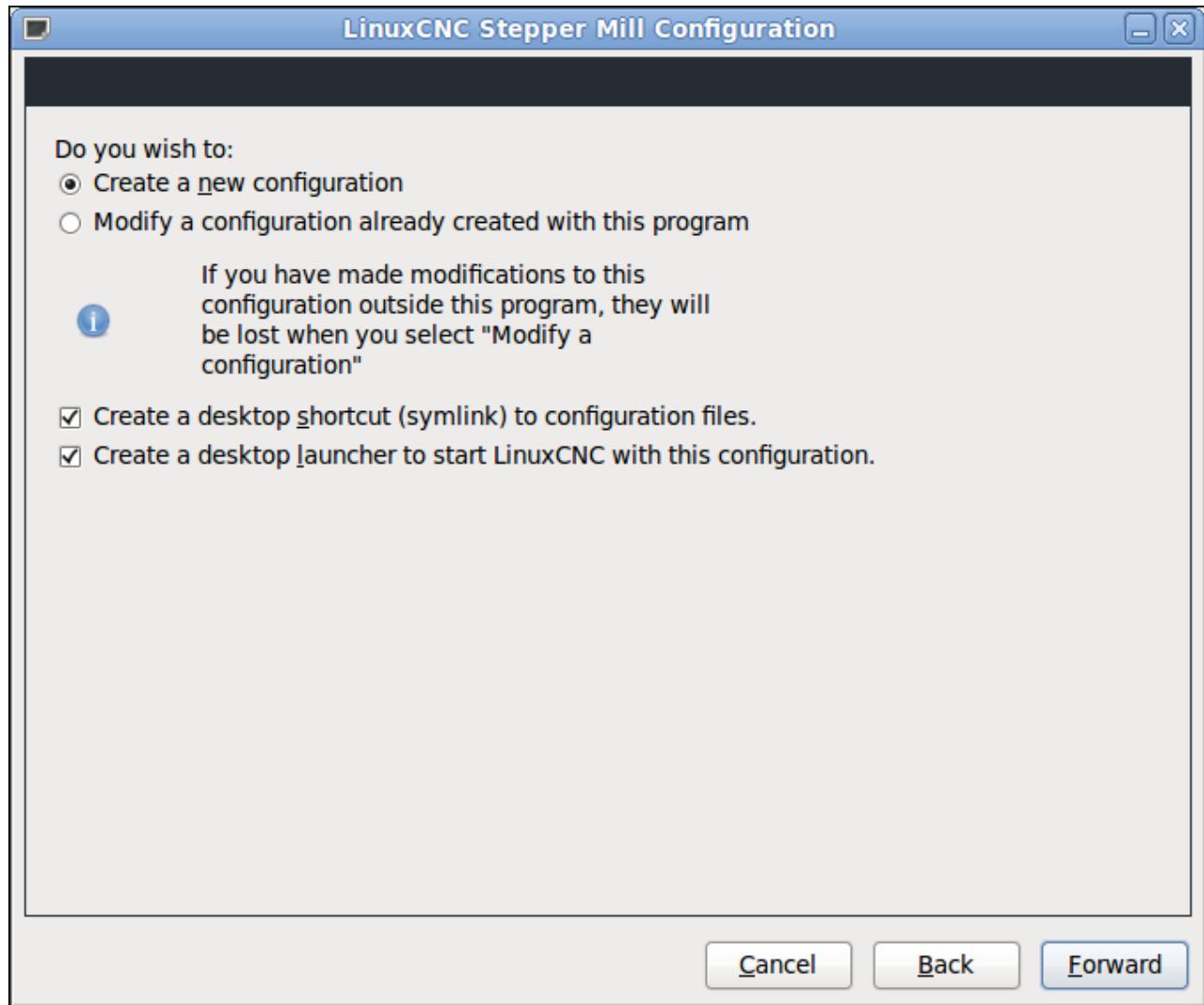


Figure 5.1: Entry Page

- *Create New* - Creates a fresh configuration.
- *Modify* - Modify an existing configuration. After selecting this a file picker pops up so you can select the .stepconf file for modification. If you made any modifications to the main .hal or the .ini file these will be lost. Modifications to custom.hal and custom_postgui.hal will not be changed by the Stepconf Wizard.
- *Create Desktop Shortcut* - This will place a link on your desktop to the files.
- *Create Desktop Launcher* - This will place a launcher on your desktop to start your application.

5.2 Basic Information

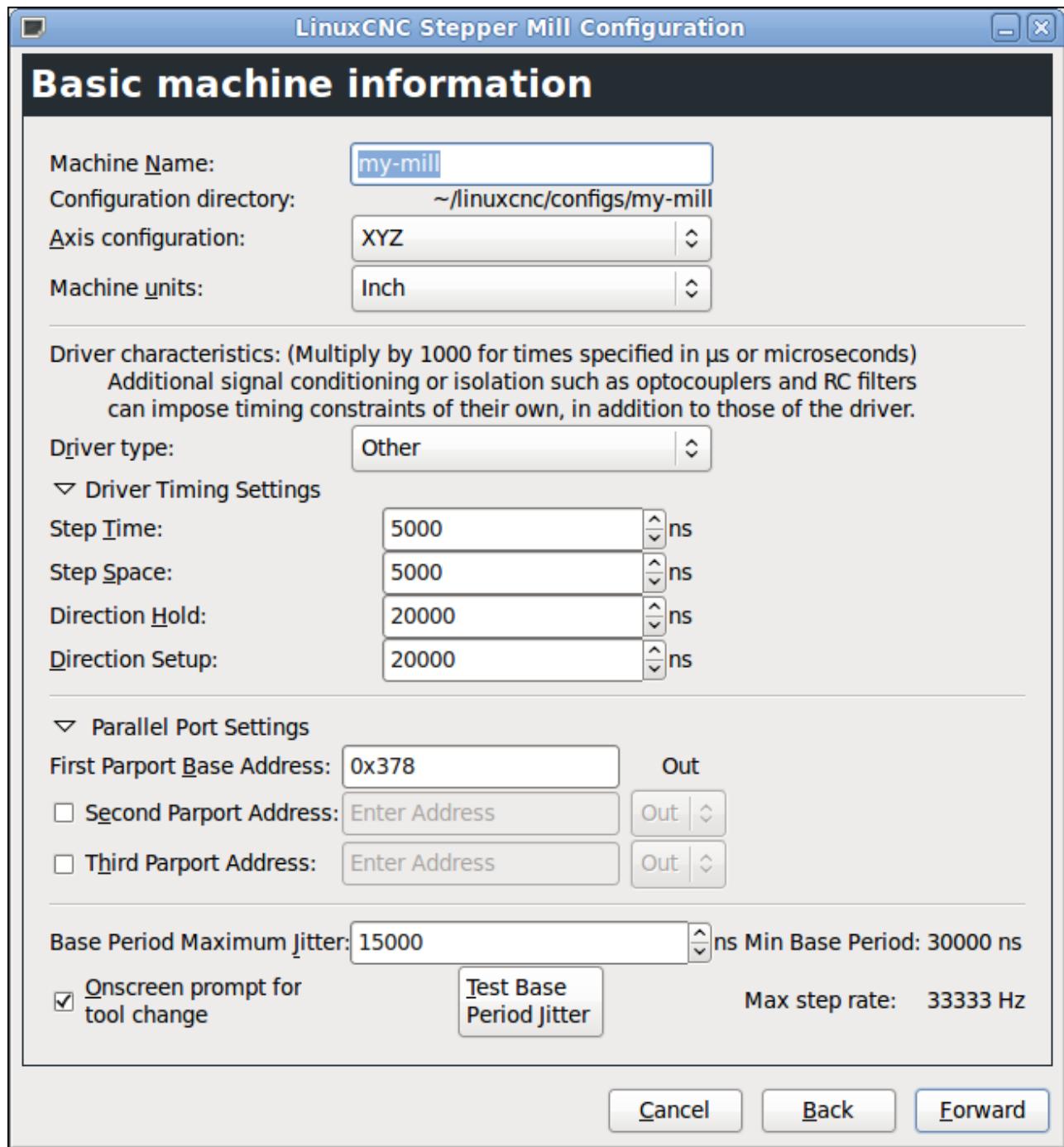


Figure 5.2: Basic Information Page

- *Machine Name* - Choose a name for your machine. Use only uppercase letters, lowercase letters, digits, - and _.
- *Axis Configuration* - Choose XYZ (Mill), XYZA (4-axis mill) or XZ (Lathe).

- *Machine Units* - Choose Inch or mm. All subsequent entries will be in the chosen units
- *Driver Type* - If you have one of the stepper drivers listed in the pull down box, choose it. Otherwise, select *Other* and find the timing values in your driver's data sheet and enter them as *nano seconds* in the *Driver Timing Settings*. If the data sheet gives a value in microseconds, multiply by 1000. For example, enter 4.5us as 4500ns.

A list of some popular drives, along with their timing values, is on the LinuxCNC.org Wiki under [Stepper Drive Timing](#).

Additional signal conditioning or isolation such as optocouplers and RC filters on break out boards can impose timing constraints of their own, in addition to those of the driver. You may find it necessary to add some time to the drive requirements to allow for this.

The LinuxCNC Configuration Selector has configs for Sherline already configured.

- *Step Time* - How long the step pulse is *on* in nano seconds. If your not sure about this setting a value of 20,000 will work with most drives.
- *Step Space* - Minimum time between step pulses in nano seconds. If your not sure about this setting a value of 20,000 will work with most drives.
- *Direction Hold* - How long the direction pin is held after a change of direction in nanoseconds. If your not sure about this setting a value of 20,000 will work with most drives.
- *Direction Setup* - How long before a direction change after the last step pulse in nanoseconds. If your not sure about this setting a value of 20,000 will work with most drives.
- *First Parport* - Usually the default of 0x378 is correct.
- *Second Parport* - If you need to specify additional parallel ports enter the address and the type. For information on finding the address of PCI parallel ports see the Port Address in the Integrator Manual. (Try 0x278 or 0x3BC first.)
- *Base Period Maximum Jitter* - Enter the result of the Latency Test here. To run a latency test press the *Test Base Period Jitter* button. See the [Latency Test](#) section for more details.
- *Max Step Rate* - Stepconf automatically calculates the Max Step Rate based on the driver characteristics entered and the latency test result.
- *Min Base Period* - Stepconf automatically determines the Min Base Period based on the driver characteristics entered and latency test result.
- *Onscreen Prompt For Tool Change* - If this box is checked, LinuxCNC will pause and prompt you to change the tool when *M6* is encountered. This feature is usually only useful if you have presettable tools.

5.3 Latency Test

While the test is running, you should *abuse* the computer. Move windows around on the screen. Surf the web. Copy some large files around on the disk. Play some music. Run an OpenGL program such as glxgears. The idea is to put the PC through its paces while the latency test checks to see what the worst case numbers are. Run the test at least a few minutes. The longer you run the test the better it will be at catching events that might occur at less frequent intervals. This is a test for your computer only, so no hardware needs to be connected to run the test.



Warning

Do not attempt run LinuxCNC while the latency test is running.

Let this test run for a few minutes, then note the maximum Jitter. You will use it while configuring emc2.

While the test is running, you should "abuse" the computer. Move windows around on the screen. Surf the web. Copy some large files around on the disk. Play some music. Run an OpenGL program such as glxgears. The idea is to put the PC through its paces while the latency test checks to see what the worst case numbers are.

	Max Interval (ns)	Max Jitter (ns)	Last interval (ns)
Servo thread (1.0ms):	1001089	5929	995302
Base thread (25.0μs):	33954	9075	24843
Reset Statistics			

Figure 5.3: Latency Test

Latency is how long it takes the PC to stop what it is doing and respond to an external request. In our case, the request is the periodic *heartbeat* that serves as a timing reference for the step pulses. The lower the latency, the faster you can run the heartbeat, and the faster and smoother the step pulses will be.

Latency is far more important than CPU speed. The CPU isn't the only factor in determining latency. Motherboards, video cards, USB ports, SMI issues, and a number of other things can hurt the latency.

Troubleshooting SMI Issues (LinuxCNC.org Wiki)

Fixing Realtime problems caused by SMI on Ubuntu

<http://wiki.linuxcnc.org/cgi-bin/wiki.pl?FixingSMIIssues>

The important numbers are the *max jitter*. In the example above 9075 nanoseconds, or 9.075 microseconds, is the highest jitter. Record this number, and enter it in the Base Period Maximum Jitter box.

If your Max Jitter number is less than about 15-20 microseconds (15000-20000 nanoseconds), the computer should give very nice results with software stepping. If the max latency is more like 30-50 microseconds, you can still get good results, but your maximum step rate might be a little disappointing, especially if you use microstepping or have very fine pitch leadscrews. If the numbers are 100 us or more (100,000 nanoseconds), then the PC is not a good candidate for software stepping. Numbers over 1 millisecond (1,000,000 nanoseconds) mean the PC is not a good candidate for LinuxCNC, regardless of whether you use software stepping or not.

5.4 Parallel Port Setup

Parallel Port Setup

Outputs (PC to Mill):	Invert	Inputs (Mill to PC):	Invert
Pin 1: ESTOP Out	<input type="checkbox"/>	Pin 10: Unused	<input type="checkbox"/>
Pin 2: X Step	<input type="checkbox"/>	Pin 11: Unused	<input type="checkbox"/>
Pin 3: X Direction	<input type="checkbox"/>	Pin 12: Unused	<input type="checkbox"/>
Pin 4: Y Step	<input type="checkbox"/>	Pin 13: Unused	<input type="checkbox"/>
Pin 5: Y Direction	<input type="checkbox"/>	Pin 15: Unused	<input type="checkbox"/>
Pin 6: Z Step	<input type="checkbox"/>		
Pin 7: Z Direction	<input type="checkbox"/>		
Pin 8: A Step	<input type="checkbox"/>		
Pin 9: A Direction	<input type="checkbox"/>		
Pin 14: Spindle CW	<input type="checkbox"/>		
Pin 16: Spindle PWM	<input type="checkbox"/>		
Pin 17: Amplifier Enable	<input type="checkbox"/>		

Output pinout presets:

- [Sherline Outputs](#)
- [Xylotex Outputs](#)

[Cancel](#) [Back](#) [Forward](#)

Figure 5.4: Parallel Port Setup Page

For each pin, choose the signal which matches your parallel port pinout. Turn on the *invert* check box if the signal is inverted (0V for true/active, 5V for false/inactive).

- *Output pinout presets* - Automatically set pins 2 through 9 according to the Sherline standard (Direction on pins 2, 4, 6, 8) or the Xylotex standard (Direction on pins 3, 5, 7, 9).
- *Inputs and Outputs* - If the input or output is not used set the option to *Unused*.
- *External E Stop* - This can be selected from an input pin drop down box. A typical E Stop chain uses all normally closed contacts.
- *Homing & Limit Switches* - These can be selected from an input pin drop down box for most configurations.
- *Charge Pump* - If your driver board requires a charge pump signal select Charge Pump from the drop down list for the output pin you wish to connect to your charge pump input. The charge pump output is connected to the base thread by Stepconf. The charge pump output will be about 1/2 of the maximum step rate shown on the Basic Machine Configuration page.

5.5 Axis Configuration

X Axis Configuration

<u>Motor steps per revolution:</u>	<input type="text" value="200"/>	
<u>Driver Microstepping:</u>	<input type="text" value="2"/>	
<u>Pulley teeth (Motor:Leadscrew):</u>	<input type="text" value="1"/> : <input type="text" value="1"/>	
<u>Leadscrew Pitch:</u>	<input type="text" value="20"/>	rev / in
<u>Maximum Velocity:</u>	<input type="text" value="1"/>	in / s
<u>Maximum Acceleration:</u>	<input type="text" value="30"/>	in / s ²
<hr/>		
<u>Home location:</u>	<input type="text" value="0"/>	
<u>Table travel:</u>	<input type="text" value="0"/> to <input type="text" value="8"/>	
<u>Home Switch location:</u>	<input type="text" value="0"/>	
<u>Home Search velocity:</u>	<input type="text" value="0.05"/>	
<u>Home Latch direction:</u>	<input type="text" value="Same"/>	
<hr/>		
<u>Time to accelerate to max speed:</u>	0.0333 s	
<u>Distance to accelerate to max speed:</u>	0.0167 in	
<u>Pulse rate at max speed:</u>	8000.0 Hz	
<u>Axis SCALE:</u>	8000.0 Steps / in	

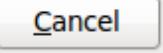
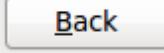
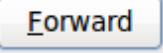
  

Figure 5.5: Axis Configuration Page

- **Motor Steps Per Revolution** - The number of full steps per motor revolution. If you know how many degrees per step the motor is (e.g., 1.8 degree), then divide 360 by the degrees per step to find the number of steps per motor revolution.
- **Driver Microstepping** - The amount of microstepping performed by the driver. Enter 2 for half-stepping.
- **Pulley Ratio** - If your machine has pulleys between the motor and leadscrew, enter the ratio here. If not, enter 1:1.
- **Leadscrew Pitch** - Enter the pitch of the leadscrew here. If you chose *Inch* units, enter the number of threads per inch. If you chose *mm* units, enter the number of millimeters per revolution (e.g., enter 2 for 2mm/rev). If the machine travels in the wrong direction, enter a negative number here instead of a positive number, or invert the direction pin for the axis.
- **Maximum Velocity** - Enter the maximum velocity for the axis in units per second.
- **Maximum Acceleration** - The correct values for these items can only be determined through experimentation. See [Finding Maximum Velocity](#) to set the speed and [Finding Maximum Acceleration](#) to set the acceleration.

- **Home Location** - The position the machine moves to after completing the homing procedure for this axis. For machines without home switches, this is the location the operator manually moves the machine to before pressing the Home button. If you combine the home and limit switches you must move off of the switch to the home position or you will get a joint limit error.
- **Table Travel** - The range of travel for that axis based on the machine origin. The home location must be inside the *Table Travel* and not equal to one of the Table Travel values.
- **Home Switch Location** - The location at which the home switch trips or releases relative to the machine origin. This item and the two below only appear when Home Switches were chosen in the Parallel Port Pinout. If you combine home and limit switches the home switch location can not be the same as the home position or you will get a joint limit error.
- **Home Search Velocity** - The velocity to use when searching for the home switch. If the switch is near the end of travel, this velocity must be chosen so that the axis can decelerate to a stop before hitting the end of travel. If the switch is only closed for a short range of travel (instead of being closed from its trip point to one end of travel), this velocity must be chosen so that the axis can decelerate to a stop before the switch opens again, and homing must always be started from the same side of the switch. If the machine moves the wrong direction at the beginning of the homing procedure, negate the value of *Home Search Velocity*.
- **Home Latch Direction** - Choose *Same* to have the axis back off the switch, then approach it again at a very low speed. The second time the switch closes, the home position is set. Choose *Opposite* to have the axis back off the switch and when the switch opens, the home position is set.
- **Time to accelerate to max speed** - Time to reach maximum speed calculated from *Max Acceleration* and *Max Velocity*.
- **Distance to accelerate to max speed** - Distance to reach maximum speed from a standstill.
- **Pulse rate at max speed** - Information computed based on the values entered above. The greatest *Pulse rate at max speed* determines the *BASE_PERIOD*. Values above 20000Hz may lead to slow response time or even lockups (the fastest usable pulse rate varies from computer to computer)
- **Axis SCALE** - The number that will be used in the ini file [SCALE] setting. This is how many steps per user unit.
- **Test this axis** - This will open a window to allow testing for each axis. This can be used after filling out all the information for this axis.

5.5.1 Test This Axis

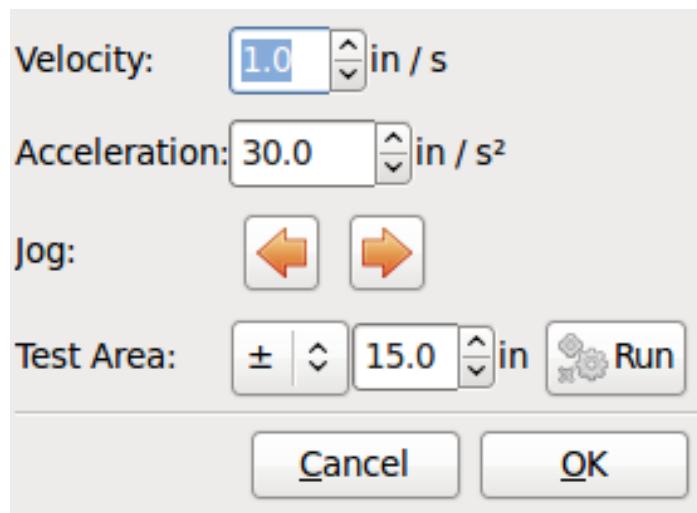


Figure 5.6: Test This Axis

Test this axis is a basic tester that only outputs step and direction signals to try different values for acceleration and velocity.

**Important**

In order to use test this axis you have to manually enable the axis if this is required. If your driver has a charge pump you will have to bypass it. Test this axis does not react to limit switch inputs. Use with caution.

5.5.1.1 Finding Maximum Velocity

Begin with a low Acceleration (for example, **2 inches/s²** or **50 mm/s²**) and the velocity you hope to attain. Using the buttons provided, jog the axis to near the center of travel. Take care because with a low acceleration value, it can take a surprising distance for the axis to decelerate to a stop.

After gaging the amount of travel available, enter a safe distance in Test Area, keeping in mind that after a stall the motor may next start to move in an unexpected direction. Then click Run. The machine will begin to move back and forth along this axis. In this test, it is important that the combination of Acceleration and Test Area allow the machine to reach the selected Velocity and *cruise* for at least a short distance — the more distance, the better this test is. The formula $d = 0.5 * v * v/a$ gives the minimum distance required to reach the specified velocity with the given acceleration. If it is convenient and safe to do so, push the table against the direction of motion to simulate cutting forces. If the machine stalls, reduce the speed and start the test again.

If the machine did not obviously stall, click the *Run* button off. The axis now returns to the position where it started. If the position is incorrect, then the axis stalled or lost steps during the test. Reduce Velocity and start the test again.

If the machine doesn't move, stalls, or loses steps, no matter how low you turn Velocity, verify the following:

- Correct step waveform timings
- Correct pinout, including *Invert* on step pins
- Correct, well-shielded cabling
- Physical problems with the motor, motor coupling, leadscrew, etc.

Once you have found a speed at which the axis does not stall or lose steps during this testing procedure, reduce it by 10% and use that as the axis *Maximum Velocity*.

5.5.1.2 Finding Maximum Acceleration

With the Maximum Velocity you found in the previous step, enter the acceleration value to test. Using the same procedure as above, adjust the Acceleration value up or down as necessary. In this test, it is important that the combination of Acceleration and Test Area allow the machine to reach the selected Velocity. Once you have found a value at which the axis does not stall or lose steps during this testing procedure, reduce it by 10% and use that as the axis Maximum Acceleration.

5.6 Spindle Configuration

Spindle Configuration

PWM Rate: Hz Enter 0 Hz for "PDM" mode

Calibration:

Speed 1: PWM 1:

Speed 2: PWM 2:

Cycles per revolution:

Figure 5.7: Spindle Configuration Page

This page only appears when *Spindle PWM* is chosen in the *Parallel Port Pinout* page for one of the outputs.

5.6.1 Spindle Speed Control

If *Spindle PWM* appears on the pinout, the following information should be entered:

- *PWM Rate* - The *carrier frequency* of the PWM signal to the spindle. Enter 0 for PDM mode, which is useful for generating an analog control voltage. Refer to the documentation for your spindle controller for the appropriate value.
- *Speed 1 and 2, PWM 1 and 2* - The generated configuration file uses a simple linear relationship to determine the PWM value for a given RPM value. If the values are not known, they can be determined. For more information see [Determining Spindle Calibration](#).

5.6.2 Spindle-synchronized motion

When the appropriate signals from a spindle encoder are connected to LinuxCNC via HAL, LinuxCNC supports lathe threading. These signals are:

- *Spindle Index* - Is a pulse that occurs once per revolution of the spindle.
- *Spindle Phase A* - This is a pulse that occurs in multiple equally-spaced locations as the spindle turns.
- *Spindle Phase B (optional)* - This is a second pulse that occurs, but with an offset from Spindle Phase A. The advantages to using both A and B are direction sensing, increased noise immunity, and increased resolution.

If *Spindle Phase A* and *Spindle Index* appear on the pinout, the following information should be entered:

- *Cycles per revolution* - The number of cycles of the *Spindle A* signal during one revolution of the spindle. This option is only enabled when an input has been set to *Spindle Phase A*
- *Maximum speed in thread* - The maximum spindle speed used in threading. For a high spindle RPM or a spindle encoder with high resolution, a low value of *BASE_PERIOD* is required.

5.6.3 Determining Spindle Calibration

Enter the following values in the Spindle Configuration page:

Speed 1:	0	PWM 1:	0
Speed 2:	1000	PWM 2:	1

Finish the remaining steps of the configuration process, then launch LinuxCNC with your configuration. Turn the machine on and select the MDI tab. Start the spindle turning by entering: *M3 S100*. Change the spindle speed by entering a different S-number: *S800*. Valid numbers (at this point) range from 1 to 1000.

For two different S-numbers, measure the actual spindle speed in RPM. Record the S-numbers and actual spindle speeds. Run Stepconf again. For *Speed* enter the measured speed, and for *PWM* enter the S-number divided by 1000.

Because most spindle drivers are somewhat nonlinear in their response curves, it is best to:

- Make sure the two calibration speeds are not too close together in RPM
- Make sure the two calibration speeds are in the range of speeds you will typically use while milling

For instance, if your spindle will go from 0 RPM to 8000 RPM, but you generally use speeds from 400 RPM (10%) to 4000 RPM (100%), then find the PWM values that give 1600 RPM (40%) and 2800 RPM (70%).

5.7 Advanced Configuration Options

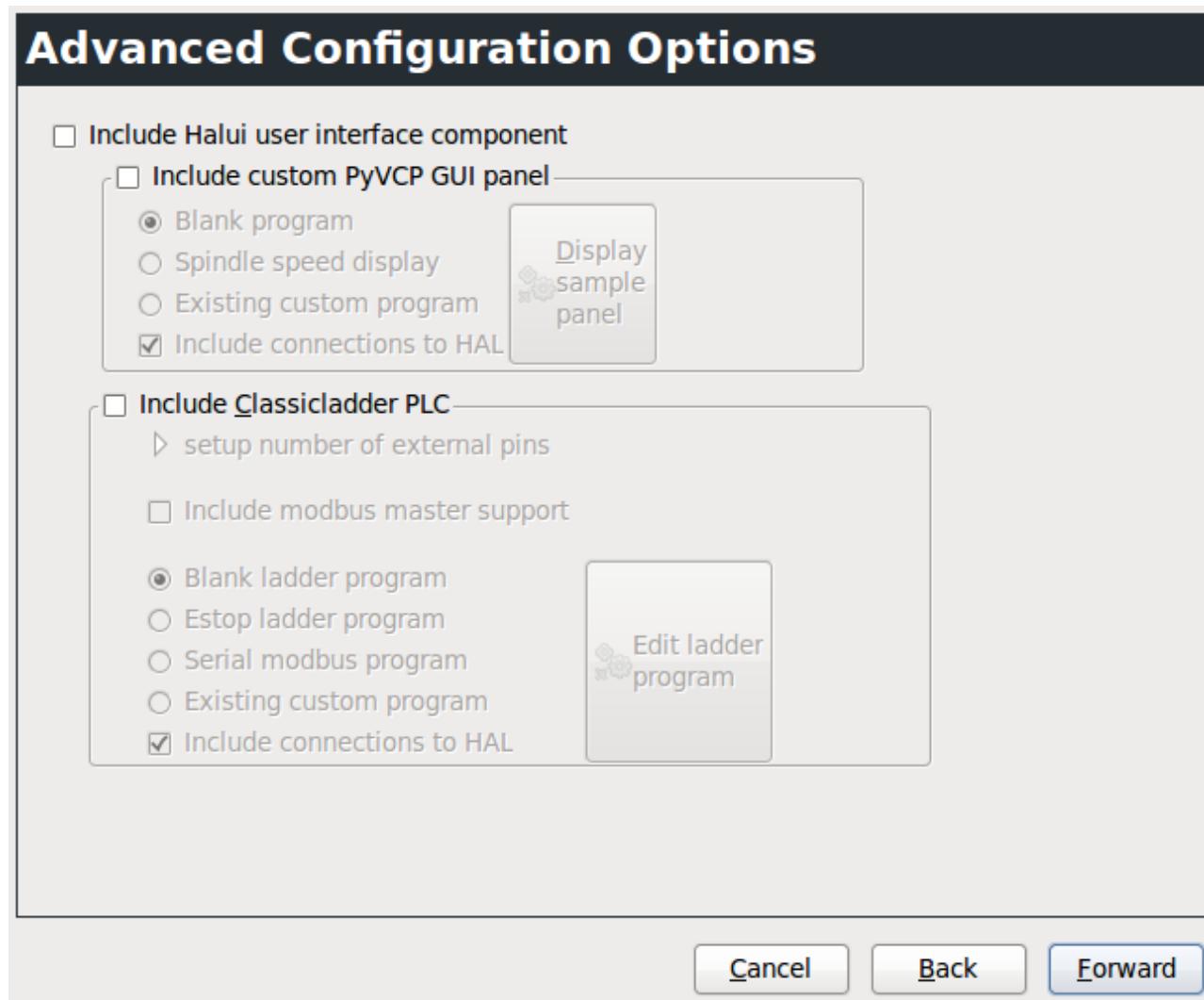


Figure 5.8: Advanced Configuration

- *Include Halui* - This will add the Halui user interface component. See the Integrator Manual for more information on Halui.
- *Include pyVCP* - This option adds the pyVCP panel base file or a sample file to work on. See the Integrator Manual for more information on pyVCP.
- *Include ClassicLadder PLC* - This option will add the ClassicLadder PLC (Programmable Logic Controller). See the Integrator Manual for more information on ClassicLadder.

5.8 Machine Configuration Complete

Click *Apply* to write the configuration files. Later, you can re-run this program and tweak the settings you entered before.

5.9 Axis Travel, Home Location, and Home Switch Location

For each axis, there is a limited range of travel. The physical end of travel is called the *hard stop*.

Before the *hard stop* there is a *limit switch*. If the limit switch is encountered during normal operation, LinuxCNC shuts down the motor amplifier. The distance between the *hard stop* and *limit switch* must be long enough to allow an unpowered motor to coast to a stop.

Before the *limit switch* there is a *soft limit*. This is a limit enforced in software after homing. If a MDI command or g code program would pass the soft limit, it is not executed. If a jog would pass the soft limit, it is terminated at the soft limit.

The *home switch* can be placed anywhere within the travel (between hard stops). As long as external hardware does not deactivate the motor amplifiers when the limit switch is reached, one of the limit switches can be used as a home switch.

The *zero position* is the location on the axis that is 0 in the machine coordinate system. Usually the *zero position* will be within the *soft limits*. On lathes, constant surface speed mode requires that machine $X=0$ correspond to the center of spindle rotation when no tool offset is in effect.

The *home position* is the location within travel that the axis will be moved to at the end of the homing sequence. This value must be within the *soft limits*. In particular, the *home position* should never be exactly equal to a *soft limit*.

5.9.1 Operating without Limit Switches

A machine can be operated without limit switches. In this case, only the soft limits stop the machine from reaching the hard stop. Soft limits only operate after the machine has been homed.

5.9.2 Operating without Home Switches

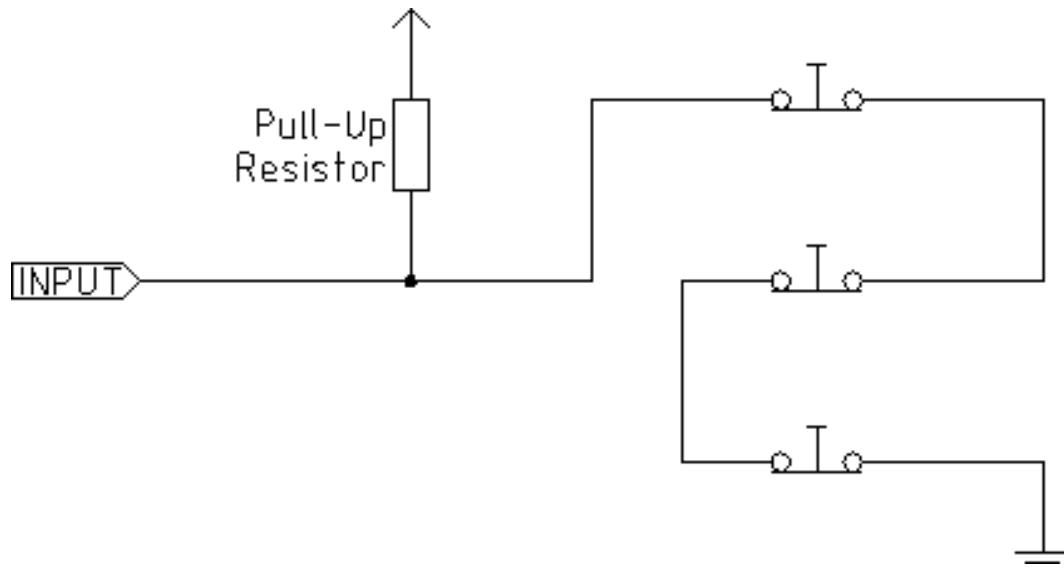
A machine can be operated without home switches. If the machine has limit switches, but no home switches, it is best to use a limit switch as the home switch (e.g., choose *Minimum Limit + Home X* in the pinout). If the machine has no switches at all, or the limit switches cannot be used as home switches for another reason, then the machine must be homed *by eye* or by using match marks. Homing by eye is not as repeatable as homing to switches, but it still allows the soft limits to be useful.

5.9.3 Home and Limit Switch wiring options

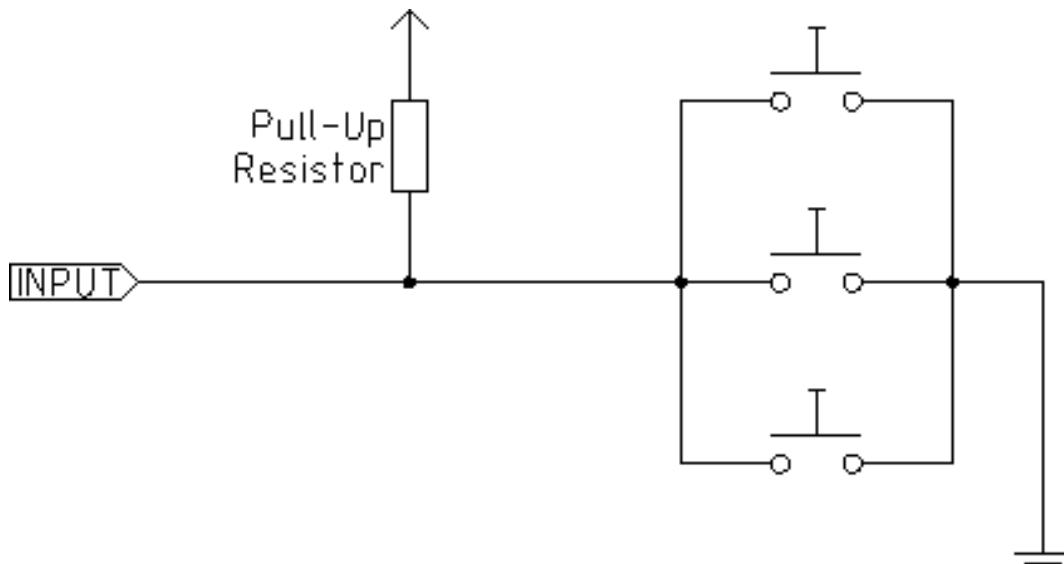
The ideal wiring for external switches would be one input per switch. However, the PC parallel port only offers a total of 5 inputs, while there are as many as 9 switches on a 3-axis machine. Instead, multiple switches are wired together in various ways so that a smaller number of inputs are required.

The figures below show the general idea of wiring multiple switches to a single input pin. In each case, when one switch is actuated, the value seen on INPUT goes from logic HIGH to LOW. However, LinuxCNC expects a TRUE value when a switch is closed, so the corresponding *Invert* box must be checked on the pinout configuration page. The pull up resistor shown in the diagrams pulls the input high until the connection to ground is made and then the input goes low. Otherwise the input might float between on and off when the circuit is open. Typically for a parallel port you might use 47k.

Normally Closed Switches Wiring N/C switches in series (simplified diagram)



Normally Open Switches Wiring N/O switches in parallel (simplified diagram)



The following combinations of switches are permitted in Stepconf:

- Combine home switches for all axes
- Combine limit switches for all axes
- Combine both limit switches for one axis
- Combine both limit switches and the home switch for one axis
- Combine one limit switch and the home switch for one axis

Chapter 6

Mesa Configuration Wizard

PNCconf is made to help build configurations that utilize specific Mesa *Anything I/O* products.

It can configure closed loop servo systems or hardware stepper systems. It uses a similar *wizard* approach as Stepconf (used for software stepping, parallel port driven systems).

PNCconf is still in a development stage (Beta) so there are some bugs and lacking features. Please report bugs and suggestions to the LinuxCNC forum page or mail-list.

There are two trains of thought when using PNCconf:

One is to use PNCconf to always configure your system - if you decide to change options, reload PNCconf and allow it to configure the new options. This will work well if your machine is fairly standard and you can use custom files to add non standard features. PNCconf tries to work with you in this regard.

The other is to use PNCconf to build a config that is close to what you want and then hand edit everything to tailor it to your needs. This would be the choice if you need extensive modifications beyond PNCconf's scope or just want to tinker with / learn about LinuxCNC

You navigate the wizard pages with the forward, back, and cancel buttons there is also a help button that gives some help information about the pages, diagrams and an output page.

Tip

PNCconf's help page should have the most up to date info and has additional details.

Step by Step Instructions

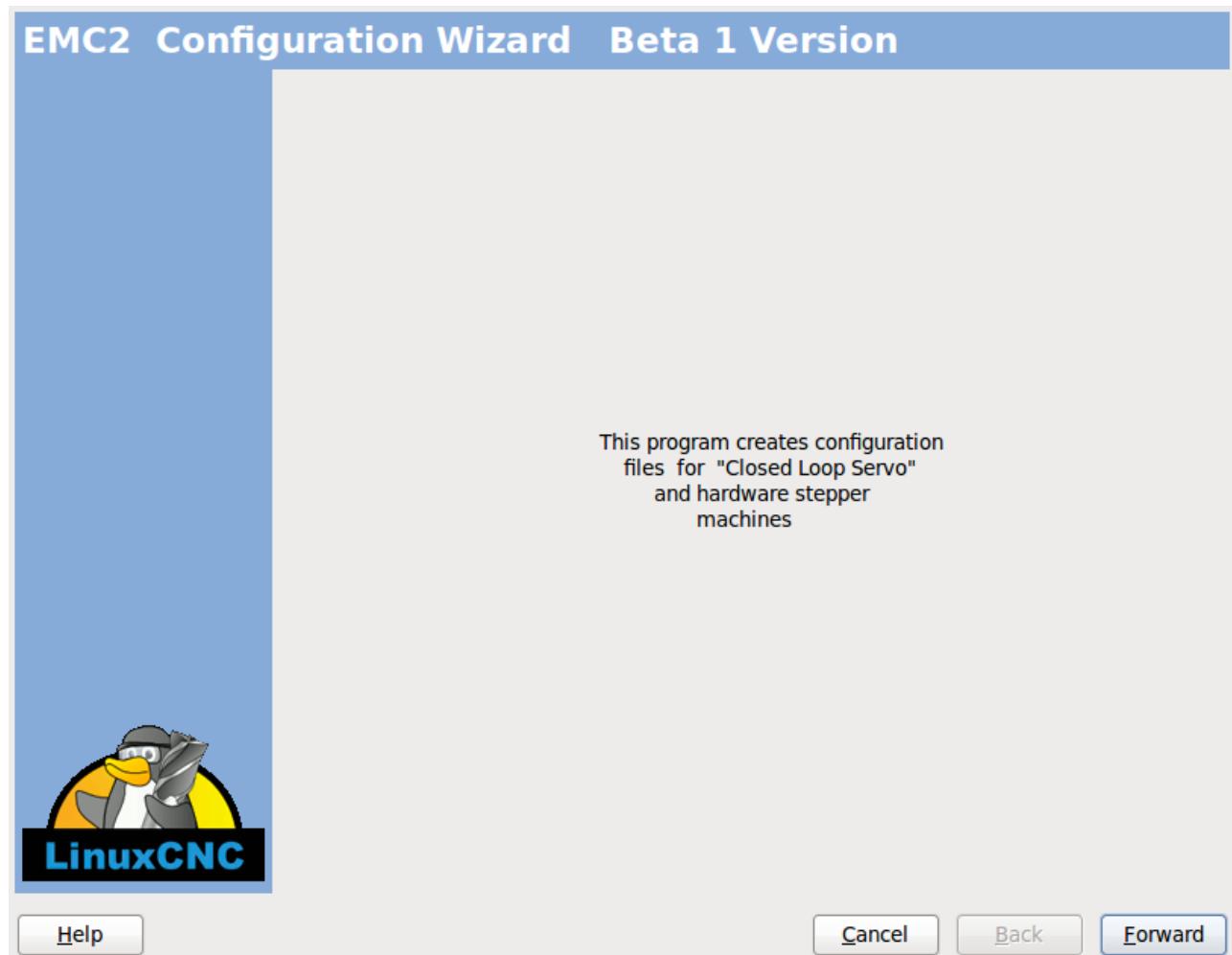


Figure 6.1: PnCConf Splash

6.1 Create or Edit

This allows you to select a previously saved configuration or create a new one. If you pick *Modify a configuration* and then press next a file selection box will show. Pncconf preselects your last saved file. Choose the config you wish to edit. It also allows you to select desktop shortcut / launcher options. A desktop shortcut will place a folder icon on the desktop that points to your new configuration files. Otherwise you would have to look in your home folder under emc2/configs.

A Desktop launcher will add an icon to the desktop for starting your config directly. You can also launch it under Applications->cnc/emc2 and selecting your config name.

6.2 Basic Machine Information

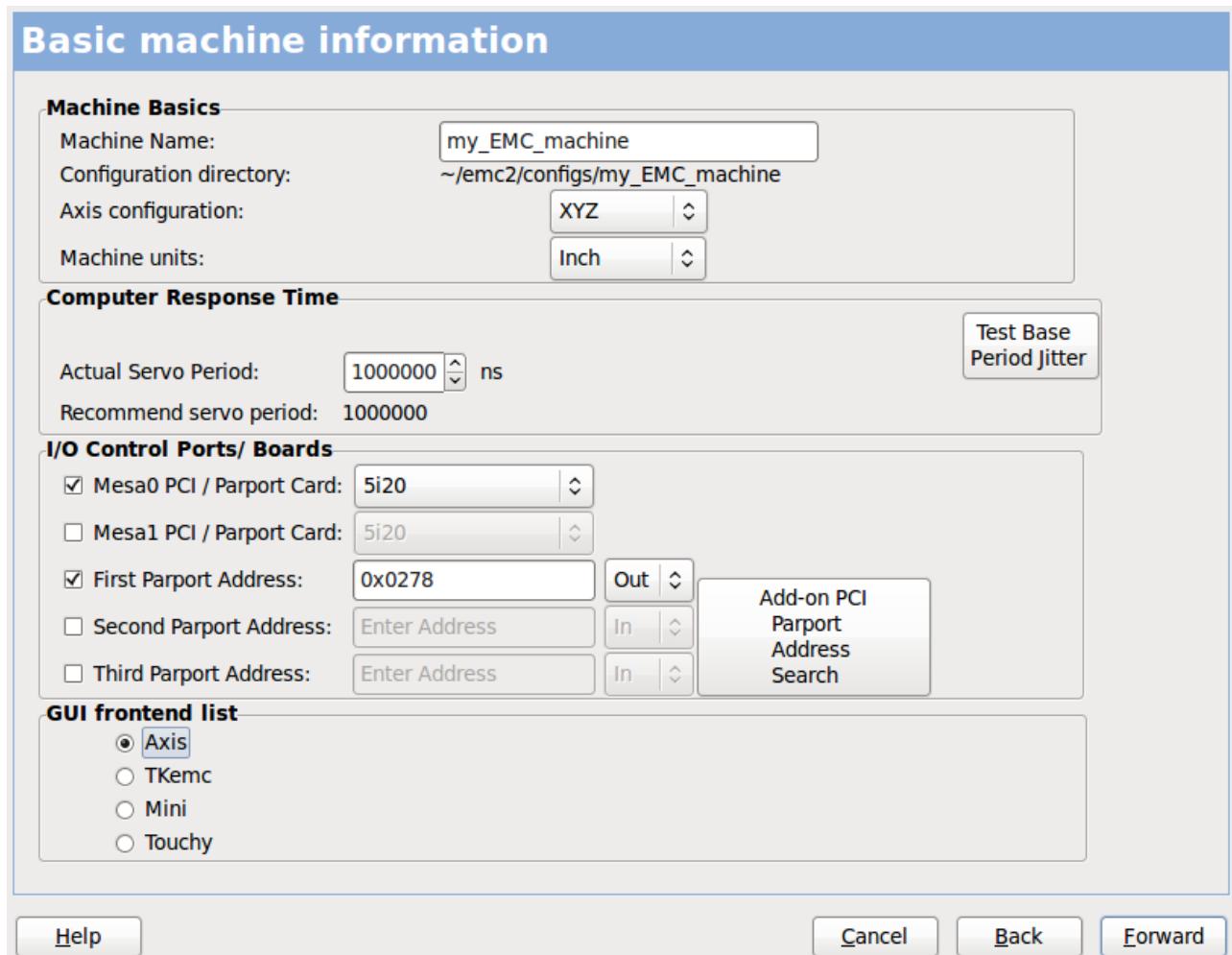


Figure 6.2: PnCConf Basic

Machine Basics

If you use a name with spaces PNCconf will replace the spaces with underscore (as a loose rule Linux doesn't like spaces in names) Pick an axis configuration - this selects what type of machine you are building and what axes are available. The Machine units selector allows data entry of metric or imperial units in the following pages.

Tip

Defaults are not converted when using metric so make sure they are sane values!

Computer Response Time

The servo period sets the heart beat of the system. Latency refers to the amount of time the computer can be longer than that period. Just like a railroad, LinuxCNC requires everything on a very tight and consistent time line or bad things happen. LinuxCNC requires and uses a *real time* operating system, which just means it has a low latency (lateness)

response time when LinuxCNC requires its calculations and when doing LinuxCNC's calculations it cannot be interrupted by lower priority requests (such as user input to screen buttons or drawing etc).

Testing the latency is very important and a key thing to check early. Luckily by using the Mesa card to do the work that requires the fastest response time (encoder counting and PWM generation) we can endure a lot more latency than if we used the parallel port for these things. The standard test in LinuxCNC is checking the BASE period latency (even though we are not using a base period). If you press the *test base period jitter* button, this launches the latency test window (you can also load this directly from the applications/cnc panel). The test mentions to run it for a few minutes but the longer the better. consider 15 minutes a bare minimum and overnight even better. At this time use the computer to load things, use the net, use USB etc we want to know the worst case latency and to find out if any particular activity hurts our latency. We need to look at base period jitter. Anything under 20000 is excellent - you could even do fast software stepping with the machine 20000 - 50000 is still good for software stepping and fine for us. 50000 - 100000 is really not that great but could still be used with hardware cards doing the fast response stuff. So anything under 100000 is useable to us. If the latency is disappointing or you get a bad hiccup periodically you may still be able to improve it.

Tip

There is a user compiled list of equipment and the latency obtained on the LinuxCNC wiki : <http://wiki.linuxcnc.org/cgi-bin-wiki.pl?Latency-Test> Please consider adding your info to the list. Also on that page are links to info about fixing some latency problems.

Now we are happy with the latency and must pick a servo period. In most cases a servo period of 1000000 ns is fine (that gives a 1 kHz servo calculation rate - 1000 calculations a second) if you are building a closed loop servo system that controls torque (current) rather than velocity (voltage) a faster rate would be better - something like 200000 (5 kHz calculation rate). The problem with lowering the servo rate is that it leaves less time available for the computer to do other things besides LinuxCNC's calculations. Typically the display (GUI) becomes less responsive. You must decide on a balance. Keep in mind that if you tune your closed loop servo system then change the servo period you probably will need to tune them again.

I/O Control Ports/Boards

PNCconf is capable of configuring machines that have up to two Mesa boards and three parallel ports. Parallel ports can only be used for simple low speed (servo rate) I/O.

Mesa

You must choose at least one Mesa board as PNCconf will not configure the parallel ports to count encoders or output step or PWM signals. The mesa cards available in the selection box are based on what PNCconf finds for firmware on the systems. There are options to add custom firmware and/or *blacklist* (ignore) some firmware or boards using a preference file. If no firmware is found PNCconf will show a warning and use internal sample firmware - no testing will be possible. One point to note is that if you choose two PCI Mesa cards there currently is no way to predict which card is 0 and which is 1 - you must test - moving the cards could change their order. If you configure with two cards both cards must be installed for tests to function.

Parallel Port

Up to 3 parallel ports (referred to as parports) can be used as simple I/O. You must set the address of the parport. You can either enter the Linux parallel port numbering system (0,1,or 2) or enter the actual address. The address for an on board parport is often 0x0278 or 0x0378 (written in hexadecimal) but can be found in the BIOS page. The BIOS page is found when you first start your computer you must press a key to enter it (such as F2). On the BIOS page you can find the parallel port address and set the mode such as SPP, EPP, etc on some computers this info is displayed for a few seconds during start up. For PCI parallel port cards the address can be found by pressing the *parport address search* button. This pops up the help output page with a list of all the PCI devices that can be found. In there should be a reference to a parallel port device with a list of addresses. One of those addresses should work. Not all PCI parallel ports work properly. Either type can be selected as *in* (maximum amount of input pins) or *out* (maximum amount of output pins)

GUI Frontend list

This specifies the graphical display screens LinuxCNC will use. Each one has different option.

AXIS

- fully supports lathes.

- is the most developed and used frontend
- is designed to be used with mouse and keyboard
- is tkinter based so integrates PYVCP (python based virtual control panels) naturally.
- has a 3D graphical window.
- allows VCP integrated on the side or in center tab

TOUCHY

- Touchy was designed to be used with a touchscreen, some minimal physical switches and a MPG wheel.
- requires cycle-start, abort, and single-step signals and buttons
- It also requires shared axis MPG jogging to be selected.
- is GTK based so integrates GLADE VCP (virtual control panels) naturally.
- allows VCP panels integrated in the center Tab
- has no graphical window
- look can be changed with custom themes

MINI

- standard on OEM Sherline machines
- does not use Estop
- no VCP integration

TkLinuxCNC

- hi contrast bright blue screen
- separate graphics window
- no VCP integration

6.3 External Configuration

This page allows you to select external controls such as for jogging or overrides.

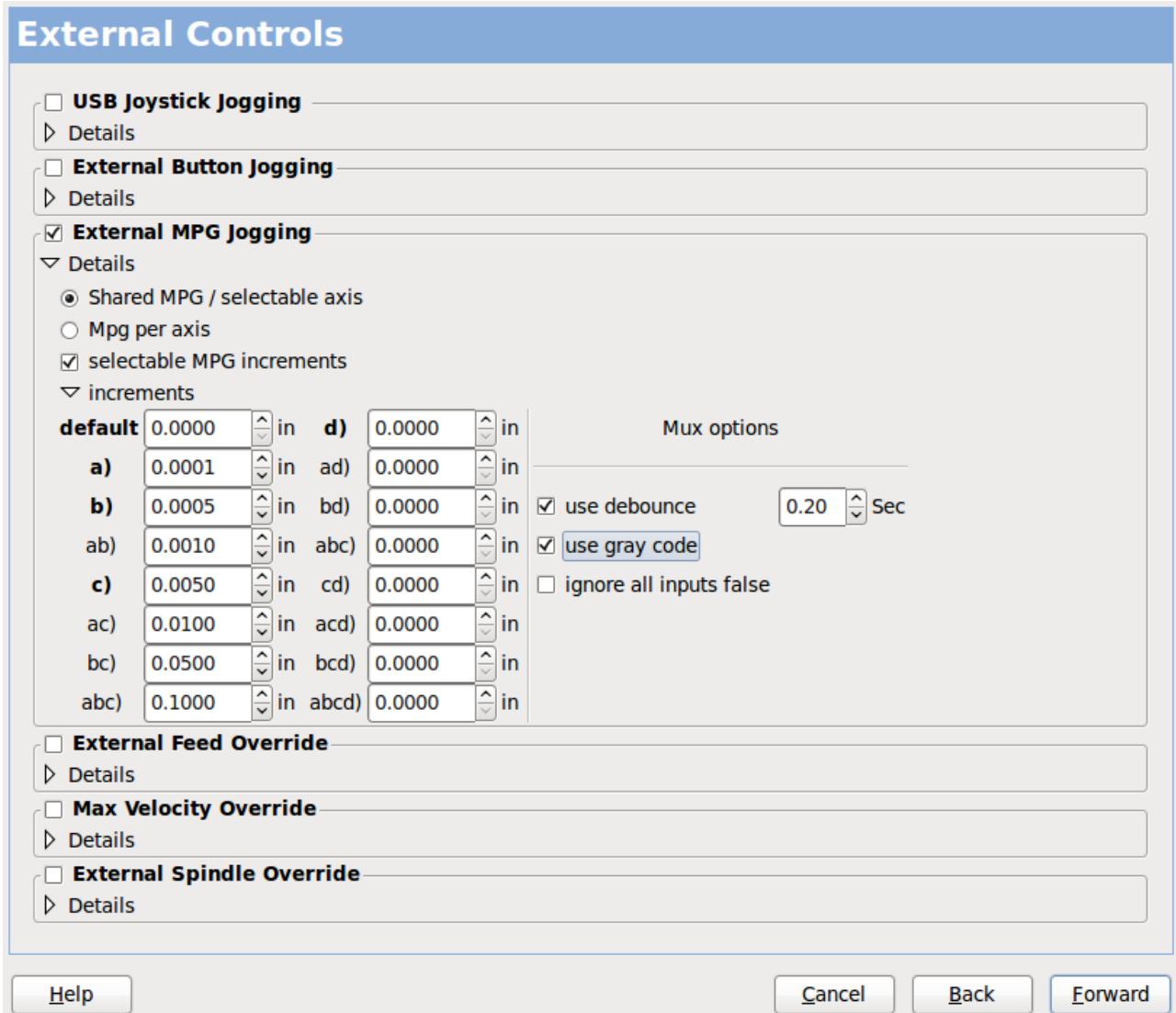


Figure 6.3: GUI External

If you select a Joystick for jogging, You will need it always connected for LinuxCNC to load. To use the analog sticks for useful jogging you probably need to add some custom HAL code. MPG jogging requires a pulse generator connected to a MESA encoder counter. Override controls can either use a pulse generator (MPG) or switches (such as a rotary dial). External buttons might be used with a switch based OEM joystick.

Joystick jogging

Requires a custom *device rule* to be installed in the system. This is a file that LinuxCNC uses to connect to LINUX's device list. PNCconf will help to make this file.

Search for device rule will search the system for rules, you can use this to find the name of devices you have already built with PNCconf.

Add a device rule will allow you to configure a new device by following the prompts. You will need your device available. *test device* allows you to load a device, see its pin names and check its functions with halmeter.

joystick jogging uses HALUI and hal_input components.

External buttons

allows jogging the axis with simple buttons at a specified jog rate. Probably best for rapid jogging.

MPG Jogging

Allows you to use a Manual Pulse Generator to jog the machine's axis.

MPG's are often found on commercial grade machines. They output quadrature pulses that can be counted with a MESA encoder counter. PNCconf allows for an MPG per axis or one MPG shared with all axis. It allows for selection of jog speeds using switches or a single speed.

The selectable increments option uses the mux16 component. This component has options such as debounce and gray code to help filter the raw switch input.

Overrides

PNCconf allows overrides of feedrates and/or spindle speed using a pulse generator (MPG) or switches (eg. rotary).

6.4 GUI Configuration

Here you can set defaults for the display screens, add virtual control panels (VCP), and set some LinuxCNC options..

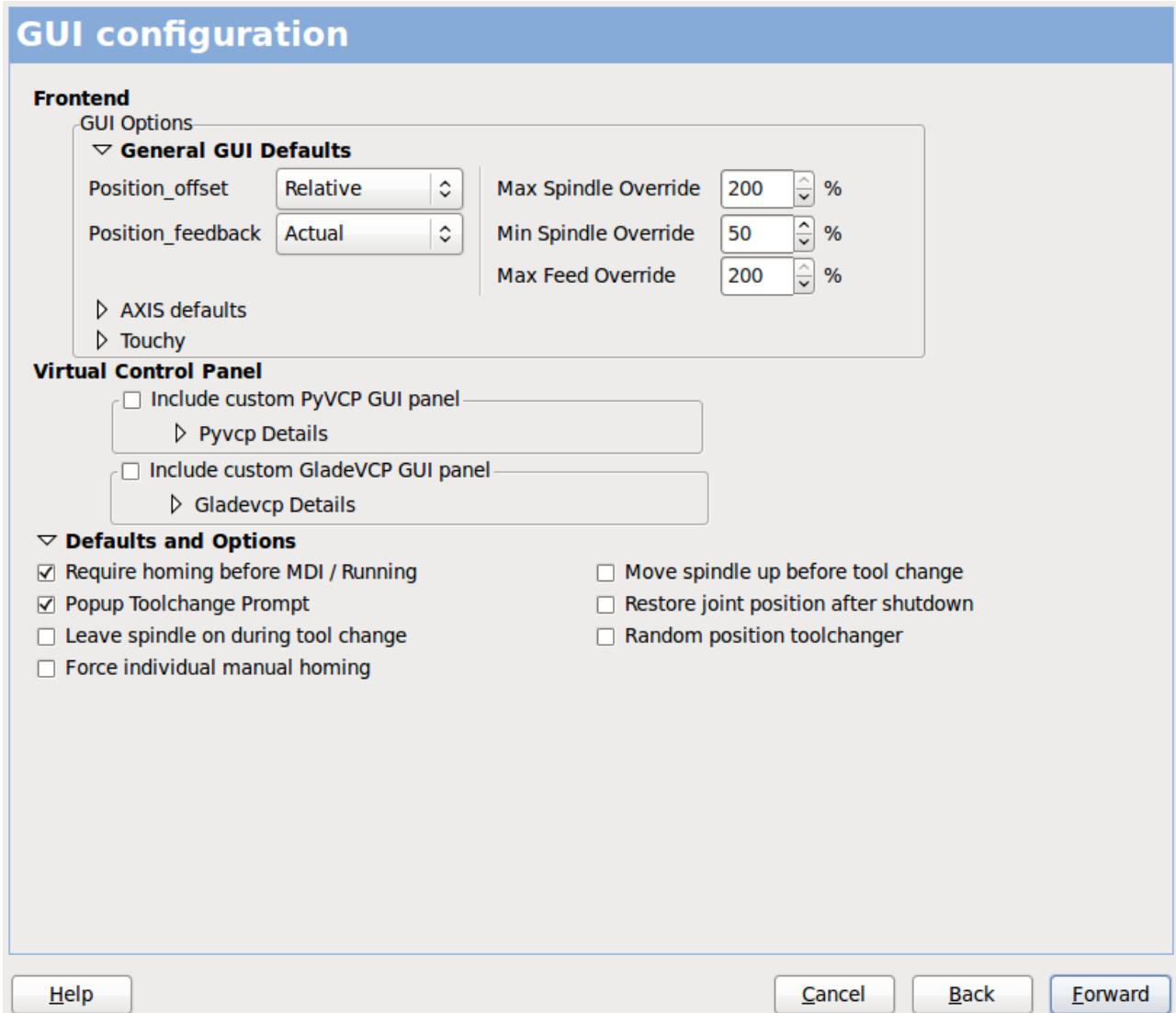


Figure 6.4: GUI Configuration

Frontend GUI Options

The default options allows general defaults to be chosen for any display screen.

AXIS defaults are options specific to AXIS. If you choose size , position or force maximize options then PNCconf will ask if it's alright to overwrite a preference file (.axisrc). Unless you have manually added commands to this file it is fine to allow it. Position and force max can be used to move AXIS to a second monitor if the system is capable.

Touchy defaults are options specific to Touchy. Most of Touchy's options can be changed while Touchy is running using the preference page. Touchy uses GTK to draw its screen, and GTK supports themes. Themes controls the basic look and feel of a program. You can download themes from the net or edit them yourself. There are a list of the current themes on the computer that you can pick from. To help some of the text to stand out PNCconf allows you to override the Themes's defaults. The position and force max options can be used to move Touchy to a second monitor if the system is capable.

VCP options

Virtual Control Panels allow one to add custom controls and displays to the screen. AXIS and Touchy can integrate these controls inside the screen in designated positions. There are two kinds of VCPs - pyVCP which uses *Tkinter* to draw the screen and GLADE VCP that uses *GTK* to draw the screen.

PyVCP

PyVCPS screen XML file can only be hand built. PyVCPS fit naturally in with AXIS as they both use TKinter.

HAL pins are created for the user to connect to inside their custom HAL file. There is a sample spindle display panel for the user to use as-is or build on. You may select a blank file that you can later add your controls *widgets* to or select a spindle display sample that will display spindle speed and indicate if the spindle is at requested speed.

PNCconf will connect the proper spindle display HAL pins for you. If you are using AXIS then the panel will be integrated on the right side. If not using AXIS then the panel will be separate *stand-alone* from the frontend screen.

You can use the geometry options to size and move the panel, for instance to move it to a second screen if the system is capable. If you press the *Display sample panel* button the size and placement options will be honoured.

GLADE VCP

GLADE VCPs fit naturally inside of TOUCHY screen as they both use GTK to draw them, but by changing GLADE VCP's theme it can be made to blend pretty well in AXIS. (try Redmond)

It uses a graphical editor to build its XML files. HAL pins are created for the user to connect to, inside of their custom HAL file.

GLADE VCP also allows much more sophisticated (and complicated) programming interaction, which PNCconf currently doesn't leverage. (see GLADE VCP in the manual)

PNCconf has sample panels for the user to use as-is or build on. With GLADE VCP PNCconf will allow you to select different options on your sample display.

Under *sample options* select which ones you would like. The zero buttons use HALUI commands which you could edit later in the HALUI section.

Auto Z touch-off also requires the classicladder touch-off program and a probe input selected. It requires a conductive touch-off plate and a grounded conductive tool. For an idea on how it works see:

http://wiki.linuxcnc.org/cgi-bin/wiki.pl?ClassicLadderExamples#Single_button_probe_touchoff

Under *Display Options*, size, position, and force max can be used on a *stand-alone* panel for such things as placing the screen on a second monitor if the system is capable.

You can select a GTK theme which sets the basic look and feel of the panel. You Usually want this to match the frontend screen. These options will be used if you press the *Display sample button*. With GLADE VCP depending on the frontend screen, you can select where the panel will display.

You can force it to be stand-alone or with AXIS it can be in the center or on the right side, with Touchy it can be in the center.

Defaults and Options

- Require homing before MDI / Running
 - If you want to be able to move the machine before homing uncheck this checkbox.
- Popup Tool Prompt
 - Choose between an on screen prompt for tool changes or export standard signal names for a User supplied custom tool changer Hal file
- Leave spindle on during tool change:
 - Used for lathes
- Force individual manual homing
- Move spindle up before tool change
- Restore joint position after shutdown
 - Used for non-trivial kinematics machines
- Random position toolchangers
 - Used for toolchangers that do not return the tool to the same pocket. You will need to add custom HAL code to support toolchangers.

6.5 Mesa Configuration

The Mesa configuration pages allow one to utilize different firmwares. On the basic page you selected a Mesa card here you pick the available firmware and select what and how many components are available.

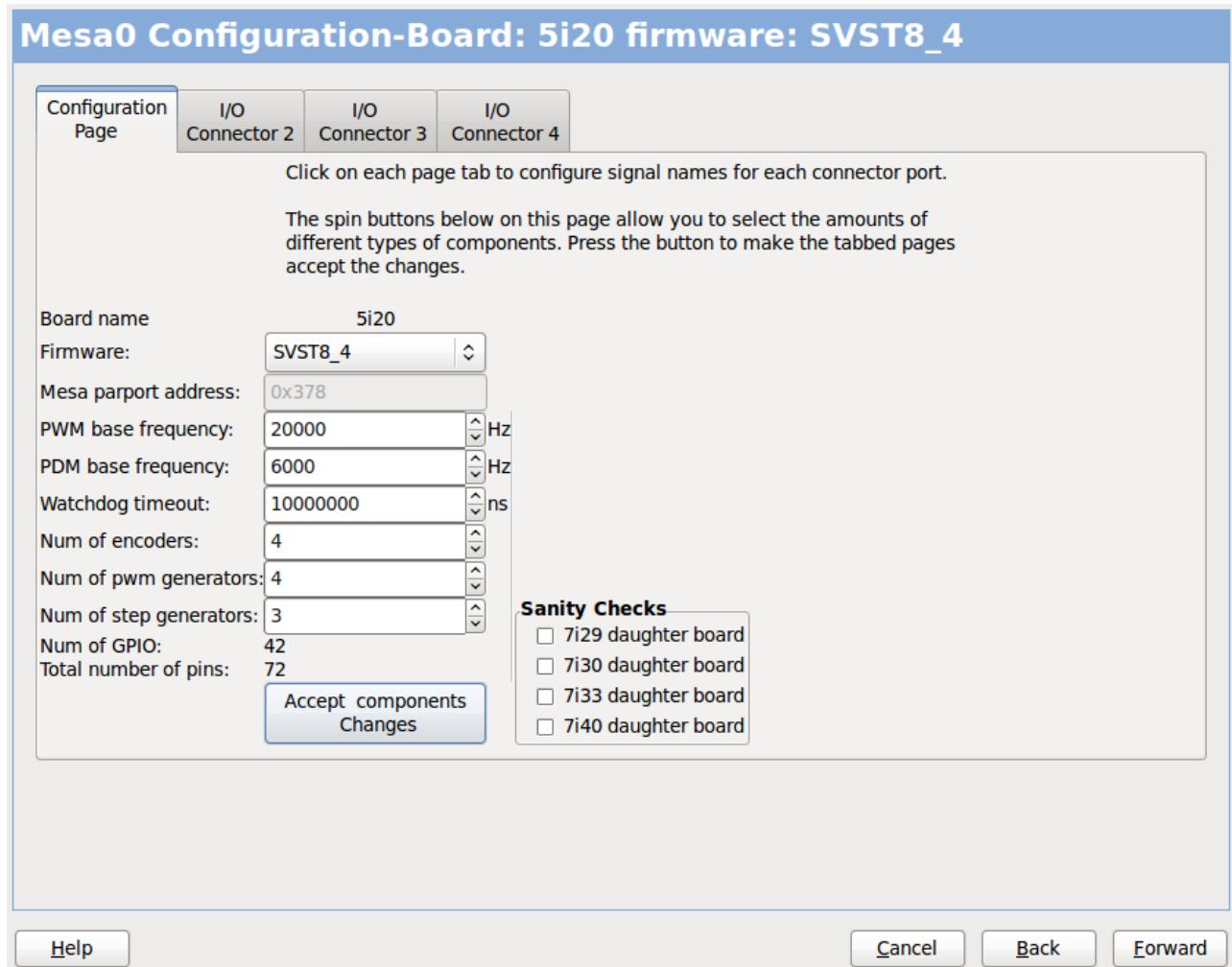


Figure 6.5: Mesa Configuration

Parport address is used only with Mesa parport card, the 7i43. An onboard parallel port usually uses 0x278 or 0x378 though you should be able to find the address from the BIOS page. The 7i43 requires the parallel port to use the EPP mode, again set in the BIOS page. If using a PCI parallel port the address can be searched for by using the search button on the basic page.

Note

Many PCI cards do not support the EPP protocol properly.

PDM PWM and 3PWM base frequency sets the balance between ripple and linearity. If using Mesa daughter boards the docs for the board should give recommendations

**Important**

It's important to follow these to avoid damage and get the best performance.

The 7i33 requires PDM and a PDM base frequency of 6 mHz
The 7i29 requires PWM and a PWM base frequency of 20 Khz
The 7i30 requires PWM and a PWM base frequency of 20 Khz
The 7i40 requires PWM and a PWM base frequency of 50 Khz
The 7i48 requires UDM and a PWM base frequency of 24 Khz

Watchdog time out is used to set how long the MESA board will wait before killing outputs if communication is interrupted from the computer. Please remember Mesa uses *active low* outputs meaning that when the output pin is on, it is low (approx 0 volts) and if it's off the output is high (approx 5 volts) make sure your equipment is safe when in the off (watchdog bitten) state.

You may choose the number of available components by deselecting unused ones. Not all component types are available with all firmware.

Choosing less than the maximum number of components allows one to gain more GPIO pins. If using daughter boards keep in mind you must not deselect pins that the card uses. For instance some firmware supports two 7i33 cards, If you only have one you may deselect enough components to utilize the connector that supported the second 7i33. Components are deselected numerically by the highest number first then down with out skipping a number. If by doing this the components are not where you want them then you must use a different firmware. The firmware dictates where, what and the max amounts of the components. Custom firmware is possible, ask nicely when contacting the LinuxCNC developers and Mesa. Using custom firmware in PNCconf requires special procedures and is not always possible - Though I try to make PNCconf as flexible as possible.

After choosing all these options press the *Accept Component Changes* button and PNCconf will update the I/O setup pages. Only I/O tabs will be shown for available connectors, depending on the Mesa board.

6.6 Mesa I/O Setup

The tabs are used to configure the input and output pins of the Mesa boards. PNCconf allows one to create custom signal names for use in custom HAL files.

Mesa0 Configuration-Board: 5i20 firmware: SVST8_4

Configuration Page		I/O Connector 2	I/O Connector 3	I/O Connector 4			
Num	function	Pin Type	Inv	Num	function	Pin Type	Inv
1:	X Encoder	Quad Encoder-B	<input type="checkbox"/>	3:	Multi Hand Wheel	Quad Encoder-B	<input type="checkbox"/>
0:	X Encoder	Quad Encoder-A	<input type="checkbox"/>	2:	Unused Encoder	Quad Encoder-B	<input type="checkbox"/>
1:	Spindle Encoder	Quad Encoder-B	<input type="checkbox"/>	3:	Multi Hand Wheel	Quad Encoder-A	<input type="checkbox"/>
0:	Spindle Encoder	Quad Encoder-A	<input type="checkbox"/>	2:	Unused Encoder	Quad Encoder-A	<input type="checkbox"/>
	X Encoder	Quad Encoder-I	<input type="checkbox"/>		Multi Hand Wheel	Quad Encoder-I	<input type="checkbox"/>
	Spindle Encoder	Quad Encoder-I	<input type="checkbox"/>		Unused Encoder	Quad Encoder-I	<input type="checkbox"/>
1:	X Axis PWM	Pulse Width Gen-P	<input type="checkbox"/>	3:	Unused PWM Gen	Pulse Width Gen-P	<input type="checkbox"/>
0:	Spindle PWM	Pulse Width Gen-P	<input type="checkbox"/>	2:	Unused PWM Gen	Pulse Width Gen-P	<input type="checkbox"/>
	X Axis PWM	Pulse Width Gen-D	<input type="checkbox"/>		Unused PWM Gen	Pulse Width Gen-D	<input type="checkbox"/>
	Spindle PWM	Pulse Width Gen-D	<input type="checkbox"/>		Unused PWM Gen	Pulse Width Gen-D	<input type="checkbox"/>
1:	X Axis PWM	Pulse Width Gen-E	<input type="checkbox"/>	3:	Unused PWM Gen	Pulse Width Gen-E	<input type="checkbox"/>
0:	Spindle PWM	Pulse Width Gen-E	<input type="checkbox"/>	2:	Unused PWM Gen	Pulse Width Gen-E	<input type="checkbox"/>

[Launch test panel](#)

[Help](#)

[Cancel](#) [Back](#) [Forward](#)

Figure 6.6: Mesa I/O C2

On this tab with this firmware the components are setup for a 7i33 daughter board, usually used with closed loop servos. Note the component numbers of the encoder counters and PWM drivers are not in numerical order. This follows the daughter board requirements.

Mesa0 Configuration-Board: 5i20 firmware: SVST8_4

Configuration Page	I/O Connector 2	I/O Connector 3	I/O Connector 4	
Num	function	Pin Type	Inv	
024:	X Minimum Limit + Hom	GPIO Input	<input type="checkbox"/>	
025:	X Maximum Limit	GPIO Input	<input type="checkbox"/>	
026:	Unused Input	GPIO Input	<input type="checkbox"/>	
027:	Unused Input	GPIO Input	<input type="checkbox"/>	
028:	Limits	>	GPIO Input	<input type="checkbox"/>
029:	Home	>	GPIO Input	<input type="checkbox"/>
030:	Limits/Home Shared	>	GPIO Input	<input type="checkbox"/>
031:	Digital	>	GPIO Input	<input type="checkbox"/>
032:	Axis Selection	>	GPIO Input	<input type="checkbox"/>
033:	Overrides	>	GPIO Input	<input type="checkbox"/>
034:	Spindle	>	GPIO Input	<input type="checkbox"/>
035:	Operation	>	GPIO Input	<input type="checkbox"/>
	External Control	>	GPIO Input	<input type="checkbox"/>
	Axis rapid	>		
	X BLDC Control	>		
	Y BLDC Control	>		
	Z BLDC Control	>		
	A BLDC Control	>		
	S BLDC Control	>		
	Custom Signals			
Launch test panel				
Help				
				Cancel Back Forward

Figure 6.7: Mesa I/O C3

On this tab all the pins are GPIO. Note the 3 digit numbers - they will match the HAL pin number. GPIO pins can be selected as input or output and can be inverted.

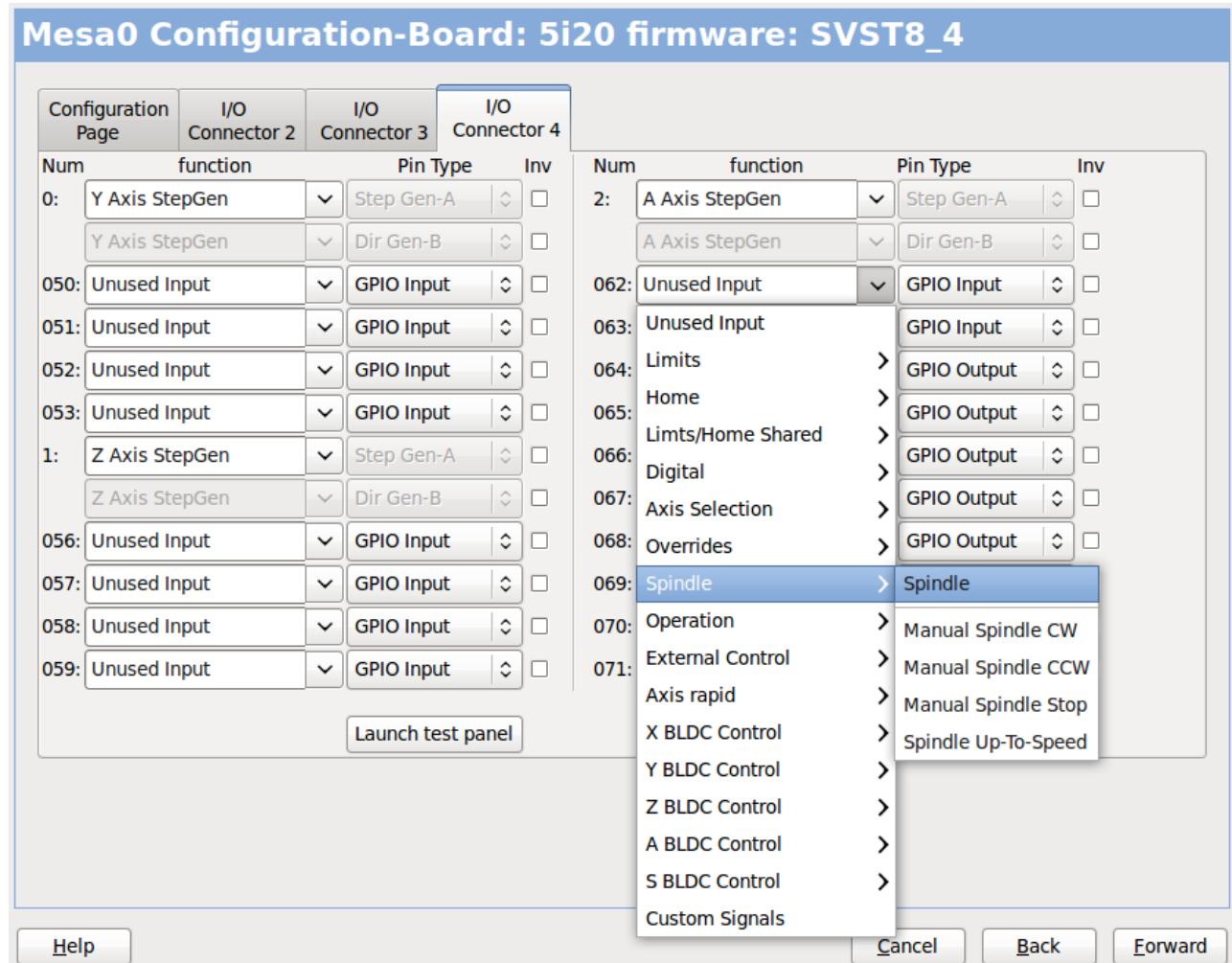


Figure 6.8: Mesa I/O C4

On this tab there are a mix of step generators and GPIO. Step generators output and direction pins can be inverted. Note that inverting a Step Gen-A pin (the step output pin) changes the step timing. It should match what your controller expects.

6.7 Parport configuration

First Parallel Port set for OUTPUT

Outputs (PC to Machine):	Invert	Inputs (Machine to PC):	Invert
Pin 1: Digital out 0	<input type="checkbox"/>	Pin 2: Unused Input	<input type="checkbox"/>
Pin 2: Machine Is Enabled	<input type="checkbox"/>	Pin 3: Unused Input	<input type="checkbox"/>
Pin 3: X Amplifier Enable	<input type="checkbox"/>	Pin 4: Unused Input	<input type="checkbox"/>
Pin 4: Z Amplifier Enable	<input type="checkbox"/>	Pin 5: Unused Input	<input type="checkbox"/>
Pin 5: Unused Output	<input type="checkbox"/>	Pin 6: Unused Input	<input type="checkbox"/>
Pin 6: Unused Output	<input type="checkbox"/>	Pin 7: Unused Input	<input type="checkbox"/>
Pin 7: Unused Output	<input type="checkbox"/>	Pin 8: Unused Input	<input type="checkbox"/>
Pin 8: Unused Output	<input type="checkbox"/>	Pin 9: Unused Input	<input type="checkbox"/>
Pin 9: Unused Output	<input type="checkbox"/>	Pin 10: Digital in 0	<input type="checkbox"/>
Pin 14: Unused Output	<input type="checkbox"/>	Pin 11: Unused Input	<input type="checkbox"/>
Pin 16: Unused Output	<input type="checkbox"/>	Pin 12: Unused Input	<input type="checkbox"/>
Pin 17: Unused Output	<input type="checkbox"/>	Pin 13: Unused Input	<input type="checkbox"/>
		Pin 15: Unused Input	<input type="checkbox"/>

Launch Test Panel

Help **Cancel** **Back** **Forward**

The parallel port can be used for simple I/O similar to Mesa's GPIO pins.

6.8 Axis Configuration

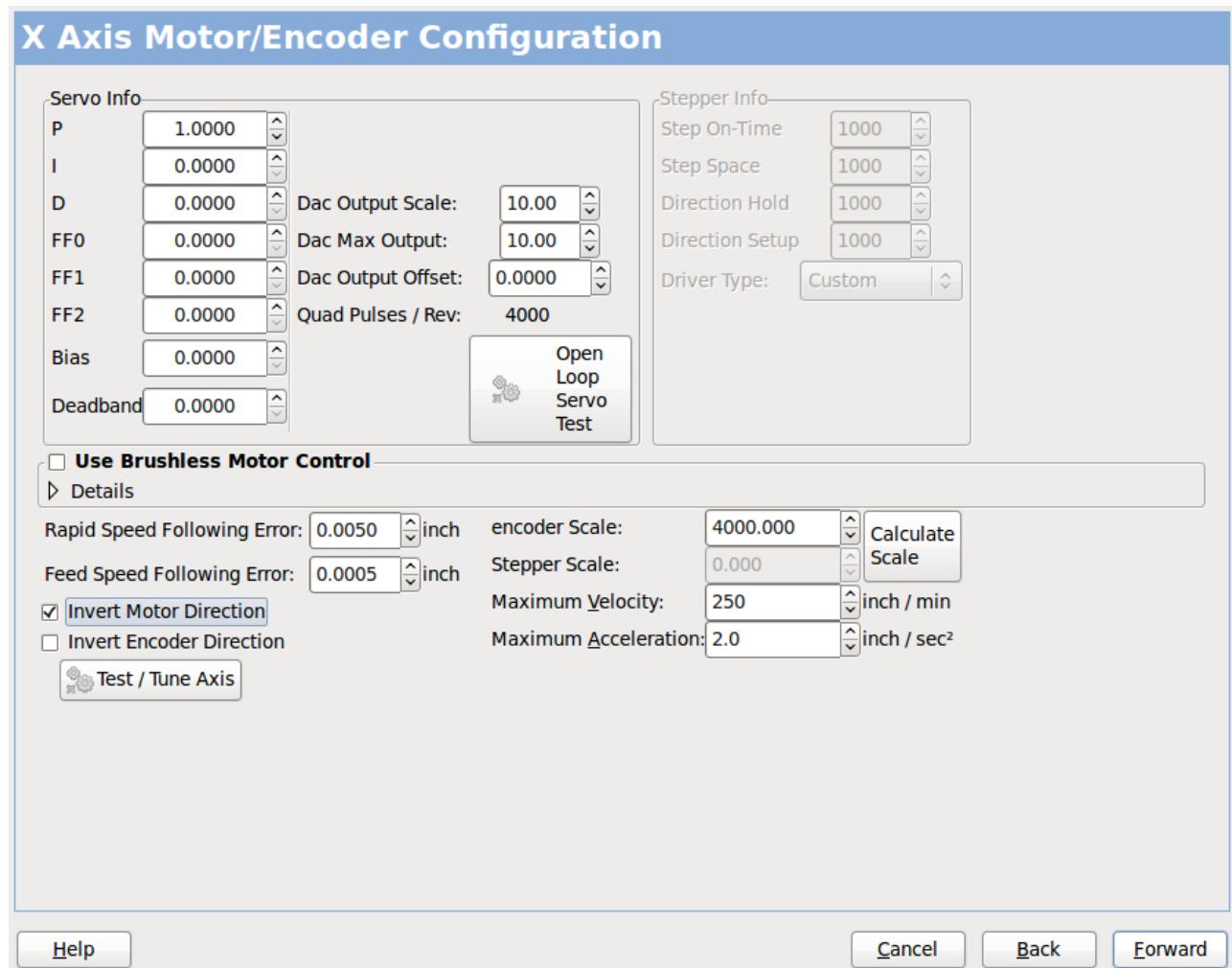


Figure 6.9: Axis Drive Configuration

This page allows configuring and testing of the motor and/or encoder combination . If using a servo motor an open loop test is available, if using a stepper a tuning test is available.

Open Loop Test

An open loop test is important as it confirms the direction of the motor and encoder. The motor should move the axis in the positive direction when the positive button is pushed and also the encoder should count in the postive direction. The axis movement should follow the Machinery's Handbook ¹ standards or AXIS graphical display will not make much sense. Hopefully the help page and diagrams can help figure this out. Note that axis directions are based on TOOL movement not table movement. There is no acceleration ramping with the open loop test so start with lower DAC numbers. By moving the axis a known distance one can confirm the encoder scaling. The encoder should count even without the amp enabled depending on how power is supplied to the encoder.

¹ "axis nomenclature" in the chapter "Numerical Control" in the "Machinery's Handbook" published by Industrial Press.

**Warning**

If the motor and encoder do not agree on counting direction then the servo will run away when using PID control.

Since at the moment PID settings can not be tested in PNCconf the settings are really for when you re-edit a config - enter your tested PID settings.

DAC scaling, max output and offset are used to tailor the DAC output.

Compute DAC

These two values are the scale and offset factors for the axis output to the motor amplifiers. The second value (offset) is subtracted from the computed output (in volts), and divided by the first value (scale factor), before being written to the D/A converters. The units on the scale value are in true volts per DAC output volts. The units on the offset value are in volts. These can be used to linearize a DAC.

Specifically, when writing outputs, the LinuxCNC first converts the desired output in quasi-SI units to raw actuator values, e.g., volts for an amplifier DAC. This scaling looks like: The value for scale can be obtained analytically by doing a unit analysis, i.e., units are [output SI units]/[actuator units]. For example, on a machine with a velocity mode amplifier such that 1 volt results in 250 mm/sec velocity, Note that the units of the offset are in machine units, e.g., mm/sec, and they are pre-subtracted from the sensor readings. The value for this offset is obtained by finding the value of your output which yields 0.0 for the actuator output. If the DAC is linearized, this offset is normally 0.0.

The scale and offset can be used to linearize the DAC as well, resulting in values that reflect the combined effects of amplifier gain, DAC non-linearity, DAC units, etc. To do this, follow this procedure:

- Build a calibration table for the output, driving the DAC with a desired voltage and measuring the result:

Table 6.1: Output Voltage Measurements

Raw	Measured
-10	-9.93
-9	-8.83
0	-0.96
1	-0.03
9	9.87
10	10.07

- Do a least-squares linear fit to get coefficients a, b such that $\text{meas} = a * \text{raw} + b$
- Note that we want raw output such that our measured result is identical to the commanded output. This means
 - $\text{cmd} = a * \text{raw} + b$
 - $\text{raw} = (\text{cmd} - b) / a$
- As a result, the a and b coefficients from the linear fit can be used as the scale and offset for the controller directly.

MAX OUTPUT: The maximum value for the output of the PID compensation that is written to the motor amplifier, in volts. The computed output value is clamped to this limit. The limit is applied before scaling to raw output units. The value is applied symmetrically to both the plus and the minus side.

Tuning Test The tuning test unfortunately only works with stepper based systems. Again confirm the directions on the axis is correct. Then test the system by running the axis back and forth, If the acceleration or max speed is too high you will lose steps. While jogging, Keep in mind it can take a while for an axis with low acceleration to stop. Limit switches are not functional during this test. You can set a pause time so each end of the test movement. This would allow you to set up and read a dial indicator to see if you are loosing steps.

Stepper Timing Stepper timing needs to be tailored to the step controller's requirements. Pncconf supplies some default controller timing or allows custom timing settings . See http://wiki.linuxcnc.org/cgi-bin/wiki.pl?Stepper_Drive_Timing for some more known timing numbers (feel free to add ones you have figured out). If in doubt use large numbers such as 5000 this will only limit max speed.

Brushless Motor Control These options are used to allow low level control of brushless motors using special firmware and daughter boards. It also allows conversion of HALL sensors from one manufacturer to another. It is only partially supported and will require one to finish the HAL connections. Contact the mail-list or forum for more help.

Step Motor Scale

<input checked="" type="checkbox"/> Pulley teeth (motor:Leadscrew):	1 : 2
<input type="checkbox"/> Worm turn ratio (Input:Output)	1 : 1
<input checked="" type="checkbox"/> Microstep Multiplication Factor:	5
<input type="checkbox"/> Leadscrew Metric Pitch	5.0000 mm / rev
<input checked="" type="checkbox"/> Leadscrew TPI	5.0000 TPI
Motor steps per revolution:	200

Encoder Scale

<input type="checkbox"/> Pulley teeth (encoder:Leadscrew):	1 : 1
<input type="checkbox"/> Worm turn ratio (Input:Output)	1 : 1
<input type="checkbox"/> Leadscrew Metric Pitch	5.0000 mm / rev
<input type="checkbox"/> Leadscrew TPI	5.0000 TPI
Encoder lines per revolution:	1000 X 4 = Pulses/Rev

Calculated Scale

motor steps per unit:	10000.0000
encoder pulses per unit:	4000.0000

Motion Data

Calculated Axis SCALE:	10000.0 Steps / inch
Resolution:	0.0001000 inch / Step
Time to accelerate to max speed:	0.8335 sec
Distance to achieve max speed:	0.6947 inch
Pulse rate at max speed:	16.7 KHz
Motor RPM at max speed:	1000 RPM

Figure 6.10: Axis Scale Calculation

The scale settings can be directly entered or one can use the *calculate scale* button to assist. Use the check boxes to select appropriate calculations. Note that *pulley teeth* requires the number of teeth not the gear ratio. Worm turn ratio is just the opposite it requires the gear ratio. If you are happy with the scale press apply otherwise push cancel and enter the scale directly.

X Axis Configuration

Positive Travel Distance (Machine zero Origin to end of + travel):

Negative Travel Distance (Machine zero Origin to end of - travel):

Home Position location (offset from machine zero Origin):

Home Switch location (Offset from machine zero Origin):

Home Search Velocity:	<input type="text" value="3"/>	inch / min
Home Search Direction:	<input type="text" value="Towards Negative limit"/>	
Home latch Velocity:	<input type="text" value="1"/>	inch / min
Home Latch Direction:	<input type="text" value="Same"/>	
Home Final Velocity:	<input type="text" value="0"/>	inch / min
Use Encoder Index For Home:	<input type="text" value="NO"/>	

Use Compensation File: filename:

Use Backlash Compensation:

[Help](#) [Cancel](#) [Back](#) [Forward](#)

Figure 6.11: Axis Configuration

Also refer to the diagram tab for two examples of home and limit switches. These are two examples of many different ways to set homing and limits.


Important

It is very important to start with the axis moving in the right direction or else getting homing right is very difficult!

Remember positive and negative directions refer to the TOOL not the table as per the Machinists handbook.

ON A TYPICAL KNEE OR BED MILL

- when the TABLE moves out that is the positive Y direction
- when the TABLE moves left that is the positive X direction
- when the TABLE moves down that is the positive Z direction
- when the HEAD moves up that is the positive Z direction

ON A TYPICAL LATHE

- when the TOOL moves right, away from the chuck
- that is the positive Z direction
- when the TOOL moves toward the operator
- that is the positive X direction. Some lathes have X
- opposite (eg tool on back side), that will work fine but
- AXIS graphical display can not be made to reflect this.

When using homing and / or limit switches LinuxCNC expects the HAL signals to be true when the switch is being pressed / tripped. If the signal is wrong for a limit switch then LinuxCNC will think the machine is on end of limit all the time. If the home switch search logic is wrong LinuxCNC will seem to home in the wrong direction. What it actually is doing is trying to BACK off the home switch.

Decide on limit switch location.

Limit switches are the back up for software limits in case something electrical goes wrong eg. servo runaway. Limit switches should be placed so that the machine does not hit the physical end of the axis movement. Remember the axis will coast past the contact point if moving fast. Limit switches should be *active low* on the machine. eg. power runs through the switches all the time - a loss of power (open switch) trips. While one could wire them the other way, this is fail safe. This may need to be inverted so that the HAL signal in LinuxCNC in *active high* - a TRUE means the switch was tripped. When starting LinuxCNC if you get an on-limit warning, and axis is NOT tripping the switch, inverting the signal is probably the solution. (use HALMETER to check the corresponding HAL signal eg. axis.0.pos-lim-sw-in X axis positive limit switch)

Decide on the home switch location.

If you are using limit switches You may as well use one as a home switch. A separate home switch is useful if you have a long axis that in use is usually a long way from the limit switches or moving the axis to the ends presents problems of interference with material. eg a long shaft in a lathe makes it hard to home to limits with out the tool hitting the shaft, so a separate home switch closer to the middle may be better. If you have an encoder with index then the home switch acts as a coarse home and the index will be the actual home location.

Decide on the MACHINE ORIGIN position.

MACHINE ORIGIN is what LinuxCNC uses to reference all user coordinate systems from. I can think of little reason it would need to be in any particular spot. There are only a few G codes that can access the MACHINE COORDINATE system.(G53, G30 and G28) If using tool-change-at-G30 option having the Origin at the tool change position may be convenient. By convention, it may be easiest to have the ORIGIN at the home switch.

Decide on the (final) HOME POSITION.

this just places the carriage at a consistent and convenient position after LinuxCNC figures out where the ORIGIN is.

Measure / calculate the positive / negative axis travel distances.

Move the axis to the origin. Mark a reference on the movable slide and the non-moveable support (so they are in line) move the machine to the end of limits. Measure between the marks that is one of the travel distances. Move the table to the other end of travel. Measure the marks again. That is the other travel distance. If the ORIGIN is at one of the limits then that travel distance will be zero.

(machine) ORIGIN

The Origin is the MACHINE zero point. (not the zero point you set your cutter / material at). LinuxCNC uses this point to reference everything else from. It should be inside the software limits. LinuxCNC uses the home switch location to calculate the origin position (when using home switches or must be manually set if not using home switches).

Travel distance

This is the maximum distance the axis can travel in each direction. This may or may not be able to be measured directly from origin to limit switch. The positive and negative travel distances should add up to the total travel distance.

POSITIVE TRAVEL DISTANCE

This is the distance the Axis travels from the Origin to the positive travel distance or the total travel minus the negative travel distance. You would set this to zero if the origin is positioned at the positive limit. This will always be zero or a positive number.

NEGATIVE TRAVEL DISTANCE

This is the distance the Axis travels from the Origin to the negative travel distance, or the total travel minus the positive travel distance. You would set this to zero if the origin is positioned at the negative limit. This will always be zero or a negative number. If you forget to make this negative PNCconf will do it internally.

(Final) HOME POSITION

This is the position the home sequence will finish at. It is referenced from the Origin so can be negative or positive depending on what side of the Origin it is located. When at the (final) home position if you must move in the Positive direction to get to the Origin, then the number will be negative.

HOME SWITCH LOCATION

This is the distance from the home switch to the Origin. It could be negative or positive depending on what side of the Origin it is located. When at the home switch location if you must move in the Positive direction to get to the Origin, then the number will be negative. If you set this to zero then the Origin will be at the location of the limit switch (plus distance to find index if used)

Home Search Velocity

Course home search velocity in units per minute.

Home Search Direction

Sets the home switch search direction either negative (ie. towards negative limit switch) or positive (ie. towards positive limit switch)

Home Latch Velocity

Fine Home search velocity in units per minute

Home Final Velocity

Velocity used from latch position to (final) home position in units per minute. Set to 0 for max rapid speed

Home latch Direction

Allows setting of the latch direction to the same or opposite of the search direction.

Use Encoder Index For Home

LinuxCNC will search for an encoder index pulse while in the latch stage of homing.

Use Compensation File

Allows specifying a Comp filename and type. Allows sophisticated compensation. See Manual.

Use Backlash Compensation

Allows setting of simple backlash compensation. Can not be used with Compensation File. See Manual.

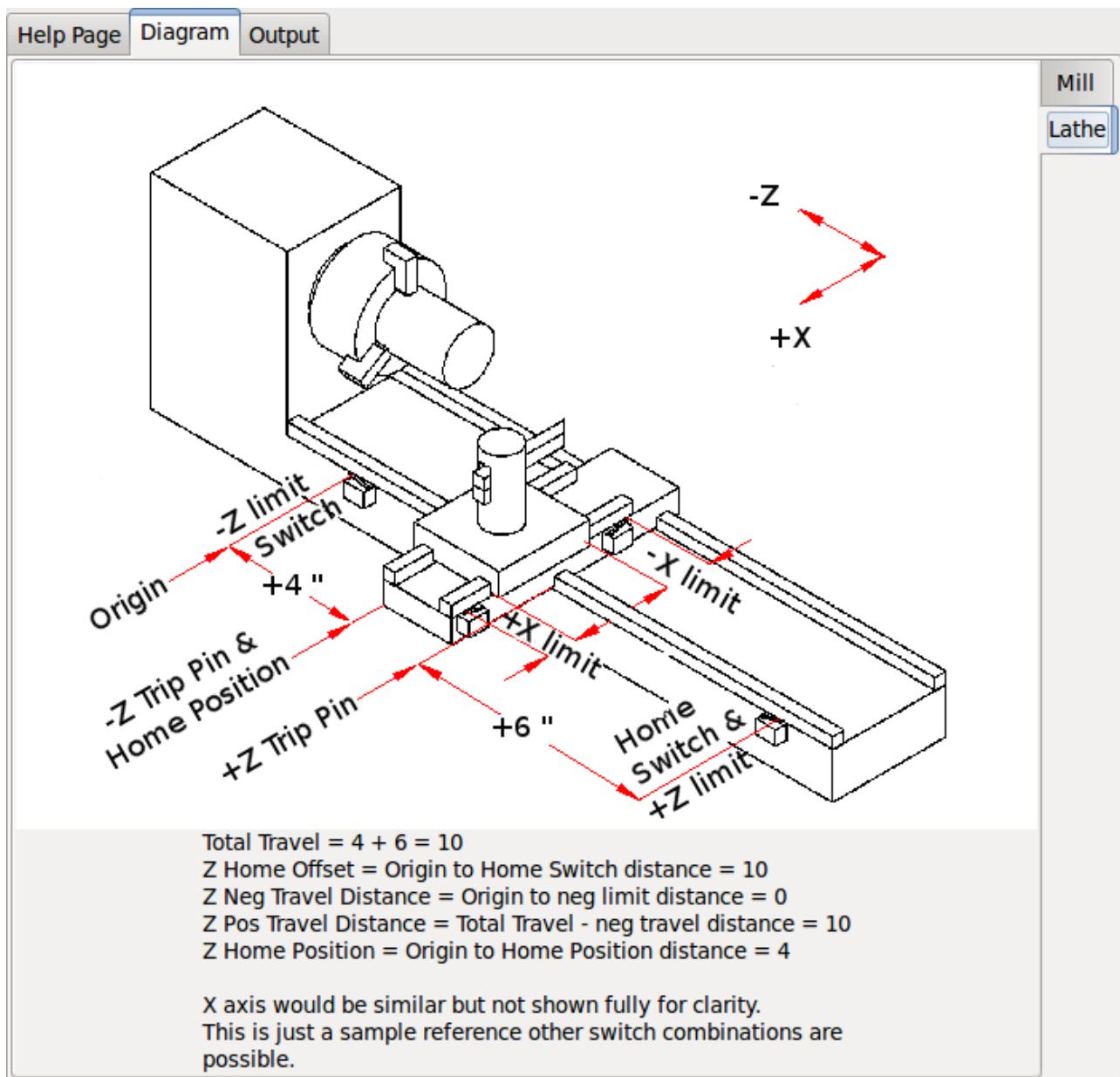


Figure 6.12: AXIS Help Diagram

The diagrams should help to demonstrate an example of limit switches and standard axis movement directions. In this example the Z axis has two limit switches, the positive switch is shared as a home switch. The MACHINE ORIGIN (zero point) is located at the negative limit. The left edge of the carriage is the negative trip pin and the right the positive trip pin. We wish the FINAL HOME POSITION to be 4 inches away from the ORIGIN on the positive side. If the carriage was moved to the positive limit we would measure 10 inches between the negative limit and the negative trip pin.

6.9 Spindle Configuration

If you select spindle signals then this page is available to configure spindle control.

Tip

Many of the option on this page will not show unless the proper option was selected on previous pages!

Spindle Motor/Encoder Configuration

Servo Info	Stepper Info
P 1.0000	Step On-Time 1000
I 0.0000	Step Space 1000
D 0.0000	Direction Hold 1000
FF0 0.0000	Direction Setup 1000
FF1 0.0000	Driver Type: Custom
FF2 0.0000	
Bias 0.0000	
Deadband 0.0000	
Dac Output Scale: 10.00 Dac Max Output: 10.00 Dac Output Offset: 0.0000 Quad Pulses / Rev: 4000 <input type="button" value="Open Loop Servo Test"/>	
<input type="checkbox"/> Use Brushless Motor Control <input type="button" value="Details"/>	
<input type="checkbox"/> Use Spindle-At-Speed Scale: 95 %	
Rapid Speed Following Error: 0.0000 rev	encoder Scale: 4000.000
Feed Speed Following Error: 0.0000 rev	Stepper Scale: 0.000
<input type="checkbox"/> Invert Motor Direction	Maximum Velocity: 100 rev / min
<input type="checkbox"/> Invert Encoder Direction	Maximum Acceleration: 2.0 rev / sec ²
<input type="button" value="Test / Tune Axis"/>	

[Help](#)[Cancel](#)[Back](#)[Forward](#)

Figure 6.13: Spindle Configuration

This page is similar to the axis motor configuration page.

There are some differences:

- Unless one has chosen a stepper driven spindle there is no acceleration or velocity limiting.
- There is no support for gear changes or ranges.
- If you picked a VCP spindle display option then spindle-at-speed scale and filter settings may be shown.
- Spindle-at-speed allows LinuxCNC to wait till the spindle is at the requested speed before moving the axis. This is particularly handy on lathes with constant surface feed and large speed diameter changes. It requires either encoder feedback or a digital spindle-at-speed signal typically connected to a VFD drive.
- If using encoder feedback, you may select a spindle-at-speed scale setting that specifies how close the actual speed must be to the requested speed to be considered at-speed.

- If using encoder feedback, the VCP speed display can be erratic - the filter setting can be used to smooth out the display. The encoder scale must be set for the encoder count / gearing used.
- If you are using a single input for a spindle encoder you must add the line: `setp hm2_7i43.0.encoder.00.counter-mode 1` (changing the board name and encoder number to your requirements) into a custom HAL file. See the Hostmot2 section on encoders for more info about counter mode.

6.10 Advanced Options

This allows setting of HALUI commands and loading of classicladder and sample ladder programs. If you selected GLADE VCP options such as for zeroing axis, there will be commands showing. See the manual about info on HALUI for using custom halemds. There are several ladder program options. The Estop program allows an external ESTOP switch or the GUI frontend to throw an Estop. It also has a timed lube pump signal. The Z auto touch-off is with a touch-off plate, the GLADE VCP touch-off button and special HALUI commands to set the current user origin to zero and rapid clear. The serial modbus program is basically a blank template program that sets up classicladder for serial modbus. See the classicladder section in the manual.

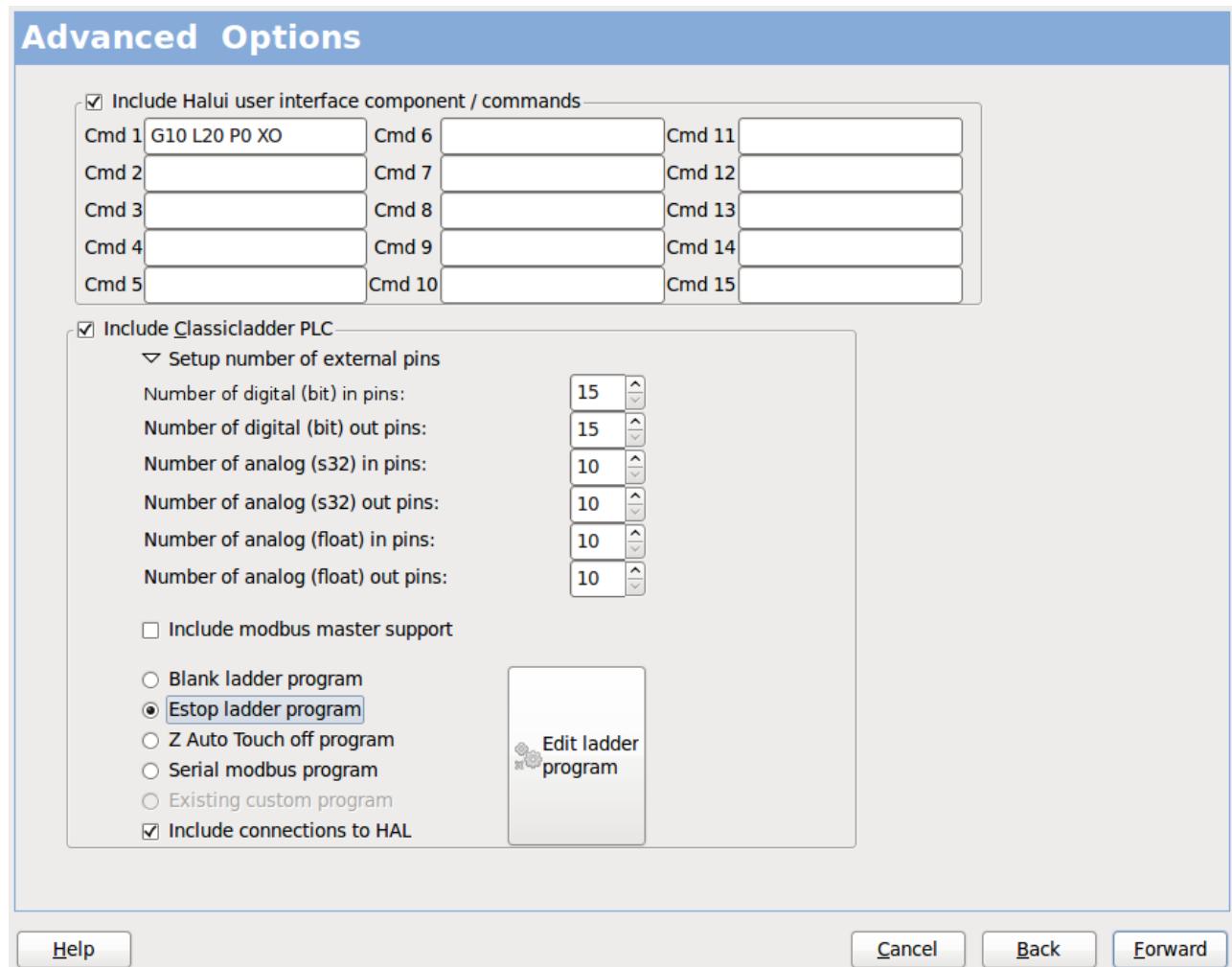


Figure 6.14: Advanced Options

6.11 HAL Components

On this page you can add additional HAL components you might need for custom HAL files. In this way one should not have to hand edit the main HAL file, while still allowing user needed components.

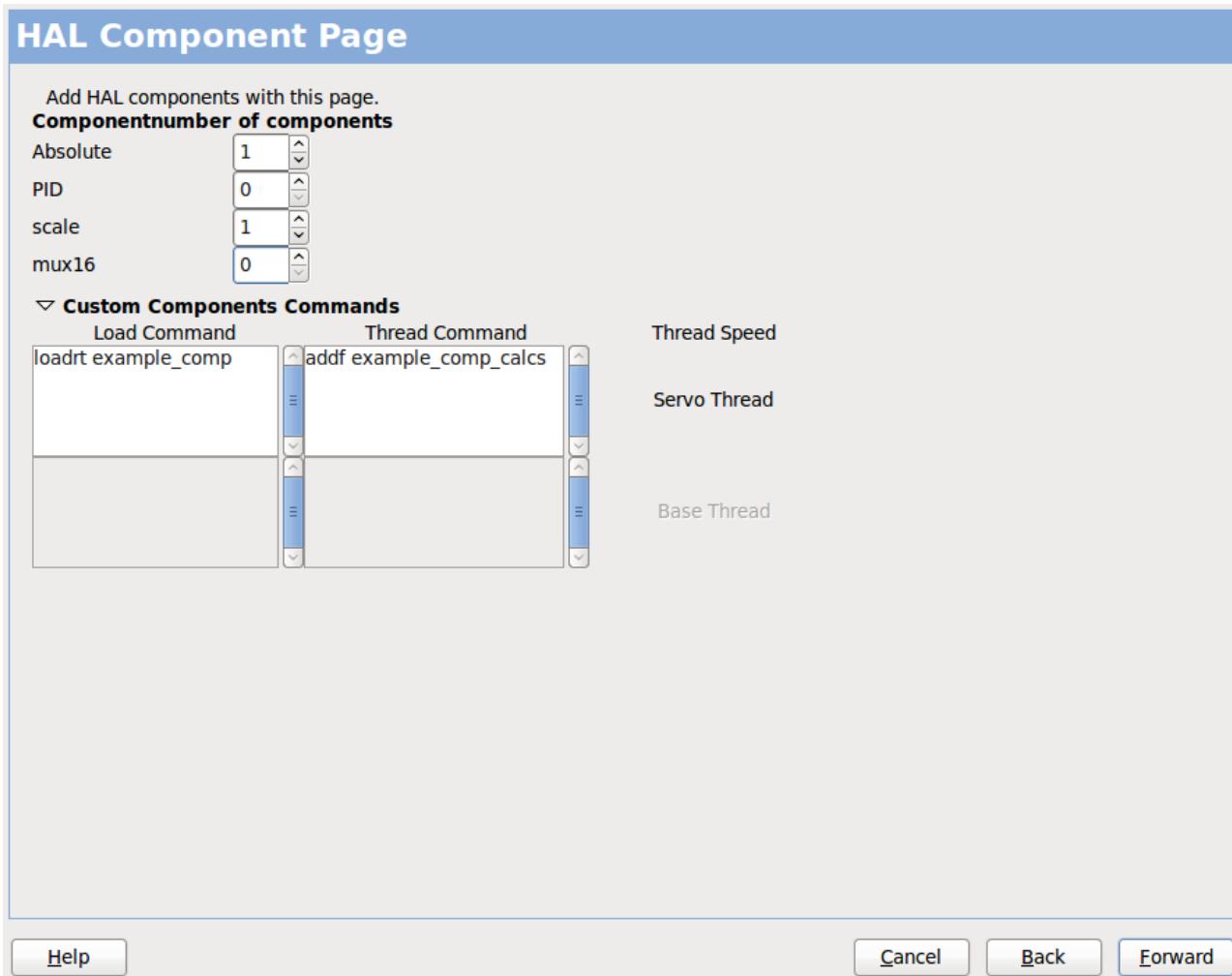


Figure 6.15: HAL Components

The first selection is components that pncconf uses internally. You may configure pncconf to load extra instances of the components for your custom HAL file.

Select the number of instances your custom file will need, pncconf will add what it needs after them.

Meaning if you need 2 and pncconf needs 1 pncconf will load 3 instances and use the last one.

Custom Component Commands

This selection will allow you to load HAL components that pncconf does not use. Add the loadrt or loadusr command, under the heading *loading command*. Add the addf command under the heading *Thread command*. The components will be added to the thread between reading of inputs and writing of outputs, in the order you write them in the *thread command*.

6.12 Advanced Usage Of PNCconf

PNCconf does its best to allow flexible customization by the user. PNCconf has support for custom signal names, custom loading of components, custom HAL files and custom firmware.

There are also signal names that PNCconf always provides regardless of options selected, for user's custom HAL files. With some thought most customizations should work regardless if you later select different options in PNCconf.

Eventually if the customizations are beyond the scope of PNCconf's frame work you can use PNCconf to build a base config or use one of LinuxCNC's sample configurations and just hand edit it to what ever you want.

Custom Signal Names

If you wish to connect a component to something in a custom HAL file write a unique signal name in the combo entry box. Certain components will add endings to your custom signal name:

Encoders will add <customname> +:

- position
- count
- velocity
- index-enable
- reset

Steppers add:

- enable
- counts
- position-cmd
- position-fb
- velocity-fb

PWM add:

- enable
- value

GPIO pins will just have the entered signal name connected to it

In this way one can connect to these signals in the custom HAL files and still have the option to move them around later.

Custom Signal Names

The Hal Components page can be used to load components needed by a user for customization.

Loading Custom Firmware

PNCconf searches for firmware on the system and then looks for the XML file that it can convert to what it understands. These XML files are only supplied for officially released firmware from the LinuxCNC team. To utilize custom firmware one must convert it to an array that PNCconf understands and add its filepath to PNCconf's preference file. By default this path searches the desktop for a folder named custom_firmware and a file named firmware.py.

The hidden preference file is in the user's home file, is named .pncconf-preferences and require one to select *show hidden files* to see and edit it. The contents of this file can be seen when you first load PNCconf - press the help button and look at the output page.

Ask on the LinuxCNC mail-list or forum for info about converting custom firmware. Not all firmware can be utilized with PNCconf.

Custom HAL Files

There are four custom files that you can use to add HAL commands to:

- custom.hal is for HAL commands that don't have to be run after the GUI frontend loads. It is run after the configuration-named HAL file.
- custom_postgui.hal is for commands that must be run after AXIS loads or a standalone PYVCP display loads.
- custom_gvcp.hal is for commands that must be run after glade VCP is loaded.
- shutdown.hal is for commands to run when LinuxCNC shuts down in a controlled manner.

Chapter 7

Running LinuxCNC

7.1 Invoking LinuxCNC

After installation, LinuxCNC starts just like any other Linux program: run it from the terminal by issuing the command *emc*, or select it in the Applications - CNC menu.

7.2 Configuration Selector

By default, the Configuration Selector dialog is shown when you first run LinuxCNC. Your own personalized configurations are shown at the top of the list, followed by sample configurations. Because each sample configuration is for a different type of hardware interface, almost all will not run without the hardware installed. The configurations listed under the category *sim* run entirely without attached hardware.

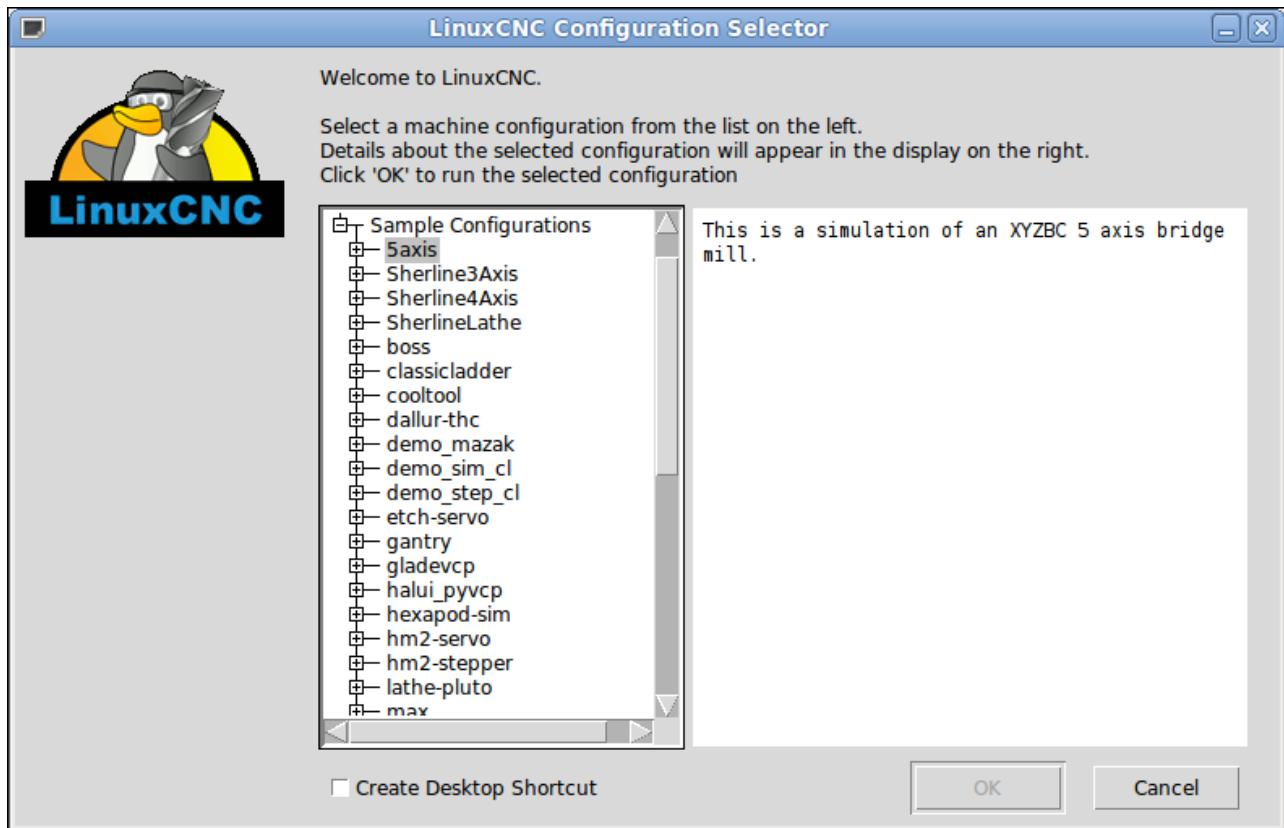


Figure 7.1: LinuxCNC Configuration Selector

Click any of the listed configurations to display specific information about it. Double-click a configuration or click OK to start the configuration. Select *Create Desktop Shortcut* and then click OK to add an icon on the Ubuntu desktop to directly launch this configuration without showing the Configuration Selector screen.

When you select a configuration from the Sample Configurations section, it will automatically place a copy of that config in the emc/configs directory.

7.3 Next steps in configuration

After finding the sample configuration that uses the same interface hardware as your machine, and saving a copy to your home directory, you can customize it according to the details of your machine. Refer to the Integrator Manual for topics on configuration.

Chapter 8

Linux FAQ

These are some basic Linux commands and techniques for new to Linux users. More complete information can be found on the web or by using the man pages.

8.1 Automatic Login

When you install LinuxCNC with the Ubuntu LiveCD the default is to have to log in each time you turn the computer on. To enable automatic login go to *System > Administration > Login Window*. If it is a fresh install the Login Window might take a second or three to pop up. You will have to have your password that you used for the install to gain access to the Login Window Preferences window. In the Security tab check off Enable Automatic Login and pick a user name from the list (that would be you).

8.2 Automatic Startup

To have LinuxCNC start automatically with your config after turning on the computer go to *System > Preferences > Sessions > Startup Applications*, click Add. Browse to your config and select the .ini file. When the file picker dialog closes, add emc and a space in front of the path to your .ini file.

Example:

```
emc /home/mill/emc2/config/mill/mill.ini
```

8.3 Man Pages

Man pages are automatically generated manual pages in most cases. Man pages are usually available for most programs and commands in Linux.

To view a man page open up a terminal window by going to *Applications > Accessories > Terminal*. For example if you wanted to find out something about the find command in the terminal window type:

```
man find
```

Use the Page Up and Page Down keys to view the man page and the Q key to quit viewing.

8.4 List Modules

Sometimes when troubleshooting you need to get a list of modules that are loaded. In a terminal window type:

```
lsmod
```

If you want to send the output from lsmod to a text file in a terminal window type:

```
lsmod > mymod.txt
```

The resulting text file will be located in the home directory if you did not change directories when you opened up the terminal window and it will be named mymod.txt or what ever you named it.

8.5 Editing a Root File

When you open the file browser and you see the Owner of the file is root you must do extra steps to edit that file. Editing some root files can have bad results. Be careful when editing root files. Generally, you can open and view most root files, but they will open in *read only* mode.

8.5.1 The Command Line Way

Open up *Applications > Accessories > Terminal*.

In the terminal window type

```
sudo gedit
```

Open the file with *File > Open > Edit*

8.5.2 The GUI Way

1. Right click on the desktop and select Create Launcher
2. Type a name in like sudo edit
3. Type *gksudo "gnome-open %u"* as the command and save the launcher to your desktop
4. Drag a file onto your launcher to open and edit

8.5.3 Root Access

In Ubuntu you can become root by typing in "sudo -i" in a terminal window then typing in your password. Be careful, because you can really foul things up as root if you don't know what you're doing.

8.6 Terminal Commands

8.6.1 Working Directory

To find out the path to the present working directory in the terminal window type:

```
pwd
```

8.6.2 Changing Directories

To move up one level in the terminal window type:

```
cd ..
```

To move up two levels in the terminal window type:

```
cd .../..
```

To move down to the emc2/configs subdirectory in the terminal window type:

```
cd emc2/configs
```

8.6.3 Listing files in a directory

To view a list of all the files and subdirectories in the terminal window type:

```
dir
```

or

```
ls
```

8.6.4 Finding a File

The find command can be a bit confusing to a new Linux user. The basic syntax is:

```
find starting-directory parameters actions
```

For example to find all the .ini files in your emc2 directory you first need to use the pwd command to find out the directory. Open a new terminal window and type:

```
pwd
```

And pwd might return the following result:

```
/home/joe
```

With this information put the command together like this:

```
find /home/joe/linuxcnc -name \*.ini -print
```

The -name is the name of the file your looking for and the -print tells it to print out the result to the terminal window. The *.ini tells find to return all files that have the .ini extension. The backslash is needed to escape the shell meta-characters. See the find man page for more information on find.

8.6.5 Searching for Text

```
grep -irl 'text to search for' *
```

This will find all the files that contain the *text to search for* in the current directory and all the subdirectories below it, while ignoring the case. The -i is for ignore case and the -r is for recursive (include all subdirectories in the search). The -l option will return a list of the file names, if you leave the -l off you will also get the text where each occourance of the "text to search for" is found. The * is a wild card for search all files. See the grep man page for more information.

8.6.6 Bootup Messages

To view the bootup messages use "dmesg" from the command window. To save the bootup messages to a file use the redirection operator, like this:

```
dmesg > bootmsg.txt
```

The contents of this file can be copied and pasted on line to share with people trying to help you diagnose your problem.

To clear the message buffer type this:

```
sudo dmesg -c
```

This can be helpful to do just before launching LinuxCNC, so that there will only be a record of information related to the current launch of LinuxCNC.

To find the built in parallel port address use grep to filter the information out of dmesg.

After boot up open a terminal and type:

```
dmesg | grep parport
```

8.7 Convenience Items

8.7.1 Terminal Launcher

If you want to add a terminal launcher to the panel bar on top of the screen you typically can right click on the panel at the top of the screen and select "Add to Panel". Select Custom Application Launcher and Add. Give it a name and put gnome-terminal in the command box.

8.8 Hardware Problems

8.8.1 Hardware Info

To find out what hardware is connected to your motherboard in a terminal window type:

```
lspci -v
```

8.8.2 Monitor Resolution

During installation Ubuntu attempts to detect the monitor settings. If this fails you are left with a generic monitor with a maximum resolution of 800x600.

Instructions for fixing this are located here:

<https://help.ubuntu.com/community/FixVideoResolutionHowto>

8.9 Paths

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.

```
./f0      is the same as f0, e.g., a file named f0 in the startup directory  
../f1     refers to a file f1 in the parent directory  
../../f2   refers to a file f2 in the parent of the parent directory  
...../f3 etc.
```

Chapter 9

Legal Section

9.1 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

9.2 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 10

Index

A

Advanced Configuration Options, [27](#)
Automatic Login, [60](#)
Automatic Startup, [60](#)
Axis Configuration, [22](#)
Axis Travel, [28](#)

C

`cd`, [62](#)
Changing Directories, [62](#)

D

Determining Spindle Calibration, [26](#)
`dir`, [62](#)

E

Editing a Root File, [61](#)

F

`find`, [62](#)
Finding a File, [62](#)
Finding Maximum Acceleration, [24](#)
Finding Maximum Velocity, [24](#)

G

`gksudo`, [61](#)
`grep`, [62](#)

H

Home and Limit Switch wiring options, [28](#)
Home Location, [28](#)
Home Switch Location, [28](#)

L

Latency Test, [19](#)
Lathe Threading, [26](#)
Linux FAQ, [60](#)
Listing files in a directory, [62](#)
`ls`, [62](#)

M

Man Pages, [60](#)
Mesa Configuration Wizard, [30](#)
Minimum Requirements, [2](#)

M

Multi-session Download, [5](#)
O
Operating without Home Switches, [28](#)
Operating without Limit Switches, [28](#)

P

Parallel Port Setup, [21](#)
Point n Click Configuration Wizard, [30](#)
`pwd`, [61](#)

R

Running LinuxCNC, [58](#)

S

Searching for Text, [62](#)
Sherline, [13](#)
Spindle Configuration, [25](#)
spindle speed control, [25](#)
Spindle-synchronized motion, [26](#)
Stepconf Wizard, [16](#)
Stepper Quickstart, [13](#)
`sudo gedit`, [61](#)
System Requirements, [2](#)

T

Terminal Commands, [61](#)

U

Updates to LinuxCNC, [7](#)

W

`Wget Linux`, [5](#)
`Wget Windows`, [5](#)
Working Directory, [61](#)

X

Xylotex, [13](#)

User Manual V2.6.8-5-g8676c92, 2015-05-14

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	13
2.5 Languages	15
2.6 Thinking Like a Machine Operator	15
2.7 Modes of Operation	15
3 Important User Concepts	17
3.1 Trajectory Control	17
3.1.1 Trajectory Planning	17
3.1.2 Path Following	17
3.1.3 Programming the Planner	17
3.1.4 Planning Moves	18
3.2 G Code	19
3.2.1 Defaults	19
3.2.2 Feed Rate	19
3.2.3 Tool Radius Offset	19
3.3 Homing	19
3.4 Tool Changes	19
3.5 Coordinate Systems	20
3.5.1 G53 Machine Coordinate	20
3.5.2 G54-59.3 User Coordinates	20
3.5.3 When You're Lost	20
3.6 Machine Configurations	20

II User Interfaces	22
4 CONFIGURATION SELECTOR	23
5 AXIS GUI	25
5.1 Introduction	25
5.2 Getting Started	26
5.2.1 A Typical Session	26
5.3 AXIS Display	27
5.3.1 Menu Items	27
5.3.2 Toolbar buttons	30
5.3.3 Graphical Display Area	31
5.3.4 Text Display Area	33
5.3.5 Manual Control	33
5.3.6 MDI	35
5.3.7 Feed Override	35
5.3.8 Spindle Speed Override	36
5.3.9 Jog Speed	36
5.3.10 Max Velocity	36
5.4 Keyboard Controls	36
5.5 Show LinuxCNC Status (linuxcnctop)	37
5.6 MDI interface	38
5.7 axis-remote	38
5.8 Manual Tool Change	38
5.9 Python modules	39
5.10 Using AXIS in Lathe Mode	39
5.11 Advanced Configuration	40
5.11.1 Program Filters	40
5.11.2 The X Resource Database	41
5.11.3 Physical jog wheels	41
5.11.4 ./axisrc	41
5.11.5 External Editor	41
5.11.6 Virtual Control Panel	41
5.11.7 Axis Preview Control	42

6 gmoccapy	43
6.1 Introduction	43
6.2 Requirements	44
6.3 How to get gmoccapy	44
6.4 Basic configuration	45
6.4.1 The DISPLAY Section	46
6.4.1.1 The configuration of tabs and side panels	47
6.4.1.2 Configuration of User Created Messages	53
6.4.2 The RS274NGC Section	54
6.4.3 The MACRO Section	54
6.4.4 The TRAJ Section	57
6.5 HAL Pins	57
6.5.1 Right and bottom button lists	57
6.5.2 Velocities and overrides	59
6.5.3 jog hal pins	60
6.5.4 jog velocities and turtle-jog hal pin	61
6.5.5 jog increment hal pins	61
6.5.6 hardware unlock pin	61
6.5.7 Error pins	62
6.5.8 User Created Message HAL Pins	62
6.5.9 Spindle feedback pins	62
6.5.10 Pins to indicate program progress information	63
6.5.11 Pins to modify soft limits	63
6.5.12 Tool related pin	63
6.5.12.1 Tool Change Pin	63
6.5.12.2 tool offset pins	64
6.6 Auto Tool Measurement	65
6.6.1 Tool measurement pins	66
6.6.2 Tool Measurement INI File modifications	66
6.6.2.1 The RS274NGC section	66
6.6.2.2 The tool sensor section	66
6.6.2.3 The Change position section	66
6.6.2.4 The Python section	67
6.6.3 Needed Files	67
6.6.4 Needed Hal connections	67
6.7 The settings page	68
6.7.1 Appearance	68
6.7.1.1 Main Window	69
6.7.1.2 Keyboard	69

6.7.1.3	On Touch Off	70
6.7.1.4	DRO Options	70
6.7.1.5	Preview	71
6.7.1.6	File to load on start up	71
6.7.1.7	Jump to dir	72
6.7.1.8	Themes and Sounds	72
6.7.2	Hardware	73
6.7.2.1	Hardware MPG Scales	73
6.7.2.2	Keyboard shortcuts	73
6.7.2.3	Unlock options	74
6.7.2.4	Spindle	74
6.7.2.5	Turtle Jog	74
6.7.3	Advanced Settings	75
6.7.3.1	Tool Measurement	75
6.7.3.2	Message behavior and appearance	76
6.7.3.3	Run from line option	77
6.7.3.4	Log Actions	77
6.8	LATHE specific section	77
6.9	Plasma specific section	80
6.10	VIDEO on you tube	80
6.10.1	Basic Usage	81
6.10.2	Simulated Jog Wheels	81
6.10.3	Settings Page	81
6.10.4	Simulated Hardware Button	81
6.10.5	User Tabs	81
6.10.6	Tool_Measurement_Video	81
6.11	Known problems	81
6.11.1	Strange numbers in the info area	81
7	NGCGUI	83
7.1	Overview	84
7.2	Demonstration Configurations	84
7.3	Library Locations	86
7.4	Standalone Usage	87
7.4.1	Standalone NGCGUI	87
7.4.2	Standalone PYNGCGUI	87
7.5	Embedding NGCGUI	88
7.5.1	Embedding NGCGUI in Axis	88
7.5.2	Embedding PYNGCGUI as a gladevcp tab page in a gui	88

7.5.3	Additional INI File items required for ngcgui or pyngcgui	89
7.5.4	Truetype Tracer	90
7.5.5	INI File Path Specifications	90
7.5.6	Summary of INI File item details for NGCGUI usage	91
7.6	File Requirements for NGCGUI Compatibility	93
7.6.1	Single-File Gcode (.ngc) Subroutine Requirements	93
7.6.2	Gcode-meta-compiler (.gcmc) file requirements	95
7.7	DB25 Example	97
8	Touchy GUI	100
8.1	Panel Configuration	101
8.1.1	HAL connections	101
8.1.1.1	Required controls	101
8.1.1.2	Optional controls	101
8.1.1.3	Optional panel lamps	101
8.1.2	Recommended for any setup	101
8.2	Setup	101
8.2.1	Enabling Touchy	101
8.2.2	Preferences	102
8.2.3	Macros	102
9	TkLinuxCNC GUI	103
9.1	Introduction	103
9.2	Getting Started	104
9.2.1	A typical session with TkLinuxCNC	104
9.3	Elements of the TkLinuxCNC window	104
9.3.1	Main buttons	105
9.3.2	Offset display status bar	105
9.3.3	Coordinate Display Area	105
9.3.3.1	Backplot	105
9.3.4	Automatic control	106
9.3.4.1	Buttons for control	106
9.3.4.2	Text Program Display Area	106
9.3.5	Manual Control	106
9.3.5.1	Implicit keys	106
9.3.5.2	The Spindle group	107
9.3.5.3	The Coolant group	107
9.3.6	Code Entry	107
9.3.6.1	MDI:	107

9.3.6.2 Active G-Codes	107
9.3.7 Jog Speed	108
9.3.8 Feed Override	108
9.3.9 Spindle speed Override	108
9.4 Keyboard Controls	108
10 MINI GUI	109
10.1 Introduction	109
10.2 Screen layout	110
10.3 Menu Bar	111
10.4 Control Button Bar	112
10.4.1 MANUAL	112
10.4.2 AUTO	113
10.4.3 MDI	114
10.4.4 [FEEDHOLD]—[CONTINUE]	114
10.4.5 [ABORT]	114
10.4.6 [ESTOP]	115
10.5 Left Column	115
10.5.1 Axis Position Displays	115
10.5.2 Feed rate Override	116
10.5.3 Messages	116
10.6 Right Column	116
10.6.1 Program Editor	117
10.6.2 Backplot Display	118
10.6.3 Tool Page	118
10.6.4 Offset Page	119
10.7 Keyboard Bindings	120
10.7.1 Common Keys	120
10.7.2 Manual Mode	121
10.7.3 Auto Mode	122
10.8 Misc	122
11 KEYSTICK GUI	123
11.1 Introduction	123
11.2 Installing	124
11.3 Using	124

III Using LinuxCNC	125
12 CNC Machine Overview	126
12.1 Mechanical Components	126
12.1.1 Axes	126
12.1.2 Spindle	126
12.1.3 Coolant	126
12.1.4 Feed and Speed Override	127
12.1.5 Block Delete Switch	127
12.1.6 Optional Program Stop Switch	127
12.2 Control and Data Components	127
12.2.1 Linear Axes	127
12.2.2 Rotational Axes	127
12.2.3 Controlled Point	127
12.2.4 Coordinated Linear Motion	128
12.2.5 Feed Rate	128
12.2.6 Coolant	128
12.2.7 Dwell	128
12.2.8 Units	128
12.2.9 Current Position	128
12.2.10 Selected Plane	129
12.2.11 Tool Carousel	129
12.2.12 Tool Change	129
12.2.13 Pallet Shuttle	129
12.2.14 Path Control Mode	129
12.3 Interpreter Interaction with Switches	129
12.3.1 Feed and Speed Override Switches	129
12.3.2 Block Delete Switch	129
12.3.3 Optional Program Stop Switch	129
12.4 Tool Table	130
12.5 Parameters	130
13 Coordinate System	131
13.1 Introduction	131
13.2 The Machine Position Command (G53)	131
13.3 Fixture Offsets (G54-G59.3)	132
13.3.1 Default coordinate system	133
13.3.2 Setting coordinate (fixture) offsets from G code	133
13.4 G92 Offsets	133
13.4.1 The G92 commands	133
13.4.2 Setting G92 values	134
13.4.3 G92 Cautions	134
13.5 Sample Program Using Offsets	135

14 Tool Compensation	137
14.1 Tool Length Offsets	137
14.1.1 Touch Off	137
14.1.2 Using G10 L1/L10/L11	138
14.2 Tool Table	138
14.2.1 Tool Table Format	138
14.2.2 Tool Changers	139
14.3 Cutter Compensation	140
14.3.1 Overview	141
14.3.2 Examples	142
15 G Code Overview	144
15.1 Overview	144
15.2 Format of a line	144
15.3 Block Delete	145
15.4 Line Number	145
15.5 Word	145
15.6 Number	146
15.7 Parameters	146
15.7.1 Numbered Parameters	147
15.7.2 Subroutine Parameters	149
15.7.3 Named Parameters	149
15.7.4 Predefined Named Parameters	150
15.7.5 System Parameters	151
15.8 Expressions	152
15.9 Binary Operators	153
15.9.1 Equality and floating-point values	153
15.10 Functions	153
15.11 Repeated Items	154
15.12 Item order	154
15.13 Commands and Machine Modes	155
15.14 Polar Coordinates	155
15.15 Modal Groups	157
15.16 Comments	158
15.17 Messages	159
15.18 Probe Logging	159
15.19 Logging	159
15.20 Debug Messages	159
15.21 Print Messages	159

15.22 Comment Parameters	159
15.23 File Requirements	160
15.24 File Size	160
15.25 G Code Order of Execution	160
15.26 G Code Best Practices	161
15.26.1 Use an appropriate decimal precision	161
15.26.2 Use consistent white space	161
15.26.3 Use Center-format arcs	161
15.26.4 Put important modal settings at the top of the file	161
15.26.5 Don't put too many things on one line	161
15.26.6 Don't set & use a parameter on the same line	161
15.26.7 Don't use line numbers	162
15.27 Linear and Rotary Axis	162
15.28 Common Error Messages	162
A Numbered Parameters persistence	163
16 G Codes	164
16.1 Conventions	164
16.2 G Code Quick Reference Table	164
16.3 G0 Rapid Move	165
16.3.1 Rapid Velocity Rate	166
16.4 G1 Linear Move	166
16.5 G2, G3 Arc Move	167
16.5.1 Center Format Arcs	167
16.5.2 Center Format Examples	169
16.5.3 Radius Format Arcs	171
16.6 G4 Dwell	171
16.7 G5 Cubic spline	172
16.8 G5.1 Quadratic spline	172
16.9 G5.2 G5.3 NURBS Block	173
16.10 G7 Lathe Diameter Mode	174
16.11 G8 Lathe Radius Mode	174
16.12 G10 L1 Set Tool Table	175
16.13 G10 L2 Set Coordinate System	175
16.14 G10 L10 Set Tool Table	177
16.15 G10 L11 Set Tool Table	177
16.16 G10 L20 Set Coordinate System	178
16.17 G17 - G19.1 Plane Selection	178

16.18G20, G21 Units	178
16.19G28, G28.1 Go to Predefined Position	179
16.20G30, G30.1 Go to Predefined Position	179
16.21G33 Spindle Synchronized Motion	180
16.22G33.1 Rigid Tapping	180
16.23G38.x Straight Probe	181
16.24G40 Compensation Off	182
16.25G41, G42 Cutter Compensation	183
16.26G41.1, G42.1 Dynamic Cutter Compensation	183
16.27G43 Tool Length Offset	184
16.28G43.1: Dynamic Tool Length Offset	184
16.29G43.2: Apply additional Tool Length Offset	185
16.30G49: Cancel Tool Length Compensation	185
16.31G53 Move in Machine Coordinates	185
16.32G54-G59.3 Select Coordinate System	186
16.33G61, G61.1 Exact Path Mode	186
16.34G64 Path Blending	186
16.35G73 Drilling Cycle with Chip Breaking	187
16.36G76 Threading Cycle	188
16.37Canned Cycles	190
16.37.1 Common Words	190
16.37.2 Sticky Words	190
16.37.3 Repeat Cycle	191
16.37.4 Retract Mode	191
16.37.5 Canned Cycle Errors	191
16.37.6 Preliminary and In-Between Motion	191
16.37.7 Why use a canned cycle?	192
16.38G80 Cancel Canned Cycle	193
16.39G81 Drilling Cycle	194
16.40G82 Drilling Cycle, Dwell	197
16.41G83 Peck Drilling Cycle	198
16.42G84 Right-Hand Tapping Cycle	198
16.43G85 Boring Cycle, Feed Out	198
16.44G86 Boring Cycle, Spindle Stop, Rapid Move Out	199
16.45G87 Back Boring Cycle	199
16.46G88 Boring Cycle, Spindle Stop, Manual Out	199
16.47G89 Boring Cycle, Dwell, Feed Out	199
16.48G90, G91 Distance Mode	199
16.49G90.1, G91.1 Arc Distance Mode	200

16.50G92 Coordinate System Offset	200
16.51G92.1, G92.2 Reset Coordinate System Offsets	200
16.52G92.3 Restore Axis Offsets	201
16.53G93, G94, G95: Feed Rate Mode	201
16.54G96, G97 Spindle Control Mode	201
16.55G98, G99 Canned Cycle Return Level	202
17 M Codes	203
17.1 M Code Quick Reference Table	203
17.2 M0, M1 Program Pause	203
17.3 M2, M30 Program End	204
17.4 M60 Pallet Change Pause	204
17.5 M3, M4, M5 Spindle Control	204
17.6 M6 Tool Change	204
17.6.1 Manual Tool Change	204
17.6.2 Tool Changer	205
17.7 M7, M8, M9 Coolant Control	205
17.8 M19 Orient Spindle	205
17.9 M48, M49 Speed and Feed Override Control	206
17.10M50 Feed Override Control	206
17.11M51 Spindle Speed Override Control	206
17.12M52 Adaptive Feed Control	207
17.13M53 Feed Stop Control	207
17.14M61 Set Current Tool Number	207
17.15M62 to M65 Output Control	207
17.16M66 Wait on Input	208
17.17M67 Synchronized Analog Output	208
17.18M68 Analog Output	209
17.19M70 Save Modal State	209
17.20M71 Invalidate Stored Modal State	210
17.21M72 Restore Modal State	210
17.22M73 Save and Autorestore Modal State	211
17.22.1 Selectively restoring modal state by testing predefined parameters	212
17.23M100 to M199 User Defined Commands	212
18 O Codes	214
18.1 Subroutines	215
18.2 Looping	215
18.3 Conditional	216
18.4 Repeat	217
18.5 Indirection	217
18.6 Calling Files	217
18.7 Subroutine return values	218

19 Other Codes	219
19.1 F: Set Feed Rate	219
19.2 S: Set Spindle Speed	219
19.3 T: Select Tool	219
20 G Code Examples	221
20.1 Mill Examples	221
20.1.1 Helical Hole Milling	221
20.1.2 Slotting	221
20.1.3 Grid Probe	221
20.1.4 Smart Probe	221
20.1.5 Tool Length Probe	222
20.1.6 Hole Probe	222
20.1.7 Cutter Compensation	222
20.2 Lathe Examples	222
20.2.1 Threading	222
21 Lathe User Information	223
21.1 Lathe Mode	223
21.2 Lathe Tool Table	223
21.3 Lathe Tool Orientation	223
21.4 Tool Touch Off	227
21.4.1 X Touch Off	227
21.4.2 Z Touch Off	227
21.4.3 The Z Machine Offset	228
21.5 Spindle Synchronized Motion	228
21.6 Arcs	228
21.6.1 Arcs and Lathe Design	229
21.6.2 Radius & Diameter Mode	229
21.7 Tool Path	229
21.7.1 Control Point	229
21.7.2 Cutting Angles without Cutter Comp	230
21.7.3 Cutting a Radius	231
21.7.4 Using Cutter Comp	233
22 RS274/NGC Differences	234
22.1 Changes from RS274/NGC	234
22.2 Additions to RS274/NGC	234

23 Image to G Code	236
23.1 What is a depth map?	236
23.2 Integrating image-to-gcode with the AXIS user interface	236
23.3 Using image-to-gcode	237
23.4 Option Reference	237
23.4.1 Units	237
23.4.2 Invert Image	237
23.4.3 Normalize Image	237
23.4.4 Expand Image Border	237
23.4.5 Tolerance (units)	237
23.4.6 Pixel Size (units)	237
23.4.7 Plunge Feed Rate (units per minute)	237
23.4.8 Feed Rate (units per minute)	238
23.4.9 Spindle Speed (RPM)	238
23.4.10 Scan Pattern	238
23.4.11 Scan Direction	238
23.4.12 Depth (units)	238
23.4.13 Step Over (pixels)	238
23.4.14 Tool Diameter	238
23.4.15 Safety Height	238
23.4.16 Tool Type	239
23.4.17 Lace bounding	239
23.4.18 Contact angle	239
23.4.19 Roughing offset and depth per pass	239
24 Glossary	241
25 Legal Section	246
25.1 Copyright Terms	246
25.2 GNU Free Documentation License	246
26 Index	250

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.

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

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

² Found 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.

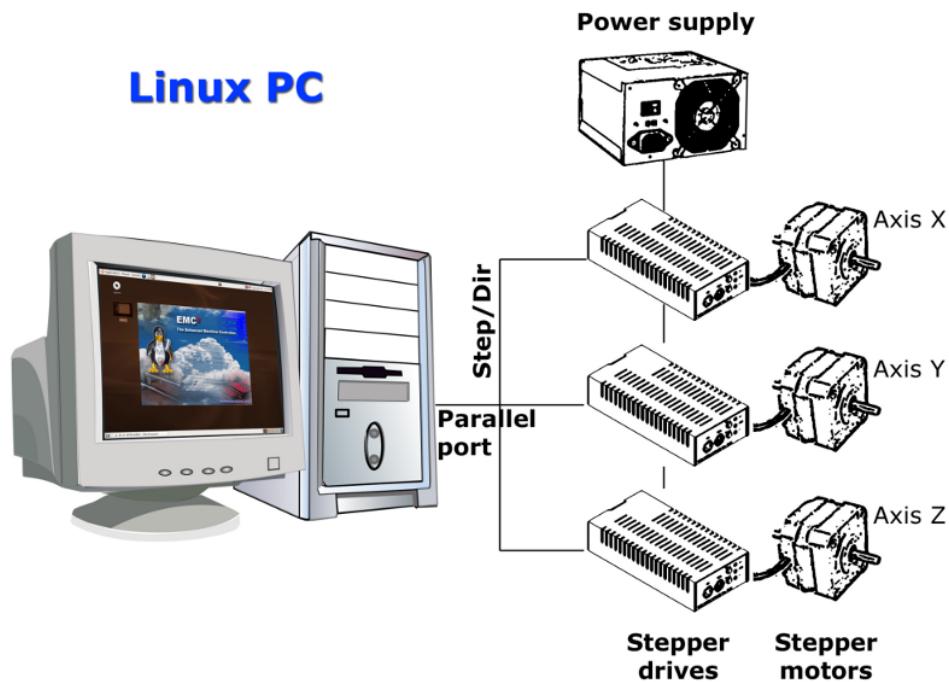


Figure 2.1: Simple LinuxCNC Controlled Machine

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.

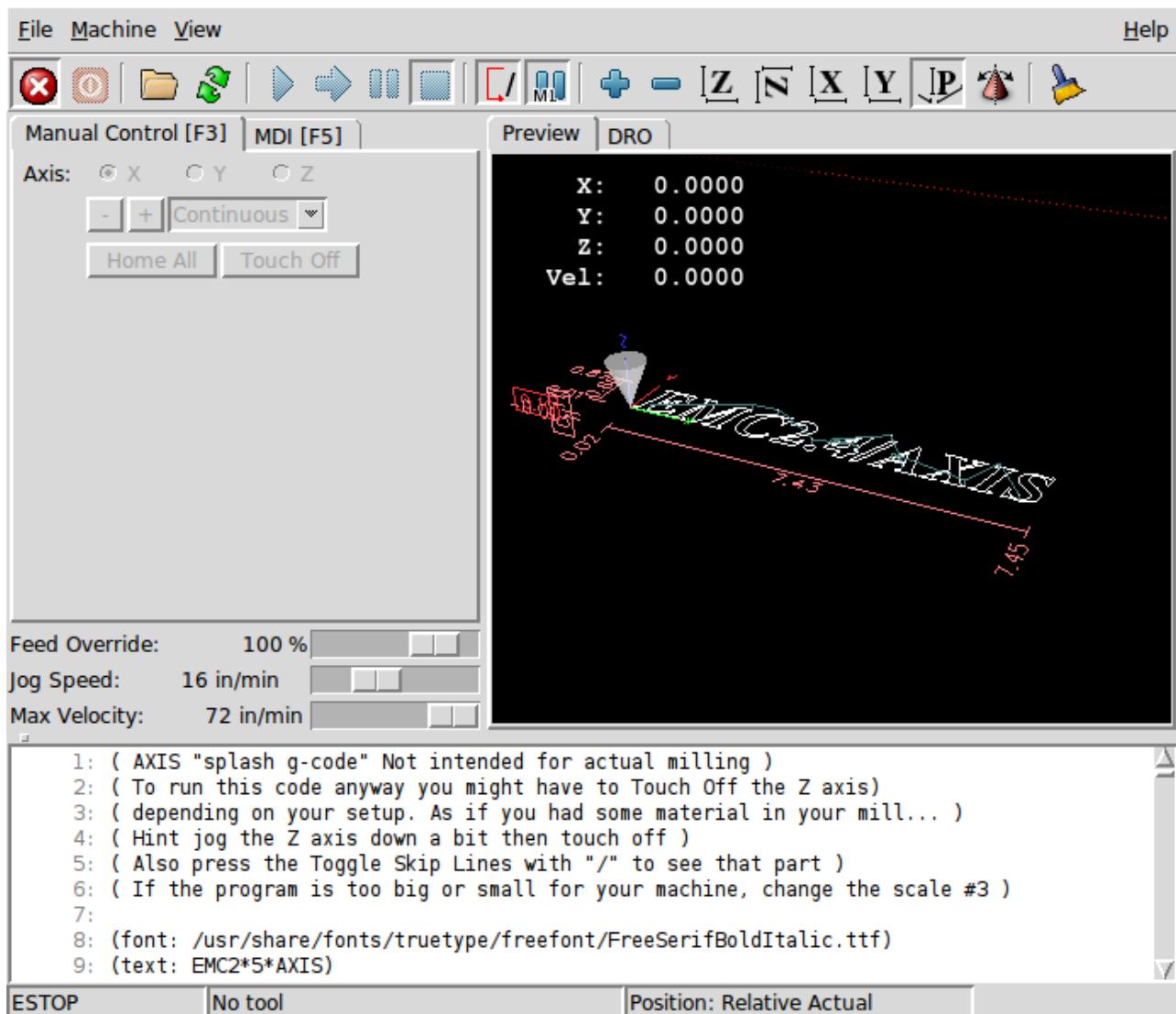


Figure 2.2: Axis GUI

- [gmoccapy](#), A industrial like GUI.

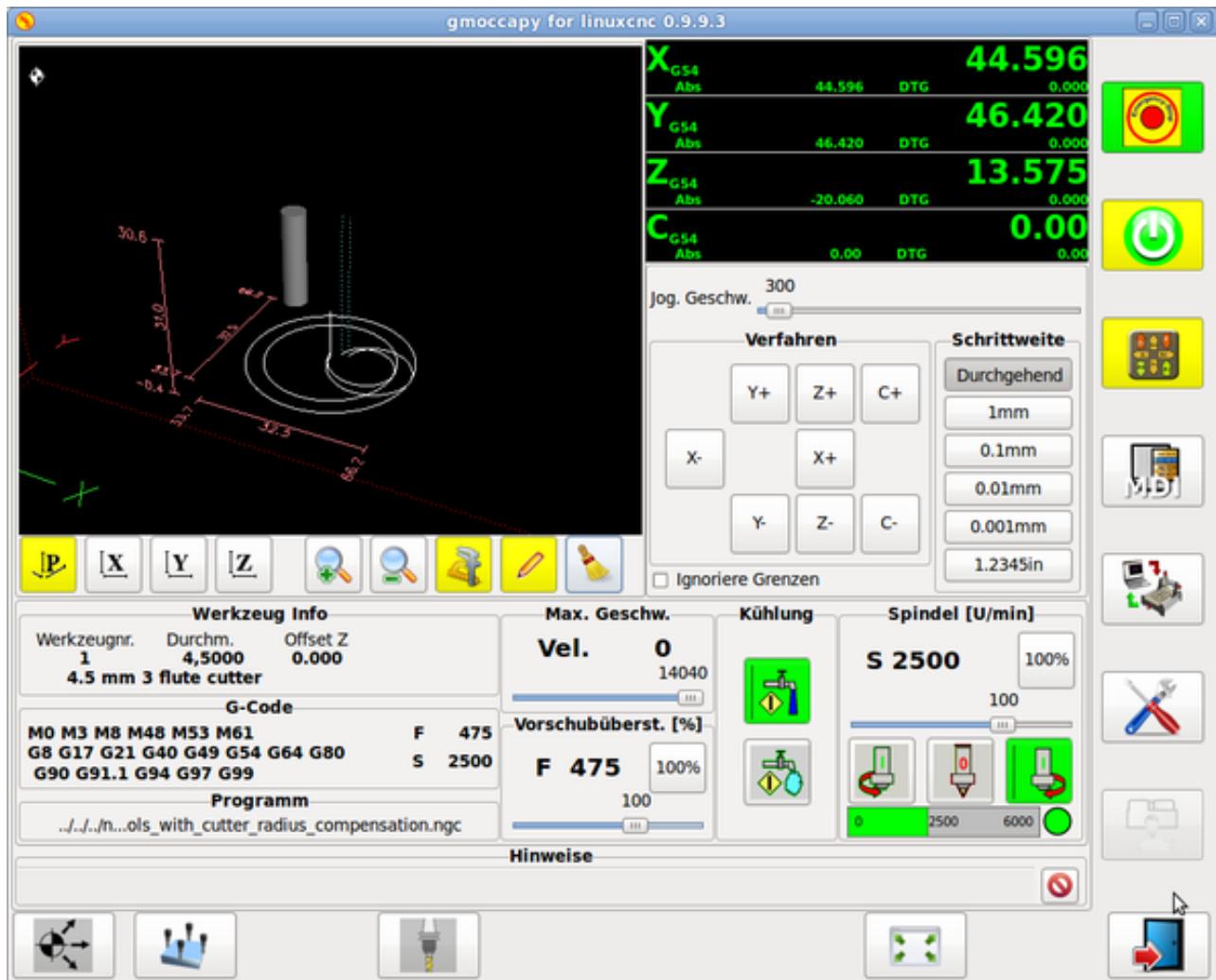


Figure 2.3: Gmoccapy

- [Touchy](#), a touch screen GUI.

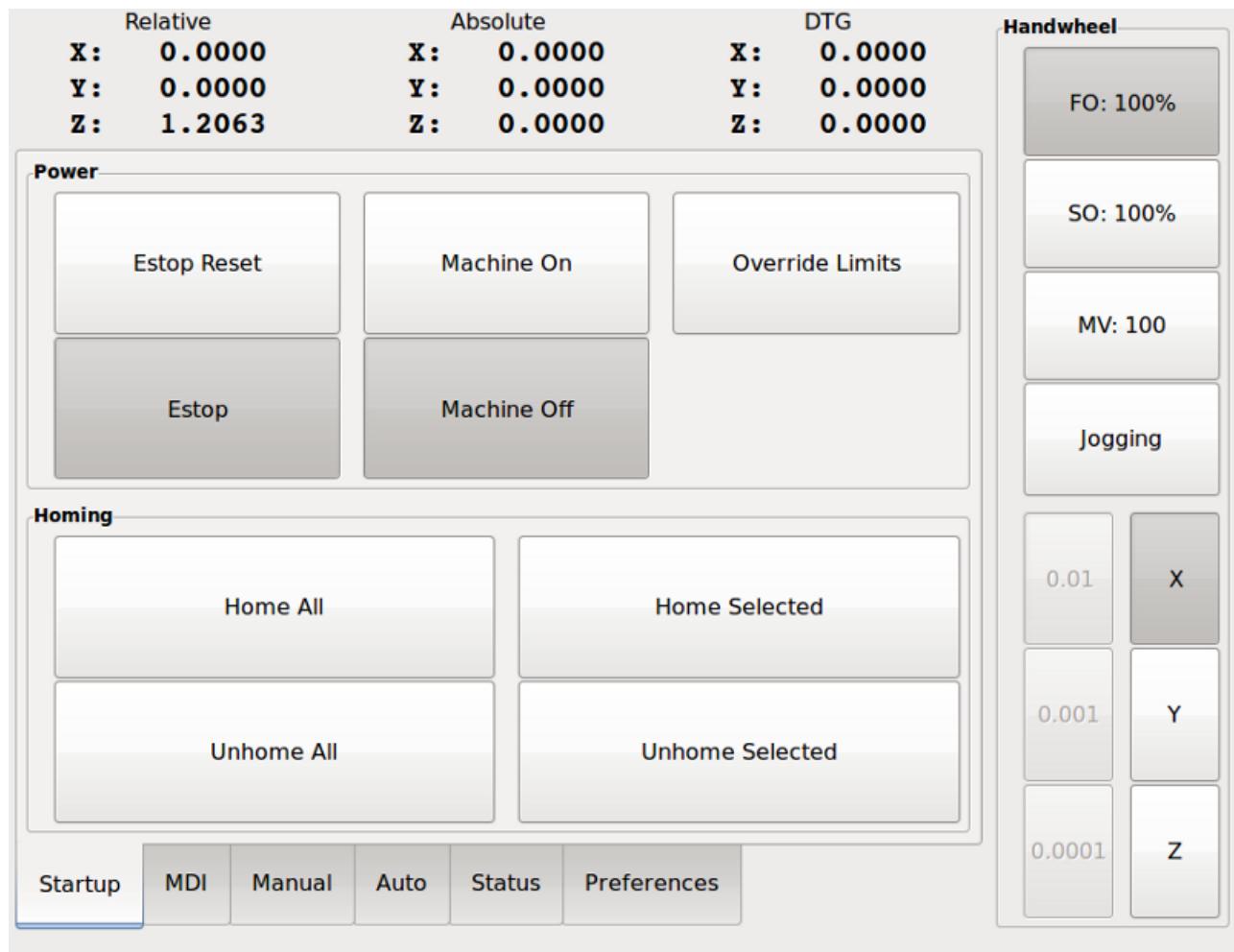


Figure 2.4: Touchy GUI

- **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.

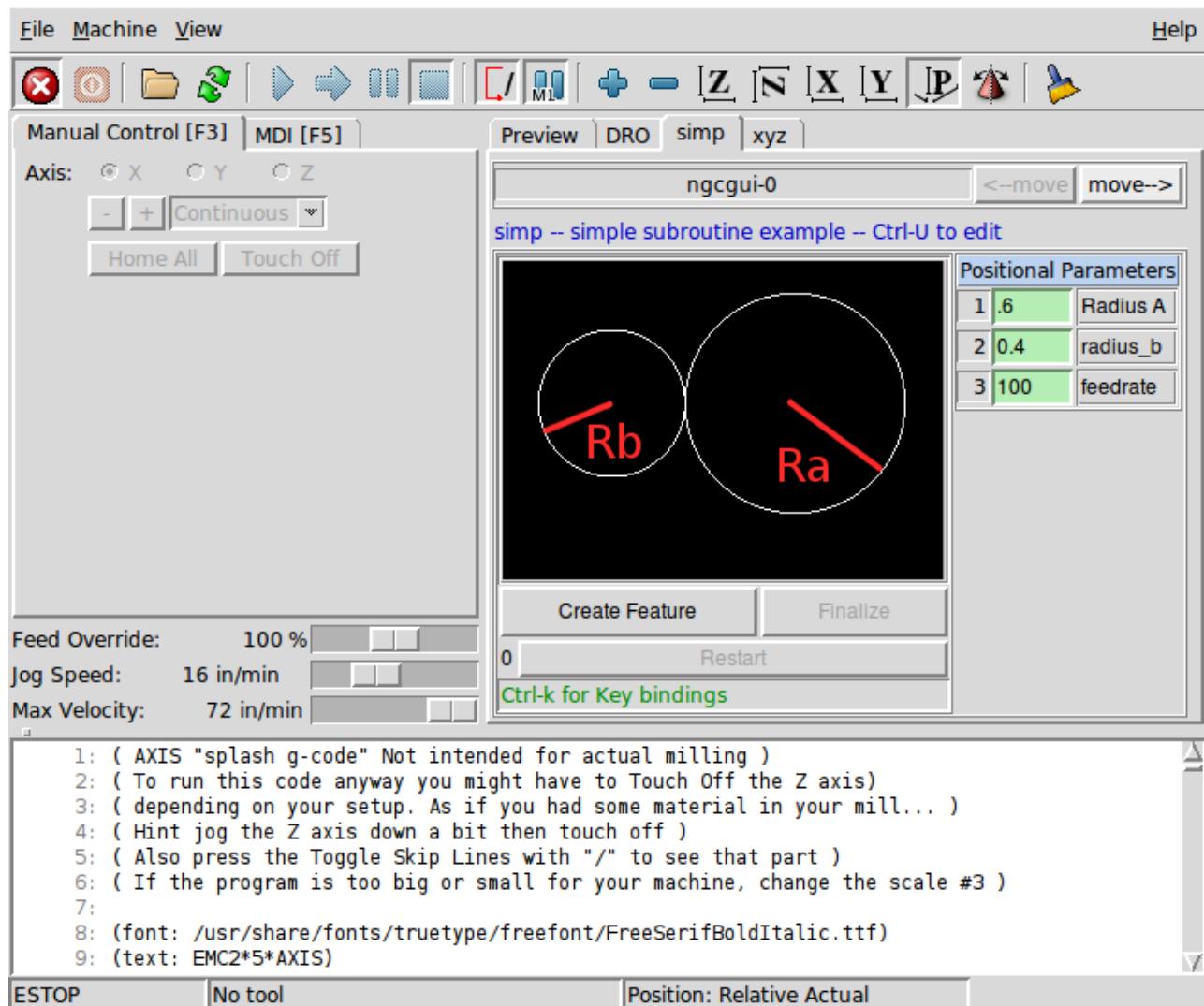


Figure 2.5: NGCGUI GUI embedded into Axis

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

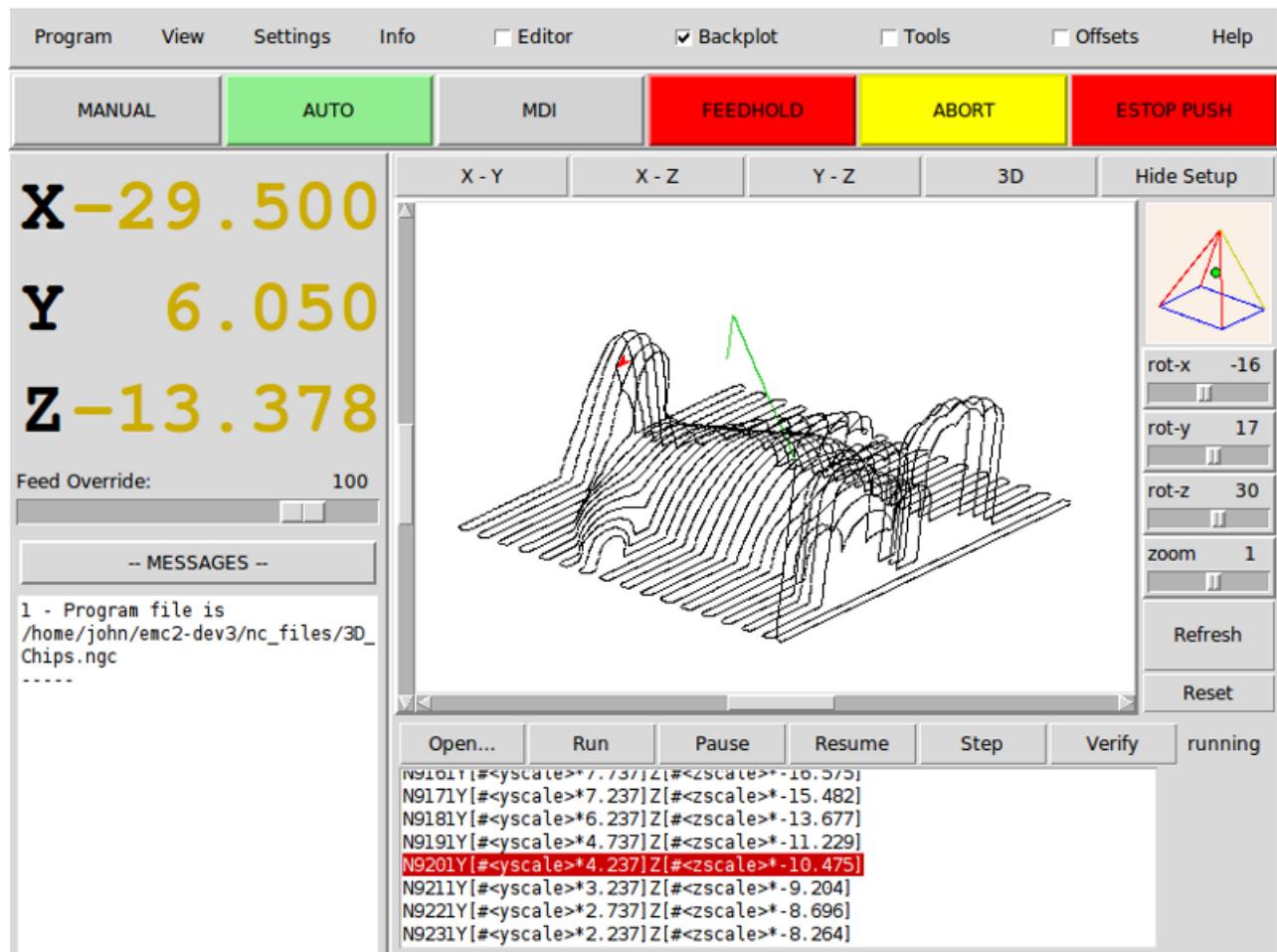


Figure 2.6: The Mini GUI

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



Figure 2.7: The TkLinuxCNC GUI

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

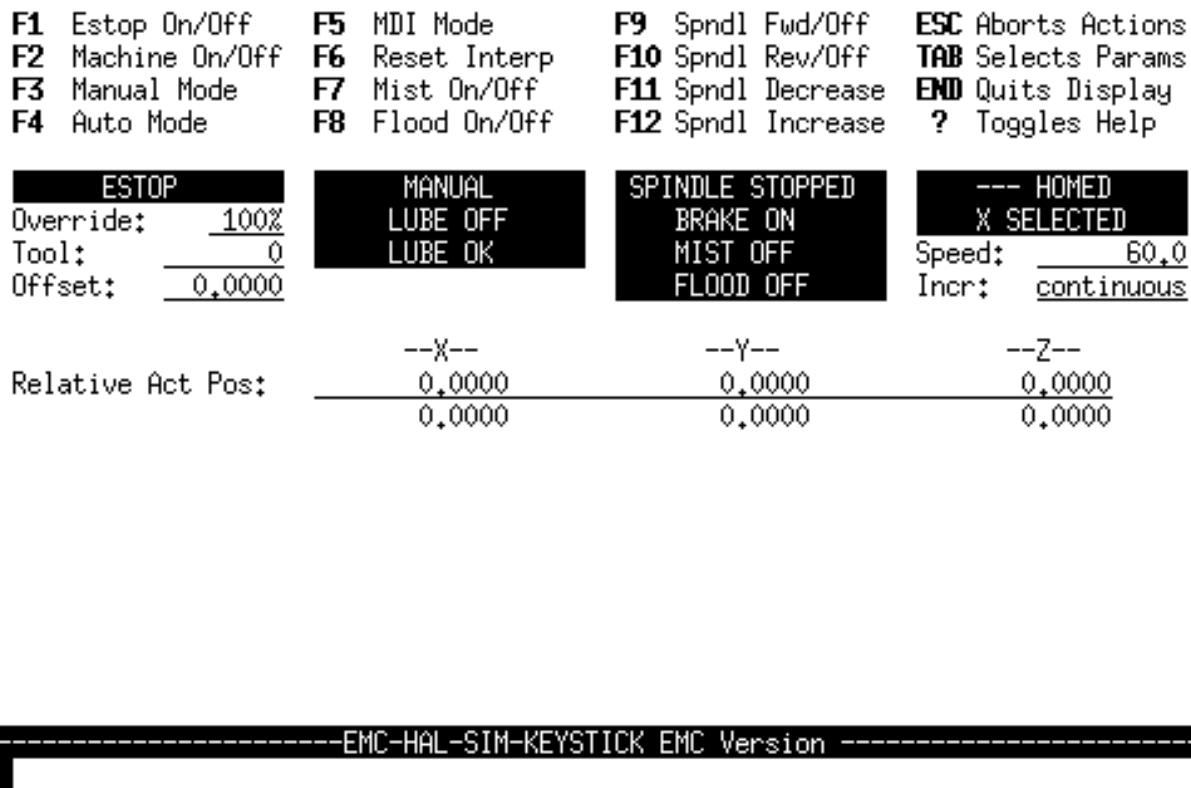


Figure 2.8: The Keystick GUI

- *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.
- *linuxcnrcrsh* - 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.

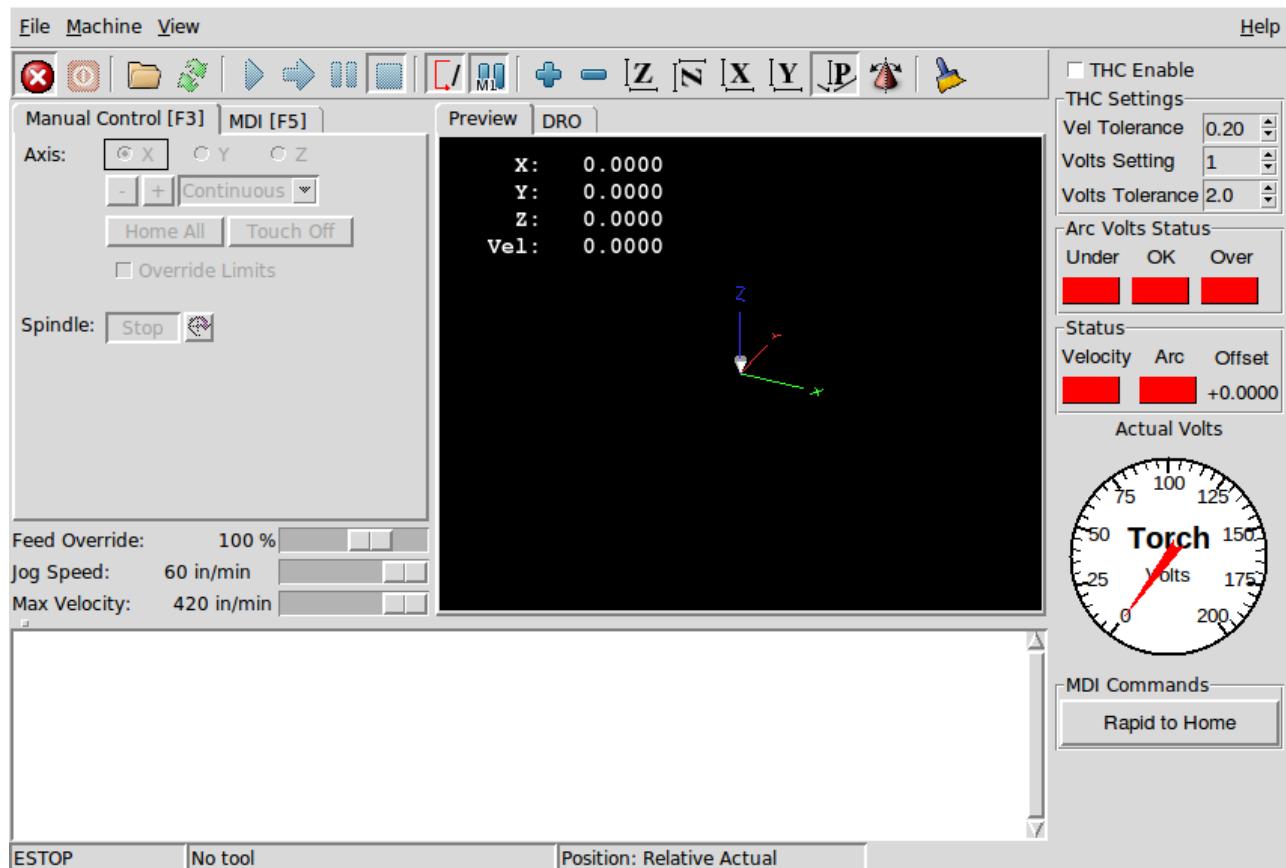


Figure 2.9: PyVCP with Axis

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

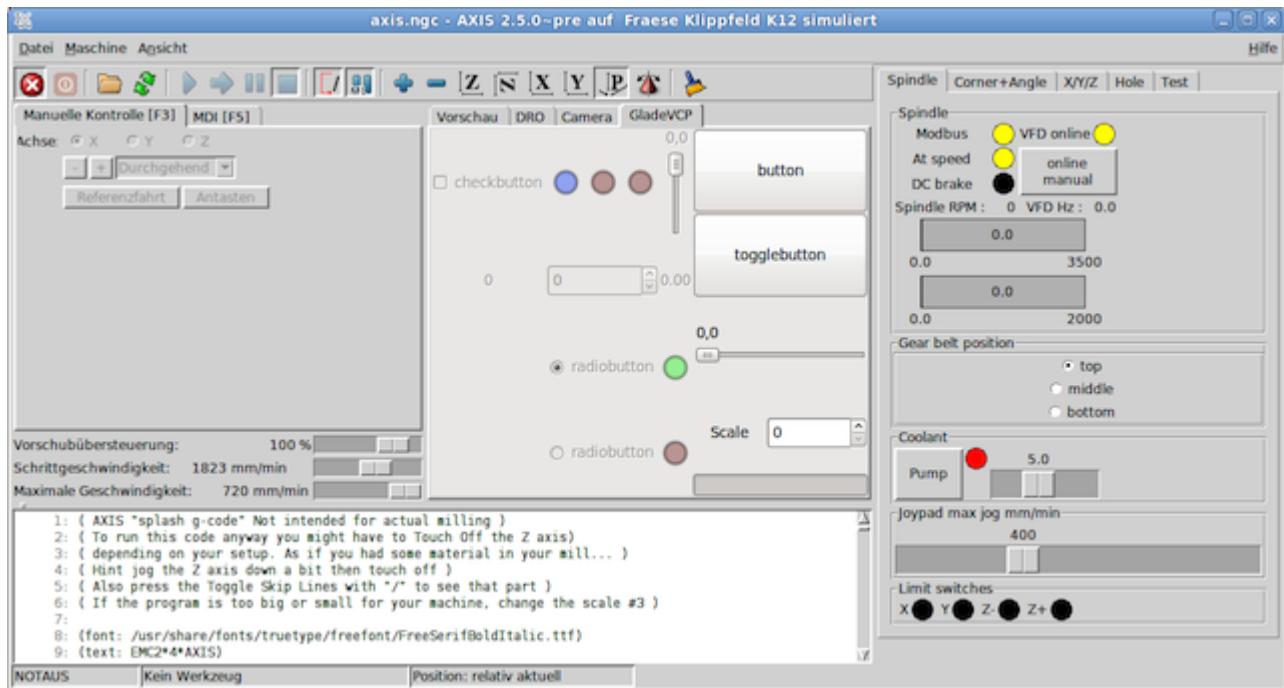


Figure 2.10: GladeVCP with Axis

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.

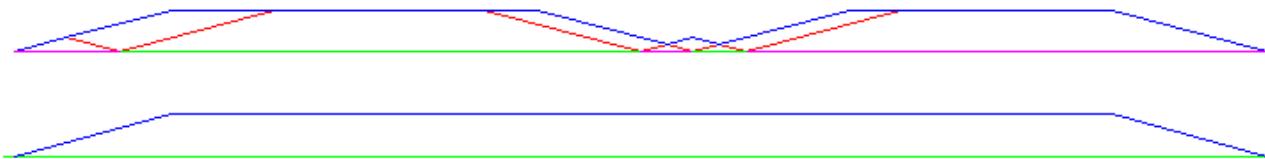


Figure 3.1: Naive Cam Detector

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 $d_a = F*t_a/2$. The deceleration time and distance are the same, making the critical distance $d = d_a + d_d = 2 * d_a = F^2/A$.

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 **$5^2/100 = 25/10 = 0.025$ 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.

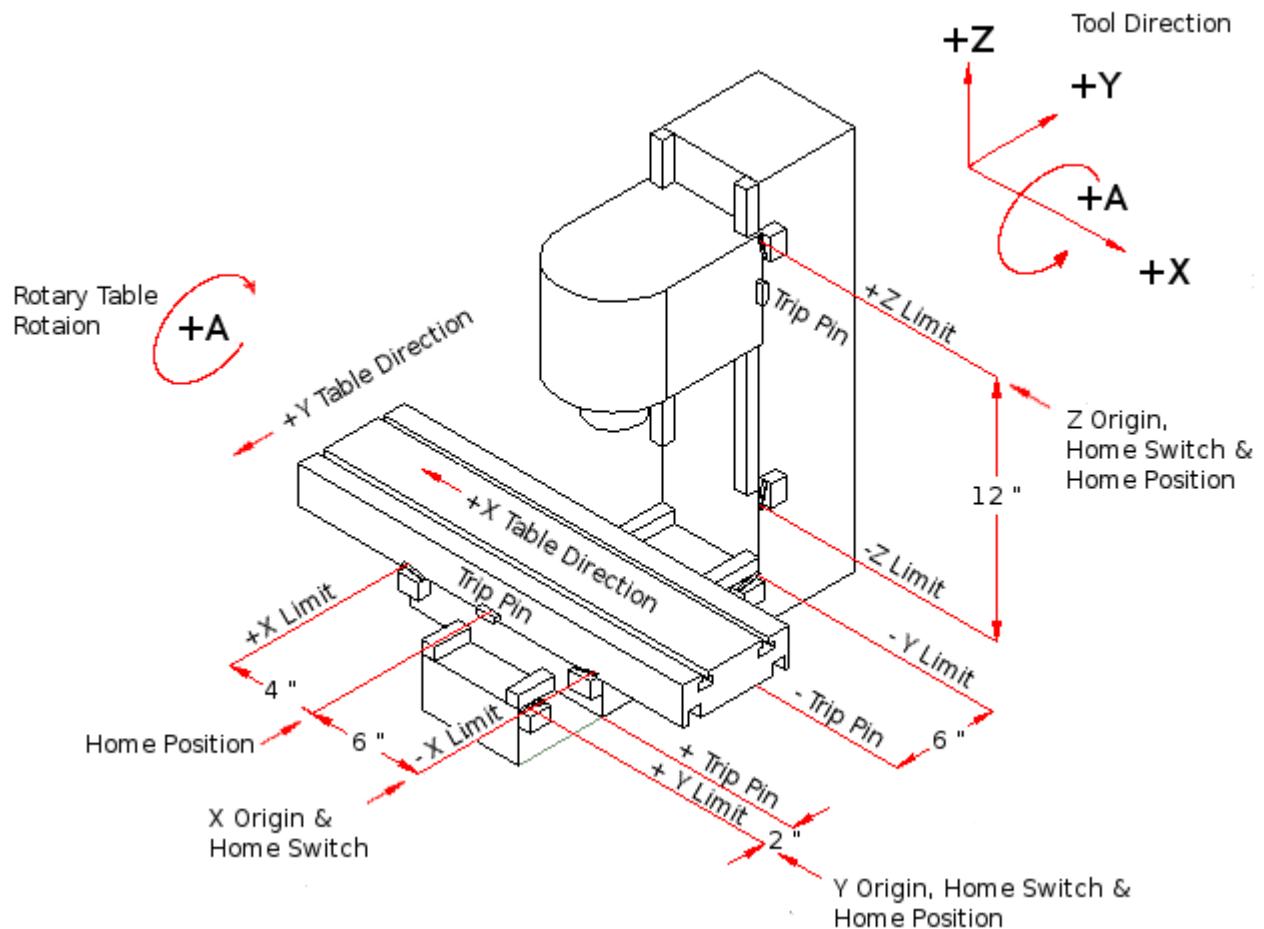


Figure 3.2: Mill Configuration

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

[images/lathe-diagram.png](#)

Figure 3.3: Lathe Configuration

Part II

User Interfaces

Chapter 4

CONFIGURATION SELECTOR

The Configuration Selector gui is activated from the system main menu when CNC / LinuxCNC is selected.

The Configuration Selector offers a selection of configurations organized:

My Configurations ----- User configurations

Sample Configurations ---- Sample configurations that
can be copied to 'My Configurations'

sim ----- simulation configurations. These
can be used for testing or learning

by_interface ----- configurations organized by interface

by_machine ----- configurations organized by machine

apps ----- applications that do not require
starting linuxcnc but may be useful
for testing or trying applications
like pyvcp or gladevcp

attic ----- obsolete or historical configurations

The sim configurations are often the most useful starting point for new users and are organized around supported guis:

axis
gmoccapy
gscreen
low_graphics
tklinuxcnc
touchy

A gui configuration directory may contain subdirectories with configurations that illustrate special situations or the embedding of other applications.

The by_interface configurations are organized around common, supported interfaces like:

general mechatronics
mesa
parport

```
pico  
pluto  
servotogo  
vigilant  
vitalsystems
```

Related hardware may be required to use these configurations as starting points for a system.

The by_machine configurations are organized around complete, known systems like:

```
boss  
cooltool  
sherline  
smithy  
tormach
```

A complete system may be required to use these configurations.

The apps items are typically 1) utilities that don't require starting linuxcnc or 2) demonstrations of applications that can be used with linuxcnc:

```
info ----- creates a file with system information that  
                  may be useful for problem diagnosis  
gladevcp -- example gladevcp applications  
halrun --- starts halrun in an terminal  
latency --- applications to investigate latency  
parport --- applications to test parport  
pyvcp ----- example pyvcp applications  
xhc-hb04 -- applications to test an xhc-hb04 USB wireless MPG
```

The attic directory stores obsolete or historical configurations.

When started, the Configuration Selector allows the user to pick one of their existing configurations (My Configurations) or select a new one (from the Sample Configurations) to be copied to their home directory. Copied configurations will appear under My Configurations on the next invocation of the Configuration Selector.

Note

Under the Apps directory, only applications that are usefully modified by the user are offered for copying to the user's directory.

Chapter 5

AXIS GUI

5.1 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.

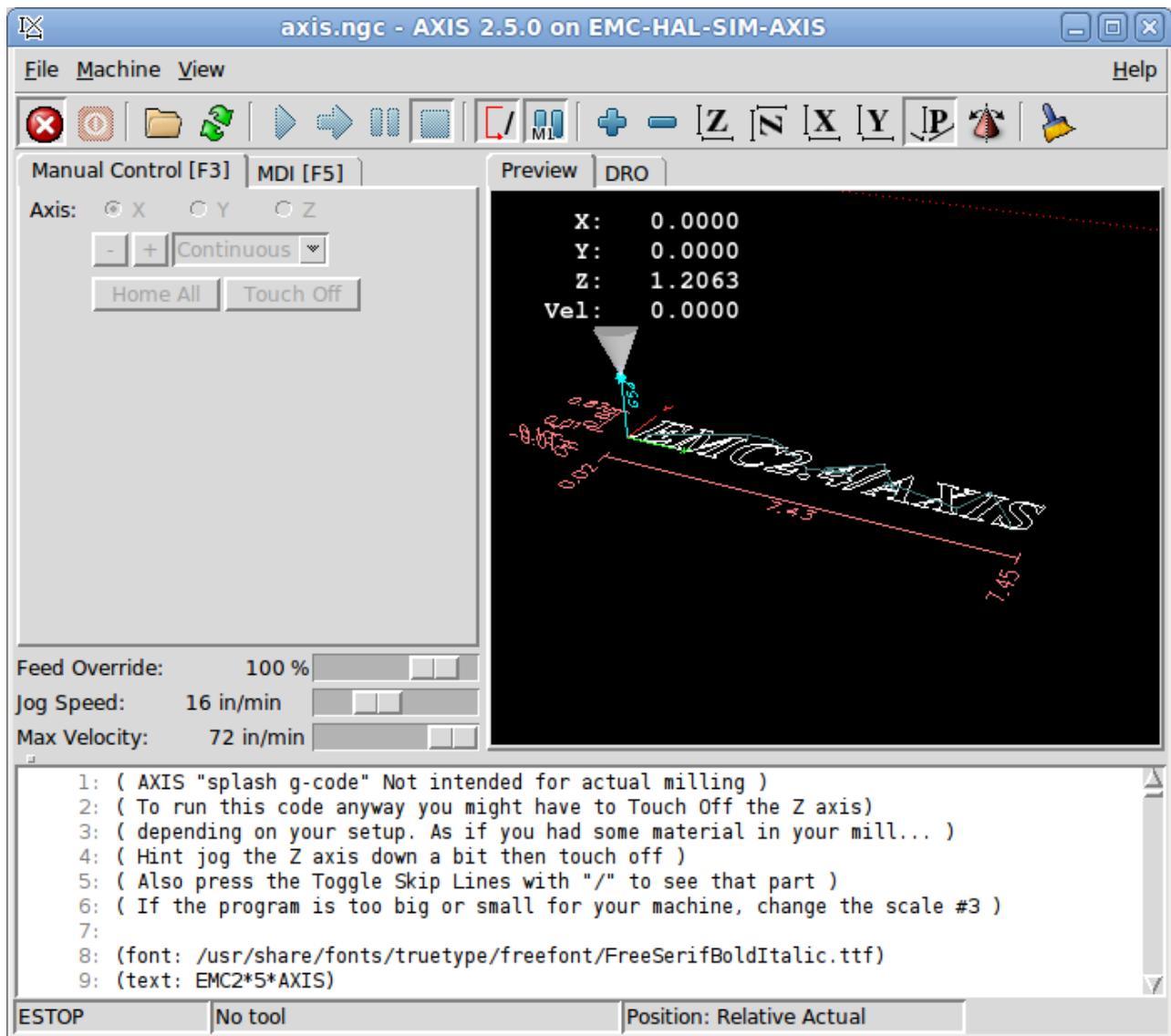


Figure 5.1: AXIS Window

5.2 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.

5.2.1 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.

5.3 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).

5.3.1 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 the Servo Axis Calibration assistant. Calibration reads the HAL file and for every *setp* that uses a variable from the ini file that is in an [AXIS_n] section it creates an entry that can be edited and tested.
- *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.
 - *Unhomming* - 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.

5.3.2 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]

5.3.3 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 5.2: 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.

Grid Axis can optionally display a grid when in orthogonal views. Enable or disable the grid using the *Grid* menu under *View*. When enabled, the grid is shown in the top and rotated top views; when coordinate system is not rotated, the grid is shown in the front and side views as well. The presets in the *Grid* menu are controlled by the infile item [DISPLAY] GRIDS; if unspecified, the default is 10mm 20mm 50mm 100mm 1in 2in 5in 10in.

Specifying a very small grid may decrease performance.

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.

5.3.4 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.

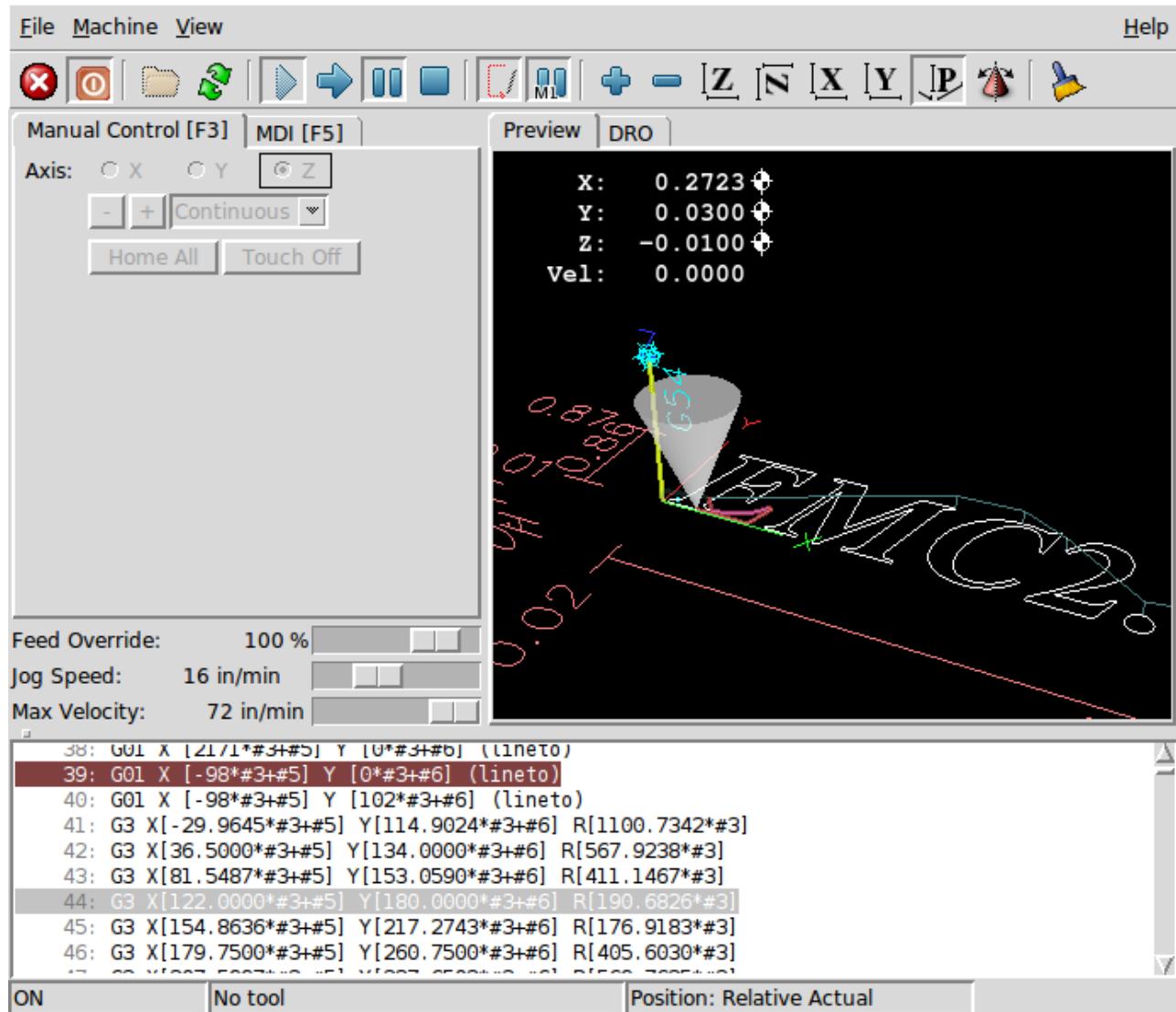


Figure 5.3: Current and Selected Lines

5.3.5 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.

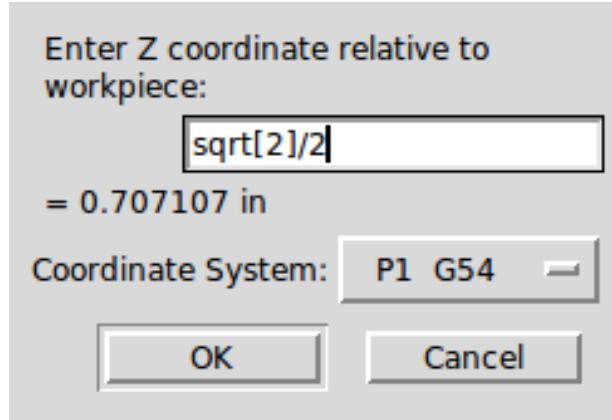


Figure 5.4: Touch Off

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 check box 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.

5.3.6 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.

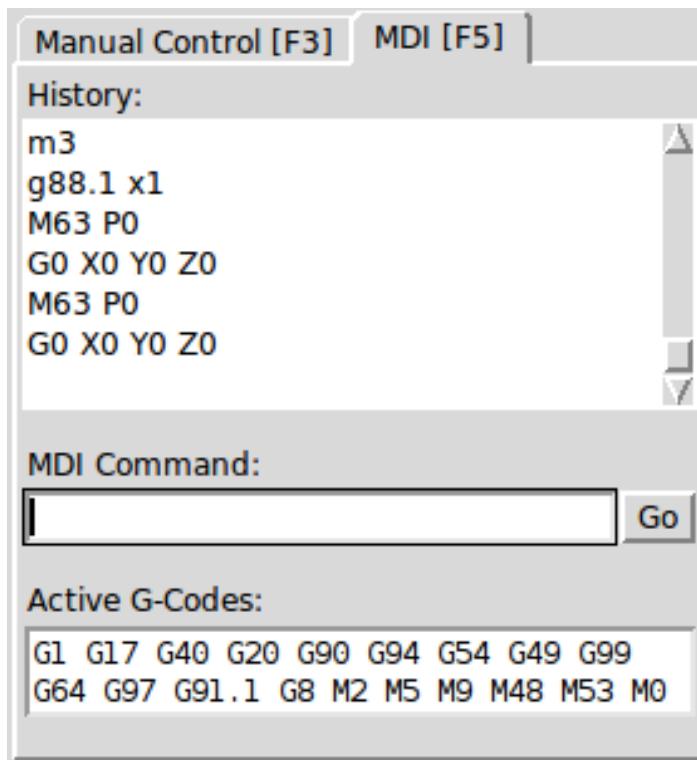


Figure 5.5: The MDI tab

- *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.

5.3.7 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.

5.3.8 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.

5.3.9 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).

5.3.10 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.

5.4 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 programmed. 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 5.1: 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

Table 5.1: (continued)

Keystroke	Action Taken	Mode
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

5.5 Show LinuxCNC Status (linuxcncstatus)

AXIS includes a program called *linuxcnctop* which shows some of the details of LinuxCNC's state. You can run this program by invoking Machine > Show LinuxCNC Status

Figure 5.6: LinuxCNC Status Window

The name of each item is shown in the left column. The current value is shown in the right column. If the value has recently changed, it is shown on a red background.

5.6 MDI interface

AXIS includes a program called `mdi` which allows text-mode entry of MDI commands to a running LinuxCNC session. You can run this program by opening a terminal and typing

```
mdi
```

Once it is running, it displays the prompt *MDI>*. When a blank line is entered, the machine's current position is shown. When a command is entered, it is sent to LinuxCNC to be executed.

This is a sample session of `mdi`.

```
$ mdi
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)
```

5.7 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`).

5.8 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>*.

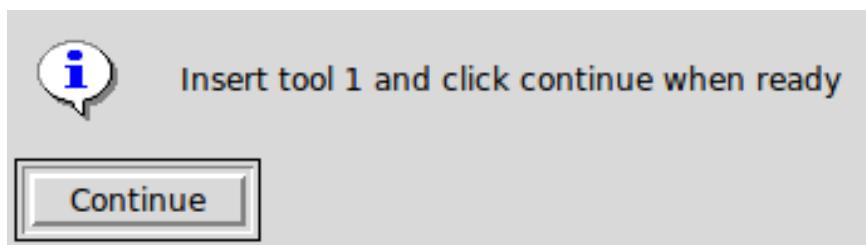


Figure 5.7: The Manual Toolchange Window

5.9 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

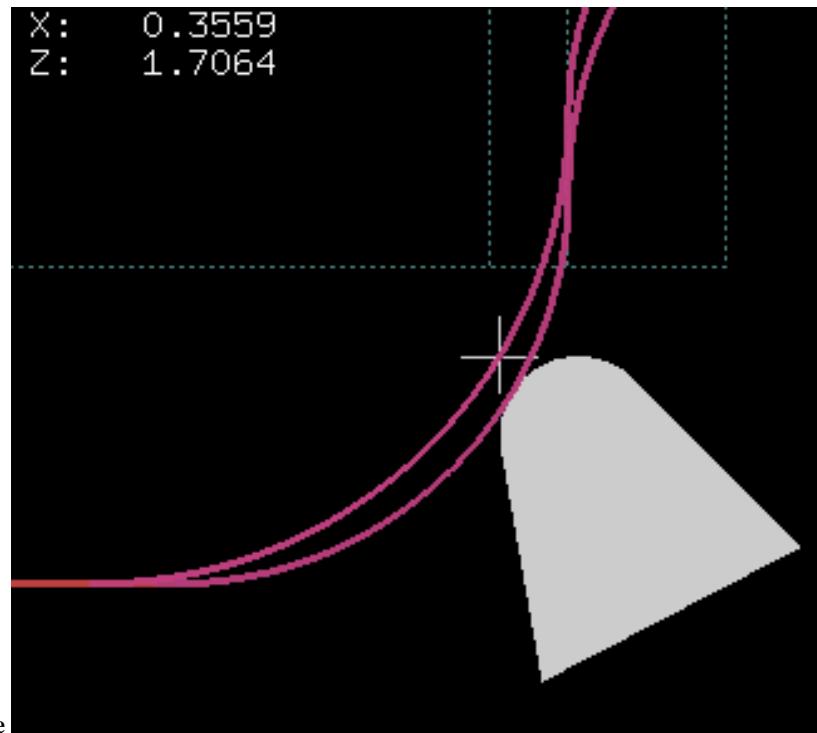
To use these modules in your own scripts, you must ensure that the directory where they reside is on Python's module path. When running an installed version of LinuxCNC, this should happen automatically. When running *in-place*, this can be done by using `scripts/rip-environment`.

5.10 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.



5.11 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.

5.11.1 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.

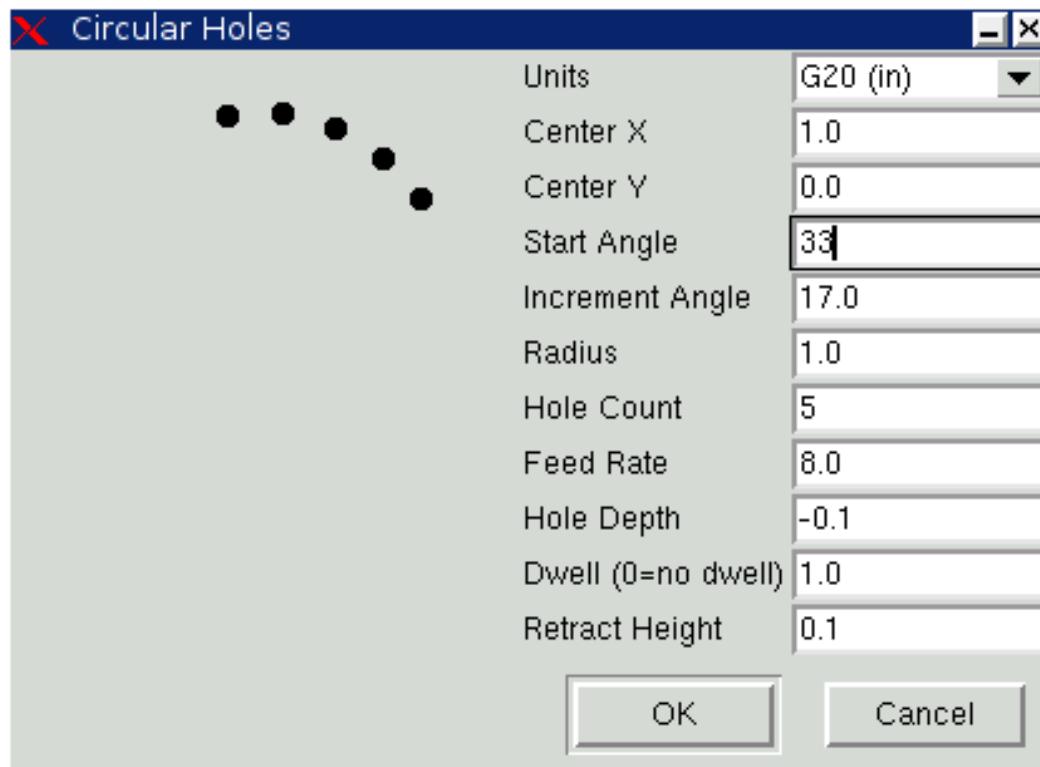


Figure 5.8: Circular Holes

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.

5.11.2 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.

5.11.3 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 *[HALJPOSTGUI_HALFILE]*. Unlike *[HALJHALFILE]*, only one such file may be used.

5.11.4 ~./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"))
```

5.11.5 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.

5.11.6 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.

5.11.7 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 6

gmoccapy

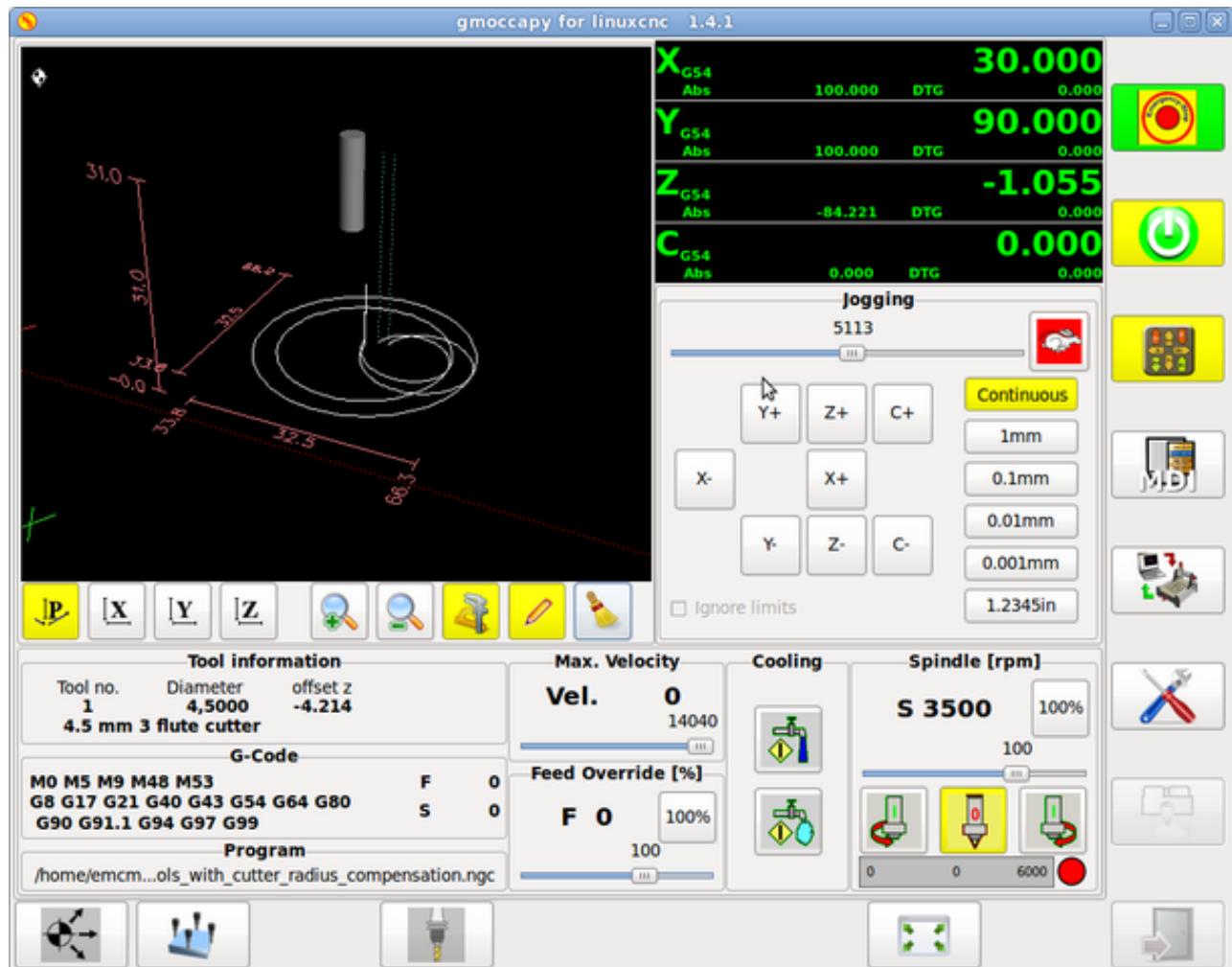
6.1 Introduction

GMOCCAPY is a GUI for linuxcnc, designed to be used with a touch screen, but can also be used on normal screens with a mouse or hardware buttons and MPG wheels, as it presents HAL Pins for the most common needs. Please find more information in the following.

It offers the possibility to display up to 4 axis, support a lathe mode for normal and back tool lathe and can be adapted to nearly every need, because gmoccapy supports embedded tabs and side panels. As a good example for that see [gmoccapy_plasma](#)

It has support for integrated onboard keyboard (onboard or matchbox-keyboard), so there is no need for a hardware keyboard or mouse, but it can also be used with that hardware. Gmoccapy offers a separate settings page to configure most settings of the GUI without editing files.

gmoccapy can be localized very easy, because the corresponding files are separated from the linuxcnc.po files, so there is no need to translate unneeded stuff. The files are placed in /src/po/gmoccapy. Just copy the gmoccapy.pot file to something like frpo and translate that file with gtranslator or poedit. After a make you got the GUI in your preferred language. Please publish your translation, so it can be included in the official packages and be published to other users. At the Moment it is available in English, German, Spanish, Polish and Serbian. Feel free to help me to introduce more languages, nieson@web.de. If you need help, don't hesitate to ask.



6.2 Requirements

Gmoccapy has been tested on UBUNTU 10.04 and 12.04 and DEBIAN Wheezy, with LinuxCNC 2.6 ,2.7 , master and machinekit, if you use other versions, please inform about problems or solutions on the forum or the [emc-users mailing list](#)

in German [Peters CNC Ecke](#)

in English [gmoccapy on linuxcnc](#)

The minimum screen resolution for gmoccapy, using it without side panels is **979 x 750 Pixel**, so it should fit to every standard screen.

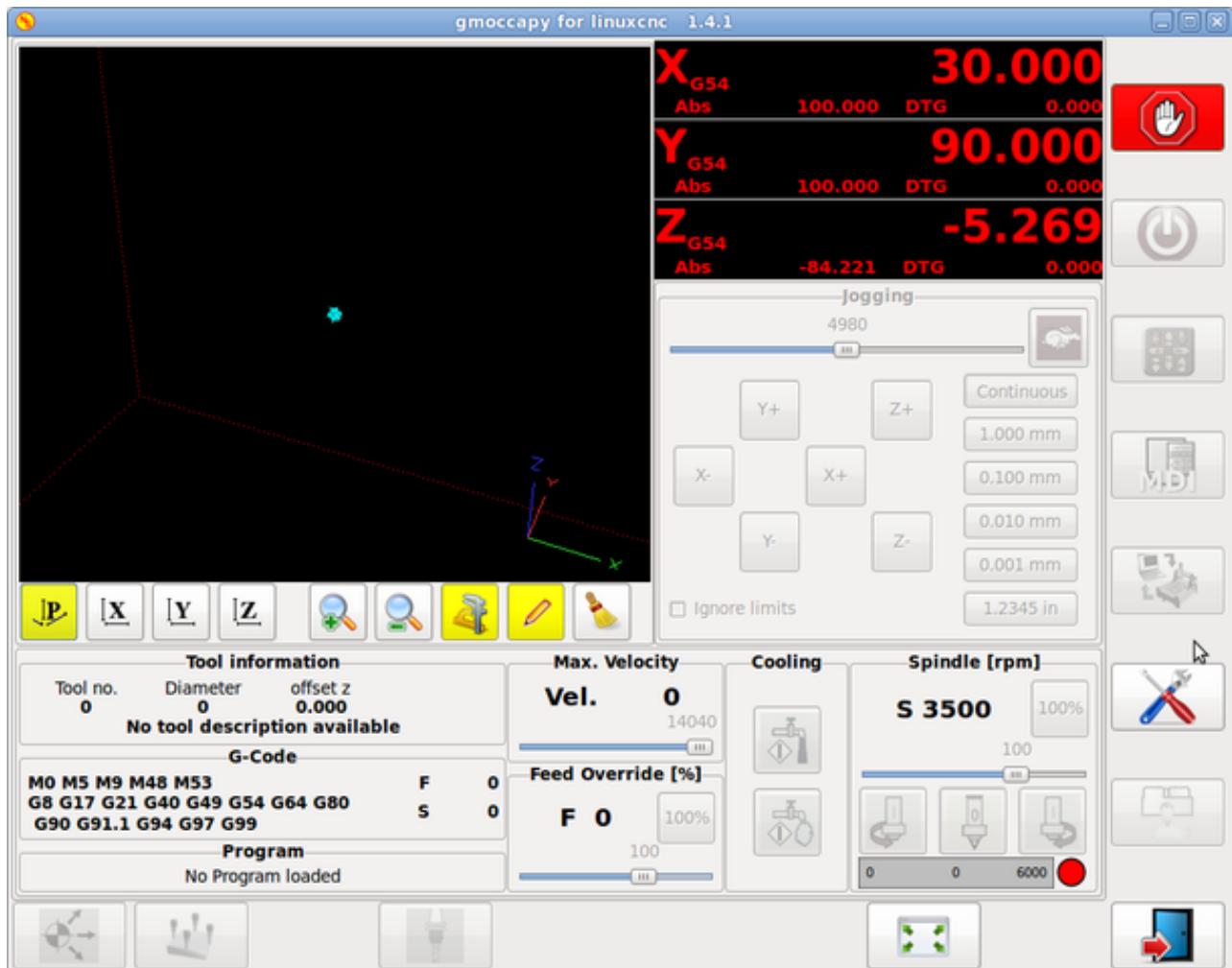
6.3 How to get gmoccapy

Beginning with LinuxCNC 2.6 gmoccapy is included in the standard installation. So the easiest way to get gmoccapy on your controlling PC, is just to get the [actual ISO](#) and install from the CD / DVD /USB-Stick.

If you do have already installed an earlier LinuxCNC version, check how to update [here](#)

You will receive updates with the regular deb packages.

You will get a similar screen to the following if your locale settings is **de** (German): The design may variate depending on your config.



6.4 Basic configuration

There is really not too much to configure just to run gmoccapy, but there are some points you should take care off if you want to use all the features of the GUI.

You will find the following INI files included, just to show the basics:

* gmoccapy.ini * gmoccapy_4_axis.ini * gmoccapy_jog_wheels.ini * gmoccapy_lathe.ini * gmoccapy_lathe_imperial.ini * gmoccapy_left_panel.ini * gmoccapy_right_panel.ini * gmoccapy_sim_hardware_button.ini * gmoccapy_tool_sensor.ini * gmoccapy_with_user_tabs.ini * gmoccapy_messages.ini

The names should explain the main intention of the different INI Files.

If you use an existing configuration of your machine, just edit your INI according to this document.



Important

If you want to use [MACROS](#), don't forget to set the path to your macros or subroutines folder as described below.

So let us take a closer look to the the INI file and what you need to include to use gmoccapy on your machine:

6.4.1 The DISPLAY Section

```
[DISPLAY]
DISPLAY = gmoccapy
PREFERENCE_FILE_PATH = gmoccapy_preferences
DEFAULT_LINEAR_VELOCITY = 166.666
MAX_LINEAR_VELOCITY = 166.666
MAX_FEED_OVERRIDE = 1.5
MAX_SPINDLE_OVERRIDE = 1.2
MIN_SPINDLE_OVERRIDE = 0.5
LATHE = 1
BACK_TOOL_LATHE = 1
PROGRAM_PREFIX = ../../nc_files/
```

The most important part is to tell LinuxCNC to use gmoccapy, editing the [DISPLAY] section.

```
[DISPLAY]
DISPLAY = gmoccapy

PREFERENCE_FILE_PATH = gmoccapy_preferences
```

The line PREFERENCE_FILE_PATH gives the location and name of the preferences file to be used. In most cases this line will not be needed, it is used by gmoccapy to store your settings of the GUI, like themes, DRO units, colors, and keyboard settings, etc., see [SETTINGS](#) for more details.

Note

If no path or file is given, gmoccapy will use as default <your_machinename>.pref, if no machine name is given in your INI File it will use gmoccapy.pref The file will be stored in your config dir, so the settings will not be mixed if you use several configs. If you only want to use one file for several machines, you need to include PREFERENCE_FILE_PATH in your INI.

DEFAULT_LINEAR_VELOCITY = 166.666

Sets the default linear velocity in machine units per second.

Note

If no value is given, a value of 15 will be applied. If you don't set max linear velocity, the default linear velocity will be reduced to the default value max linear velocity (60)
If you don't set max velocity in TRAJ, it may be reduced as well see [TRAY section](#)

MAX_LINEAR_VELOCITY = 166.666

Sets the value of the max velocity for jogging in machine units per second.

Note

If no value is given, a value of 60 will be applied. " MAX_FEED_OVERRIDE = 1.5

Sets the maximum feed override, in the example given, you will be allowed to override the feed by 150% "

MAX_SPINDLE_OVERRIDE = 1.2

MIN_SPINDLE_OVERRIDE = 0.5

will allow you to change the spindle override within a limit from 50% to 120%

```
LATHE = 1
BACK_TOOL_LATHE = 1
```

the first line set the screen layout to control a lathe.

The second line is optional and will switch the X axis in a way you need for a back tool lathe. Also the keyboard shortcuts will react in a different way.

Tip

See also [LATHE specific section](#)

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

Is the entry to tell linuxcnc/gmoccapy where to look for the ngc files,

Note

if not omitted we will look in the following order:

- ~/linuxcnc/nc_files
- ~/

6.4.1.1 The configuration of tabs and side panels

You can add embedded programs to gmoccapy like you can do in axis, touchy and gscreen. All is done by gmoccapy automatically if you include a few lines in your INI file in the DISPLAY section.

If you never used a glade panel, I recommend to read the excellent documentation. [Glade VCP](#)

Example

```
EMBED_TAB_NAME = DRO
EMBED_TAB_LOCATION = ntb_user_tabs
EMBED_TAB_COMMAND = gladevcp -x {XID} dro.glade
```

```
EMBED_TAB_NAME = Second user tab
EMBED_TAB_LOCATION = ntb_preview
EMBED_TAB_COMMAND = gladevcp -x {XID} vcp_box.glade
```

all you have to take care off, is that you include for every tab or side panel the mentioned three lines,

EMBED_TAB_NAME

represents the name of the tab or side panel, it is up to you what name you use, but it must be present!

EMBED_TAB_LOCATION

is the place where your program will be placed in the GUI. valid values are:

valid values are:

- ntb_user_tabs (as main tab, covering the complete screen)'
- ntb_preview (as tab on the preview side)'
- box_left (on the left, complete high of the screen)
- box_right (on the right, in between the normal screen and the button list)
- box_coolant_and_spindle (will hide the coolant and spindle frames and introduce your glade file here)
- box_cooling (will hide the cooling frame and introduce your glade file)
- box_spindle (will hide the spindle frame and introduce your glade file)
- box_vel_info (will hide the velocity frames and introduce your glade file)
- box_custom_1 (will introduce your glade file left of vel_frame)
- box_custom_2 (will introduce your glade file left of cooling_frame)
- box_custom_3 (will introduce your glade file left of spindle_frame)
- box_custom_4 (will introduce your glade file right of spindle_frame)

see the different INI files included to see the differences

EMBED_TAB_COMMAND

is the command to execute, i.e.

```
gladevcp -x {XID} dro.glade
```

includes a custom glade file called dro.glade in the mentioned location The file must be placed in the config folder of your machine

```
gladevcp h_buttonlist.glade
```

will just open a new user window called h_buttonlist.glade note the difference, this one is stand alone, and can be moved around independent from gmoccapy window

```
camview-emc -w {XID}
```

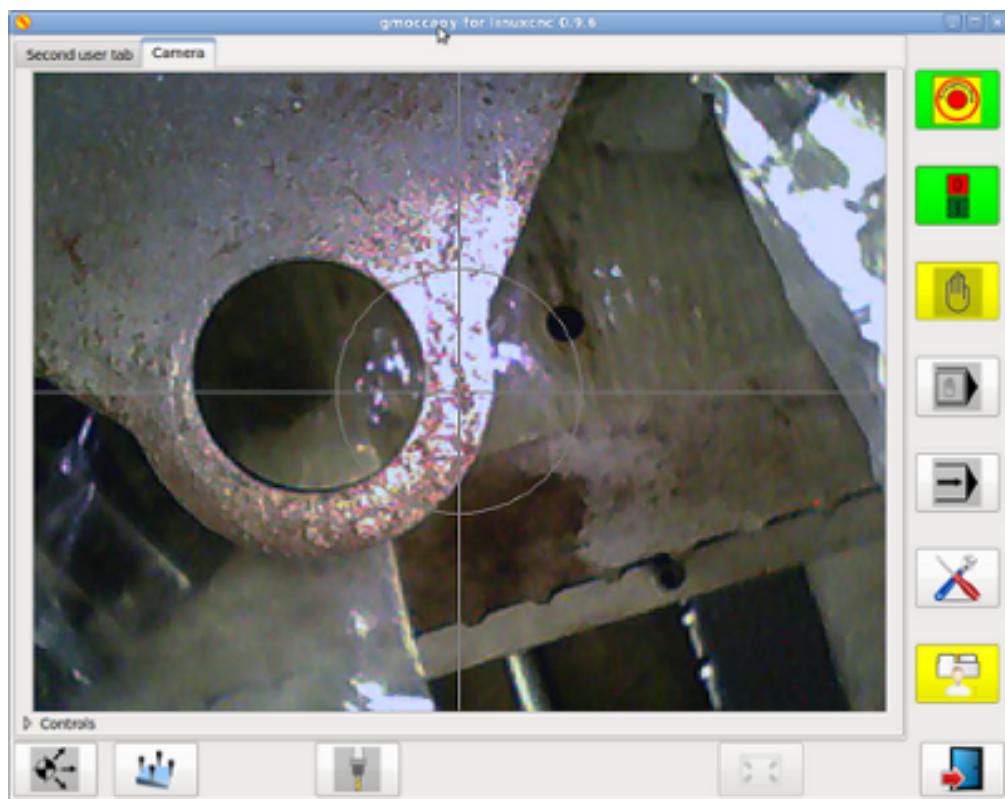
will add a live image from a web cam to the location you specified. take care that camview-emc is installed, as it is not by default. You find detailed information for camview and linuxcnc at: [cam view](#)

```
gladevcp -c gladevcp -u hitcounter.py -H manual-example.hal manual-example.ui
```

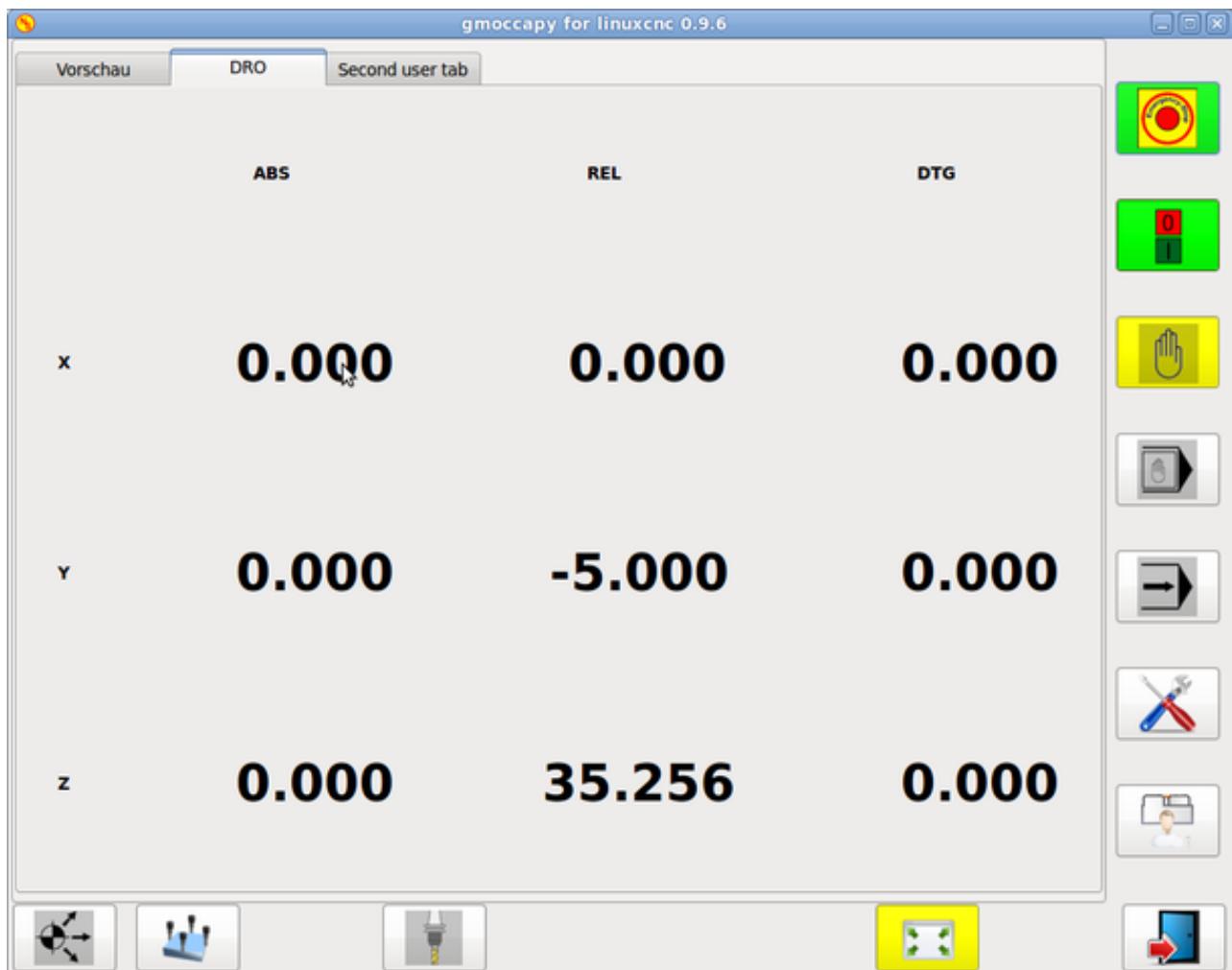
will add a the panel manual-example.ui, include a custom python handler, hitcounter.py and make all connections after realizing the panel according to manual-example.hal.

here are some examples:

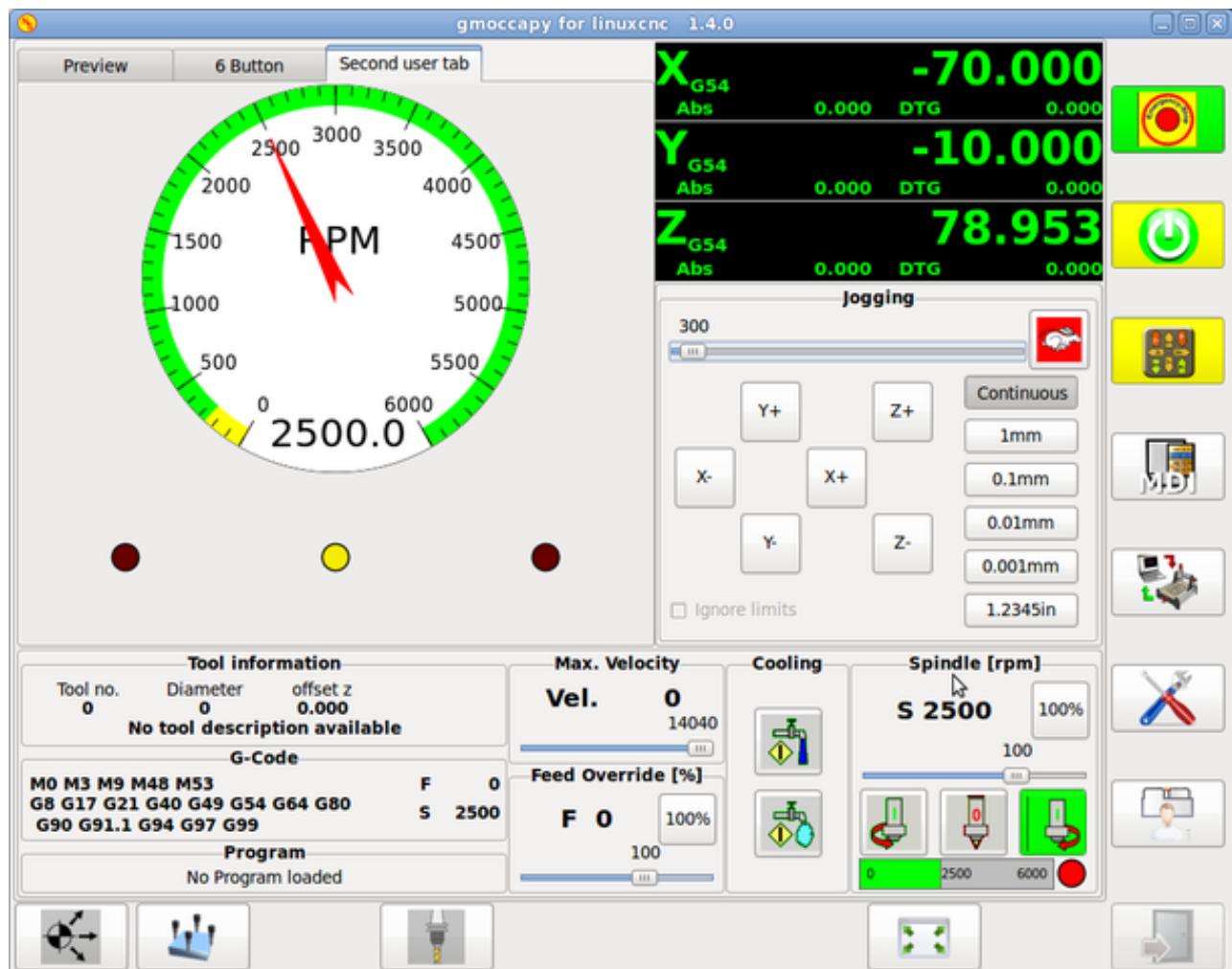
ntb_user_tabs - with integrated camview program



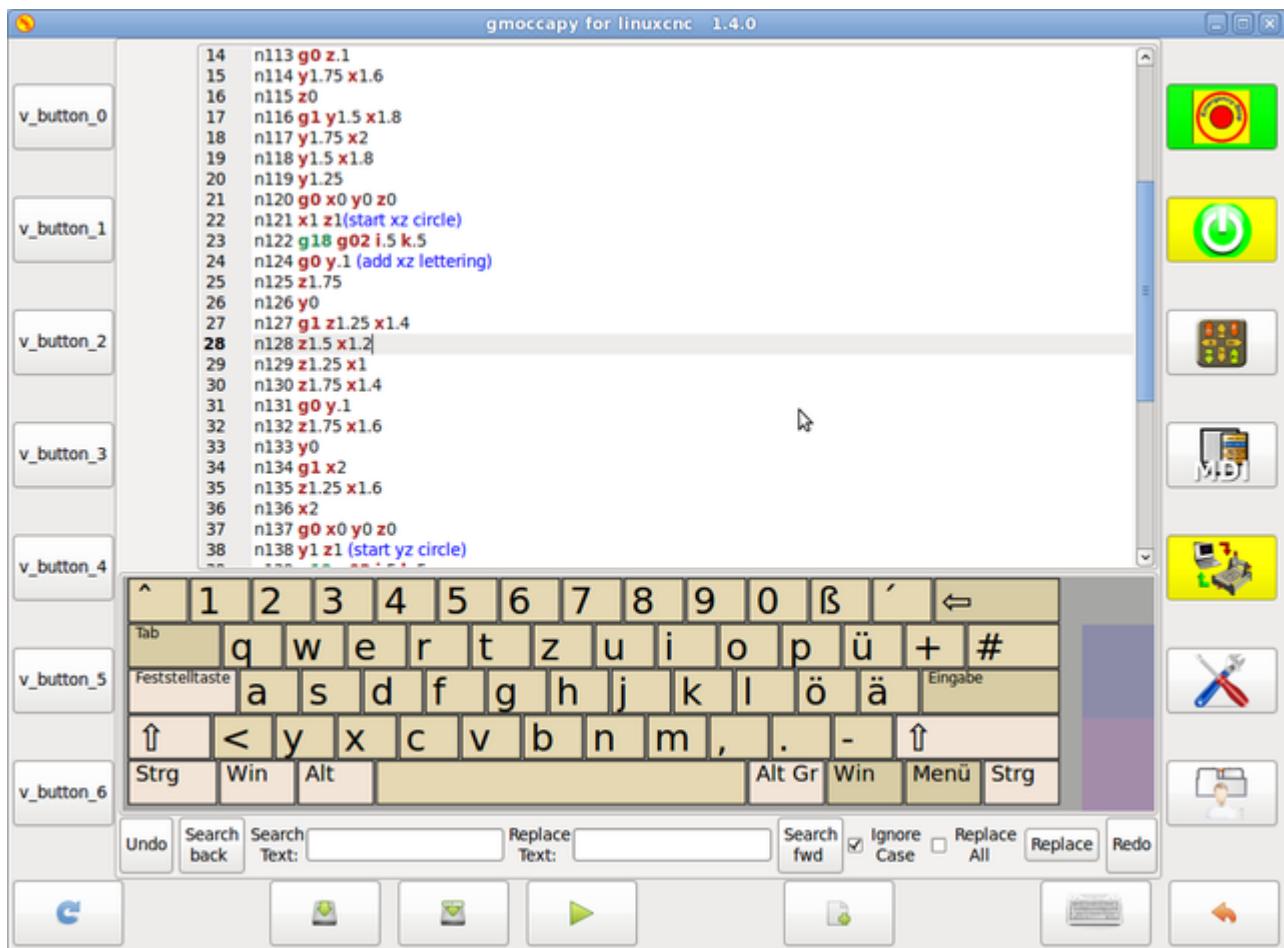
ntb_preview - as maximized version



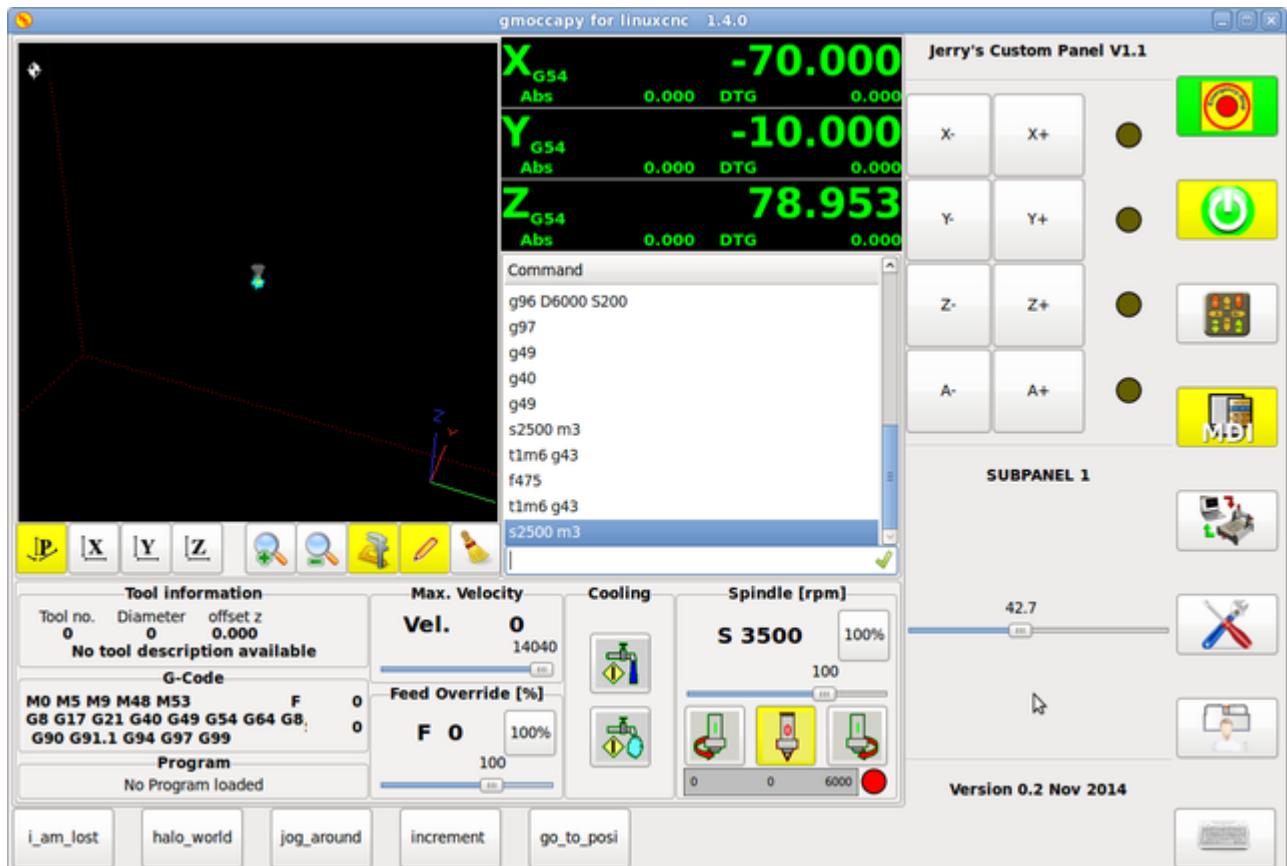
ntb_preview



box_left - showing gmoccapy in edit mode



box_right - and gmoccapy in MDI mode



6.4.1.2 Configuration of User Created Messages

Gmoccapy has the ability to create hal driven user messages. To use them you need to introduce some lines in the [DISPLAY] section of the INI file.

Here is how to set up 3 user popup message dialogs the messages support pango markup language. Detailed information about the markup language can be found at [Pango Markup](#)

```
MESSAGE_TEXT      = The text to be displayed, may be pango markup formated+
MESSAGE_TYPE     = "status" , "okdialog" , "yesnodialog"+
MESSAGE_PINNAME  = is the name of the hal pin group to be created+
```

- *status* : Will just display a message as popup window, using the messaging system of gmoccapy
- *okdialog* : Will hold focus on the message dialog and will activate a "-waiting" Hal_Pin OUT. Closing the message will reset the waiting pin
- *yesnodialog* : Will hold focus on the message dialog and will activate a "-waiting" Hal_Pin bit OUT it will also give access to an "-response" Hal_Pin Bit Out, this pin will hold 1 if the user clicks OK, and in all other states it will be 0 Closing the message will reset the waiting pin The response Hal Pin will remain 1 until the dialog is called again

Example

```

MESSAGE_TEXT = This is a <span background="#ff0000" foreground="#ffffff">info- ←
    message</span> test
MESSAGE_TYPE = status
MESSAGE_PINNAME = statustest

MESSAGE_TEXT = This is a yes no dialog test
MESSAGE_TYPE = yesnodialog
MESSAGE_PINNAME = yesnodialog

MESSAGE_TEXT = Text can be <small>small</small>, <big>big</big>, <b>bold</b>, ←
    <i>italic</i>, and even be <span color="red">colored</span>.
MESSAGE_TYPE = okdialog
MESSAGE_PINNAME = okdialog

```

The specific hal pin conventions for these can be found under the [User Messages hal pin section](#)

6.4.2 The RS274NGC Section

```
[RS274NGC]
SUBROUTINE_PATH = macros
```

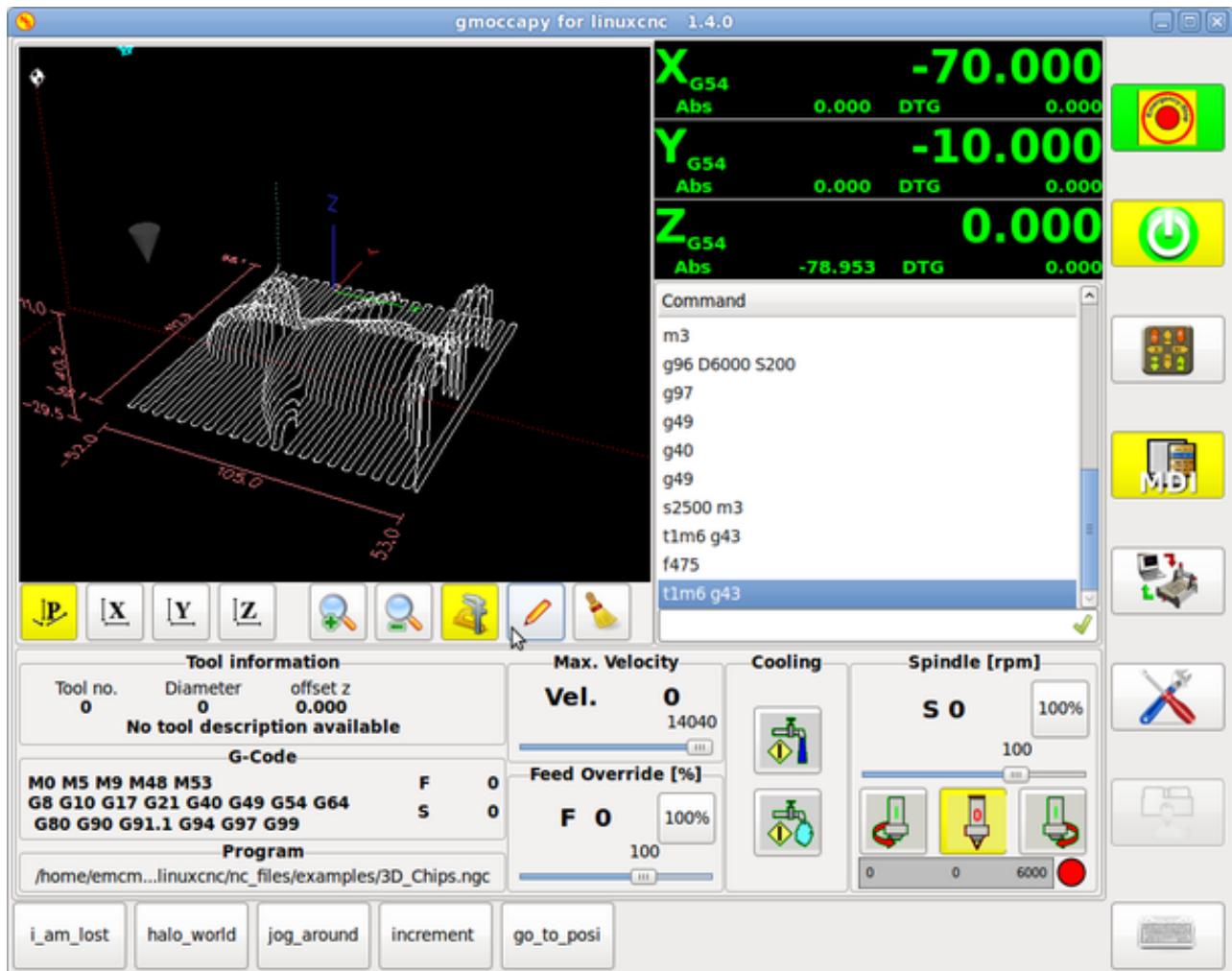
sets the path to search for macros and other subroutines.

6.4.3 The MACRO Section

You can add macros to gmoccapy, similar to touchy's way.
A macro is nothing else than a ngc-file. You are able to execute complete
CNC programs in MDI mode, by just pushing one button.
To do so, you have to add a section like so:

```
[MACROS]
MACRO = i_am_lost
MACRO = halo_world
MACRO = jog_around
MACRO = increment xinc yinc
MACRO = go_to_position X-pos Y-pos Z-pos
```

This will add 5 macros to the MDI button list. Please note, that maximal 9 macros will appear in the GUI, due to place reasons.
But it is no error placing more in your INI file.



The name of the file must be **exactly the same** as the name given in the MACRO line.
So the macro *i_am_lost* will call the file *i_am_lost.ngc*.

The macro files must follow some rules:

- the name of the file need to be the same as the name mentioned in the macro line, just with the ngc extension
- The file must contain a subroutine like so: *O<i_am_lost> sub*, the name of the sub must match exactly (**case sensitive**) the name of the macro
- the file must end with an endsub *O<i_am_lost> endsub* followed by an *M2* command
- the files need to be placed in a folder specified in your INI file in the RS274NGC section (see [RS274NGC](#))

The code in between sub and endsub will be executed by pushing the corresponding macro button.

Note

You will find the sample macros in macros folder placed in the gmoccapy sim folder.

Gmoccapy will also accept macros asking for parameters like:

```
go_to_position X-pos Y-pos Z-pos
```

The parameters must be separated by spaces. This calls a file *go_to_position.ngc* with the following content:

```
; Testfile go to position
; will jog the machine to a given position

O<go_to_position> sub

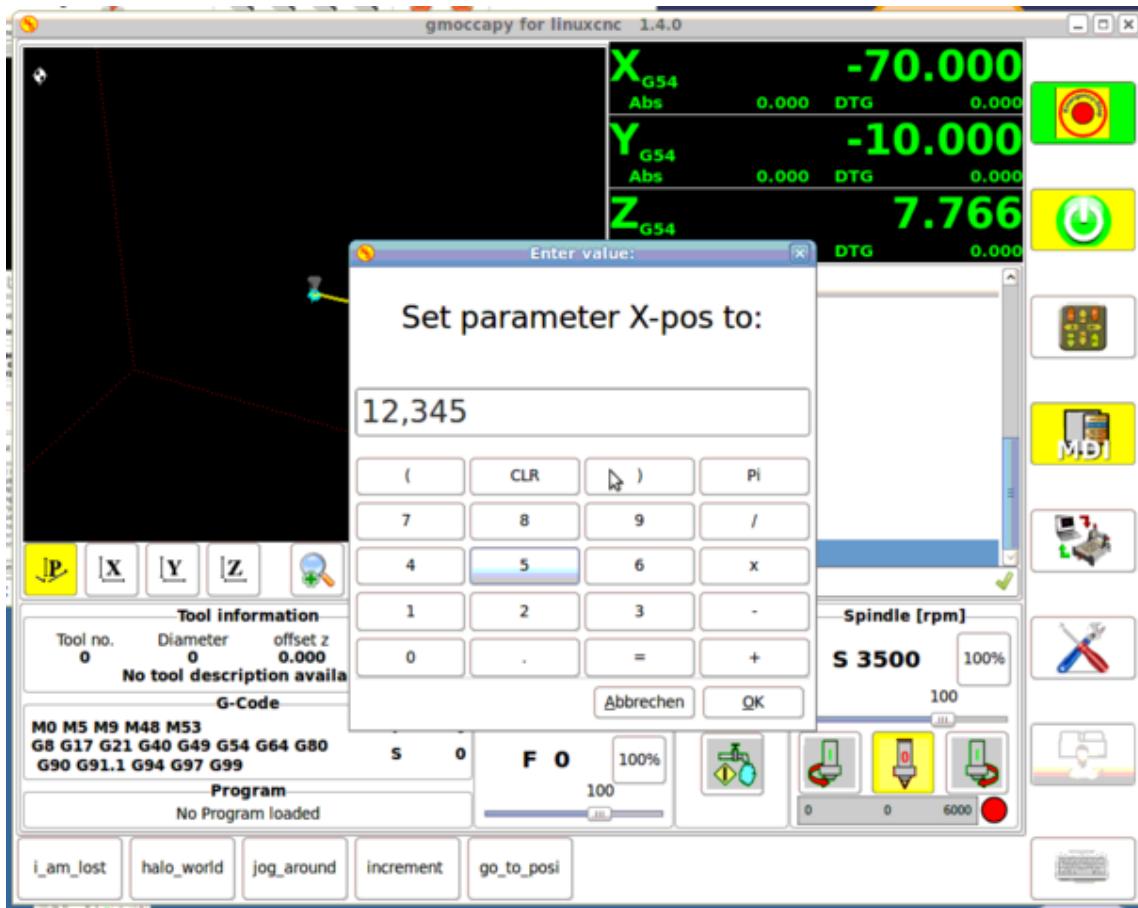
G17
G21
G54
G61
G40
G49
G80
G90

;#1 = <X-Pos>
;#2 = <Y-Pos>
;#3 = <Z-Pos>

(DBG, Will now move machine to X = #1 , Y = #2 , Z = #3)
G0 X #1 Y #2 Z #3

O<go_to_position> endsub
M2
```

after pushing the **execute macro button**, you will be asked to enter the values for **X-pos Y-pos Z-pos** and the macro will only run if all values have been given.



6.4.4 The TRAJ Section

MAX_VELOCITY = 230.000

Sets the maximal velocity of the machine, this value will also take influence to default velocity.

6.5 HAL Pins

gmoccapy exports several hal pin to be able to react to hardware devices.

The goal is to get a GUI that may be operated in a tool shop, completely/mostly without mouse or keyboard.

Note

You will have to do all connections to gmoccapy pins in your postgui.hal file, because they are not available before loading the GUI completely.

6.5.1 Right and bottom button lists

The screen has two main button lists, one on the right side and one on the bottom.

The right handed buttons will not change during operation, but the bottom button list will change very often. The buttons are count from up to down and from left to right beginning with "0".

In hal_show you will see the right (vertical) buttons are:

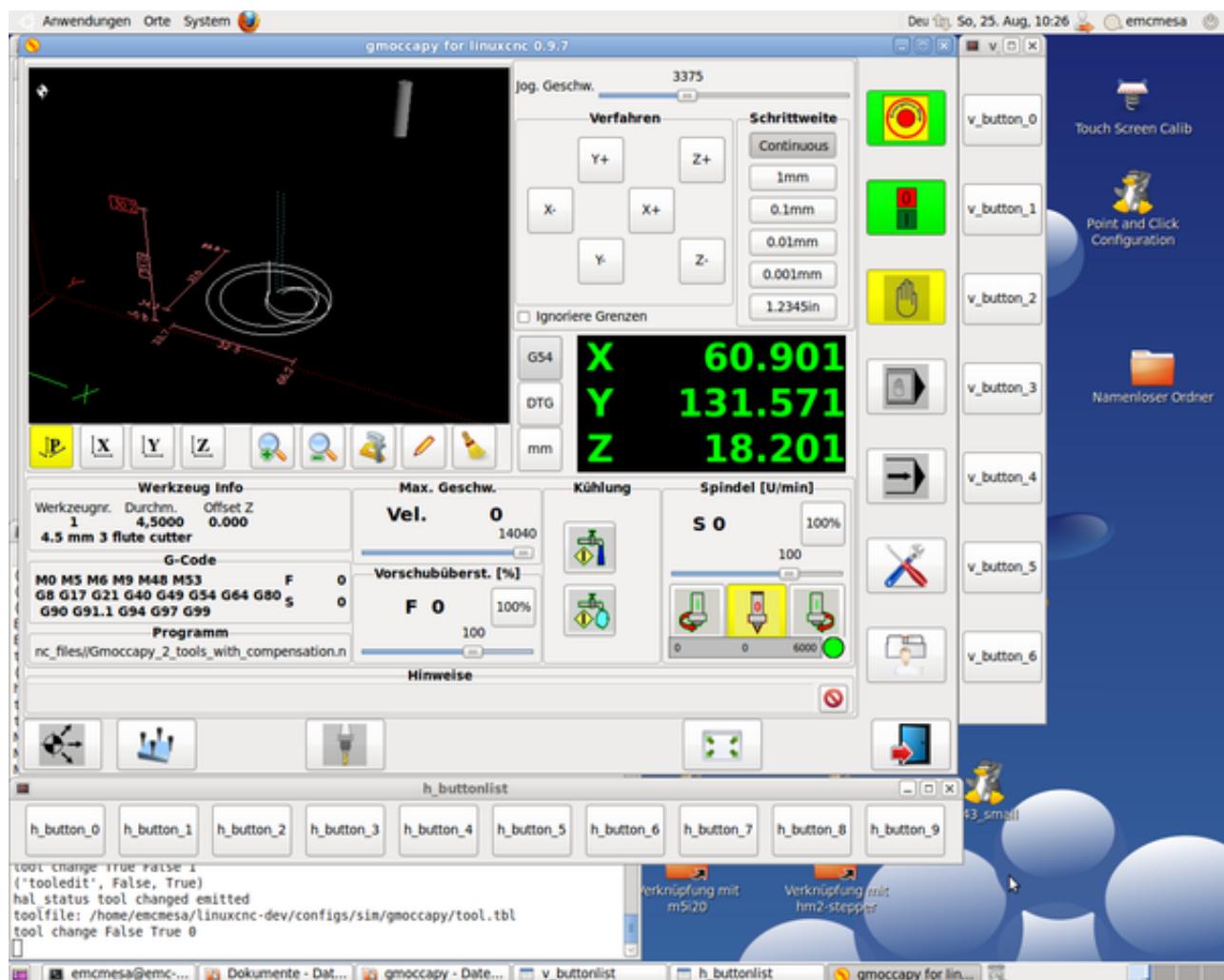
- gmoccapy.v-button-0
- gmoccapy.v-button-1
- gmoccapy.v-button-2
- gmoccapy.v-button-3
- gmoccapy.v-button-4
- gmoccapy.v-button-5
- gmoccapy.v-button-6

and the bottom (horizontal) buttons are:

- gmoccapy.h-button-0
- gmoccapy.h-button-1
- gmoccapy.h-button-2
- gmoccapy.h-button-3
- gmoccapy.h-button-4
- gmoccapy.h-button-5
- gmoccapy.h-button-6
- gmoccapy.h-button-7
- gmoccapy.h-button-8
- gmoccapy.h-button-9

as the buttons in the bottom list will change according the mode and other influences, the hardware buttons will activate different functions, and you don't have to take care about switching functions around in hal, because that is done completely by gmoccapy!

The sens of this is to be able to use the screen without an touch panel, or protect it from excessive use by placing hardware buttons around the panel.



6.5.2 Velocities and overrides

All sliders from gmoccapy can be connected to hardware encoder or hardware potmeters.

To connect encoders the following pin are exported:

- gmoccapy.max-vel-counts = HAL_S32 ; (Maximal Velocity of the machine)
- gmoccapy.jog-speed-counts = HAL_S32 ; (Jog velocity)
- gmoccapy.spindle-override-counts = HAL_S32 ; (spindle override)
- gmoccapy.feed-override-counts = HAL_S32 ; (feed override)
- gmoccapy.reset-feed-override = HAL_BIT ; (reset the feed override to 100 %)
- gmoccapy.reset-spindle-override = HAL_BIT ; (reset the spindle override to 100 %)

To connect potmeters, use the following hal pin:

- gmoccapy.analog-enable = HAL_BIT ; Must be True, to allow analog inputs
- gmoccapy.jog-vel-value = HAL_FLOAT ; To adjust the jog velocity slider
- gmoccapy.max-vel-value = HAL_FLOAT ; To adjust the max velocity slider
- gmoccapy.feed-override-value = HAL_FLOAT ; To adjust the feed override slider
- gmoccapy.spindle-override-value = HAL_FLOAT ; To adjust the spindle override slider

The float pin do accept values from 0.0 to 1.0, being the percentage value you want to set the slider value.

**Warning**

If you use both connection types, do not connect the same slider to both pin, as the influences between the two has not been tested! Different sliders may be connected to the one or other hal connection type.

**Important**

Please be aware, that for the jog velocity depends on the turtle button state, it will lead to different slider scales depending on the mode (turtle or rabbit). Please take also a look to [jog velocities and turtle-jog hal pin](#) for more details.

Example

```
Spindle Override Min Value = 20 %
Spindle Override Max Value = 120 %
gmoccapy.analog-enable = 1
gmoccapy.spindle-override-value = 0.25
```

```
value to set = Min Value + (Max Value - Min Value) * gmoccapy.spindle-override-value
value to set = 20 + (120 - 20) * 0.25
value to set = 45 %
```

6.5.3 jog hal pins

All axis given in the INI File have a jog-plus and a jog-minus pin, so hardware momentary switches can be used to jog the axis.

For the standard config following hal Pin will be available:

- gmoccapy.jog-x-plus
- gmoccapy.jog-x-minus
- gmoccapy.jog-y-plus
- gmoccapy.jog-y-minus
- gmoccapy.jog-z-plus
- gmoccapy.jog-z-minus

if you use a 4 axis INI file, there will be two additional pins

- gmoccapy.jog-<your fourth axis letter >-plus
- gmoccapy.jog-<your fourth axis letter >-minus

for a "C" axis you will see:

- gmoccapy.jog-c-plus
- gmoccapy.jog-c-minus

6.5.4 jog velocities and turtle-jog hal pin

The jog velocity can be selected with the corresponding slider.

The scale of the slider will be modified if the turtle button (the one showing a rabbit or a turtle) has been toggled. If the button is not visible, it might have been disabled on the [settings page](#). If the button shows the rabbit-icon, the scale is from min to max machine velocity. If it shows the turtle, the scale will reach only 1/20 of max velocity by default. The used divider can be set on the [settings page](#).

So using a touch screen it is much easier to select smaller velocities.

6.5.5 jog increment hal pins

The jog increments are selectable through hal pins, so a select hardware switch can be used to select the increment to use. There will be a maximum of 10 hal pin for the increments given in the INI File, if you give more increments in your INI File, they will be not reachable from the GUI as they will not be displayed.

If you have 6 increments in your hal you will get **7** pins:

- gmoccapy.jog-inc-0
- gmoccapy.jog-inc-1
- gmoccapy.jog-inc-2
- gmoccapy.jog-inc-3
- gmoccapy.jog-inc-4
- gmoccapy.jog-inc-5
- gmoccapy.jog-inc-6

jog-inc-0 is unchangeable and will represent continuous jogging.

6.5.6 hardware unlock pin

to be able to use a key switch to unlock the settings page the following pin is exported.

- gmoccapy.unlock-settings

The settings page is unlocked if the pin is high.

To use this pin, you need to activate it on the settings page.

6.5.7 Error pins

- gmoccapy.error
- gmoccapy.delete-message

gmoccapy.error is an bit out pin, to indicate an error, so a light can lit or even the machine may be stopped. It will be reseted with the pin gmoccapy.delete-message. gmoccapy.delete-message will delete the first error and reset the gmoccapy.error pin to False after the last error has been cleared.

Note

Messages or user infos will not affect the gmoccapy.error pin, but the gmoccapy.delete-message pin will delete the last message if no error is shown!

6.5.8 User Created Message HAL Pins

gmoccapy may react to external errors, using 3 different user messages:
All are HAL_BIT pin.

Status

- gmoccapy.messages.statustest

Yesnodialog

- gmoccapy.messages.yesnodialog
- gmoccapy.messages.yesnodialog-waiting
- gmoccapy.messages.yesnodialog-responce

Okdialog

- gmoccapy.messages.okdialog
- gmoccapy.messages.okdialog-waiting

6.5.9 Spindle feedback pins

There are two pins for spindle feedback

- gmoccapy.spindle_feedback_bar
- gmoccapy.spindle_at_speed_led

gmoccapy.spindle_feedback_bar will accept an float input to show the spindle speed
gmoccapy.spindle_at_speed_led is an bit-pin to lit the GUI led if spindle is at speed

6.5.10 Pins to indicate program progress information

There are three pins giving information over the program progress

- gmoccapy.program.length = HAL_S32 ; showing the total number of lines of the program
- gmoccapy.program.current-line = HAL_S32 ; indicating the current working line of the program
- gmoccapy.program.progress = HAL_FLOAT ; giving the program progress in percentage

The values may not be very accurate, if you are working with subroutines or large remap procedures, also loops will cause different values.

6.5.11 Pins to modify soft limits

Gmoccapy allows you to modify the soft limits using hal pin. The pin do allow to reduce or enlarge the soft limits, so you are able to protect your rotary table from collision just switching a hal pin. If you do not use it, just reduce your soft limits. An other option would be a tool changer which is placed in the working area, you do not want a collision during normal work, but you are forced to enter the tool change area to change a tool, so during tool change you enlarge the soft limits.

- gmoccapy.axis-to-set = HAL_S32 ; indicating the joint of the axis to modify (X=0, Y=1, Z=2 ...)
- gmoccapy.set-max-limit = HAL_BIT ; if set the value will modify the max limit value, else the min value
- gmoccapy.limit-value = HAL_FLOAT ; the new value to set as soft limit

This has not been tested with a rotary axis, so be careful with that kind of axis. There will be no check to the limits you set, so if you set a limit and the tool is out of your limit area you will get an error about being outside the soft limits.



Important

You are responsible to take care about that!

You are only allowed to set limits if the machine is on! A value of 0 as limit is not allowed, please give a value of 0.000001 instead. The GUI will react to changes of the limit-value pin, so to modify the limits you have to apply a value change to that pin.



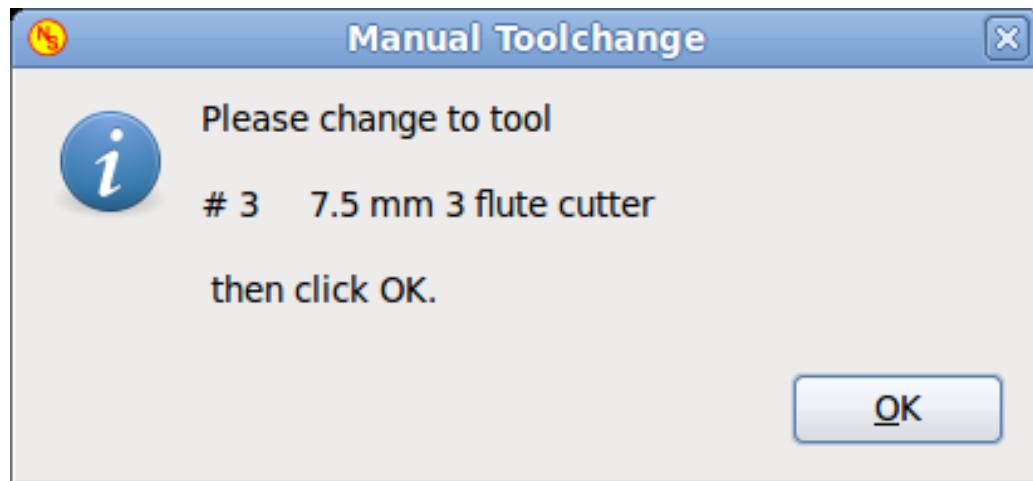
Warning

This is very experimental at the moment, so please take care for your security.

6.5.12 Tool related pin

6.5.12.1 Tool Change Pin

This pin are provided to use gmoccapy's internal tool change dialog, similar to the one known from axis, but with several modifications, so you will not only get the message to change to *tool number 3*, but also the description of that tool like *7.5 mm 3 flute cutter*. The information is taken from the tool table, so it is up to you what to display.



- gmoccapy.toolchange-number = HAL_S32 ; The number of the tool to be changed
- gmoccapy.toolchange-change = HAL_BIT ; Indicate that a tool has to be changed
- gmoccapy.toolchange-changed = HAL_BIT ; Indicate toll has been changed

usually they are connected like this for a manual tool change:

```
net tool-change          gmoccapy.toolchange-change    <=  iocontrol.0.tool-change
net tool-changed         gmoccapy.toolchange-changed  <=  iocontrol.0.tool-changed
net tool-prep-number     gmoccapy.toolchange-number   <=  iocontrol.0.tool-prep-number
net tool-prep-loop       iocontrol.0.tool-prepare      <=  iocontrol.0.tool-prepared
```

6.5.12.2 tool offset pins

This pins allows you to show the active tool offset values for X and Z in the tool information frame. You should know that they are only active after G43 has been send.

Tool information				
Tool no.	Diameter	offset z	offset x	
1	0,4000	0.017	1.161	
		60 Grad vorn		

- gmoccapy.tooloffset-x
- gmoccapy.tooloffset-z

just connect them like so in your postgui hal.

```
net tooloffset-x gmoccapy.tooloffset-x <= motion.tooloffset.x
net tooloffset-z gmoccapy.tooloffset-z <= motion.tooloffset.z
```

Please note, that gmoccapy takes care of its own to update the offsets, sending an G43 after any tool change, **but not in auto mode!**



Important

So writing a program makes you responsible to include an G43 after each tool change!

6.6 Auto Tool Measurement

Gmoccapy offers an integrated auto tool measurement.

To use this feature, you will need to do some additional settings and you may want to use the offered hal pin to get values in your own ngc remap procedure.



Important

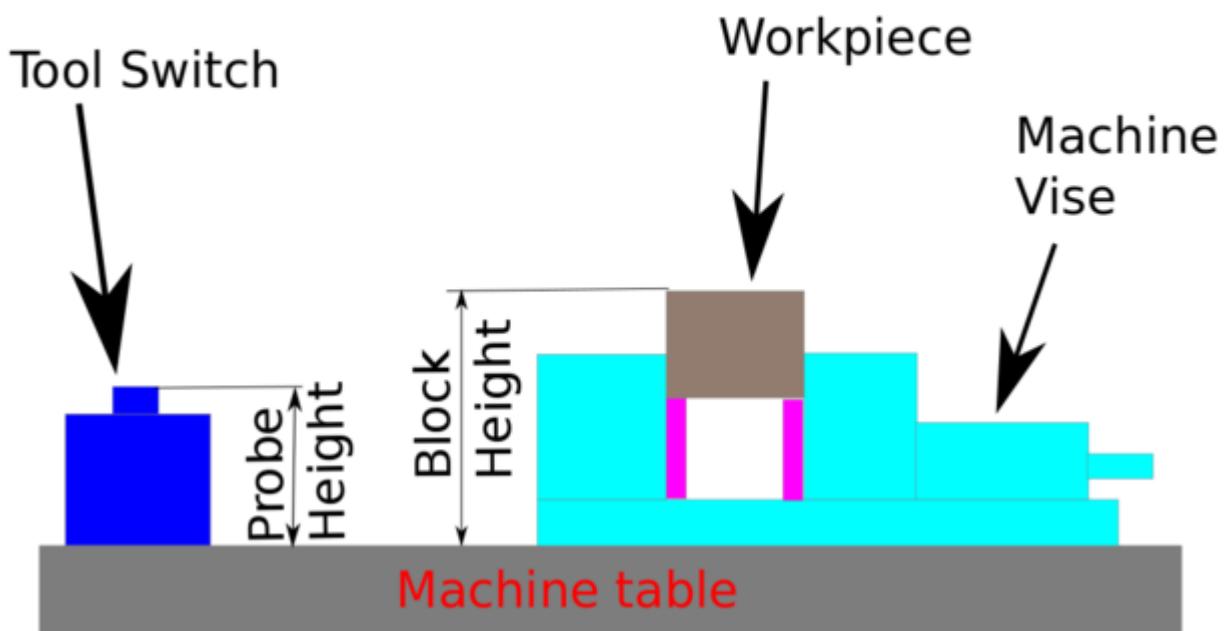
Before starting the first test, do not forget to enter the Probe height and probe velocities on the settings page! See [Settings Page Tool Measurement](#)

It might be also a good idea to take a look at the tool measurement video: see [tool measurement related videos](#)

Tool Measurement in gmoccapy is done a little bit different to many other GUI.

You should follow these steps: * touch of your workpiece in X and Y * measure the height of your block from the base where your tool switch is located, to the upper face of the block (including chuck etc.) * Push the button block height and enter the measured value * Go to auto mode and start your program

here is a small sketch:



With the first given tool change the tool will be measured and the offset will be set automatically to fit the block height. The advantage of the gmoccapy way is, that you do not need a reference tool.

Note

Your program must contain a tool change at the beginning! The tool will be measured, even it has been used before, so there is no danger, if the block height has changed. There are several videos showing the way to do that on you tube.

6.6.1 Tool measurement pins

Gmoccapy offers 5 pins for tool measurement purpose

The pins are mostly used to be read from a gcode subroutine, so the code can react to different values.

- gmoccapy.toolmeasurement = HAL_BIT ; enable or not tool measurement
- gmoccapy.blockheight = HAL_FLOAT ; the measured value of the top face of the workpiece
- gmoccapy.probeheight = HAL_FLOAT ; the probe switch height
- gmoccapy.searchvel = HAL_FLOAT ; the velocity to search for the tool probe switch
- gmoccapy.probevel = HAL_FLOAT ; the velocity to probe tool length

6.6.2 Tool Measurement INI File modifications

Modify your INI File to include the following:

6.6.2.1 The RS274NGC section

```
[RS274NGC]
# Enables the reading of INI and HAL values from gcode
FEATURES=12

# is the sub, with is called when a error during tool change happens
ON_ABORT_COMMAND=0 <on_abort> call

# The remap code
REMAP=M6 modalgroup=6 prolog=change_prolog ngc=change epilog=change_epilog
```

6.6.2.2 The tool sensor section

The position of the tool sensor and the start position of the probing movement, all values are absolute coordinates, except MAXPROBE, what must be given in relative movement.

```
[TOOLSENSOR]
X = 10
Y = 10
Z = -20
MAXPROBE = -20
```

6.6.2.3 The Change position section

this is not named TOOL_CHANGE_POSITION on purpose - canon uses that name and will interfere otherwise
The position to move the machine before giving the change tool command. All values are in absolute coordinates

```
[CHANGE_POSITION]
X = 10
Y = 10
Z = -2
```

6.6.2.4 The Python section

the Python plugins serves interpreter and task

```
[PYTHON]
# The path to start a search for user modules
PATH_PREPEND = python
# The start point for all.
TOLEVEL = python/toplevel.py
```

6.6.3 Needed Files

You must copy the following files to your config dir

First make a directory *python* in your config folder
 from *your_linuxcnc-dev_directory/configs/sim/gmoccapy/python*
 Copy *toplevel.py* to your *config_dir/python* folder
 Copy *remap.py* to your *config_dir/python* folder
 Copy *stdglue.py* to your *config_dir/python* folder

from *your_linuxcnc-dev_directory/configs/sim/gmoccapy/macros*
 copy *on_abort.ngc* to the directory specified as SUBROUTINE_PATH see [RS274NGC Section](#)
 from *your_linuxcnc-dev_directory/configs/sim/gmoccapy/macros*
 copy *change.ngc* to the directory specified as SUBROUTINE_PATH see [RS274NGC Section](#)
 open *change.ngc* with a editor and uncomment the following lines (49 and 50):

```
F #<_hal[gmoccapy.probevel]>
G38.2 Z-4
```

You may want to modify this file to fit more your needs, feel free, but do not ask for support ;-)

6.6.4 Needed Hal connections

connect the tool probe in your hal file like so

```
net probe motion.probe-input <= <your_input_pin>
```

The line might look like this:

```
net probe motion.probe-input <= parport.0.pin-15-in
```

In your postgui.hal file add:

```
# The next lines are only needed if the pins had been connected before
unlinkp iocontrol.0.tool-change
unlinkp iocontrol.0.tool-changed
unlinkp iocontrol.0.tool-prep-number
unlinkp iocontrol.0.tool-prepared

# link to gmoccapy toolchange, so you get the advantage of tool description on change ←
# dialog
net tool-change          gmoccapy.toolchange-change    <=  iocontrol.0.tool-change
net tool-changed         gmoccapy.toolchange-changed  <=  iocontrol.0.tool-changed
net tool-prep-number     gmoccapy.toolchange-number   <=  iocontrol.0.tool-prep-number
net tool-prep-loop       iocontrol.0.tool-prepare      <=  iocontrol.0.tool-prepared
```

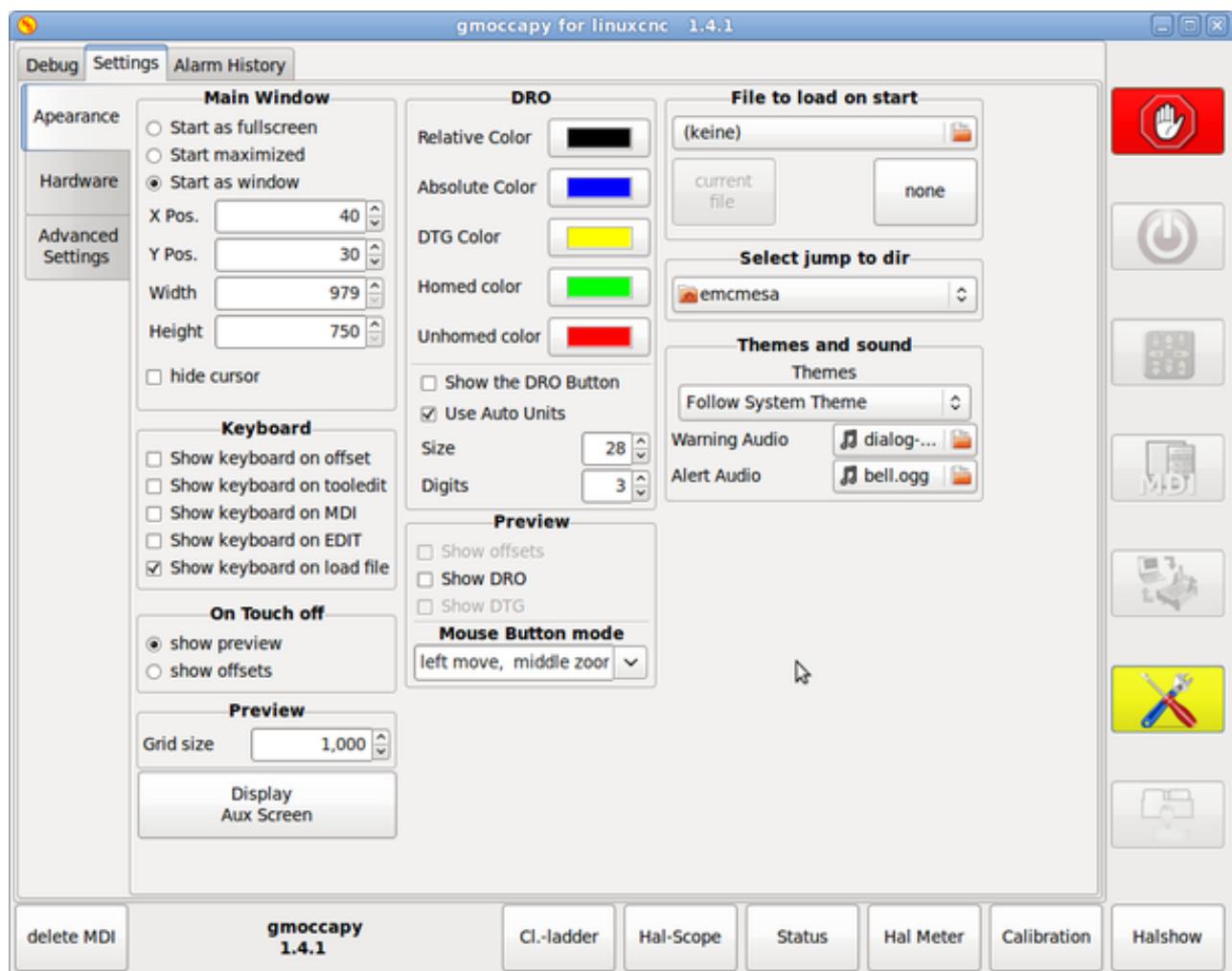
6.7 The settings page

To enter the page you will have to click on:



and give an unlock code, which is **123** as default. If you want to change it at this time you will have to edit the hidden preference file, see [the display section](#) for details.

The page looks at the moment like so:



The page is separated in three main tabs:

6.7.1 Appearance

on this tab you will find the following options:

6.7.1.1 Main Window

Here you can select how you wish the GUI to start. The main reason for this was the wish to get an easy way for the user to set the starting options without the need to touch code.

You have three options:

- start as fullscreen
- start maximized
- start as window

If you select start as window the spinboxes to set the position and size will get active. One time set, the GUI will start every time on the place and with the size selected. Nevertheless the user can change the size and position using the mouse, but that will not have any influence on the settings.

hide the cursor does allow to hide the cursor, what is very useful if you use a touch screen.

6.7.1.2 Keyboard

Note

If this section is not sensitive, you have not installed a virtual keyboard, supported are **onboard** and **matchbox-keyboard**.

The check-boxes allows the user to select if he want the on board keyboard to be shown immediately, when entering the MDI Mode, when entering the offset page, the tooledit widget or when open a program in the EDIT mode. The keyboard button on the bottom button list will not been affected by this settings, so you be able to show or hide the keyboard by pressing the button. The default behavior will be set by the check-boxes.

Default are :

- show keyboard on offset = True
- show keyboard on tooledit = False
- show keyboard on MDI = True
- show keyboard on EDIT = True
- show keyboard on load file = False

If the keyboard layout is not correct, i.e. clicking X gives Z, than the layout has not been set properly, related to your locale settings.

For onboard it can be solved with a small batch file with the following content:

```
#!/bin/bash
setxkbmap -model pc105 -layout de -variant basic
```

the letters "de" are for German, you will have to set them according to your locale settings

Just execute this file before starting LinuxCNC, it can be done also adding a starter to your local folder

```
./config/autostart
```

so that the layout is set automatically on starting.

For matchbox-keyboard you will have to make your own layout, for a German layout ask in the forum.

6.7.1.3 On Touch Off

give the option to show the preview tab or the offset page tab if you enter the touch off mode by clicking the corresponding bottom button.

- show preview
- show offsets

As the notebook tabs are shown, you are able to switch between both views in any case.

6.7.1.4 DRO Options

You have the option to select the background colors of the different DRO states. So users suffering from protanopia (red/green weakness) are able to select proper colors

By default the backgrounds are:

- Relative mode = black
- Absolute mode = blue
- Distance to go = yellow

and the foreground color of the DRO can be selected with:

- homed color = green
- unhomed color = red

show dro in preview, the DRO will be shown in the preview window

show offsets, the Offsets will be shown in the preview window

show DTG , the distance to go will be shown in the preview window +

show DRO Button will allow you to display additional buttons on the left side of the DRO.

It will display:

one button to switch from relative to absolute coordinates,

one button to toggle between distance to go and the other states

and one button to toggle the units from metric to imperial and vice versa.



Warning

It is not recommended to use this option, because the user will lose the auto unit option, which will toggle the units according to the active gcode G20 / G21

Note

You can change through the DRO modes (absolute, relative, distance to go) by clicking on the DRO!

Use Auto Units allows to disable the auto units option of the display, so you can run a program in inches and watch the DRO in mm.

size allows to set the size of the DRO font, default is 28, if you use a bigger screen you may want to increase the size up to 56. If you do use 4 axis, the DRO font size will be 3/4 of the value, because of space reason.

digits sets the number of digits of the DRO from 1 to 5. NOTE: Imperial will show one digit more than metric. So if you are in imperial machine units and set the digit value to 1, you will get no digit at all in metric.

6.7.1.5 Preview

Grid Size

Sets the grid size of the preview window.

Unfortunately the size **has to be set in inches**, even if your machine units are metric. We do hope to fix that in a future release.

Note

The grid will not be shown in perspective view.

Show DRO

Will show the a DRO also in the preview window, it will be shown automatically in fullsize preview

Show DTG

Will show also the DTG (direct distance to end point) in the preview, only if Show DRO is active and not fullsize preview.

Show Offsets

Will show the offsets in the preview window,

Note

If you only check this option and leave the others unchecked, you will get in fullsize preview a offset page

Mouse Button Mode

With this combobox you can select the button behavior of the mouse to rotate, move or zoom within the preview.

- left rotate, middle move, right zoom
- left zoom, middle move, right rotate
- left move, middle rotate, right zoom
- left zoom, middle rotate, right move
- left move, middle zoom, right rotate
- left rotate, middle zoom, right move

Default is left move, middle zoom, right rotate

The mouse wheel will still zoom the preview in every mode.

Tip

If you select an element in the preview, the selected element will be taken as rotation center point.

6.7.1.6 File to load on start up

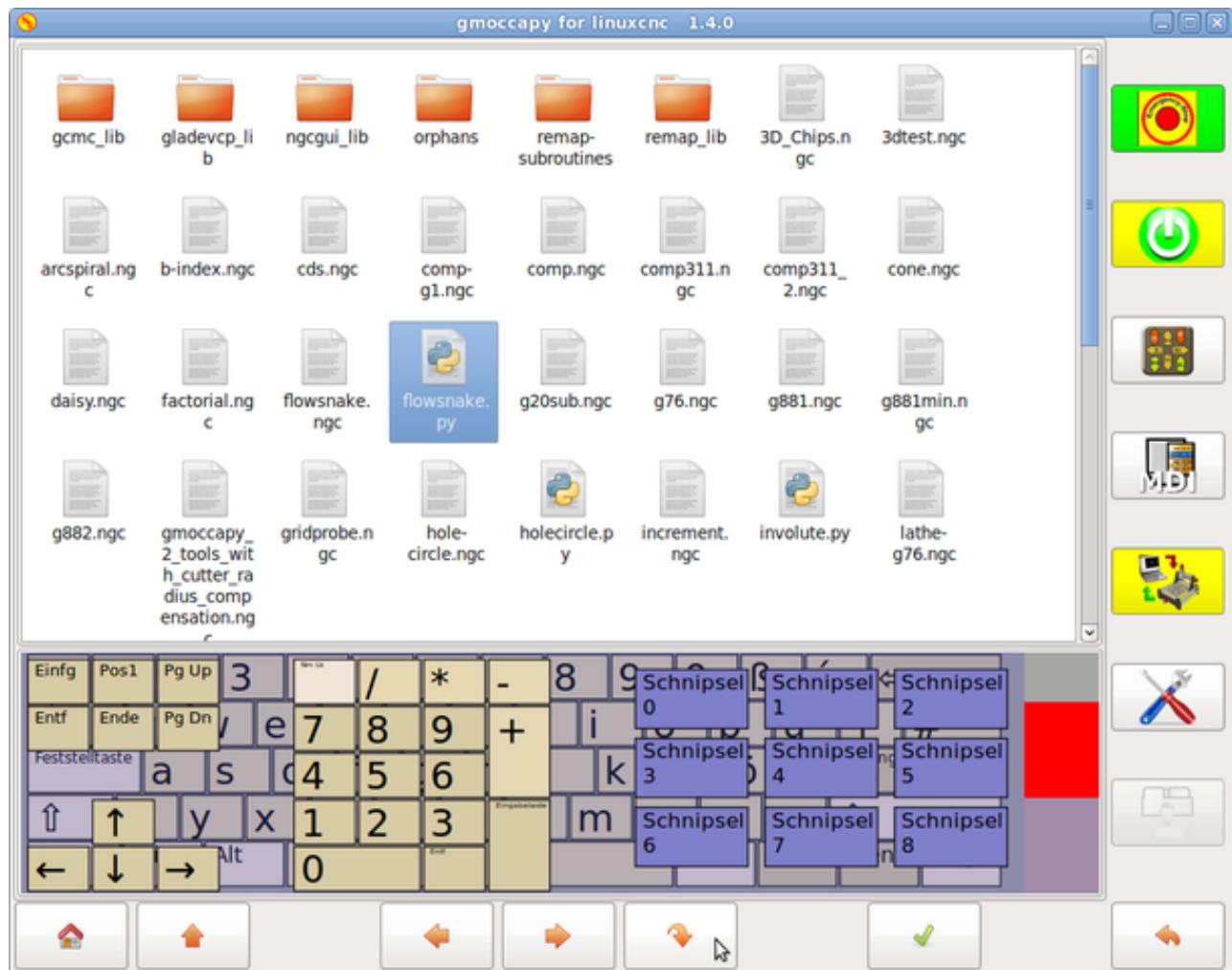
Select the file you want to be loaded on start up. In other GUI changing this was very cumbersome, because the users where forced to edit the INI File.

If a file is loaded, it can be set by pressing the current button to avoid that any program is loaded at start up, just press the None button.

The file selection screen will use the filters you have set in the INI File, if there aren't any filters given, you will only see **ngc** files. The path will be set according to the INI settings in [DISPLAY] PROGRAM_PREFIX

6.7.1.7 Jump to dir

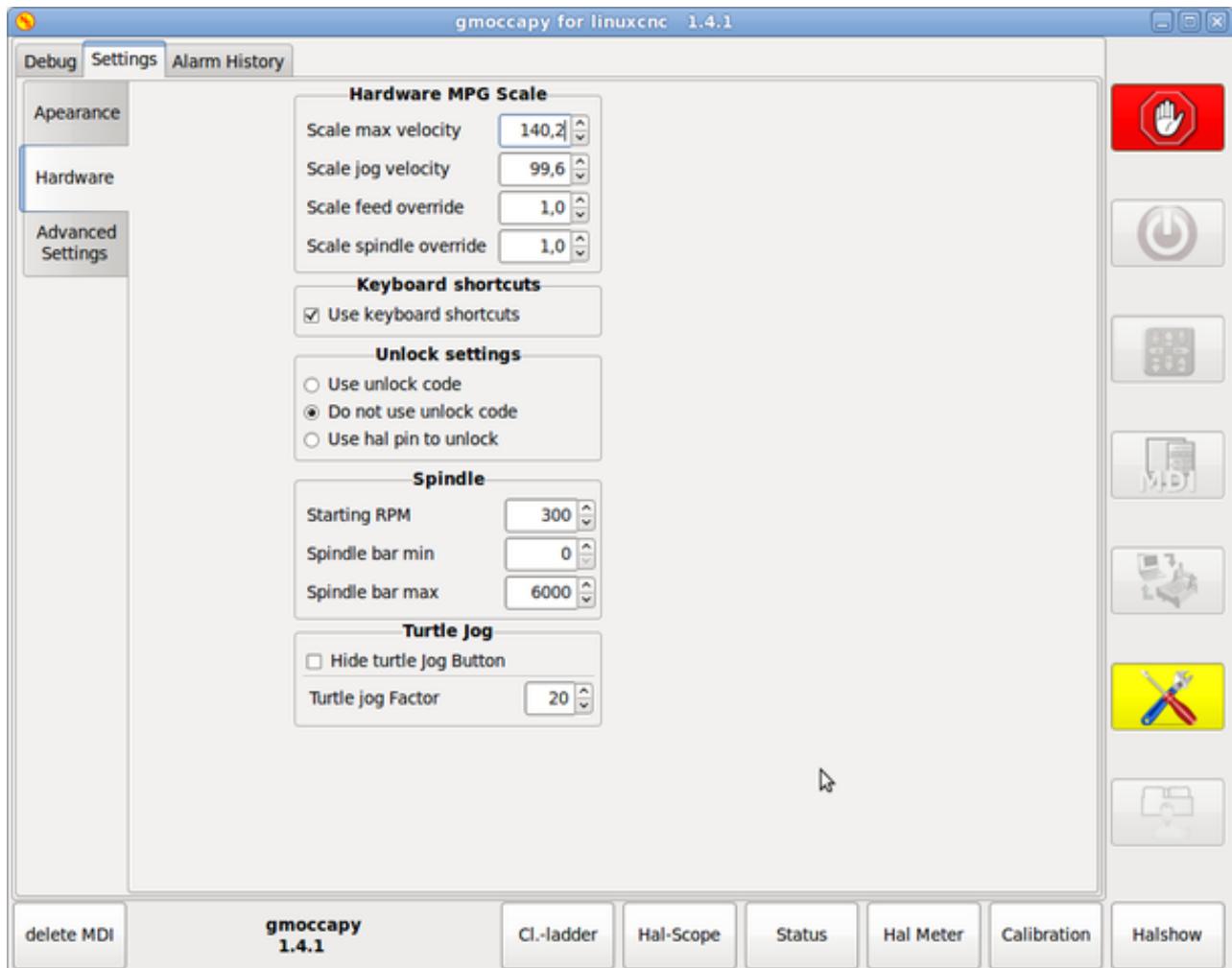
you can set here the directory to jump to if you press the corresponding button in the file selection dialog.



6.7.1.8 Themes and Sounds

This lets the user select what desktop theme to apply and what error and messages sounds should be played. By default "Follow System Theme" is set.

6.7.2 Hardware



6.7.2.1 Hardware MPG Scales

For the different Hal Pin to connect MPG Wheels to, you may select individual scales to be applied. The main reason for this was my own test to solve this through hal connections, resulting in a very complex hal file. Imagine a user having an MPG Wheel with 100 ipr and he wants to slow down the max vel from 14000 to 2000 mm/min, that needs 12000 impulses, resulting in 120 turns of the wheel! Or an other user having a MPG Wheel with 500 ipr and he wants to set the spindle override which has limits from 50 to 120 % so he goes from min to max within 70 impulses, meaning not even 1/4 turn.

By default all scales are set using the calculation:

$$(\text{MAX} - \text{MIN}) / 100$$

6.7.2.2 Keyboard shortcuts

Some users want to jog their machine using the keyboard buttons and there are others that will never allow this. So everybody can select whether to use them or not.

Default is to use keyboard shortcuts.

Please take care if you use a lathe, than the shortcuts will be different. See [the Lathe section](#)

- Arrow Left = X minus
- Arrow Right = X plus
- Arrow up = Y plus
- Arrow Down = Y minus
- Page Up = Z plus
- Page Down = Z minus
- F1 = Estop (will work even if keyboard shortcuts are disabled)
- F2 = Machine on
- ESC = Abort

There are additional keys for message handling, see [Message behavior and appearance](#) * WINDOWS = Delete last message * <STRG><SPACE> = Delete all messages

6.7.2.3 Unlock options

you have three options to unlock the settings page:

- use unlock code (the user must give a code to get in)
- Do not use unlock code (There will be no security check, not recommended)
- Use hal pin to unlock (hardware pin must be high to unlock the settings, see [hardware unlock pin](#))

Default is use unlock code (default = **123**)

6.7.2.4 Spindle

The start RPM sets the rpm to be used if the spindle is started and no S value has been set.

With the MIN and MAX settings you set the limits of the spindle bar shown in the INFO frame on the main screen. It is no error giving wrong values. If you give a maximum of 2000 and your spindle makes 4000 rpm, only the bar level will be wrong on higher speeds than 2000 rpm.

default values are
MIN = 0
MAX = 6000

6.7.2.5 Turtle Jog

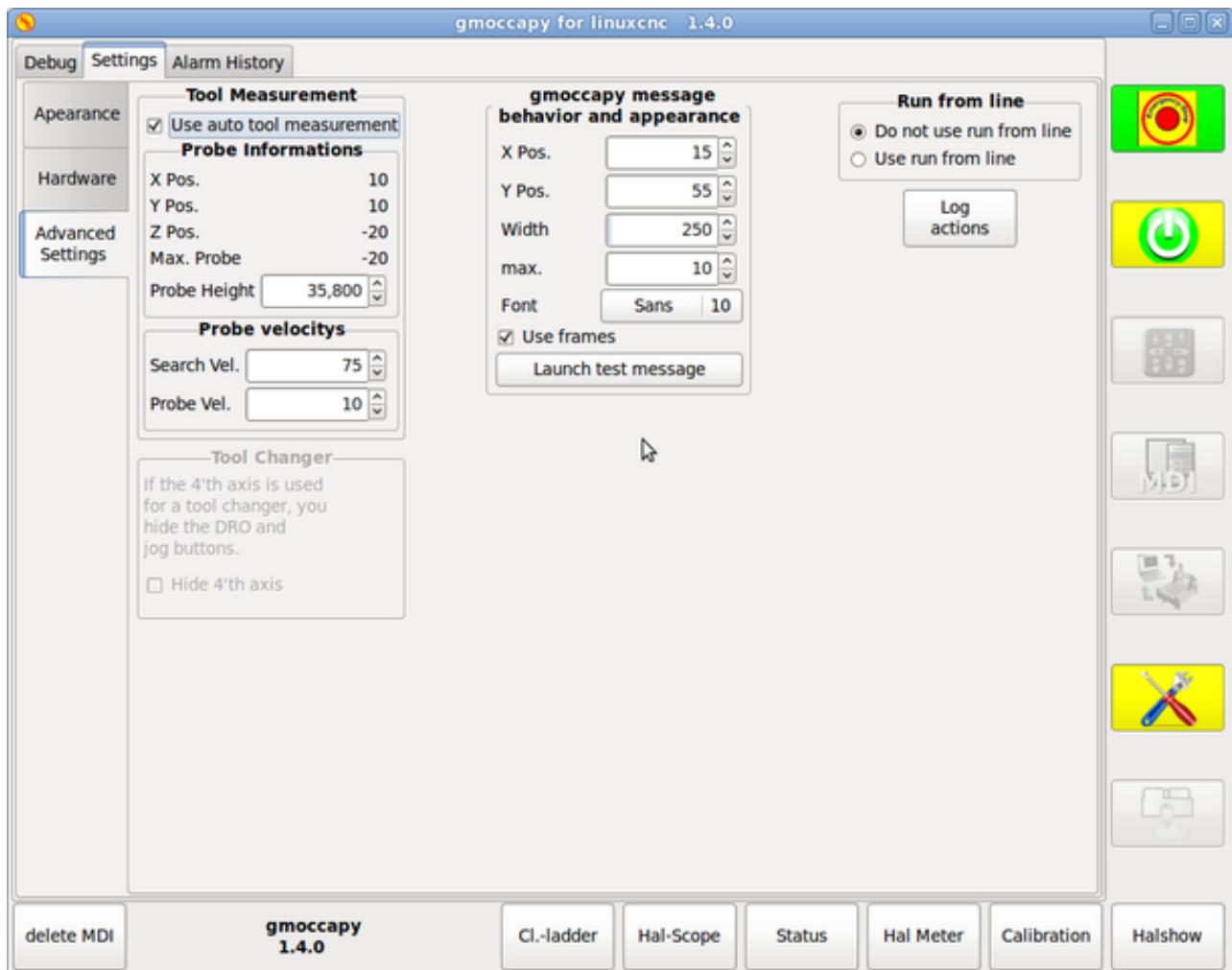
This settings will have influence on the jog velocities.

- *hide turtle jog button* will hide the button right of the jog velocity slider, if you hide this button, please take care that it shows the rabbit icon, otherwise you will not be able to jog faster than the turtle jog velocity, which is calculated using the turtle jog factor.
- *Turtle jog factor* sets the scale to apply for turtle jog mode. If you set a factor of 20, the max jog velocity will be 1/20 of max velocity of the machine if in turtle mode (button pressed, showing the turtle)

Note

This button can be activated using the [turtle-jog](#) hal pin.

6.7.3 Advanced Settings



6.7.3.1 Tool Measurement

If this part is not sensitive, you do not have a valid INI file configuration to use tool measurement.

Please check [Auto Tool Measurement](#)

- Use auto tool measurement : If checked, after each tool change, a tool measurement will be done, the result will be stored in the tool table and an G43 will be executed after the change.

Probe informations

The following informations are taken from your INI file and must be given in absolute coordinates

- X Pos. = The X position of the tool switch
- Y Pos. = The Y position of the tool switch
- Z Pos. = The Z position of the tool switch, we will go as rapid move to this coordinate
- Max. Probe = is the distance to search for contact, an error will be launched, if no contact is given. The distance has to be given in relative coordinates, beginning the move from Z Pos., so you have to give a negative value to go down!

- Probe Height = is the height of your probe switch, you can measure it. Just touch off the base, where the probe switch is located and set that to zero. Then make a tool change and watch the tool_offset_z value, that is the hight you must enter here.

Probe velocities

- Search Vel. = The velocity to search for the tool switch, after contact the tool will go up again and then goes toward the probe again with probe vel, so you will get better results.
- Probe Vel. = Is the velocity for the second movement to the switch, it should be slower to get better touch results.(In sim mode, this is commented out in macros/change.ngc, otherwise the user would have to click twice on the probe button)

Tool Changer

If your 4'th axis is used in a tool changer, you may want to hide the DRO and all the other buttons related to that axis.

You can do that by checking the checkbox, that will hide:

- 4'th axis DRO
- 4'th axis Jog button
- 4'th axis home button
- column of 4'th axis in the offsetpage
- column of 4'th axis in the tolleditor

6.7.3.2 Message behavior and appearance

This will display small popup windows displaying the message or error text, the behavior is very similar to the one axis uses. You can delete a specific message, by clicking on its close button, if you want to delete the last one, just hit the WINDOWS key on your keyboard, or delete all messages at ones with <STRG><SPACE>.

You are able to set some options:

- X Pos = The position of the top left corner of the message in X counted in pixel from the top left corner of the screen.
- Y Pos = The position of the top left corner of the message in Y counted in pixel from the top left corner of the screen.
- Width = The width of the message box
- max = The maximum messages you want to see at ones, if you set this to 10, the 11th message will delete the first one, so you will only see the last 10 ones.
- Font = The font and size you want to use to display the messages
- use frames = If you activate the checkbox, each message will be displayed in a frame, so it is much easier to distinguish the messages. But you will need a little bit more space.
- The button launch test message will just do what it is supposed to, it will show a message, so you can see the changes of your settings without the need to generate an error.

6.7.3.3 Run from line option

You can allow or disallow the run from line. This will set the corresponding button insensitive (grayed out), so the user will not be able to use this option.



Warning

It is not recommended to use run from line, as LinuxCNC will not take care of any previous lines in the code before the starting line. So errors or crashes are very probable.

Default is disable run from line

6.7.3.4 Log Actions

If this button is active, nearly every button press or relevant action of LinuxCNC will be logged in the ALARM history. This is very useful for debugging.

6.8 LATHE specific section

If in the INI File LATHE = 1 is given, the GUI will change its appearance to the special needs for a lathe. Mainly the Y axis will be hidden and the jog buttons will be arranged in a different order.

Normal Lathe:



Back Tool Lathe:



As you see the R DRO has a black background and the D DRO is gray. This will change according to the active G-Code G7 or G8. The active mode is visible by the black background, meaning in the shown images G8 is active.

The next difference to the standard screen is the location of the Jog Button. X and Z have changed places and Y is gone. You will note that the X+ and X- buttons changes there places according to normal or back tool lathe.

Also the keyboard behavior will change:

Normal Lathe:

- Arrow Left = Z minus
- Arrow Right = Z plus
- Arrow up = X minus
- Arrow Down = X plus

Back Tool Lathe:

- Arrow Left = Z minus
- Arrow Right = Z plus
- Arrow up = X plus

- Arrow Down = X minus

The tool information frame will show not only the Z offset, but also the X offset and the tool table is showing all lathe relevant information.

6.9 Plasma specific section



There is a very good WIKI, which is actually growing, maintained by Marius
see [Plasma wiki page](#)

6.10 VIDEO on you tube

This are videos showing gmoccapy in action, unfortunately the videos don't show the latest version of gmoccapy, but the way to use it will not change much in the future. I will try to actualize the videos as soon as possible.

6.10.1 Basic Usage

German =

English = <https://www.youtube.com/watch?v=O5B-s3uiI6g>

6.10.2 Simulated Jog Wheels

English = <http://youtu.be/ag34SGxt97o>

6.10.3 Settings Page

German Settings =

English Settings = <https://www.youtube.com/watch?v=AuwhSHRJoiI>

6.10.4 Simulated Hardware Button

German Hardware_Button = <http://www.youtube.com/watch?v=DTqhY-MfzDE>

English Hardware Button = <http://www.youtube.com/watch?v=ItVWJBK9WFA>

6.10.5 User Tabs

English User Tabs = <http://www.youtube.com/watch?v=rG1zmeqXyZI>

6.10.6 Tool_Measurement_Video

English Auto Tool Measurement Simmulation = <http://youtu.be/rkkMw6rUFdk>

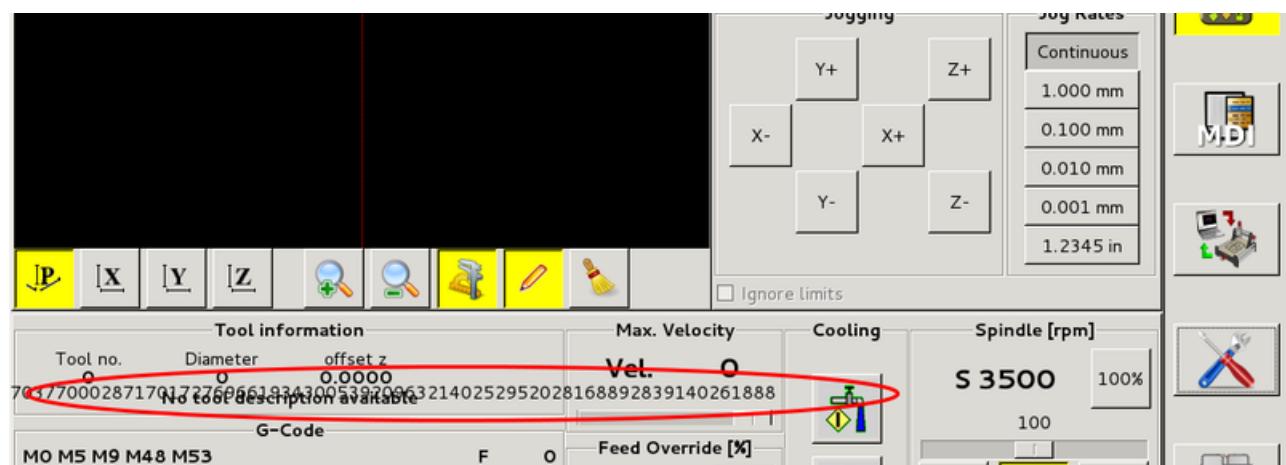
English Auto Tool Measurement Screen = <http://youtu.be/Z2ULDj9dzvk>

English Auto Tool Measurement Machine = <http://youtu.be/1arucCaDdX4>

6.11 Known problems

6.11.1 Strange numbers in the info area

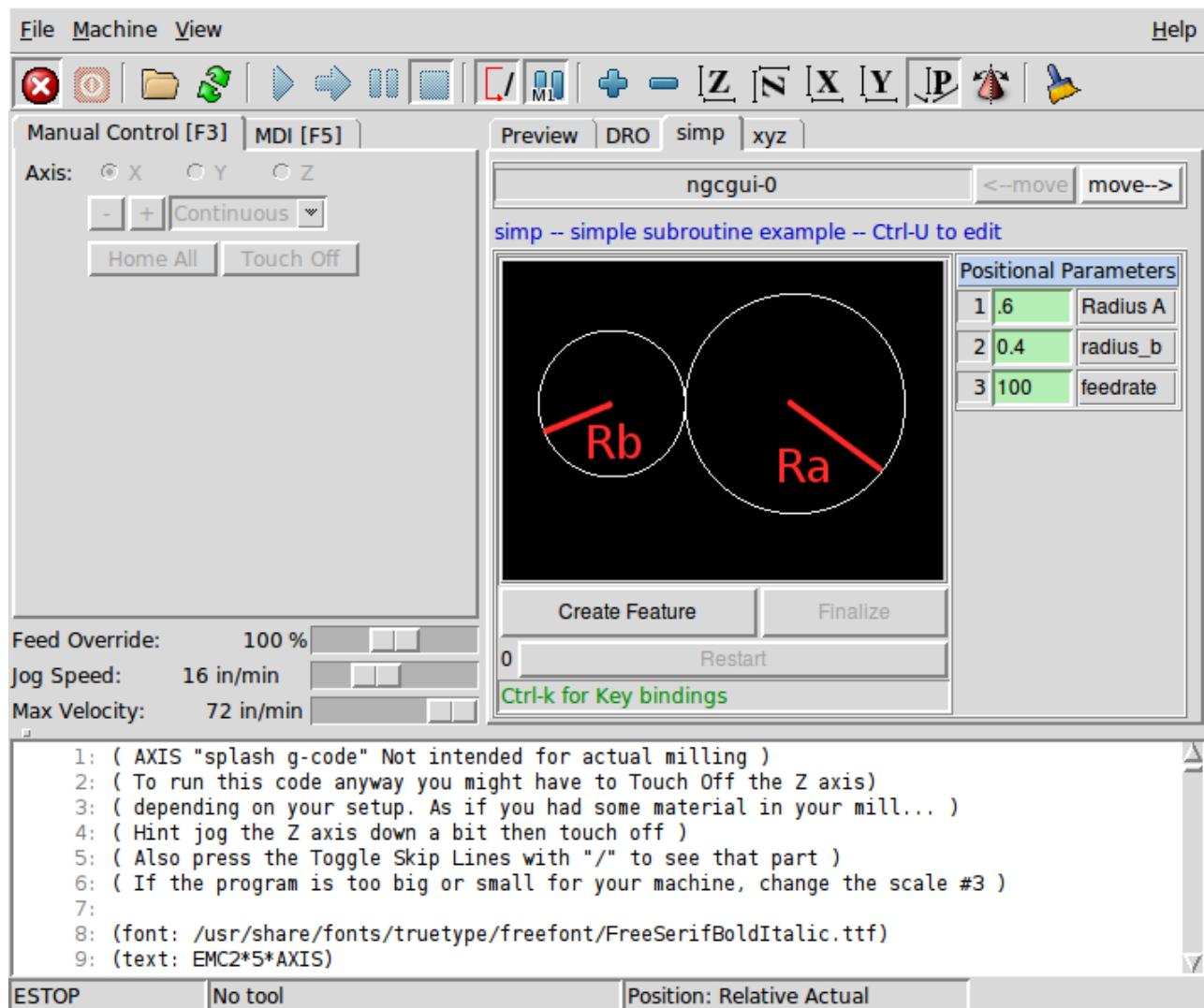
If you get strange numbers in the info area of gmoccapy like:



You have made your config file using an older version of StepConfWizard. It has made a wrong entry in the INI file under the [TRAJ] named MAX_LINEAR_VELOCITY = xxx. Change that entry to MAX_VELOCITY = xxx

Chapter 7

NGCGUI



7.1 Overview

- *NGCGUI* is a tcl application for using LinuxCNC subroutines.
- *NGCGUI* can run as a standalone application or can be embedded in multiple tab pages in the axis gui
- *PYNGCGUI* is an alternate, python implementation of ngcgui.
- *PYNGCGUI* can run as a standalone application or can be embedded as a tab page (with its own set of multiple subroutine tabs) in any gui that supports embedding of gladevcp applications. Example guis include: axis, touchy, gscreen and gmoccapy

Using NGCGUI or PYNGCGUI:

- Tab pages are provided for subroutines specified in the INI file
- New subroutines tab pages can be added using Custom tab pages
- Each subroutine tab page provides entry boxes for all subroutine parameters
- The entry boxes can have a default value and an label that are identified by special comments in the subroutine file
- Subroutine invocations are concatenated together to form a multiple step program
- Any single-file gcode subroutine that conforms to ngcgui conventions can be used
- Any gcmc (Gcode-meta-compiler) program that conforms to ngcgui conventions for tagging variables can be used. (The gcmc executable must be installed separately, see: <http://www.vagrearg.org/content/gcmc>)

Note

NGCGUI and PYNGCGUI implement the same functions and both process .ngc and .gcmc files that conform to a few ngcgui-specific conventions. In this document, the term NGCGUI generally refers to either application.

7.2 Demonstration Configurations

A number of demonstration configurations are located in the sim directory of the Sample Configurations offered by the LinuxCNC configuration picker. The configuration picker is on the system's main menu:

Applications > CNC > LinuxCNC

Examples are included for the axis, touchy, gscreen, and gmoccapy guis. These examples demonstrate both 3-axis (XYZ) cartesian configurations (like mills) and lathe (XZ) setups. Some examples show the use of a popupkeyboard for touch screen systems and other examples demonstrate the use of files created for the gcmc (Gcode Meta Compiler) application. The touchy examples also demonstrate incorporation of a gladevcp backplot viewer (gremlin_view).

The simplest application is found as:

```
Sample Configurations / sim / axis / ngcgui /ngcgui_simple
```

A comprehensive example showing gcmc compatibility is at:

```
Sample Configurations / sim / axis / ngcgui / ngcgui_gcmc
```

A comprehensive example embedded as a gladevcp app and using gcmc is at:

```
Sample Configurations / sim / gscreen / ngcgui / pyngcgui_gcmc
```

The example sim configurations make use of library files that provide example gcode subroutine (.ngc) files and Gcode-meta-compiler (.gcmc) files:

- *nc_files/ngcgui_lib*
 - *arc1.ngc* - basic arc using cutter radius compensation
 - *arc2.ngc* - arc speced by center, offset, width, angle (calls arc1)
 - *backlash.ngc* - routine to measure an axis backlash with dial indicator
 - *db25.ngc* - creates a DB25 plug cutout
 - *gosper.ngc* - a recursion demo (flowsnake)
 - *helix.ngc* - helix or D-hole cutting
 - *helix_rtheta.ngc* - helix or D-hole positioned by radius and angle
 - *hole_circle.ngc* - equally spaced holes on a circle
 - *ihex.ngc* - internal hexagon
 - *iquad.ngc* - internal quadrilateral
 - *ohex.ngc* - outside hexagon
 - *oquad.ngc* - outside quadrilateral
 - *qpex_mm.ngc* - demo of qpockets (mm based)
 - *qpex.ngc* - demo of qpockets (inch based)
 - *qpocket.ngc* - quadrilateral pocket
 - *rectangle_probe.ngc* - probe a rectangular area
 - *simp.ngc* - a simple subroutine example that creates two circles
 - *slot.ngc* - slot from connecting two endpoints
 - *xyz.ngc* - machine exerciser constrained to a box shape
- *nc_files/ngcgui_lib/lathe*
 - *g76base.ngc* - gui for g76 threading
 - *g76diam.ngc* - threading speced by major, minor diameters
 - *id.ngc* - bores the inside diameter
 - *od.ngc* - turns the outside diameter
 - *taper-od.ngc* - turns a taper on the outside diameter
- *nc_files/gcmc_lib*
 - *drill.gcmc* - drill holes in rectangle pattern
 - *square.gcmc* - simple demo of variable tags for gcmc files
 - *star.gcmc* - gcmc demo illustrating functions and arrays
 - *wheels.gcmc* - gcmc demo of complex patterns

To try a demonstration, select a sim configuration and start the linuxCNC program.

If using the axis gui, press the *E-Stop*  then *Machine Power*  then *Home All*. Pick a ngcgui tab, fill in any empty blanks with sensible values and press *Create Feature* then *Finalize*. Finally press the *Run*  button to watch it run. Experiment by creating multiple features and features from different tab pages.

Other guis will have similar functionality but the buttons and names may be different.

Notes

The demonstration configs create tab pages for just a few of the provided examples. Any gui with a Custom tab page can open any of the library example subroutines or any user file if it is in the linuxCNC subroutine path.

To see special key bindings, click inside an ngcgui tab page to get focus and then presss Control-k.

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.

7.3 Library Locations

In linuxCNC installations installed from deb packages, the simulation configs for ngcgui use symbolic links to non-user-writable LinuxCNC libraries for:

- *nc_files/ngcgui_lib* ncgui-compatible subfiles
- *nc_files/ngcgui_lib/lathe* ncgui-compatible lathe subfiles
- *nc_files/gcmc_lib* ncgui-gcmc-compatible programs
- *nc_files/ngcgui_lib/utilitysubs* Helper subroutines
- *nc_files/ngcgui_lib/mfiles* User M files

These libraries are located by ini file items that specify the search paths used by linuxCNC (and ncogui):

```
[RS274NGC]
SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/gcmc_lib:../../nc_files/ ←
    ncgui_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 (:). No spaces should occur between directory names.

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 with the commands:

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

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

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

To use files in new directories, the ini file must be edited to include the new subfiles and to augment the search path(s). For this example:

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

[DISPLAY]
...
NGCGUI_SUBFILE = example.ngc
...
```

LinuxCNC (and ncogui) use the first file found when searching directories in the search path. With this behavior, you can supersede an ncogui_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.

7.4 Standalone Usage

7.4.1 Standalone NGCGUI

For usage, type in a terminal:

```
ngcgui --help
Usage:
  ngcgui --help | -?
  ngcgui [Options] -D nc_files_directory_name
  ngcgui [Options] -i LinuxCNC_inifile_name
  ngcgui [Options]

Options:
  [-S subroutine_file]
  [-p preamble_file]
  [-P postamble_file]
  [-o output_file]
  [-a autosend_file]           (autosend to axis default:auto.ngc)
  [--noauto]                   (no autosend to axis)
  [-N | --nom2]                (no m2 terminator (use %))
  [--font [big|small|fontspec]] (default: "Helvetica -10 normal")
  [--horiz|--vert]             (default: --horiz)
  [--cwidth comment_width]     (width of comment field)
  [--vwidth varname_width]     (width of varname field)
  [--quiet]                    (fewer comments in outfile)
  [--noiframe]                 (default: frame displays image)
```

Note

As a standalone application, ngcgui handles a single subroutine file which can be invoked multiple times. Multiple standalone ngcgui applications can be started independently.

7.4.2 Standalone PYNGCGUI

For usage, type in a terminal:

```
pyngcgui --help
Usage:
  pyngcgui [Options] [sub_filename]
Options requiring values:
  [-d | --demo] [0|1|2] (0: DEMO standalone toplevel)
                           (1: DEMO embed new notebook)
                           (2: DEMO embed within existing notebook)
  [-S | --subfile] sub_filename
  [-p | --preamble] preamble_filename
  [-P | --postamble] postamble_filename
  [-i | --ini] ini_filename
  [-a | --autofile] autoauto_filename
  [-t | --test] testno
  [-H | --height] height_of_entry_widget] (typ 20-40)
  [-K | --keyboardfile] glade_file] (use custom popupkeyboard glade file)
Solo Options:
  [-v | --verbose]
  [-D | --debug]
  [-N | --nom2]           (no m2 terminator (use %))
  [-n | --noauto]          (save but do not automatically send result)
  [-k | --keyboard]        (use default popupkeybaord)
```

```
[-s | --sendtoaxis]    (send generated ngc file to axis gui)
Notes:
A set of files is comprised of a preamble, subfile, postamble.
The preamble and postamble are optional.
One set of files can be specified from cmdline.
Multiple sets of files can be specified from an ini file.
If --ini is NOT specified:
    search for a running linuxCNC and use it's ini file
```

Note

As a standalone application, pyngcgui can read an ini file (or a running linuxCNC application) to create tab pages for multiple subfiles.

7.5 Embedding NGCGUI

7.5.1 Embedding NGCGUI in Axis

The following INI file items go in the [DISPLAY] section. (See additional sections below for additional items needed)

- *TKPKG = Ngcgui 1.0* - the NGCGUI package
- *TKPKG = Ngcguittt 1.0* - the True Type Tracer package for generating text for engraving (optional, must follow TKPKG = Ngcgui).
- *TTT = truetype-tracer* - name of the truetype tracer program (it must be in user PATH)
- *TTT_PREAMBLE = in_std.ngc* - Optional, specifies filename for preamble used for ttt created subfiles. (alternate: mm_std.ngc)

Note

The optional truetype tracer items are used to specify an ngcgui-compatible tab page that uses the application truetype-tracer. The truetype-tracer application must be installed independently and located in the user PATH.

7.5.2 Embedding PYNGCGUI as a gladevcp tab page in a gui

The following INI file items go in the [DISPLAY] section for use with the axis, gscreen, or touchy guis. (See additional sections below for additional items needed)

EMBED_ Items

```
EMBED_TAB_NAME = Pyngcgui - name to appear on embedded tab
EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui - invokes gladevcp
EMBED_TAB_LOCATION = name_of_location - where the embeded page is located
```

Note

The EMBED_TAB_LOCATION specifier is not used for the axis gui. While pyngcgui can be embedded in axis, integration is more complete when using ngcgui (using TKPKG = Ngcgui 1.0). To specify the EMBED_TAB_LOCATION for other guis, see the example ini files.

Note

The truetype tracer gui front-end is not currently available for gladevcp applications.

7.5.3 Additional INI File items required for ncgui or pyngcgui

The following INI file items go in the [DISPLAY] section for any gui that embeds either ncgui or pyngcgui.

- *NGCGUI_FONT* = *Helvetica -12 normal* - specifies the font name, size, normal/bold
- *NGCGUI_PREAMBLE* = *in_std.ngc* - the preamble file to be added in front of the subroutines. When concatenating several common subroutine invocations, this preamble is only added once. For mm-based machines, use *mm_std.ngc*
- *NGCGUI_SUBFILE* = *filename1.ngc* - creates a tab from the *filename1* subroutine
- *NGCGUI_SUBFILE* = *filename2.ngc* - creates a tab from the *filename2* subroutine
- ... etc
- *NGCGUI_SUBFILE* = *gcmcname1.gcmc* - creates a tab from the *gcmcname1* file
- *NGCGUI_SUBFILE* = *gcmcname2.gcmc* - creates a tab from the *gcmcname2* file
- ... etc
- *NGCGUI_SUBFILE* = "" - creates a custom tab that can open any subroutine in the search path
- *NGCGUI_OPTIONS* = *opt1 opt2 ...* - NGCGUI options
 - *nonew* - disallow making a new custom tab
 - *noremove* - disallow removing any tab page
 - *noauto* - no autosend (use *makeFile*, then *save* or manually send)
 - *noiframe* - no internal image, display images on separate top level widget
 - *nom2* - do not terminate with *m2*, use % terminator. This option eliminates all the side effects of *m2* termination
- *GCMC_INCLUDE_PATH* = *dirname1:dirname2* - search directories for gcmc include files

This is an example of embedding NGCGUI into Axis. The subroutines need to be in a directory specified by the [RS274NGC]SUBROUTINE_PATH item. 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 item.

The Gcode-meta-compiler (gcmc) can include statements like: include("filename.inc.gcmc"); By default, gcmc includes the current directory which, for linuxCNC, will be the directory containing the linuxCNC ini file. Additional directories can be prepended to the gcmc search order with the GCMC_INCLUDE_PATH item.

Sample axis-gui-based INI

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

[DISPLAY]
TKPKG              = Ncgui      1.0
TKPKG              = Ngcguittt 1.0
# Ncgui must precede Ngcguittt

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:
#   none    -- 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
GCMC_INCLUDE_PATH = /home/myname/gcmc_includes

TTT                  = truetype-tracer
TTT_PREAMBLE        = in_std.ngc

PROGRAM_PREFIX       = ../../nc_files

```

Note

The above is not a complete axis gui INI—the items shown are those used by ngcgui. Many additional items are required by LinuxCNC to have a complete INI file.

7.5.4 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 be 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.

7.5.5 INI File Path Specifications

Ngcgui uses the linuxCNC search path to find files.

The search path begins with the standard directory specified by:

[DISPLAY] PROGRAM_PREFIX = directory_name

followed by multiple directories specified by:

```
[RS274NGC]SUBROUTINE_PATH = directory1_name:directory1_name:directory3_name ...
```

Directories may be specified as absolute paths or relative paths.

Example: [DISPLAY]PROGRAM_PREFIX = /home/myname/linuxcnc/nc_files

Example: [DISPLAY]PROGRAM_PREFIX = ~/linuxcnc/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.

Multiple Directories Example:

```
[RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ngcgui_lib/utilitysubs ←
                           :/tmp/tmpngc`
```

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

LinuxCNC (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/linuxcnc/mysubs will be found before files in ../../nc_files/ngcgui_lib.

Adding User Directory Example:

```
[RS274NGC]SUBROUTINE_PATH = /home/myname/linuxcnc/mysubs:../../nc_files/ngcgui_lib:../../ ←
                           nc_files/ngcgui_lib/utilitysubs`
```

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.

7.5.6 Summary of INI File item details for NGCGUI usage

Item: [RS274NGC]SUBROUTINE_PATH = dirname1:dirname2:dirname3 ...

Example: [RS274NGC]SUBROUTINE_PATH = ../../nc_files/ngcgui_lib:../../nc_files/ ←
 ngcgui_lib/utilitysubs

Note: Optional, but very useful to organize subfiles and utility files

Item: [RS274NGC]USER_M_PATH = dirname1:dirname2:dirname3 ...
Example: [RS274NGC]USER_M_PATH = ../../nc_files/ngcgui_lib/mfiles
Note: Optional, needed to locate custom user mfiles

Item: [DISPLAY]EMBED_TAB_NAME = name to display on embedded tab page
Example: [DISPLAY]EMBED_TAB_NAME = Pyngcgui
Note: The entries: EMBED_TAB_NAME, EMBED_TAB_COMMAND, EMBED_TAB_LOCATION define an embedded application for several linuxCNC guis

Item: [DISPLAY]EMBED_TAB_COMMAND = programname followed by arguments
Example: [DISPLAY]EMBED_TAB_COMMAND = gladevcp -x {XID} pyngcgui_axis.ui
Note: For gladevcp applications, see the man page for gladevcp

Item: [DISPLAY]EMBED_TAB_LOCATION = name_of_location
Example: [DISPLAY]EMBED_TAB_LOCATION = notebook_main
Note: See example INI files for possible locations
Not required for the axis gui

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

item: [DISPLAY]TKPKG = Ngcgui version_number
Example: [DISPLAY]TKPKG = Ngcgui 1.0
Note: Required only for axis gui embedding, 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
Smaller font sizes may be useful for small screens
Larger font sizes may be helpful for touch screen applications

Item: [DISPLAY]NGCGUI_SUBFILE = subfile_filename
Example: [DISPLAY]NGCGUI_SUBFILE = simp.ngc
Example: [DISPLAY]NGCGUI_SUBFILE = square.gcmc
Example: [DISPLAY]NGCGUI_SUBFILE = ""
Note: Use one or more items to specify ngcgui-compatible subfiles or gcmc programs that require a 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 a subfile.
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 a 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 linuxCNC
 4) an image frame (iframe) is made available to display an image for the subfile (if an image is provided)
 5) the ngcgui result file sent to linuxCNC is terminated with an m2 (and incurs m2 side-effects)

The options nonew, noremove, noauto, noiframe, nom2 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.

Item: [DISPLAY]GCMC_INCLUDE_PATH = dirname1:dirname2:...
 Example: [DISPLAY]GCMC_INCLUDE_PATH = /home/myname/gcmc_includes:/home/myname/ ↵ gcmc_includes2
 Note: Optional, each directory will be included when gcmc is invoked using the option: --include dirname

7.6 File Requirements for NGCGUI Compatibility

7.6.1 Single-File Gcode (.ngc) 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:

```
(info: info_text_to_appear_at_top_of_tab_page)
; comment line beginning with semicolon
( comment line using parentheses)
o<examp> sub
  BODY_OF_SUBROUTINE
o<examp> endsub
; comment line beginning with semicolon
( comment line using parentheses)
```

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 and ngcgui:

(global named parameters have a leading underscore in the name, like #<_someglobalname>)

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.

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

While input global named parameters are discouraged, LinuxCNC 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> (immediately localize the helper global result)
#<_helper:answer> = 0.0      (nullify global named parameter used by subroutine)
...
o<examp> endsub
```

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 example.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. Another library (gcmc_lib) provides examples for subroutine files for the Gcode meta compiler (gcmc)

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

7.6.2 Gcode-meta-compiler (.gcmc) file requirements

Files for the Gcode-meta-compiler (gcmc) are read by ngcgui and it creates entry boxes for variables tagged in the file. When a feature for the file is finalized, ngcgui passes the file as input to the gcmc compiler and, if the compilation is successful, the resulting gcode file is sent to linuxCNC for execution. The resulting file is formatted as single-file subroutine; .gcmc files and .ngc files can be intermixed by ngcgui.

The variables identified for inclusion in ngcgui are tagged with lines that will appear as comments to the gcmc compiler.

Example variable tags formats

```
//ngogui: varname1 =
//ngogui: varname2 = value2
//ngogui: varname3 = value3, label3;
```

Examples:

```
//ngogui: zsafe =
//ngogui: feedrate = 10
//ngogui: xl = 0, x limit
```

For these examples, the entry box for varname1 will have no default, the entry box for varname2 will have a default of value2, and the entry box for varname3 will have a default of value3 and a label label3 (instead of varname3). The default values must be numbers.

To make it easier to modify valid lines in a gcmc file, alternate tag line formats accepted. The alternate formats ignore trailing semicolons (;) and trailing comment markers (//). With this provision, it is often makes it possible to just add the //ngogui: tag to existing lines in a .gcmc file.

Alternate variable tag formats

```
//ngogui: varname2 = value2;
//ngogui: varname3 = value3; //, label3;
```

Examples:

```
//ngogui: feedrate = 10;
//ngogui: xl = 0; //, x limit
```

An info line that will appear at the top of a tab page may be optionally included with a line tagged as:

Info tag

```
//ngogui: info: text_to_appear_at_top_of_tab_page
```

When required, options can be passed to the gcmc compiler with a line tagged:

Option line tag format

```
//ngogui: -option_name [ [=] option_value]
```

Examples:

```
//ngogui: -I
//ngogui: --imperial
//ngogui: --precision 5
//ngogui: --precision=6
```

Options for gcmc are available with the terminal command:

```
gcmc --help
```

A gcmc program by default uses metric mode. The mode can be set to inches with the option setting:

```
//ngogui: --imperial
```

A preamble file, if used, can set a mode (g20 or g21) that conflicts with the mode used by a gcmc file. To ensure that the gcmc program mode is in effect, include the following statement in the .gcmc file:

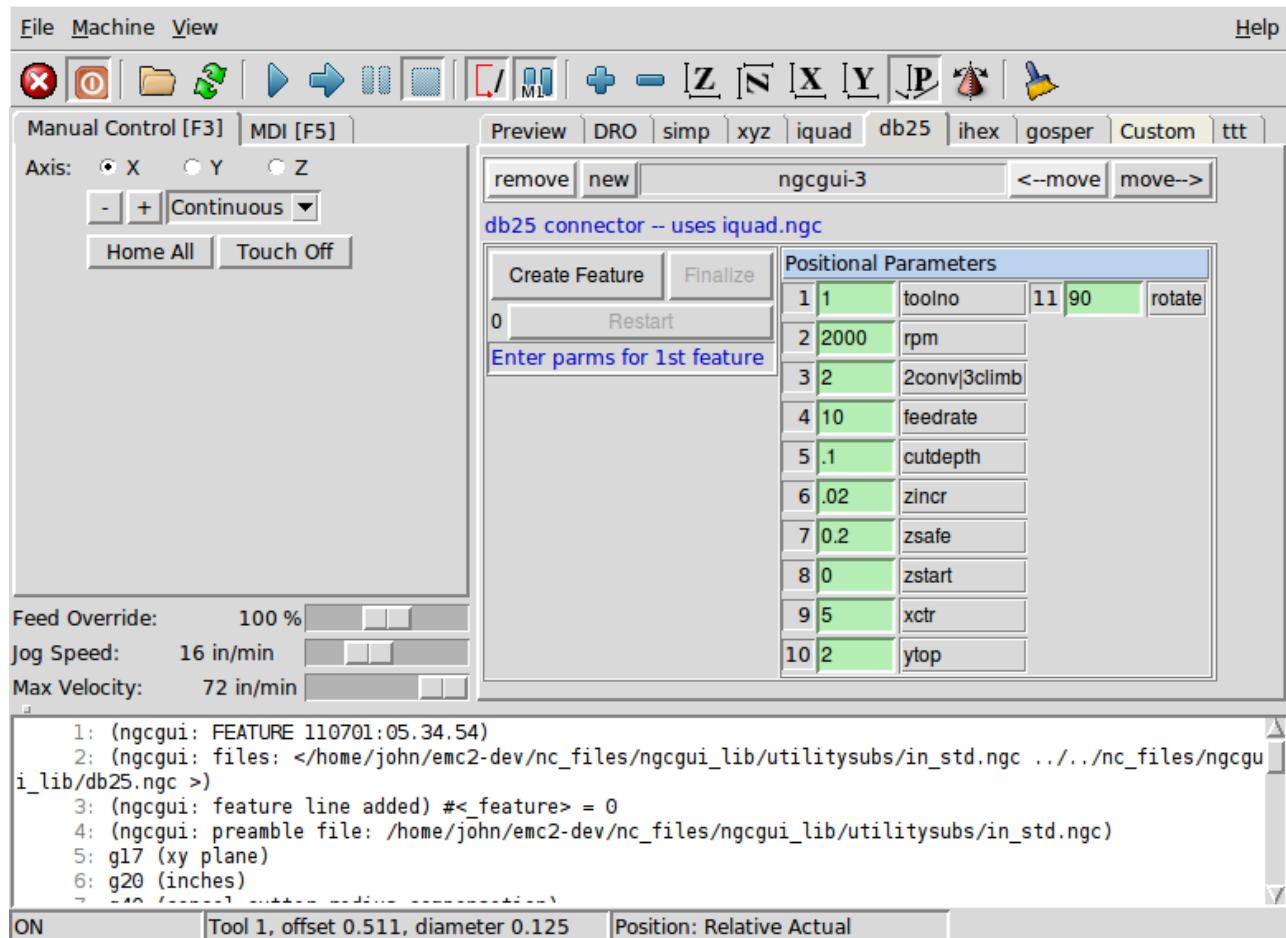
```
include("ensure_mode.gcmc")
```

and provide a proper path for gcmc include_files in the ini file, for example:

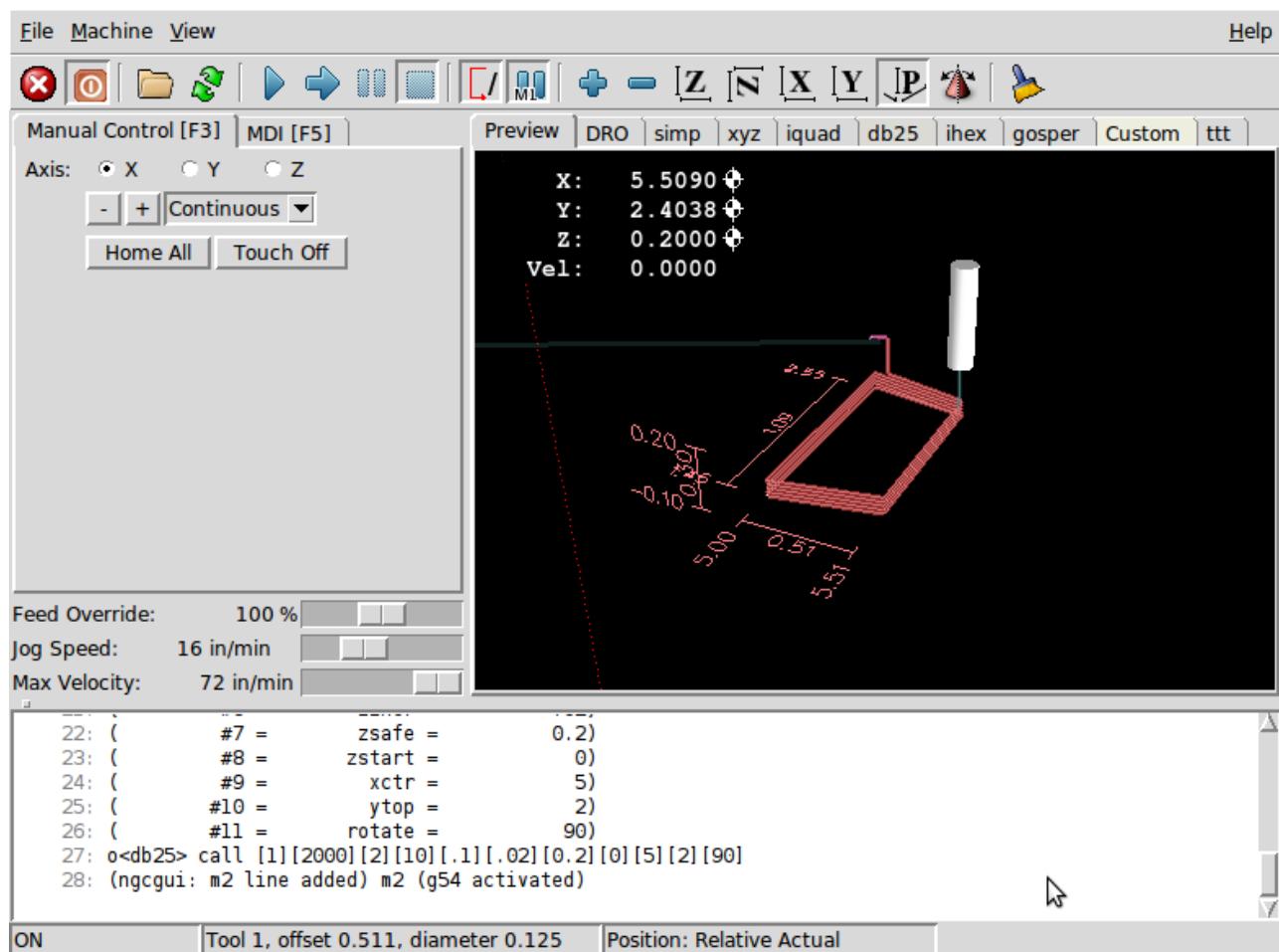
```
[DISPLAY]
GCMC_INCLUDE_PATH = ../../nc_files/gcmc_lib
```

7.7 DB25 Example

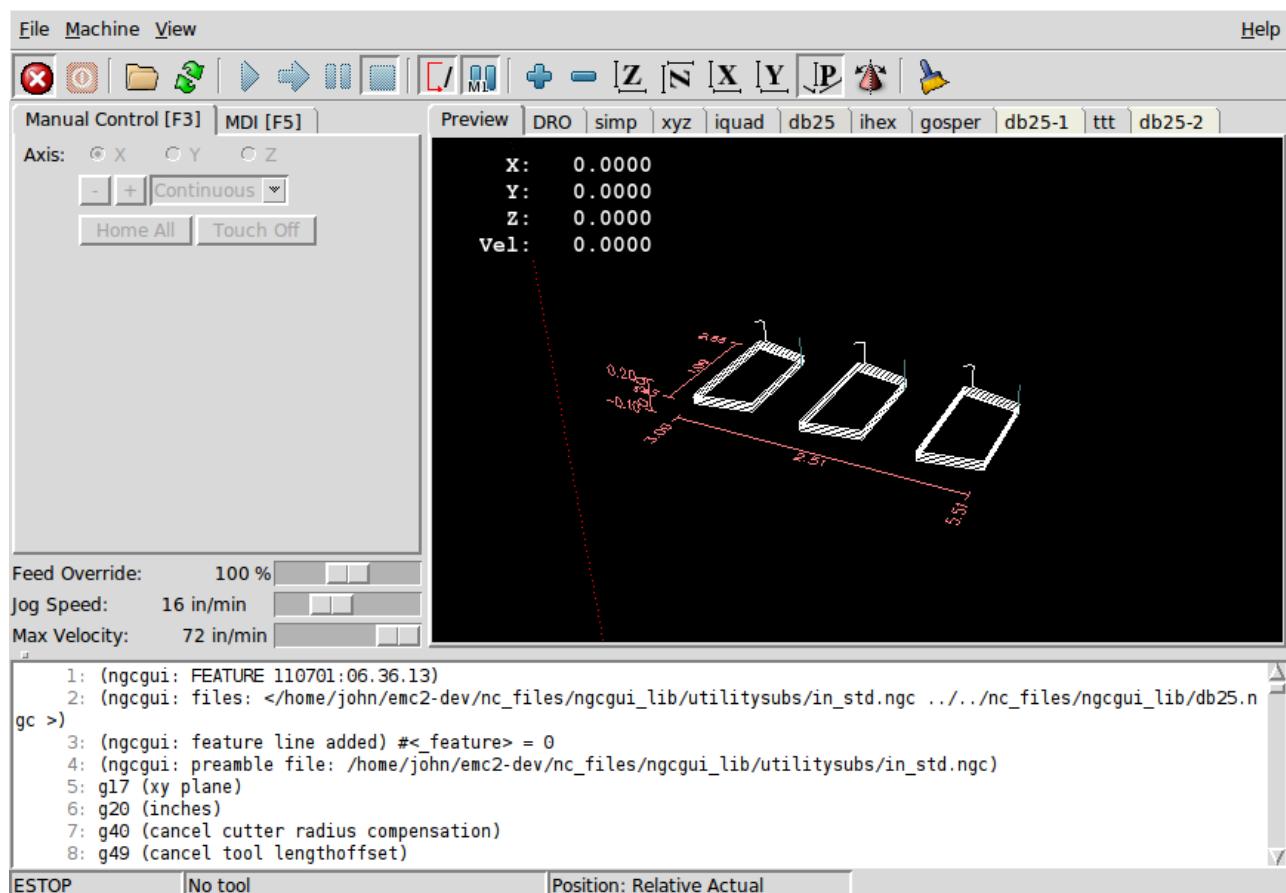
The following shows the DB25 subroutine. In the first photo you see where you fill in the blanks for each variable.



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 8

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.

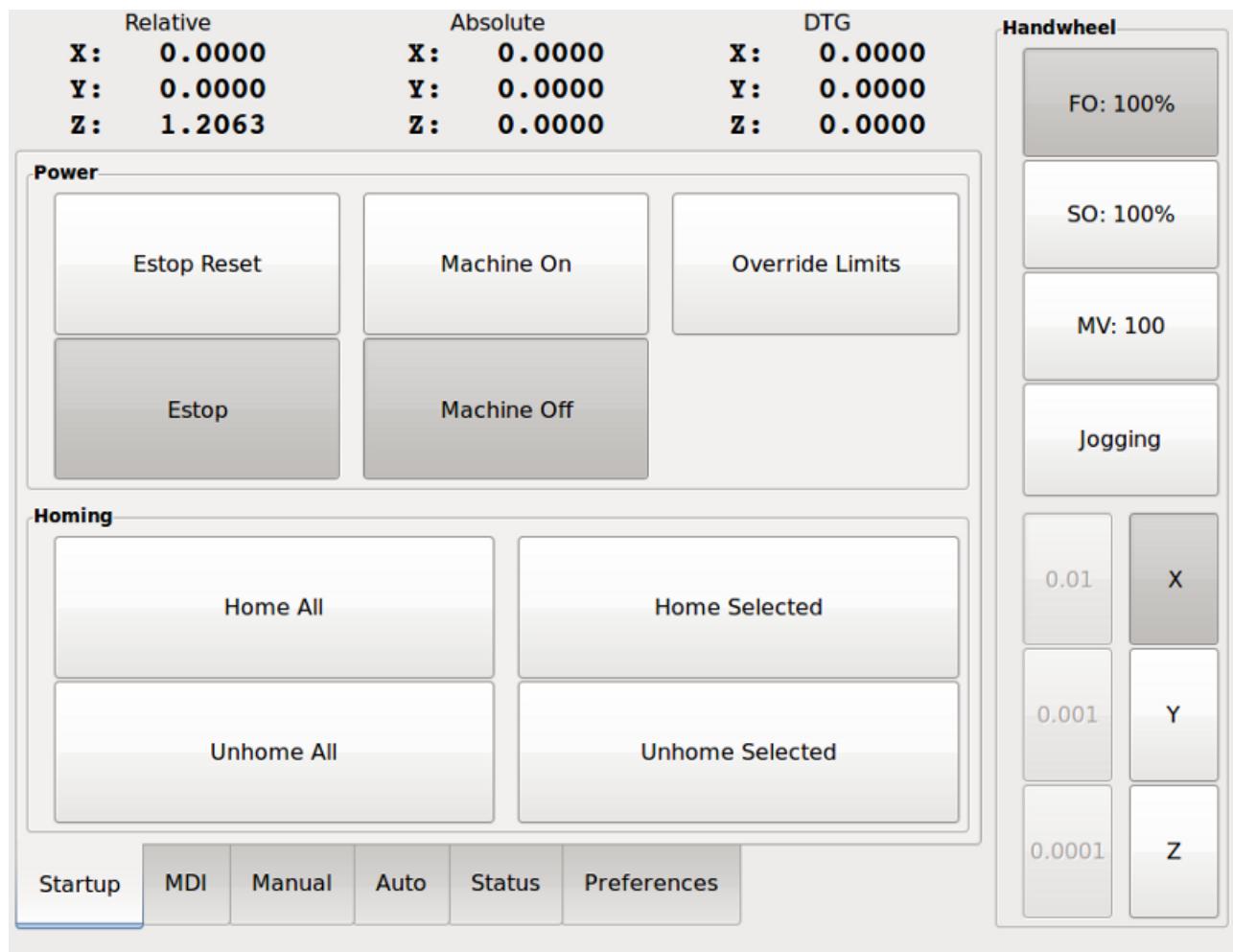


Figure 8.1: Touchy

8.1 Panel Configuration

8.1.1 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*.

8.1.1.1 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*

8.1.1.2 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*.

8.1.1.3 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.

8.1.2 Recommended for any setup

- Estop button hardwired in the estop chain

8.2 Setup

8.2.1 Enabling Touchy

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

8.2.2 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.

8.2.3 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 9

TkLinuxCNC GUI

9.1 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.

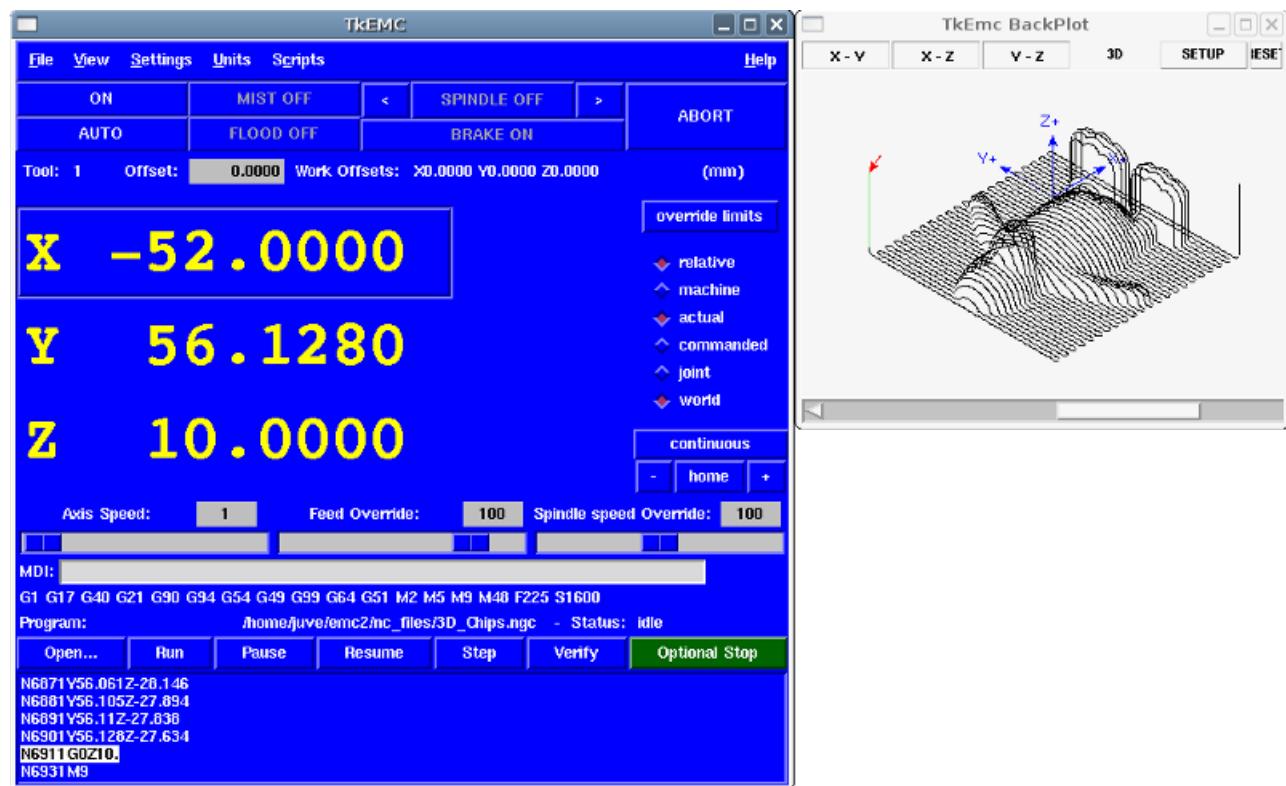


Figure 9.1: TkLinuxCNC Window

9.2 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.

9.2.1 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.

9.3 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

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

9.3.1 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

9.3.2 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).

9.3.3 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).

9.3.3.1 Backplot

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

9.3.4 Automatic control

9.3.4.1 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).

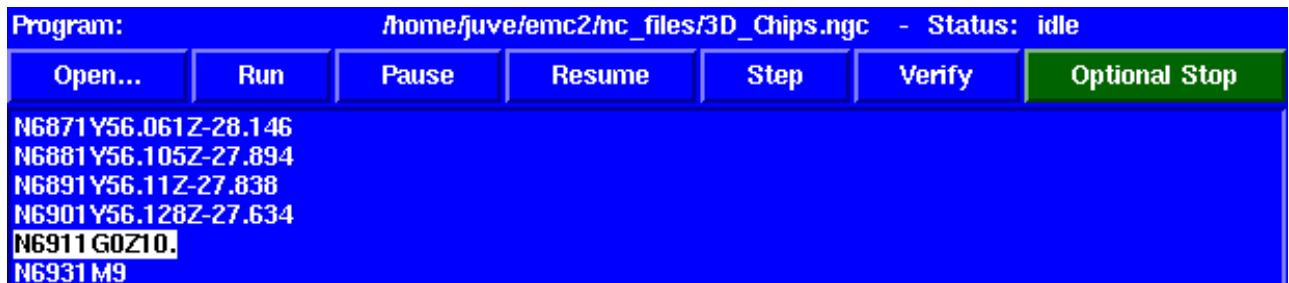


Figure 9.2: TkLinuxCNC Interpreter / program control

9.3.4.2 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.

9.3.5 Manual Control

9.3.5.1 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 \ 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).

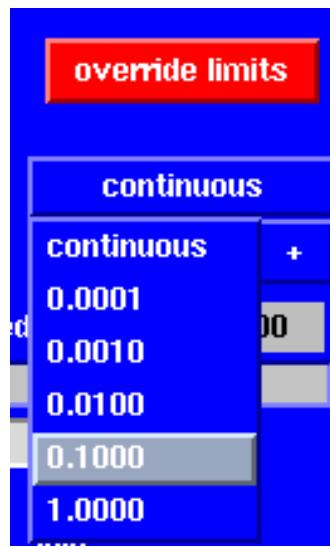


Figure 9.3: TkLinuxCNC Override Limits & Jogging increments example

9.3.5.2 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.

9.3.5.3 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.

9.3.6 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 9.4: The Code Entry tab

9.3.6.1 MDI:

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

9.3.6.2 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.

9.3.7 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.

9.3.8 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.

9.3.9 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.

9.4 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 9.1: 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 10

MINI GUI

10.1 Introduction



Figure 10.1: 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

10.2 Screen layout

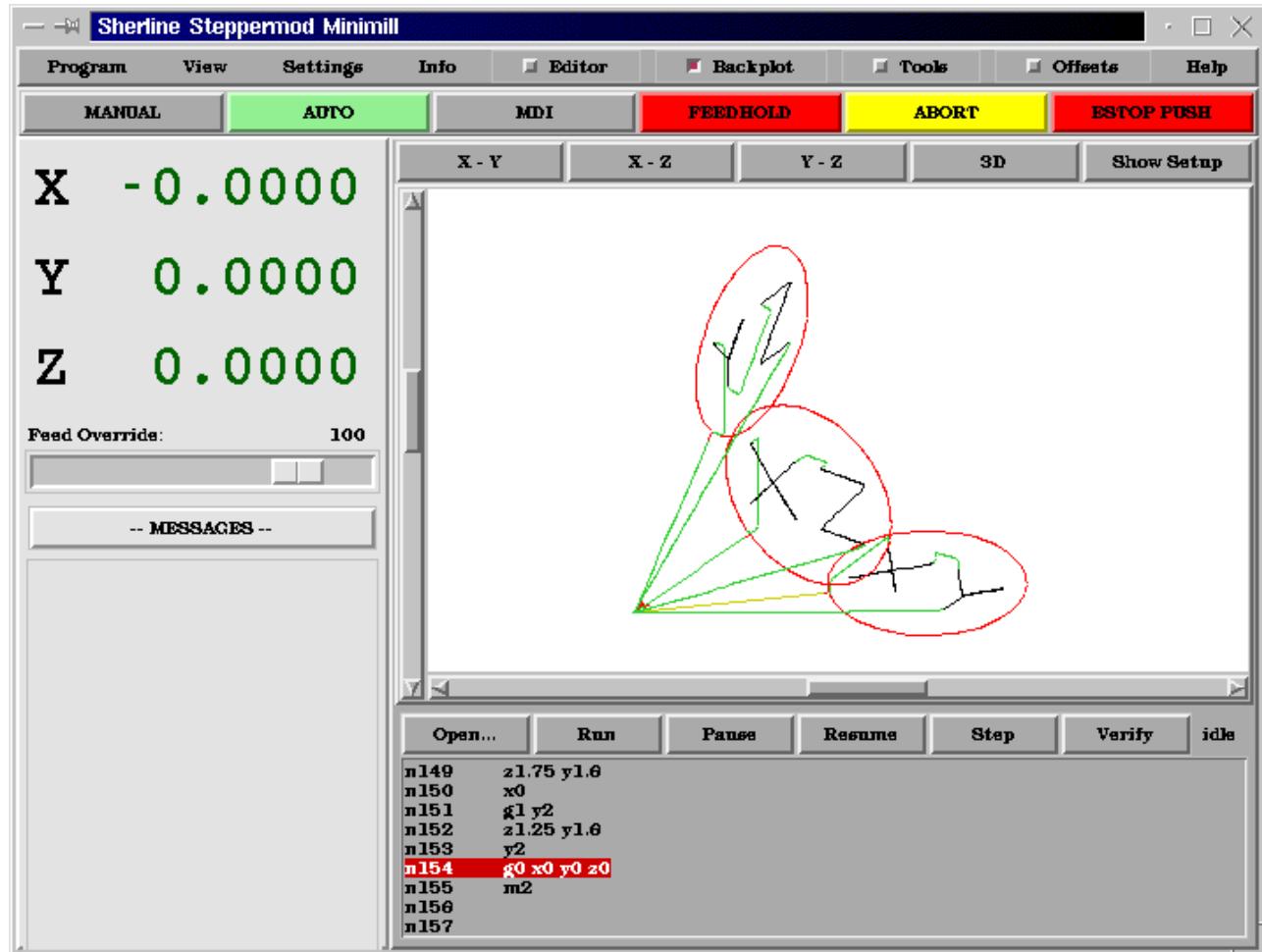


Figure 10.2: Mini Display for a Running LinuxCNC

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

¹ Much 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.

10.3 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 shown 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.

10.4 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.

10.4.1 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 manual mode, you will see spindle and lube control buttons as well as the motion buttons. A keyboard <i> or <l> will switch from continuous jog to incremental jog. Pressing that key again will toggle the increment size through the available sizes.



Figure 10.3: 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.

10.4.2 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.

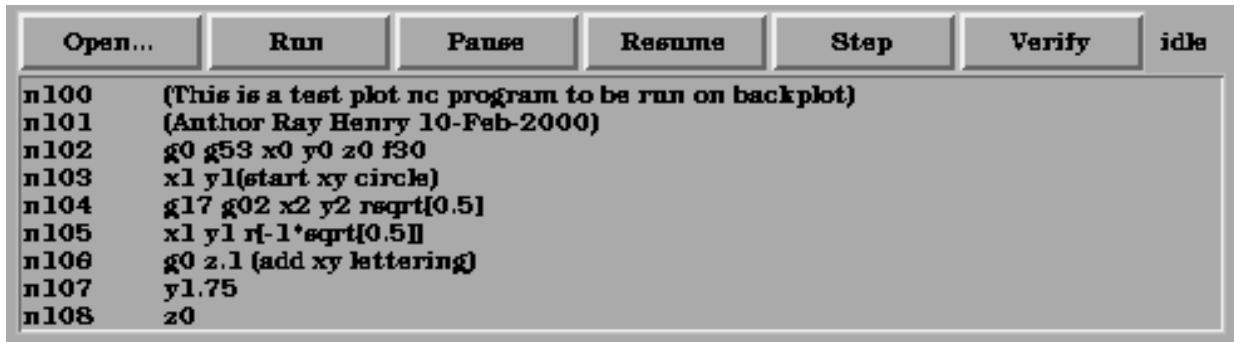


Figure 10.4: Auto Mode

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.

10.4.3 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.

10.4.4 [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.

10.4.5 [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.

10.4.6 [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 taken 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.

10.5 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.

10.5.1 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.

10.5.2 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.

10.5.3 Messages

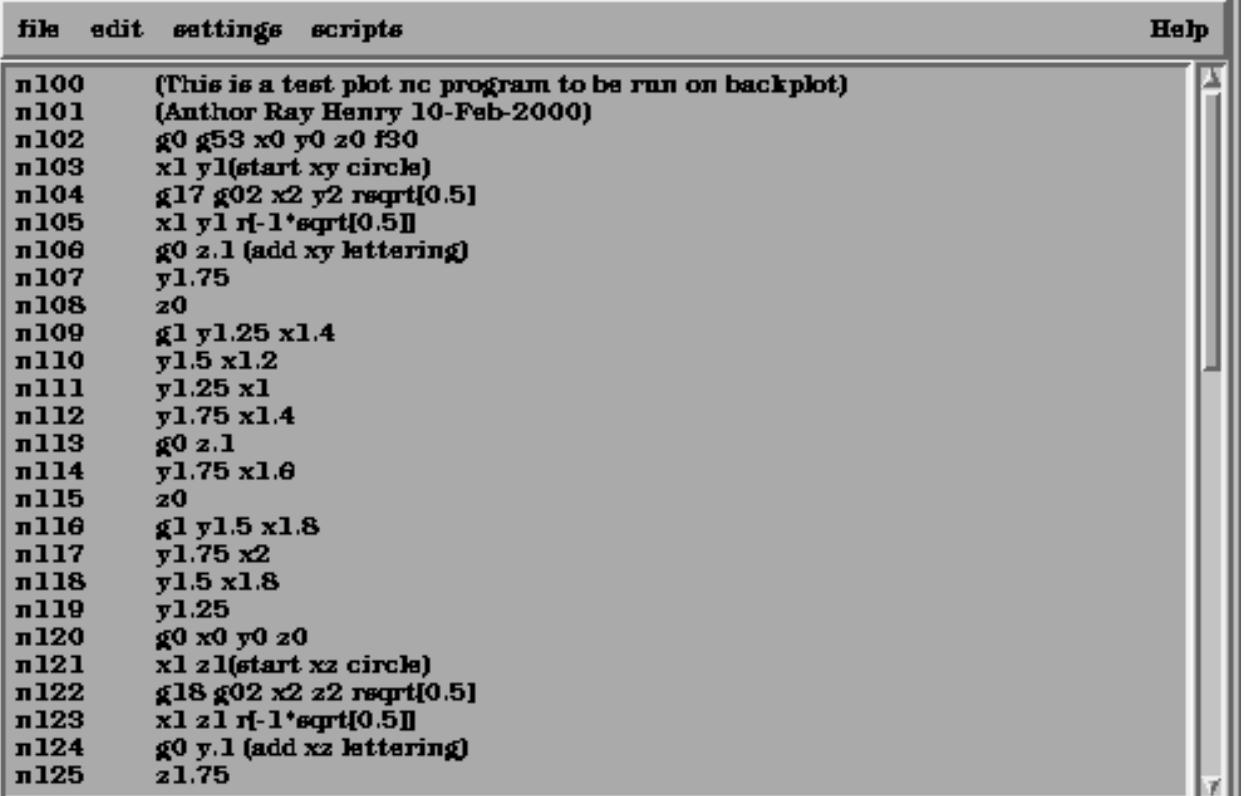
The message display located under the axis positions is a sort of scratch pad for LinuxCNC. If there are problems it will report them there. If you try to home or move an axis when the [ESTOP] button is pressed, you'll get a message that says something about commanding motion when LinuxCNC is not ready. 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.

10.6 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.

10.6.1 Program Editor



The screenshot shows a window titled "Mini Text Editor". The menu bar includes "file", "edit", "settings", "scripts", and "Help". The main area contains the following G-code program:

```
n100 (This is a test plot nc program to be run on backplot)
n101 (Author Ray Henry 10-Feb-2000)
n102 g0 g53 x0 y0 z0 f30
n103 x1 y1(start xy circle)
n104 g17 g02 x2 y2 r:sqrt[0.5]
n105 x1 y1 r:-1*sqrt[0.5]
n106 g0 z.1 (add xy lettering)
n107 y1.75
n108 z0
n109 g1 y1.25 x1.4
n110 y1.5 x1.2
n111 y1.25 x1
n112 y1.75 x1.4
n113 g0 z.1
n114 y1.75 x1.6
n115 z0
n116 g1 y1.5 x1.8
n117 y1.75 x2
n118 y1.5 x1.8
n119 y1.25
n120 g0 x0 y0 z0
n121 x1 z1(start xz circle)
n122 g18 g02 x2 z2 r:sqrt[0.5]
n123 x1 z1 r:-1*sqrt[0.5]
n124 g0 y.1 (add xz lettering)
n125 z1.75
```

Figure 10.5: 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.

10.6.2 Backplot Display

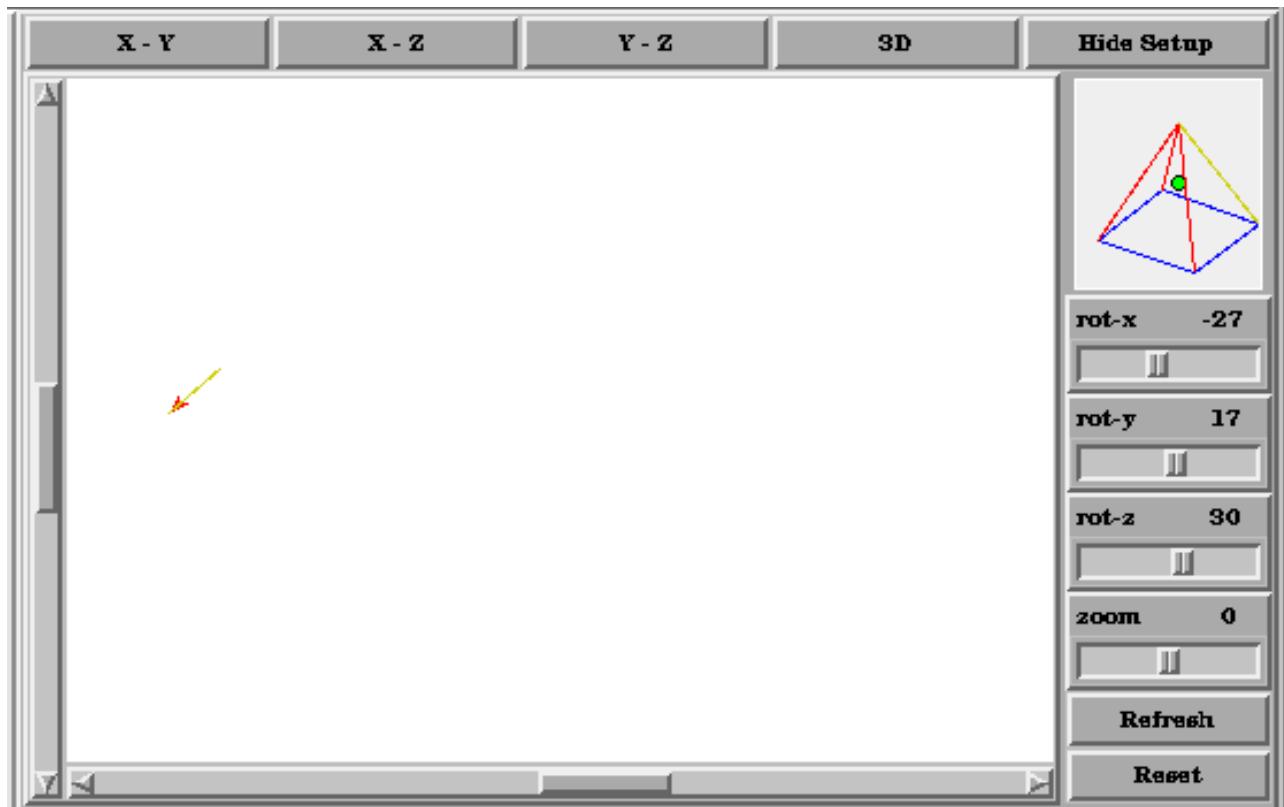


Figure 10.6: Minis Backplotter

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.

10.6.3 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 10.7: 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.

10.6.4 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.

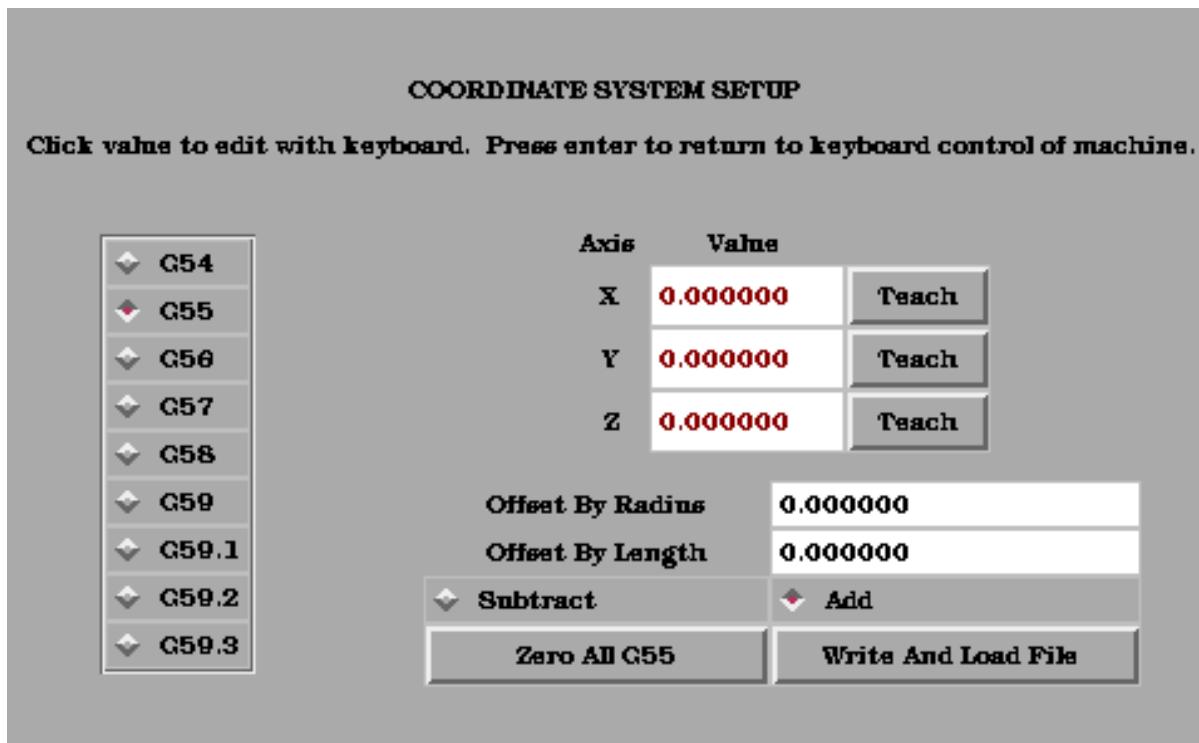


Figure 10.8: Mini Offset Display

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.

10.7 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.

10.7.1 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

10.7.2 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

10.7.3 Auto Mode

- *I-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

10.8 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 11

KEYSTICK GUI

11.1 Introduction

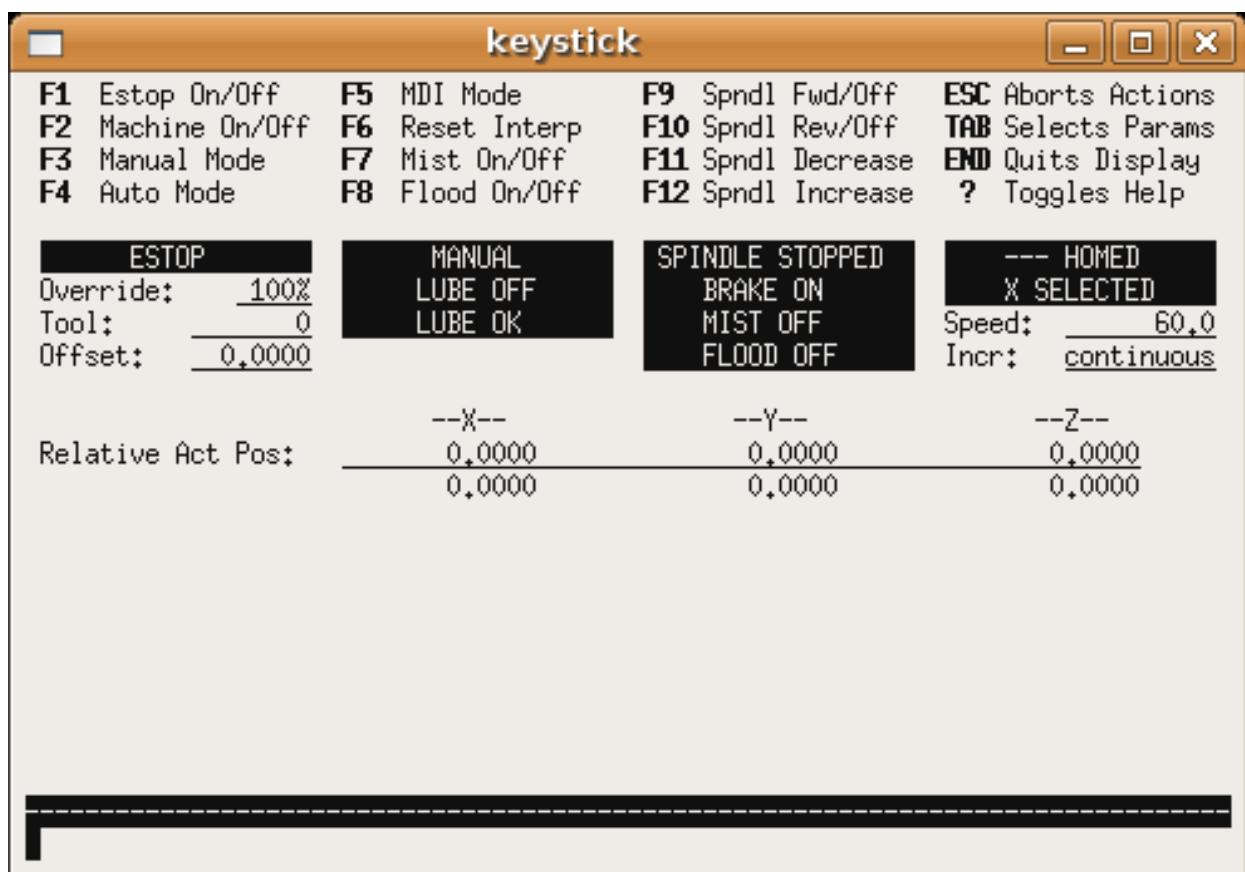


Figure 11.1: The Mini Graphical Interface

Keystick is a minimal text based interface.

11.2 Installing

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

```
DISPLAY = keystick
```

11.3 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 12

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.

12.1 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.

12.1.1 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 XYBZ 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 axes **primary linear primary linear** The X, Y, and Z axes produce linear motion in three mutually orthogonal directions.

Secondary Linear Axes axes **secondary 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 axes **rotational 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.

12.1.2 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.

12.1.3 Coolant

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

¹ If 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*.

12.1.4 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.

12.1.5 Block Delete Switch

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

12.1.6 Optional Program Stop Switch

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

12.2 Control and Data Components

12.2.1 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).

12.2.2 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

12.2.3 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).

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

12.2.4 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.

12.2.5 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.

12.2.6 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](#).

12.2.7 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.

12.2.8 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](#).

12.2.9 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.

12.2.10 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.

12.2.11 Tool Carousel

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

12.2.12 Tool Change

A machining center may be commanded to change tools.

12.2.13 Pallet Shuttle

The two pallets may be exchanged by command.

12.2.14 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](#).

12.3 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.

12.3.1 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.

12.3.2 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.

12.3.3 Optional Program Stop Switch

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

12.4 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.

12.5 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 12.1: 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 13

Coordinate System

13.1 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)

13.2 The Machine Position Command (G53)

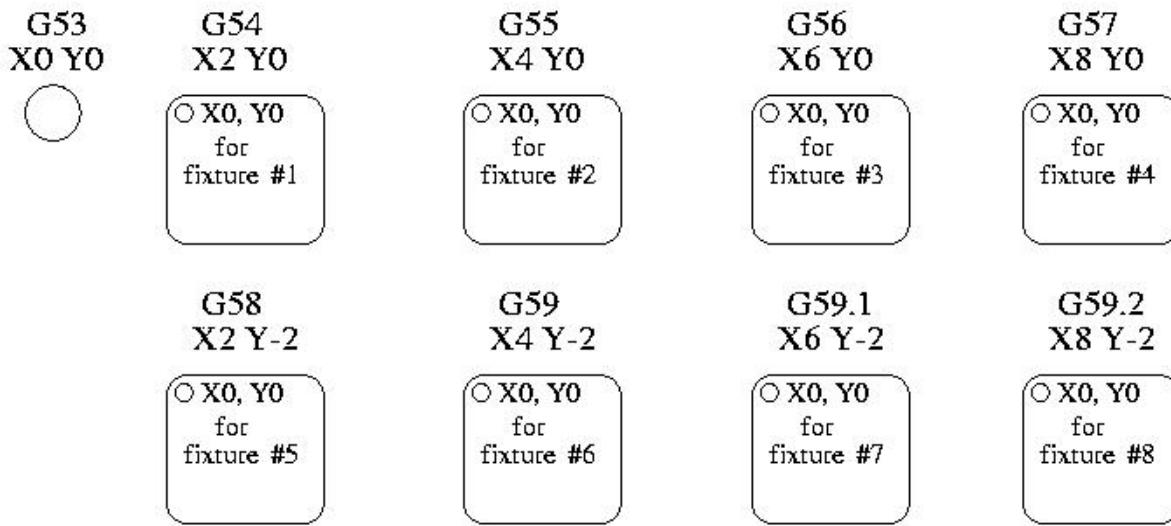
Regardless of any offsets that may be in effect, putting a G53 in a block of code tells the interpreter to go to the real or absolute axis positions commanded in the block. For example

```
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.

13.3 Fixture Offsets (G54-G59.3)



Fixture Offsets Work or fixture offset are used to make a part home that is different from the absolute, machine coordinate system. This allows the part programmer to set up home positions for multiple parts. A typical operation that uses fixture offsets would be to mill multiple copies of parts on multiple part holders or vises.

The values for offsets are stored in the VAR file that is requested by the INI file during the startup of an LinuxCNC. In our example below we'll use G55. The values for each axis for G55 are stored as variable numbers.

Variable	Value
5241	0.000000
5242	0.000000
5243	0.000000
5244	0.000000
5245	0.000000
5246	0.000000

In the VAR file scheme, the first variable number stores the X offset, the second the Y offset and so on for all six axes. There are numbered sets like this for each of the fixture offsets.

Each of the graphical interfaces has a way to set values for these offsets. You can also set these values by editing the VAR file itself and then restarting LinuxCNC so that the LinuxCNC reads the new values however this is not the recommended way. G10, G92, G28.1, etc are better ways to affect variables. For our example let's directly edit the file so that G55 takes on the following values.

Variable	Value
5241	2.000000
5242	1.000000
5243	-2.000000
5244	0.000000
5245	0.000000
5246	0.000000

You should read this as moving the zero positions of G55 to X = 2 units, Y= 1 unit, and Z = -2 units away from the absolute zero position.

Once there are values assigned, a call to G55 in a program block would shift the zero reference by the values stored. The following line would then move each axis to the new zero position. Unlike G53, G54 through G59.3 are modal commands. They will act on all blocks of code after one of them has been set. The program that might be run using fixture offsets would require only a single coordinate reference for each of the locations and all of the work to be done there. The following code is offered as an example of making a square using the G55 offsets that we set above.

```
G55 G0 X0 Y0 Z0
G1 F2 Z-0.2000
X1
Y1
X0
Y0
G0 Z0
G54 X0 Y0 Z0
M2
```

But, you say, why is there a G54 in there near the end. Many programmers leave the G54 coordinate system with all zero values so that there is a modal code for the absolute machine based axis positions. This program assumes that we have done that and use the ending command as a command to machine zero. It would have been possible to use g53 and arrive at the same place but that command would not have been modal and any commands issued after it would have returned to using the G55 offsets because that coordinate system would still be in effect.

```
G54      use preset work coordinate system 1
G55      use preset work coordinate system 2
G56      use preset work coordinate system 3
G57      use preset work coordinate system 4
G58      use preset work coordinate system 5
G59      use preset work coordinate system 6
G59.1    use preset work coordinate system 7
G59.2    use preset work coordinate system 8
G59.3    use preset work coordinate system 9
```

13.3.1 Default coordinate system

One other variable in the VAR file becomes important when we think about offset systems. This variable is named 5220. In the default files its value is set to 1.00000. This means that when the LinuxCNC starts up it should use the first coordinate system as its default. If you set this to 9.00000 it would use the ninth offset system as its default for start up and reset. Any value other than an integer (decimal really) between 1 and 9, or a missing 5220 variable will cause the LinuxCNC to revert to the default value of 1.00000 on start up.

13.3.2 Setting coordinate (fixture) offsets from G code

The G10 L2x command can be used to set coordinate (fixture) offsets: (these are just quick summaries, see the G code section for full details)

- *G10 L2 P(fixture 1-9)* - Set offset(s) to a value. Current position irrelevant. (see [G10 L2](#) for details)
- *G10 L20 P(fixture 1-9)* - Set offset(s) so current position becomes a value. (see [G10 L20](#) for details)

13.4 G92 Offsets

13.4.1 The G92 commands

This set of commands include;

- *G92* - This command, when used with axis names, sets values to offset variables.
- *G92.1* - This command sets zero values to the G92 variables.
- *G92.2* - This command suspends but does not zero out the G92 variables.
- *G92.3* - This command applies offset values that have been suspended.

When the commands are used as described above, they will work pretty much as you would expect.

To make the current point, what ever it is, have the coordinates X0, Y0, and Z0 you would use G92 X0 Y0 Z0. G92 **does not** work from absolute machine coordinates. It works from **current location**.

G92 also works from current location as modified by any other offsets that are in effect when the G92 command is invoked. While testing for differences between work offsets and actual offsets it was found that a G54 offset could cancel out a G92 and thus give the appearance that no offsets were in effect. However, the G92 was still in effect for all coordinates and did produce expected work offsets for the other coordinate systems.

It is a good practice to clear the G92 offsets at the end of their use with G92.1 or G92.2. When starting up LinuxCNC if any offsets are in the G92 variables they will be applied when an axis is homed.

13.4.2 Setting G92 values

There are at least two ways to set G92 values.

- right mouse click on position displays of tkLinuxCNC will popup a window into which you can type a value.
- the G92 command

Both of these work from the current location of the axis to which the offset is to be applied.

Issuing *G92 X Y Z A B C U V W* does in fact set values to the G92 variables such that each axis takes on the value associated with its name. These values are assigned to the current position of the machine axis. These results satisfy paragraphs one and two of the NIST document.

G92 commands work from current axis location and add and subtract correctly to give the current axis position the value assigned by the G92 command. The effects work even though previous offsets are in.

So if the X axis is currently showing 2.0000 as its position a *G92 X0* will set an offset of -2.0000 so that the current location of X becomes zero. A *G92 X2* will set an offset of 0.0000 and the displayed position will not change. A *G92 X5.0000* will set an offset of 3.0000 so that the current displayed position becomes 5.0000.

13.4.3 G92 Cautions

Sometimes the values of a G92 offset will remain in the VAR file. This can happen when a file is aborted during processing that has G92 offsets in effect. When this happens reset or a startup will cause them to become active again.

The variables are named:

Variable	Value
5211	0.000000
5212	0.000000
5213	0.000000
5214	0.000000
5215	0.000000
5216	0.000000

where 5211 is the X axis offset and so on. If you are seeing unexpected positions as the result of a commanded move, as a result of storing an offset in a previous program and not clearing them at the end then issue a G92.1 in the MDI window to clear the

stored offsets.

If G92 values exist in the VAR file when LinuxCNC starts up, the G92 values in the var file will be applied to the values of the current location of each axis. If this is home position and home position is set as machine zero everything will be correct. Once home has been established using real machine switches, or by moving each axis to a known home position and issuing an axis home command, any G92 offsets will be applied. If you have a G92 X1 in effect when you home the X axis the DRO will read *X: 1.000* instead of the expected *X: 0.000* because the G92 was applied to the machine origin. If you issue a G92.1 and the DRO now reads all zeros then you had a G92 offset in effect when you last ran LinuxCNC.

Unless your intention is to use the same G92 offsets in the next program, the best practice is to issue a G92.1 at the end of any G Code files where you use G92 offsets.

13.5 Sample Program Using Offsets

This sample engraving project mills a set of four .1 radius circles in roughly a star shape around a center circle. We can setup the individual circle pattern like this.

```
G10 L2 P1 X0 Y0 Z0 (ensure that G54 is set to machine zero)
G0 X-0.1 Y0 Z0
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
M2
```

We can issue a set of commands to create offsets for the four other circles like this.

```
G10 L2 P2 X0.5 (offsets G55 X value by 0.5 inch)
G10 L2 P3 X-0.5 (offsets G56 X value by -0.5 inch)
G10 L2 P4 Y0.5 (offsets G57 Y value by 0.5 inch)
G10 L2 P5 Y-0.5 (offsets G58 Y value by -0.5 inch)
```

We put these together in the following program:

```
(a program for milling five small circles in a diamond shape)

G10 L2 P1 X0 Y0 Z0 (ensure that G54 is machine zero)
G10 L2 P2 X0.5 (offsets G55 X value by 0.5 inch)
G10 L2 P3 X-0.5 (offsets G56 X value by -0.5 inch)
G10 L2 P4 Y0.5 (offsets G57 Y value by 0.5 inch)
G10 L2 P5 Y-0.5 (offsets G58 Y value by -0.5 inch)

G54 G0 X-0.1 Y0 Z0 (center circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G55 G0 X-0.1 Y0 Z0 (first offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G56 G0 X-0.1 Y0 Z0 (second offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0

G57 G0 X-0.1 Y0 Z0 (third offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G0 Z0
```

```
G58 G0 X-0.1 Y0 Z0 (fourth offset circle)
G1 F1 Z-0.25
G3 X-0.1 Y0 I0.1 J0
G54 G0 X0 Y0 Z0

M2
```

Now comes the time when we might apply a set of G92 offsets to this program. You'll see that it is running in each case at Z0. If the mill were at the zero position, a G92 Z1.0000 issued at the head of the program would shift everything down an inch. You might also shift the whole pattern around in the XY plane by adding some X and Y offsets with G92. If you do this you should add a G92.1 command just before the m2 that ends the program. If you do not, other programs that you might run after this one will also use that G92 offset. Furthermore it would save the G92 values when you shut down the LinuxCNC and they will be recalled when you start up again.

Chapter 14

Tool Compensation

14.1 Tool Length Offsets

14.1.1 Touch Off

Using the Touch Off Screen in the AXIS interface you can update the tool table automatically.

Typical steps for updating the tool table:

- After homing load a tool with $Tn M6$ where n is the tool number.
- Move tool to an established point using a gauge or take a test cut and measure.
- Click the "Touch Off" button in the Manual Control tab (or hit the End button on your keyboard).
- Select *Tool Table* in the Coordinate System drop down box.
- Enter the gauge or measured dimension and select OK.

The Tool Table will be changed with the correct Z length to make the DRO display the correct Z position and a G43 command will be issued so the new tool Z length will be in effect. Tool table touch off is only available when a tool is loaded with $Tn M6$.

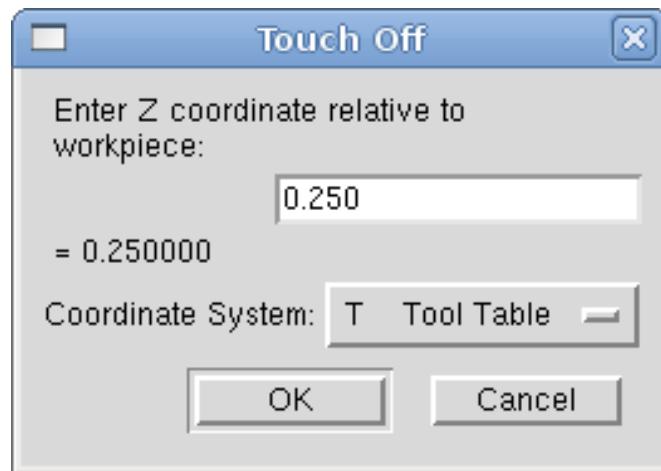


Figure 14.1: Touch Off Tool Table

14.1.2 Using G10 L1/L10/L11

The G10 L1/L10/L11 commands can be used to set tool table offsets: (these are just quick summaries, see the G code section for full details)

- *G10 L1 Pn* - Set offset(s) to a value. Current position irrelevant. (see [G10 L1](#) for details)
- *G10 L10 Pn* - Set offset(s) so current position w/ fixture 1-8 becomes a value. (see [G10 L10](#) for details)
- *G10 L11 Pn* - Set offset(s) so current position w/ fixture 9 becomes a value. (see [G10 L11](#) for details)

14.2 Tool Table

The *Tool Table* is a text file that contains information about each tool. The file is located in the same directory as your configuration and is called *tool.tbl*. The tools might be in a tool changer or just changed manually. The file can be edited with a text editor or be updated using G10 L1. See the [Lathe Tool Table](#) Section for an example of the lathe tool table format. The maximum number of entries in the tool table is 56. The maximum tool and pocket number is 99999.

The [Tool Editor](#) or a text editor can be used to edit the tool table. If you use a text editor make sure you reload the tool table in the GUI.

14.2.1 Tool Table Format

Table 14.1: Tool Table Format

T#	P#	X	Y	Z	A	B	C	U	V	W	Dia	FA	BA	Ori	Rem
; (no data after opening semicolon)															
T1	P17	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem
T2	P5	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem
T3	P12	X0	Y0	Z0	A0	B0	C0	U0	V0	W0	D0	I0	J0	Q0	;rem

In general, the new tool table line format is:

- ; - opening semicolon, no data
- T - tool number, 0-99999 (tool numbers must be unique)
- P - pocket number, 1-99999 (pocket numbers must be unique)
- X..W - tool offset on specified axis - floating-point
- D - tool diameter - floating-point, absolute value
- I - front angle (lathe only) - floating-point
- J - back angle (lathe only) - floating-point
- Q - tool orientation (lathe only) - integer, 0-9
- ; - beginning of comment or remark - text

The file consists of one opening semicolon on the first line, followed by up to a maximum of 56 tool entries.¹

¹ Although tool numbers up to 99999 are allowed, the number of entries in the tool table, at the moment, is still limited to a maximum of 56 tools for technical reasons. The LinuxCNC developers plan to remove that limitation eventually. If you have a very large tool changer, please be patient.

Earlier versions of LinuxCNC had two different tool table formats for mills and lathes, but since the 2.4.x release, one tool table format is used for all machines. Just ignore the parts of the tool table that don't pertain to your machine, or which you don't need to use.

Each line of the tool table file after the opening semicolon contains the data for one tool. One line may contain as many as 16 entries, but will likely contain much fewer.

The units used for the length, diameter, etc., are in machine units.

You will probably want to keep the tool entries in ascending order, especially if you are going to be using a randomizing tool changer. Although the tool table does allow for tool numbers in any order.

Each line may have up to 16 entries. The first two entries are required. The last entry (a remark or comment, preceded by a semicolon) is optional. It makes reading easier if the entries are arranged in columns, as shown in the table, but the only format requirement is that there be at least one space or tab after each of the entries on a line and a newline character at the end of each entry.

The meanings of the entries and the type of data to be put in each are as follows.

Tool Number (required) The *T* column contains the number (unsigned integer) which represents a code number for the tool. The user may use any code for any tool, as long as the codes are unsigned integers.

Pocket Number (required) The *P* column contains the number (unsigned integer) which represents the pocket number (slot number) of the tool changer slot where the tool can be found. The entries in this column must all be different.

The pocket numbers will typically start at 1 and go up to the highest available pocket on your tool changer. But not all tool changers follow this pattern. Your pocket numbers will be determined by the numbers that your tool changer uses to refer to the pockets. So all this is to say that the pocket numbers you use will be determined by the numbering scheme used in your tool changer, and the pocket numbers you use must make sense on your machine.

Data Offset Numbers (optional) The *Data Offset* columns (XYZABCUVW) contain real numbers which represent tool offsets in each axis. This number will be used if tool length offsets are being used and this tool is selected. These numbers can be positive, zero, or negative, and are in fact completely optional. Although you will probably want to make at least one entry here, otherwise there would be little point in making an entry in the tool table to begin with.

In a typical mill, you probably want an entry for Z (tool length offset). In a typical lathe, you probably want an entry for X (X tool offset) and Z (Z tool offset). In a typical mill using cutter diameter compensation (cutter comp), you probably also want to add an entry for D (cutter diameter). In a typical lathe using tool nose diameter compensation (tool comp), you probably also want to add an entry for D (tool nose diameter).

A lathe also requires some additional information to describe the shape and orientation of the tool. So you probably want to have entries for I (tool front angle) and J (tool back angle). You probably also want an entry for Q (tool orientation).

A complete description of the lathe entries can be found in the lathe section of the user manual [here](#).

The *Diameter* column contains a real number. This number is used only if cutter compensation is turned on using this tool. If the programmed path during compensation is the edge of the material being cut, this should be a positive real number representing the measured diameter of the tool. If the programmed path during compensation is the path of a tool whose diameter is nominal, this should be a small number (positive or negative, but near zero) representing only the difference between the measured diameter of the tool and the nominal diameter. If cutter compensation is not used with a tool, it does not matter what number is in this column.

The *Comment* column may optionally be used to describe the tool. Any type of description is OK. This column is for the benefit of human readers only. The comment must be preceded by a semicolon.

14.2.2 Tool Changers

LinuxCNC supports three types of tool changers: *manual*, *random location* and *fixed location*. Information about configuring an LinuxCNC tool changer is in the Integrator Manual.

Manual Tool Changer Manual tool changer (you change the tool by hand) is treated like a fixed location tool changer and the P number is ignored. Using the manual tool changer only makes sense if you have tool holders that remain with the tool (Cat, NMTB, Kwik Switch etc.) when changed thus preserving the location of the tool to the spindle. Machines with R-8 or router collet type tool holders do not preserve the location of the tool and the manual tool changer should not be used.

Fixed Location Tool Changers Fixed location tool changers always return the tools to a fixed position in the tool changer. This would also include designs like lathe turrets. When LinuxCNC is configured for a fixed location tool changer the *P* number is ignored (but read, preserved and rewritten) by LinuxCNC, so you can use *P* for any bookkeeping number you want.

Random Location Tool Changers Random location tool changers swap the tool in the spindle with the one in the changer. With this type of tool changer the tool will always be in a different pocket after a tool change. When a tool is changed LinuxCNC rewrites the pocket number to keep track of where the tools are. *T* can be any number but *P* must be a number that makes sense for the machine.

14.3 Cutter Compensation

Cutter Compensation allows the programmer to program the tool path without knowing the exact tool diameter. The only caveat is the programmer must program the lead in move to be at least as long as the largest tool radius that might be used.

There are two possible paths the cutter can take while cutter compensation is on to the left or right side of a line when facing the direction of cutter motion from behind the cutter. To visualize this imagine you were standing on the part walking behind the tool as it progresses across the part. G41 is your left side of the line and G42 is the right side of the line.

The end point of each move depends on the next move. If the next move creates an outside corner the move will be to the end point of the compensated cut line. If the next move creates an inside corner the move will stop short so to not gouge the part. The following figure shows how the compensated move will stop at different points depending on the next move.

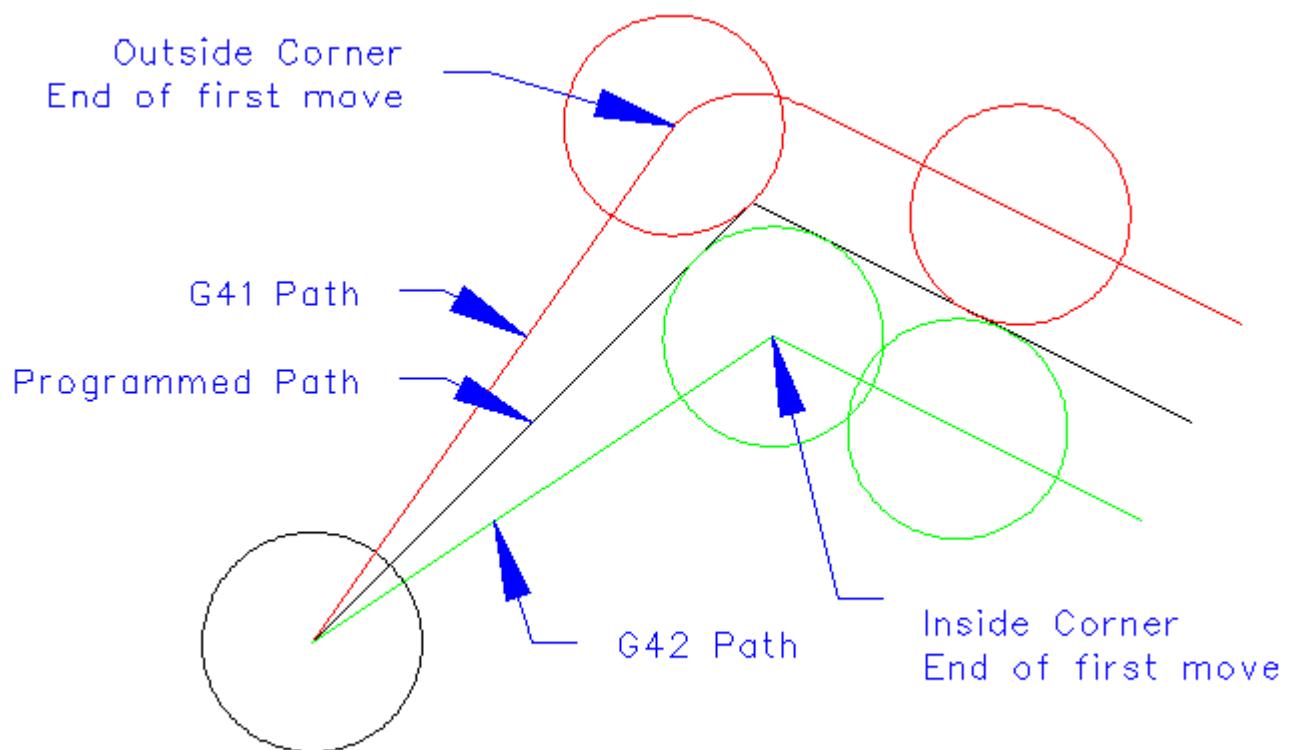


Figure 14.2: Compensation End Point

14.3.1 Overview

Tool Table Cutter compensation uses the data from the tool table to determine the offset needed. The data can be set at run time with G10 L1.

Programming Entry Moves Any move that is long enough to perform the compensation will work as the entry move. The minimum length is the cutter radius. This can be a rapid move above the work piece. If several rapid moves are issued after a G41/42 only the last one will move the tool to the compensated position.

In the following figure you can see that the entry move is compensated to the right of the line. This puts the center of the tool to the right of X0 in this case. If you were to program a profile and the end is at X0 the resulting profile would leave a bump due to the offset of the entry move.

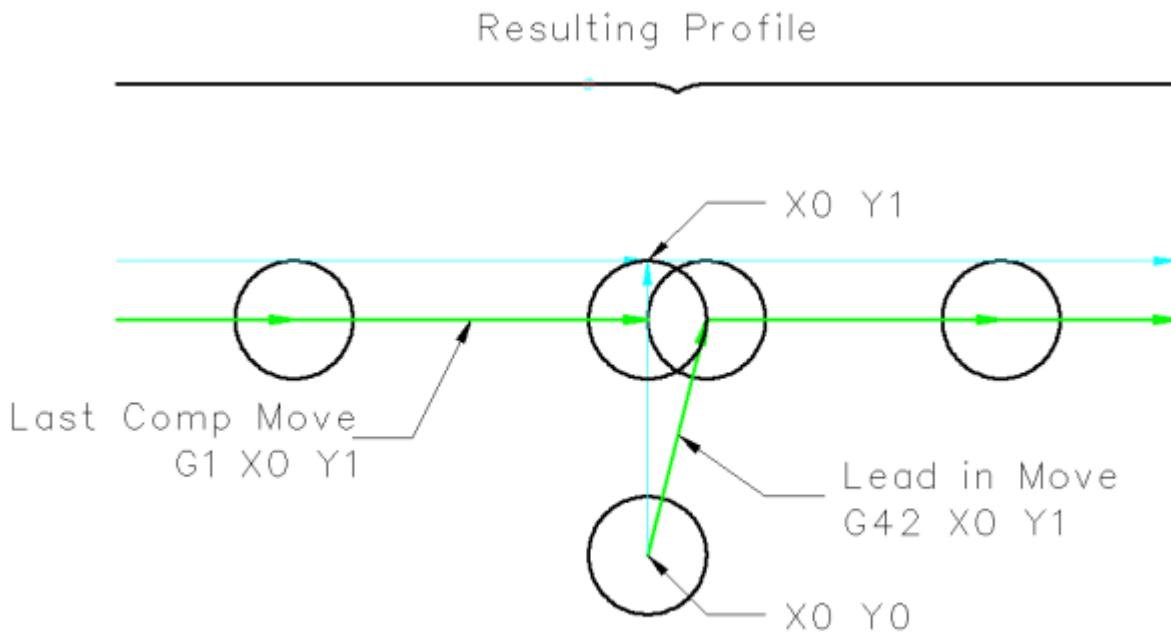


Figure 14.3: Entry Move

Z Motion Z axis motion may take place while the contour is being followed in the XY plane. Portions of the contour may be skipped by retracting the Z axis above the part and by extending the Z-axis at the next start point.

Rapid Moves Rapid moves may be programmed while compensation is turned on.

GOOD PRACTICES

- Start a program with G40 to make sure compensation is off.

14.3.2 Examples

```
G-Code  
F25 ( Set Feed Rate )  
G40 ( Cancel Comp )  
G10 L1 P1 R0.25 Z1 ( Set Tool Table )  
T1 M6 ( Load Tool )  
G42 ( Start Comp Right )  
G1 X1 Y1 (Lead In Move)  
X5 ( Cut Path )  
Y5  
X1  
Y1  
G40 ( Cancel Comp )  
G0 X0 Y0 ( Exit Move )  
M2 ( End Program )
```

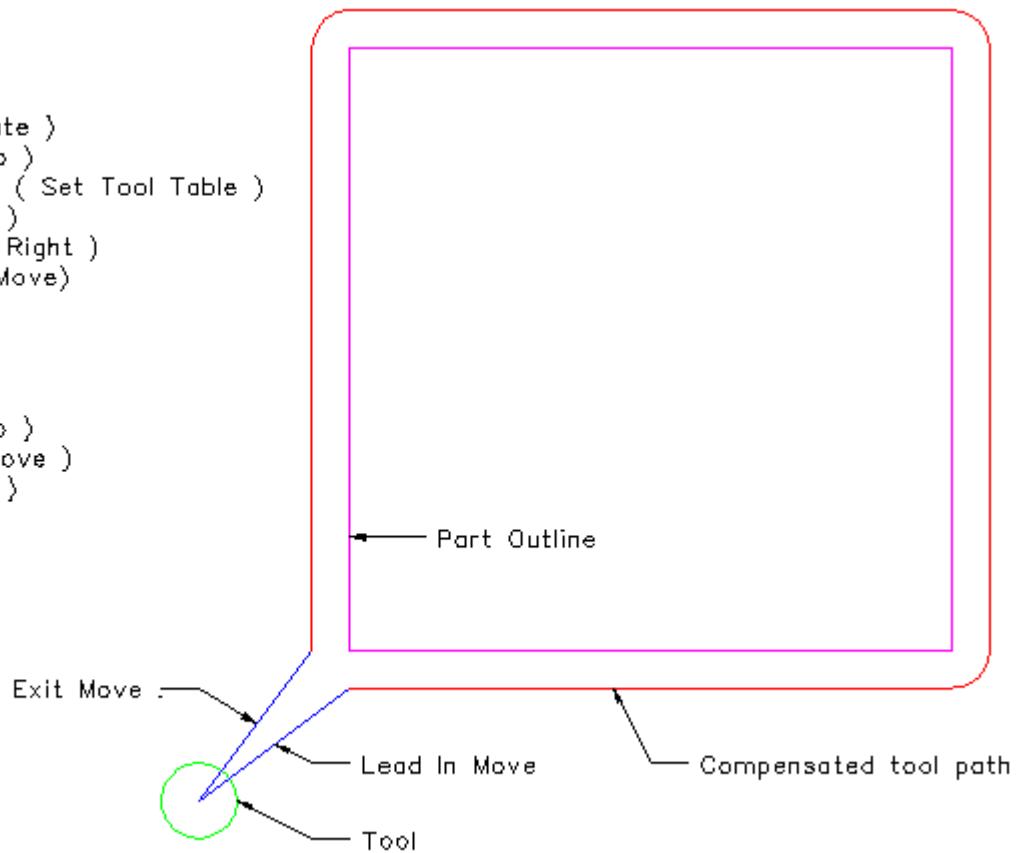


Figure 14.4: Outside Profile

```
G20 ( Inch Mode )
F30 ( Set Feed Rate )
G10 L1 P1 R.25 Z1 ( Set Tool Table )
T1 M6 ( Load the Tool )
G0 Z0 ( Move to safe Z height )
G41 ( Start Cutter Comp Left )
X4 Y3 ( Rapid to start point )
G1 X5 Z-1 ( Move to cut height )
G3 X6 Y4 J1 ( Arc into cut path )
G1 Y6 ( Cut Profile )
X2
Y2
X6
Y4
G3 X5 Y5 I-1 ( Arc out of cut path )
G0 Z0 ( Move cutter to safe Z height )
G40 ( Stop Cutter Comp )
G0 X1 Y1 ( Move to safe position )
T0 M6 ( Remove Tool )
M2 ( End Program )
```

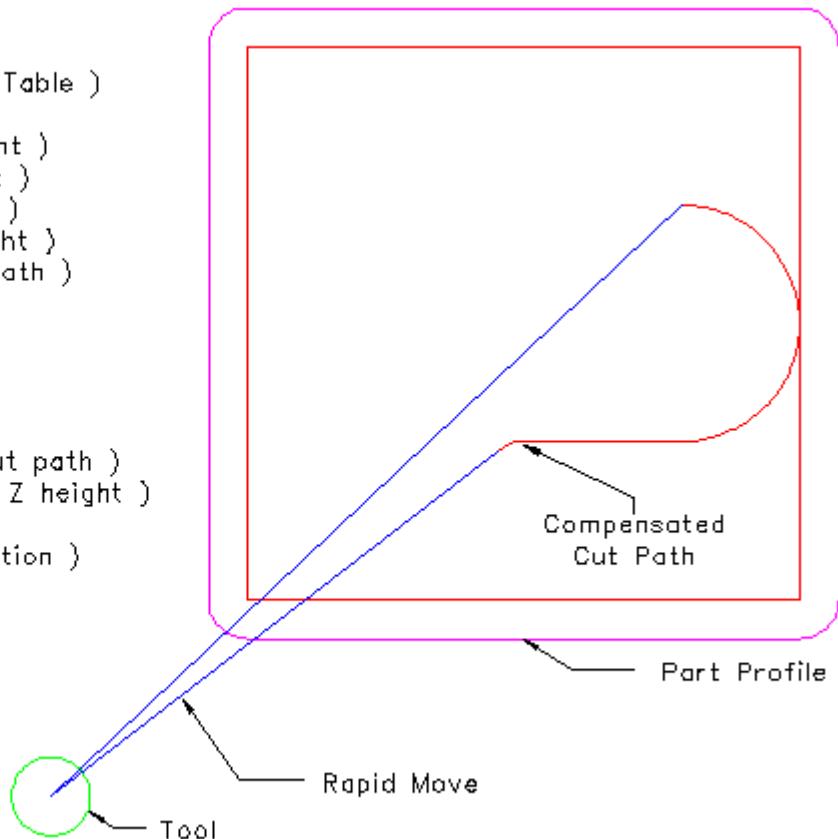


Figure 14.5: Inside Profile

Chapter 15

G Code Overview

15.1 Overview

The LinuxCNC G Code language is based on the RS274/NGC language. The G Code language is based on lines of code. Each line (also called a *block*) may include commands to do several different things. Lines of code may be collected in a file to make a program.

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

The LinuxCNC language has no indicator for the start of a program. The Interpreter, however, deals with files. A single program may be in a single file, or a program may be spread across several files. A file may be demarcated with percents in the following way. The first non-blank line of a file may contain nothing but a percent sign, %, possibly surrounded by white space, and later in the file (normally at the end of the file) there may be a similar line. Demarcating a file with percents is optional if the file has an *M2* or *M30* in it, but is required if not. An error will be signaled if a file has a percent line at the beginning but not at the end. The useful contents of a file demarcated by percents stop after the second percent line. Anything after that is ignored.

The LinuxCNC G Code language has two commands (*M2* or *M30*), either of which ends a program. A program may end before the end of a file. Lines of a file that occur after the end of a program are not to be executed. The interpreter does not even read them.

15.2 Format of a line

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

1. an optional block delete character, which is a slash /.
2. an optional line number.
3. any number of words, parameter settings, and comments.
4. an end of line marker (carriage return or line feed or both).

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

Spaces and tabs are allowed anywhere on a line of code and do not change the meaning of the line, except inside comments. This makes some strange-looking input legal. The line *G0X +0.12 34Y 7* is equivalent to *G0 x+0.1234 Y7*, for example.

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

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

15.3 Block Delete

The optional block delete character the slash / when placed first on a line can be used by some user interfaces to skip lines of code when needed. In Axis the key combination Alt-m-/ toggles block delete on and off. When block delete is on any lines starting with the slash / are skipped.

15.4 Line Number

A line number is the letter N followed by an unsigned integer, optionally followed by a period and another unsigned integer. For example, N1234 and N56.78 are valid line numbers. They may be repeated or used out of order, although normal practice is to avoid such usage. Line numbers may also be skipped, and that is normal practice. A line number is not required to be used, but must be in the proper place if used.

15.5 Word

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

Words may begin with any of the letters shown in the following Table. The table includes N for completeness, even though, as defined above, line numbers are not words. Several letters (I, J, K, L, P, R) may have different meanings in different contexts. Letters which refer to axis names are not valid on a machine which does not have the corresponding axis.

Table 15.1: Words and their meanings

Letter	Meaning
A	A axis of machine
B	B axis of machine
C	C axis of machine
D	Tool radius compensation number
F	Feed rate
G	General function (See table Modal Groups)
H	Tool length offset index
I	X offset for arcs and G87 canned cycles
J	Y offset for arcs and G87 canned cycles
K	Z offset for arcs and G87 canned cycles. Spindle-Motion Ratio for G33 synchronized movements.
L	generic parameter word for G10, M66 and others
M	Miscellaneous function (See table Modal Groups)
N	Line number
P	Dwell time in canned cycles and with G4. Key used with G10.
Q	Feed increment in G73, G83 canned cycles
R	Arc radius or canned cycle plane
S	Spindle speed
T	Tool selection
U	U axis of machine
V	V axis of machine
W	W axis of machine

Table 15.1: (continued)

Letter	Meaning
X	X axis of machine
Y	Y axis of machine
Z	Z axis of machine

15.6 Number

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

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

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

Numbers used for specific purposes in RS274/NGC are often restricted to some finite set of values or some to some range of values. In many uses, decimal numbers must be close to integers; this includes the values of indexes (for parameters and carousel slot numbers, for example), M codes, and G codes multiplied by ten. A decimal number which is supposed to be close to an integer is considered close enough if it is within 0.0001 of an integer.

15.7 Parameters

The RS274/NGC language supports *parameters* - what in other programming languages would be called *variables*. There are several types of parameter of different purpose and appearance, each described in the following sections. The only value type supported by parameters is floating-point; there are no string, boolean or integer types in G-code like in other programming languages. However, logic expressions can be formulated with [boolean operators](#) (*AND*, *OR*, *XOR*, and the comparison operators *EQ*, *NE*, *GT*, *GE*, *LT*, *LE*), and the *MOD*, *ROUND*, *FUP* and *FIX* [operators](#) support integer arithmetic.

Parameters differ in syntax, scope, behavior when not yet initialized, mode, persistence and intended use.

Syntax

There are three kinds of syntactic appearance:

- *numbered* - #4711
- *named local* - #<localvalue>
- *named global* - #<_globalvalue>

Scope

The scope of a parameter is either global, or local within a subroutine. Subroutine parameters and local named variables have local scope. Global named parameters and numbered parameters starting from number 31 are global in scope. RS274/NGC uses *lexical scoping* - in a subroutine only the local variables defined therein, and any global variables are visible. The local variables of a calling procedure are not visible in a called procedure.

Behavior of uninitialized parameters

1. uninitialized global parameters, and unused subroutine parameters return the value zero when used in an expression.
2. uninitialized named parameters signal an error when used in an expression.

Mode

Most parameters are read/write and may be assigned to within an assignment statement. However, for many predefined parameters this does not make sense, so they are read-only - they may appear in expressions, but not on the left-hand side of an assignment statement.

Persistence

When LinuxCNC is shut down, volatile parameters lose their values. All parameters except numbered parameters in the current persistent range ¹ are volatile. Persistent parameters are saved in the .var file and restored to their previous values when LinuxCNC is started again. Volatile numbered parameters are reset to zero.

Intended Use

1. user parameters:: numbered parameters in the range 31..5000, and named global and local parameters except predefined parameters. These are available for general-purpose storage of floating-point values, like intermediate results, flags etc, throughout program execution. They are read/write (can be assigned a value).
2. **subroutine parameters** - these are used to hold the actual parameters passed to a subroutine.
3. **numbered parameters** - most of these are used to access offsets of coordinate systems.
4. **system parameters** - used to determine the current running version. They are read-only.

15.7.1 Numbered Parameters

A numbered parameter is the pound character # followed by an integer between 1 and (currently) 5602 ². The parameter is referred to by this integer, and its value is whatever number is stored in the parameter.

A value is stored in a parameter with the = operator; for example:

```
#3 = 15 (set parameter 3 to 15)
```

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

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

31-5000

G-Code user parameters. These parameters are global in the G Code file, and available for general use. Volatile.

5061-5069

Coordinates of a "G38.2" Probe result - X, Y, Z, A, B, C, U, V & W. Volatile.

5070

"G38" probe result - 1 if success, 0 if probe failed to close. Used with G38.3 and G38.5. Volatile.

5161-5169

"G28" Home for X, Y, Z, A, B, C, U, V & W. Persistent.

5181-5189

"G30" Home for X, Y, Z, A, B, C, U, V & W. Persistent.

¹ The range of persistent parameters may change as development progresses. This range is currently 5161- 5390. It is defined in the *_required_parameters array* in file the src/emc/rs274ngc/interp_array.cc .

² The RS274/NGC interpreter maintains an array of numbered parameters. Its size is defined by the symbol RS274NGC_MAX_PARAMETERS in the file src/emc/rs274ngc/interp_internal.hh). This number of numerical parameters may also increase as development adds support for new parameters.

5211-5219

"G92" offset for X, Y, Z, A, B, C, U, V & W. Persistent.

5210

1 if "G92" offset is currently applied, 0 otherwise. Persistent.

5211-5219

G92 offset (X Y Z A B C U V W).

5220

Coordinate System number 1 - 9 for G54 - G59.3. Persistent.

5221-5230

Coordinate System 1, G54 for X, Y, Z, A, B, C, U, V, W & R. R denotes the XY rotation angle around the Z axis. Persistent.

5241-5250

Coordinate System 2, G55 for X, Y, Z, A, B, C, U, V, W & R. Persistent.

5261-5270

Coordinate System 3, G56 for X, Y, Z, A, B, C, U, V, W & R. Persistent.

5281-5290

Coordinate System 4, G57 for X, Y, Z, A, B, C, U, V, W & R. Persistent.

5301-5310

Coordinate System 5, G58 for X, Y, Z, A, B, C, U, V, W & R. Persistent.

5321-5330

Coordinate System 6, G59 for X, Y, Z, A, B, C, U, V, W & R. Persistent.

5341-5350

Coordinate System 7, G59.1 for X, Y, Z, A, B, C, U, V, W & R. Persistent.

5361-5370

Coordinate System 8, G59.2 for X, Y, Z, A, B, C, U, V, W & R. Persistent.

5381-5390

Coordinate System 9, G59.3 for X, Y, Z, A, B, C, U, V, W & R. Persistent.

5399

Result of M66 - Check or wait for input. Volatile.

5400

Tool Number. Volatile.

5401-5409

Tool Offsets for X, Y, Z, A, B, C, U, V & W. Volatile.

5410

Tool Diameter. Volatile.

5411

Tool Front Angle. Volatile.

5412

Tool Back Angle. Volatile.

5413

Tool Orientation. Volatile.

5420-5428

Current relative position in the active coordinate system including all offsets and in the current program units for X, Y, Z, A, B, C, U, V & W, volatile.

5599

flag for controlling the output of (DEBUG,) statements. 1=output, 0=no output; default=1. Volatile.

5600

toolchanger fault indicator. Used with the iocontrol-v2 component. 1: toolchanger faulted, 0: normal. Volatile.

5601

toolchanger fault code. Used with the iocontrol-v2 component. Reflects the value of the `toolchanger-reason` HAL pin if a fault occurred. Volatile.

15.7.2 Subroutine Parameters

1-30

Subroutine local parameters of call arguments. These parameters are local to the subroutine. Volatile. See also the chapter on [O-Codes](#).

15.7.3 Named Parameters

Named parameters work like numbered parameters but are easier to read. All parameter names are converted to lower case and have spaces and tabs removed, so `<param>` and `<Param>` refer to the same parameter. Named parameters must be enclosed with `< >` marks.

`#<named parameter here>` is a local named parameter. By default, a named parameter is local to the scope in which it is assigned. You can't access a local parameter outside of its subroutine - this is so that two subroutines can use the same parameter names without fear of one subroutine overwriting the values in another.

`#<_global named parameter here>` is a global named parameter. They are accessible from within called subroutines and may set values within subroutines that are accessible to the caller. As far as scope is concerned, they act just like regular numeric parameters. They are not stored in files.

Examples:

- Declaration of named global variable

```
#<_endmill_dia> = 0.049
```

- Reference to previously declared global variable

```
#<_endmill_rad> = [#<_endmill_dia>/2.0]
```

- Mixed literal and named parameters

```
o100 call [0.0] [0.0] [#<_inside_cutout>-#<_endmill_dia>] [#<_zcut>] [#<_feedrate>]
```

Named parameters spring into existence when they are assigned a value for the first time. Local named parameters vanish when their scope is left: when a subroutine returns, all its local parameters are deleted and cannot be referred to anymore.

It is an error to use a non-existent named parameter within an expression, or at the right-hand side of an assignment. Printing the value of a non-existent named parameter with a DEBUG statement - like `(DEBUG, #<no_such_parameter>)` will display the string #####.

Global parameters, as well as local parameters assigned to at the global level, retain their value once assigned even when the program ends, and have these values when the program is run again.

The [EXISTS function](#) tests whether a given named parameter exists.

15.7.4 Predefined Named Parameters

The following global read only named parameters are available to access internal state of the interpreter and machine state. They can be used in arbitrary expressions, for instance to control flow of the program with if-then-else statements. Note that [new predefined named parameters](#) can be added easily without changes to the source code.

- #<_vmajor> - Major package version. If current version was 2.5.2 would return 2.5.
- #<_vminor> - Minor package version. If current version was 2.6.2 it would return 0.2.
- #<_line> - Sequence number. If running a G-Code file, this returns the current line number.
- #<_motion_mode> - Return the interpreter's current motion mode:

Motion mode	return value
G1	10
G2	20
G3	30
G33	330
G38.2	382
G38.3	383
G38.4	384
G38.5	385
G5.2	52
G73	730
G76	760
G80	800
G81	810
G82	820
G83	830
G84	840
G85	850
G86	860
G87	870
G88	880
G89	890

- #<_plane> - returns the value designating the current plane:

Plane	return value
G17	170
G18	180
G19	190
G17.1	171
G18.1	181
G19.1	191

- #<_ccomp> - Status of cutter compensation. Return values:

Mode	return value
G40	400

Mode	return value
G41	410
G41.1	411
G41	410
G42	420
G42.1	421

- #<_metric> - Return 1 if G21 is on, else 0.
- #<_imperial> - Return 1 if G20 is on, else 0.
- #<_absolute> - Return 1 if G90 is on, else 0.
- #<_incremental> - Return 1 if G91 is on, else 0.
- #<_inverse_time> - Return 1 if inverse feed mode (G93) is on, else 0.
- #<_units_per_minute> - Return 1 if Units/minute feed mode (G94) is on, else 0.
- #<_units_per_rev> - Return 1 if Units/revolution mode (G95) is on, else 0.
- #<_coord_system> - Return index of the current coordinate system (G54..G59.3)

Mode	return value
G54	0
G55	1
G56	2
G57	3
G58	4
G59	5
G59.1	6
G59.2	7
G59.3	8

- #<_tool_offset> - Return 1 if tool offset (G43) is on, else 0.
- #<_retract_r_plane> - Return 1 if G98 is set, else 0.
- #<_retract_old_z> - Return 1 if G99 is on, else 0.

15.7.5 System Parameters

- #<_spindle_rpm_mode> - Return 1 if spindle rpm mode (G97) is on, else 0.
- #<_spindle_css_mode> - Return 1 if constant surface speed mode (G96) is on, else 0.
- #<_ijk_absolute_mode> - Return 1 if Absolute Arc distance mode (G90.1) is on, else 0.
- #<_lathe_diameter_mode> - Return 1 if this is a lathe configuration and diameter (G7) mode is on, else 0.
- #<_lathe_radius_mode> - Return 1 if this is a lathe configuration and radius (G8) mode is on, else 0.
- #<_spindle_on> - Return 1 if spindle currently running (M3 or M4) else 0.
- #<_spindle_cw> - Return 1 if spindle direction is clockwise (M3) else 0.
- #<_mist> - Return 1 if mist (M7) is on.

- #<_flood> - Return 1 if flood (M8) is on.
- #<_speed_override> - Return 1 if feed override (M48 or M50 P1) is on, else 0.
- #<_feed_override> - Return 1 if feed override (M48 or M51 P1) is on, else 0.
- #<_adaptive_feed> - Return 1 if adaptive feed (M52 or M52 P1) is on, else 0.
- #<_feed_hold> - Return 1 if feed hold switch is enabled (M53 P1), else 0.
- #<_feed> - Return the current feed value (F).
- #<_rpm> - Return the current spindle speed (S).
- #<_x> - Return current X coordinate. Same as #5420.
- #<_y> - Return current Y coordinate. Same as #5421.
- #<_z> - Return current Z coordinate. Same as #5422.
- #<_a> - Return current A coordinate. Same as #5423.
- #<_b> - Return current B coordinate. Same as #5424.
- #<_c> - Return current C coordinate. Same as #5425.
- #<_u> - Return current U coordinate. Same as #5426.
- #<_v> - Return current V coordinate. Same as #5427.
- #<_w> - Return current W coordinate. Same as #5428.
- #<_current_tool> - Return number of the current tool in spindle. Same as #5400.
- #<_current_pocket> - Return pocket number of the current tool.
- #<_selected_tool> - Return number of the selected tool post a T code. Default -1.
- #<_selected_pocket> - Return number of the selected pocket post a T code. Default -1 (no pocket selected).
- #<_value> - Return value from the last O-word return or endsub. Default value 0 if no expression after return or endsub. Initialized to 0 on program start.
- #<_value_returned> - 1.0 if the last O-word return or endsub returned a value, 0 otherwise. Cleared by the next O-word call.
- #<_task> - 1.0 if the executing interpreter instance is part of milltask, 0.0 otherwise. Sometimes it is necessary to treat this case specially to retain proper preview, for instance when testing the success of a probe (G38.x) by inspecting #5070, which will always fail in the preview interpreter (e.g. Axis).
- #<_call_level> - current nesting level of O-word procedures. For debugging.
- #<_remap_level> - current level of the remap stack. Each remap in a block adds one to the remap level. For debugging.

15.8 Expressions

An expression is a set of characters starting with a left bracket `/` and ending with a balancing right bracket `/`. In between the brackets are numbers, parameter values, mathematical operations, and other expressions. An expression is evaluated to produce a number. The expressions on a line are evaluated when the line is read, before anything on the line is executed. An example of an expression is `[1 + acos[0] - [#3 ** [4.0/2]]]`.

15.9 Binary Operators

Binary operators only appear inside expressions. There are four basic mathematical operations: addition (+), subtraction (-), multiplication (*), and division (/). There are three logical operations: non-exclusive or (*OR*), exclusive or (*XOR*), and logical and (*AND*). The eighth operation is the modulus operation (*MOD*). The ninth operation is the *power* operation (**) of raising the number on the left of the operation to the power on the right. The relational operators are equality (*EQ*), inequality (*NE*), strictly greater than (*GT*), greater than or equal to (*GE*), strictly less than (*LT*), and less than or equal to (*LE*).

The binary operations are divided into several groups according to their precedence. (see table [Operator-Precedence](#)) If operations in different precedence groups are strung together (for example in the expression $[2.0 / 3 * 1.5 - 5.5 / 11.0]$), operations in a higher group are to be performed before operations in a lower group. If an expression contains more than one operation from the same group (such as the first / and * in the example), the operation on the left is performed first. Thus, the example is equivalent to: $[[2.0 / 3] * 1.5] - [5.5 / 11.0]$, which is equivalent to $[1.0 - 0.5]$, which is 0.5.

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

Table 15.2: Operator Precedence

Operators	Precedence
**	<i>highest</i>
* / MOD	
+ -	
EQ NE GT GE LT LE	
AND OR XOR	<i>lowest</i>

15.9.1 Equality and floating-point values

The RS274/NGC language only supports floating-point values of finite precision. Therefore, testing for equality or inequality of two floating-point values is inherently problematic. The interpreter solves this problem by considering values equal if their absolute difference is less than 0.0001 (this value is defined as `TOLERANCE_EQUAL` in `src/emc/rs274ngc/interp_internal.hh`).

15.10 Functions

A function is either *ATAN* followed by one expression divided by another expression (for example *ATAN* $[2]/[1+3]$) or any other function name followed by an expression (for example *SIN* $[90]$). The available functions are shown in table [Functions](#). Arguments to unary operations which take angle measures (*COS*, *SIN*, and *TAN*) are in degrees. Values returned by unary operations which return angle measures (*ACOS*, *ASIN*, and *ATAN*) are also in degrees.

Table 15.3: Functions

Function Name	Function result
ATAN[Y]/[X]	Four quadrant inverse tangent
ABS[arg]	Absolute value
ACOS[arg]	Inverse cosine
ASIN[arg]	Inverse sine
COS[arg]	Cosine
EXP[arg]	e raised to the given power
FIX[arg]	Round down to integer
FUP[arg]	Round up to integer
ROUND[arg]	Round to nearest integer

Table 15.3: (continued)

Function Name	Function result
LN[arg]	Base-e logarithm
SIN[arg]	Sine
SQRT[arg]	Square Root
TAN[arg]	Tangent
EXISTS[arg]	Check named Parameter

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

The EXISTS function checks for the existence of a single named parameter. It takes only one named parameter and returns 1 if it exists and 0 if it does not exist. It is an error if you use a numbered parameter or an expression. Here is an example for the usage of the EXISTS function:

```

o<test> sub
o10 if [EXISTS[#<_global>]]
    (debug, _global exists and has the value #<_global>)
o10 else
    (debug, _global does not exist)
o10 endif
o<test> endsub

o<test> call
#<_global> = 4711
o<test> call
m2

```

15.11 Repeated Items

A line may have any number of G words, but two G words from the same modal group may not appear on the same line. See the [Modal Groups](#) Section for more information.

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

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

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

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

15.12 Item order

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

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

If the second group (the parameter settings) is reordered, there will be no change in the meaning of the line unless the same parameter is set more than once. In this case, only the last setting of the parameter will take effect. For example, after the line

#3=15 #3=6 has been interpreted, the value of parameter 3 will be 6. If the order is reversed to #3=6 #3=15 and the line is interpreted, the value of parameter 3 will be 15.

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

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

15.13 Commands and Machine Modes

Many commands cause the controller to change from one mode to another, and the mode stays active until some other command changes it implicitly or explicitly. Such commands are called *modal*. For example, if coolant is turned on, it stays on until it is explicitly turned off. The G codes for motion are also modal. If a G1 (straight move) command is given on one line, for example, it will be executed again on the next line if one or more axis words is available on the line, unless an explicit command is given on that next line using the axis words or canceling motion.

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

15.14 Polar Coordinates

Polar Coordinates can be used to specify the XY coordinate of a move. The @n is the distance and ^n is the angle. The advantage of this is for things like bolt hole circles which can be done very simply by moving to a point in the center of the circle, setting the offset and then moving out to the first hole then run the drill cycle. Polar Coordinates always are from the current XY zero position. To shift the Polar Coordinates from machine zero use an offset or select a coordinate system.

In Absolute Mode the distance and angle is from the XY zero position and the angle starts with 0 on the X Positive axis and increases in a CCW direction about the Z axis. The code G1 @1^90 is the same as G1 Y1.

In Relative Mode the distance and angle is also from the XY zero position but it is cumulative. This can be confusing at first how this works in incremental mode.

For example if you have the following program you might expect it to be a square pattern.

```
F100 G1 @.5 ^90
G91 @.5 ^90
@.5 ^90
@.5 ^90
@.5 ^90
G90 G0 X0 Y0 M2
```

You can see from the following figure that the output is not what you might expect. Because we added 0.5 to the distance each time the distance from the XY zero position increased with each line.

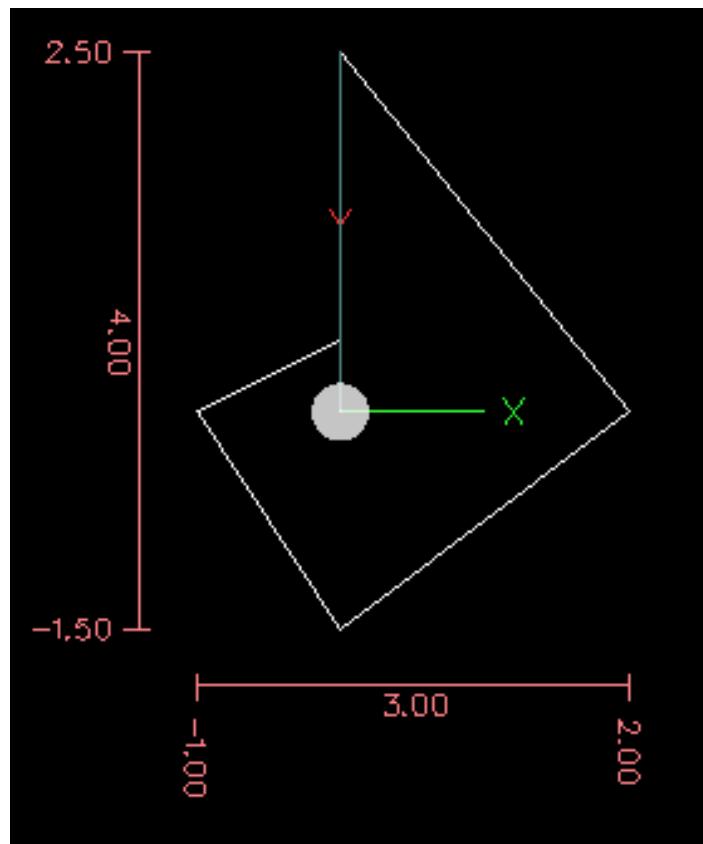


Figure 15.1: Polar Spiral

The following code will produce our square pattern.

```
F100 G1 @.5 ^90  
G91 ^90  
^90  
^90  
^90  
G90 G0 X0 Y0 M2
```

As you can see by only adding to the angle by 90 degrees each time the end point distance is the same for each line.

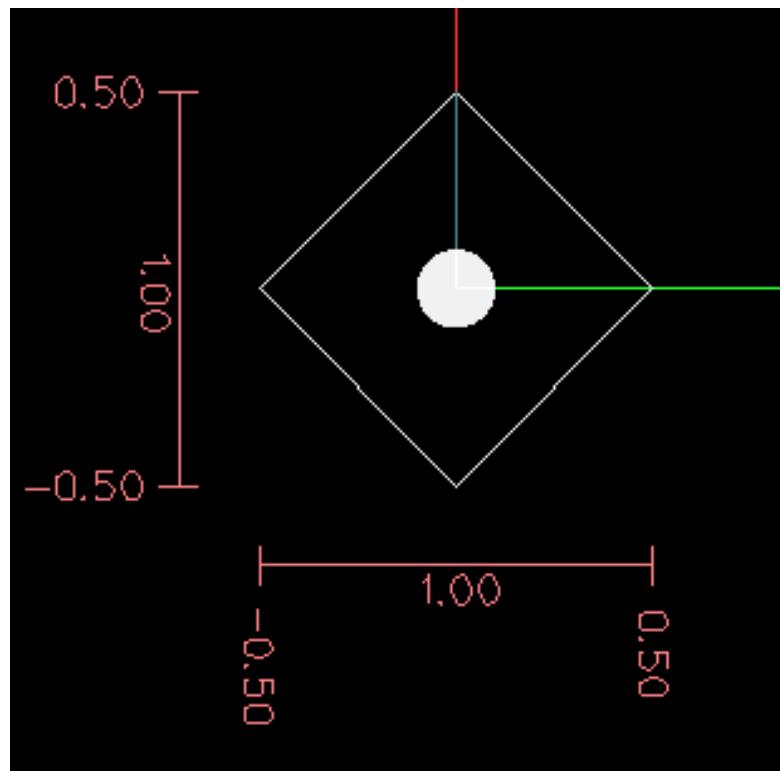


Figure 15.2: Polar Square

It is an error if:

- An incremental move is started at the origin
- A mix of Polar and X or Y words are used

15.15 Modal Groups

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

Table 15.4: G-Code Modal Groups

Modal Group Meaning	Member Words
Non-modal codes (Group 0)	G4, G10 G28, G30, G53 G92, G92.1, G92.2, G92.3,
Motion (Group 1)	G0, G1, G2, G3, G33, G38.x, G73, G76, G80, G81 G82, G83, G84, G85, G86, G87, G88, G89
Plane selection (Group 2)	G17, G18, G19, G17.1, G18.1, G19.1
Distance Mode (Group 3)	G90, G91
Arc IJK Distance Mode (Group 4)	G90.1, G91.1
Feed Rate Mode (Group 5)	G93, G94, G95
Units (Group 6)	G20, G21
Cutter Diameter Compensation (Group 7)	G40, G41, G42, G41.1, G42.1

Table 15.4: (continued)

Modal Group Meaning	Member Words
Tool Length Offset (Group 8)	G43, G43.1, G49
Canned Cycles Return Mode (Group 10)	G98, G99
Coordinate System (Group 12)	G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3
Control Mode (Group 13)	G61, G61.1, G64
Spindle Speed Mode (Group 14)	G96, G97
Lathe Diameter Mode (Group 15)	G7, G8

Table 15.5: M-Code Modal Groups

Modal Group Meaning	Member Words
Stopping (Group 4)	M0, M1, M2, M30, M60
I/O on/off (Group 5)	M6 Tn
Tool Change (Group 6)	M6 Tn
Spindle (Group 7)	M3, M4, M5
Coolant (Group 8)	(M7 M8 can both be on), M9
Override Switches (Group 9)	M48, M49
User Defined (Group 10)	M100-M199

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

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

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

It is an error to include any unrelated words on a line with *O-* flow control.

15.16 Comments

Comments can be added to lines of G code to help clear up the intention of the programmer. Comments can be embedded in a line using parentheses () or for the remainder of a line using a semi-colon. The semi-colon is not treated as the start of a comment when enclosed in parentheses.

Comments may appear between words, but not between words and their corresponding parameter. So, *S100(set speed)F200(feed)* is OK while *S(speed)100F(feed)* is not.

```
G0 (Rapid to start) X1 Y1
G0 X1 Y1 (Rapid to start; but don't forget the coolant)
M2 ; End of program.
```

There are several *active* comments which look like comments but cause some action, like *(debug...)* or *(print...)*. If there are several comments on a line, only the last comment will be interpreted according to these rules. Hence, a normal comment following an active comment will in effect disable the active comment. For example, *(foo) (debug,#1)* will print the value of parameter #1, however *(debug,#1)(foo)* will not.

A comment introduced by a semicolon is by definition the last comment on that line, and will always be interpreted for active comment syntax.

15.17 Messages

- *(MSG,)* - displays message if *MSG* appears after the left parenthesis and before any other printing characters. Variants of *MSG* which include white space and lower case characters are allowed. The rest of the characters before the right parenthesis are considered to be a message. Messages should be displayed on the message display device of the user interface if provided.

Message Example

```
(MSG, This is a message)
```

15.18 Probe Logging

- *(PROBEOPEN filename.txt)* - will open *filename.txt* and store the 9-number coordinate consisting of XYZABCUVW of each successful straight probe in it.
- *(PROBECLOSE)* - will close the open probelog file.

For more information on probing see the [G38](#) Section.

15.19 Logging

- *(LOGOPEN,filename.txt)* - opens the named log file. If the file already exists, it is truncated.
- *(LOGAPPEND,filename)* - opens the named log file. If the file already exists, the data is appended.
- *(LOGCLOSE)* - closes an open log file.
- *(LOG,)* - everything past the , is written to the log file if it is open. Supports expansion of parameters as described below.

15.20 Debug Messages

- *(DEBUG,)* - displays a message like *(MSG,)* with the addition of special handling for comment parameters as described below.

15.21 Print Messages

- *(PRINT,)* - messages are output to *stderr* with special handling for comment parameters as described below.

15.22 Comment Parameters

In the DEBUG, PRINT and LOG comments, the values of parameters in the message are expanded.

For example: to print a named global variable to *stderr* (the default console window) add a line to your G code like...

Parameters Example

```
(print,endmill dia = #<_endmill_dia>)
(print,value of variable 123 is: #123)
```

Inside the above types of comments, sequences like *#123* are replaced by the value of the parameter 123. Sequences like *#<named parameter>* are replaced by the value of the named parameter. Named parameters will have white space removed from them. So, *#<named parameter>* will be converted to *#<namedparameter>*.

15.23 File Requirements

A G code file must contain one or more lines of G code and be terminated with a [Program End](#). Any G code past the program end is not evaluated.

If a program end code is not used a pair of percent signs % with the first percent sign on the first line of the file followed by one or more lines of G code and a second percent sign. Any code past the second percent sign is not evaluated.

Note

The file must be created with a text editor like Gedit and not a word processor like Open Office Word Processor.

15.24 File Size

The interpreter and task are carefully written so that the only limit on part program size is disk capacity. The TkLinuxCNC and Axis interface both load the program text to display it to the user, though, so RAM becomes a limiting factor. In Axis, because the preview plot is drawn by default, the redraw time also becomes a practical limit on program size. The preview can be turned off in Axis to speed up loading large part programs. In Axis sections of the preview can be turned off using [preview control](#) comments.

15.25 G Code Order of Execution

The order of execution of items on a line is defined not by the position of each item on the line, but by the following list:

- O-word commands (optionally followed by a comment but no other words allowed on the same line)
- Comment (including message)
- Set feed rate mode (G93, G94).
- Set feed rate (F).
- Set spindle speed (S).
- Select tool (T).
- HAL pin I/O (M62-M68).
- Change tool (M6) and Set Tool Number (M61).
- Spindle on or off (M3, M4, M5).
- Save State (M70, M73), Restore State (M72), Invalidate State (M71).
- Coolant on or off (M7, M8, M9).
- Enable or disable overrides (M48, M49,M50,M51,M52,M53).
- User-defined Commands (M100-M199).
- Dwell (G4).
- Set active plane (G17, G18, G19).
- Set length units (G20, G21).
- Cutter radius compensation on or off (G40, G41, G42)
- Cutter length compensation on or off (G43, G49)

- Coordinate system selection (G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3).
- Set path control mode (G61, G61.1, G64)
- Set distance mode (G90, G91).
- Set retract mode (G98, G99).
- Go to reference location (G28, G30) or change coordinate system data (G10) or set axis offsets (G92, G92.1, G92.2, G94).
- Perform motion (G0 to G3, G33, G38.x, G73, G76, G80 to G89), as modified (possibly) by G53.
- Stop (M0, M1, M2, M30, M60).

15.26 G Code Best Practices

15.26.1 Use an appropriate decimal precision

Use at least 3 digits after the decimal when milling in millimeters, and at least 4 digits after the decimal when milling in inches.

15.26.2 Use consistent white space

G-code is most legible when at least one space appears before words. While it is permitted to insert white space in the middle of numbers, there is no reason to do so.

15.26.3 Use Center-format arcs

Center-format arcs (which use *I-* *J-* *K-* instead of *R-*) behave more consistently than R-format arcs, particularly for included angles near 180 or 360 degrees.

15.26.4 Put important modal settings at the top of the file

When correct execution of your program depends on modal settings, be sure to set them at the beginning of the part program. Modes can carry over from previous programs and from the MDI commands.

As a good preventative measure, put a line similar to the following at the top of all your programs:

```
G17 G20 G40 G49 G54 G80 G90 G94
```

(XY plane, inch mode, cancel diameter compensation, cancel length offset, coordinate system 1, cancel motion, non-incremental motion, feed/minute mode)

Perhaps the most critical modal setting is the distance units—if you do not include G20 or G21, then different machines will mill the program at different scales. Other settings, such as the return mode in canned cycles may also be important.

15.26.5 Don't put too many things on one line

Ignore everything in Section [Order of Execution](#), and instead write no line of code that is the slightest bit ambiguous.

15.26.6 Don't set & use a parameter on the same line

Don't use and set a parameter on the same line, even though the semantics are well defined. Updating a variable to a new value, such as `#1=#1+#2` is OK.

15.26.7 Don't use line numbers

Line numbers offer no benefits. When line numbers are reported in error messages, the numbers refer to the line number in the file, not the N-word value.

15.27 Linear and Rotary Axis

Because the meaning of an F-word in feed-per-minute mode varies depending on which axes are commanded to move, and because the amount of material removed does not depend only on the feed rate, it may be easier to use G93 inverse time feed mode to achieve the desired material removal rate.

15.28 Common Error Messages

- *G code out of range* - A G code greater than G99 was used, the scope of G codes in LinuxCNC is 0 - 99. Not every number between 0 and 99 is a valid G code.
- *Unknown g code used* - A G code was used that is not part of the LinuxCNC G code language.
- *i,j,k word with no Gx to use it* - i, j and k words must be used on the same line as the G code.
- *Cannot use axis values without a g code that uses them* - Axis values can not be used on a line without either a modal G code in effect or a G code on the same line.
- *File ended with no percent sign or program end* - Every G code file must end in a M2 or M30 or be wrapped with the percent sign %.

Appendix A

Numbered Parameters persistence

The values of parameters in the persistent range are retained over time, even if the machining center is powered down. LinuxCNC uses a parameter file to ensure persistence. It is managed by the Interpreter. The Interpreter reads the file when it starts up, and writes the file when it exits.

The format of a parameter file is shown in Table [Parameter File Format](#).

The Interpreter expects the file to have two columns. It skips any lines which do not contain exactly two numeric values. The first column is expected to contain an integer value (the parameter's number). The second column contains a floating point number (this parameter's last value). The value is represented as a double-precision floating point number inside the Interpreter, but a decimal point is not required in the file.

Parameters in the user-defined range (31-5000) may be added to this file. Such parameters will be read by the Interpreter and written to the file as it exits.

Missing Parameters in the persistent range will be initialized to zero and written with their current values on the next save operation.

The parameter numbers must be arranged in ascending order. An `Parameter file out of order` error will be signaled if they are not in ascending order.

The original file is saved as a backup file when the new file is written.

Table A.1: Parameter File Format

Parameter Number	Parameter Value
5161	0.0
5162	0.0

Chapter 16

G Codes

16.1 Conventions

Conventions used in this section

In the G code prototypes the hyphen (-) stands for a real value and (<>) denotes an optional item.

If L - is written in a prototype the - will often be referred to as the *L number*, and so on for any other letter.

In the G code prototypes the word *axes* stands for any axis as defined in your configuration.

An optional value will be written like this < L >.

A real value may be:

- An explicit number, 4
- An expression, /2+2]
- A parameter value, #88
- A unary function value, *acos*[0]

In most cases, if *axis* words are given (any or all of X Y Z A B C U V W), they specify a destination point.

Axis numbers are in the currently active coordinate system, unless explicitly described as being in the absolute coordinate system.

Where axis words are optional, any omitted axes will retain their original value.

Any items in the G code prototypes not explicitly described as optional are required.

The values following letters are often given as explicit numbers. Unless stated otherwise, the explicit numbers can be real values. For example, *G10 L2* could equally well be written *G/2*5] L[1+1]*. If the value of parameter 100 were 2, *G10 L#100* would also mean the same.

If L - is written in a prototype the - will often be referred to as the *L number*, and so on for any other letter.

16.2 G Code Quick Reference Table

Code	Description
G0	Coordinated Straight Motion Rapid Rate
G1	Coordinated Straight Motion Feed Rate
G2 G3	Coordinated Helical Motion Feed Rate
G4	Dwell

Code	Description
G5	Cubic Spline
G5.1	Quadratic B-Spline
G5.2	NURBS, add control point
G5.3	NURBS, execute
G7	Diameter Mode (lathe)
G8	Radius Mode (lathe)
G10 L1	Set Tool Table Entry
G10 L10	Set Tool Table, Calculated, Workpiece
G10 L11	Set Tool Table, Calculated, Fixture
G10 L2	Coordinate System Origin Setting
G10 L20	Coordinate System Origin Setting Calculated
G17 - G19.1	Plane Select
G20 G21	Units of Measure
G28 - G28.1	Go to Predefined Position
G30 - G30.1	Go to Predefined Position
G33	Spindle Synchronized Motion
G33.1	Rigid Tapping
G38.2 - G38.5	Probing
G40	Cancel Cutter Compensation
G41 G42	Cutter Compensation
G41.1 G42.1	Dynamic Cutter Compensation
G43	Use Tool Length Offset from Tool Table
G43.1	Dynamic Tool Length Offset
G43.2	Apply additional Tool Length Offset
G49	Cancel Tool Length Offset
G53	Motion in Machine Coordinate System
G54-G59	Select Coordinate System (1 - 6)
G59.1-G59.3	Select Coordinate System (7 - 9)
G61 G61.1	Path Control Mode
G64	Path Control Mode with Optional Tolerance
G73	Drilling Cycle with Chip Breaking
G76	Multi-pass Threading Cycle (Lathe)
G80	Cancel Motion Modes
G81	Drilling Cycle
G82	Drilling Cycle with Dwell
G83	Drilling Cycle with Peck
G85	Boring Cycle, No Dwell, Feed Out
G86	Boring Cycle, Stop, Rapid Out
G89	Boring Cycle, Dwell, Feed Out
G90 G91	Distance Mode
G90.1 G91.1	Arc Distance Mode
G92	Coordinate System Offset
G92.1 G92.2	Cancel Coordinate System Offsets
G92.3	Restore Axis Offsets
G93 G94 G95	Feed Modes
G96	Constant Surface Speed
G97	RPM Mode
G98 G99	Canned Cycle Z Retract Mode

16.3 G0 Rapid Move

G0 axes

For rapid linear (straight line) motion, program G0 'axes', where all the axis words are optional. The G0 is optional if the current

motion mode is *G0*. This will produce coordinated linear motion to the destination point at the maximum rapid rate (or slower). It is expected that cutting will not take place when a *G0* command is executing.

16.3.1 Rapid Velocity Rate

The MAX_VELOCITY setting in the ini file [TRAJ] section defines the maximum rapid traverse rate. The maximum rapid traverse rate can be higher than the individual axes MAX_VELOCITY setting during a coordinated move. The maximum rapid traverse rate can be slower than the MAX_VELOCITY setting in the [TRAJ] section if an axis MAX_VELOCITY or trajectory constraints limit it.

G0 Example

```
G90 (set absolute distance mode)
G0 X1 Y-2.3 (Rapid linear move from current location to X1 Y-2.3)
M2 (end program)
```

- See [G90](#) & [M2](#) sections for more information.

If cutter compensation is active, the motion will differ from the above; see the [Cutter Compensation](#) Section.

If *G53* is programmed on the same line, the motion will also differ; see the [G53](#) Section for more information.

The path of a *G0* rapid motion can be rounded at direction changes and depends on the [trajectory control](#) settings and maximum acceleration of the axes.

It is an error if:

- An axis letter is without a real value.
- An axis letter is used that is not configured

16.4 G1 Linear Move

G1 axes

For linear (straight line) motion at programmed [feed rate](#) (for cutting or not), program *G1* 'axes', where all the axis words are optional. The *G1* is optional if the current motion mode is *G1*. This will produce coordinated linear motion to the destination point at the current feed rate (or slower if the machine will not go that fast).

G1 Example

```
G90 (set absolute distance mode)
G1 X1.2 Y-3 F10 (linear move at a feed rate of 10 from current position to X1.2 Y-3)
Z-2.3 (linear move at same feed rate from current position to Z-2.3)
Z1 F25 (linear move at a feed rate of 25 from current position to Z1)
M2 (end program)
```

- See [G90](#) & [F](#) & [M2](#) sections for more information.

If cutter compensation is active, the motion will differ from the above; see the [Cutter Compensation](#) Section.

If *G53* is programmed on the same line, the motion will also differ; see the [G53](#) Section for more information.

It is an error if:

- No feed rate has been set.
- An axis letter is without a real value.
- An axis letter is used that is not configured

16.5 G2, G3 Arc Move

G2 or G3 axes offsets (center format)
 G2 or G3 axes R- (radius format)
 G2 or G3 offsets <P> (full circles)

A circular or helical arc is specified using either *G2* (clockwise arc) or *G3* (counterclockwise arc) at the current [feed rate](#). The direction (CW, CCW) is as viewed from the positive end of the axis about which the circular motion occurs.

The axis of the circle or helix must be parallel to the X, Y, or Z axis of the machine coordinate system. The axis (or, equivalently, the plane perpendicular to the axis) is selected with *G17* (Z-axis, XY-plane), *G18* (Y-axis, XZ-plane), or *G19* (X-axis, YZ-plane). Planes 17.1, 18.1, and 19.1 are not currently supported. If the arc is circular, it lies in a plane parallel to the selected plane.

To program a helix, include the axis word perpendicular to the arc plane: for example, if in the *G17* plane, include a *Z* word. This will cause the *Z* axis to move to the programmed value during the circular *XY* motion.

To program an arc that gives more than one full turn, use the *P* word specifying the number of full turns plus the programmed arc. The *P* word must be an integer. If *P* is unspecified, the behavior is as if *P1* was given: that is, only one full or partial turn will result. For example, if a 180 degree arc is programmed with a *P2*, the resulting motion will be 1 1/2 rotations. For each *P* increment above 1 an extra full circle is added to the programmed arc. Multi turn helical arcs are supported and give motion useful for milling holes or threads.

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

If cutter compensation is active, the motion will differ from the above; see the [Cutter Compensation](#) Section.

The arc center is absolute or relative as set by [G90.1](#) or [G91.1](#) respectively.

Two formats are allowed for specifying an arc: Center Format and Radius Format.

It is an error if:

- No feed rate has been set.
- The *P* word is not an integer.

16.5.1 Center Format Arcs

Center format arcs are more accurate than radius format arcs and are the preferred format to use.

The end point of the arc along with the offset to the center of the arc from the current location are used to program arcs that are less than a full circle. It is OK if the end point of the arc is the same as the current location.

The offset to the center of the arc from the current location and optionally the number of turns are used to program full circles.

When programming arcs an error due to rounding can result from using a precision of less than 4 decimal places (0.0000) for inch and less than 3 decimal places (0.000) for millimeters.

Incremental Arc Distance Mode Arc center offsets are a relative distance from the start location of the arc. Incremental Arc Distance Mode is default.

One or more axis words and one or more offsets must be programmed for an arc that is less than 360 degrees.

No axis words and one or more offsets must be programmed for full circles. The *P* word defaults to 1 and is optional.

For more information on 'Incremental Arc Distance Mode' see the [G91.1](#) section.

Absolute Arc Distance Mode Arc center offsets are the absolute distance from the current 0 position of the axis.

One or more axis words and *both* offsets must be programmed for arcs less than 360 degrees.

No axis words and both offsets must be programmed for full circles. The *P* word defaults to 1 and is optional.

For more information on 'Absolute Arc Distance Mode' see the [G90.1](#) section.

XY-plane (G17)

```
G2 or G3 <X- Y- Z- I- J- P->
```

- Z - helix
- I - X offset
- J - Y offset
- P - number of turns

XZ-plane (G18)

```
G2 or G3 <X- Z- Y- I- K- P->
```

- Y - helix
- I - X offset
- K - Z offset
- P - number of turns

YZ-plane (G19)

```
G2 or G3 <Y- Z- X- J- K- P->
```

- X - helix
- J - Y offset
- K - Z offset
- P - number of turns

It is an error if:

- No feed rate is set with the **F** word.
- No offsets are programmed.
- When the arc is projected on the selected plane, the distance from the current point to the center differs from the distance from the end point to the center by more than (.05 inch/.5 mm) OR (.0005 inch/.005mm) AND .1% of radius.

Deciphering the Error message *Radius to end of arc differs from radius to start:*

- *start* - the current position
- *center* - the center position as calculated using the i,j or k words
- *end* - the programmed end point
- *r1* - radius from the start position to the center
- *r2* - radius from the end position to the center

16.5.2 Center Format Examples

Calculating arcs by hand can be difficult at times. One option is to draw the arc with a cad program to get the coordinates and offsets. Keep in mind the tolerance mentioned above, you may have to change the precision of your cad program to get the desired results. Another option is to calculate the coordinates and offset using formulas. As you can see in the following figures a triangle can be formed from the current position the end position and the arc center.

In the following figure you can see the start position is X0 Y0, the end position is X1 Y1. The arc center position is at X1 Y0. This gives us an offset from the start position of 1 in the X axis and 0 in the Y axis. In this case only an I offset is needed.

G2 Example Line

```
G0 X0 Y0
G2 X1 Y1 I1 F10 (clockwise arc in the XY plane)
```

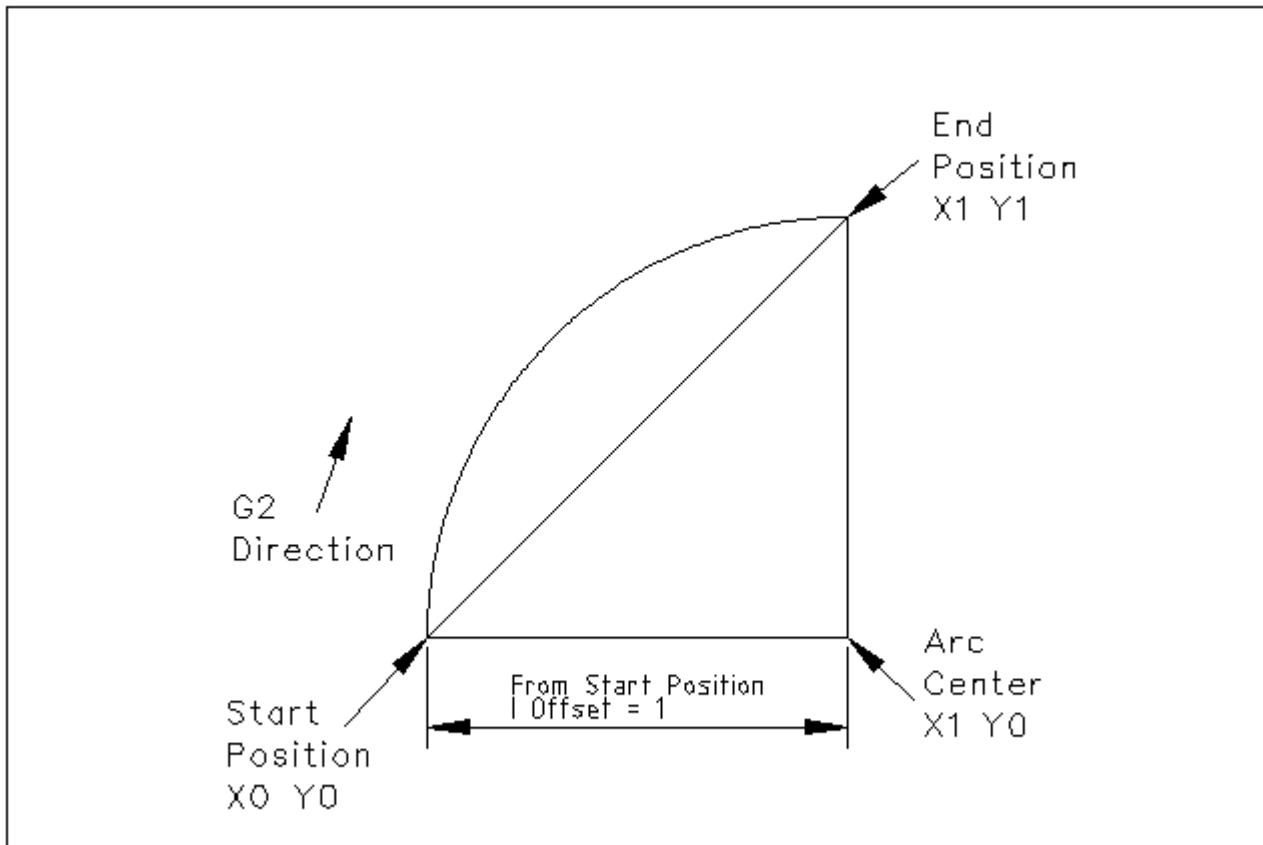


Figure 16.1: G2 Example

In the next example we see the difference between the offsets for Y if we are doing a G2 or a G3 move. For the G2 move the start position is X0 Y0, for the G3 move it is X0 Y1. The arc center is at X1 Y0.5 for both moves. The G2 move the J offset is 0.5 and the G3 move the J offset is -0.5.

G2-G3 Example Line

```
G0 X0 Y0
```

```
G2 X0 Y1 I1 J0.5 F25 (clockwise arc in the XY plane)
G3 X0 Y0 I1 J-0.5 F25 (counterclockwise arc in the XY plane)
```

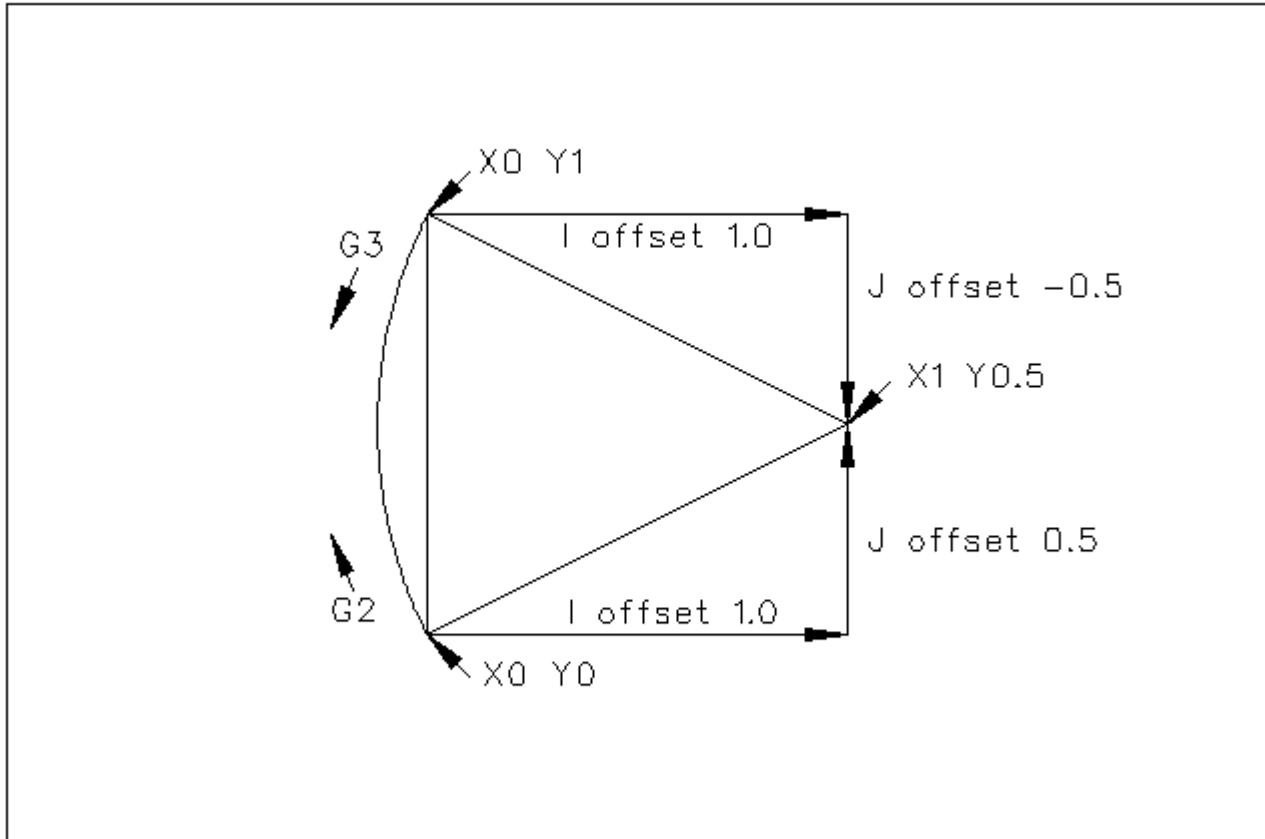


Figure 16.2: G2-G3 Example

In the next example we show how the arc can make a helix in the Z axis by adding the Z word.

G2 Example Helix

```
G0 X0 Y0 Z0
G17 G2 X10 Y16 I3 J4 Z-1 (helix arc with Z added)
```

In the next example we show how to make more than one turn using the P word.

P word Example

```
G0 X0 Y0 Z0
G2 X0 Y1 Z-1 I1 J0.5 P2 F25
```

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

16.5.3 Radius Format Arcs

```
G2 or G3 axes R-
```

- R - radius from current position

It is not good practice to program radius format arcs that are nearly full circles or nearly semicircles because a small change in the location of the end point will produce a much larger change in the location of the center of the circle (and, hence, the middle of the arc). The magnification effect is large enough that rounding error in a number can produce out-of-tolerance cuts. For instance, a 1% displacement of the endpoint of a 180 degree arc produced a 7% displacement of the point 90 degrees along the arc. Nearly full circles are even worse. Other size arcs (in the range tiny to 165 degrees or 195 to 345 degrees) are OK.

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

It is an error if:

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

G2 Example Line

```
G17 G2 X10 Y15 R20 Z5 (radius format with arc)
```

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

16.6 G4 Dwell

```
G4 P-
```

- P - seconds to dwell (floating point)

The P number is the time in seconds that all axes will remain unmoving. The P number is a floating point number so fractions of a second may be used. *G4* does not affect spindle, coolant and any I/O.

G4 Example Line

```
G4 P0.5 (wait for 0.5 seconds before proceeding)
```

It is an error if:

- the P number is negative or not specified.

16.7 G5 Cubic spline

```
G5 X- Y- <I- J-> P- Q-
```

- *I* - X incremental offset from start point to first control point
- *J* - Y incremental offset from start point to first control point
- *P* - X incremental offset from end point to second control point
- *Q* - Y incremental offset from end point to second control point

G5 creates a cubic B-spline in the XY plane with the X and Y axes only. P and Q must both be specified for every G5 command.

For the first G5 command in a series of G5 commands, I and J must both be specified. For subsequent G5 commands, either both I and J must be specified, or neither. If I and J are unspecified, the starting direction of this cubic will automatically match the ending direction of the previous cubic (as if I and J are the negation of the previous P and Q).

For example, to program a curvy N shape:

G5 Sample initial cubic spline

```
G90 G17
G0 X0 Y0
G5 I0 J3 P0 Q-3 X1 Y1
```

A second curvy N that attaches smoothly to this one can now be made without specifying I and J:

G5 Sample subsequent cubic spline

```
G5 P0 Q-3 X2 Y2
```

It is an error if:

- P and Q are not both specified
- Just one of I or J are specified
- I or J are unspecified in the first of a series of G5 commands
- An axis other than X or Y is specified
- The active plane is not G17

16.8 G5.1 Quadratic spline

```
G5.1 X- Y- I- J-
```

- *I* - X incremental offset from start point to control point
- *J* - Y incremental offset from start point to control point

G5.1 creates a quadratic B-spline in the XY plane with the X and Y axis only. Not specifying I or J gives zero offset for the unspecified axis, so one or both must be given.

For example, to program a parabola, through the origin, from X-2 Y4 to X2 Y4:

G5.1 Sample quadratic spline

```
G90 G17  
G0 X-2 Y4  
G5.1 X2 I2 J-8
```

It is an error if:

- both I and J offset are unspecified or zero
- An axis other than X or Y is specified
- The active plane is not G17

16.9 G5.2 G5.3 NURBS Block

```
G5.2 <P-> <X- Y-> <L->  
X- Y- <P->  
...  
G5.3
```

Warning: G5.2, G5.3 is experimental and not fully tested.

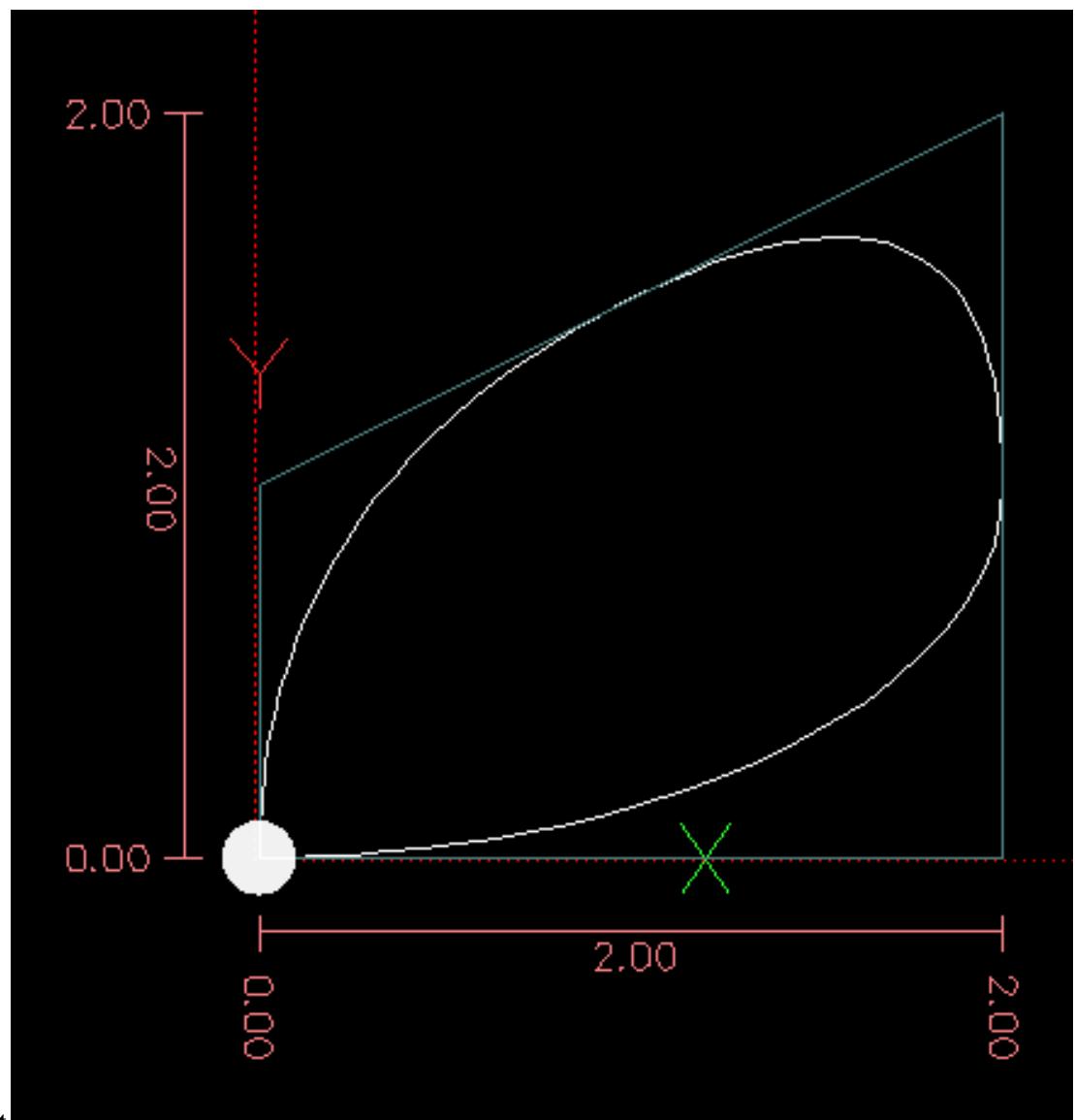
G5.2 is for opening the data block defining a NURBS and G5.3 for closing the data block. In the lines between these two codes the curve control points are defined with both their related *weights* (P) and the parameter (L) which determines the order of the curve.

The current coordinate, before the first G5.2 command, is always taken as the first NURBS control point. To set the weight for this first control point, first program G5.2 P- without giving any X Y.

The default weight if P is unspecified is 1. The default order if L is unspecified is 3.

G5.2 Example

```
G0 X0 Y0 (rapid move)  
F10 (set feed rate)  
G5.2 P1 L3  
    X0 Y1 P1  
    X2 Y2 P1  
    X2 Y0 P1  
    X0 Y0 P2  
G5.3  
; The rapid moves show the same path without the NURBS Block  
G0 X0 Y1  
    X2 Y2  
    X2 Y0  
    X0 Y0  
M2
```



Sample NURBS Output

More information on NURBS can be found here:

<http://wiki.linuxcnc.org/cgi-bin/wiki.pl?NURBS>

16.10 G7 Lathe Diameter Mode

G7

Program G7 to enter the diameter mode for axis X on a lathe. When in the diameter mode the X axis moves on a lathe will be 1/2 the distance to the center of the lathe. For example X1 would move the cutter to 0.500" from the center of the lathe thus giving a 1" diameter part.

16.11 G8 Lathe Radius Mode

G8

Program G8 to enter the radius mode for axis X on a lathe. When in Radius mode the X axis moves on a lathe will be the distance from the center. Thus a cut at X1 would result in a part that is 2" in diameter. G8 is default at power up.

16.12 G10 L1 Set Tool Table

```
G10 L1 P- axes <R- I- J- Q->
```

- P - tool number
- R - radius of tool
- I - front angle (lathe)
- J - back angle (lathe)
- Q - orientation (lathe)

G10 L1 sets the tool table for the P tool number to the values of the words.

A valid G10 L1 rewrites and reloads the tool table.

G10 L1 Example Line

```
G10 L1 P1 Z1.5 (set tool 1 Z offset from the machine origin to 1.5)
G10 L1 P2 R0.015 Q3 (lathe example setting tool 2 radius to 0.015 and orientation to 3)
```

It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

For more information on cutter orientation used by the Q word, see the [Lathe Tool Orientation](#) diagram.

16.13 G10 L2 Set Coordinate System

```
G10 L2 P- <axes R->
```

- P - coordinate system (0-9)
- R - rotation about the Z axis

G10 L2 offsets the origin of the axes in the coordinate system specified to the value of the axis word. The offset is from the machine origin established during homing. The offset value will replace any current offsets in effect for the coordinate system specified. Axis words not used will not be changed.

Program P0 to P9 to specify which coordinate system to change.

Table 16.1: Coordinate System

P Value	Coordinate System	G code
0	Active	n/a
1	1	G54
2	2	G55
3	3	G56
4	4	G57
5	5	G58
6	6	G59
7	7	G59.1
8	8	G59.2
9	9	G59.3

Optionally program R to indicate the rotation of the XY axis around the Z axis. The direction of rotation is CCW as viewed from the positive end of the Z axis.

All axis words are optional.

Being in incremental distance mode ([G91](#)) has no effect on *G10 L2*.

Important Concepts:

- G10 L2 Pn does not change from the current coordinate system to the one specified by P, you have to use G54-59.3 to select a coordinate system.
- When a rotation is in effect jogging an axis will only move that axis in a positive or negative direction and not along the rotated axis.
- If a G92 origin offset was in effect before *G10 L2*, it will continue to be in effect afterwards.
- The coordinate system whose origin is set by a *G10* command may be active or inactive at the time the *G10* is executed. If it is currently active, the new coordinates take effect immediately.

It is an error if:

- The P number does not evaluate to an integer in the range 0 to 9.
- An axis is programmed that is not defined in the configuration.

G10 L2 Example Line

```
G10 L2 P1 X3.5 Y17.2
```

In the above example the origin of the first coordinate system (the one selected by *G54*) is set to be X=3.5 and Y=17.2. Because only X and Y are specified, the origin point is only moved in X and Y; the other coordinates are not changed.

G10 L2 Example Line

```
G10 L2 P1 X0 Y0 Z0 (clear offsets for X,Y & Z axes in coordinate system 1)
```

The above example sets the XYZ coordinates of the coordinate system 1 to the machine origin.

The coordinate system is described in the [Coordinate System](#) Section.

16.14 G10 L10 Set Tool Table

```
G10 L10 P- axes <R- I- J- Q->
```

- P - tool number
- R - radius of tool
- I - front angle (lathe)
- J - back angle (lathe)
- Q - orientation (lathe)

G10 L10 changes the tool table entry for tool P so that if the tool offset is reloaded, with the machine in its current position and with the current G5x and G92 offsets active, the current coordinates for the given axes will become the given values. The axes that are not specified in the G10 L10 command will not be changed. This could be useful with a probe move as described in the [G38](#) section.

G10 L10 Example

```
T1 M6 G43 (load tool 1 and tool length offsets)
G10 L10 P1 Z1.5 (set the current position for Z to be 1.5)
G43 (reload the tool length offsets from the changed tool table)
M2 (end program)
```

- See [T & M6](#), and [G43/G43.1](#) sections for more information.

It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

16.15 G10 L11 Set Tool Table

```
G10 L11 P- axes <R- I- J- Q->
```

- P - tool number
- R - radius of tool
- I - front angle (lathe)
- J - back angle (lathe)
- Q - orientation (lathe)

G10 L11 is just like G10 L10 except that instead of setting the entry according to the current offsets, it is set so that the current coordinates would become the given value if the new tool offset is reloaded and the machine is placed in the G59.3 coordinate system without any G92 offset active.

This allows the user to set the G59.3 coordinate system according to a fixed point on the machine, and then use that fixture to measure tools without regard to other currently-active offsets.

It is an error if:

- Cutter Compensation is on
- The P number is unspecified
- The P number is not a valid tool number from the tool table
- The P number is 0

16.16 G10 L20 Set Coordinate System

```
G10 L20 P- axes
```

- *P* - coordinate system (0-9)

G10 L20 is similar to G10 L2 except that instead of setting the offset/entry to the given value, it is set to a calculated value that makes the current coordinates become the given value.

G10 L20 Example Line

```
G10 L20 P1 X1.5 (set the X axis current location in coordinate system 1 to 1.5)
```

It is an error if:

- The P number does not evaluate to an integer in the range 0 to 9.
- An axis is programmed that is not defined in the configuration.

16.17 G17 - G19.1 Plane Selection

These codes set the current plane as follows:

- *G17* - XY (default)
- *G18* - ZX
- *G19* - YZ
- *G17.1* - UV
- *G18.1* - WU
- *G19.1* - VW

The UV, WU and VW planes do not support arcs.

It is a good idea to include a plane selection in the preamble of each G code file.

The effects of having a plane selected are discussed in Section [G2 G3](#) and Section [G81 G89](#)

16.18 G20, G21 Units

- *G20* - to use inches for length units.
- *G21* - to use millimeters for length units.

It is a good idea to include units in the preamble of each G code file.

16.19 G28, G28.1 Go to Predefined Position

**Warning**

Only use G28 when your machine is homed to a repeatable position and the desired G28 position has been stored with G28.1.

G28 uses the values stored in [parameters](#) 5161-5166 as the X Y Z A B C U V W final point to move to. The parameter values are *absolute* machine coordinates in the native machine *units* as specified in the ini file. All axes defined in the ini file will be moved when a G28 is issued.

- *G28* - makes a [rapid move](#) from the current position to the *absolute* position of the values in parameters 5161-5166.
- *G28 axes* - makes a [rapid move](#) to the position specified by *axes* including any offsets, then will make a rapid move to the *absolute* position of the values in parameters 5161-5166 for *axes* specified. Any *axis* not specified will not move.
- *G28.1* - stores the current *absolute* position into parameters 5161-5166.

G28 Example Line

```
G28 Z2.5 (rapid to Z2.5 then to location specified in the G28 stored parameters)
```

It is an error if :

- Cutter Compensation is turned on

16.20 G30, G30.1 Go to Predefined Position

**Warning**

Only use G30 when your machine is homed to a repeatable position and the desired G30 position has been stored with G30.1.

G30 functions the same as G28 but uses the values stored in [parameters](#) 5181-5186 as the X Y Z A B C U V W final point to move to. The parameter values are *absolute* machine coordinates in the native machine *units* as specified in the ini file. All axes defined in the ini file will be moved when a G30 is issued.

Note

G30 parameters will be used to move the tool when a M6 is programmed if TOOL_CHANGE_AT_G30=1 is in the [EMCIO] section of the ini file.

- *G30* - makes a [rapid move](#) from the current position to the *absolute* position of the values in parameters 5181-5186.
- *G30 axes* - makes a [rapid move](#) to the position specified by *axes* including any offsets, then will make a rapid move to the *absolute* position of the values in parameters 5181-5186 for *axes* specified. Any *axis* not specified will not move.
- *G30.1* - stores the current absolute position into parameters 5181-5186.

G30 Example Line

```
G30 Z2.5 (rapid to Z2.5 then to the location specified in the G30 stored parameters)
```

It is an error if :

- Cutter Compensation is turned on

16.21 G33 Spindle Synchronized Motion

G33 X- Y- Z- K-

- K - distance per revolution

For spindle-synchronized motion in one direction, code $G33 X- Y- Z- K-$ where K gives the distance moved in XYZ for each revolution of the spindle. For instance, if starting at $Z=0$, $G33 Z-1 K.0625$ produces a 1 inch motion in Z over 16 revolutions of the spindle. This command might be part of a program to produce a 16TPI thread. Another example in metric, $G33 Z-15 K1.5$ produces a movement of 15mm while the spindle rotates 10 times for a thread of 1.5mm.

Spindle-synchronized motion waits for the spindle index and spindle at speed pins, so multiple passes line up. $G33$ moves end at the programmed endpoint. $G33$ could be used to cut tapered threads or a fusee.

All the axis words are optional, except that at least one must be used.

Note

K follows the drive line described by $X- Y- Z-$. K is not parallel to the Z axis if X or Y endpoints are used for example when cutting tapered threads.

Technical Info At the beginning of each $G33$ pass, LinuxCNC uses the spindle speed and the machine acceleration limits to calculate how long it will take Z to accelerate after the index pulse, and determines how many degrees the spindle will rotate during that time. It then adds that angle to the index position and computes the Z position using the corrected spindle angle. That means that Z will reach the correct position just as it finishes accelerating to the proper speed, and can immediately begin cutting a good thread.

HAL Connections The pins *motion.spindle-at-speed* and the *encoder.n.phase-Z* for the spindle must be connected in your HAL file before $G33$ will work. See the Integrators Manual for more information on spindle synchronized motion.

G33 Example

```
G90 (absolute distance mode)
G0 X1 Z0.1 (rapid to position)
S100 M3 (start spindle turning)
G33 Z-2 K0.125 (move Z axis to -2 at a rate to equal 0.125 per revolution)
G0 X1.25 (rapid move tool away from work)
Z0.1 (rapid move to starting Z position)
M2 (end program)
```

- See [G90](#) & [G0](#) & [M2](#) sections for more information.

It is an error if:

- All axis words are omitted.
- The spindle is not turning when this command is executed
- The requested linear motion exceeds machine velocity limits due to the spindle speed

16.22 G33.1 Rigid Tapping

G33.1 X- Y- Z- K-

- K - distance per revolution

For rigid tapping (spindle synchronized motion with return), code *G33.1 X- Y- Z- K-* where *K-* gives the distance moved for each revolution of the spindle. A rigid tapping move consists of the following sequence:

**Warning**

If the X Y coordinates specified are not the current coordinates when calling G33.1 for tapping the move will not be along the Z axis but will **rapid move** from the current location to the X Y location specified.

1. A move to the specified coordinate, synchronized with the spindle at the given ratio and starting with a spindle index pulse.
2. When reaching the endpoint, a command to reverse the spindle (e.g., from clockwise to counterclockwise).
3. Continued synchronized motion beyond the specified end coordinate until the spindle actually stops and reverses.
4. Continued synchronized motion back to the original coordinate.
5. When reaching the original coordinate, a command to reverse the spindle a second time (e.g., from counterclockwise to clockwise).
6. Continued synchronized motion beyond the original coordinate until the spindle actually stops and reverses.
7. An **unsynchronized** move back to the original coordinate.

Spindle-synchronized motions wait for spindle index, so multiple passes line up. *G33.1* moves end at the original coordinate.

All the axis words are optional, except that at least one must be used.

G33.1 Example

```
G90 (set absolute mode)
G0 X1.000 Y1.000 Z0.100 (rapid move to starting position)
G33.1 Z-0.750 K0.05 (rigid tap a 20 TPI thread 0.750 deep)
M2 (end program)
```

- See [G90](#) & [G0](#) & [M2](#) sections for more information.

It is an error if:

- All axis words are omitted.
- The spindle is not turning when this command is executed
- The requested linear motion exceeds machine velocity limits due to the spindle speed

16.23 G38.x Straight Probe

```
G38.x axes
```

- *G38.2* - probe toward workpiece, stop on contact, signal error if failure
- *G38.3* - probe toward workpiece, stop on contact
- *G38.4* - probe away from workpiece, stop on loss of contact, signal error if failure
- *G38.5* - probe away from workpiece, stop on loss of contact

**Important**

You will not be able to use a probe move until your machine has been set up to provide a probe input signal. The probe input signal must be connected to *motion.probe-input* in a .hal file. G38.x uses motion.probe-input to determine when the probe has made (or lost) contact. TRUE for probe contact closed (touching), FALSE for probe contact open.

Program *G38.x axes* to perform a straight probe operation. The axis words are optional, except that at least one of them must be used. The axis words together define the destination point that the probe will move towards, starting from the current location. If the probe is not tripped before the destination is reached G38.2 and G38.4 will signal an error.

The tool in the spindle must be a probe or contact a probe switch.

In response to this command, the machine moves the controlled point (which should be at the center of the probe ball) in a straight line at the current [feed rate](#) toward the programmed point. In inverse time feed mode, the feed rate is such that the whole motion from the current point to the programmed point would take the specified time. The move stops (within machine acceleration limits) when the programmed point is reached, or when the requested change in the probe input takes place, whichever occurs first.

After successful probing, parameters 5061 to 5069 will be set to the coordinates of X, Y, Z, A, B, C, U, V, W of the location of the controlled point at the time the probe changed state. After unsuccessful probing, they are set to the coordinates of the programmed point. Parameter 5070 is set to 1 if the probe succeeded and 0 if the probe failed. If the probing operation failed, G38.2 and G38.4 will signal an error by posting a message on screen if the selected GUI supports that. And by halting program execution.

A comment of the form (*PROBEOPEN filename.txt*) will open *filename.txt* and store the 9-number coordinate consisting of XYZABCUVW of each successful straight probe in it. The file must be closed with (*PROBECLOSE*). For more information see the [Comments](#) Section.

An example file *smartprobe.ngc* is included (in the examples directory) to demonstrate using probe moves to log to a file the coordinates of a part. The program *smartprobe.ngc* could be used with *ngcgui* with minimal changes.

It is an error if:

- the current point is the same as the programmed point.
- no axis word is used
- cutter compensation is enabled
- the feed rate is zero
- the probe is already in the target state

16.24 G40 Compensation Off

- *G40* - turn cutter compensation off. If tool compensation was on the next move must be a linear move and longer than the tool diameter. It is OK to turn compensation off when it is already off.

G40 Example

```
; current location is X1 after finishing cutter compensated move
G40 (turn compensation off)
G0 X1.6 (linear move longer than current cutter diameter)
M2 (end program)
```

See [G0](#) & [M2](#) sections for more information.

It is an error if:

- A G2/G3 arc move is programmed next after a G40.
- The linear move after turning compensation off is less than the tool diameter.

16.25 G41, G42 Cutter Compensation

```
G41 <D-> (left of programmed path)
G42 <D-> (right of programmed path)
```

- *D* - tool number

The D word is optional; if there is no D word the radius of the currently loaded tool will be used (if no tool is loaded and no D word is given, a radius of 0 will be used).

If supplied, the D word is the tool number to use. This would normally be the number of the tool in the spindle (in which case the D word is redundant and need not be supplied), but it may be any valid tool number.

Note

G41/G42 D0 is a little special. Its behavior is different on random tool changer machines and nonrandom tool changer machines (see the [Tool Changers](#) section). On nonrandom tool changer machines, *G41/G42 D0* applies the TLO of the tool currently in the spindle, or a TLO of 0 if no tool is in the spindle. On random tool changer machines, *G41/G42 D0* applies the TLO of the tool T0 defined in the tool table file (or causes an error if T0 is not defined in the tool table).

To start cutter compensation to the left of the part profile, use G41. G41 starts cutter compensation to the left of the programmed line as viewed from the positive end of the axis perpendicular to the plane.

To start cutter compensation to the right of the part profile, use G42. G42 starts cutter compensation to the right of the programmed line as viewed from the positive end of the axis perpendicular to the plane.

The lead in move must be at least as long as the tool radius. The lead in move can be a rapid move.

Cutter compensation may be performed if the XY-plane or XZ-plane is active.

User M100-M199 commands are allowed when Cutter Compensation is on.

The behavior of the machining center when cutter compensation is on is described in the [Cutter Compensation](#) Section along with code examples.

It is an error if:

- The D number is not a valid tool number or 0.
- The YZ plane is active.
- Cutter compensation is commanded to turn on when it is already on.

16.26 G41.1, G42.1 Dynamic Cutter Compensation

```
G41.1 D- <L-> (left of programmed path)
G42.1 D- <L-> (right of programmed path)
```

- *D* - cutter diameter

- *L* - tool orientation (see [lathe tool orientation](#))

G41.1 & G42.1 function the same as G41 & G42 with the added scope of being able to program the tool diameter. The L word defaults to 0 if unspecified.

It is an error if:

- The YZ plane is active.
- The L number is not in the range from 0 to 9 inclusive.
- The L number is used when the XZ plane is not active.
- Cutter compensation is commanded to turn on when it is already on.

16.27 G43 Tool Length Offset

```
G43 <H->
```

- *H* - tool number (optional)

G43 enables tool length compensation. G43 changes subsequent motions by offsetting the axis coordinates by the length of the offset. G43 does not cause any motion. The next time a compensated axis is moved, that axis's endpoint is the compensated location.

G43 without an H word uses the currently loaded tool from the last *Tn M6*.

G43 Hn uses the offset for tool n.

Note

G43 H0 is a little special. Its behavior is different on random tool changer machines and nonrandom tool changer machines (see the [Tool Changers](#) section). On nonrandom tool changer machines, *G43 H0* applies the TLO of the tool currently in the spindle, or a TLO of 0 if no tool is in the spindle. On random tool changer machines, *G43 H0* applies the TLO of the tool T0 defined in the tool table file (or causes an error if T0 is not defined in the tool table).

G43 H- Example Line

```
G43 H1 (set tool offsets using the values from tool 1 in the tool table)
```

It is an error if:

- the H number is not an integer, or
- the H number is negative, or
- the H number is not a valid tool number (though note that 0 is a valid tool number on nonrandom tool changer machines, it means "the tool currently in the spindle")

16.28 G43.1: Dynamic Tool Length Offset

```
G43.1 axes
```

- *G43.1 axes* - change subsequent motions by offsetting the Z and/or X offsets stored in the tool table. G43.1 does not cause any motion. The next time a compensated axis is moved, that axis's endpoint is the compensated location.

G43.1 Example

```
G90 (set absolute mode)
T1 M6 G43 (load tool 1 and tool length offsets, Z is at machine 0 and DRO shows Z1.500)
G43.1 Z0.250 (offset current tool offset by 0.250, DRO now shows Z1.250)
M2 (end program)
```

- See [G90](#) & [T](#) & [M6](#) sections for more information.

It is an error if:

- motion is commanded on the same line as *G43.1*

16.29 G43.2: Apply additional Tool Length Offset

G43.2 H-

- H - tool number

G43.2 applies an additional simultaneous tool offset.

G43.2 Example

```
G90 (set absolute mode)
T1 M6 (load tool 1)
G43 (or G43 H1 - replace all tool offsets with T1's offset)
G43.2 H10 (also add in T10's tool offset)
M2 (end program)
```

You can sum together an arbitrary number of offsets by calling G43.2 more times. There are no built-in assumptions about which numbers are geometry offsets and which are wear offsets, or that you should have only one of each.

Like the other G43 commands, G43.2 does not cause any motion. The next time a compensated axis is moved, that axis's endpoint is the compensated location.

It is an error if:

- H is unspecified, or
- the given tool number does not exist in the tool table

16.30 G49: Cancel Tool Length Compensation

- G49 - cancels tool length compensation

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

16.31 G53 Move in Machine Coordinates

G53 axes

To move in the machine coordinate system, program G53 on the same line as a linear move. G53 is not modal and must be programmed on each line. G0 or G1 does not have to be programmed on the same line if one is currently active. For example G53 G0 X0 Y0 Z0 will move the axes to the home position even if the currently selected coordinate system has offsets in effect.

G53 Example Line

```
G53 G0 X0 Y0 Z0 (rapid linear move to the machine origin)
G53 X2 (rapid linear move to absolute coordinate X2)
```

- See [G0](#) section for more information.

It is an error if:

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

16.32 G54-G59.3 Select Coordinate System

- *G54* - select coordinate system 1
- *G55* - select coordinate system 2
- *G56* - select coordinate system 3
- *G57* - select coordinate system 4
- *G58* - select coordinate system 5
- *G59* - select coordinate system 6
- *G59.1* - select coordinate system 7
- *G59.2* - select coordinate system 8
- *G59.3* - select coordinate system 9

The coordinate systems store the axis values and the XY rotation angle around the Z axis in the parameters shown in the following table.

Table 16.2: Coordinate System Parameters

Select	CS	X	Y	Z	A	B	C	U	V	W	R
G54	1	5221	5222	5223	5224	5225	5226	5227	5228	5229	5230
G55	2	5241	5242	5243	5244	5245	5246	5247	5248	5249	5250
G56	3	5261	5262	5263	5264	5265	5266	5267	5268	5269	5270
G57	4	5281	5282	5283	5284	5285	5286	5287	5288	5289	5290
G58	5	5301	5302	5303	5304	5305	5306	5307	5308	5309	5310
G59	6	5321	5322	5323	5324	5325	5326	5327	5328	5329	5330
G59.1	7	5341	5342	5343	5344	5345	5346	5347	5348	5349	5350
G59.2	8	5361	5362	5363	5364	5365	5366	5367	5368	5369	5370
G59.3	9	5381	5382	5383	5384	5385	5386	5387	5388	5389	5390

It is an error if:

- selecting a coordinate system is used while cutter compensation is on.

See the [Coordinate System](#) Section for an overview of coordinate systems.

16.33 G61, G61.1 Exact Path Mode

- *G61* - exact path mode. G61 visits the programmed point exactly, even though that means temporarily coming to a complete stop.
- *G61.1* - exact stop mode. Same as G61

16.34 G64 Path Blending

```
G64 <P- <Q->>
```

- P - motion blending tolerance
- Q - naive cam tolerance
- $G64$ - best possible speed.
- $G64 P- <Q- >$ blending with tolerance.
- $G64$ - without P means to keep the best speed possible, no matter how far away from the programmed point you end up.
- $G64 P- Q-$ is a way to fine tune your system for best compromise between speed and accuracy. The P - tolerance means that the actual path will be no more than P - away from the programmed endpoint. The velocity will be reduced if needed to maintain the path. In addition, when you activate $G64 P- Q-$ it turns on the *naive cam detector*; when there are a series of linear XYZ feed moves at the same *feed rate* that are less than Q - away from being collinear, they are collapsed into a single linear move. On G2/G3 moves in the G17 (XY) plane when the maximum deviation of an arc from a straight line is less than the $G64 P$ - 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. It is OK to program for the mode that is already active. See also the [Trajectory Control](#) Section for more information on these modes. If Q is not specified then it will have the same behavior as before and use the value of P -.

G64 P- Example Line

```
G64 P0.015 (set path following to be within 0.015 of the actual path)
```

It is a good idea to include a path control specification in the preamble of each G code file.

16.35 G73 Drilling Cycle with Chip Breaking

```
G73 X- Y- Z- R- Q- <L->
```

- R - retract position along the Z axis.
- Q - delta increment along the Z axis.
- L - repeat

The $G73$ cycle is drilling or milling with chip breaking. This cycle takes a Q number which represents a *delta* increment along the Z axis.

1. Preliminary motion.
 - If the current Z position is below the R position, The Z axis does a *rapid move* to the R position.
 - Move to the X Y coordinates
2. Move the Z-axis only at the current *feed rate* downward by delta or to the Z position, whichever is less deep.
3. Rapid up a bit.
4. Repeat steps 2 and 3 until the Z position is reached at step 2.
5. The Z axis does a rapid move to the R position.

It is an error if:

- the Q number is negative or zero.
- the R number is not specified

16.36 G76 Threading Cycle

G76 P- Z- I- J- R- K- Q- H- E- L-

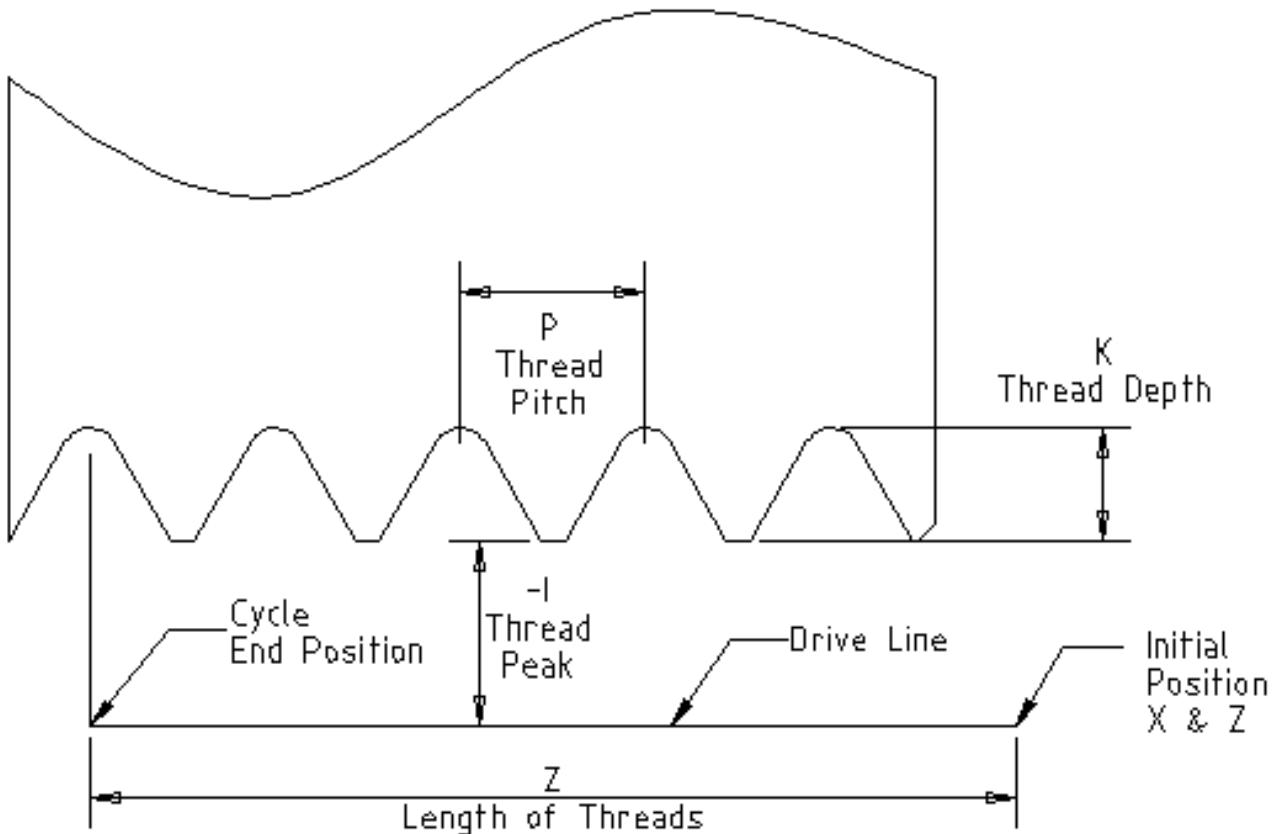


Figure 16.3: G76 Threading

- *Drive Line* - A line through the initial X position parallel to the Z.
- *P-* - The *thread pitch* in distance per revolution.
- *Z-* - The final position of threads. At the end of the cycle the tool will be at this Z position.

Note

When G7 *Lathe Diameter Mode* is in force the values for *I*, *J* and *K* are diameter measurements. When G8 *Lathe Radius Mode* is in force the values for *I*, *J* and *K* are radius measurements.

- *I-* - The *thread peak* offset from the *drive line*. Negative *I* values are external threads, and positive *I* values are internal threads. Generally the material has been turned to this size before the G76 cycle.
- *J-* - A positive value specifying the *initial cut depth*. The first threading cut will be *J* beyond the *thread peak* position.
- *K-* - A positive value specifying the *full thread depth*. The final threading cut will be *K* beyond the *thread peak* position.

Optional settings

- R - - The *depth depression*. $R1.0$ selects constant depth on successive threading passes. $R2.0$ selects constant area. Values between 1.0 and 2.0 select decreasing depth but increasing area. Values above 2.0 select decreasing area. Beware that unnecessarily high depression values will cause a large number of passes to be used. (depression = a descent by stages or steps.)
- Q - - The *compound slide angle* is the angle (in degrees) describing to what extent successive passes should be offset along the drive line. This is used to cause one side of the tool to remove more material than the other. A positive Q value causes the leading edge of the tool to cut more heavily. Typical values are 29, 29.5 or 30.
- H - - The number of *spring passes*. Spring passes are additional passes at full thread depth. If no additional passes are desired, program $H0$.
- E - - Specifies the distance along the drive line used for the taper. The angle of the taper will be so the last pass tapers to the thread crest over the distance specified with E . $E0.2$ will give a taper for the first/last 0.2 length units along the thread. For a 45 degree taper program E the same as K
- L - - Specifies which ends of the thread get the taper. Program $L0$ for no taper (the default), $L1$ for entry taper, $L2$ for exit taper, or $L3$ for both entry and exit tapers. Entry tapers will pause at the drive line to synchronize with the index pulse then move at the [feed rate](#) in to the beginning of the taper. No entry taper and the tool will rapid to the cut depth then synchronize and begin the cut.

The tool is moved to the initial X and Z positions prior to issuing the G76. The X position is the *drive line* and the Z position is the start of the threads.

The tool will pause briefly for synchronization before each threading pass, so a relief groove will be required at the entry unless the beginning of the thread is past the end of the material or an entry taper is used.

Unless using an exit taper, the exit move is not synchronized to the spindle speed and will be a [rapid move](#). With a slow spindle, the exit move might take only a small fraction of a revolution. If the spindle speed is increased after several passes are complete, subsequent exit moves will require a larger portion of a revolution, resulting in a very heavy cut during the exit move. This can be avoided by providing a relief groove at the exit, or by not changing the spindle speed while threading.

The final position of the tool will be at the end of the *drive line*. A safe Z move will be needed with an internal thread to remove the tool from the hole.

It is an error if:

- The active plane is not the ZX plane
- Other axis words, such as X- or Y-, are specified
- The R - depression value is less than 1.0.
- All the required words are not specified
- P -, J -, K - or H - is negative
- E - is greater than half the drive line length

HAL Connections The pins *motion.spindle-at-speed* and the *encoder.n.phase-Z* for the spindle must be connected in your HAL file before G76 will work. See the Integrators Manual for more information on spindle synchronized motion.

Technical Info The G76 canned cycle is based on the G33 Spindle Synchronized Motion. For more information see the G33 [Technical Info](#).

The sample program *g76.ngc* shows the use of the G76 canned cycle, and can be previewed and executed on any machine using the *sim/lathe.ini* configuration.

G76 Example

```
G0 Z-0.5 X0.2
G76 P0.05 Z-1 I-.075 J0.008 K0.045 Q29.5 L2 E0.045
```

In the figure the tool is in the final position after the G76 cycle is completed. You can see the entry path on the right from the Q29.5 and the exit path on the left from the L2 E0.045. The white lines are the cutting moves.

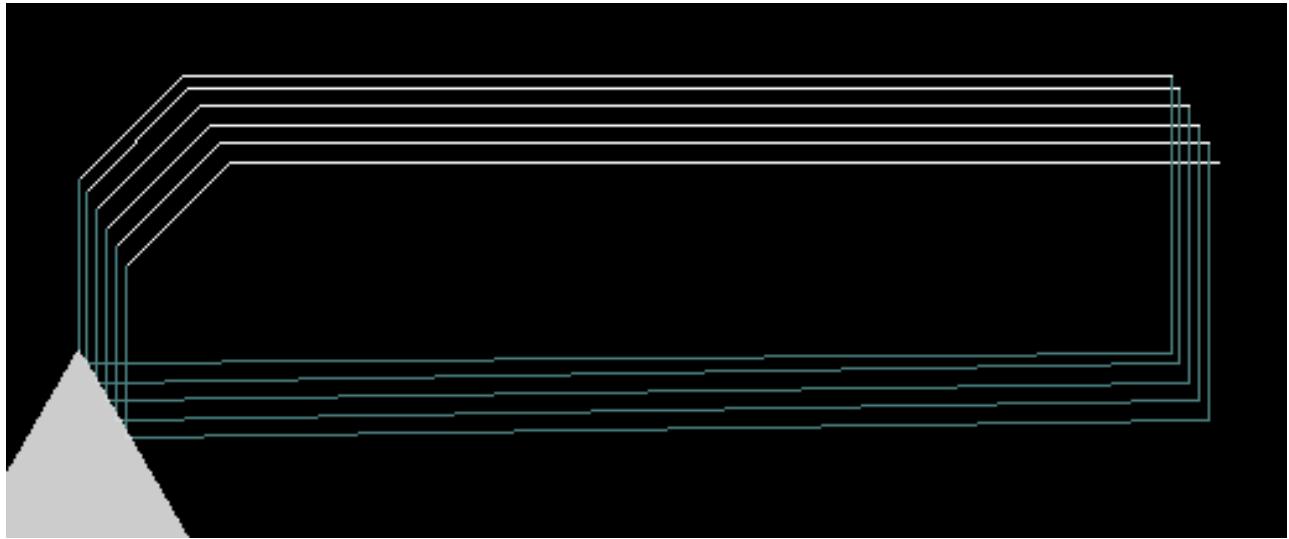


Figure 16.4: G76 Example

16.37 Canned Cycles

The canned cycles *G81* through *G89* and the canned cycle stop *G80* are described in this section.

All canned cycles are performed with respect to the currently-selected plane. Any of the six planes may be selected. Throughout this section, most of the descriptions assume the XY-plane has been selected. The behavior is analogous if another plane is selected, and the correct words must be used. For instance, in the *G17.1* plane, the action of the canned cycle is along W, and the locations or increments are given with U and V. In this case substitute U,V,W for X,Y,Z in the instructions below.

Rotary axis words are not allowed in canned cycles. When the active plane is one of the XYZ family, the UVW axis words are not allowed. Likewise, when the active plane is one of the UVW family, the XYZ axis words are not allowed.

16.37.1 Common Words

All canned cycles use X, Y, Z, or U, V, W groups depending on the plane selected and R words. The R (usually meaning retract) position is along the axis perpendicular to the currently selected plane (Z-axis for XY-plane, etc.) Some canned cycles use additional arguments.

16.37.2 Sticky Words

For canned cycles, we will call a number *sticky* if, when the same cycle is used on several lines of code in a row, the number must be used the first time, but is optional on the rest of the lines. Sticky numbers keep their value on the rest of the lines if they are not explicitly programmed to be different. The R number is always sticky.

In incremental distance mode X, Y, and R numbers are treated as increments from the current position and Z as an increment from the Z-axis position before the move involving Z takes place. In absolute distance mode, the X, Y, R, and Z numbers are absolute positions in the current coordinate system.

16.37.3 Repeat Cycle

The L number is optional and represents the number of repeats. L=0 is not allowed. If the repeat feature is used, it is normally used in incremental distance mode, so that the same sequence of motions is repeated in several equally spaced places along a straight line. When L- is greater than 1 in incremental mode with the XY-plane selected, the X and Y positions are determined by adding the given X and Y numbers either to the current X and Y positions (on the first go-around) or to the X and Y positions at the end of the previous go-around (on the repetitions). Thus, if you program *L10*, you will get 10 cycles. The first cycle will be distance X,Y from the original location. The R and Z positions do not change during the repeats. The L number is not sticky. In absolute distance mode, L>1 means *do the same cycle in the same place several times*. Omitting the L word is equivalent to specifying L=1.

16.37.4 Retract Mode

The height of the retract move at the end of each repeat (called *clear Z* in the descriptions below) is determined by the setting of the retract mode: either to the original Z position (if that is above the R position and the retract mode is *G98, OLD_Z*), or otherwise to the R position. See the [G98 G99 Section](#).

16.37.5 Canned Cycle Errors

It is an error if:

- axis words are all missing during a canned cycle,
- axis words from different groups (XYZ) (UVW) are used together,
- a P number is required and a negative P number is used,
- an L number is used that does not evaluate to a positive integer,
- rotary axis motion is used during a canned cycle,
- inverse time feed rate is active during a canned cycle,
- or cutter compensation is active during a canned cycle.

If the XY plane is active, the Z number is sticky, and it is an error if:

- the Z number is missing and the same canned cycle was not already active,
- or the R number is less than the Z number.

If other planes are active, the error conditions are analogous to the XY conditions above.

16.37.6 Preliminary and In-Between Motion

Preliminary motion is a set of motions that is common to all of the milling canned cycles. If the current Z position is below the R position, the Z axis does a [rapid move](#) to the R position. This happens only once, regardless of the value of L.

In addition, at the beginning of the first cycle and each repeat, the following one or two moves are made

1. A [rapid move](#) parallel to the XY-plane to the given XY-position,
2. The Z-axis make a rapid move to the R position, if it is not already at the R position.

If another plane is active, the preliminary and in-between motions are analogous.

16.37.7 Why use a canned cycle?

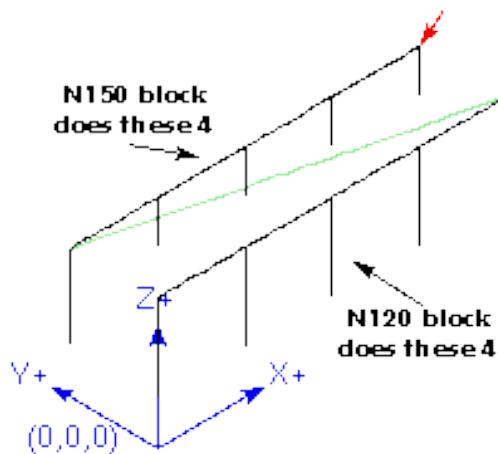
There are at least two reasons for using canned cycles. The first is the economy of code. A single bore would take several lines of code to execute.

The G81 [Example 1](#) demonstrates how a canned cycle could be used to produce 8 holes with ten lines of G code within the canned cycle mode. The program below will produce the same set of 8 holes using five lines for the canned cycle. It does not follow exactly the same path nor does it drill in the same order as the earlier example. But the program writing economy of a good canned cycle should be obvious.

Eight Holes

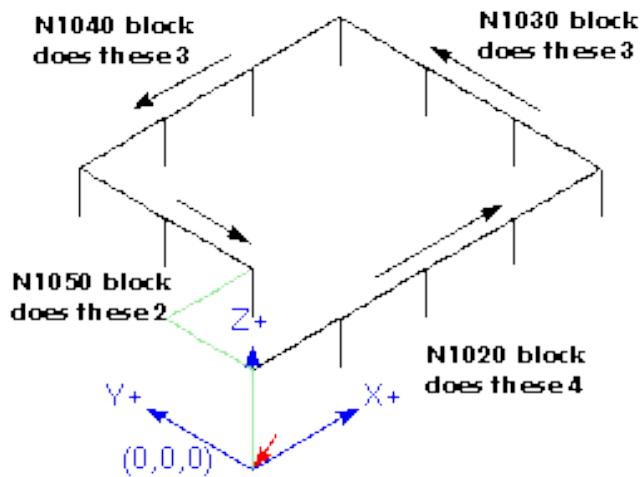
```
G90 G0 X0 Y0 Z0 (move coordinate home)
G1 F10 X0 G4 P0.1
G91 G81 X1 Y0 Z-1 R1 L4(canned drill cycle)
G90 G0 X0 Y1
Z0
G91 G81 X1 Y0 Z-0.5 R1 L4(canned drill cycle)
G80 (turn off canned cycle)
M2 (program end)
```

The G98 to the second line above means that the return move will be to the value of Z in the first line since it is higher than the R value specified.



Twelve Holes in a Square This example demonstrates the use of the L word to repeat a set of incremental drill cycles for successive blocks of code within the same G81 motion mode. Here we produce 12 holes using five lines of code in the canned motion mode.

```
G90 G0 X0 Y0 Z0 (move coordinate home)
G1 F50 X0 G4 P0.1
G91 G81 X1 Y0 Z-0.5 R1 L4 (canned drill cycle)
X0 Y1 R0 L3 (repeat)
X-1 Y0 L3 (repeat)
X0 Y-1 L2 (repeat)
G80 (turn off canned cycle)
G90 G0 X0 (rapid move home)
Y0
Z0
M2 (program end)
```



The second reason to use a canned cycle is that they all produce preliminary moves and returns that you can anticipate and control regardless of the start point of the canned cycle.

16.38 G80 Cancel Canned Cycle

- *G80* - cancel canned cycle modal motion. *G80* is part of modal group 1, so programming any other G code from modal group 1 will also cancel the canned cycle.

It is an error if:

- Axis words are programmed when *G80* is active.

G80 Example

```
G90 G81 X1 Y1 Z1.5 R2.8 (absolute distance canned cycle)
G80 (turn off canned cycle motion)
G0 X0 Y0 Z0 (rapid move to coordinate home)
```

The following code produces the same final position and machine state as the previous code.

G0 Example

```
G90 G81 X1 Y1 Z1.5 R2.8 (absolute distance canned cycle)
G0 X0 Y0 Z0 (rapid move to coordinate home)
```

The advantage of the first set is that, the *G80* line clearly turns off the *G81* canned cycle. With the first set of blocks, the programmer must turn motion back on with *G0*, as is done in the next line, or any other motion mode G word.

If a canned cycle is not turned off with *G80* or another motion word, the canned cycle will attempt to repeat itself using the next block of code that contains an X, Y, or Z word. The following file drills (*G81*) a set of eight holes as shown in the following caption.

G80 Example 1

```
N100 G90 G0 X0 Y0 Z0 (coordinate home)
N110 G1 X0 G4 P0.1
N120 G81 X1 Y0 Z0 R1 (canned drill cycle)
N130 X2
N140 X3
N150 X4
```

```

N160 Y1 Z0.5
N170 X3
N180 X2
N190 X1
N200 G80 (turn off canned cycle)
N210 G0 X0 (rapid move home)
N220 Y0
N230 Z0
N240 M2 (program end)

```

Note

Notice the z position change after the first four holes. Also, this is one of the few places where line numbers have some value, being able to point a reader to a specific line of code.

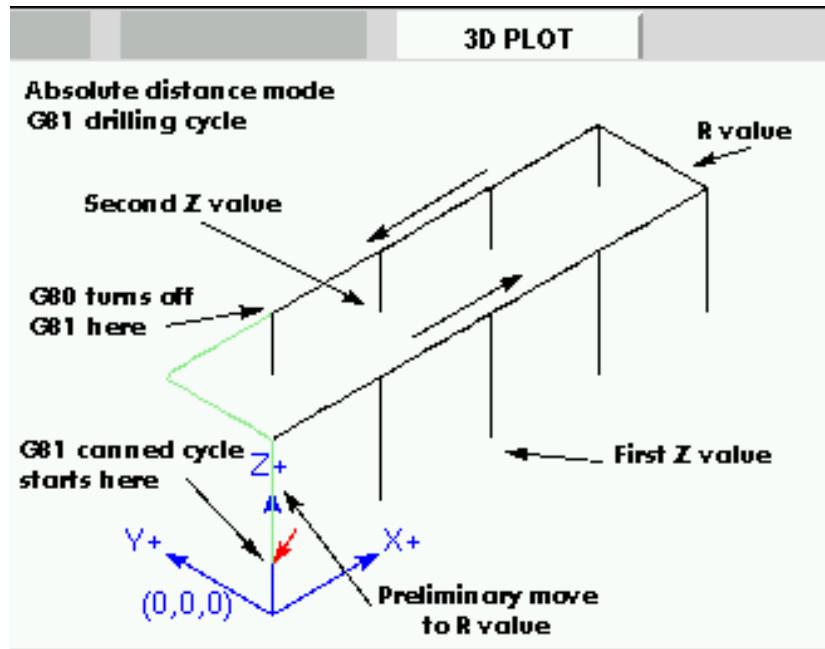


Figure 16.5: G80 Cycle

The use of G80 in line N200 is optional because the G0 on the next line will turn off the G81 cycle. But using the G80 as shown in Example 1, will provide for easier to read canned cycle. Without it, it is not so obvious that all of the blocks between N120 and N200 belong to the canned cycle.

16.39 G81 Drilling Cycle

```
G81 (X- Y- Z-) or (U- V- W-) R- L-
```

The *G81* cycle is intended for drilling.

The cycle functions as follows:

1. Preliminary motion, as described in the [Preliminary and In-Between Motion](#) section.

2. Move the Z-axis at the current **feed rate** to the Z position.
3. The Z-axis does a **rapid move** to clear Z.

Example 1 - Absolute Position G81 Suppose the current position is (X1, Y2, Z3) and the following line of NC code is interpreted.

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

This calls for absolute distance mode (G90) and OLD_Z retract mode (G98) and calls for the G81 drilling cycle to be performed once.

The X value and X position are 4.

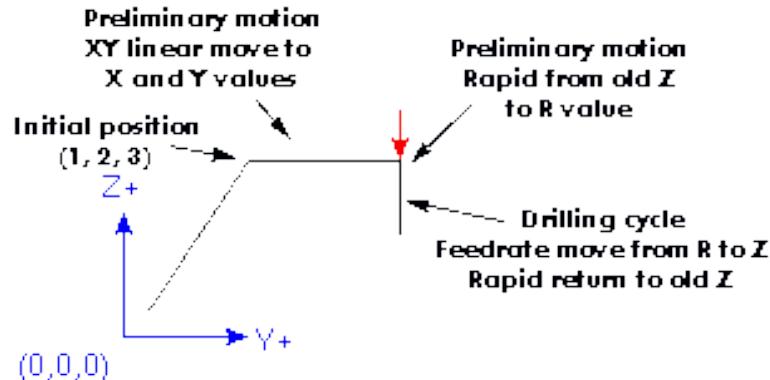
The Y value and Y position are 5.

The Z value and Z position are 1.5.

The R value and clear Z are 2.8. OLD_Z is 3.

The following moves take place:

1. a **rapid move** parallel to the XY plane to (X4, Y5)
2. a rapid move parallel to the Z-axis to (Z2.8).
3. move parallel to the Z-axis at the **feed rate** to (Z1.5)
4. a rapid move parallel to the Z-axis to (Z3)



Example 2 - Relative Position G81 Suppose the current position is (X1, Y2, Z3) and the following line of NC code is interpreted.

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

This calls for incremental distance mode (G91) and OLD_Z retract mode (G98). It also calls for the G81 drilling cycle to be repeated three times. The X value is 4, the Y value is 5, the Z value is -0.6 and the R value is 1.8. The initial X position is 5 (=1+4), the initial Y position is 7 (=2+5), the clear Z position is 4.8 (=1.8+3), and the Z position is 4.2 (=4.8-0.6). OLD_Z is 3.

The first preliminary move is a maximum rapid move along the Z axis to (X1,Y2,Z4.8), since OLD_Z < clear Z.

The first repeat consists of 3 moves.

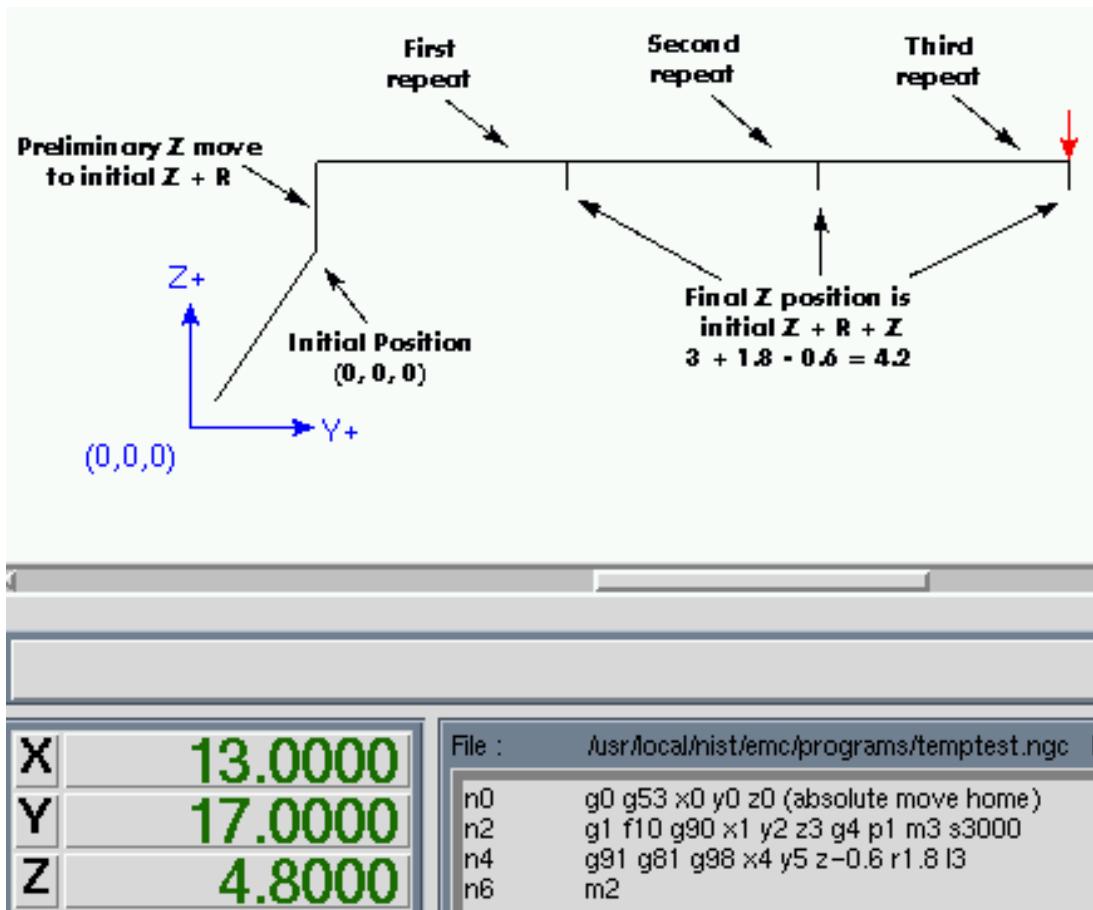
1. a **rapid move** parallel to the XY-plane to (X5, Y7)
2. move parallel to the Z-axis at the **feed rate** to (Z4.2)
3. a rapid move parallel to the Z-axis to (X5, Y7, Z4.8)

The second repeat consists of 3 moves. The X position is reset to 9 (=5+4) and the Y position to 12 (=7+5).

1. a **rapid move** parallel to the XY-plane to (X9, Y12, Z4.8)
2. move parallel to the Z-axis at the feed rate to (X9, Y12, Z4.2)
3. a rapid move parallel to the Z-axis to (X9, Y12, Z4.8)

The third repeat consists of 3 moves. The X position is reset to 13 (=9+4) and the Y position to 17 (=12+5).

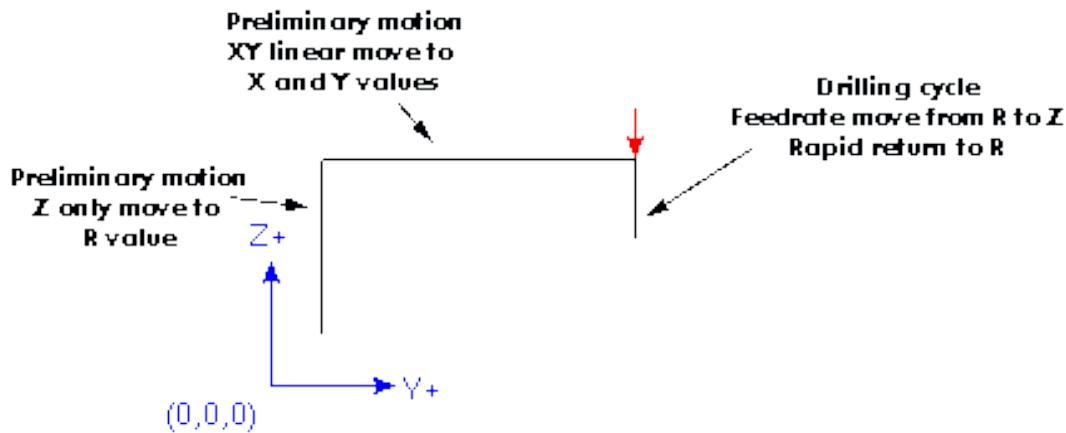
1. a **rapid move** parallel to the XY-plane to (X13, Y17, Z4.8)
2. move parallel to the Z-axis at the feed rate to (X13, Y17, Z4.2)
3. a rapid move parallel to the Z-axis to (X13, Y17, Z4.8)



Example 3 - Relative Position G81 Now suppose that you execute the first G81 block of code but from (X0, Y0, Z0) rather than from (X1, Y2, Z3).

```
G90 G98 G81 X4 Y5 Z1.5 R2.8
```

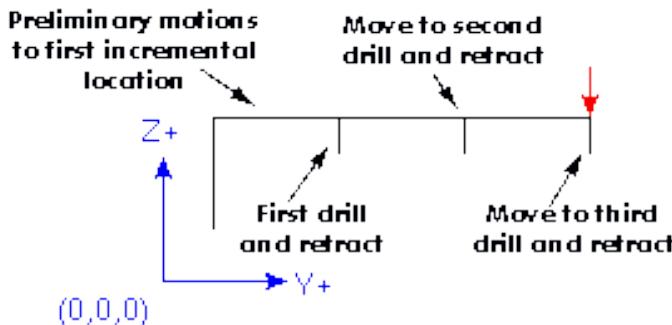
Since OLD_Z is below the R value, it adds nothing for the motion but since the initial value of Z is less than the value specified in R, there will be an initial Z move during the preliminary moves.



Example 4 - Absolute G81 R > Z This is a plot of the path of motion for the second g81 block of code.

```
G91 G98 G81 X4 Y5 Z-0.6 R1.8 L3
```

Since this plot starts with (X0, Y0, Z0), the interpreter adds the initial Z0 and R1.8 and rapid moves to that location. After that initial Z move, the repeat feature works the same as it did in example 3 with the final Z depth being 0.6 below the R value.



Example 5 - Relative position R > Z

```
G90 G98 G81 X4 Y5 Z-0.6 R1.8
```

Since this plot starts with (X0, Y0, Z0), the interpreter adds the initial Z0 and R1.8 and rapid moves to that location as in *Example 4*. After that initial Z move, the **rapid move** to X4 Y5 is done. Then the final Z depth being 0.6 below the R value. The repeat function would make the Z move in the same location again.

16.40 G82 Drilling Cycle, Dwell

```
G82 (X- Y- Z-) or (U- V- W-) R- L- P-
```

The *G82* cycle is intended for drilling with a dwell at the bottom of the hole.

1. Preliminary motion, as described in the [Preliminary and In-Between Motion](#) section.
2. Move the Z-axis at the current [feed rate](#) to the Z position.

3. Dwell for the P number of seconds.
4. The Z-axis does a [rapid move](#) to clear Z.

The motion of a G82 canned cycle looks just like G81 with the addition of a dwell at the bottom of the Z move. The length of the dwell is specified by a *P*- word in the G82 block.

16.41 G83 Peck Drilling Cycle

```
G83 (X- Y- Z-) or (U- V- W-) R- L- Q-
```

The *G83* cycle (often called peck drilling) is intended for deep drilling or milling with chip breaking. The retracts in this cycle clear the hole of chips and cut off any long stringers (which are common when drilling in aluminum). This cycle takes a *Q* number which represents a *delta* increment along the Z-axis. The retract before final depth will always be to the *retract* plane even if G98 is in effect. The final retract will honor the G98/99 in effect. G83 functions the same as G81 with the addition of retracts during the drilling operation.

1. Preliminary motion, as described in the [Preliminary and In-Between Motion](#) section.
2. Move the Z-axis at the current [feed rate](#) downward by delta or to the Z position, whichever is less deep.
3. Rapid move back out to the retract plane specified by the R word.
4. Rapid move back down to the current hole bottom, backed off a bit.
5. Repeat steps 2, 3, and 4 until the Z position is reached at step 2.
6. The Z-axis does a [rapid move](#) to clear Z.

It is an error if:

- the *Q* number is negative or zero.

16.42 G84 Right-Hand Tapping Cycle

This code is currently unimplemented in LinuxCNC. It is accepted, but the behavior is undefined. See section [G33.1](#)

16.43 G85 Boring Cycle, Feed Out

```
G85 (X- Y- Z-) or (U- V- W-) R- L-
```

The *G85* cycle is intended for boring or reaming, but could be used for drilling or milling.

1. Preliminary motion, as described in the [Preliminary and In-Between Motion](#) section.
2. Move the Z-axis only at the current [feed rate](#) to the Z position.
3. Retract the Z-axis at the current feed rate to the R plane if it is lower than the initial Z.
4. Retract at the traverse rate to clear Z.

16.44 G86 Boring Cycle, Spindle Stop, Rapid Move Out

```
G86 (X- Y- Z-) or (U- V- W-) R- L- P-
```

The *G86* cycle is intended for boring. This cycle uses a *P* number for the number of seconds to dwell.

1. Preliminary motion, as described in the [Preliminary and In-Between Motion](#) section.
2. Move the *Z*-axis only at the current [feed rate](#) to the *Z* position.
3. Dwell for the *P* number of seconds.
4. Stop the spindle turning.
5. The *Z*-axis does a [rapid move](#) to clear *Z*.
6. Restart the spindle in the direction it was going.

It is an error if:

- the spindle is not turning before this cycle is executed.

16.45 G87 Back Boring Cycle

This code is currently unimplemented in LinuxCNC. It is accepted, but the behavior is undefined.

16.46 G88 Boring Cycle, Spindle Stop, Manual Out

This code is currently unimplemented in LinuxCNC. It is accepted, but the behavior is undefined.

16.47 G89 Boring Cycle, Dwell, Feed Out

```
G89 (X- Y- Z-) or (U- V- W-) R- L- P-
```

The *G89* cycle is intended for boring. This cycle uses a *P* number, where *P* specifies the number of seconds to dwell.

1. Preliminary motion, as described in the [Preliminary and In-Between Motion](#) section.
2. Move the *Z*-axis only at the current [feed rate](#) to the *Z* position.
3. Dwell for the *P* number of seconds.
4. Retract the *Z*-axis at the current feed rate to clear *Z*.

16.48 G90, G91 Distance Mode

- *G90* - absolute distance mode In absolute distance mode, axis numbers (*X*, *Y*, *Z*, *A*, *B*, *C*, *U*, *V*, *W*) usually represent positions in terms of the currently active coordinate system. Any exceptions to that rule are described explicitly in the [G80 G89](#) Section.
- *G91* - incremental distance mode In incremental distance mode, axis numbers usually represent increments from the current coordinate.

G90 Example

```
G90 (set absolute distance mode)
G0 X2.5 (rapid move to coordinate X2.5 including any offsets in effect)
```

G91 Example

```
G91 (set incremental distance mode)
G0 X2.5 (rapid move 2.5 from current position along the X axis)
```

- See [G0](#) section for more information.

16.49 G90.1, G91.1 Arc Distance Mode

- *G90.1* - absolute distance mode for I, J & K offsets. When G90.1 is in effect I and J both must be specified with G2/3 for the XY plane or J and K for the XZ plane or it is an error.
- *G91.1* - incremental distance mode for I, J & K offsets. G91.1 Returns I, J & K to their default behavior.

16.50 G92 Coordinate System Offset

G92 axes

G92 makes the current point have the coordinates you want (without motion), where the axis words contain the axis numbers you want. All axis words are optional, except that at least one must be used. If an axis word is not used for a given axis, the coordinate on that axis of the current point is not changed.

When *G92* is executed, the origins of all coordinate systems move. They move such that the value of the current controlled point, in the currently active coordinate system, becomes the specified value. All coordinate system's origins are offset this same distance.

For example, suppose the current point is at X=4 and there is currently no *G92* offset active. Then *G92 x7* is programmed. This moves all origins -3 in X, which causes the current point to become X=7. This -3 is saved in parameter 5211.

Being in incremental distance mode has no effect on the action of *G92*.

G92 offsets may be already be in effect when the *G92* is called. If this is the case, the offset is replaced with a new offset that makes the current point become the specified value.

It is an error if:

- all axis words are omitted.

LinuxCNC stores the *G92* offsets and reuses them on the next run of a program. To prevent this, one can program a *G92.1* (to erase them), or program a *G92.2* (to remove them - they are still stored).

See the [Coordinate System](#) Section for an overview of coordinate systems.

See the [Offsets](#) Section for more information.

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

16.51 G92.1, G92.2 Reset Coordinate System Offsets

- *G92.1* - reset axis offsets to zero and set [parameters](#) 5211 - 5219 to zero.
- *G92.2* - reset axis offsets to zero.

16.52 G92.3 Restore Axis Offsets

- *G92.3* - set the axis offset to the values saved in parameters 5211 to 5219

You can set axis offsets in one program and use the same offsets in another program. Program *G92* in the first program. This will set parameters 5211 to 5219. Do not use *G92.1* in the remainder of the first program. The parameter values will be saved when the first program exits and restored when the second one starts up. Use *G92.3* near the beginning of the second program. That will restore the offsets saved in the first program.

16.53 G93, G94, G95: Feed Rate Mode

- *G93* - is Inverse Time Mode. In inverse time feed rate mode, an F word means the move should be completed in [one divided by the F number] minutes. For example, if the F number is 2.0, the move should be completed in half a minute. When the inverse time feed rate mode is active, an F word must appear on every line which has a G1, G2, or G3 motion, and an F word on a line that does not have G1, G2, or G3 is ignored. Being in inverse time feed rate mode does not affect G0 ([rapid move](#)) motions.
- *G94* - is Units per Minute Mode. In units per minute feed mode, an F word is interpreted to mean the controlled point should move at a certain number of inches per minute, millimeters per minute, or degrees per minute, depending upon what length units are being used and which axis or axes are moving.
- *G95* - is Units per Revolution Mode In units per revolution mode, an F word is interpreted to mean the controlled point should move a certain number of inches per revolution of the spindle, depending on what length units are being used and which axis or axes are moving. G95 is not suitable for threading, for threading use G33 or G76. G95 requires that motion.spindle-speed-in to be connected.

It is an error if:

- Inverse time feed mode is active and a line with G1, G2, or G3 (explicitly or implicitly) does not have an F word.
- A new feed rate is not specified after switching to G94 or G95

16.54 G96, G97 Spindle Control Mode

```
G96 <D-> S- (Constant Surface Speed)
G97 (RPM Mode)
```

- *D* - maximum spindle RPM
- *S* - surface speed
- *G96 D- S-* - selects constant surface speed of *S* feet per minute (if G20 is in effect) or meters per minute (if G21 is in effect). *D*- is optional.
When using G96, ensure that X0 in the current coordinate system (including offsets and tool lengths) is the center of rotation or LinuxCNC will not give the desired spindle speed. G96 is not affected by radius or diameter mode.
- *G97* selects RPM mode.

G96 Example Line

```
G96 D2500 S250 (set CSS with a max rpm of 2500 and a surface speed of 250)
```

It is an error if:

- *S* is not specified with G96
- A feed move is specified in G96 mode while the spindle is not turning

16.55 G98, G99 Canned Cycle Return Level

- *G98* - retract to the position that axis was in just before this series of one or more contiguous canned cycles was started.
- *G99* - retract to the position specified by the R word of the canned cycle.

Program a *G98* and the canned cycle will use the Z position prior to the canned cycle as the Z return position if it is higher than the R value specified in the cycle. If it is lower, the R value will be used. The R word has different meanings in absolute distance mode and incremental distance mode.

G98 Retract to Origin

```
G0 X1 Y2 Z3  
G90 G98 G81 X4 Y5 Z-0.6 R1.8 F10
```

The *G98* to the second line above means that the return move will be to the value of Z in the first line since it is higher than the R value specified.

The *initial* (*G98*) plane is reset any time cycle motion mode is abandoned, whether explicitly (*G80*) or implicitly (any motion code that is not a cycle). Switching among cycle modes (say *G81* to *G83*) does NOT reset the *initial* plane. It is possible to switch between *G98* and *G99* during a series of cycles.

Chapter 17

M Codes

17.1 M Code Quick Reference Table

Code	Description
M0 M1	Program Pause
M2 M30	Program End
M60	Pallet Change Pause
M3 M4 M5	Spindle Control
M6	Tool Change
M7 M8 M9	Coolant Control
M19	Orient Spindle
M48 M49	Feed & Spindle Overrides Enable/Disable
M50	Feed Override Control
M51	Spindle Override Control
M52	Adaptive Feed Control
M53	Feed Stop Control
M61	Set Current Tool Number
M62-M65	Output Control
M66	Input Control
M67	Analog Output Control
M68	Analog Output Control
M70	Save Modal State
M71	Invalidate Stored Modal State
M72	Restore Modal State
M73	Save Autorestore Modal State
M100-M199	User Defined M-Codes

17.2 M0, M1 Program Pause

- *M0* - pause a running program temporarily. LinuxCNC remains in the Auto Mode so MDI and other manual actions are not enabled. Pressing the cycle start button will restart the program at the following line.
- *M1* - pause a running program temporarily if the optional stop switch is on. LinuxCNC remains in the Auto Mode so MDI and other manual actions are not enabled. Pressing the cycle start button will restart the program at the following line.

Note

It is OK to program *M0* and *M1* in MDI mode, but the effect will probably not be noticeable, because normal behavior in MDI mode is to stop after each line of input anyway.

17.3 M2, M30 Program End

- *M2* - end the program. Pressing cycle start will start the program at the beginning of the file.
- *M30* - exchange pallet shuttles and end the program. Pressing cycle start will start the program at the beginning of the file.

Both of these commands have the following effects:

1. Change from Auto mode to MDI mode.
2. Origin offsets are set to the default (like *G54*).
3. Selected plane is set to XY plane (like *G17*).
4. Distance mode is set to absolute mode (like *G90*).
5. Feed rate mode is set to units per minute (like *G94*).
6. Feed and speed overrides are set to ON (like *M48*).
7. Cutter compensation is turned off (like *G40*).
8. The spindle is stopped (like *M5*).
9. The current motion mode is set to feed (like *G1*).
10. Coolant is turned off (like *M9*).

Note

Lines of code after M2/M30 will not be executed. Pressing cycle start will start the program at the beginning of the file.

17.4 M60 Pallet Change Pause

- *M60* - exchange pallet shuttles and then pause a running program temporarily (regardless of the setting of the optional stop switch). Pressing the cycle start button will restart the program at the following line.

17.5 M3, M4, M5 Spindle Control

- *M3* - start the spindle clockwise at the *S* speed.
- *M4* - start the spindle counterclockwise at the *S* speed.
- *M5* - stop the spindle.

It is OK to use *M3* or *M4* if the *S* spindle speed is set to zero. If this is done (or if the speed override switch is enabled and set to zero), the spindle will not start turning. If, later, the spindle speed is set above zero (or the override switch is turned up), the spindle will start turning. It is OK to use *M3* or *M4* when the spindle is already turning or to use *M5* when the spindle is already stopped.

17.6 M6 Tool Change

17.6.1 Manual Tool Change

If the HAL component `hal_manualtoolchange` is loaded, M6 will stop the spindle and prompt the user to change the tool based on the last *T*-number programmed. For more information on `hal_manualtoolchange` see the ([Manual Tool Change](#)) Section.

17.6.2 Tool Changer

To change a tool in the spindle from the tool currently in the spindle to the tool most recently selected (using a T word - see Section [Select Tool](#)), program *M6*. When the tool change is complete:

- The spindle will be stopped.
- The tool that was selected (by a T word on the same line or on any line after the previous tool change) will be in the spindle.
- If the selected tool was not in the spindle before the tool change, the tool that was in the spindle (if there was one) will be placed back into the tool changer magazine.
- If configured in the .ini file some axis positions may move when a *M6* is issued. See the EMCIO section of the Integrator's Manual for more information on tool change options.
- No other changes will be made. For example, coolant will continue to flow during the tool change unless it has been turned off by an *M9*.



Warning

The tool length offset is not changed by *M6*, use *G43* after the *M6* to change the tool length offset.

The tool change may include axis motion. It is OK (but not useful) to program a change to the tool already in the spindle. It is OK if there is no tool in the selected slot; in that case, the spindle will be empty after the tool change. If slot zero was last selected, there will definitely be no tool in the spindle after a tool change. The tool changer will have to be setup to perform the tool change in hal and possibly classcladder.

17.7 M7, M8, M9 Coolant Control

- *M7* - turn mist coolant on.
- *M8* - turn flood coolant on.
- *M9* - turn all coolant off.

It is OK to use any of these commands, regardless of the current coolant state.

17.8 M19 Orient Spindle

- *M19 R- [P-]*
- *R* Position to rotate to from 0, valid range is 0-360 degrees
- *Q* Number of seconds to wait until orient completes. If motion.spindle.is_oriented does not become true within Q timeout an error occurs.
- *P* Direction to rotate to position.
 - 0 rotate for smallest angular movement (default)
 - 1 always rotate clockwise (same as M3 direction)
 - 2 always rotate counterclockwise (same as M4 direction)

M19 is cleared by any of M3,M4,M5.

Spindle orientation requires a differential encoder with an index to sense the spindle shaft position and direction of rotation.

INI Settings in the [RS274NGC] section.

ORIENT_OFFSET = 0-360 (fixed offset in degrees added to M19 R word)

HAL Pins

- *motion.spindle-orient-angle* (out float) Desired spindle orientation for M19. Value of the M19 R word parameter plus the value of the [RS274NGC]ORIENT_OFFSET ini parameter.
- *motion.spindle-orient-mode* (out s32) Desired spindle rotation mode. Reflects M19 P parameter word, Default = 0
- *motion.spindle-orient* (out bit) Indicates start of spindle orient cycle. Set by M19. Cleared by any of M3,M4,M5. If spindle-orient-fault is not zero during spindle-orient true, the M19 command fails with an error message.
- *motion.spindle-is-oriented* (in bit) Acknowledge pin for spindle-orient. Completes orient cycle. If spindle-orient was true when spindle-is-oriented was asserted, the spindle-orient pin is cleared and the spindle-locked pin is asserted. Also, the spindle-brake pin is asserted.
- *motion.spindle-orient-fault* (in s32) Fault code input for orient cycle. Any value other than zero will cause the orient cycle to abort.
- *motion.spindle-locked* (out bit) Spindle orient complete pin. Cleared by any of M3,M4,M5.

17.9 M48, M49 Speed and Feed Override Control

- *M48* - enable the spindle speed and feed rate override controls.
- *M49* - disable both controls.

It is OK to enable or disable the controls when they are already enabled or disabled. See the [Feed Rate Section](#) for more details.

17.10 M50 Feed Override Control

- *M50 <P1>* - enable the feed rate override control. The P1 is optional.
- *M50 P0* - disable the feed rate control.

While disabled the feed override will have no influence, and the motion will be executed at programmed feed rate. (unless there is an adaptive feed rate override active).

17.11 M51 Spindle Speed Override Control

- *M51 <P1>* - enable the spindle speed override control. The P1 is optional.
- *M51 P0* - disable the spindle speed override control program. While disabled the spindle speed override will have no influence, and the spindle speed will have the exact program specified value of the S-word (described in [Spindle Speed Section](#)).

17.12 M52 Adaptive Feed Control

- *M52 <P1>* - use an adaptive feed. The P1 is optional.
- *M52 P0* - stop using adaptive feed.

When adaptive feed is enabled, some external input value is used together with the user interface feed override value and the commanded feed rate to set the actual feed rate. In LinuxCNC, the HAL pin *motion.adaptive-feed* is used for this purpose. Values on *motion.adaptive-feed* should range from 0 (feed hold) to 1 (full speed).

17.13 M53 Feed Stop Control

- *M53 <P1>* - enable the feed stop switch. The P1 is optional. Enabling the feed stop switch will allow motion to be interrupted by means of the feed stop control. In LinuxCNC, the HAL pin *motion.feed-hold* is used for this purpose. A *true* value will cause the motion to stop when *M53* is active.
- *M53 P0* - disable the feed stop switch. The state of *motion.feed-hold* will have no effect on feed when *M53* is not active.

17.14 M61 Set Current Tool Number

- *M61 Q-* - change the current tool number while in MDI or Manual mode. One use is when you power up LinuxCNC with a tool currently in the spindle you can set that tool number without doing a tool change.

It is an error if:

- Q- is not 0 or greater

17.15 M62 to M65 Output Control

- *M62 P-* - turn on digital output synchronized with motion. The P- word specifies the digital output number.
- *M63 P-* - turn off digital output synchronized with motion. The P- word specifies the digital output number.
- *M64 P-* - turn on digital output immediately. The P- word specifies the digital output number.
- *M65 P-* - turn off digital output immediately. The P- word specifies the digital output number.

The P-word ranges from 0 to a default value of 3. If needed the the number of I/O can be increased by using the *num_dio* parameter when loading the motion controller. See the Integrator's Manual Configuration Section LinuxCNC and HAL section for more information.

The M62 & M63 commands will be queued. Subsequent commands referring to the same output number will overwrite the older settings. More than one output change can be specified by issuing more than one M62/M63 command.

The actual change of the specified outputs will happen at the beginning of the next motion command. If there is no subsequent motion command, the queued output changes won't happen. It's best to always program a motion G code (G0, G1, etc) right after the M62/63.

M64 & M65 happen immediately as they are received by the motion controller. They are not synchronized with movement, and they will break blending.

Note

M62-65 will not function unless the appropriate *motion.digital-out-nn* pins are connected in your hal file to outputs.

17.16 M66 Wait on Input

```
M66 P- | E- <L->
```

- *P-* - specifies the digital input number from 0 to 3.
- *E-* - specifies the analog input number from 0 to 3.
- *L-* - specifies the wait mode type.
 - *Mode 0: IMMEDIATE* - no waiting, returns immediately. The current value of the input is stored in parameter #5399
 - *Mode 1: RISE* - waits for the selected input to perform a rise event.
 - *Mode 2: FALL* - waits for the selected input to perform a fall event.
 - *Mode 3: HIGH* - waits for the selected input to go to the HIGH state.
 - *Mode 4: LOW* - waits for the selected input to go to the LOW state.
- *Q-* - specifies the timeout in seconds for waiting. If the timeout is exceeded, the wait is interrupt, and the variable #5399 will be holding the value -1. The Q value is ignored if the L-word is zero (IMMEDIATE). A Q value of zero is an error if the L-word is non-zero.
- Mode 0 is the only one permitted for an analog input.

M66 Example Lines

```
M66 P0 L3 Q5 (wait up to 5 seconds for digital input 0 to turn on)
```

M66 wait on an input stops further execution of the program, until the selected event (or the programmed timeout) occurs.

It is an error to program M66 with both a P-word and an E-word (thus selecting both an analog and a digital input). In LinuxCNC these inputs are not monitored in real time and thus should not be used for timing-critical applications.

The number of I/O can be increased by using the num_dio or num_aio parameter when loading the motion controller. See the Integrator's Manual, Core Components Section, Motion subsection, for more information.

Note

M66 will not function unless the appropriate motion.digital-in-nn pins or motion.analog-in-nn pins are connected in your hal file to an input.

Example HAL Connection

```
net signal-name motion.digital-in-00 <= parport.0.pin10-in
```

17.17 M67 Synchronized Analog Output

```
M67 E- Q-
```

- *M67* - set an analog output synchronized with motion.
- *E-* - output number ranging from 0 to 3.
- *Q-* - is the value to set (set to 0 to turn off).

The actual change of the specified outputs will happen at the beginning of the next motion command. If there is no subsequent motion command, the queued output changes won't happen. It's best to always program a motion G code (G0, G1, etc) right after the M67. M67 functions the same as M62-63.

The number of I/O can be increased by using the num_dio or num_aio parameter when loading the motion controller. See the Integrator's Manual, Core Components Section, Motion subsection, for more information.

Note

M67 will not function unless the appropriate motion.analog-out-nn pins are connected in your hal file to outputs.

17.18 M68 Analog Output

M68 E- Q-

- *M68* - set an analog output immediately.
- *E* - output number ranging from 0 to 3.
- *Q* - is the value to set (set to 0 to turn off).

M68 output happen immediately as they are received by the motion controller. They are not synchronized with movement, and they will break blending. M68 functions the same as M64-65.

The number of I/O can be increased by using the num_dio or num_aio parameter when loading the motion controller. See the Integrator's Manual, Core Components Section, Motion subsection, for more information.

Note

M68 will not function unless the appropriate motion.analog-out-nn pins are connected in your hal file to outputs.

17.19 M70 Save Modal State

To explicitly save the modal state at the current call level, program *M70*. Once modal state has been saved with *M70*, it can be restored to exactly that state by executing an *M72*.

A pair of *M70* and *M72* instructions will typically be used to protect a program against inadvertant modal changes within subroutines.

The state saved consists of:

- current G20/G21 settings (imperial/metric)
- selected plane (G17/G18/G19 G17.1,G18.1,G19.1)
- status of cutter compensation (G40,G41,G42,G41.1,G42,1)
- distance mode - relative/absolute (G90/G91)
- feed mode (G93/G94,G95)
- current coordinate system (G54-G59.3)
- tool length compensation status (G43,G43.1,G49)
- retract mode (G98,G99)

- spindle mode (G96-css or G97-RPM)
- arc distance mode (G90.1, G91.1)
- lathe radius/diameter mode (G7,G8)
- path control mode (G61, G61.1, G64)
- current feed and speed (*F* and *S* values)
- spindle status (M3,M4,M5) - on/off and direction
- mist (M7) and flood (M8) status
- speed override (M51) and feed override (M50) settings
- adaptive feed setting (M52)
- feed hold setting (M53)

Note that in particular, the motion mode (G1 etc) is NOT restored.

current call level means either:

- executing in the main program. There is a single storage location for state at the main program level; if several *M70* instructions are executed in turn, only the most recently saved state is restored when an *M72* is executed.
- executing within a G-code subroutine. The state saved with *M70* within a subroutine behaves exactly like a local named parameter - it can be referred to only within this subroutine invocation with an *M72* and when the subroutine exits, the parameter goes away.

A recursive invocation of a subroutine introduces a new call level.

17.20 M71 Invalidate Stored Modal State

Modal state saved with an *M70* or by an *M73* at the current call level is invalidated (cannot be restored from anymore).

A subsequent *M72* at the same call level will fail.

If executed in a subroutine which protects modal state by an *M73*, a subsequent return or *endsub* will **not** restore modal state.

The usefulness of this feature is dubious. It should not be relied upon as it might go away.

17.21 M72 Restore Modal State

Modal state saved with an *M70* code can be restored by executing an *M72*.

The handling of G20/G21 is specially treated as feeds are interpreted differently depending on G20/G21: if length units (mm/in) are about to be changed by the restore operation, 'M72' will restore the distance mode first, and then all other state including feed to make sure the feed value is interpreted in the correct unit setting.

It is an error to execute an *M72* with no previous *M70* save operation at that level.

The following example demonstrates saving and explicitly restoring modal state around a subroutine call using *M70* and *M72*. Note that the *imperialsub* subroutine is not "aware" of the M7x features and can be used unmodified:

```

O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsing> sub
g20 (imperial)
g91 (relative mode)
F5 (low feed)
S300 (low rpm)
(debug, in subroutine, state now:)
o<showstate> call
O<imperialsing> endsub

; main program
g21 (metric)
g90 (absolute)
f200 (fast speed)
S2500 (high rpm)

(debug, in main, state now:)
o<showstate> call

M70 (save caller state in at global level)
O<imperialsing> call
M72 (explicitly restore state)

(debug, back in main, state now:)
o<showstate> call
m2

```

17.22 M73 Save and Autorestore Modal State

To save modal state within a subroutine, and restore state on subroutine *endsub* or any *return* path, program *M73*.

Aborting a running program in a subroutine which has an *M73* operation will **not** restore state .

Also, the normal end (*M2*) of a main program which contains an *M73* will **not** restore state.

The suggested use is at the beginning of a O-word subroutine as in the following example. Using *M73* this way enables designing subroutines which need to modify modal state but will protect the calling program against inadvertent modal changes. Note the use of [predefined named parameters](#) in the *showstate* subroutine.

```

O<showstate> sub
(DEBUG, imperial=#<_imperial> absolute=#<_absolute> feed=#<_feed> rpm=#<_rpm>)
O<showstate> endsub

O<imperialsing> sub
M73 (save caller state in current call context, restore on return or endsub)
g20 (imperial)
g91 (relative mode)
F5 (low feed)
S300 (low rpm)
(debug, in subroutine, state now:)
o<showstate> call

; note - no M72 is needed here - the following endsub or an
; explicit 'return' will restore caller state
O<imperialsing> endsub

; main program

```

```

g21 (metric)
g90 (absolute)
f200 (fast speed)
S2500 (high rpm)
(debug, in main, state now:)
o<showstate> call
o<imperialspeed> call
(debug, back in main, state now:)
o<showstate> call
m2

```

17.22.1 Selectively restoring modal state by testing predefined parameters

Executing an *M72* or returning from a subroutine which contains an *M73* will restore [all modal state saved](#).

If only some aspects of modal state should be preserved, an alternative is the usage of [predefined named parameters](#), local parameters and conditional statements. The idea is to remember the modes to be restored at the beginning of the subroutine, and restore these before exiting. Here is an example, based on snippet of *nc_files/tool-length-probe.ngc*:

```

O<measure> sub      (measure reference tool)
;
#<absolute> = #<_absolute>  (remember in local variable if G90 was set)
;
g30 (above switch)
g38.2 z0 f15 (measure)
g91 g0z.2 (off the switch)
#1000=#5063 (save reference tool length)
#print,reference length is #1000
;
O<restore_abs> if [#<absolute>]
    g90 (restore G90 only if it was set on entry:)
O<restore_abs> endif
;
O<measure> endsub

```

17.23 M100 to M199 User Defined Commands

M1-- <P- Q->

- *M1--* - an integer in the range of 100 - 199.
- *P-* - a number passed to the file as the first parameter.
- *Q-* - a number passed to the file as the second parameter.

Note

After creating a new *M1nn* file you must restart the GUI so it is aware of the new file, otherwise you will get an *Unknown m code* error.

The external program named *M100* through *M199* (no extension and a capital M) is executed with the optional P and Q values as its two arguments. Execution of the G code file pauses until the external program exits. Any valid executable file can be used. The file must be located in the search path specified in the ini file configuration. See the ini config section of the Integrators Manual for more information on search paths.

**Warning**

Do not use a word processor to create or edit the files. A word processor will leave unseen codes that will cause problems and may prevent a bash or python file from working. Use a text editor like Gedit in Ubuntu or Notepad++ in other operating systems to create or edit the files.

The error *Unknown M code used* denotes one of the following

- The specified User Defined Command does not exist
- The file is not an executable file
- The file name has an extension
- The file name does not follow this format M1nn where nn = 00 through 99
- The file name used a lower case M

For example to open and close a collet closer that is controlled by a parallel port pin using a bash script file using M101 and M102. Create two files named M101 and M102. Set them as executable files (typically right click/properties/permissions) before running LinuxCNC. Make sure the parallel port pin is not connected to anything in a HAL file.

M101 Example File

```
#!/bin/bash
# file to turn on parport pin 14 to open the collet closer
halcmd setp parport.0.pin-14-out True
exit 0
```

M102 Example File

```
#!/bin/bash
# file to turn off parport pin 14 to open the collet closer
halcmd setp parport.0.pin-14-out False
exit 0
```

To pass a variable to a M1nn file you use the P and Q option like this:

```
M100 P123.456 Q321.654
```

M100 Example file

```
#!/bin/bash
voltage=$1
feedrate=$2
halcmd setp thc.voltage $voltage
halcmd setp thc.feedrate $feedrate
exit 0
```

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 18

O Codes

O-codes provide for flow control in NC programs. Each block has an associated number, which is the number used after O. Care must be taken to properly match the O-numbers. O codes use the letter *O* not the number zero as the first character in the number like O100.

Numbering Example

```
o100 sub
(notice that the if-endif block uses a different number)
    o110 if [#2 GT 5]
        (some code here)
    o110 endif
        (some more code here)
o100 endsub
```

The behavior is undefined if:

- The same number is used for more than one block
- Other words are used on a line with an O- word
- Comments are used on a line with an O-word

Note

Using the lower case o makes it easier to distinguish from a 0 that might have been mistyped. For example o100 is easier to see than O100 that it is not a 0.

The following statements cause an error message and abort the interpreter:

- a return or endsub not within a sub defintion
- a label on repeat which is defined elsewhere
- a label on while which is defined elsewhere and not referring to a do
- a label on if defined elsewhere
- a undefined label on else or elseif
- a label on else, elseif or endif not pointing to a matching if
- a label on break or continue which does not point to a matching while or do
- a label on endrepeat or endwhile no referring to a corresponding while or repeat

To make these errors non-fatal warnings on stderr, set bit 0x20 in the [RS274NGC]FEATURE= mask ini option.

18.1 Subroutines

Subroutines extend from a *O-sub* to an *O-endsub*. The lines between *O-sub* and *O-endsub* are not executed until the subroutine is called with *O-call*.

Subroutine Example

```

o100 sub
  G53 G0 X0 Y0 Z0 (rapid move to machine home)
o100 endsub
...
o100 call (call the subroutine here)
M2

```

See [G53](#) & [G0](#) & [M2](#) sections for more information.

O-Return Inside a subroutine, *O-return* can be executed. This immediately returns to the calling code, just as though *O-endsub* was encountered.

O-Return Example

```

o100 sub
  o110 if [#2 GT 5] (test if parameter #2 is greater than 5)
    o100 return (return to top of subroutine if test is true)
  o110 endif
  (some code here that only gets executed if parameter #2 is less than 5)
o100 endsub

```

See the [Binary Operators & Parameters](#) sections for more information.

O-Call *O-Call* takes up to 30 optional arguments, which are passed to the subroutine as #1, #2, ..., #N. Parameters from #N+1 to #30 have the same value as in the calling context. On return from the subroutine, the values of parameters #1 through #30 (regardless of the number of arguments) will be restored to the values they had before the call. Parameters #1 - #30 are local to the subroutine.

Because *I 2 3* is parsed as the number 123, the parameters must be enclosed in square brackets. The following calls a subroutine with 3 arguments:

O-Call Example

```

o200 call [1] [2] [3]

```

Subroutine bodies may not be nested. They may only be called after they are defined. They may be called from other functions, and may call themselves recursively if it makes sense to do so. The maximum subroutine nesting level is 10.

Subroutines do not have *return values*, but they may change the value of parameters above #30 and those changes will be visible to the calling code. Subroutines may also change the value of global named parameters.

18.2 Looping

The *while* loop has two structures: *while/endwhile*, and *do/while*. In each case, the loop is exited when the *while* condition evaluates to false. The difference is when the test condition is done. The *do/while* loop runs the code in the loop then checks the test condition. The *while/endwhile* loop does the test first.

While Endwhile Example

```

(draw a sawtooth shape)
G0 X1 Y0 (move to start position)
#1 = 1 (assign parameter #1 the value of 0)
F25 (set a feed rate)
o101 while [#1 LT 10]

```

```

G1 X0
G1 Y[#1/10] X1
#1 = [#1+1] (increment the test counter)
o101 endwhile
M2 (end program)

```

Do While Example

```

#1 = 0 (assign parameter #1 the value of 0)
o100 do
  (debug, parameter 1 = #1)
  o110 if [#1 EQ 2]
    #1 = 3 (assign the value of 3 to parameter #1)
    (msg, #1 has been assigned the value of 3)
    o100 continue (skip to start of loop)
  o110 endif
  (some code here)
  #1 = [#1 + 1] (increment the test counter)
o100 while [#1 LT 3]
  (msg, Loop Done!)
M2

```

Inside a while loop, *O-break* immediately exits the loop, and *O-continue* immediately skips to the next evaluation of the *while* condition. If it is still true, the loop begins again at the top. If it is false, it exits the loop.

18.3 Conditional

The *if* conditional consists of a group of statements with the same *o* number that start with *if* and end with *endif*. Optional *elseif* and *else* conditions may be between the starting *if* and the ending *endif*.

If the *if* conditional evaluates to true then the group of statements following the *if* up to the next conditional line are executed.

If the *if* conditional evaluates to false then the *elseif* conditions are evaluated in order until one evaluates to true. If the *elseif* condition is true then the statements following the *elseif* up to the next conditional line are executed. If none of the *if* or *elseif* conditions evaluate to true then the statements following the *else* are executed. When a condition is evaluated to true no more conditions are evaluated in the group.

If Endif Example

```

o101 if [#31 EQ 3] (if parameter #31 is equal to 3 set S2000)
  S2000
o101 endif

```

If ElseIf Else EndIf Example

```

o102 if [#2 GT 5] (if parameter #2 is greater than 5 set F100)
  F100
o102 elseif [#2 LT 2] (else if parameter #2 is less than 2 set F200)
  F200
o102 else (else if parameter #2 is 2 through 5 set F150)
  F150
o102 endif

```

Several conditons may be tested for by *elseif* statements until the *else* path is finally executed if all preceding conditons are false:

If Elseif Else Endif Example

```

o102 if [#2 GT 5] (if parameter #2 is greater than 5 set F100)
  F100
o102 elseif [#2 LT 2] (else if parameter #2 less than 2 set F200)
  F20

```

```
o102 else (parameter #2 between 2 and 5)
  F200
o102 endif
```

18.4 Repeat

The *repeat* will execute the statements inside of the repeat/endrepeat the specified number of times. The example shows how you might mill a diagonal series of shapes starting at the present position.

Repeat Example

```
(Mill 5 diagonal shapes)
G91 (Incremental mode)
o103 repeat [5]
... (insert milling code here)
G0 X1 Y1 (diagonal move to next position)
o103 endrepeat
G90 (Absolute mode)
```

18.5 Indirection

The O-number may be given by a parameter and/or calculation.

Indirection Example

```
o[#101+2] call
```

Computing values in O-words For more information on computing values see the following sections

- [Parameters](#)
- [Expressions](#)
- [Binary Operators](#)
- [Functions](#)

18.6 Calling Files

To call a separate file with a subroutine name the file the same as your call and include a sub and endsub in the file. The file must be in the directory pointed to by *PROGRAM_PREFIX* or *SUBROUTINE_PATH* in the ini file. The file name can include **lowercase** letters, numbers, dash, and underscore only. A named subroutine file can contain only a single subroutine definition.

Named File Example

```
o<myfile> call
```

Numbered File Example

```
o123 call
```

In the called file you must include the oxxx sub and endsub and the file must be a valid file.

Called File Example

```
(filename myfile.ngc)
o<myfile> sub
  (code here)
o<myfile> endsub
M2
```

Note

The file names are lowercase letters only so *o<MyFile>* is converted to *o<myfile>* by the interpreter. More information about the search path and options for the search path are in the INI Configuration Section.

18.7 Subroutine return values

Subroutines may optionally return a value by an optional expression at an *endsub* or *return* statement.

Return value example

```
o123 return [#2 *5]
...
o123 endsub [3 * 4]
```

A subroutine return value is stored in the *<_value> predefined named parameter*, and the *<_value_returned> predefined parameter* is set to 1, to indicate a value was returned. Both parameters are global, and are cleared just before the next subroutine call.

Chapter 19

Other Codes

19.1 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.

19.2 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%.

19.3 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 20

G Code Examples

After you install LinuxCNC several sample files are placed in the /nc_files folder. Make sure the sample file is appropriate for your machine before running.

20.1 Mill Examples

20.1.1 Helical Hole Milling

File Name: useful-subroutines.ngc

Description: Subroutine for milling a hole using parameters.

20.1.2 Slotting

File Name: useful-subroutines.ngc

Description: Subroutine for milling a slot using parameters.

20.1.3 Grid Probe

File Name: gridprobe.ngc

Description: Rectangular Probing

This program repeatedly probes in a regular XY grid and writes the probed location to the file *probe-results.txt* in the same directory as the .ini file.

20.1.4 Smart Probe

File Name: smartprobe.ngc

Description: Rectangular Probing

This program repeatedly probes in a regular XY grid and writes the probed location to the file *probe-results.txt* in the same directory as the .ini file. This is improved from the grid probe file.

20.1.5 Tool Length Probe

File Name: tool-length-probe.ngc

Description: Tool Length Probing

This program shows an example of how to measure tool lengths automatically using a switch hooked to the probe input. This is useful for machines without tool holders, where the length of a tool is different every time it is inserted.

20.1.6 Hole Probe

File Name: probe-hole.ngc

Description: Finding the Center and Diameter of a hole.

The program demonstrates how to find the center of a hole, measure the hole diameter and record the results.

20.1.7 Cutter Compensation

To be added

20.2 Lathe Examples

20.2.1 Threading

File Name lathe-g76.ngc

Description: Facing, threading and parting off.

This file shows an example of threading on a lathe using parameters.

Chapter 21

Lathe User Information

This chapter attempts to bring together all the lathe specific information and is currently under construction.

21.1 Lathe Mode

If your CNC machine is a lathe, there are some specific changes you will probably want to make to your ini file in order to get the best results from LinuxCNC.

If you are using the AXIS display, have Axis display your lathe tools properly. See the INI Configuration section of the Integrator Manual for more details but you will probably want to make an entry like this to set up AXIS for Lathe Mode.

```
[DISPLAY]
# Tell the Axis Display our machine is a lathe.
LATHE = TRUE
```

Lathe Mode in Axis does not set your default plane to G18 (XZ). You must program that in the preamble of each gcode file or (better) add it to your ini file, like this:

```
[RS274NGC]
# g-code modal codes (modes) that the interpreter is initialized with on startup
RS274NGC_STARTUP_CODE = G18 G20 G90
```

21.2 Lathe Tool Table

The "Tool Table" is a text file that contains information about each tool. The file is located in the same directory as your configuration and is called "tool.tbl" by default. The tools might be in a tool changer or just changed manually. The file can be edited with a text editor or be updated using G10 L1,L10,L11. There is also a built-in tool table editor in the Axis display. The maximum number of entries in the tool table is 56. The maximum tool and pocket number is 99999.

Earlier versions of LinuxCNC had two different tool table formats for mills and lathes, but since the 2.4.x release, one tool table format is used for all machines. Just ignore the parts of the tool table that don't pertain to your machine, or which you don't need to use. For more information on the specifics of the tool table format, see the [Tool Table](#) Section.

21.3 Lathe Tool Orientation

The following figure shows the lathe tool orientations with the center line angle of each orientation and info on FRONTANGLE and BACKANGLE. The FRONTANGLE and BACKANGLE are clockwise starting at a line parallel to Z+.

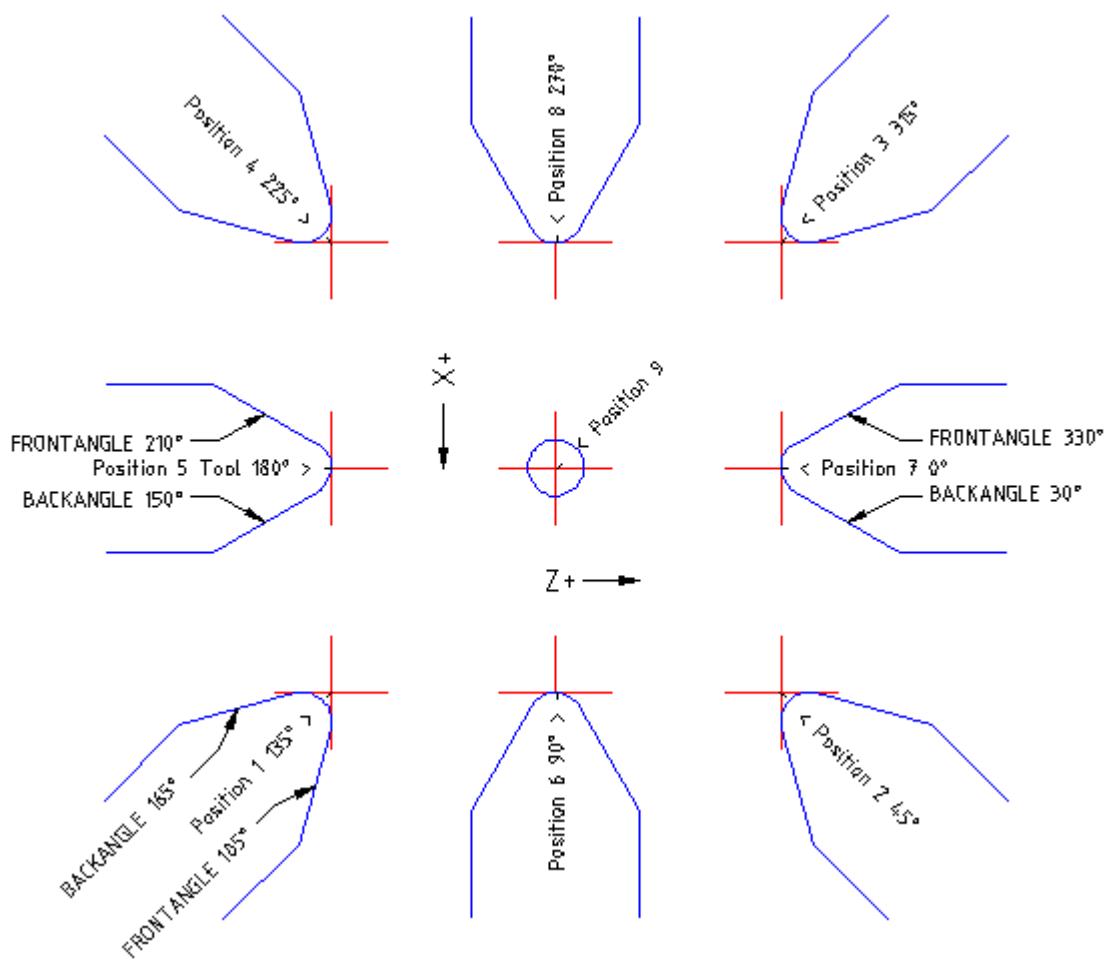


Figure 21.1: Lathe Tool Orientations

In AXIS the following figures show what the Tool Positions look like, as entered in the tool table.

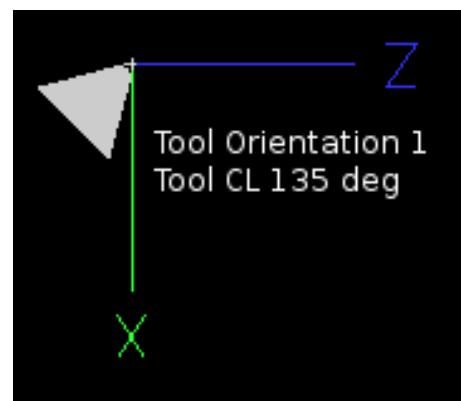
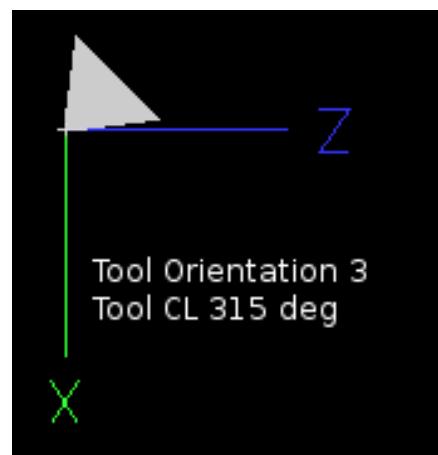
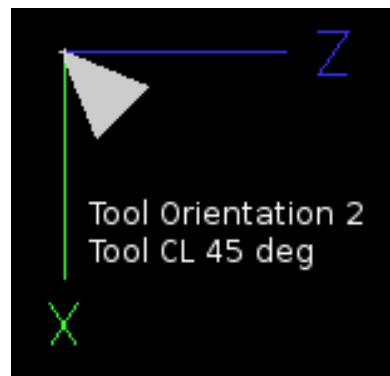


Figure 21.2: Tool Positions 1, 2, 3 & 4



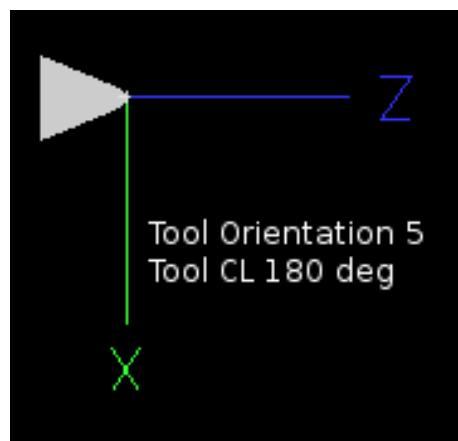
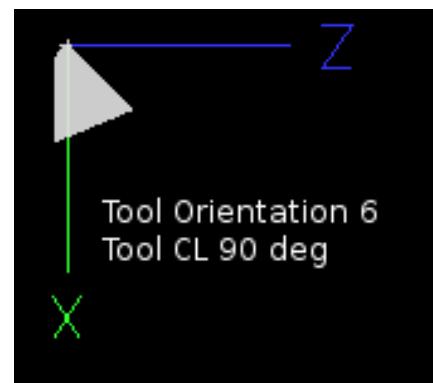
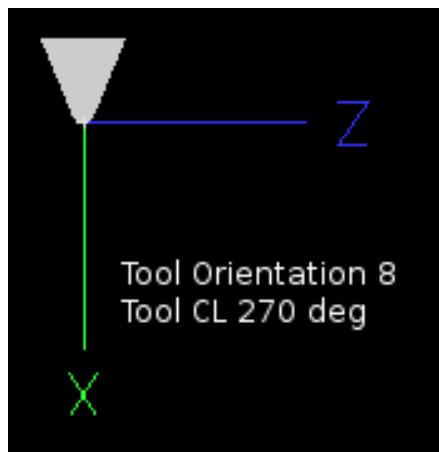


Figure 21.3: Tool Positions 5, 6, 7 & 8





21.4 Tool Touch Off

When running in lathe mode in AXIS you can set the X and Z in the tool table using the Touch Off window. If you have a tool turret you normally have *Touch off to fixture* selected when setting up your turret. When setting the material Z zero you have *Touch off to material* selected. For more information on the G codes used for tools see [M6](#), [Tn](#), and [G43](#). For more information on tool touch off options in Axis see [Tool Touch Off](#).

21.4.1 X Touch Off

The X axis offset for each tool is normally an offset from the center line of the spindle.

One method is to take your normal turning tool and turn down some stock to a known diameter. Using the Tool Touch Off window enter the measured diameter (or radius if in radius mode) for that tool. Then using some layout fluid or a marker to coat the part bring each tool up till it just touches the dye and set its X offset to the diameter of the part used using the tool touch off. Make sure any tools in the corner quadrants have the nose radius set properly in the tool table so the control point is correct. Tool touch off automatically adds a G43 so the current tool is the current offset.

A typical session might be:

1. Home each axis if not homed.
2. Set the current tool with *Tn M6 G43* where *n* is the tool number.
3. Select the X axis in the Manual Control window.
4. Move the X to a known position or take a test cut and measure the diameter.
5. Select Touch Off and pick Tool Table then enter the position or the diameter.
6. Follow the same sequence to correct the Z axis.

Note: if you are in Radius Mode you must enter the radius, not the diameter.

21.4.2 Z Touch Off

The Z axis offsets can be a bit confusing at first because there are two elements to the Z offset. There is the tool table offset, and the machine coordinate offset. First we will look at the tool table offsets. One method is to use a fixed point on your lathe and set the Z offset for all tools from this point. Some use the spindle nose or chuck face. This gives you the ability to change to a new tool and set its Z offset without having to reset all the tools.

A typical session might be:

1. Home each axis if not homed.
2. Make sure no offsets are in effect for the current coordinate system.
3. Set the current tool with $Tn M6 G43$ where n is the tool number.
4. Select the Z axis in the Manual Control window.
5. Bring the tool close to the control surface. Using a cylinder move the Z away from the control surface until the cylinder just passes between the tool and the control surface.
6. Select Touch Off and pick Tool Table and set the position to 0.0.
7. Repeat for each tool using the same cylinder.

Now all the tools are offset the same distance from a standard position. If you change a tool like a drill bit you repeat the above and it is now in sync with the rest of the tools for Z offset. Some tools might require a bit of cyphering to determine the control point from the touch off point. For example, if you have a 0.125" wide parting tool and you touch the left side off but want the right to be Z0, then enter 0.125" in the touch off window.

21.4.3 The Z Machine Offset

Once all the tools have the Z offset entered into the tool table, you can use any tool to set the machine offset using the machine coordinate system.

A typical session might be:

1. Home each axis if not homed.
2. Set the current tool with "Tn M6" where "n" is the tool number.
3. Issue a G43 so the current tool offset is in effect.
4. Bring the tool to the work piece and set the machine Z offset.

If you forget to set the G43 for the current tool when you set the machine coordinate system offset, you will not get what you expect, as the tool offset will be added to the current offset when the tool is used in your program.

21.5 Spindle Synchronized Motion

Spindle synchronized motion requires a quadrature encoder connected to the spindle with one index pulse per revolution. See the motion man page and Spindle Control Example in integrators manual for more information.

Threading The G76 threading cycle is used for both internal and external threads. For more information see the [G76](#) Section.

Constant Surface Speed CSS or Constant Surface Speed uses the machine X origin modified by the tool X offset to compute the spindle speed in RPM. CSS will track changes in tool offsets. The X machine origin should be when the reference tool (the one with zero offset) is at the center of rotation. For more information see the [G96](#) Section.

Feed per Revolution Feed per revolution will move the Z axis by the F amount per revolution. This is not for threading, use G76 for threading. For more information see the [G95](#) Section.

21.6 Arcs

Calculating arcs can be mind challenging enough without considering radius and diameter mode on lathes as well as machine coordinate system orientation. The following applies to center format arcs. On a lathe you should include G18 in your preamble as the default is G17 even if you're in lathe mode, in the user interface Axis. Arcs in G18 XZ plane use I (X axis) and K (Z axis) offsets.

21.6.1 Arcs and Lathe Design

The typical lathe has the spindle on the left of the operator and the tools on the operator side of the spindle center line. This is typically set up with the imaginary Y axis (+) pointing at the floor.

The following will be true on this type of setup:

- The Z axis (+) points to the right, away from the spindle.
- The X axis (+) points toward the operator, and when on the operator side of the spindle the X values are positive.

Some lathes with tools on the back side have the imaginary Y axis (+) pointing up.

G2/G3 Arc directions are based on the axis they rotate around. In the case of lathes, it is the imaginary Y axis. If the Y axis (+) points toward the floor, you have to look up for the arc to appear to go in the correct direction. So looking from above you reverse the G2/G3 for the arc to appear to go in the correct direction.

21.6.2 Radius & Diameter Mode

When calculating arcs in radius mode you only have to remember the direction of rotation as it applies to your lathe.

When calculating arcs in diameter mode X is diameter and the X offset (I) is radius even if you're in G7 diameter mode.

21.7 Tool Path

21.7.1 Control Point

The control point for the tool follows the programmed path. The control point is the intersection of a line parallel to the X and Z axis and tangent to the tool tip diameter, as defined when you touch off the X and Z axes for that tool. When turning or facing straight sided parts the cutting path and the tool edge follow the same path. When turning radius and angles the edge of the tool tip will not follow the programmed path unless cutter comp is in effect. In the following figures you can see how the control point does not follow the tool edge as you might assume.

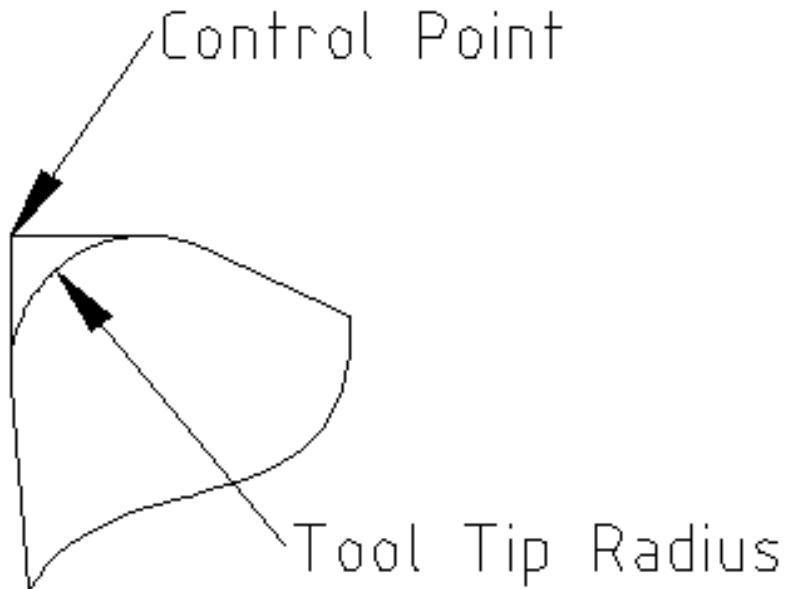


Figure 21.4: Control Point

21.7.2 Cutting Angles without Cutter Comp

Now imagine we program a ramp without cutter comp. The programmed path is shown in the following figure. As you can see in the figure the programmed path and the desired cut path are one and the same as long as we are moving in an X or Z direction only.

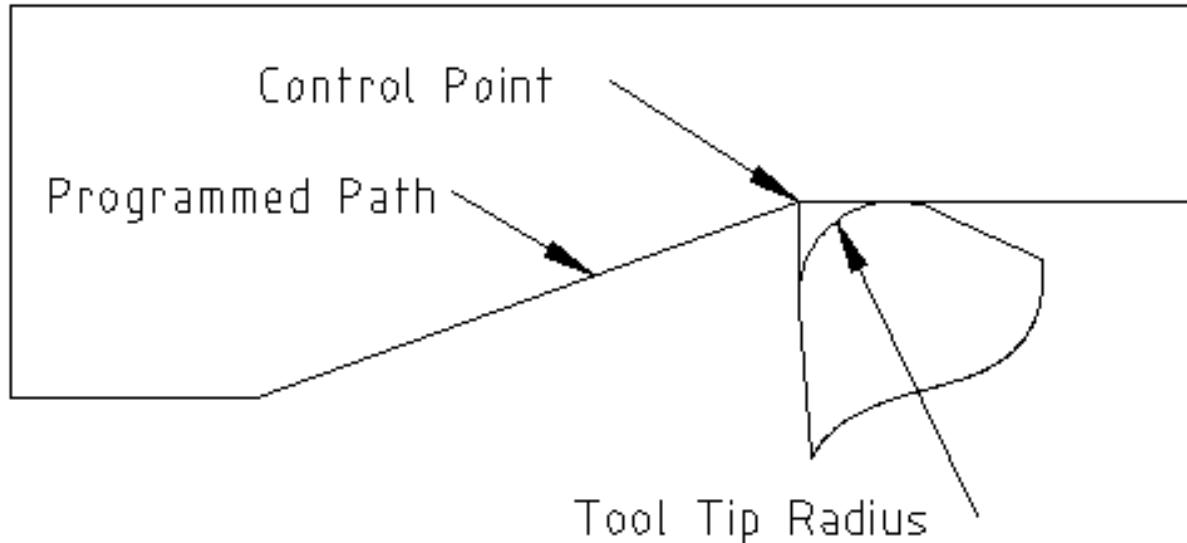


Figure 21.5: Ramp Entry

Now as the control point progresses along the programmed path the actual cutter edge does not follow the programmed path as shown in the following figure. There are two ways to solve this, cutter comp and adjusting your programmed path to compensate for tip radius.

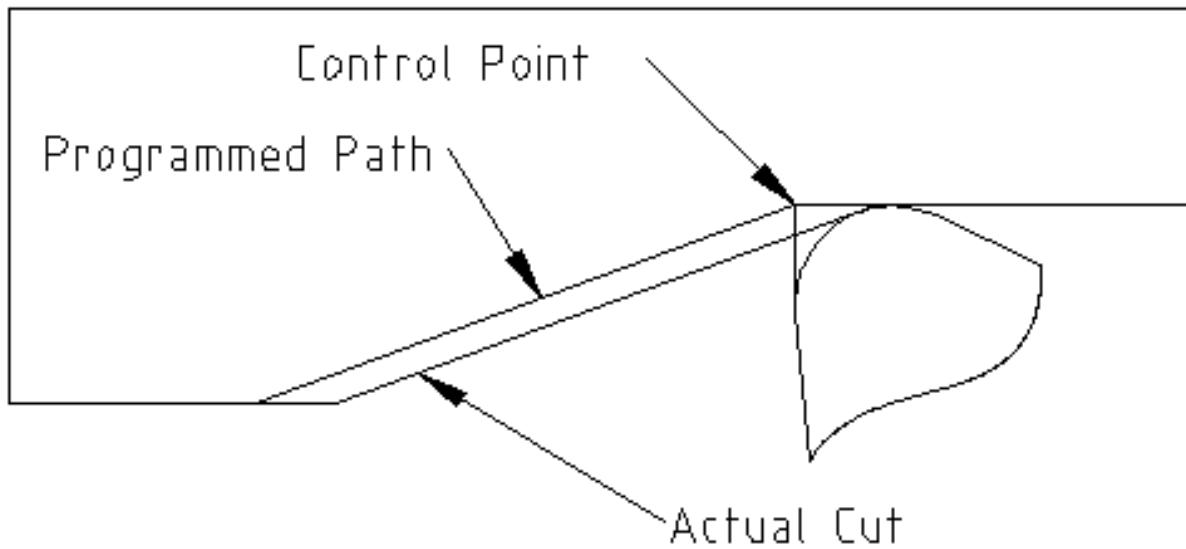


Figure 21.6: Ramp Path

In the above example it is a simple exercise to adjust the programmed path to give the desired actual path by moving the programmed path for the ramp to the left the radius of the tool tip.

21.7.3 Cutting a Radius

In this example we will examine what happens during a radius cut without cutter comp. In the next figure you see the tool turning the OD of the part. The control point of the tool is following the programmed path and the tool is touching the OD of the part.

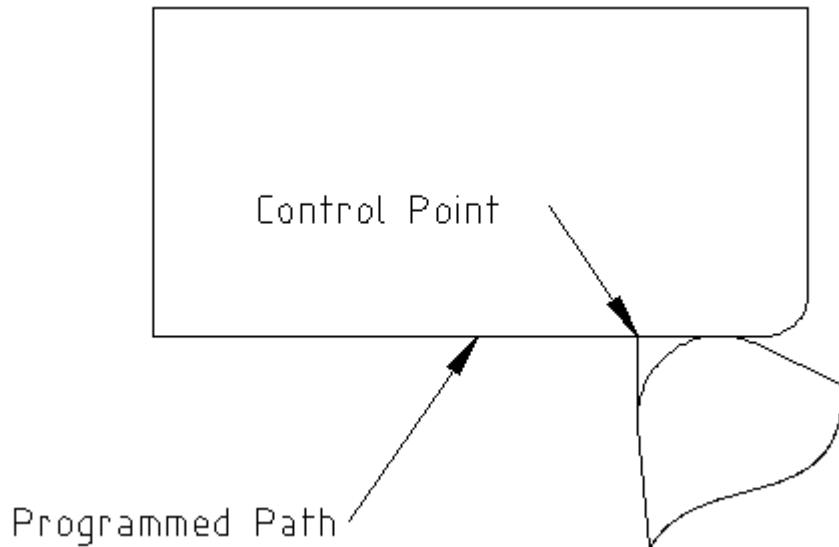


Figure 21.7: Turning Cut

In this next figure you can see as the tool approaches the end of the part the control point still follows the path but the tool tip has left the part and is cutting air. You can also see that even though a radius has been programmed the part will actually end up with a square corner.

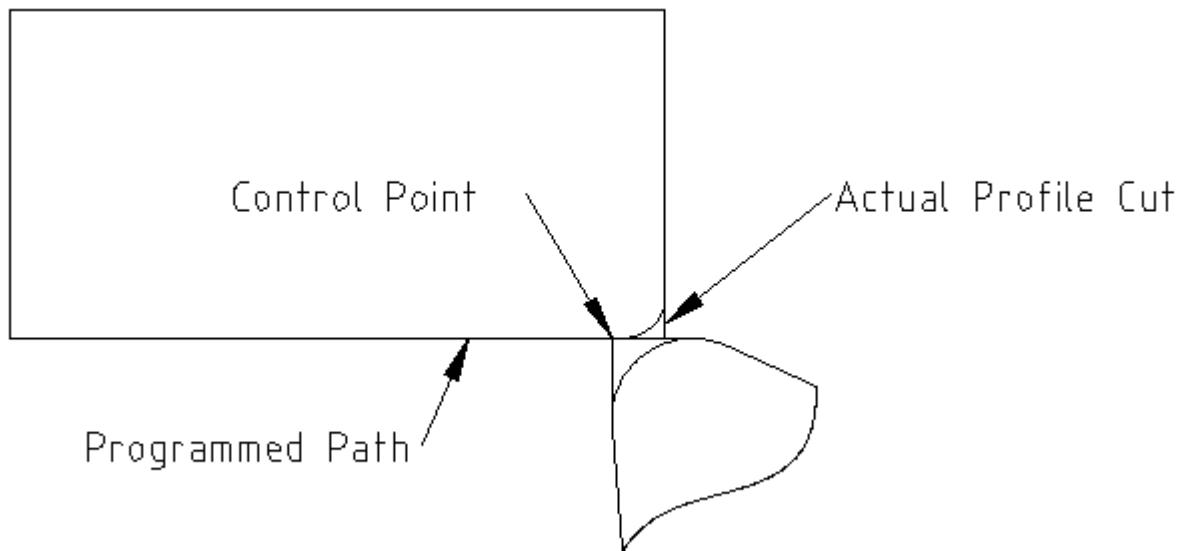


Figure 21.8: Radius Cut

Now you can see as the control point follows the radius programmed the tool tip has left the part and is now cutting air.

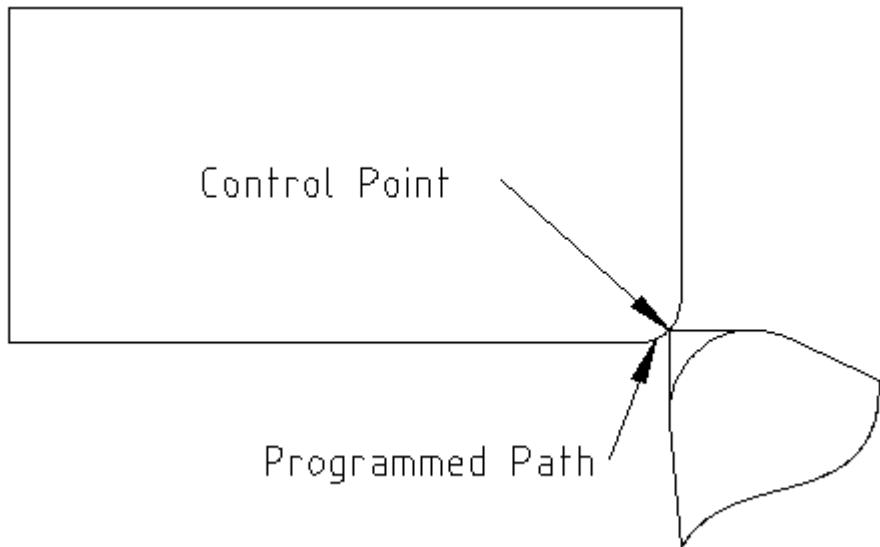


Figure 21.9: Radius Cut

In the final figure we can see the tool tip will finish cutting the face but leave a square corner instead of a nice radius. Notice also that if you program the cut to end at the center of the part a small amount of material will be left from the radius of the tool. To finish a face cut to the center of a part you have to program the tool to go past center at least the nose radius of the tool.

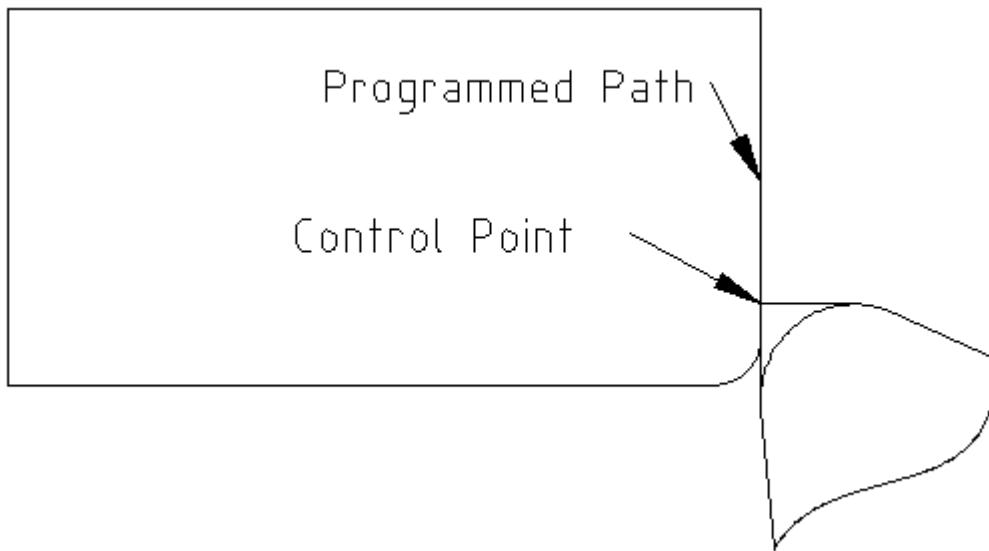


Figure 21.10: Face Cut

21.7.4 Using Cutter Comp

When using cutter comp on a lathe think of the tool tip radius as the radius of a round cutter. When using cutter comp the path must be large enough for a round tool that will not gouge into the next line. When cutting straight lines on the lathe you might not want to use cutter comp. For example boring a hole with a tight fitting boring bar you may not have enough room to do the exit move. The entry move into a cutter comp arc is important to get the correct results.

Chapter 22

RS274/NGC Differences

22.1 Changes from RS274/NGC

DIFFERENCES THAT CHANGE THE MEANING OF RS274/NGC PROGRAMS

Location after a tool change

In LinuxCNC, the machine does not return to its original position after a tool change. This change was made because the new tool might be longer than the old tool, and the move to the original machine position could therefore leave the tool tip too low.

Offset parameters are ini file units

In LinuxCNC, the values stored in parameters for the G28 and G30 home locations, the P1...P9 coordinate systems, and the G92 offset are in "ini file units". This change was made because otherwise the meaning of a location changed depending on whether G20 or G21 was active when G28, G30, G10 L2, or G92.3 is programmed.

Tool table lengths/diameters are in ini file units

In LinuxCNC, the tool lengths (offsets) and diameters in the tool table are specified in ini file units only. This change was made because otherwise the length of a tool and its diameter would change based on whether G20 or G21 was active when initiating G43, G41, G42 modes. This made it impossible to run G code in the machine's non-native units, even when the G code was simple and well-formed (starting with G20 or G21, and didn't change units throughout the program), without changing the tool table.

G84, G87 not implemented

G84 and G87 are not currently implemented, but may be added to a future release of LinuxCNC.

G28, G30 with axis words

When G28 or G30 is programmed with only some axis words present, LinuxCNC only moves the named axes. This is common on other machine controls. To move some axes to an intermediate point and then move all axes to the predefined point, write two lines of G code:

G0 X- Y- (axes to move to intermediate point) G28 (move all axes to predefined point)

22.2 Additions to RS274/NGC

DIFFERENCES THAT DO NOT CHANGE THE MEANING OF RS274/NGC PROGRAMS

G33, G76 threading codes

These codes are not defined in RS274/NGC.

G38.2

The probe tip is not retracted after a G38.2 movement. This retraction move may be added in a future release of LinuxCNC.

G38.3...G38.5

These codes are not defined in RS274/NGC

O-codes

These codes are not defined in RS274/NGC

M50...M53 overrides

These codes are not defined in RS274/NGC

M61..M66

These codes are not defined in RS274/NGC

G43, G43.1*Negative Tool Lengths*

The RS274/NGC spec says "it is expected that" all tool lengths will be positive. However, G43 works for negative tool lengths.

Lathe tools

G43 tool length compensation can offset the tool in both the X and Z dimensions. This feature is primarily useful on lathes.

Dynamic tool lengths

LinuxCNC allows specification of a computed tool length through G43.1 I K.

G41.1, G42.1

LinuxCNC allows specification of a tool diameter and, if in lathe mode, orientation in the G code. The format is G41.1/G42.1 D L, where D is diameter and L (if specified) is the lathe tool orientation.

G43 without H word

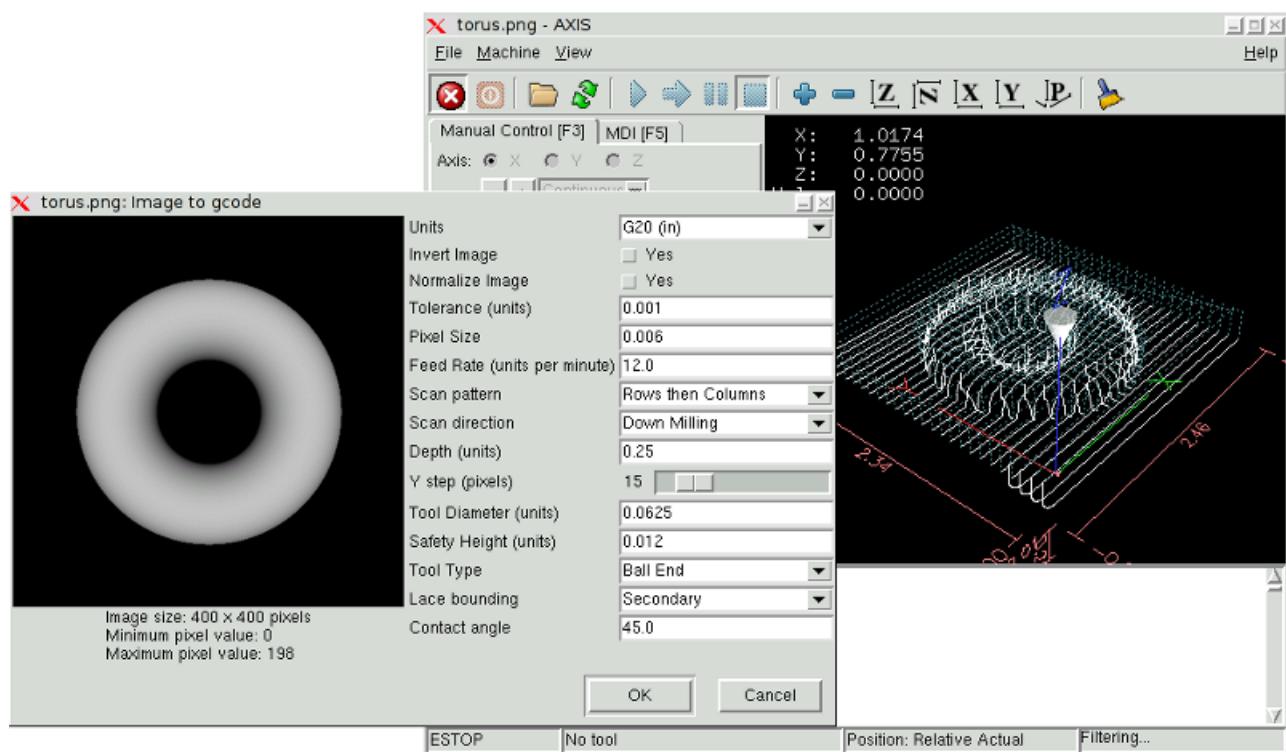
In ngc, this is not allowed. In LinuxCNC, it sets length offsets for the currently loaded tool. If no tool is currently loaded, it is an error. This change was made so the user doesn't have to specify the tool number in two places for each tool change, and because it's consistent with the way G41/G42 work when the D word is not specified.

U, V, and W axes

LinuxCNC allows machines with up to 9 axes by defining an additional set of 3 linear axes known as U, V and W

Chapter 23

Image to G Code



23.1 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.

23.2 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.

23.3 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.

23.4 Option Reference

23.4.1 Units

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

23.4.2 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.

23.4.3 Normalize Image

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

23.4.4 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.

23.4.5 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.

23.4.6 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 $2.5 / 400 = .00625$.

23.4.7 Plunge Feed Rate (units per minute)

The feed rate for the initial plunge movement.

23.4.8 Feed Rate (units per minute)

The feed rate for other parts of the path.

23.4.9 Spindle Speed (RPM)

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

23.4.10 Scan Pattern

Possible scan patterns are:

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

23.4.11 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

23.4.12 Depth (units)

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

23.4.13 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 $.015/.006=2.5$ '.

23.4.14 Tool Diameter

The diameter of the cutting part of the tool.

23.4.15 Safety Height

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

23.4.16 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”

23.4.17 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.

23.4.18 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.

23.4.19 Roughing offset and depth per pass

Image-to-gcode can optionally perform roughing 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.

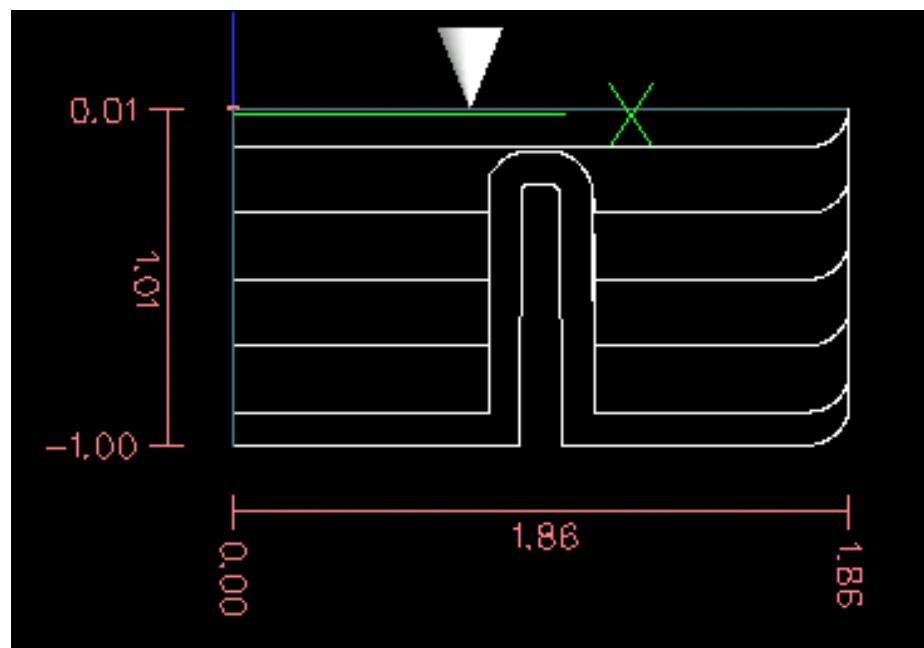


Figure 23.1: Roughing passes and final pass

Chapter 24

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.

Gmoccapy (GUI)

A Graphical User Interfaces available to users of LinuxCNC. It features the use and feel of an industrial control and can be used with touch screen, mouse and keyboard. It support embedded tabs and hal driven user messages, it offers a lot of hal beans to be controled with hardware. Gmoccapy is highly cusomizable.

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 a 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 25

Legal Section

25.1 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

25.2 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 26

Index

- axisrc, 41
- A**
 - acme screw, 241
 - Arc Distance Mode, 200
 - Arc Move, 167
 - Auto, 113, 122
 - AXIS, 25, 39
 - axis, 241
 - Axis GUI, 25
 - AXIS in lathe mode, 39
 - Axis Menu, 27
 - Axis Preview Control, 42
 - AxisUI
 - coolant, 35
 - feed override, 35
 - jog speed, 36
 - keyboard shortcuts, 36
 - Max Velocity, 36
 - MDI, 35
 - spindle, 34
 - spindle speed override, 36
- B**
 - backlash, 241
 - backlash compensation, 241
 - backplot, 118
 - ball nut, 241
 - ball screw, 241
 - Block Delete, 145
 - block delete, 129
 - break, 215
- C**
 - call, 215
 - Calling Files, 217
 - CNC, 27, 242
 - CNC Machine Overview, 126
 - comp, 242
 - Conditional: if
 - elseif
 - else, 216
- D**
 - CONFIGURATION SELECTOR, 23
 - continue, 215
 - controlled point, 127
 - coolant, 35, 126, 128
 - coordinate measuring machine, 242
 - Coordinate System, 131
- E**
 - Debug Messages, 159
 - display units, 242
 - do, 215
 - DRO, 242
 - Dwell
 - Feed Out, 199
 - dwell, 128
- F**
 - EDM, 242
 - else, 216
 - elseif, 216
 - else, 216
 - EMC, 242
 - EMCIO, 242
 - EMCMOT, 242
 - encoder, 242
 - endif, 216
 - endsub, 215
 - endwhile, 215
 - ESTOP), 30
 - External Editor, 41
- G**
 - F: Set Feed Rate, 219
 - feed, 242
 - Feed Out, 198, 199
 - feed override, 35, 116, 127
 - feed rate, 128, 242
 - feedback, 242
 - feedrate override, 243
- G**
 - G Code Best Practices, 161
 - G Code Order of Execution, 160
 - G Code Overview, 144

G Code Table, 164
G Codes, 164
G-Code, 243
G0 Rapid Move, 165
G1 Linear Move, 166
G10 L1 Tool Table, 175
G10 L10 Set Tool Table, 177
G10 L11 Set Tool Table, 177
G10 L2 Coordinate System, 175
G10 L20 Set Coordinate System, 178
G17 G18 G19 Plane Selection, 178
G2
 G3 Arc Move, 167
G20 Inches, 178
G21 Millimeters, 178
G28, 179
 G3 Arc Move, 167
G30, 179
 G33 Spindle Synchronized Motion, 180
 G33.1 Rigid Tapping, 180
 G38.x Probe, 181
 G4 Dwell, 171
 G40 Cutter Compensation Off, 182
 G41 G42 Cutter Compensation, 183
 G41.1 G42.1 Dynamic Compensation, 183
 G43 Tool Length Offset, 184
 G43.1 Dynamic Tool Length Offset, 184
 G43.2 Apply additional Tool Length Offset, 185
 G49 Cancel Tool Length Offset, 185
 G5 Cubic spline, 172
 G5.1 Quadratic spline, 172
 G5.2 G5.3 NURBS Block, 173
 G53 Machine Coordinates, 185
 G54-G59.3 Select Coordinate System, 186
 G55, 132
 G61 G61.1 G64 Path Control, 186
 G64 Path Blending, 186
 G7 Lathe Diameter Mode, 174
 G73 Drilling Cycle Chip Break, 187
 G76 Threading, 188
 G8 Lathe Radius Mode, 174
 G80 Cancel Modal Motion, 193
 G80-G89 Canned Cycles, 190
 G81 Drilling Cycle, 194
 G82 Drilling Cycle Dwell, 197
 G83 Peck Drilling, 198
 G84 Right-Hand Tapping, 198
 G85 Boring
 Feed Out, 198
 G86 Boring
 Spindle Stop
 Rapid Move Out, 199
 G87 Back Boring, 199
 G88 Boring Cycle
 Spindle Stop
 Manual Out, 199
 G89 Boring
Dwell
 Feed Out, 199
G90
 G91 Distance Mode, 199
G91 Distance Mode, 199
G92 Coordinate System Offset, 200
G93
 G94
 G95: Feed Rate Mode, 201
G94
 G95: Feed Rate Mode, 201
G95: Feed Rate Mode, 201
G96
 G97 Spindle Control Mode, 201
G97 Spindle Control Mode, 201
G98
 G99 Canned Cycle Return, 202
G99 Canned Cycle Return, 202
GMOCCAPY, 43
gmoccapy, 43
GUI, 241, 243

H

HAL, 243
home, 243

I

if, 216
Image to G Code, 236
Indirection, 217
INI, 243
Instance, 243

J

jog, 243
jog speed, 36
joint coordinates, 243

K

keyboard shortcuts, 36
KEYSTICK, 123
kinematics, 243

L

Lathe User Information, 223
lead screw, 243
Line Number, 145
Linear Move, 166
Linux, 6
LinuxCNC User Introduction, 5
Logging, 159
loop, 244
Looping, 215

M

M Codes, 203
M0 Program Pause, 203
M1 Program Optional Pause, 203

- M100 to M199 User Defined Commands, [212](#)
M19 Orient Spindle, [205](#)
M2 Program End, [204](#)
M3 Spindle CW, [204](#)
M30 Program End, [204](#)
M4 Spindle CCW, [204](#)
M48
 M49 Override Control, [206](#)
M49 Override Control, [206](#)
M5 Spindle Stop, [204](#)
M50 Feed Override Control, [206](#)
M51 Spindle Speed Override, [206](#)
M52 Adaptive Feed Control, [207](#)
M53 Feed Stop Control, [207](#)
M6-Tool-Change, [204](#)
M60 Pallet Change Pause, [204](#)
M61 Set Current Tool Number, [207](#)
M62 to M65 Output Control, [207](#)
M66 Input Control, [208](#)
M67 Analog Motion Output Control, [208](#)
M68 Analog Aux Output Control, [209](#)
M7 Mist Coolant, [205](#)
M70 Save Modal State, [209](#)
M71 Invalidate Stored Modal State, [210](#)
M72 Restore Modal State, [210](#)
M73 Save and Autorestore Modal State, [211](#)
M8 Flood Coolant, [205](#)
M9 Coolant Off, [205](#)
machine on, [30](#)
machine units, [243](#)
Manual, [33](#), [112](#), [121](#)
Manual Out, [199](#)
Manual Tool Change, [38](#)
Max Velocity, [36](#)
MDI, [35](#), [244](#)
Messages, [159](#)
Mini GUI, [109](#)
Modal Groups, [157](#)
- N**
NGCGUI, [83](#)
NIST, [244](#)
NML, [244](#)
- O**
O Codes, [214](#)
offsets, [244](#)
OpenGL, [25](#)
operator precedence, [153](#)
optional block delete, [127](#)
optional program stop, [127](#), [129](#)
Other Codes, [219](#)
- P**
Parameters, [146](#)
parameters, [130](#)
part Program, [244](#)
- Path Control, [186](#)
path control mode, [129](#)
Plane Selection, [178](#)
Polar Coordinates, [155](#)
preview plot, [32](#)
Print Messages, [159](#)
Probe Logging, [159](#)
program extents, [32](#)
program units, [244](#)
Programming the Planner, [17](#)
Python, [25](#), [39](#)
- R**
rapid, [244](#)
Rapid Move, [165](#)
Rapid Move Out, [199](#)
rapid rate, [244](#)
real-time, [244](#)
Repeat, [217](#)
return, [215](#)
Return Values, [218](#)
RS274/NGC Programs, [234](#)
RS274NGC, [244](#)
RTAI, [244](#)
RTAPI, [244](#)
RTLINUX, [244](#)
- S**
S: Set Spindle Speed, [219](#)
servo motor, [244](#)
Sherline, [110](#)
Signed Integer, [245](#)
spindle, [34](#), [126](#), [245](#)
spindle speed override, [36](#), [127](#)
Spindle Stop
 Manual Out, [199](#)
 Rapid Move Out, [199](#)
stepper motor, [245](#)
sub, [215](#)
Subroutines, [215](#)
- T**
T: Select Tool, [219](#)
TASK, [245](#)
Tk, [25](#), [245](#)
tkLinuxCNC, [103](#)
TKLinuxCNC GUI, [103](#)
Tool Compensation, [137](#)
Tool Touch Off, [29](#)
Tool-Table-Format, [138](#)
Touch Off, [137](#)
Touchy GUI, [100](#)
Trajectory Control, [17](#), [186](#)
Traverse Move, [245](#)
- U**
units, [128](#), [245](#)

Unsigned Integer, [245](#)

User Concepts, [17](#)

User Defined Commands M100-M199, [212](#)

User Foreword, [3](#)

V

Virtual Control Panel, [41](#)

W

while, [215](#)

Word, [145](#)

world coordinates, [245](#)

NAME

linuxcnc – LinuxCNC (The Enhanced Machine Controller)

SYNOPSIS

linuxcnc [-v] [-d] [*INIFILE*]

DESCRIPTION

linuxcnc is used to start LinuxCNC (The Enhanced Machine Controller). It starts the realtime system and then initializes a number of LinuxCNC components (IO, Motion, GUI, HAL, etc). The most important parameter is *INIFILE*, which specifies the configuration name you would like to run. If *INIFILE* is not specified, the **linuxcnc** script presents a graphical wizard to let you choose one.

OPTIONS

- v Be a little bit verbose. This causes the script to print information as it works.
- d Print lots of debug information. All executed commands are echoed to the screen. This mode is useful when something is not working as it should.

INIFILE

The ini file is the main piece of an LinuxCNC configuration. It is not the entire configuration; there are various other files that go with it (NML files, HAL files, TBL files, VAR files). It is, however, the most important one, because it is the file that holds the configuration together. It can adjust a lot of parameters itself, but it also tells **linuxcnc** which other files to load and use.

There are several ways to specify which config to use:

Specify the absolute path to an ini, e.g.

linuxcnc /usr/local/linuxcnc/configs/sim/sim.ini

Specify a relative path from the current directory, e.g.

linuxcnc configs/sim/sim.ini

Otherwise, in the case where the **INIFILE** is not specified, the behavior will depend on whether you configured **linuxcnc** with **--enable-run-in-place**. If so, the **linuxcnc** config chooser will search only the configs directory in your source tree. If not (or if you are using a packaged version of **linuxcnc**), it may search several directories. The config chooser is currently set to search the path:

~/linuxcnc/configs:/home/buildslave/emc2-buildbot/wheezy-amd64-clang/docs/build/configs

EXAMPLES

linuxcnc

linuxcnc configs/sim/sim.ini

linuxcnc /etc/linuxcnc/sample-configs/stepper/stepper_mm.ini

SEE ALSO

halcmd(1)

Much more information about LinuxCNC and HAL is available in the LinuxCNC and HAL User Manuals, found at /usr/share/doc/linuxcnc/.

HISTORY

BUGS

None known at this time.

AUTHOR

This man page written by Alex Joni, as part of the LinuxCNC Enhanced Machine Controller project.

REPORTING BUGS

Report bugs to alex_joni AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2006 Alex Joni.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

axis-remote – AXIS Remote Interface

SYNOPSIS

axis-remote *OPTIONS|FILENAME*

DESCRIPTION

axis-remote is a small script that triggers commands in a running AXIS GUI. Use **axis-remote --help** for further information.

OPTIONS

--ping, -p

Check whether AXIS is running.

--reload, -r

Make AXIS reload the currently loaded file.

--clear, -c

Make AXIS clear the backplot.

--quit, -q

Make AXIS quit.

--help, -h, -?

Display a list of valid parameters for **axis-remote**.

--mdi COMMAND, -m COMMAND

Run the MDI command **COMMAND**.

FILENAME

Load the G-code file **FILENAME**.

SEE ALSO

axis(1)

Much more information about LinuxCNC and HAL is available in the LinuxCNC and HAL User Manuals, found at /usr/share/doc/linuxcnc/.

HISTORY**BUGS**

None known at this time.

AUTHOR

This man page written by Alex Joni, as part of the LinuxCNC project.

REPORTING BUGS

Report bugs to alex_joni AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2007 Alex Joni.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

axis – AXIS LinuxCNC Graphical User Interface

SYNOPSIS

axis -ini INIFILE

DESCRIPTION

axis is one of the Graphical User Interfaces (GUI) for LinuxCNC It gets run by the runscript usually.

OPTIONS**INIFILE**

The ini file is the main piece of an LinuxCNC configuration. It is not the entire configuration; there are various other files that go with it (NML files, HAL files, TBL files, VAR files). It is, however, the most important one, because it is the file that holds the configuration together. It can adjust a lot of parameters itself, but it also tells **LinuxCNC** which other files to load and use.

SEE ALSO

LinuxCNC(1)

Much more information about LinuxCNC and HAL is available in the LinuxCNC and HAL User Manuals, found at /usr/share/doc/LinuxCNC/.

HISTORY**BUGS**

None known at this time.

AUTHOR

This man page written by Alex Joni, as part of the LinuxCNC project.

REPORTING BUGS

Report bugs to alex_joni AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2007 Alex Joni.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

comp – Build, compile and install LinuxCNC HAL components

SYNOPSIS

```
comp [--compile|--preprocess|--document|--view-doc] compfile...
sudo comp [--install|--install-doc] compfile...
    comp --compile --userspace cfile...
sudo comp --install --userspace cfile...
sudo comp --install --userspace pyfile...
```

DESCRIPTION

comp performs many different functions:

- Compile **.comp** and **.c** files into **.so** or **.ko** HAL realtime components (the **--compile** flag)
- Compile **.comp** and **.c** files into HAL userspace components (the **--compile --userspace** flag)
- Preprocess **.comp** files into **.c** files (the **--preprocess** flag)
- Extract documentation from **.comp** files into **.9** manpage files (the **--document** flag)
- Display documentation from **.comp** files onscreen (the **--view-doc** flag)
- Compile and install **.comp** and **.c** files into the proper directory for HAL realtime components (the **--install** flag), which may require **sudo** to write to system directories.
- Install **.c** and **.py** files into the proper directory for HAL userspace components (the **--install --userspace** flag), which may require **sudo** to write to system directories.
- Extract documentation from **.comp** files into **.9** manpage files in the proper system directory (the **--install** flag), which may require **sudo** to write to system directories.
- Preprocess **.comp** files into **.c** files (the **--preprocess** flag)

SEE ALSO

Comp HAL Component Generator in the LinuxCNC documentation for a full description of the **.comp** syntax, along with examples

pydoc hal and *Creating Userspace Python Components* in the LinuxCNC documentation for documentation on the Python interface to HAL components

comp(9) for documentation on the "two input comparator with hysteresis", a HAL realtime component with the same name as this program

NAME

gladevcp – Virtual Control Panel for LinuxCNC based on Glade, Gtk and HAL widgets

SYNOPSIS

gladevcp [-g *WxH+X+Y*] [-c *component-name*] [-u *handler*] [-U *useroption*] [-H *halfile*] [-d] *myfile.ui*

OPTIONS

-g *WxH+X+Y*

This sets the initial geometry of the root window. Use 'WxH' for just size, '+X+Y' for just position, or 'WxH+X+Y' for both. Size / position use pixel units. Position is referenced from top left.

-c *component-name*

Use *component-name* as the HAL component name. If the component name is not specified, the basename of the ui file is used.

-u *handler*

Instructs gladevcp to inspect the Python script *handler* for event handlers, and connect them to signals in the ui file.

-U *useroption*

gladevcp collects all *useroption* strings and passes them to the handler init() method as a list of strings without further inspection.

-x *XID* Reparent gladevcp into an existing window *XID* instead of creating a new top level window.

-H *halfile*

gladevcp runs *halfile* - a list of HAL commands - by executing *halcmd -c halfile* after the HAL component is finalized.

-d enable debug output.

-R *gtkrcfile*

explicitly load a gtkrc file.

-t *THEME*

set gtk theme. Default is *system* theme. Different panels can have different themes.

-m *MAXIMUM*

force panel window to maxumize. Together with the **-g** *geometry* option one can move the panel to a second monitor and force it to use all of the screen

-R

explicitly deactivate workaround for a gtk bug which makes matches of widget and widget_class matches in gtk theme and gtkrc files fail. Normally not needed.

SEE ALSO

GladeVCP in the LinuxCNC documentation for a description of gladevcp's capabilities and the associated HAL widget set, along with examples

NAME

gs2_vfd - HAL userspace component for Automation Direct GS2 VFD's

SYNOPSIS

gs2_vfd [OPTIONS]

DESCRIPTION

This manual page explains the **gs2_vfd** component. This component reads and writes to the GS2 via a modbus connection.

gs2_vfd is for use with LinuxCNC

OPTIONS

-b, --bits <n>

(default 8) Set number of data bits to <n>, where n must be from 5 to 8 inclusive

-d, --device <path>

(default /dev/ttyS0) Set the name of the serial device node to use.

-v, --verbose

Turn on verbose mode.

-g, --debug

Turn on debug messages. Note that if there are serial errors, this may become annoying. Debug mode will cause all modbus messages to be printed in hex on the terminal.

-n, --name <string>

(default gs2_vfd) Set the name of the HAL module. The HAL comp name will be set to <string>, and all pin and parameter names will begin with <string>.

-p, --parity [even,odd,none]

(default odd) Set serial parity to even, odd, or none.

-r, --rate <n>

(default 38400) Set baud rate to <n>. It is an error if the rate is not one of the following: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200

-s, --stopbits [1,2]

(default 1) Set serial stop bits to 1 or 2

-t, --target <n>

(default 1) Set MODBUS target (slave) number. This must match the device number you set on the GS2.

-A, --accel-seconds <n>

(default 10.0) Seconds to accelerate the spindle from 0 to Max RPM.

-D, --decel-seconds <n>

(default 0.0) Seconds to decelerate the spindle from Max RPM to 0. If set to 0.0 the spindle will be allowed to coast to a stop without controlled deceleration.

-R, --braking-resistor

This argument should be used when a braking resistor is installed on the GS2 VFD (see Appendix A of the GS2 manual). It disables deceleration over-voltage stall prevention (see GS2 modbus Parameter 6.05), allowing the VFD to keep braking even in situations where the motor is regenerating high voltage. The regenerated voltage gets safely dumped into the braking resistor.

PINS

<name>.DC-bus-volts (float, out)
 from the VFD

<name>.at-speed (bit, out)
 when drive is at commanded speed

<name>.err-reset (bit, in)
 reset errors sent to VFD

<name>.firmware-revision (s32, out)
 from the VFD

<name>.frequency-command (float, out)
 from the VFD

<name>.frequency-out (float, out)
 from the VFD

<name>.is-stopped (bit, out)
 when the VFD reports 0 Hz output

<name>.load-percentage (float, out)
 from the VFD

<name>.motor-RPM (float, out)
 from the VFD

<name>.output-current (float, out)
 from the VFD

<name>.output-voltage (float, out)
 from the VFD

<name>.power-factor (float, out)
 from the VFD

<name>.scale-frequency (float, out)
 from the VFD

<name>.speed-command (float, in)
 speed sent to VFD in RPM It is an error to send a speed faster than the Motor Max RPM as set in the VFD

<name>.spindle-fwd (bit, in)
 1 for FWD and 0 for REV sent to VFD

<name>.spindle-on (bit, in)
 1 for ON and 0 for OFF sent to VFD, only on when running

<name>.spindle-rev (bit, in)
 1 for ON and 0 for OFF, only on when running

<name>.status-1 (s32, out)
 Drive Status of the VFD (see the GS2 manual)

<name>.status-2 (s32, out)
 Drive Status of the VFD (see the GS2 manual) Note that the value is a sum of all the bits that are on. So a 163 which means the drive is in the run mode is the sum of 3 (run) + 32 (freq set by serial) + 128 (operation set by serial).

PARAMETERS

<name>.error-count (s32, RW)

<name>.loop-time (float, RW)
how often the modbus is polled (default 0.1)

<name>.nameplate-HZ (float, RW)
Nameplate Hz of motor (default 60)

<name>.nameplate-RPM (float, RW)
Nameplate RPM of motor (default 1730)

<name>.retval (s32, RW)
the return value of an error in HAL

<name>.tolerance (float, RW)
speed tolerance (default 0.01)

<name>.ack-delay (s32, RW)
number of read/write cycles before checking at-speed (default 2)

SEE ALSO

GS2 Driver in the LinuxCNC documentation for a full description of the **GS2** syntax

GS2 Examples in the LinuxCNC documentation for examples using the **GS2** component

BUGS

AUTHOR

John Thornton

LICENSE

GPL

NAME

hal_input – control HAL pins with any Linux input device, including USB HID devices

SYNOPSIS

loadusr hal_input [-KRAL] inputspec ...

DESCRIPTION

hal_input is an interface between HAL and any Linux input device, including USB HID devices. For each device named, **hal_input** creates pins corresponding to its keys, absolute axes, and LEDs. At a fixed rate of approximately 10ms, it synchronizes the device and the HAL pins.

INPUT SPECIFICATION

The *inputspec* may be in one of several forms:

A string *S*

A substring or shell-style pattern match will be tested against the "name" of the device, the "phys" (which gives information about how it is connected), and the "id", which is a string of the form "Bus=... Vendor=... Product=... Version=...". You can view the name, phys, and id of attached devices by executing **less /proc/bus/input/devices**. Examples:

```
SpaceBall
"Vendor=001f Product=0001"
serio*/input0
```

A number *N*

This opens **/dev/input/event*N***. Except for devices that are always attached to the system, this number may change over reboots or when the device is removed. For this reason, using an integer is not recommended.

When several devices are identified by the same string, add ":"*N*" where *N* is the index of the desired device. For example, if **Mouse** matches **input3** and **input10**, then **Mouse** and **Mouse:0** select **input3**. Specifying **mouse:1** selects **input10**.

For devices that appear as multiple entries in **/dev/input**, these indices are likely to stay the same every time. For multiple identical devices, these indices are likely to depend on the insertion order, but stay the same across reboots as long as the devices are not moved to different ports or unplugged while the machine is booted.

If the first character of the *inputspec* is a "+", then **hal_input** requests exclusive access to the device. The first device matching an *inputspec* is used. Any number of *inputspecs* may be used.

A *subset option* may precede each *inputspec*. The subset option begins with a dash. Each letter in the subset option specifies a device feature to **include**. Features that are not specified are excluded. For instance, to export keyboard LEDs to HAL without exporting keys, use

```
hal_input -L keyboard ...
```

DEVICE FEATURES SUPPORTED

- EV_KEY (buttons and keys). Subset -K
- EV_ABS (absolute analog inputs). Subset -A
- EV_REL (relative analog inputs). Subset -R
- EV_LED (LED outputs). Subset -L

HAL PINS AND PARAMETERS**For buttons**

input.N.btn-name bit out

input.N.btn-name-not bit out

Created for each button on the device.

For keys**input.N.key-name****input.N.key-name-not**

Created for each key on the device.

For absolute axes**input.N.abs-name-counts** s32 out**input.N.abs-name-position** float out**input.N.abs-name-scale** parameter float rw**input.N.abs-name-offset** parameter float rw**input.N.abs-name-fuzz** parameter s32 rw**input.N.abs-name-flat** parameter s32 rw**input.N.abs-name-min** parameter s32 r**input.N.abs-name-max** parameter s32 r

Created for each absolute axis on the device. Device positions closer than **flat** to **offset** are reported as **offset** in **counts**, and **counts** does not change until the device position changes by at least **fuzz**. The position is computed as **position** = (**counts** - **offset**) / **scale**. The default value of **scale** and **offset** map the range of the axis reported by the operating system to [-1,1]. The default values of **fuzz** and **flat** are those reported by the operating system. The values of **min** and **max** are those reported by the operating system.

For relative axes**input.N.rel-name-counts** s32 out**input.N.rel-name-position** float out**input.N.rel-name-reset** bit in**input.N.rel-name-scale** parameter float rw**input.N.rel-name-absolute** parameter s32 rw**input.N.rel-name-precision** parameter s32 rw**input.N.rel-name-last** parameter s32 rw

Created for each relative axis on the device. As long as **reset** is true, **counts** is reset to zero regardless of any past or current axis movement. Otherwise, **counts** increases or decreases according to the motion of the axis. **counts** is divided by position-scale to give **position**. The default value of **position** is 1. There are some devices, notably scroll wheels, which return signed values with less resolution than 32 bits. The default value of **precision** is 32. **precision** can be set to 8 for a device that returns signed 8 bit values, or any other value from 1 to 32. **absolute**, when set true, ignores duplicate events with the same value. This allows for devices that repeat events without any user action to work correctly. **last** shows the most recent count value returned by the device, and is used in the implementation of **absolute**.

For LEDs**input.N.led-name** bit out**input.N.led-name-invert** parameter bit rw

Created for each LED on the device.

PERMISSIONS AND UDEV

By default, the input devices may not be accessible to regular users--**hal_input** requires read-write access, even if the device has no outputs. To change the default permission of a device, add a new file to /etc/udev/rules.d to set the device's GROUP to "plugdev". You can do this for all input devices with this rule:

SUBSYSTEM=="input", MODE="0660", GROUP="plugdev"

You can also make more specific rules for particular devices. For instance, a SpaceBall input device uses the 'spaceball' kernel module, so a udev entry for it would read:

DRIVER=="spaceball", MODE="0660", GROUP="plugdev"

the next time the device is attached to the system, it will be accessible to the "plugdev" group.

For USB devices, the udev line would refer to the device's Vendor and Product values, such as

SYSFS{idProduct}=="c00e", SYSFS{idVendor}=="046d", MODE="0660", GROUP="plugdev"

for a particular logitech-brand mouse.

For more information on writing udev rules, see **udev(8)**.

BUGS

The initial state of keys, buttons, and absolute axes are erroneously reported as FALSE or 0 until an event is received for that key, button, or axis.

SEE ALSO

udev(8)

NAME

halcmd – manipulate the LinuxCNC HAL from the command line

SYNOPSIS

halcmd [*OPTIONS*] [*COMMAND* [*ARG*]]

DESCRIPTION

halcmd is used to manipulate the HAL (Hardware Abstraction Layer) from the command line. **halcmd** can optionally read commands from a file, allowing complex HAL configurations to be set up with a single command.

If the **readline** library is available when LinuxCNC is compiled, then **halcmd** offers commandline editing and completion when running interactively. Use the up arrow to recall previous commands, and press tab to complete the names of items such as pins and signals.

OPTIONS

- I** Before tearing down the realtime environment, run an interactive halcmd. **halrun** only. If **-I** is used, it must precede all other commandline arguments.
- f [file]** Ignore commands on command line, take input from *file* instead. If *file* is not specified, take input from *stdin*.
- i *infile*** Use variables from *infile* for substitutions. See **SUBSTITUTION** below.
- k** Keep going after failed command(s). The default is to stop and return failure if any command fails.
- q** display errors only (default)
- Q** display nothing, execute commands silently
- s** Script-friendly mode. In this mode, *show* will not output titles for the items shown. Also, module names will be printed instead of ID codes in pin, param, and funct listings. Threads are printed on a single line, with the thread period, FP usage and name first, followed by all of the functions in the thread, in execution order. Signals are printed on a single line, with the type, value, and signal name first, followed by a list of pins connected to the signal, showing both the direction and the pin name. No prompt will be printed if both **-s** and **-f** are specified.
- R** Release the HAL mutex. This is useful for recovering when a HAL component has crashed while holding the HAL mutex.
- v** display results of each command
- V** display lots of debugging junk
- h [*command*]** display a help screen and exit, displays extended help on *command* if specified

COMMANDS

Commands tell **halcmd** what to do. Normally **halcmd** reads a single command from the command line and executes it. If the '**-f**' option is used to read commands from a file, **halcmd** reads each line of the file as a new command. Anything following '#' on a line is a comment.

loadrt *modname*

(*load* realtime module) Loads a realtime HAL module called *modname*. **halcmd** looks for the module in a directory specified at compile time.

In systems with realtime, **halcmd** calls the **linuxcnc_module_helper** to load realtime modules. **linuxcnc_module_helper** is a setuid program and is compiled with a whitelist of modules it is allowed to load. This is currently just a list of LinuxCNC-related modules. The **linuxcnc_module_helper** execs insmod, so return codes and error messages are those from insmod. Administrators who wish to restrict which users can load these LinuxCNC-related kernel modules can do this

by setting the permissions and group on **linuxcnc_module_helper** appropriately.

In systems without realtime **halcmd** calls the **rtapi_app** which creates the simulated realtime environment if it did not yet exist, and then loads the requested component with a call to **dlopen(3)**.

unloadrt *modname*

(*unload realtime module*) Unloads a realtime HAL module called *modname*. If *modname* is "all", it will unload all currently loaded realtime HAL modules. **unloadrt** also works by execing **linuxcnc_module_helper** or **rtapi_app**, just like **loadrt**.

loadusr [*flags*] *unix-command*

(*load Userspace component*) Executes the given *unix-command*, usually to load a userspace component. [*flags*] may be one or more of:

- **-W** to wait for the component to become ready. The component is assumed to have the same name as the first argument of the command.
- **-Wn name** to wait for the component, which will have the given name.
- **-w** to wait for the program to exit
- **-i** to ignore the program return value (with -w)

waitusr *name*

(*wait for Userspace component*) Waits for user space component *name* to disconnect from HAL (usually on exit). The component must already be loaded. Usefull near the end of a HAL file to wait until the user closes some user interface component before cleaning up and exiting.

unloadusr *compname*

(*unload Userspace component*) Unloads a userspace component called *compname*. If *compname* is "all", it will unload all userspace components. **unloadusr** works by sending SIGTERM to all userspace components.

unload *compname*

Unloads a userspace component or realtime module. If *compname* is "all", it will unload all userspace components and realtime modules.

newsig *signame type*

(OBSOLETE - use **net** instead) (*new signal*) Creates a new HAL signal called *signame* that may later be used to connect two or more HAL component pins. *type* is the data type of the new signal, and must be one of "bit", "s32", "u32", or "float". Fails if a signal of the same name already exists.

delsig *signame*

(*delete signal*) Deletes HAL signal *signame*. Any pins currently linked to the signal will be unlinked. Fails if *signame* does not exist.

sets *signame value*

(*set signal*) Sets the value of signal *signame* to *value*. Fails if *signame* does not exist, if it already has a writer, or if *value* is not a legal value. Legal values depend on the signals's type.

stype *name*

(*signal type*) Gets the type of signal *name*. Fails if *name* does not exist as a signal.

gets *signame*

(*get signal*) Gets the value of signal *signame*. Fails if *signame* does not exist.

links *pinname [arrow] signame*

(OBSOLETE - use **net** instead) (*link pin to signal*) Establishes a link between a HAL component pin *pinname* and a HAL signal *signame*. Any previous link to *pinname* will be broken. *arrow* can be ">=", "<=", "<=>", or omitted. **halcmd** ignores arrows, but they can be useful in command files to document the direction of data flow. Arrows should not be used on the command line since

the shell might try to interpret them. Fails if either *pinname* or *signame* does not exist, or if they are not the same type type.

linksp *signame* [*arrow*] *pinname*

(OBSOLETE - use **net** instead) (*link* signal to *pin*) Works like **linkps** but reverses the order of the arguments. **halcmd** treats both link commands exactly the same. Use whichever you prefer.

linkpp *pinname1* [*arrow*] *pinname2*

(OBSOLETE - use **net** instead) (*link* *pin* to *pin*) Shortcut for **linkps** that creates the signal (named like the first pin), then links them both to that signal. **halcmd** treats this just as if it were:

halcmd newsig *pinname1*

halcmd linksp *pinname1* *pinname1*

halcmd linksp *pinname1* *pinname2*

net *signame* *pinname* ...

Create *signame* to match the type of *pinname* if it does not yet exist. Then, link *signame* to each *pinname* in turn. Arrows may be used as in **linkps**. When linking a pin to a signal for the first time, the signal value will inherit the pin's default value.

unlinkp *pinname*

(*unlink* *pin*) Breaks any previous link to *pinname*. Fails if *pinname* does not exist. An unlinked pin will retain the last value of the signal it was linked to.

setp *name* *value*

(*set* parameter or *pin*) Sets the value of parameter or pin *name* to *value*. Fails if *name* does not exist as a pin or parameter, if it is a parameter that is not writable, if it is a pin that is an output, if it is a pin that is already attached to a signal, or if *value* is not a legal value. Legal values depend on the type of the pin or parameter. If a pin and a parameter both exist with the given name, the parameter is acted on.

paramname = *value*

pinname = *value*

Identical to **setp**. This alternate form of the command may be more convenient and readable when used in a file.

ptype *name*

(parameter or pin type) Gets the type of parameter or pin *name*. Fails if *name* does not exist as a pin or parameter. If a pin and a parameter both exist with the given name, the parameter is acted on.

getp *name*

(get parameter or pin) Gets the value of parameter or pin *name*. Fails if *name* does not exist as a pin or parameter. If a pin and a parameter both exist with the given name, the parameter is acted on.

addf *funcname* *threadname*

(add function) Adds function *funcname* to realtime thread *threadname*. *funcname* will run after any functions that were previously added to the thread. Fails if either *funcname* or *threadname* does not exist, or if they are incompatible.

delf *funcname* *threadname*

(delete function) Removes function *funcname* from realtime thread *threadname*. Fails if either *funcname* or *threadname* does not exist, or if *funcname* is not currently part of *threadname*.

start Starts execution of realtime threads. Each thread periodically calls all of the functions that were added to it with the **addf** command, in the order in which they were added.

stop Stops execution of realtime threads. The threads will no longer call their functions.

show [item]

Prints HAL items to *stdout* in human readable format. *item* can be one of "**comp**" (components), "**pin**", "**sig**" (signals), "**param**" (parameters), "**funct**" (functions), "**thread**", or "**alias**". The type "**all**" can be used to show matching items of all the preceding types. If *item* is omitted, **show** will print everything.

item This is equivalent to **show all [item]**.

save [item]

Prints HAL items to *stdout* in the form of HAL commands. These commands can be redirected to a file and later executed using **halcmd -f** to restore the saved configuration. *item* can be one of the following: "**comp**" generates a **loadrt** command for realtime component. "**sig**" generates a **newsig** command for each signal, and "**sigu**" generates a **newsig** command for each unlinked signal (for use with **netl** and **netla**). "**link**" and "**linka**" both generate **linkps** commands for each link. (**linka** includes arrows, while **link** does not.) "**net**" and "**neta**" both generate one **newsig** command for each signal, followed by **linksp** commands for each pin linked to that signal. (**neta** includes arrows.) "**netl**" generates one **net** command for each linked signal, and "**netla**" generates a similar command using arrows. "**param**" generates one **setp** command for each parameter. "**thread**" generates one **addf** command for each function in each realtime thread. If *item* is omitted, **save** does the equivalent of **comp**, **sigu**, **link**, **param**, and **thread**.

source filename.hal

Execute the commands from *filename.hal*.

alias type name alias

Assigns "**alias**" as a second name for the pin or parameter "name". For most operations, an alias provides a second name that can be used to refer to a pin or parameter, both the original name and the alias will work.

"**type**" must be **pin** or **param**.

"**name**" must be an existing name or **alias** of the specified type.

unalias type alias

Removes any alias from the pin or parameter alias.

"**type**" must be **pin** or **param**

"**alias**" must be an existing name or **alias** of the specified type.

list type [pattern]

Prints the names of HAL items of the specified type.

'**type**' is '**comp**', '**pin**', '**sig**', '**param**', '**funct**', or '**thread**'. If '**pattern**' is specified it prints only those names that match the pattern, which may be a 'shell glob'.

For '**sig**', '**pin**' and '**param**', the first pattern may be '**-tdatatype**' where datatype is the data type (e.g., 'float') in this case, the listed pins, signals, or parameters are restricted to the given data type

Names are printed on a single line, space separated.

lock [all|tune|none]

Locks HAL to some degree.

none - no locking done.

tune - some tuning is possible (**setp** & such).

all - HAL completely locked.

unlock [all|tune]

Unlocks HAL to some degree.

tune - some tuning is possible (**setp** & such).

all - HAL completely unlocked.

status [*type*]

Prints status info about HAL.
'*type*' is '**lock**', '**mem**', or '**all**'.
If '*type*' is omitted, it assumes '**all**'.

help [*command*]

Give help information for command.
If '*command*' is omitted, list command and brief description

SUBSTITUTION

After a command is read but before it is executed, several types of variable substitution take place.

Environment Variables

Environment variables have the following formats:

\$ENVVAR followed by end-of-line or whitespace

\$(ENVVAR)

Inifile Variables

Inifile variables are available only when an inifile was specified with the halcmd **-i** flag. They have the following formats:

[SECTION]VAR followed by end-of-line or whitespace

[SECTION](VAR)

EXAMPLES**HISTORY****BUGS**

None known at this time.

AUTHOR

Original version by John Kasunich, as part of the LinuxCNC project. Now includes major contributions by several members of the project.

REPORTING BUGS

Report bugs to the [LinuxCNC bug tracker](http://sf.net/p/emc/bugs/) (<http://sf.net/p/emc/bugs/>).

COPYRIGHT

Copyright © 2003 John Kasunich.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO

halrun(1) -- a convenience script to start a realtime environment, process a .hal or a .tcl file, and optionally start an interactive command session using **halcmd** (described here) or **haltcl(1)**.

NAME

halmeter – observe HAL pins, signals, and parameters

SYNOPSIS

halmeter [-s] [**pin|sig|param** *name*] [-g *X-position Y-position [Width]*]

DESCRIPTION

halmeter is used to observe HAL (Hardware Abstraction Layer) pins, signals, or parameters. It serves the same purpose as a multimeter does when working on physical systems.

OPTIONS

pin *name*

display the HAL pin *name*.

sig *name*

display the HAL signal *name*.

param *name*

display the HAL parameter *name*.

If neither **pin**, **sig**, or **param** are specified, the

window starts out blank and the user must select an item to observe.

-s small window. Non-interactive, must be used with **pin**, **sig**, or **param** to select the item to display. The item name is displayed in the title bar instead of the window, and there are no "Select" or "Exit" buttons. Handy when you want a lot of meters in a small space.

-g geometry position. allows one to specify the intial starting position and optionally the width of the meter. Referenced from top left of screen in pixel units. Handy when you want to load a lot of meters in a script with out them displaying on top of each other.

USAGE

Unless **-s** is specified, there are two buttons, "Select" and "Exit". "Select" opens a dialog box to select the item (pin, signal, or parameter) to be observed. "Exit" does what you expect.

The selection dialog has "OK" "Apply", and "Cancel" buttons. OK displays the selected item and closes the dialog. "Apply" displays the selected item but keeps the selection dialog open. "Cancel" closes the dialog without changing the displayed item.

EXAMPLES

halmeter

Opens a meter window, with nothing initially displayed. Use the "Select" button to choose an item to observe. Does not return until the window is closed.

halmeter &

Open a meter window, with nothing initially displayed. Use the "Select" button to choose an item. Runs in the background leaving the shell free for other commands.

halmeter pin parport.0.pin-03-out &

Open a meter window, initially displaying HAL pin *parport.0.pin-03-out*. The "Select" button can be used to display other items. Runs in background.

halmeter -s pin parport.0.pin-03-out &

Open a small meter window, displaying HAL pin *parport.0.pin-03-out*. The displayed item cannot be changed. Runs in background.

halmeter -s pin parport.0.pin-03-out -g 100 500 &

Open a small meter window, displaying HAL pin *parport.0.pin-03-out*. places it 100 pixels to the left and 500 pixels down from top of screen. The displayed item cannot be changed. Runs in background.

halmeter -s pin parport.0.pin-03-out -g 100 500 400 &

Open a small meter window, displaying HAL pin *parport.0.pin-03-out*. places it 100 pixels to the left and 500 pixels down from top of screen. The width will be 400 pixels (270 is default) The displayed item cannot be changed. Runs in background.

SEE ALSO

HISTORY

BUGS

AUTHOR

Original version by John Kasunich, as part of the LinuxCNC project. Improvements by several other members of the LinuxCNC development team.

REPORTING BUGS

Report bugs to jmkasunich AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2003 John Kasunich.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

halrun – manipulate the LinuxCNC HAL from the command line

SYNOPSIS

```
halrun -h  
halrun [-I] [halcmd_opts] [filename[.hal|.tcl]]  
halrun -T [halcmd_opts] [filename[.hal|.tcl]]  
halrun -U
```

DESCRIPTION

halrun is a convenience script used to manipulate the HAL (Hardware Abstraction Layer) from the command line. When invoked, **halrun**:

- Sets up the realtime environment.
 - Executes a command interpreter (**halcmd** or **haltcl**).
 - (Optionally) runs an interactive session.
 - Tears down the realtime environment.
- If no filename is specified, an interactive session is started.
The session will use **halcmd(1)** unless -T is specified in which case **haltcl(1)** will be used.

If a filename is specified and neither the -I nor the -T option is included, the filename will be processed by the command interpreter corresponding to the filename extension (**halcmd** or **haltcl**). After processing, the realtime environment will be torn down.

If a filename is specified and the -I or -T option is included, the file is processed by the appropriate command interpreter and then an interactive session is started for **halcmd** or **haltcl** according to the -I or -T option.

OPTIONS**halcmd_opts**

When a .hal file is specified, the **halcmd_opts** are passed to **halcmd**. See the man page for **halcmd(1)**. When a .tcl file is specified, the only valid options are:

- i infile
 - f filename[.tcl|.hal] (alternate means of specifying a file)
- I** Run an interactive **halcmd** session
 - T** Run an interactive **haltcl** session.
 - U** Forcibly cause the realtime environment to exit. It releases the HAL mutex, requests that all HAL components unload, and stops the realtime system. **-U** must be the only commandline argument.
 - h** display a brief help screen and exit

EXAMPLES**HISTORY****BUGS**

None known at this time.

AUTHOR

Original version by John Kasunich, as part of the LinuxCNC Enhanced Machine Controller project. Now includes major contributions by several members of the project.

REPORTING BUGS

Report bugs to the LinuxCNC bug tracker (URL: <http://sf.net/p/emc/bugs/>).

COPYRIGHT

Copyright © 2003 John Kasunich.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO

halcmd(1), halctl(1)

NAME

halsampler – sample data from HAL in realtime

SYNOPSIS

halsampler [*options*]

DESCRIPTION

sampler(9) and **halsampler** are used together to sample HAL data in real time and store it in a file. **sampler** is a realtime HAL component that exports HAL pins and creates a FIFO in shared memory. It then begins sampling data from the HAL and storing it to the FIFO. **halsampler** is a user space program that copies data from the FIFO to stdout, where it can be redirected to a file or piped to some other program.

OPTIONS

-c CHAN

instructs **halsampler** to read from FIFO *CHAN*. FIFOs are numbered from zero, and the default value is zero, so this option is not needed unless multiple FIFOs have been created.

-n COUNT

instructs **halsampler** to read *COUNT* samples from the FIFO, then exit. If **-n** is not specified, **halsampler** will read continuously until it is killed.

-t instructs **halsampler** to tag each line by printing the sample number in the first column.

FILENAME

instructs **halsampler** to write to **FILENAME** instead of to stdout.

USAGE

A FIFO must first be created by loading **sampler**(9) with **halcmd loadrt** or a **loadrt** command in a .hal file. Then **halsampler** can be invoked to begin printing data from the FIFO to stdout.

Data is printed one line per sample. If **-t** was specified, the sample number is printed first. The data follows, in the order that the pins were defined in the config string. For example, if the **sampler** config string was "ffbs" then a typical line of output (without **-t**) would look like:

```
123.55 33.4 0 -12
```

halsampler prints data as fast as possible until the FIFO is empty, then it retries at regular intervals, until it is either killed or has printed *COUNT* samples as requested by **-n**. Usually, but not always, data printed by **halsampler** will be redirected to a file or piped to some other program.

The FIFO size should be chosen to absorb samples captured during any momentary disruptions in the flow of data, such as disk seeks, terminal scrolling, or the processing limitations of subsequent program in a pipeline. If the FIFO gets full and **sampler** is forced to overwrite old data, **halsampler** will print 'overrun' on a line by itself to mark each gap in the sampled data. If **-t** was specified, gaps in the sequential sample numbers in the first column can be used to determine exactly how many samples were lost.

The data format for **halsampler** output is the same as for **halstreamer**(1) input, so 'waveforms' captured with **halsampler** can be replayed using **halstreamer**. The **-t** option should not be used in this case.

EXIT STATUS

If a problem is encountered during initialization, **halsampler** prints a message to stderr and returns failure.

Upon printing *COUNT* samples (if **-n** was specified) it will shut down and return success. If it is terminated before printing the specified number of samples, it returns failure. This means that when **-n** is not specified, it will always return failure when terminated.

SEE ALSO

sampler(9) **streamer**(9) **halstreamer**(1)

HISTORY**BUGS****AUTHOR**

Original version by John Kasunich, as part of the LinuxCNC project. Improvements by several other members of the LinuxCNC development team.

REPORTING BUGS

Report bugs to jmkasunich AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2006 John Kasunich.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

halstreamer – stream file data into HAL in real time

SYNOPSIS

halstreamer [*options*]

DESCRIPTION

streamer(9) and **halstreamer** are used together to stream data from a file into the HAL in real time. **streamer** is a realtime HAL component that exports HAL pins and creates a FIFO in shared memory. **hal_streamer** is a user space program that copies data from stdin into the FIFO, so that **streamer** can write it to the HAL pins.

OPTIONS

-c CHAN

instructs **halstreamer** to write to FIFO *CHAN*. FIFOs are numbered from zero, and the default value is zero, so this option is not needed unless multiple FIFOs have been created.

FILENAME

instructs **halsampler** to read from **FILENAME** instead of from stdin.

USAGE

A FIFO must first be created by loading **streamer**(9) with **halcmd loadrt** or a **loadrt** command in a .hal file. Then **halstreamer** can be invoked to begin writing data into the FIFO.

Data is read from stdin, and is almost always either redirected from a file or piped from some other program, since keyboard input would be unable to keep up with even slow streaming rates.

Each line of input must match the pins that are attached to the FIFO, for example, if the **streamer** config string was "ffbs" then each line of input must consist of two floats, a bit, and a signed integer, in that order and separated by whitespace. Floats must be formatted as required by **strtod**(3), signed and unsigned integers must be formatted as required by **strtol**(3) and **strtoul**(3), and bits must be either '0' or '1'.

halstreamer transfers data to the FIFO as fast as possible until the FIFO is full, then it retries at regular intervals, until it is either killed or reads **EOF** from stdin. Data can be redirected from a file or piped from some other program.

The FIFO size should be chosen to ride through any momentary disruptions in the flow of data, such as disk seeks. If the FIFO is big enough, **halstreamer** can be restarted with the same or a new file before the FIFO empties, resulting in a continuous stream of data.

The data format for **halstreamer** input is the same as for **halsampler**(1) output, so 'waveforms' captured with **halsampler** can be replayed using **halstreamer**.

EXIT STATUS

If a problem is encountered during initialization, **halstreamer** prints a message to stderr and returns failure.

If a badly formatted line is encountered while writing to the FIFO, it prints a message to stderr, skips the line, and continues (this behavior may be revised in the future).

Upon reading **EOF** from the input, it returns success. If it is terminated before the input ends, it returns failure.

SEE ALSO

streamer(9) **sampler**(9) **halsampler**(1)

HISTORY

BUGS**AUTHOR**

Original version by John Kasunich, as part of the LinuxCNC project. Improvements by several other members of the LinuxCNC development team.

REPORTING BUGS

Report bugs to jmkasunich AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2006 John Kasunich.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

haltcl – manipulate the LinuxCNC HAL from the command line using a tcl interpreter.

SYNOPSIS

haltcl [-i *infile*] [*filename*]

DESCRIPTION

haltcl is used to manipulate the HAL (Hardware Abstraction Layer) from the command line using a tcl interpreter. **haltcl** can optionally read commands from a file (*filename*), allowing complex HAL configurations to be set up with a single command.

OPTIONS

-i *infile*

If specified, the *infile* is read and used to create tcl global variable arrays. An array is created for each SECTION of the *infile* with elements for each ITEM in the section.

For example, if the *infile* contains:

```
[SECTION_A]ITEM_1 = 1
[SECTION_A]ITEM_2 = 2
[SECTION_B]ITEM_1 = 10
```

The corresponding tcl variables are:

```
SECTION_A(ITEM_1) = 1
SECTION_A(ITEM_2) = 2
SECTION_B(ITEM_1) = 10
```

-ini *infile* -- declining usage, use **-i *infile***

filename

If specified, the tcl commands of **filename** are executed. If no *filename* is specified, **haltcl** opens an interactive session.

COMMANDS

haltcl includes the commands of a tcl interpreter augmented with commands for the hal language as described for **halcmd(1)**. The augmented commands can be listed with the command:

```
haltcl: hal --commands
```

```
addf alias delf delsig getp gets ptype stype help linkpp linkps linksp list loadrt loadusr lock net newsig
save setexact_for_test_suite_only setp sets show source start status stop unalias unlinkp unload unloadrt
unloadusr unlock waitusr
```

Two of the augmented commands, 'list' and 'gets', require special treatment to avoid conflict with tcl built-in commands having the same names. To use these commands, precede them with the keyword 'hal':

```
hal list
hal gets
```

REPORTING BUGS

Report bugs to the LinuxCNC bug tracker (URL: <http://sf.net/p/emc/bugs/>).

COPYRIGHT

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO

halcmd(1), halrun(1)

NAME

halui – observe HAL pins and command LinuxCNC through NML

SYNOPSIS

halui [-ini <path-to-ini>]

DESCRIPTION

halui is used to build a User Interface using hardware knobs and switches. It exports a big number of pins, and acts accordingly when these change.

OPTIONS

-ini name

use the *name* as the configuration file. Note: halui must find the nml file specified in the ini, usually that file is in the same folder as the ini, so it makes sense to run halui from that folder.

USAGE

When run, **halui** will export a large number of pins. A user can connect those to his physical knobs & switches & leds, and when a change is noticed halui triggers an appropriate event.

halui expects the signals to be debounced, so if needed (bad knob contact) connect the physical button to a HAL debounce filter first.

PINS

abort

halui.abort bit in
pin for clearing most errors

tool

halui.tool.length-offset.a float out
current applied tool length offset for the A axis

halui.tool.length-offset.b float out
current applied tool length offset for the B axis

halui.tool.length-offset.c float out
current applied tool length offset for the C axis

halui.tool.length-offset.u float out
current applied tool length offset for the U axis

halui.tool.length-offset.v float out
current applied tool length offset for the V axis

halui.tool.length-offset.w float out
current applied tool length offset for the W axis

halui.tool.length-offset.x float out
current applied tool length offset for the X axis

halui.tool.length-offset.y float out
current applied tool length offset for the Y axis

halui.tool.length-offset.z float out
current applied tool length offset for the Z axis

halui.tool.number u32 out
current selected tool

spindle

halui.spindle.brake-is-on bit out
 status pin that tells us if brake is on

halui.spindle.brake-off bit in
 pin for deactivating the spindle brake

halui.spindle.brake-on bit in
 pin for activating the spindle brake

halui.spindle.decrease bit in
 a rising edge on this pin decreases the current spindle speed by 100

halui.spindle.forward bit in
 a rising edge on this pin makes the spindle go forward

halui.spindle.increase bit in
 a rising edge on this pin increases the current spindle speed by 100

halui.spindle.is-on bit out
 status pin telling if the spindle is on

halui.spindle.reverse bit in
 a rising edge on this pin makes the spindle go reverse

halui.spindle.runs-backward bit out
 status pin telling if the spindle is running backward

halui.spindle.runs-forward bit out
 status pin telling if the spindle is running forward

halui.spindle.start bit in
 a rising edge on this pin starts the spindle

halui.spindle.stop bit in
 a rising edge on this pin stops the spindle

spindle override

halui.spindle-override.count-enable bit in (default: **TRUE**)
 When TRUE, modify spindle override when counts changes.

halui.spindle-override.counts s32 in
 counts X scale = spindle override percentage

halui.spindle-override.decrease bit in
 pin for decreasing the SO (-=scale)

halui.spindle-override.direct-value bit in
 pin to enable direct spindle override value input

halui.spindle-override.increase bit in
 pin for increasing the SO (+=scale)

halui.spindle-override.scale float in
 pin for setting the scale of counts for SO

halui.spindle-override.value float out
 current FO value

program

halui.program.block-delete.is-on bit out
 status pin telling that block delete is on

halui.program.block-delete.off bit in
 pin for requesting that block delete is off

halui.program.block-delete.on bit in
 pin for requesting that block delete is on

halui.program.is-idle bit out
 status pin telling that no program is running

halui.program.is-paused bit out
 status pin telling that a program is paused

halui.program.is-running bit out
 status pin telling that a program is running

halui.program.optional-stop.is-on bit out
 status pin telling that the optional stop is on

halui.program.optional-stop.off bit in
 pin requesting that the optional stop is off

halui.program.optional-stop.on bit in
 pin requesting that the optional stop is on

halui.program.pause bit in
 pin for pausing a program

halui.program.resume bit in
 pin for resuming a program

halui.program.run bit in
 pin for running a program

halui.program.step bit in
 pin for stepping in a program

halui.program.stop bit in
 pin for stopping a program (note: this pin does the same thing as halui.abort)

mode

halui.mode.auto bit in
 pin for requesting auto mode

halui.mode.is-auto bit out
 pin for auto mode is on

halui.mode.is-joint bit out
 pin showing joint by joint jog mode is on

halui.mode.is-manual bit out
 pin for manual mode is on

halui.mode.is-mdi bit out
 pin for mdi mode is on

halui.mode.is-teleop bit out
 pin showing coordinated jog mode is on

halui.mode.joint bit in
 pin for requesting joint by joint jog mode

halui.mode.manual bit in
 pin for requesting manual mode

halui.mode.mdi bit in
 pin for requesting mdi mode

halui.mode.teleop bit in
 pin for requesting coordinated jog mode

mdi (optional)

halui.mdi-command-XX bit in

halui looks for ini variables named [HALUI]MDI_COMMAND, and exports a pin for each command it finds. When the pin is driven TRUE, **halui** runs the specified MDI command. XX is a two digit number starting at 00. If no [HALUI]MDI_COMMAND variables are set in the ini file, no halui.mdi-command-XX pins will be exported by halui.

mist

halui.mist.is-on bit out
 pin for mist is on

halui.mist.off bit in
 pin for stopping mist

halui.mist.on bit in
 pin for starting mist

max-velocity

halui.max-velocity.count-enable bit in (default: **TRUE**)

When TRUE, modify max velocity when counts changes.

halui.max-velocity.counts s32 in
 counts from an encoder for example to change maximum velocity

halui.max-velocity.decrease bit in
 pin for decreasing the maximum velocity (-=scale)

halui.max-velocity.direct-value bit in
 pin for using a direct value for max velocity

halui.max-velocity.increase bit in
 pin for increasing the maximum velocity (+=scale)

halui.max-velocity.scale float in
 pin for setting the scale on changing the maximum velocity

halui.max-velocity.value float out
 Current value for maximum velocity

machine

halui.machine.is-on bit out
 pin for machine is On/Off

halui.machine.off bit in
 pin for setting machine Off

halui.machine.on bit in
 pin for setting machine On

lube

halui.lube.is-on bit out
 pin for lube is on

halui.lube.off bit in
 pin for stopping lube

halui.lube.on bit in
 pin for starting lube

joint

halui.joint.N.has-fault bit out
 status pin telling that joint N has a fault

halui.joint.N.home bit in
 pin for homing joint N

halui.joint.N.is-homed bit out
 status pin telling that joint N is homed

halui.joint.N.is-selected bit out
 status pin that joint N is selected

halui.joint.N.on-hard-max-limit bit out
 status pin telling that joint N is on the positive hardware limit

halui.joint.N.on-hard-min-limit bit out
 status pin telling that joint N is on the negative hardware limit

halui.joint.N.on-soft-max-limit bit out
 status pin telling that joint N is on the positive software limit

halui.joint.N.on-soft-min-limit bit out
 status pin telling that joint N is on the negative software limit

halui.joint.N.select bit in
 pin for selecting joint N

halui.joint.N.unhome bit in
 pin for unhoming joint N

halui.joint.selected u32 out
 selected joint

halui.joint.selected.has-fault bit out
 status pin selected joint is faulted

halui.joint.selected.home bit in
 pin for homing the selected joint

halui.joint.selected.is-homed bit out
 status pin telling that the selected joint is homed

halui.joint.selected.on-hard-max-limit bit out
 status pin telling that the selected joint is on the positive hardware limit

halui.joint.selected.on-hard-min-limit bit out
 status pin telling that the selected joint is on the negative hardware limit

halui.joint.selected.on-soft-max-limit bit out
 status pin telling that the selected joint is on the positive software limit

halui.joint.selected.on-soft-min-limit bit out
 status pin telling that the selected joint is on the negative software limit

halui.joint.selected.unhome bit in
 pin for unhoming the selected joint

jog

halui.jog.deadband float in
 pin for setting jog analog deadband (jog analog inputs smaller/slower than this are ignored)

halui.jog-speed float in
 pin for setting jog speed for plus/minus jogging.

halui.jog.N.analog float in
 pin for jogging the axis N using an float value (e.g. joystick)

halui.jog.N.increment float in
 pin for setting the jog increment for axis N when using increment-plus/minus

halui.jog.N.increment-minus bit in
 a rising edge will make axis N jog in the negative direction by the increment amount

halui.jog.N.increment-plus bit in
 a rising edge will make axis N jog in the positive direction by the increment amount

halui.jog.N.minus bit in
 pin for jogging axis N in negative direction at the halui.jog-speed velocity

halui.jog.N.plus bit in
 pin for jogging axis N in positive direction at the halui.jog-speed velocity

halui.jog.selected.increment float in
 pin for setting the jog increment for the selected axis when using increment-plus/minus

halui.jog.selected.increment-minus bit in
 a rising edge will make the selected axis jog in the negative direction by the increment amount

halui.jog.selected.increment-plus bit in
 a rising edge will make the selected axis jog in the positive direction by the increment amount

halui.jog.selected.minus bit in
 pin for jogging the selected axis in negative direction at the halui.jog-speed velocity

halui.jog.selected.plus
 pin for jogging the selected axis bit in in positive direction at the halui.jog-speed velocity

flood

halui.flood.is-on bit out
 pin for flood is on

halui.flood.off bit in
 pin for stopping flood

halui.flood.on bit in
 pin for starting flood

feed override

halui.feed-override.count-enable bit in (default: **TRUE**)
 When TRUE, modify feed override when counts changes.

halui.feed-override.counts s32 in
 counts X scale = feed override percentage

halui.feed-override.decrease bit in
 pin for decreasing the FO (-=scale)

halui.feed-override.direct-value bit in
 pin to enable direct value feed override input

halui.feed-override.increase bit in
 pin for increasing the FO (+=scale)

halui.feed-override.scale float in
 pin for setting the scale on changing the FO

halui.feed-override.value float out
 current Feed Override value

rapid override

halui.rapid-override.count-enable bit in (default: **TRUE**)
 When TRUE, modify Rapid Override when counts changes.

halui.rapid-override.counts s32 in
 counts X scale = Rapid Override percentage

halui.rapid-override.decrease bit in
 pin for decreasing the Rapid Override (-=scale)

halui.rapid-override.direct-value bit in
 pin to enable direct value Rapid Override input

halui.rapid-override.increase bit in
 pin for increasing the Rapid Override (+=scale)

halui.rapid-override.scale float in
 pin for setting the scale on changing the Rapid Override

halui.rapid-override.value float out
 current Rapid Override value

estop

halui.estop.activate bit in
 pin for setting Estop (LinuxCNC internal) On

halui.estop.is-activated bit out
 pin for displaying Estop state (LinuxCNC internal) On/Off

halui.estop.reset bit in
 pin for resetting Estop (LinuxCNC internal) Off

axis

halui.axis.N.pos-commanded float out float out
 Commanded axis position in machine coordinates

halui.axis.N.pos-feedback float out float out
 Feedback axis position in machine coordinates

halui.axis.N.pos-relative float out float out
 Commanded axis position in relative coordinates

home

halui.home-all bit in
 pin for requesting home-all (only available when a valid homing sequence is specified)

SEE ALSO

HISTORY

BUGS

none known at this time.

AUTHOR

Written by Alex Joni, as part of the LinuxCNC project. Updated by John Thornton

REPORTING BUGS

Report bugs to alex_joni AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2006 Alex Joni.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

iocontrol – accepts NML I/O commands, interacts with HAL in userspace

SYNOPSIS

loadusr io [-ini *infile*]

DESCRIPTION

These pins are created by the userspace IO controller, usually found in \$LINUXCNC_HOME/bin/io

The signals are turned on and off in userspace - if you have strict timing requirements or simply need more i/o, consider using the realtime synchronized i/o provided by **motion(9)** instead.

The infile is searched for in the directory from which halcmd was run, unless an absolute path is specified.

PINS**iocontrol.0.coolant-flood**

(Bit, Out) TRUE when flood coolant is requested

iocontrol.0.coolant-mist

(Bit, Out) TRUE when mist coolant is requested

iocontrol.0.emc-enable-in

(Bit, In) Should be driven FALSE when an external estop condition exists.

iocontrol.0.lube

(Bit, Out) TRUE when lube is requested

iocontrol.0.lube_level

(Bit, In) Should be driven FALSE when lubrication tank is empty.

iocontrol.0.tool-change

(Bit, Out) TRUE when a tool change is requested

iocontrol.0.tool-changed

(Bit, In) Should be driven TRUE when a tool change is completed.

iocontrol.0.tool-number

(s32, Out) Current tool number

iocontrol.0.tool-prep-number

(s32, Out) The number of the next tool, from the RS274NGC T-word

iocontrol.0.tool-prep-pocket

(s32, Out) The pocket number (location in tool storage mechanism) of the next tool, as described in the tool table

iocontrol.0.tool-prepare

(Bit, Out) TRUE when a *Tn* tool prepare is requested

iocontrol.0.tool-prepared

(Bit, In) Should be driven TRUE when a tool prepare is completed.

iocontrol.0.user-enable-out

(Bit, Out) FALSE when an internal estop condition exists

iocontrol.0.user-request-enable

(Bit, Out) TRUE when the user has requested that estop be cleared

SEE ALSO

motion(9)

NAME

linuxcncrsh – text-mode interface for commanding LinuxCNC over the network

SYNOPSIS

linuxcncrsh [OPTIONS] [-- LINUXCNC_OPTIONS]

DESCRIPTION

linuxcncrsh is a user interface for LinuxCNC. Instead of popping up a GUI window like axis(1) and touchy(1) do, it processes text-mode commands that it receives via the network. A human (or a program) can interface with **linuxcncrsh** using telnet(1) or nc(1) or similar programs.

All features of LinuxCNC are available via the **linuxcncrsh** interface.

OPTIONS

-p,--port PORT_NUMBER

Specify the port for linuxcncrsh to listen on. Defaults to 5007 if omitted.

-n,--name SERVER_NAME

Sets the server name that linuxcncrsh will use to identify itself during handshaking with a new client. Defaults to EMCNETSVR if omitted.

-w,--connectpw PASSWORD

Specify the connection password to use during handshaking with a new client. Note that the password is sent in the clear, so it can be read by anyone who can read packets on the network between the server and the client. Defaults to EMC if omitted.

-e,--enablepw PASSWORD

Specify the password required to enable LinuxCNC via linuxcncrsh. Note that the password is sent in the clear, so it can be read by anyone who can read packets on the network between the server and the client. Defaults to EMCTOO if omitted.

-s,--sessions MAX_SESSIONS

Specify the maximum number of simultaneous connections. Defaults to -1 (no limit) if not specified.

In addition to the options listed above, linuxcncrsh accepts an optional special LINUXCNC_OPTION at the end:

-ini LINUXCNC_INI_FILE

LinuxCNC .ini file to use. The -ini option **must** be preceded by two dashes: "--". Defaults to emc.ini if omitted.

Starting linuxcncrsh

To use linuxcncrsh instead of a normal LinuxCNC GUI like axis or touch, specify it in your .ini file like this:

[DISPLAY]

DISPLAY=linuxcncrsh

To use linuxcncrsh in addition to a normal GUI, you can either start it at the end of your .hal file, or run it by hand in a terminal window.

To start it from hal, add a line like this to the end of your .hal file:

loadusr linuxcncrsh [OPTIONS] [-- LINUXCNC_OPTIONS]

To start it from the terminal, run linuxcncrsh manually like this:

linuxcncrsh [OPTIONS] [-- LINUXCNC_OPTIONS]

Connecting

Once LinuxCNC is up and linuxcncrsh is running, you can connect to it using **telnet** or **nc** or similar:

telnet HOST PORT

HOST is the hostname or IP address of the computer running linuxcncrsh, and PORT is

the port it's listening on (5007 if you did not give linuxcncrsh the --port option).

Network protocol

linuxcncrsh accepts TCP connections on the port specified by the --port option, or 5007 if not specified.

The client sends requests, and the linuxcncrsh server returns replies. Requests consist of a command word followed by optional command-specific parameters. Requests and most request parameters are case insensitive. The exceptions are passwords, file paths and text strings.

Requests to linuxcncrsh are terminated with line endings, any combination of one or more '\r' and '\n' characters. Replies from linuxcncrsh are terminated with the sequence '\r\n'.

The supported commands are as follows:

hello <password> <client> <version>

<password> must match linuxcncrsh's connect password, or "EMC" if no --connectpw was supplied. The three arguments may not contain whitespace. If a valid password was entered the server will respond with:

HELLO ACK <ServerName> <ServerVersion>

If an invalid password or any other syntax error occurs then the server responds with:

HELLO NAK

get <subcommand> [<parameters>]

The get command takes one of the LinuxCNC sub-commands (described in the section **Linux-CNC Subcommands**, below) and zero or more additional subcommand-specific parameters.

set <subcommand> <parameters>

The set command takes one of the LinuxCNC sub-commands (described in the section **Linux-CNC Subcommands**, below) and one or more additional parameters.

quit

The quit command disconnects the associated socket connection.

shutdown

The shutdown command tells LinuxCNC to shutdown and disconnect the session. This command may only be issued if the Hello has been successfully negotiated and the connection has control of the CNC (see **enable** subcommand in the **LinuxCNC Subcommands** section, below).

help

The help command will return help information in text format over the connection. If no parameters are specified, it will itemize the available commands. If a command is specified, it will provide usage information for the specified command. Help will respond regardless of whether a "Hello" has been successfully negotiated.

LinuxCNC Subcommands

Subcommands for **get** and **set** are:

echo {on|off}

With get, any on/off parameter is ignored and the current echo state is returned. With set, sets the echo state as specified. Echo defaults to on when the connection is first established. When echo is on, all commands will be echoed upon receipt. This state is local to each connection.

verbose {on|off}

With get, any on/off parameter is ignored and the current verbose state is returned. With set, sets the verbose state as specified. When verbose mode is on, all set commands return positive acknowledgement in the form SET <COMMAND> ACK, and text error messages will be issued (FIXME: I don't know what this means). The verbose state is local to each connection, and starts out OFF on new connections.

enable {<passwd>}|off

The session's enable state indicates whether the current connection is enabled to perform control functions. With get, any parameter is ignored, and the current enable state is returned. With set

and a valid password matching linuxcncrsh's --enablepw (EMCTOO if not specified), the current connection is enabled for control functions. "OFF" may not be used as a password and disables control functions for this connection.

config [TBD]

Unused, ignore for now.

comm_mode {ascii|binary}

With get, any parameter is ignored and the current communications mode is returned. With set, will set the communications mode to the specified mode. The ascii mode is the text request/reply mode, the binary protocol is not currently designed or implemented.

comm_prot <version>

With get, any parameter is ignored and the current protocol version used by the server is returned. With set, sets the server to use the specified protocol version, provided it is lower than or equal to the highest version number supported by the server implementation.

infile

Not currently implemented! With get, returns the string "emc.ini". Should return the full path and file name of the current configuration infile. Setting this does nothing.

plat

With get, returns the string "Linux".

ini <var> <section>

Not currently implemented, do not use! Should return the string value of <var> in section <section> of the ini file.

debug <value>

With get, any parameter is ignored and the current integer value of EMC_DEBUG is returned. Note that the value of EMC_DEBUG returned is from the UI's ini file, which may be different than emc's ini file. With set, sends a command to the EMC to set the new debug level, and sets the EMC_DEBUG global here to the same value. This will make the two values the same, since they really ought to be the same.

set_wait {none|received|done}

The set_wait setting controls the wait after receiving a command. It can be "none" (return right away), "received" (after the command was sent and received), or "done" (after the command was done). With get, any parameter is ignored and the current set_wait setting is returned. With set, set the set_wait setting to the specified value.

wait {received|done}

With set, force a wait for the previous command to be received, or done. This lets you wait in the event that "set_wait none" is in effect.

set_timeout <timeout>

With set, set the timeout for commands to return to <timeout> seconds. Timeout is a real number. If it's <= 0.0, it means wait forever. Default is 0.0, wait forever.

update {none|auto}

The update mode controls whether to return fresh or stale values for "get" requests. When the update mode is "none" it returns stale values, when it's "auto" it returns fresh values. Defaults to "auto" for new connections. Set this to "none" if you like to be confused.

error

With get, returns the current error string, or "ok" if no error.

operator_display

With get, returns the current operator display string, or "ok" if none.

operator_text

With get, returns the current operator text string, or "ok" if none.

time

With get, returns the time, in seconds, from the start of the epoch. This starting time depends on the platform.

estop {on|off}

With get, ignores any parameters and returns the current estop setting as "on" or "off". With set, sets the estop as specified. Estop "on" means the machine is in the estop state and won't run.

machine {on|off}

With get, ignores any parameters and returns the current machine power setting as "on" or "off". With set, sets the machine on or off as specified.

mode {manual|auto|mdi}

With get, ignores any parameters and returns the current machine mode. With set, sets the machine mode as specified.

mist {on|off}

With get, ignores any parameters and returns the current mist coolant setting. With set, sets the mist setting as specified.

flood {on|off}

With get, ignores any parameters and returns the current flood coolant setting. With set, sets the flood setting as specified.

lube {on|off}

With get, ignores any parameters and returns the current lube pump setting. With set, sets the lube pump setting as specified.

lube_level

With get, returns the lubricant level sensor reading as "ok" or "low". With set, mocks you for wishful thinking.

spindle {forward|reverse|increase|decrease|constant|off}

With get, any parameter is ignored and the current spindle state is returned as "forward", "reverse", "increase", "decrease", or "off". With set, sets the spindle as specified. Note that "increase" and "decrease" will cause a speed change in the corresponding direction until a "constant" command is sent.

brake {on|off}

With get, any parameter is ignored and the current brake setting is returned. With set, the brake is set as specified.

tool

With get, returns the id of the currently loaded tool.

tool_offset

With get, returns the currently applied tool length offset.

load_tool_table <file>

With set, loads the tool table specified by <file>.

home {0|1|2|...}

With set, homes the indicated axis.

jog_stop {0|1|2|...}

With set, stop any in-progress jog on the specified axis.

jog {0|1|2|...} <speed>

With set, jog the specified axis at <speed>; sign of speed is direction.

jog_incr {0|1|2|...} <speed> <incr>

With set, jog the indicated axis by increment <incr> at the <speed>; sign of speed is direction.

feed_override <percent>

With get, any parameter is ignored and the current feed override is returns (as a percentage of

commanded feed). With set, sets the feed override as specified.

spindle_override <percent>

With get, any parameter is ignored and the current spindle override is returned (as a percentage of commanded speed). With set, sets the spindle override as specified.

abs_cmd_pos [{0|1|...}]

With get, returns the specified axis' commanded position in absolute coordinates. If no axis is specified, returns all axes' commanded absolute position.

abs_act_pos [{0|1|...}]

With get, returns the specified axis' actual position in absolute coordinates. If no axis is specified, returns all axes' actual absolute position.

rel_cmd_pos [{0|1|...}]

With get, returns the specified axis' commanded position in relative coordinates, including tool length offset. If no axis is specified, returns all axes' commanded relative position.

rel_act_pos [{0|1|...}]

With get, returns the specified axis' actual position in relative coordinates, including tool length offset. If no axis is specified, returns all axes' actual relative position.

joint_pos [{0|1|...}]

With get, returns the specified joint's actual position in absolute coordinates, excluding tool length offset. If no joint is specified, returns all joints' actual absolute position.

pos_offset [{X|Y|Z|R|P|W}]

With get, returns the position offset associated with the world coordinate provided.

joint_limit [{0|1|...}]

With get, returns limit status of the specified joint as "ok", "minsoft", "minhard", "maxsoft", or "maxhard". If no joint number is specified, returns the limit status of all joints.

joint_fault [{0|1|...}]

With get, returns the fault status of the specified joint as "ok" or "fault". If no joint number is specified, returns the fault status of all joints.

joint_homed [{0|1|...}]

With get, returns the homed status of the specified joint as "homed" or "not". If no joint number is specified, returns the homed status of all joints.

mdi <string>

With set, sends <string> as an MDI command.

task_plan_init

With set, initializes the program interpreter.

open <filename>

With set, opens the named file. The <filename> is opened by linuxcnc, so it should either be an absolute path or a relative path starting in the linuxcnc working directory (the directory of the active .ini file). Note that linuxcnc can only have one file open at a time, and it's up to the UI (linuxcncrsh or similar) to close any open file before opening a new file. linuxcncrsh currently does not support closing files, which rather limits the utility of this command.

run [<StartLine>]

With set, runs the opened program. If no StartLine is specified, runs from the beginning. If a StartLine is specified, start line, runs from that line. A start line of -1 runs in verify mode.

pause

With set, pause program execution.

resume

With set, resume program execution.

abort

With set, abort program or MDI execution.

step

With set, step the program one line.

program

With get, returns the name of the currently opened program, or "none".

program_line

With get, returns the currently executing line of the program.

program_status

With get, returns "idle", "running", or "paused".

program_codes

With get, returns the string for the currently active program codes.

joint_type [<joint>]

With get, returns "linear", "angular", or "custom" for the type of the specified joint (or for all joints if none is specified).

joint_units [<joint>]

With get, returns "inch", "mm", "cm", or "deg", "rad", "grad", or "custom", for the corresponding native units of the specified joint (or for all joints if none is specified). The type of the axis (linear or angular) is used to resolve which type of units are returned. The units are obtained heuristically, based on the EMC_AXIS_STAT::units numerical value of user units per mm or deg. For linear joints, something close to 0.03937 is deemed "inch", 1.000 is "mm", 0.1 is "cm", otherwise it's "custom". For angular joints, something close to 1.000 is deemed "deg", PI/180 is "rad", 100/90 is "grad", otherwise it's "custom".

program_units

Synonym for program_linear_units.

program_linear_units

With get, returns "inch", "mm", "cm", or "none", for the corresponding linear units that are active in the program interpreter.

program_angular_units

With get, returns "deg", "rad", "grad", or "none" for the corresponding angular units that are active in the program interpreter.

user_linear_units

With get, returns "inch", "mm", "cm", or "custom", for the corresponding native user linear units of the LinuxCNC trajectory level. This is obtained heuristically, based on the EMC_TRAJ_STAT::linearUnits numerical value of user units per mm. Something close to 0.03937 is deemed "inch", 1.000 is "mm", 0.1 is "cm", otherwise it's "custom".

user_angular_units

Returns "deg", "rad", "grad", or "custom" for the corresponding native user angular units of the LinuxCNC trajectory level. Like with linear units, this is obtained heuristically.

display_linear_units

With get, returns "inch", "mm", "cm", or "custom", for the linear units that are active in the display. This is effectively the value of linearUnitConversion.

display_angular_units

With get, returns "deg", "rad", "grad", or "custom", for the angular units that are active in the display. This is effectively the value of angularUnitConversion.

linear_unit_conversion {inch|mm|cm|auto}

With get, any parameter is ignored and the active unit conversion is returned. With set, sets the unit to be displayed. If it's "auto", the units to be displayed match the program units.

angular_unit_conversion {deg|rad|grad|auto}

With get, any parameter is ignored and the active unit conversion is returned. With set, sets the units to be displayed. If it's "auto", the units to be displayed match the program units.

probe_clear

With set, clear the probe tripped flag.

probe_tripped

With get, return the probe state - has the probe tripped since the last clear?

probe_value

With get, return the current value of the probe signal.

probe

With set, move toward a certain location. If the probe is tripped on the way stop motion, record the position and raise the probe tripped flag.

teleop_enable {on|off}

With get, any parameter is ignored and the current teleop mode is returned. With set, sets the teleop mode as specified.

kinematics_type

With get, returns the type of kinematics functions used (identity=1, serial=2, parallel=3, custom=4).

override_limits {on|off}

With get, any parameter is ignored and the override_limits setting is returned. With set, the override_limits parameter is set as specified. If override_limits is on, disables end of travel hardware limits to allow jogging off of a limit. If parameters is off, then hardware limits are enabled.

optional_stop {0|1}

With get, any parameter is ignored and the current "optional stop on M1" setting is returned. With set, the setting is set as specified.

Example Session

This section shows an example session. Bold items are typed by you, non-bold is machine output.

The user connects to linuxcncrsh, handshakes with the server (hello), enables machine commanding from this session (set enable), brings the machine out of estop (set estop off) and turns it on (set machine on), homes all the axes, switches the machine to mdi mode, sends an MDI g-code command, then disconnects and shuts down LinuxCNC.

```
> telnet localhost 5007
Trying 127.0.0.1...
Connected to 127.0.0.1
Escape character is '^]'.
hello EMC user-typing-at-telnet 1.0
HELLO ACK EMCNETSVR 1.1
set enable EMCTOO
set enable EMCTOO
set mode manual
set mode manual
set estop off
set estop off
set machine on
set machine on
set home 0
set home 0
set home 1
set home 1
set home 2
set home 2
```

```
set mode mdi
set mode mdi
set mdi g0x1
set mdi g0x1
shutdown
shutdown
Connection closed by foreign host.
```

NAME

milltask – Userspace task controller for LinuxCNC

DESCRIPTION

milltask is an internal process of LinuxCNC. It is generally not invoked directly. It creates the pins shown as owned by the "inihal" component, which allow runtime modification of certain values from the ini file.

PINS**Per-axis pins****ini.#.backlash**

Allows adjustment of [AXIS_#]BACKLASH

ini.#.max_acceleration

Allows adjustment of [AXIS_#]MAX_ACCELERATION

ini.#.max_velocity

Allows adjustment of [AXIS_#]MAX_VELOCITY

ini.#.max_limit

Allows adjustment of [AXIS_#]MAX_LIMIT

ini.#.min_limit

Allows adjustment of [AXIS_#]MIN_LIMIT

ini.#.ferror

Allows adjustment of [AXIS_#]FERROR

ini.#.min_ferror

Allows adjustment of [AXIS_#]MIN_FERROR

Global pins**ini.traj_default_acceleration**

Allows adjustment of [TRAJ]DEFAULT_ACCELERATION

ini.traj_default_velocity

Allows adjustment of [TRAJ]DEFAULT_VELOCITY

ini.traj_max_acceleration

Allows adjustment of [TRAJ]MAX_ACCELERATION

ini.traj_max_velocity

Allows adjustment of [TRAJ]MAX_VELOCITY

BUGS

These pins cannot be linked or set in a halfile specified by [HAL]HALFILE. In GUIs that support the feature, they can be set by the [HAL]POSTGUI_HALFILE.

The ini file is not automatically updated with these values.

NAME

pyvcp – Virtual Control Panel for LinuxCNC

SYNOPSIS

pyvcp [-g *WxH+X+Y*] [-c *component-name*] *myfile.xml*

OPTIONS

-g *WxH+X+Y*

This sets the initial geometry of the root window. Use 'WxH' for just size, '+X+Y' for just position, or 'WxH+X+Y' for both. Size / position use pixel units. Position is referenced from top left.

-c *component-name*

Use *component-name* as the HAL component name. If the component name is not specified, the basename of the xml file is used.

SEE ALSO

Python Virtual Control Panel in the LinuxCNC documentation for a description of the xml syntax, along with examples

NAME

`shuttleexpress` – control HAL pins with the ShuttleXpress device made by Contour Design

SYNOPSIS

loadusr shuttleexpress [DEVICE ...]

DESCRIPTION

`shuttleexpress` is a userspace HAL component that interfaces Contour Design's ShuttleXpress device with LinuxCNC's HAL. The ShuttleXpress has five momentary buttons, a 10 counts/revolution jog wheel with detents, and a 15-position spring-loaded outer wheel that returns to center when released.

If it is started without command-line arguments, it will probe all `/dev/hidraw*` device files for ShuttleXpress devices, and use all devices found. If it is started with command-line arguments, only will only probe the devices specified.

UDEV

The `shuttleexpress` module needs read permission on the `/dev/hidraw*` device files. This can be accomplished by adding a file `/etc/udev/rules.d/99-shuttleexpress.rules`, with the following contents:

```
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="0b33", ATTRS{idProduct}=="0020", MODE=="0444"
```

A warning about the Jog Wheel

The ShuttleXpress device has an internal 8-bit counter for the current jog-wheel position. The `shuttleexpress` driver can not know this value until the ShuttleXpress device sends its first event. When the first event comes into the driver, the driver uses the device's reported jog-wheel position to initialize counts to 0.

This means that if the first event is generated by a jog-wheel move, that first move will be lost.

Any user interaction with the ShuttleXpress device will generate an event, informing the driver of the jog-wheel position. So if you (for example) push one of the buttons at startup, the jog-wheel will work fine and notice the first click.

Pins

- (bit out) `shuttleexpress.0.button-0`
- (bit out) `shuttleexpress.0.button-0-not`
- (bit out) `shuttleexpress.0.button-1`
- (bit out) `shuttleexpress.0.button-1-not`
- (bit out) `shuttleexpress.0.button-2`
- (bit out) `shuttleexpress.0.button-2-not`
- (bit out) `shuttleexpress.0.button-3`
- (bit out) `shuttleexpress.0.button-3-not`
- (bit out) `shuttleexpress.0.button-4`
- (bit out) `shuttleexpress.0.button-4-not`

The five buttons around the outside, starting with the counter-clockwise-most one.

(s32 out) `shuttleexpress.0.counts`

Accumulated counts from the jog wheel (the inner wheel).

(s32 out) *shuttleexpress.0.spring-wheel-s32*

The current deflection of the spring-wheel (the outer wheel).
It's 0 at rest, and ranges from -7 at the counter-clockwise
extreme to +7 at the clockwise extreme.

(float out) *shuttleexpress.0.spring-wheel-f*

The current deflection of the spring-wheel (the outer wheel).
It's 0 at rest, -1 at the counter-clockwise extreme, and
+1 at the clockwise extreme. (The ShuttleXpress device reports the
spring-wheel position quantized from -7 to +7, so this pin reports
only 15 discrete values in its range.)

NAME

vfdb_vfd - HAL userspace component for Delta VFD-B Variable Frequency Drives

SYNOPSIS

vfdb_vfd [OPTIONS]

DESCRIPTION

This manual page explains the **vfdb_vfd** component. This component reads and writes to the VFD-B device via a Modbus connection.

vfdb_vfd is for use with LinuxCNC.

QUICK START

The VFD-B ships in a configuration that can not talk to this driver. The VFD-B must be reconfigured via the face plate by the integrator before it will work. This section gives a brief description of what changes need to be made, consult your Delta VFD-B manual for more details.

Switch the VFD-B to Modbus RTU frame format:

Switch parameter 09-04 from the factory default of 0 (Ascii framing) to 3, 4, or 5 (RTU framing). The setting you choose will determine several serial parameters in addition to the Modbus framing protocol.

Set the frequency control source to be Modbus, not the keypad:

Switch parameter 02-00 from factory default of 00 (keypad control) to 5 (control from RS-485).

Set the run/stop control source to be Modbus, not the keypad:

Switch parameter 02-01 from the factory default of 0 (control from keypad) to 3 (control from Modbus, with Stop enabled on the keypad).

OPTIONS

-n --name <halname>

set the HAL component name

-d --debug

Turn on debugging messages. Also toggled by sending a USR1 signal to the vfdb_vfd process.

-m --modbus-debug

Turn on Modbus debugging messages. This will cause all Modbus messages to be printed in hex on the terminal. Also toggled by sending a USR2 signal to the vfdb_vfd process.

-I --ini <inifilename>

take configuration from this ini file. Defaults to environment variable INI_FILE_NAME. Most vfdb_vfd configuration comes from the ini file, not from command-line arguments.

-S --section <section name>

take configuration from this section in the ini file. Defaults to 'VFD-B'.

-r --report-device

report device properties on console at startup

INI CONFIG VARIABLES**DEBUG**

Set to a non-zero value to enable general debug output from the VFD-B driver. Optional.

MODBUS_DEBUG

Set to a non-zero value to enable modbus debug output from the VFD-B driver. Optional.

DEVICE

Serial port device file to use for Modbus communication with the VFD-B. Defaults to '/dev/ttyS0'.

BAUD Modbus baud rate. Defaults to 19200.

BITS Modbus data bits. Defaults to 8.

PARITY

Modbus parity. Defaults to Even. Accepts 'Even', 'Odd', or 'None'.

STOPBITS

Modbus stop bits. Defaults to 1.

TARGET

Modbus target number of the VFD-B to speak to. Defaults to 1.

POLLCYCLES

Only read the less important variables from the VFD-B once in this many poll cycles. Defaults to 10.

RECONNECT_DELAY

If the connection to the VFD-B is broken, wait this many seconds before reconnecting. Defaults to 1.

MOTOR_HZ, MOTOR_RPM

The frequency of the motor (in Hz) and the corresponding speed of the motor (in RPM). This information is provided by the motor manufacturer, and is generally printed on the motor's name plate.

PINS**<name>.at-speed (bit, out)**

True when drive is at commanded speed (see *speed-tolerance* below)

<name>.enable (bit, in)

Enable the VFD. If False, all operating parameters are still read but control is released and panel control is enabled (subject to VFD setup).

<name>.frequency-command (float, out)

Current target frequency in HZ as set through speed-command (which is in RPM), from the VFD.

<name>.frequency-out (float, out)

Current output frequency of the VFD.

<name>.inverter-load-percentage (float, out)

Current load report from VFD.

<name>.is-e-stopped (bit, out)

The VFD is in emergency stop status (blinking "E" on panel).

<name>.is-stopped (bit, out)

True when the VFD reports 0 Hz output.

<name>.jog-mode (bit, in)

1 for ON and 0 for OFF, enables the VFD-B 'jog mode'. Speed control is disabled. This might be useful for spindle orientation.

<name>.max-rpm (float, out)

Actual RPM limit based on maximum frequency the VFD may generate, and the motors nameplate values. For instance, if *nameplate-HZ* is 50, and *nameplate-RPM* is 1410, but the VFD may generate up to 80Hz, then *max-rpm* would read as 2256 (80*1410/50). The frequency limit is read from the VFD at startup. To increase the upper frequency limit, the UL and FH parameters must be changed on the panel. See the VFD-B manual for instructions how to set the maximum frequency.

<name>.modbus-ok (bit, out)

True when the Modbus session is successfully established and the last 10 transactions returned without error.

<name>.motor-RPM (float, out)

Estimated current RPM value, from the VFD.

<name>.motor-RPS (float, out)

Estimated current RPS value, from the VFD.

<name>.output-voltage (float, out)

From the VFD.

<name>.output-current (float, out)

From the VFD.

<name>.speed-command (float, in)

Speed sent to VFD in RPM. It is an error to send a speed faster than the Motor Max RPM as set in the VFD.

<name>.spindle-on (bit, in)

1 for ON and 0 for OFF sent to VFD, only on when running.

<name>.max-speed (bit, in)

Ignore the loop-time parameter and run Modbus at maximum speed, at the expense of higher CPU usage. Suggested use during spindle positioning.

<name>.status (s32, out)

Drive Status of the VFD (see the VFD manual). A bitmap.

<name>.error-count (s32, out)

Total number of transactions returning a Modbus error.

<name>.error-code (s32, out)

Most recent Error Code from VFD.

<name>.frequency-limit (float, out)

Upper limit read from VFD setup.

PARAMETERS

<name>.loop-time (float, RW)

How often the Modbus is polled (default interval 0.1 seconds).

<name>.nameplate-HZ (float, RW)

Nameplate Hz of motor (default 50). Used to calculate target frequency (together with *nameplate-RPM*) for a target RPM value as given by speed-command.

<name>.nameplate-RPM (float, RW)

Nameplate RPM of motor (default 1410)

<name>.rpm-limit (float, RW)

Do-not-exceed soft limit for motor RPM (defaults to *nameplate-RPM*).

<name>.tolerance (float, RW)

Speed tolerance (default 0.01) for determining whether spindle is at speed (0.01 meaning: output frequency is within 1% of target frequency).

USAGE

The vfdb_vfd driver takes precedence over panel control while it is enabled (see *.enable* pin), effectively disabling the panel. Clearing the *.enable* pin re-enables the panel. Pins and parameters can still be set, but will not be written to the VFD until the *.enable* pin is set. Operating parameters are still read while bus control is disabled.

Exiting the vfdb_vfd driver in a controlled way will release the VFD from the bus and restore panel control.

See the LinuxCNC Integrators Manual for more information. For a detailed register description of the Delta VFD-B, see the VFD manual.

AUTHOR

Yishin Li; based on vfd11_vfd by Michael Haberler.

LICENSE

GPL

NAME

vfs11_vfd - HAL userspace component for Toshiba-Schneider VF-S11 Variable Frequency Drives

SYNOPSIS

vfs11_vfd [OPTIONS]

DESCRIPTION

This manual page explains the **vfs11_vfd** component. This component reads and writes to the vfs11 via a Modbus connection.

vfs11_vfd is for use with LinuxCNC.

OPTIONS

-n --name <halname>

set the HAL component name

-d --debug

Turn on debugging messages. Also toggled by sending a USR1 signal to the vfs11_vfd process.

-m --modbus-debug

Turn on Modbus debugging messages. This will cause all Modbus messages to be printed in hex on the terminal. Also toggled by sending a USR2 signal to the vfs11_vfd process.

-I --ini <inifilename>

take configuration from this ini file. Defaults to environment variable INI_FILE_NAME.

-S --section <section name>

take configuration from this section in the ini file. Defaults to 'VFS11'.

-r --report-device

report device properties on console at startup

PINS

<name>.acceleration-pattern (bit, in)

when true, set acceleration and deceleration times as defined in registers F500 and F501 respectively. Used in PID loops to choose shorter ramp times to avoid oscillation.

<name>.alarm-code (s32, out)

non-zero if drive is in alarmed state. Bitmap describing alarm information (see register FC91 description). Use *err-reset* (see below) to clear the alarm.

<name>.at-speed (bit, out)

when drive is at commanded speed (see *speed-tolerance* below)

<name>.current-load-percentage (float, out)

reported from the VFD

<name>.dc-brake (bit, in)

engage the DC brake. Also turns off spindle-on.

<name>.enable (bit, in)

enable the VFD. If false, all operating parameters are still read but control is released and panel control is enabled (subject to VFD setup).

<name>.err-reset (bit, in)

reset errors (alarms a.k.a Trip and e-stop status). Resetting the VFD may cause a 2-second delay until it's rebooted and Modbus is up again.

<name>.estop (bit, in)

put the VFD into emergency-stopped status. No operation possible until cleared with *err-reset* or powercycling.

<name>.frequency-command (float, out)
 current target frequency in HZ as set through speed-command (which is in RPM), from the VFD

<name>.frequency-out (float, out)
 current output frequency of the VFD

<name>.inverter-load-percentage (float, out)
 current load report from VFD

<name>.is-e-stopped (bit, out)
 the VFD is in emergency stop status (blinking "E" on panel). Use *err-reset* to reboot the VFD and clear the e-stop status.

<name>.is-stopped (bit, out)
 true when the VFD reports 0 Hz output

<name>.jog-mode (bit, in)
 1 for ON and 0 for OFF, enables the VF-S11 'jog mode'. Speed control is disabled, and the output frequency is determined by register F262 (preset to 5Hz). This might be useful for spindle orientation.

<name>.max-rpm (float, R)
 actual RPM limit based on maximum frequency the VFD may generate, and the motors nameplate values. For instance, if *nameplate-HZ* is 50, and *nameplate-RPM_* is 1410, but the VFD may generate up to 80Hz, then *max-rpm* would read as 2256 (80*1410/50). The frequency limit is read from the VFD at startup. To increase the upper frequency limit, the UL and FH parameters must be changed on the panel. See the VF-S11 manual for instructions how to set the maximum frequency.

<name>.modbus-ok (bit, out)
 true when the Modbus session is successfully established and the last 10 transactions returned without error.

<name>.motor-RPM (float, out)
 estimated current RPM value, from the VFD

<name>.output-current-percentage (float, out)
 from the VFD

<name>.output-voltage-percentage (float, out)
 from the VFD

<name>.output-voltage (float, out)
 from the VFD

<name>.speed-command (float, in)
 speed sent to VFD in RPM. It is an error to send a speed faster than the Motor Max RPM as set in the VFD

<name>.spindle-fwd (bit, in)
 1 for FWD and 0 for REV, sent to VFD

<name>.spindle-on (bit, in)
 1 for ON and 0 for OFF sent to VFD, only on when running

<name>.spindle-rev (bit, in)
 1 for ON and 0 for OFF, only on when running

<name>.max-speed (bit, in)
 ignore the loop-time parameter and run Modbus at maximum speed, at the expense of higher CPU usage. Suggested use during spindle positioning.

<name>.status (s32, out)

Drive Status of the VFD (see the TOSVERT VF-S11 Communications Function Instruction Manual, register FD01). A bitmap.

<name>.trip-code (s32, out)

trip code if VF-S11 is in tripped state.

<name>.error-count (s32, RW)

total number of transactions returning a Modbus error

PARAMETERS**<name>.frequency-limit (float, RO)**

upper limit read from VFD setup.

<name>.loop-time (float, RW)

how often the Modbus is polled (default interval 0.1 seconds)

<name>.nameplate-HZ (float, RW)

Nameplate Hz of motor (default 50). Used to calculate target frequency (together with *nameplate-RPM*) for a target RPM value as given by speed-command.

<name>.nameplate-RPM (float, RW)

Nameplate RPM of motor (default 1410)

<name>.rpm-limit (float, RW)

do-not-exceed soft limit for motor RPM (defaults to *nameplate-RPM*).

<name>.tolerance (float, RW)

speed tolerance (default 0.01) for determining whether spindle is at speed (0.01 meaning: output frequency is within 1% of target frequency)

USAGE

The vfs11_vfd driver takes precedence over panel control while it is enabled (see *.enable* pin), effectively disabling the panel. Clearing the *.enable* pin re-enables the panel. Pins and parameters can still be set, but will not be written to the VFD until the *.enable* pin is set. Operating parameters are still read while bus control is disabled.

Exiting the vfs11_vfd driver in a controlled will release the VFD from the bus and restore panel control.

See the LinuxCNC Integrators Manual for more information. For a detailed register description of the Toshiba VFD's, see the "TOSVERT VF-S11 Communications Function Instruction Manual" (Toshiba document number E6581222) and the "TOSVERT VF-S11 Instruction manual" (Toshiba document number E6581158).

AUTHOR

Michael Haberler; based on gs2_vfd by Steve Padnos and John Thornton.

LICENSE

GPL

NAME

hal – Introduction to the HAL API

DESCRIPTION

HAL stands for Hardware Abstraction Layer, and is used by LinuxCNC to transfer realtime data to and from I/O devices and other low-level modules.

hal.h defines the API and data structures used by the HAL. This file is included in both realtime and non-realtime HAL components. HAL uses the RTPAI real time interface, and the #define symbols RTAPI and ULAPI are used to distinguish between realtime and non-realtime code. The API defined in this file is implemented in hal_lib.c and can be compiled for linking to either realtime or user space HAL components.

The HAL is a very modular approach to the low level parts of a motion control system. The goal of the HAL is to allow a systems integrator to connect a group of software components together to meet whatever I/O requirements he (or she) needs. This includes realtime and non-realtime I/O, as well as basic motor control up to and including a PID position loop. What these functions have in common is that they all process signals. In general, a signal is a data item that is updated at regular intervals. For example, a PID loop gets position command and feedback signals, and produces a velocity command signal.

HAL is based on the approach used to design electronic circuits. In electronics, off-the-shelf components like integrated circuits are placed on a circuit board and their pins are interconnected to build whatever overall function is needed. The individual components may be as simple as an op-amp, or as complex as a digital signal processor. Each component can be individually tested, to make sure it works as designed. After the components are placed in a larger circuit, the signals connecting them can still be monitored for testing and troubleshooting.

Like electronic components, HAL components have pins, and the pins can be interconnected by signals.

In the HAL, a *signal* contains the actual data value that passes from one pin to another. When a signal is created, space is allocated for the data value. A *pin* on the other hand, is a pointer, not a data value. When a pin is connected to a signal, the pin's pointer is set to point at the signal's data value. This allows the component to access the signal with very little run-time overhead. (If a pin is not linked to any signal, the pointer points to a dummy location, so the realtime code doesn't have to deal with null pointers or treat unlinked variables as a special case in any way.)

There are three approaches to writing a HAL component. Those that do not require hard realtime performance can be written as a single user mode process. Components that need hard realtime performance but have simple configuration and init requirements can be done as a single kernel module, using either pre-defined init info, or insmod-time parameters. Finally, complex components may use both a kernel module for the realtime part, and a user space process to handle ini file access, user interface (possibly including GUI features), and other details.

HAL uses the RTAPI/ULAPI interface. If RTAPI is #defined hal_lib.c would generate a kernel module hal_lib.o that is insmoded and provides the functions for all kernel module based components. The same source file compiled with the ULAPI #define would make a user space hal_lib.o that is statically linked to user space code to make user space executables. The variable lists and link information are stored in a block of shared memory and protected with mutexes, so that kernel modules and any of several user mode programs can access the data.

REALTIME CONSIDERATIONS

For an explanation of realtime considerations, see **intro(3rtapi)**.

HAL STATUS CODES

Except as noted in specific manual pages, HAL returns negative errno values for errors, and nonnegative values for success.

SEE ALSO[intro\(3rtapi\)](#)

`hal_add_funct_to_thread(3hal)`

HAL

`hal_add_funct_to_thread(3hal)`

NAME

`hal_add_funct_to_thread` – cause a function to be executed at regular intervals

SYNTAX

```
int hal_add_funct_to_thread(const char *funct_name, const char *thread_name,  
                           int position)
```

```
int hal_del_funct_from_thread(const char *funct_name, const char *thread_name)
```

ARGUMENTS

funct_name

The name of the function

thread_name

The name of the thread

position

The desired location within the thread. This determines when the function will run, in relation to other functions in the thread. A positive number indicates the desired location as measured from the beginning of the thread, and a negative is measured from the end. So +1 means this function will become the first one to run, +5 means it will be the fifth one to run, -2 means it will be next to last, and -1 means it will be last. Zero is illegal.

DESCRIPTION

hal_add_funct_to_thread adds a function exported by a realtime HAL component to a realtime thread. This determines how often and in what order functions are executed.

hal_del_funct_from_thread removes a function from a thread.

RETURN VALUE

Returns a HAL status code.

REALTIME CONSIDERATIONS

Call only from realtime init code, not from user space or realtime code.

SEE ALSO

`hal_thread_new(3hal)`, `hal_export_funct(3hal)`

hal_create_thread(3hal)	HAL	hal_create_thread(3hal)
-------------------------	-----	-------------------------

NAME

hal_create_thread – Create a HAL thread

SYNTAX

```
int hal_create_thread(const char *name, unsigned long period, int uses_fp)
```

```
int hal_thread_delete(const char *name)
```

ARGUMENTS

name The name of the thread

period The interval, in nanoseconds, between iterations of the thread

uses_fp Must be nonzero if a function which uses floating-point will be attached to this thread.

DESCRIPTION

hal_create_thread establishes a realtime thread that will execute one or more HAL functions periodically.

All thread periods are rounded to integer multiples of the hardware timer period, and the timer period is based on the first thread created. Threads must be created in order, from the fastest to the slowest. HAL assigns decreasing priorities to threads that are created later, so creating them from fastest to slowest results in rate monotonic priority scheduling.

hal_delete_thread deletes a previously created thread.

REALTIME CONSIDERATIONS

Call only from realtime init code, not from user space or realtime code.

RETURN VALUE

Returns a HAL status code.

SEE ALSO

[hal_export_funct\(3hal\)](#)

`hal_exit(3hal)`

HAL

`hal_exit(3hal)`

NAME

`hal_exit` – Shut down HAL

SYNTAX

`int hal_exit(int comp_id)`

ARGUMENTS

comp_id

A HAL component identifier returned by an earlier call to **hal_init**.

DESCRIPTION

hal_exit shuts down and cleans up HAL and RTAPI. It must be called prior to exit by any module that called **hal_init**.

REALTIME CONSIDERATIONS

Call only from within user or init/cleanup code, not from realtime tasks.

RETURN VALUE

Returns a HAL status code.

NAME

hal_export_funct – create a realtime function callable from a thread

SYNTAX

```
typedef void(*hal_funct_t)(void * arg, long period)
int hal_export_funct(const char *name, hal_funct_t funct, void *arg, int uses_fp, int reentrant, int comp_id)
```

ARGUMENTS

name The name of the function.

funct The pointer to the function

arg The argument to be passed as the first parameter of *funct*

uses_fp Nonzero if the function uses floating-point operations, including assignment of floating point values with "=".

reentrant

If reentrant is non-zero, the function may be preempted and called again before the first call completes. Otherwise, it may only be added to one thread.

comp_id

A HAL component identifier returned by an earlier call to **hal_init**.

DESCRIPTION

hal_export_funct makes a realtime function provided by a component available to the system. A subsequent call to **hal_add_funct_to_thread** can be used to schedule the execution of the function as needed by the system.

When this function is placed on a HAL thread, and HAL threads are started, *funct* is called repeatedly with two arguments: *void *arg* is the same value that was given to **hal_export_funct**, and *long period* is the interval between calls in nanoseconds.

Each call to the function should do a small amount of work and return.

RETURN VALUE

Returns a HAL status code.

SEE ALSO

hal_create_thread(3hal), **hal_add_funct_to_thread(3hal)**

NAME

hal_init – Sets up HAL and RTAPI

SYNTAX

```
int hal_init(const char *modname)
```

ARGUMENTS

modname

The name of this hal module

DESCRIPTION

hal_init sets up HAL and RTAPI. It must be called by any module that intends to use the API, before any other RTAPI calls.

modname must point to a string that identifies the module. The string may be no longer than **HAL_NAME_LEN** characters.

REALTIME CONSIDERATIONS

Call only from within user or init/cleanup code, not from realtime tasks.

RETURN VALUE

On success, returns a positive integer module ID, which is used for subsequent calls to hal and rtapi APIs.
On failure, returns a HAL error code.

NAME

hal_malloc – Allocate space in the HAL shared memory area

SYNTAX

```
void *hal_malloc(long int size)
```

ARGUMENTS

size Gives the size, in bytes, of the block

DESCRIPTION

hal_malloc allocates a block of memory from the main HAL shared memory area. It should be used by all components to allocate memory for HAL pins and parameters. It allocates ‘size’ bytes, and returns a pointer to the allocated space, or NULL (0) on error. The returned pointer will be properly aligned for any type HAL supports. A component should allocate during initialization all the memory it needs.

The allocator is very simple, and there is no ‘free’. The entire HAL shared memory area is freed when the last component calls **hal_exit**. This means that if you continuously install and remove one component while other components are present, you eventually will fill up the shared memory and an install will fail. Removing all components completely clears memory and you start fresh.

RETURN VALUE

A pointer to the allocated space, which is properly aligned for any variable HAL supports. Returns NULL on error.

hal_param_new(3hal)	HAL	hal_param_new(3hal)
---------------------	-----	---------------------

NAME

hal_param_new – Create a HAL parameter

SYNTAX

```
int hal_param_bit_new(const char *name, hal_param_dir_t dir, hal_bit_t * data_addr, int comp_id)
```

```
int hal_param_float_new(const char *name, hal_param_dir_t dir, hal_float_t * data_addr, int comp_id)
```

```
int hal_param_u32_new(const char *name, hal_param_dir_t dir, hal_u32_t * data_addr, int comp_id)
```

```
int hal_param_s32_new(const char *name, hal_param_dir_t dir, hal_s32_t * data_addr, int comp_id)
```

```
int hal_param_bit_newf(hal_param_dir_t dir, hal_bit_t * data_addr, int comp_id, const char *fmt, ...)
```

```
int hal_param_float_newf(hal_param_dir_t dir, hal_float_t * data_addr, int comp_id, const char *fmt, ...)
```

```
int hal_param_u32_newf(hal_param_dir_t dir, hal_u32_t * data_addr, int comp_id, const char *fmt, ...)
```

```
int hal_param_s32_newf(hal_param_dir_t dir, hal_s32_t * data_addr, int comp_id, const char *fmt, ...)
```

```
int hal_param_new(const char *name, hal_type_t type, hal_param_dir_t dir, void *data_addr, int comp_id)
```

ARGUMENTS

name The name to give to the created parameter

dir The direction of the parameter, from the viewpoint of the component. It may be one of **HAL_RO**, or **HAL_RW**. A component may assign a value to any parameter, but other programs (such as hal-cmd) may only assign a value to a parameter that is **HAL_RW**.

data_addr

The address of the data, which must lie within memory allocated by **hal_malloc**.

*comp_id*A HAL component identifier returned by an earlier call to **hal_init**.*fmt, ...* A printf-style format string and arguments*type* The type of the parameter, as specified in **hal_type_t(3hal)**.

DESCRIPTION

The **hal_param_new** family of functions create a new *param* object.

There are functions for each of the data types that the HAL supports. Pins may only be linked to signals of the same type.

RETURN VALUE

Returns a HAL status code.

SEE ALSO

hal_type_t(3hal)

NAME

hal_parport – portable access to PC-style parallel ports

SYNTAX

```
#include "hal_parport.h"

int hal_parport_get(int comp_id, hal_parport_t *port, unsigned short base, unsigned short base_hi,
                     unsigned int modes)
void hal_parport_release(hal_parport_t *port)
```

ARGUMENTS

<i>comp_id</i>	A HAL component identifier returned by an earlier call to hal_init .
<i>port</i>	A pointer to a hal_parport_t structure
<i>base</i>	The base address of the port (if port >= 16) or the linux port number of the port (if port < 16)
<i>base_hi</i>	The "high" address of the port (location of the ECP registers), 0 to use a probed high address, or -1 to disable the high address
<i>modes</i>	Advise the driver of the desired port modes, from <linux/parport.h>. If a linux-detected port does not provide the requested modes, a warning is printed with rtapi_print_msg. This does not make the port request fail, because unfortunately, many systems that have working EPP parports are not detected as such by Linux.

DESCRIPTION

hal_parport_get allocates a parallel port for exclusive use of the named hal component. The port must be released with **hal_parport_release** before the component exits with **hal_exit**.

HIGH ADDRESS PROBING

If the port is a parallel port known to Linux, and Linux detected a high I/O address, this value is used. Otherwise, if base+0x400 is not registered to any device, it is used. Otherwise, no address is used. If no high address is detected, port->base_hi is 0.

PARPORt STRUCTURE

```
typedef struct
{
    unsigned short base;
    unsigned short base_hi;
    .... // and further unspecified fields
} hal_parport_t;
```

RETURN VALUE

hal_parport_get returns a HAL status code. On success, *port* is filled out with information about the allocated port. On failure, the contents of *port* are undefined except that it is safe (but not required) to pass this port to **hal_parport_release**.

hal_parport_release does not return a value. It always succeeds.

hal_pin_new(3hal)

HAL

hal_pin_new(3hal)

NAME

hal_pin_new – Create a HAL pin

SYNTAX

```
int hal_pin_bit_new(const char *name, hal_pin_dir_t dir, hal_bit_t ** data_ptr_addr, int comp_id)
```

```
int hal_pin_float_new(const char *name, hal_pin_dir_t dir, hal_float_t ** data_ptr_addr, int comp_id)
```

```
int hal_pin_u32_new(const char *name, hal_pin_dir_t dir, hal_u32_t ** data_ptr_addr, int comp_id)
```

```
int hal_pin_s32_new(const char *name, hal_pin_dir_t dir, hal_s32_t ** data_ptr_addr, int comp_id)
```

```
int hal_pin_bit_newf(hal_pin_dir_t dir, hal_bit_t ** data_ptr_addr, int comp_id, const char *fmt, ...)
```

```
int hal_pin_float_newf(hal_pin_dir_t dir, hal_float_t ** data_ptr_addr, int comp_id, const char *fmt, ...)
```

```
int hal_pin_u32_newf(hal_pin_dir_t dir, hal_u32_t ** data_ptr_addr, int comp_id, const char *fmt, ...)
```

```
int hal_pin_s32_newf(hal_pin_dir_t dir, hal_s32_t ** data_ptr_addr, int comp_id, const char *fmt, ...)
```

```
int hal_pin_new(const char *name, hal_type_t type, hal_pin_dir_t dir, void **data_ptr_addr, int comp_id)
```

ARGUMENTS

name The name of the pin

dir

The direction of the pin, from the viewpoint of the component. It may be one of **HAL_IN**, **HAL_OUT**, or **HAL_IO**. Any number of **HAL_IN** or **HAL_IO** pins may be connected to the same signal, but at most one **HAL_OUT** pin is permitted. A component may assign a value to a pin that is **HAL_OUT** or **HAL_IO**, but may not assign a value to a pin that is **HAL_IN**.

data_ptr_addr

The address of the pointer-to-data, which must lie within memory allocated by **hal_malloc**.

comp_id

A HAL component identifier returned by an earlier call to **hal_init**.

fmt,

A printf-style format string and arguments

type

The type of the param, as specified in **hal_type_t(3hal)**.

DESCRIPTION

The **hal_pin_new** family of functions create a new *pin* object. Once a pin has been created, it can be linked to a signal object using **hal_link**. A pin contains a pointer, and the component that owns the pin can dereference the pointer to access whatever signal is linked to the pin. (If no signal is linked, it points to a dummy signal.)

There are functions for each of the data types that the HAL supports. Pins may only be linked to signals of the same type.

RETURN VALUE

Returns a HAL status code.

SEE ALSO

hal_type_t(3hal), hal_link(3hal)

NAME

hal_ready – indicates that this component is ready

SYNTAX

hal_ready(int *comp_id*)

ARGUMENTS

comp_id

A HAL component identifier returned by an earlier call to **hal_init**.

DESCRIPTION

hal_ready indicates that this component is ready (has created all its pins, parameters, and functions). This must be called in any realtime HAL component before its **rtapi_app_init** exits, and in any userspace component before it enters its main loop.

RETURN VALUE

Returns a HAL status code.

NAME

hal_set_constructor – Set the constructor function for this component

SYNTAX

```
typedef int (*hal_constructor_t)(const char *prefix, const char *arg); int hal_set_constructor(int comp_id,  
hal_constructor_t constructor)
```

ARGUMENTS

comp_id A HAL component identifier returned by an earlier call to **hal_init**.

prefix The prefix to be given to the pins, parameters, and functions in the new instance

arg An argument that may be used by the component to customize this instance.

DESCRIPTION

As an experimental feature in HAL 2.1, components may be *constructable*. Such a component may create pins and parameters not only at the time the module is loaded, but it may create additional pins and parameters, and functions on demand.

RETURN VALUE

Returns a HAL status code.

SEE ALSO

halcmd(1)

hal_set_lock(3hal)

HAL

hal_set_lock(3hal)

NAME

hal_set_lock, hal_get_lock – Set or get the HAL lock level

SYNTAX

int hal_set_lock(unsigned char *lock_type*)

int hal_get_lock()

ARGUMENTS

lock_type

The desired lock type, which may be a bitwise combination of: **HAL_LOCK_LOAD**, **HAL_LOCK_CONFIG**, **HAL_LOCK_PARAMS**, or **HAL_LOCK_PARAMS**. **HAL_LOCK_NONE** or 0 locks nothing, and **HAL_LOCK_ALL** locks everything.

DESCRIPTION

RETURN VALUE

hal_set_lock Returns a HAL status code. **hal_get_lock** returns the current HAL lock level or a HAL status code.

NAME

hal_signal_new, hal_signal_delete, hal_link, hal_unlink – Manipulate HAL signals

SYNTAX

int hal_signal_new(const char **signal_name*, hal_type_t *type*)

int hal_signal_delete(const char **signal_name*)

int hal_link(const char **pin_name*, const char **signal_name*)

int hal_unlink(const char **pin_name*)

ARGUMENTS

signal_name

The name of the signal

pin_name

The name of the pin

type The type of the signal, as specified in **hal_type_t(3hal)**.

DESCRIPTION

hal_signal_new creates a new signal object. Once a signal has been created, pins can be linked to it with **hal_link**. The signal object contains the actual storage for the signal data. Pin objects linked to the signal have pointers that point to the data. 'name' is the name of the new signal. It may be no longer than HAL_NAME_LEN characters. If there is already a signal with the same name the call will fail.

hal_link links a pin to a signal. If the pin is already linked to the desired signal, the command succeeds. If the pin is already linked to some other signal, it is an error. In either case, the existing connection is not modified. (Use 'hal_unlink' to break an existing connection.) If the signal already has other pins linked to it, they are unaffected - one signal can be linked to many pins, but a pin can be linked to only one signal.

hal_unlink unlinks any signal from the specified pin.

hal_signal_delete deletes a signal object. Any pins linked to the object are unlinked.

RETURN VALUE

Returns a HAL status code.

SEE ALSO

hal_type_t(3hal)

hal_start_threads(3hal)	HAL	hal_start_threads(3hal)
-------------------------	-----	-------------------------

NAME

hal_start_threads – Allow HAL threads to begin executing

SYNTAX

```
int hal_start_threads()
```

```
int hal_stop_threads()
```

ARGUMENTS

DESCRIPTION

hal_start_threads starts all threads that have been created. This is the point at which realtime functions start being called.

hal_stop_threads stops all threads that were previously started by **hal_start_threads**. It should be called before any component that is part of a system exits.

RETURN VALUE

Returns a HAL status code.

SEE ALSO

[hal_export_funct\(3hal\)](#), [hal_create_thread\(3hal\)](#), [hal_add_funct_to_thread\(3hal\)](#)

NAME

`hal_type_t` – typedefs for HAL datatypes

DESCRIPTION

`typedef ... hal_bool;`

A type which may have a value of 0 or nonzero.

`typedef ... hal_bit_t;`

A volatile type which may have a value of 0 or nonzero.

`typedef ... hal_s32_t;`

A volatile type which may have a value from -2147483648 to 2147483647.

`typedef ... hal_u32_t;`

A volatile type which may have a value from 0 to 4294967295.

`typedef ... hal_float_t;`

A volatile floating-point type, which typically has the same precision and range as the C type `double`.

`typedef ... real_t;`

A nonvolatile floating-point type with at least as much precision as `hal_float_t`.

`typedef ... ireal_t;`

A nonvolatile unsigned integral type the same size as `hal_float_t`.

`typedef enum hal_type_t:`

HAL_BIT

Corresponds to the type `hal_bit_t`.

HAL_FLOAT

Corresponds to the type `hal_float_t`.

HAL_S32

Corresponds to the type `hal_s32_t`.

HAL_U32

Corresponds to the type `hal_u32_t`.

NOTES

`hal_bit_t` is typically a typedef to an integer type whose range is larger than just 0 and 1. When testing the value of a `hal_bit_t`, never compare it to 1. Prefer one of the following:

- `if(b)`
- `if(b != 0)`

It is often useful to refer to a type that can represent all the values as a hal type, but without the volatile qualifier. The following types correspond with the hal types:

`hal_bit_t` `int`

`hal_s32_t` `_s32`

`hal_u32_t` `_u32`

`hal_float_t` `hal_real_t`

Take care not to use the types `s32` and `u32`. These will compile in kernel modules but not in userspace, and not for "realtime components" when using simulated (userspace) realtime.

`hal_type_t(3hal)`

HAL

`hal_type_t(3hal)`

SEE ALSO

`hal_pin_new(3hal)`, `hal_param_new(3hal)`

NAME

undocumented – undocumented functions in HAL

SEE ALSO

The header file *hal.h*. Most hal functions have documentation in that file.

NAME

rtapi – Introduction to the RTAPI API

DESCRIPTION

RTAPI is a library providing a uniform API for several real time operating systems. As of ver 2.1, RTLinux, RTAI, and a pure userspace simulator are supported.

HEADER FILES

rtapi.h

The file **rtapi.h** defines the RTAPI for both realtime and non-realtime code. This is a change from Rev 2, where the non-realtime (user space) API was defined in ulapi.h and used different function names. The symbols RTAPI and ULAPI are used to determine which mode is being compiled, RTAPI for realtime and ULAPI for non-realtime.

rtapi_math.h

The file rtapi_math.h defines floating-point functions and constants. It should be used instead of <math.h> in rtapi real-time components.

rtapi_string.h

The file rtapi_string.h defines string-related functions. It should be used instead of <string.h> in rtapi real-time components.

rtapi_byteorder.h

This file defines the preprocessor macros RTAPI_BIG_ENDIAN, RTAPI_LITTLE_ENDIAN, and RTAPI_FLOAT_BIG_ENDIAN as true or false depending on the characteristics of the target system. It should be used instead of <endian.h> (userspace) or <linux/byteorder.h> (kernel space).

rtapi_limits.h

This file defines the minimum and maximum value of some fundamental integral types, such as INT_MIN and INT_MAX. This should be used instead of <limits.h> because that header file is not available to kernel modules.

REALTIME CONSIDERATIONS

Userspace code

Certain functions are not available in userspace code. This includes functions that perform direct device access such as **rtapi_inb(3)**.

Init/cleanup code

Certain functions may only be called from realtime init/cleanup code. This includes functions that perform memory allocation, such as **rtapi_shmem_new(3)**.

Realtime code

Only a few functions may be called from realtime code. This includes functions that perform direct device access such as **rtapi_inb(3)**. It excludes most Linux kernel APIs such as do_gettimeofday(3) and many rtapi APIs such as rtapi_shmem_new(3).

Simulator

For an RTAPI module to be buildable in the "sim" environment (fake realtime system without special privileges), it must not use **any** linux kernel APIs, and must not use the RTAPI APIs for direct device access

such as **rtapi_inb(3)**. This automatically includes any hardware device drivers, and also devices which use Linux kernel APIs to do things like create special devices or entries in the **/proc** filesystem.

RTAPI STATUS CODES

Except as noted in specific manual pages, RTAPI returns negative errno values for errors, and nonnegative values for success.

NAME

rtapi_app_exit – User-provided function to shut down a component

SYNTAX

```
#include <rtapi_app.h>
void rtapi_app_exit(void) {...}
```

ARGUMENTS

None

DESCRIPTION

The body of **rtapi_app_exit**, which is provided by the component author, generally consists of a call to **rtapi_exit** or **hal_exit**, preceded by other component-specific shutdown code.

This code is called when unloading a component which successfully initialized (i.e., returned zero from its **rtapi_app_main**). It is not called when the component did not successfully initialize.

RETURN CODE

None.

REALTIME CONSIDERATIONS

Called automatically by the rtapi infrastructure in an initialization (not realtime) context.

SEE ALSO

rtapi_app_main(3rtapi), **rtapi_exit(3rtapi)**, **hal_exit(3hal)**

NAME

rtapi_app_main – User-provided function to initialize a component

SYNTAX

```
#include <rtapi_app.h>
int rtapi_app_main(void) {...}
```

ARGUMENTS

None

DESCRIPTION

The body of **rtapi_app_main**, which is provided by the component author, generally consists of a call to **rtapi_init** or **hal_init**, followed by other component-specific initialization code.

RETURN VALUE

Return 0 for success. Return a negative errno value (e.g., -EINVAL) on error. Existing code also returns RTAPI or HAL error values, but using negative errno values gives better diagnostics from insmod.

REALTIME CONSIDERATIONS

Called automatically by the rtapi infrastructure in an initialization (not realtime) context.

SEE ALSO

[rtapi_app_exit\(3rtapi\)](#), [rtapi_init\(3rtapi\)](#), [hal_init\(3hal\)](#)

NAME

rtapi_clock_set_period – set the basic time interval for realtime tasks

SYNTAX

```
rtapi_clock_set_period(long int nsec)
```

ARGUMENTS

nsec The desired basic time interval for realtime tasks.

DESCRIPTION

rtapi_clock_set_period sets the basic time interval for realtime tasks. All periodic tasks will run at an integer multiple of this period. The first call to **rtapi_clock_set_period** with *nsec* greater than zero will start the clock, using *nsec* as the clock period in nano-seconds. Due to hardware and RTOS limitations, the actual period may not be exactly what was requested. On success, the function will return the actual clock period if it is available, otherwise it returns the requested period. If the requested period is outside the limits imposed by the hardware or RTOS, it returns **-EINVAL** and does not start the clock. Once the clock is started, subsequent calls with non-zero *nsec* return **-EINVAL** and have no effect. Calling **rtapi_clock_set_period** with *nsec* set to zero queries the clock, returning the current clock period, or zero if the clock has not yet been started.

REALTIME CONSIDERATIONS

Call only from within init/cleanup code, not from realtime tasks. This function is not available from user (non-realtime) code.

RETURN VALUE

The actual period provided by the RTOS, which may be different than the requested period, or a RTAPI status code.

NAME

rtapi_delay – Busy-loop for short delays

SYNTAX

```
void rtapi_delay(long int nsec)  
void rtapi_delay_max()
```

ARGUMENTS

nsec The desired delay length in nanoseconds

DESCRIPTION

rtapi_delay is a simple delay. It is intended only for short delays, since it simply loops, wasting CPU cycles.

rtapi_delay_max returns the max delay permitted (usually approximately 1/4 of the clock period). Any call to **rtapi_delay** requesting a delay longer than the max will delay for the max time only.

rtapi_delay_max should be called before using **rtapi_delay** to make sure the required delays can be achieved. The actual resolution of the delay may be as good as one nano-second, or as bad as a several microseconds.

REALTIME CONSIDERATIONS

May be called from init/cleanup code, and from within realtime tasks.

RETURN VALUE

rtapi_delay_max returns the maximum delay permitted.

SEE ALSO

rtapi_clock_set_period(3rtapi)

NAME

rtapi_div_u64 – unsigned division of a 64-bit number by a 32-bit number

SYNTAX

```
__u64 rtapi_div_u64_rem(__u64 dividend, __u32 divisor, __u32 *remainder)
__u64 rtapi_div_u64(__u64 dividend, __u32 divisor)
__s64 rtapi_div_s64(__s64 dividend, __s32 divisor)
__s64 rtapi_div_s64_rem(__s64 dividend, __s32 divisor, __s32 *remainder)
```

ARGUMENTS

dividend

The value to be divided

divisor The value to divide by

remainder

Pointer to the location to store the remainder. This may not be a NULL pointer. If the remainder is not desired, call **rtapi_div_u64** or **rtapi_div_s64**.

DESCRIPTION

Perform integer division (and optionally compute the remainder) with a 64-bit dividend and 32-bit divisor.

RETURN VALUE

The result of integer division of *dividend* / *divisor*. In versions with the *remainder* argument, the remainder is stored in the pointed-to location.

NOTES

If the result of the division does not fit in the return type, the result is undefined.

This function exists because in kernel space the use of the division operator on a 64-bit type can lead to an undefined symbol such as `__umoddi3` when the module is loaded.

REALTIME CONSIDERATIONS

May be called from init/cleanup code and from within realtime tasks. Available in userspace components.

NAME

rtapi_exit – Shut down RTAPI

SYNTAX

```
int rtapi_exit(int module_id)
```

ARGUMENTS

module_id

An rtapi module identifier returned by an earlier call to **rtapi_init**.

DESCRIPTION

rtapi_exit shuts down and cleans up the RTAPI. It must be called prior to exit by any module that called **rtapi_init**.

REALTIME CONSIDERATIONS

Call only from within user or init/cleanup code, not from relatime tasks.

RETURN VALUE

Returns a RTAPI status code.

NAME

rtapi_get_time – get the current time

SYNTAX

```
long long rtapi_get_time()
long long rtapi_get_clocks()
```

DESCRIPTION

rtapi_get_time returns the current time in nanoseconds. Depending on the RTOS, this may be time since boot, or time since the clock period was set, or some other time. Its absolute value means nothing, but it is monotonically increasing and can be used to schedule future events, or to time the duration of some activity. Returns a 64 bit value. The resolution of the returned value may be as good as one nano-second, or as poor as several microseconds. May be called from init/cleanup code, and from within realtime tasks.

rtapi_get_clocks returns the current time in CPU clocks. It is fast, since it just reads the TSC in the CPU instead of calling a kernel or RTOS function. Of course, times measured in CPU clocks are not as convenient, but for relative measurements this works fine. Its absolute value means nothing, but it is monotonically increasing and can be used to schedule future events, or to time the duration of some activity. (on SMP machines, the two TSC's may get out of sync, so if a task reads the TSC, gets swapped to the other CPU, and reads again, the value may decrease. RTAPI tries to force all RT tasks to run on one CPU.) Returns a 64 bit value. The resolution of the returned value is one CPU clock, which is usually a few nanoseconds to a fraction of a nanosecond.

Note that *long long* math may be poorly supported on some platforms, especially in kernel space. Also note that **rtapi_print()** will NOT print *long longs*. Most time measurements are relative, and should be done like this:

```
deltat = (long int)(end_time - start_time);
```

where *end_time* and *start_time* are *longlong* values returned from **rtapi_get_time**, and *deltat* is an ordinary *long int* (32 bits). This will work for times up to a second or so, depending on the CPU clock frequency. It is best used for millisecond and microsecond scale measurements though.

RETURN VALUE

Returns the current time in nanoseconds or CPU clocks.

NOTES

Certain versions of the Linux kernel provide a global variable **cpu_khz**. Computing

```
deltat = (end_clocks - start_clocks) / cpu_khz;
```

gives the duration measured in milliseconds. Computing

```
deltat = (end_clocks - start_clocks) * 1000000 / cpu_khz;
```

gives the duration measured in nanoseconds for deltas less than about 9 trillion clocks (e.g., 3000 seconds at 3GHz).

REALTIME CONSIDERATIONS

May be called from init/cleanup code and from within realtime tasks. Not available in userspace components.

NAME

rtapi_init – Sets up RTAPI

SYNTAX

```
int rtapi_init(const char *modname)
```

ARGUMENTS

modname

The name of this rtapi module

DESCRIPTION

rtapi_init sets up the RTAPI. It must be called by any module that intends to use the API, before any other RTAPI calls.

modname can optionally point to a string that identifies the module. The string will be truncated at **RTAPI_NAME_LEN** characters. If *modname* is **NULL**, the system will assign a name.

REALTIME CONSIDERATIONS

Call only from within user or init/cleanup code, not from relatime tasks.

RETURN VALUE

On success, returns a positive integer module ID, which is used for subsequent calls to **rtapi_xxx_new**, **rtapi_xxx_delete**, and **rtapi_exit**. On failure, returns an RTAPI error code.

NAME

rtapi_module_param – Specifying module parameters

SYNTAX

RTAPI_MP_INT(*var, description*)

RTAPI_MP_LONG(*var, description*)

RTAPI_MP_STRING(*var, description*)

RTAPI_MP_ARRAY_INT(*var, num, description*)

RTAPI_MP_ARRAY_LONG(*var, num, description*)

RTAPI_MP_ARRAY_STRING(*var, num, description*)

MODULE_LICENSE(*license*)

MODULE_AUTHOR(*author*)

MODULE_DESCRIPTION(*description*)

EXPORT_FUNCTION(*function*)

ARGUMENTS

var The variable where the parameter should be stored

description

A short description of the parameter or module

num The maximum number of values for an array parameter

license The license of the module, for instance "GPL"

author The author of the module

function

The pointer to the function to be exported

DESCRIPTION

These macros are portable ways to declare kernel module parameters. They must be used in the global scope, and are not followed by a terminating semicolon. They must be used after the associated variable or function has been defined.

NOTES

EXPORT_FUNCTION makes a symbol available for use by a subsequently loaded component. It is unrelated to hal functions, which are described in hal_export_funct(3hal)

Interpretation of license strings

MODULE_LICENSE follows the kernel's definition of license strings. Notably, "GPL" indicates "GNU Public License v2 or later". (emphasis ours).

"GPL"

GNU Public License v2 or later

"GPL v2"

GNU Public License v2

"GPL and additional rights"

GNU Public License v2 rights and more

"Dual BSD/GPL"

GNU Public License v2 or BSD license choice

"Dual MIT/GPL"

GNU Public License v2 or MIT license choice

"Dual MPL/GPL"

GNU Public License v2 or Mozilla license choice

"Proprietary"

Non-free products

It is still good practice to include a license block which indicates the author, copyright date, and disclaimer of warranty as recommended by the GNU GPL.

REALTIME CONSIDERATIONS

Not available in userspace code.

NAME

rtapi_mutex – Mutex-related functions

SYNTAX

```
int rtapi_mutex_try(unsigned long *mutex)
```

```
void rtapi_mutex_get(unsigned long *mutex)
```

```
void rtapi_mutex_give(unsigned long *mutex)
```

ARGUMENTS

mutex A pointer to the mutex.

DESCRIPTION

rtapi_mutex_try makes a non-blocking attempt to get the mutex. If the mutex is available, it returns 0, and the mutex is no longer available. Otherwise, it returns a nonzero value.

rtapi_mutex_get blocks until the mutex is available.

rtapi_mutex_give releases a mutex acquired by **rtapi_mutex_try** or **rtapi_mutex_get**.

REALTIME CONSIDERATIONS

rtapi_mutex_give and **rtapi_mutex_try** may be used from user, init/cleanup, and realtime code.

rtapi_mutex_get may not be used from realtime code.

RETURN VALUE

rtapi_mutex_try returns 0 if the mutex was claimed, and nonzero otherwise.

rtapi_mutex_get and **rtapi_mutex_give** have no return value.

NAME

rtapi_outb, **rtapi_inb** – Perform hardware I/O

SYNTAX

```
void rtapi_outb(unsigned char byte, unsigned int port)
unsigned char rtapi_inb(unsigned int port)
```

ARGUMENTS

port The address of the I/O port
byte The byte to be written to the port

DESCRIPTION

rtapi_outb writes a byte to a hardware I/O port. **rtapi_inb** reads a byte from a hardware I/O port.

REALTIME CONSIDERATIONS

May be called from init/cleanup code and from within realtime tasks. Not available in userspace components.

RETURN VALUE

rtapi_inb returns the byte read from the given I/O port

NOTES

The I/O address should be within a region previously allocated by **rtapi_request_region**. Otherwise, another real-time module or the Linux kernel might attempt to access the I/O region at the same time.

SEE ALSO

rtapi_region(3rtapi)

NAME

rtapi_print, **rtapi_print_msg** – print diagnostic messages

SYNTAX

```
void rtapi_print(const char *fmt, ...)
```

```
void rtapi_print_msg(int level, const char *fmt, ...)
```

```
typedef void(*rtapi_msg_handler_t)(msg_level_t level, const char *msg);
```

```
void rtapi_set_msg_handler(rtapi_msg_handler_t handler);
```

```
rtapi_msg_handler_t rtapi_set_msg_handler(void);
```

ARGUMENTS

level A message level: One of **RTAPI_MSG_ERR**, **RTAPI_MSG_WARN**, **RTAPI_MSG_INFO**, or **RTAPI_MSG_DBG**.

handler

A function to call from **rtapi_print** or **rtapi_print_msg** to actually output the message.

fmt

... Other arguments are as for *printf(3)*.

DESCRIPTION

rtapi_print and **rtapi_print_msg** work like the standard C *printf* functions, except that a reduced set of formatting operations are supported.

Depending on the RTOS, the default may be to print the message to stdout, stderr, a kernel log, etc. In RTAPI code, the action may be changed by a call to **rtapi_set_msg_handler**. A **NULL** argument to **rtapi_set_msg_handler** restores the default handler. **rtapi_msg_get_handler** returns the current handler. When the message came from **rtapi_print**, *level* is **RTAPI_MSG_ALL**.

rtapi_print_msg works like **rtapi_print** but only prints if *level* is less than or equal to the current message level.

REALTIME CONSIDERATIONS

rtapi_print and **rtapi_print_msg** May be called from user, init/cleanup, and realtime code. **rtapi_get_msg_handler** and **rtapi_set_msg_handler** may be called from realtime init/cleanup code. A message handler passed to **rtapi_set_msg_handler** may only call functions that can be called from real-time code.

RETURN VALUE

None.

`rtapi_print(3rtapi)`

RTAPI

`rtapi_print(3rtapi)`

SEE ALSO

`rtapi_set_msg_level(3rtapi)`, `rtapi_get_msg_level(3rtapi)`, `printf(3)`

NAME

rtapi_prio – thread priority functions

SYNTAX

```
int rtapi_prio_highest()  
int rtapi_prio_lowest()  
int rtapi_prio_next_higher(int prio)  
int rtapi_prio_next_lower(int prio)
```

ARGUMENTS

prio A value returned by a prior **rtapi_prio_xxx** call

DESCRIPTION

The **rtapi_prio_xxxx** functions provide a portable way to set task priority. The mapping of actual priority to priority number depends on the RTOS. Priorities range from **rtapi_prio_lowest** to **rtapi_prio_highest**, inclusive. To use this API, use one of two methods:

- 1) Set your lowest priority task to **rtapi_prio_lowest**, and for each task of the next lowest priority, set their priorities to **rtapi_prio_next_higher(previous)**.
- 2) Set your highest priority task to **rtapi_prio_highest**, and for each task of the next highest priority, set their priorities to **rtapi_prio_next_lower(previous)**.

N.B. A high priority task will pre-empt or interrupt a lower priority task. Linux is always the lowest priority!

REALTIME CONSIDERATIONS

Call these functions only from within init/cleanup code, not from realtime tasks.

RETURN VALUE

Returns an opaque real-time priority number.

SEE ALSO

rtapi_task_new(3rtapi)

NAME

rtapi_region – functions to manage I/O memory regions

SYNTAX

```
void *rtapi_request_region(unsigned long base, unsigned long int size, const char *name)
```

```
void rtapi_release_region(unsigned long base, unsigned long int size)
```

ARGUMENTS

base The base address of the I/O region

size The size of the I/O region

name The name to be shown in /proc/ioports

DESCRIPTION

rtapi_request_region reserves I/O memory starting at *base* and going for *size* bytes.

REALTIME CONSIDERATIONS

May be called from realtime init/cleanup code only.

BUGS

On kernels before 2.4.0, **rtapi_request_region** always succeeds.

RETURN VALUE

rtapi_request_region returns NULL if the allocation fails, and a non-NUL value otherwise.

rtapi_release_region has no return value.

NAME

rtapi_get_msg_level, **rtapi_set_msg_level** – Get or set the logging level

SYNTAX

```
int rtapi_set_msg_level(int level)
```

```
int rtapi_get_msg_level()
```

ARGUMENTS

level The desired logging level

DESCRIPTION

Get or set the RTAPI message level used by **rtapi_print_msg**. Depending on the RTOS, this level may apply to a single RTAPI module, or it may apply to a group of modules.

REALTIME CONSIDERATIONS

May be called from user, init/cleanup, and realtime code.

RETURN VALUE

rtapi_set_msg_level returns a status code, and **rtapi_get_msg_level** returns the current level.

SEE ALSO

rtapi_print_msg(3rtapi)

NAME

rtapi_shmem – Functions for managing shared memory blocks

SYNTAX

```
int rtapi_shmem_new(int key, int module_id, unsigned long int size)
```

```
int rtapi_shmem_delete(int shmem_id, int module_id)
```

```
int rtapi_shmem_getptr(int shmem_id, void ** ptr)
```

ARGUMENTS

key Identifies the memory block. Key must be nonzero. All modules wishing to use the same memory must use the same key.

module_id

Module identifier returned by a prior call to **rtapi_init**.

size The desired size of the shared memory block, in bytes

ptr The pointer to the shared memory block. Note that the block may be mapped at a different address for different modules.

DESCRIPTION

rtapi_shmem_new allocates a block of shared memory. *key* identifies the memory block, and must be non-zero. All modules wishing to access the same memory must use the same key. *module_id* is the ID of the module that is making the call (see **rtapi_init**). The block will be at least *size* bytes, and may be rounded up. Allocating many small blocks may be very wasteful. When a particular block is allocated for the first time, the first 4 bytes are zeroed. Subsequent allocations of the same block by other modules or processes will not touch the contents of the block. Applications can use those bytes to see if they need to initialize the block, or if another module already did so. On success, it returns a positive integer ID, which is used for all subsequent calls dealing with the block. On failure it returns a negative error code.

rtapi_shmem_delete frees the shared memory block associated with *shmem_id*. *module_id* is the ID of the calling module. Returns a status code.

rtapi_shmem_getptr sets **ptr* to point to shared memory block associated with *shmem_id*.

REALTIME CONSIDERATIONS

rtapi_shmem_getptr may be called from user code, init/cleanup code, or realtime tasks.

rtapi_shmem_new and **rtapi_shmem_dete** may not be called from realtime tasks.

RETURN VALUE

NAME

rtapi_snprintf, rtapi_vsnprintf – Perform snprintf-like string formatting

SYNTAX

```
int rtapi_snprintf(char *buf, unsigned long int size, const char *fmt, ...)
```

```
int rtapi_vsnprintf(char *buf, unsigned long int size, const char *fmt, va_list apfB)
```

ARGUMENTS

As for *snprintf(3)* or *vsnprintf(3)*.

DESCRIPTION

These functions work like the standard C printf functions, except that a reduced set of formatting operations are supported.

In particular: formatting of long long values is not supported. Formatting of floating-point values is done as though with %A even when other formats like %f are specified.

REALTIME CONSIDERATIONS

May be called from user, init/cleanup, and realtime code.

RETURN VALUE

The number of characters written to *buf*.

SEE ALSO

[printf\(3\)](#)

NAME

rtapi_task_new – create a realtime task

SYNTAX

```
int rtapi_task_new(void (*taskcode)(void*), void *arg,           int prio, unsigned long stacksize, int
                   uses_fp)                                         taskcode
```

```
int rtapi_task_delete(int task_id)
```

ARGUMENTS

taskcode

A pointer to the function to be called when the task is started

arg An argument to be passed to the *taskcode* function when the task is started

prio A task priority value returned by **rtapi_prio_xxxx**

uses_fp A flag that tells the OS whether the task uses floating point or not.

task_id A task ID returned by a previous call to **rtapi_task_new**

DESCRIPTION

rtapi_task_new creates but does not start a realtime task. The task is created in the "paused" state. To start it, call either **rtapi_task_start** for periodic tasks, or **rtapi_task_resume** for free-running tasks.

REALTIME CONSIDERATIONS

Call only from within init/cleanup code, not from realtime tasks.

RETURN VALUE

On success, returns a positive integer task ID. This ID is used for all subsequent calls that need to act on the task. On failure, returns an RTAPI status code.

SEE ALSO

rtapi_prio(3rtapi), **rtapi_task_start(3rtapi)**, **rtapi_task_wait(3rtapi)**, **rtapi_task_resume(3rtapi)**

NAME

rtapi_task_pause, **rtapi_task_resume** – pause and resume real-time tasks

SYNTAX

```
void rtapi_task_pause(int task_id)  
void rtapi_task_resume(int task_id)
```

ARGUMENTS

task_id An RTAPI task identifier returned by an earlier call to **rtapi_task_new**.

DESCRIPTION

rtapi_task_resume starts a task in free-running mode. The task must be in the "paused" state.

A free running task runs continuously until either:

- 1) It is preempted by a higher priority task. It will resume as soon as the higher priority task releases the CPU.
- 2) It calls a blocking function, like **rtapi_sem_take**. It will resume when the function unblocks.
- 3) It is returned to the "paused" state by **rtapi_task_pause**. May be called from init/cleanup code, and from within realtime tasks.

rtapi_task_pause causes a task to stop execution and change to the "paused" state. The task can be free-running or periodic. Note that **rtapi_task_pause** may be called from any task, or from init or cleanup code, not just from the task that is to be paused. The task will resume execution when either **rtapi_task_resume** or **rtapi_task_start** (depending on whether this is a free-running or periodic task) is called.

REALTIME CONSIDERATIONS

May be called from init/cleanup code, and from within realtime tasks.

RETURN VALUE

An RTAPI status code.

SEE ALSO

rtapi_task_new(3rtapi), **rtapi_task_start(3rtapi)**

NAME

rtapi_task_start – start a realtime task in periodic mode

SYNTAX

```
int rtapi_task_start(int task_id, unsigned long period_nsec)
```

ARGUMENTS

task_id A task ID returned by a previous call to **rtapi_task_new**

period_nsec

The clock period in nanoseconds between iterations of a periodic task

DESCRIPTION

rtapi_task_start starts a task in periodic mode. The task must be in the *paused* state.

REALTIME CONSIDERATIONS

Call only from within init/cleanup code, not from realtime tasks.

RETURN VALUE

Returns an RTAPI status code.

SEE ALSO

rtapi_task_new(3rtapi), **rtapi_task_pause(3rtapi)**, **rtapi_task_resume(3rtapi)**

NAME

rtapi_task_wait – suspend execution of this periodic task

SYNTAX

```
void rtapi_task_wait()
```

DESCRIPTION

rtapi_task_wait suspends execution of the current task until the next period. The task must be periodic. If not, the result is undefined.

REALTIME CONSIDERATIONS

Call only from within a periodic realtime task

RETURN VALUE

None

SEE ALSO

rtapi_task_start(3rtapi), **rtapi_task_pause(3rtapi)**

NAME

undocumented – undocumented functions in RTAPI

SEE ALSO

The header file *rtapi.h*. Most rtapi functions have documentation in that file.

NAME

abs – Compute the absolute value and sign of the input signal

SYNOPSIS

loadrt abs [count=N|names=name1[,name2...]]

FUNCTIONS

abs.N (requires a floating-point thread)

PINS

abs.N.in float in
Analog input value

abs.N.out float out
Analog output value, always positive

abs.N.sign bit out
Sign of input, false for positive, true for negative

abs.N.is-positive bit out
TRUE if input is positive, FALSE if input is 0 or negative

abs.N.is-negative bit out
TRUE if input is negative, FALSE if input is 0 or positive

LICENSE

GPL

NAME

abs_s32 – Compute the absolute value and sign of the input signal

SYNOPSIS

loadrt abs_s32 [count=N|names=name1[,name2...]]

FUNCTIONS

abs-s32.N

PINS

abs-s32.N.in s32 in
input value

abs-s32.N.out s32 out
output value, always non-negative

abs-s32.N.sign bit out
Sign of input, false for positive, true for negative

abs-s32.N.is-positive bit out
TRUE if input is positive, FALSE if input is 0 or negative

abs-s32.N.is-negative bit out
TRUE if input is negative, FALSE if input is 0 or positive

LICENSE

GPL

NAME

and2 – Two-input AND gate

SYNOPSIS

loadrt and2 [count=N|names=name1[,name2...]]

FUNCTIONS

and2.N

PINS

and2.N.in0 bit in

and2.N.in1 bit in

and2.N.out bit out

out is computed from the value of **in0** and **in1** according to the following rule:

in0=TRUE in1=TRUE

out=TRUE

Otherwise,

out=FALSE

LICENSE

GPL

NAME

at_pid – proportional/integral/derivative controller with auto tuning

SYNOPSIS

```
loadrt at_pid [num_chan=num | names=name1[,name2...]]
```

DESCRIPTION

at_pid is a classic Proportional/Integral/Derivative controller, used to control position or speed feedback loops for servo motors and other closed-loop applications.

at_pid supports a maximum of sixteen controllers. The number that are actually loaded is set by the **num_chan** argument when the module is loaded. Alternatively, specify **names=** and unique names separated by commas.

The **num_chan=** and **names=** specifiers are mutually exclusive. If neither **num_chan=** nor **names=** are specified, the default value is three.

If **debug** is set to 1 (the default is 0), some additional HAL parameters will be exported, which might be useful for tuning, but are otherwise unnecessary.

at_pid has a built in auto tune mode. It works by setting up a limit cycle to characterize the process. From this, **Pgain/Igain/Dgain** or **Pgain/Igain/FF1** can be determined using Ziegler-Nichols. When using **FF1**, scaling must be set so that **output** is in user units per second.

During auto tuning, the **command** input should not change. The limit cycle is setup around the commanded position. No initial tuning values are required to start auto tuning. Only **tune-cycles**, **tune-effort** and **tune-mode** need be set before starting auto tuning. When auto tuning completes, the tuning parameters will be set. If running from LinuxCNC, the FERROR setting for the axis being tuned may need to be loosened up as it must be larger than the limit cycle amplitude in order to avoid a following error.

To perform auto tuning, take the following steps. Move the axis to be tuned, to somewhere near the center of its travel. Set **tune-cycles** (the default value should be fine in most cases) and **tune-mode**. Set **tune-effort** to a small value. Set **enable** to true. Set **tune-mode** to true. Set **tune-start** to true. If no oscillation occurs, or the oscillation is too small, slowly increase **tune-effort**. Auto tuning can be aborted at any time by setting **enable** or **tune-mode** to false.

NAMING

The names for pins, parameters, and functions are prefixed as:

pid.N. for N=0,1,...,num-1 when using **num_chan=num**

nameN. for nameN=name1,name2,... when using **names=name1,name2,...**

The **pid.N.** format is shown in the following descriptions.

FUNCTIONS

pid.N.do-pid-calc (uses floating-point)

Does the PID calculations for control loop *N*.

PINS

pid.N.command float in

The desired (commanded) value for the control loop.

pid.N.feedback float in

The actual (feedback) value, from some sensor such as an encoder.

pid.N.error float out

The difference between command and feedback.

pid.N.output float out

The output of the PID loop, which goes to some actuator such as a motor.

pid.N.enable bit in

When true, enables the PID calculations. When false, **output** is zero, and all internal integrators, etc, are reset.

pid.N.tune-mode bit in

When true, enables auto tune mode. When false, normal PID calculations are performed.

pid.N.tune-start bit io

When set to true, starts auto tuning. Cleared when the auto tuning completes.

PARAMETERS**pid.N.Pgain** float rw

Proportional gain. Results in a contribution to the output that is the error multiplied by **Pgain**.

pid.N.Igain float rw

Integral gain. Results in a contribution to the output that is the integral of the error multiplied by **Igain**. For example an error of 0.02 that lasted 10 seconds would result in an integrated error (**errorI**) of 0.2, and if **Igain** is 20, the integral term would add 4.0 to the output.

pid.N.Dgain float rw

Derivative gain. Results in a contribution to the output that is the rate of change (derivative) of the error multiplied by **Dgain**. For example an error that changed from 0.02 to 0.03 over 0.2 seconds would result in an error derivative (**errorD**) of 0.05, and if **Dgain** is 5, the derivative term would add 0.25 to the output.

pid.N.bias float rw

bias is a constant amount that is added to the output. In most cases it should be left at zero. However, it can sometimes be useful to compensate for offsets in servo amplifiers, or to balance the weight of an object that moves vertically. **bias** is turned off when the PID loop is disabled, just like all other components of the output. If a non-zero output is needed even when the PID loop is disabled, it should be added with an external HAL sum2 block.

pid.N.FF0 float rw

Zero order feed-forward term. Produces a contribution to the output that is **FF0** multiplied by the commanded value. For position loops, it should usually be left at zero. For velocity loops, **FF0** can compensate for friction or motor counter-EMF and may permit better tuning if used properly.

pid.N.FF1 float rw

First order feed-forward term. Produces a contribution to the output that is **FF1** multiplied by the derivative of the commanded value. For position loops, the contribution is proportional to speed, and can be used to compensate for friction or motor CEMF. For velocity loops, it is proportional to acceleration and can compensate for inertia. In both cases, it can result in better tuning if used properly.

pid.N.FF2 float rw

Second order feed-forward term. Produces a contribution to the output that is **FF2** multiplied by the second derivative of the commanded value. For position loops, the contribution is proportional to acceleration, and can be used to compensate for inertia. For velocity loops, it should usually be left at zero.

pid.N.deadband float rw

Defines a range of "acceptable" error. If the absolute value of **error** is less than **deadband**, it will be treated as if the error is zero. When using feedback devices such as encoders that are inherently quantized, the deadband should be set slightly more than one-half count, to prevent the control loop from hunting back and forth if the command is between two adjacent encoder values. When

the absolute value of the error is greater than the deadband, the deadband value is subtracted from the error before performing the loop calculations, to prevent a step in the transfer function at the edge of the deadband. (See **BUGS**.)

pid.N.maxoutput float rw

Output limit. The absolute value of the output will not be permitted to exceed **maxoutput**, unless **maxoutput** is zero. When the output is limited, the error integrator will hold instead of integrating, to prevent windup and overshoot.

pid.N.maxerror float rw

Limit on the internal error variable used for P, I, and D. Can be used to prevent high **Pgain** values from generating large outputs under conditions when the error is large (for example, when the command makes a step change). Not normally needed, but can be useful when tuning non-linear systems.

pid.N.maxerrorD float rw

Limit on the error derivative. The rate of change of error used by the **Dgain** term will be limited to this value, unless the value is zero. Can be used to limit the effect of **Dgain** and prevent large output spikes due to steps on the command and/or feedback. Not normally needed.

pid.N.maxerrorI float rw

Limit on error integrator. The error integrator used by the **Igain** term will be limited to this value, unless it is zero. Can be used to prevent integrator windup and the resulting overshoot during/after sustained errors. Not normally needed.

pid.N.maxcmdD float rw

Limit on command derivative. The command derivative used by **FF1** will be limited to this value, unless the value is zero. Can be used to prevent **FF1** from producing large output spikes if there is a step change on the command. Not normally needed.

pid.N.maxcmdDD float rw

Limit on command second derivative. The command second derivative used by **FF2** will be limited to this value, unless the value is zero. Can be used to prevent **FF2** from producing large output spikes if there is a step change on the command. Not normally needed.

pid.N.tune-type u32 rw

When set to 0, **Pgain/Igain/Dgain** are calculated. When set to 1, **Pgain/Igain/FF1** are calculated.

pid.N.tune-cycles u32 rw

Determines the number of cycles to run to characterize the process. **tune-cycles** actually sets the number of half cycles. More cycles results in a more accurate characterization as the average of all cycles is used.

pid.N.tune-effort float rw

Determines the effort used in setting up the limit cycle in the process. **tune-effort** should be set to a positive value less than **maxoutput**. Start with something small and work up to a value that results in a good portion of the maximum motor current being used. The smaller the value, the smaller the amplitude of the limit cycle.

pid.N.errorI float ro (only if debug=1)

Integral of error. This is the value that is multiplied by **Igain** to produce the Integral term of the output.

pid.N.errorD float ro (only if debug=1)

Derivative of error. This is the value that is multiplied by **Dgain** to produce the Derivative term of the output.

pid.N.commandD float ro (only if debug=1)

Derivative of command. This is the value that is multiplied by **FF1** to produce the first order feed-forward term of the output.

pid.N.commandDD float ro (only if debug=1)

Second derivative of command. This is the value that is multiplied by **FF2** to produce the second order feed-forward term of the output.

pid.N.ultimate-gain float ro (only if debug=1)

Determined from process characterization. **ultimate-gain** is the ratio of **tune-effort** to the limit cycle amplitude multiplied by 4.0 divided by Pi. **pid.N.ultimate-period** float ro (only if debug=1)

Determined from process characterization. **ultimate-period** is the period of the limit cycle.

BUGS

Some people would argue that deadband should be implemented such that error is treated as zero if it is within the deadband, and be unmodified if it is outside the deadband. This was not done because it would cause a step in the transfer function equal to the size of the deadband. People who prefer that behavior are welcome to add a parameter that will change the behavior, or to write their own version of **at_pid**. However, the default behavior should not be changed.

NAME

bin2gray – convert a number to the gray-code representation

SYNOPSIS

loadrt bin2gray [count=N|names=name1[,name2...]]

DESCRIPTION

Converts a number into gray-code

FUNCTIONS

bin2gray.N

PINS

bin2gray.N.in u32 in
binary code in

bin2gray.N.out u32 out
gray code out

AUTHOR

andy pugh

LICENSE

GPL

NAME

biquad – Biquad IIR filter

SYNOPSIS

loadrt biquad [count=N|names=name1[,name2...]]

DESCRIPTION

Biquad IIR filter. Implements the following transfer function: $H(z) = (n_0 + n_1z^{-1} + n_2z^{-2}) / (1 + d_1z^{-1} + d_2z^{-2})$

FUNCTIONS

biquad.N (requires a floating-point thread)

PINS

biquad.N.in float in
Filter input.

biquad.N.out float out
Filter output.

biquad.N.enable bit in (default: 0)
Filter enable. When false, the in is passed to out without any filtering. A transition from false to true causes filter coefficients to be calculated according to parameters

biquad.N.valid bit out (default: 0)
When false, indicates an error occurred when calculating filter coefficients.

PARAMETERS

biquad.N.type u32 rw (default: 0)
Filter type determines the type of filter coefficients calculated. When 0, coefficients must be loaded directly. When 1, a low pass filter is created. When 2, a notch filter is created.

biquad.N.f0 float rw (default: 250.0)
The corner frequency of the filter.

biquad.N.Q float rw (default: 0.7071)
The Q of the filter.

biquad.N.d1 float rw (default: 0.0)
1st-delayed denominator coef

biquad.N.d2 float rw (default: 0.0)
2nd-delayed denominator coef

biquad.N.n0 float rw (default: 1.0)
non-delayed numerator coef

biquad.N.n1 float rw (default: 0.0)
1st-delayed numerator coef

biquad.N.n2 float rw (default: 0.0)
2nd-delayed numerator coef

biquad.N.s1 float rw (default: 0.0)
biquad.N.s2 float rw (default: 0.0)

LICENSE

GPL

NAME

bitslice – Converts an unsigned-32 input into individual bits

SYNOPSIS

loadrt bitslice [count=N|names=name1[,name2...]] [personality=P,P,...]

DESCRIPTION

This component creates individual bit-outputs for each bit of an unsigned-32 input. The number of bits can be limited by the "personality" modparam. The inverse process can be performed by the weighted_sum HAL component.

FUNCTIONS

bitslice.N

PINS

bitslice.N.in u32 in

The input value

bitslice.N.out-MM bit out (MM=00..personality)

AUTHOR

Andy Pugh

LICENSE

GPL2+

NAME

bitwise – Computes various bitwise operations on the two input values

SYNOPSIS

loadrt bitwise [count=N|names=name1[,name2...]]

FUNCTIONS

bitwise.N

PINS

bitwise.N.in0 u32 in

First input value

bitwise.N.in1 u32 in

Second input value

bitwise.N.out-and u32 out

The bitwise AND of the two inputs

bitwise.N.out-or u32 out

The bitwise OR of the two inputs

bitwise.N.out-xor u32 out

The bitwise XOR of the two inputs

bitwise.N.out-nand u32 out

The inverse of the bitwise AND

bitwise.N.out-nor u32 out

The inverse of the bitwise OR

bitwise.N.out-xnor u32 out

The inverse of the bitwise XOR

AUTHOR

Andy Pugh

LICENSE

GPL 2+

NAME

bldc – BLDC and AC-servo control component

SYNOPSIS

loadrt bldc personality=*P*

DESCRIPTION

This component is designed as an interface between the most common forms of three-phase motor feedback devices and the corresponding types of drive. However there is no requirement that the motor and drive should necessarily be of inherently compatible types.

SYNOPSIS

(ignore the auto-generated SYNOPSIS above)

loadrt bldc cfg=qi6,aH

Each instance of the component is defined by a group of letters describing the input and output types. A comma separates individual instances of the component.

Tags

Input type definitions are all lower-case.

n No motor feedback. This mode could be used to drive AC induction motors, but is also potentially useful for creating free-running motor simulators for drive testing.

h Hall sensor input. Brushless DC motors (electronically commutated permanent magnet 3-phase motors) typically use a set of three Hall sensors to measure the angular position of the rotor. A lower-case **h** in the cfg string indicates that these should be used.

a Absolute encoder input. (Also possibly used by some forms of Resolver conversion hardware). The presence of this tag over-rides all other inputs. Note that the component still requires to be connected to the **rawcounts** encoder pin to prevent loss of commutation on index-reset.

q Incremental (quadrature) encoder input. If this input is used then the rotor will need to be homed before the motor can be run.

i Use the index of an incremental encoder as a home reference.

f Use a 4-bit Gray-scale pattern to determine rotor alignment. This scheme is only used on the Fanuc "Red Cap" motors. This mode could be used to control one of these motors using a non-Fanuc drive.

Output type descriptions are all upper-case.

Defaults The component will always calculate rotor angle, phase angle and the absolute value of the input **value** for interfacing with drives such as the Mesa 8i20. It will also default to three individual, bipolar phase output values if no other output type modifiers are used.

B Bit level outputs. Either 3 or 6 logic-level outputs indicating which high or low gate drivers on an external drive should be used.

6 Create 6 rather than the default 3 outputs. In the case of numeric value outputs these are separate positive and negative drive amplitudes. Both have positive magnitude.

H Emulated Hall sensor output. This mode can be used to control a drive which expects 3x Hall signals, or to convert between a motor with one hall pattern and a drive which expects a different one.

F Emulated Fanuc Red Cap Gray-code encoder output. This mode might be used to drive a non-Fanuc

motor using a Fanuc drive intended for the "Red-Cap" motors.

T Force Trapezoidal mode.

OPERATING MODES

The component can control a drive in either Trapezoidal or Sinusoidal mode, but will always default to sinusoidal if the input and output modes allow it. This can be over-ridden by the **T** tag. Sinusoidal commutation is significantly smoother (trapezoidal commutation induces 13% torque ripple).

ROTOR HOMING.

To use an encoder for commutation a reference 0-degrees point must be found. The component uses the convention that motor zero is the point that an unloaded motor aligns to with a positive voltage on the A (or U) terminal and the B & C (or V and W) terminals connected together and to -ve voltage. There will be two such positions on a 4-pole motor, 3 on a 6-pole and so on. They are all functionally equivalent as far as driving the motor is concerned. If the motor has Hall sensors then the motor can be started in trapezoidal commutation mode, and will switch to sinusoidal commutation when an alignment is found. If the mode is **qh** then the first Hall state-transition will be used. If the mode is **qhi** then the encoder index will be used. This gives a more accurate homing position if the distance in encoder counts between motor zero and encoder index is known. To force homing to the Hall edges instead simply omit the **i**.

Motors without Hall sensors may be homed in synchronous/direct mode. The better of these options is to home to the encoder zero using the **iq** config parameter. When the **init** pin goes high the motor will rotate (in a direction determined by the **rev** pin) until the encoder indicates an index-latch (the servo thread runs too slowly to rely on detecting an encoder index directly). If there is no encoder index or its location relative to motor zero can not be found, then an alternative is to use *magnetic* homing using the **q** config. In this mode the motor will go through an alignment sequence ending at motor zero when the init pin goes high. It will then set the final position as motor zero. Unfortunately the motor is rather *springy* in this mode and so alignment is likely to be fairly sensitive to load.

FUNCTIONS

bldc.N (requires a floating-point thread)

PINS

bldc.N.hall1 bit in [if personality & 0x01]
Hall sensor signal 1

bldc.N.hall2 bit in [if personality & 0x01]
Hall sensor signal 2

bldc.N.hall3 bit in [if personality & 0x01]
Hall sensor signal 3

bldc.N.hall-error bit out [if personality & 0x01]

Indicates that the selected hall pattern gives inconsistent rotor position data. This can be due to the pattern being wrong for the motor, or one or more sensors being unconnected or broken. A consistent pattern is not necessarily valid, but an inconsistent one can never be valid.

bldc.N.C1 bit in [if (personality & 0x10)]
Fanuc Gray-code bit 0 input

bldc.N.C2 bit in [if (personality & 0x10)]
Fanuc Gray-code bit 1 input

bldc.N.C4 bit in [if (personality & 0x10)]
Fanuc Gray-code bit 2 input

bldc.N.C8 bit in [if (personality & 0x10)]
 Fanuc Gray-code bit 3 input

bldc.N.value float in
 PWM master amplitude input

bldc.N.lead-angle float in [if personality & 0x06] (default: 90)
 The phase lead between the electrical vector and the rotor position in degrees

bldc.N.rev bit in
 Set this pin true to reverse the motor. Negative PWM amplitudes will also reverse the motor and there will generally be a Hall pattern that runs the motor in each direction too.

bldc.N.frequency float in [if (personality & 0x0F) == 0]
 Frequency input for motors with no feedback at all, or those with only an index (which is ignored)

bldc.N.initvalue float in [if personality & 0x04] (default: 0.2)
 The current to be used for the homing sequence in applications where an incremental encoder is used with no hall-sensor feedback

bldc.N.rawcounts s32 in [if personality & 0x06] (default: 0)
 Encoder counts input. This must be linked to the encoder rawcounts pin or encoder index resets will cause the motor commutation to fail

bldc.N.index-enable bit io [if personality & 0x08]
 This pin should be connected to the associated encoder index-enable pin to zero the encoder when it passes index. This is only used indicate to the bldc control component that an index has been seen

bldc.N.init bit in [if (personality & 0x05) == 4]
 A rising edge on this pin starts the motor alignment sequence. This pin should be connected in such a way that the motors re-align any time that encoder monitoring has been interrupted. Typically this will only be at machine power-off. The alignment process involves powering the motor phases in such a way as to put the motor in a known position. The encoder counts are then stored in the **offset** parameter. The alignment process will tend to cause a following error if it is triggered while the axis is enabled, so should be set before the matching axis.N.enable pin. The complementary **init-done** pin can be used to handle the required sequencing.

Both pins can be ignored if the encoder offset is known explicitly, such as is the case with an absolute encoder. In that case the **offset** parameter can be set directly in the HAL file

bldc.N.init-done bit out [if (personality & 0x05) == 4] (default: 0)
 Indicates homing sequence complete

bldc.N.A-value float out [if (personality & 0xF00) == 0]
 Output amplitude for phase A

bldc.N.B-value float out [if (personality & 0xF00) == 0]
 Output amplitude for phase B

bldc.N.C-value float out [if (personality & 0xF00) == 0]
 Output amplitude for phase C

bldc.N.A-on bit out [if (personality & 0xF00) == 0x100]
 Output bit for phase A

bldc.N.B-on bit out [if (personality & 0xF00) == 0x100]
 Output bit for phase B

bldc.N.C-on bit out [if (personality & 0xF00) == 0x100]
 Output bit for phase C

bldc.N.A-high float out [if (personality & 0xF00) == 0x200]
 High-side driver for phase A

bldc.N.B-high float out [if (personality & 0xF00) == 0x200]
 High-side driver for phase B

bldc.N.C-high float out [if (personality & 0xF00) == 0x200]
 High-side driver for phase C

bldc.N.A-low float out [if (personality & 0xF00) == 0x200]
 Low-side driver for phase A

bldc.N.B-low float out [if (personality & 0xF00) == 0x200]
 Low-side driver for phase B

bldc.N.C-low float out [if (personality & 0xF00) == 0x200]
 Low-side driver for phase C

bldc.N.A-high-on bit out [if (personality & 0xF00) == 0x300]
 High-side driver for phase A

bldc.N.B-high-on bit out [if (personality & 0xF00) == 0x300]
 High-side driver for phase B

bldc.N.C-high-on bit out [if (personality & 0xF00) == 0x300]
 High-side driver for phase C

bldc.N.A-low-on bit out [if (personality & 0xF00) == 0x300]
 Low-side driver for phase A

bldc.N.B-low-on bit out [if (personality & 0xF00) == 0x300]
 Low-side driver for phase B

bldc.N.C-low-on bit out [if (personality & 0xF00) == 0x300]
 Low-side driver for phase C

bldc.N.hall1-out bit out [if (personality & 0x400)]
 Hall 1 output

bldc.N.hall2-out bit out [if (personality & 0x400)]
 Hall 2 output

bldc.N.hall3-out bit out [if (personality & 0x400)]
 Hall 3 output

bldc.N.C1-out bit out [if (personality & 0x800)]
 Fanuc Gray-code bit 0 output

bldc.N.C2-out bit out [if (personality & 0x800)]
 Fanuc Gray-code bit 1 output

bldc.N.C4-out bit out [if (personality & 0x800)]
 Fanuc Gray-code bit 2 output

bldc.N.C8-out bit out [if (personality & 0x800)]
 Fanuc Gray-code bit 3 output

bldc.N.phase-angle float out (default: 0)
 Phase angle including lead/lag angle after encoder zeroing etc. Useful for angle/current drives.
 This value has a range of 0 to 1 and measures electrical revolutions. It will have two zeros for a 4 pole motor, three for a 6-pole etc

bldc.N.rotor-angle float out (default: 0)
 Rotor angle after encoder zeroing etc. Useful for angle/current drives which add their own phase offset such as the 8i20. This value has a range of 0 to 1 and measures electrical revolutions. It will have two zeros for a 4 pole motor, three for a 6-pole etc

bldc.N.out float out

Current output, including the effect of the dir pin and the alignment sequence

bldc.N.out-dir bit out

Direction output, high if /fBvalue/fR is negative XOR /fBrev/fR is true.

bldc.N.out-abs float out

Absolute value of the input value

PARAMETERS**bldc.N.in-type** s32 r (default: -1)

state machine output, will probably hide after debug

bldc.N.out-type s32 r (default: -1)

state machine output, will probably hide after debug

bldc.N.scale s32 rw [if personality & 0x06] (default: 512)

The number of encoder counts per rotor revolution.

bldc.N.poles s32 rw [if personality & 0x06] (default: 4)

The number of motor poles. The encoder scale will be divided by this value to determine the number of encoder counts per electrical revolution

bldc.N.encoder-offset s32 rw [if personality & 0x0A] (default: 0)

The offset, in encoder counts, between the motor electrical zero and the encoder zero modulo the number of counts per electrical revolution

bldc.N.offset-measured s32 r [if personality & 0x04] (default: 0)

The encoder offset measured by the homing sequence (in certain modes)

bldc.N.drive-offset float rw (default: 0)

The angle, in degrees, applied to the commanded angle by the drive in degrees. This value is only used during the homing sequence of drives with incremental encoder feedback. It is used to back-calculate from commanded angle to actual phase angle. It is only relevant to drives which expect rotor-angle input rather than phase-angle demand. Should be 0 for most drives.

bldc.N.output-pattern u32 rw [if personality & 0x400] (default: 25)

Commutation pattern to be output in Hall Signal translation mode. See the description of /fBpattern/fR for details

bldc.N.pattern u32 rw [if personality & 0x01] (default: 25)

Commutation pattern to use, from 0 to 47. Default is type 25. Every plausible combination is included. The table shows the excitation pattern along the top, and the pattern code on the left hand side. The table entries are the hall patterns in H1, H2, H3 order. Common patterns are: 0 (30 degree commutation) and 26, its reverse. 17 (120 degree). 18 (alternate 60 degree). 21 (300 degree, Bodine). 22 (240 degree). 25 (60 degree commutation).

Note that a number of incorrect commutations will have non-zero net torque which might look as if they work, but don't really.

If your motor lacks documentation it might be worth trying every pattern.

Phases, Source - Sink						
pat	B-A	C-A	C-B	A-B	A-C	B-C
0	000	001	011	111	110	100
1	001	000	010	110	111	101
2	000	010	011	111	101	100
3	001	011	010	110	100	101
4	010	011	001	101	100	110
5	011	010	000	100	101	111
6	010	000	001	101	111	110
7	011	001	000	100	110	111
8	000	001	101	111	110	010
9	001	000	100	110	111	011
10	000	010	110	111	101	001
11	001	011	111	110	100	000
12	010	011	111	101	100	000
13	011	010	110	100	101	001
14	010	000	100	101	111	011
15	011	001	101	100	110	010
16	000	100	101	111	011	010
17	001	101	100	110	010	011
18	000	100	110	111	011	001
19	001	101	111	110	010	000
20	010	110	111	101	001	000
21	011	111	110	100	000	001
22	010	110	100	101	001	011
23	011	111	101	100	000	010
24	100	101	111	011	010	000
25	101	100	110	010	011	001
26	100	110	111	011	001	000
27	101	111	110	010	000	001
28	110	111	101	001	000	010
29	111	110	100	000	001	011
30	110	100	101	001	011	010
31	111	101	100	000	010	011
32	100	101	001	011	010	110
33	101	100	000	010	011	111
34	100	110	010	011	001	101
35	101	111	011	010	000	100
36	110	111	011	001	000	100
37	111	110	010	000	001	101
38	110	100	000	001	011	111
39	111	101	001	000	010	110
40	100	000	001	011	111	110
41	101	001	000	010	110	111
42	100	000	010	011	111	101
43	101	001	011	010	110	100
44	110	010	011	001	101	100
45	111	011	010	000	100	101
46	110	010	000	001	101	111
47	111	011	001	000	100	110

BLDC(9)

HAL Component

BLDC(9)

AUTHOR

Andy Pugh

LICENSE

GPL

NAME

bldc_hall3 – 3-wire BLDC motor driver using Hall sensors and trapezoidal commutation.

SYNOPSIS

The functionality of this component is now included in the generic "bldc" component. This component is likely to be removed in a future release

DESCRIPTION

This component produces a 3-wire bipolar output. This suits upstream drivers that interpret a negative input as a low-side drive and positive as a high-side drive. This includes the Hostmot2 3pwmgen function, which is likely to be the most common application of this component.

FUNCTIONS

bldc-hall3.N (requires a floating-point thread)

Interpret Hall sensor patterns and set 3-phase amplitudes

PINS

bldc-hall3.N.hall1 bit in
Hall sensor signal 1

bldc-hall3.N.hall2 bit in
Hall sensor signal 2

bldc-hall3.N.hall3 bit in
Hall sensor signal 3

bldc-hall3.N.value float in
PWM master amplitude input

bldc-hall3.N.dir bit in
Forwards / reverse selection. Negative PWM amplitudes will also reverse the motor and there will generally be a pattern that runs the motor in each direction too.

bldc-hall3.N.A-value float out
Output amplitude for phase A

bldc-hall3.N.B-value float out
Output amplitude for phase B

bldc-hall3.N.C-value float out
Output amplitude for phase C

PARAMETERS

bldc-hall3.N.pattern u32 rw (default: 25)

Commutation pattern to use, from 0 to 47. Default is type 25. Every plausible combination is included. The table shows the excitation pattern along the top, and the pattern code on the left hand side. The table entries are the hall patterns in H1, H2, H3 order. Common patterns are: 0 (30 degree commutation) and 26, its reverse. 17 (120 degree). 18 (alternate 60 degree). 21 (300 degree, Bodine). 22 (240 degree). 25 (60 degree commutation).

Note that a number of incorrect commutations will have non-zero net torque which might look as if they work, but don't really.

If your motor lacks documentation it might be worth trying every pattern.

Phases, Source - Sink						
pat	B-A	C-A	C-B	A-B	A-C	B-C
0	000	001	011	111	110	100
1	001	000	010	110	111	101
2	000	010	011	111	101	100
3	001	011	010	110	100	101
4	010	011	001	101	100	110
5	011	010	000	100	101	111
6	010	000	001	101	111	110
7	011	001	000	100	110	111
8	000	001	101	111	110	010
9	001	000	100	110	111	011
10	000	010	110	111	101	001
11	001	011	111	110	100	000
12	010	011	111	101	100	000
13	011	010	110	100	101	001
14	010	000	100	101	111	011
15	011	001	101	100	110	010
16	000	100	101	111	011	010
17	001	101	100	110	010	011
18	000	100	110	111	011	001
19	001	101	111	110	010	000
20	010	110	111	101	001	000
21	011	111	110	100	000	001
22	010	110	100	101	001	011
23	011	111	101	100	000	010
24	100	101	111	011	010	000
25	101	100	110	010	011	001
26	100	110	111	011	001	000
27	101	111	110	010	000	001
28	110	111	101	001	000	010
29	111	110	100	000	001	011
30	110	100	101	001	011	010
31	111	101	100	000	010	011
32	100	101	001	011	010	110
33	101	100	000	010	011	111
34	100	110	010	011	001	101
35	101	111	011	010	000	100
36	110	111	011	001	000	100
37	111	110	010	000	001	101
38	110	100	000	001	011	111
39	111	101	001	000	010	110
40	100	000	001	011	111	110
41	101	001	000	010	110	111
42	100	000	010	011	111	101
43	101	001	011	010	110	100
44	110	010	011	001	101	100
45	111	011	010	000	100	101
46	110	010	000	001	101	111
47	111	011	001	000	100	110

SEE ALSO

bldc_hall6 6-wire unipolar driver for BLDC motors.

AUTHOR

Andy Pugh

LICENSE

GPL

NAME

blend – Perform linear interpolation between two values

SYNOPSIS

loadrt blend [count=N|names=name1[,name2...]]

FUNCTIONS

blend.N (requires a floating-point thread)

PINS

blend.N.in1 float in

First input. If select is equal to 1.0, the output is equal to in1

blend.N.in2 float in

Second input. If select is equal to 0.0, the output is equal to in2

blend.N.select float in

Select input. For values between 0.0 and 1.0, the output changes linearly from in2 to in1

blend.N.out float out

Output value.

PARAMETERS

blend.N.open bit rw

If true, select values outside the range 0.0 to 1.0 give values outside the range in2 to in1. If false, outputs are clamped to the the range in2 to in1

LICENSE

GPL

NAME

charge_pump – Create a square-wave for the 'charge pump' input of some controller boards

SYNOPSIS

loadrt charge_pump

DESCRIPTION

The 'Charge Pump' should be added to the base thread function. When enabled the output is on for one period and off for one period. To calculate the frequency of the output $1/(period\ time\ in\ seconds \times 2) = \text{hz}$. For example if you have a base period of 100,000ns that is 0.0001 seconds and the formula would be $1/(0.0001 \times 2) = 5,000\ \text{hz}$ or 5 KHz. Two additional outputs are provided that run a factor of 2 and 4 slower for hardware that requires a lower frequency.

FUNCTIONS

charge-pump

Toggle the output bit (if enabled)

PINS

charge-pump.out bit out

Square wave if 'enable' is TRUE or unconnected, low if 'enable' is FALSE

charge-pump.out-2 bit out

Square wave at half the frequency of 'out'

charge-pump.out-4 bit out

Square wave at a quarter of the frequency of 'out'

charge-pump.enable bit in (default: *TRUE*)

If FALSE, forces all 'out' pins to be low

LICENSE

GPL

NAME

clarke2 – Two input version of Clarke transform

SYNOPSIS

loadrt clarke2 [count=N|names=name1[,name2...]]

DESCRIPTION

The Clarke transform can be used to translate a vector quantity from a three phase system (three components 120 degrees apart) to a two phase Cartesian system.

clarke2 implements a special case of the Clarke transform, which only needs two of the three input phases. In a three wire three phase system, the sum of the three phase currents or voltages must always be zero. As a result only two of the three are needed to completely define the current or voltage. **clarke2** assumes that the sum is zero, so it only uses phases A and B of the input. Since the H (homopolar) output will always be zero in this case, it is not generated.

FUNCTIONS

clarke2.N (requires a floating-point thread)

PINS

clarke2.N.a float in
clarke2.N.b float in
 first two phases of three phase input
clarke2.N.x float out
clarke2.N.y float out
 cartesian components of output

SEE ALSO

clarke3 for the general case, **clarkeinv** for the inverse transform.

LICENSE

GPL

NAME

clarke3 – Clarke (3 phase to cartesian) transform

SYNOPSIS

loadrt clarke3 [count=N|names=name1[,name2...]]

DESCRIPTION

The Clarke transform can be used to translate a vector quantity from a three phase system (three components 120 degrees apart) to a two phase Cartesian system (plus a homopolar component if the three phases don't sum to zero).

clarke3 implements the general case of the transform, using all three phases. If the three phases are known to sum to zero, see **clarke2** for a simpler version.

FUNCTIONS

clarke3.N (requires a floating-point thread)

PINS

clarke3.N.a float in

clarke3.N.b float in

clarke3.N.c float in

three phase input vector

clarke3.N.x float out

clarke3.N.y float out

cartesian components of output

clarke3.N.h float out

homopolar component of output

SEE ALSO

clarke2 for the 'a+b+c=0' case, **clarkeinv** for the inverse transform.

LICENSE

GPL

NAME

clarkeinv – Inverse Clarke transform

SYNOPSIS

```
loadrt clarkeinv [count=N|names=name1[,name2...]]
```

DESCRIPTION

The inverse Clarke transform can be used rotate a vector quantity and then translate it from Cartesian coordinate system to a three phase system (three components 120 degrees apart).

FUNCTIONS

clarkeinv.N (requires a floating-point thread)

PINS

clarkeinv.N.x float in

clarkeinv.N.y float in

cartesian components of input

clarkeinv.N.h float in

homopolar component of input (usually zero)

clarkeinv.N.theta float in

rotation angle: 0.00 to 1.00 = 0 to 360 degrees

clarkeinv.N.a float out

clarkeinv.N.b float out

clarkeinv.N.c float out

three phase output vector

SEE ALSO

clarke2 and **clarke3** for the forward transform.

LICENSE

GPL

NAME

classicladder – realtime software plc based on ladder logic

SYNOPSIS

```
loadrt classicladder_rt [numRungs=N] [numBits=N] [numWords=N] [numTimers=N] [numMonostables=N] [numCounters=N] [numPhysInputs=N] [numPhysOutputs=N] [numArithmExpr=N] [numSections=N] [numSymbols=N] [numS32in=N] [numS32out=N] [numFloatIn=N] [numFloatOut=N]
```

DESCRIPTION

These pins and parameters are created by the realtime **classicladder_rt** module. Each period (minimum 1000000 ns), classicladder reads the inputs, evaluates the ladder logic defined in the GUI, and then writes the outputs.

PINS**classicladder.0.in-NN IN bit**

These bit signal pins map to **%INN** variables in classicladder

classicladder.0.out-NN OUT bit

These bit signal pins map to **%QNN** variables in classicladder Output from classicladder

classicladder.0.s32in-NN IN s32

Integer input from classicladder These s32 signal pins map to **%IWNN** variables in classicladder

classicladder.0.s32out-NN OUT s32

Integer output from classicladder These s32 signal pins map to **%QWNN** variables in classicladder

classicladder.0.floatin-NN IN float

Integer input from classicladder These float signal pins map to **%IFNN** variables in classicladder
These are truncated to S32 values internally. eg 7.5 will be 7

classicladder.0.floatout-NN OUT float

Float output from classicladder These float signal pins map to **%QFNN** variables in classicladder

classicladder.0.hide_gui IN bit

This bit pin hides the classicladder window, while still having the userspace code run. This is usually desirable when modbus is used, as modbus requires the userspace code to run.

PARAMETERS**classicladder.0.refresh.time** RO s32

Tells you how long the last refresh took

classicladder.0.refresh.tmax RW s32

Tells you how long the longest refresh took

classicladder.0.ladder-state RO s32

Tells you if the program is running or not

FUNCTIONS

classicladder.0.refresh FP

The rung update rate. Add this to the servo thread. You can added it to a faster thread but it Will update no faster than once every 1 millisecond (1000000 ns).

BUGS

See http://wiki.linuxcnc.org/cgi-bin/wiki.pl?ClassicLadder_Ver_7.124 for the latest.

SEE ALSO

Classicladder chapters in the LinuxCNC documentation for a full description of the **Classicladder** syntax and examples

http://wiki.linuxcnc.org/cgi-bin/wiki.pl?ClassicLadder_Ver_7.124

NAME

comp – Two input comparator with hysteresis

SYNOPSIS

loadrt comp [count=N|names=name1[,name2...]]

FUNCTIONS

comp.N (requires a floating-point thread)

Update the comparator

PINS

comp.N.in0 float in

Inverting input to the comparator

comp.N.in1 float in

Non-inverting input to the comparator

comp.N.out bit out

Normal output. True when **in1 > in0** (see parameter **hyst** for details)

comp.N.equal bit out

Match output. True when difference between **in1** and **in0** is less than **hyst/2**

PARAMETERS

comp.N.hyst float rw (default: 0.0)

Hysteresis of the comparator (default 0.0)

With zero hysteresis, the output is true when **in1 > in0**. With nonzero hysteresis, the output switches on and off at two different values, separated by distance **hyst** around the point where **in1 = in0**. Keep in mind that floating point calculations are never absolute and it is wise to always set **hyst** if you intend to use equal

LICENSE

GPL

CONSTANT(9)

HAL Component

CONSTANT(9)

NAME

constant – Use a parameter to set the value of a pin

SYNOPSIS

loadrt constant [count=N|names=name1[,name2...]]

FUNCTIONS

constant.N (requires a floating-point thread)

PINS

constant.N.out float out

PARAMETERS

constant.N.value float rw

LICENSE

GPL

NAME

conv_bit_s32 – Convert a value from bit to s32

SYNOPSIS

loadrt conv_bit_s32 [count=N|names=name1[,name2...]]

FUNCTIONS

conv-bit-s32.N

Update 'out' based on 'in'

PINS

conv-bit-s32.N.in bit in

conv-bit-s32.N.out s32 out

LICENSE

GPL

NAME

conv_bit_u32 – Convert a value from bit to u32

SYNOPSIS

loadrt conv_bit_u32 [count=N|names=name1[,name2...]]

FUNCTIONS

conv-bit-u32.N

Update 'out' based on 'in'

PINS

conv-bit-u32.N.in bit in

conv-bit-u32.N.out u32 out

LICENSE

GPL

NAME

conv_float_s32 – Convert a value from float to s32

SYNOPSIS

loadrt conv_float_s32 [count=N|names=name1[,name2...]]

FUNCTIONS

conv-float-s32.N (requires a floating-point thread)

Update 'out' based on 'in'

PINS

conv-float-s32.N.in float in

conv-float-s32.N.out s32 out

conv-float-s32.N.out-of-range bit out

TRUE when 'in' is not in the range of s32

PARAMETERS

conv-float-s32.N.clamp bit rw

If TRUE, then clamp to the range of s32. If FALSE, then allow the value to "wrap around".

LICENSE

GPL

NAME

conv_float_u32 – Convert a value from float to u32

SYNOPSIS

loadrt conv_float_u32 [count=N|names=name1[,name2...]]

FUNCTIONS

conv-float-u32.N (requires a floating-point thread)

Update 'out' based on 'in'

PINS

conv-float-u32.N.in float in

conv-float-u32.N.out u32 out

conv-float-u32.N.out-of-range bit out

TRUE when 'in' is not in the range of u32

PARAMETERS

conv-float-u32.N.clamp bit rw

If TRUE, then clamp to the range of u32. If FALSE, then allow the value to "wrap around".

LICENSE

GPL

NAME

conv_s32_bit – Convert a value from s32 to bit

SYNOPSIS

loadrt conv_s32_bit [count=N|names=name1[,name2...]]

FUNCTIONS

conv-s32-bit.N

Update 'out' based on 'in'

PINS

conv-s32-bit.N.in s32 in

conv-s32-bit.N.out bit out

conv-s32-bit.N.out-of-range bit out

TRUE when 'in' is not in the range of bit

PARAMETERS

conv-s32-bit.N.clamp bit rw

If TRUE, then clamp to the range of bit. If FALSE, then allow the value to "wrap around".

LICENSE

GPL

NAME

conv_s32_float – Convert a value from s32 to float

SYNOPSIS

loadrt conv_s32_float [count=N|names=name1[,name2...]]

FUNCTIONS

conv-s32-float.N (requires a floating-point thread)

Update 'out' based on 'in'

PINS

conv-s32-float.N.in s32 in

conv-s32-float.N.out float out

LICENSE

GPL

NAME

conv_s32_u32 – Convert a value from s32 to u32

SYNOPSIS

loadrt conv_s32_u32 [count=N|names=name1[,name2...]]

FUNCTIONS

conv-s32-u32.N

Update 'out' based on 'in'

PINS

conv-s32-u32.N.in s32 in

conv-s32-u32.N.out u32 out

conv-s32-u32.N.out-of-range bit out

TRUE when 'in' is not in the range of u32

PARAMETERS

conv-s32-u32.N.clamp bit rw

If TRUE, then clamp to the range of u32. If FALSE, then allow the value to "wrap around".

LICENSE

GPL

NAME

conv_u32_bit – Convert a value from u32 to bit

SYNOPSIS

loadrt conv_u32_bit [count=N|names=name1[,name2...]]

FUNCTIONS

conv-u32-bit.N

Update 'out' based on 'in'

PINS

conv-u32-bit.N.in u32 in

conv-u32-bit.N.out bit out

conv-u32-bit.N.out-of-range bit out

TRUE when 'in' is not in the range of bit

PARAMETERS

conv-u32-bit.N.clamp bit rw

If TRUE, then clamp to the range of bit. If FALSE, then allow the value to "wrap around".

LICENSE

GPL

NAME

conv_u32_float – Convert a value from u32 to float

SYNOPSIS

loadrt conv_u32_float [count=N|names=name1[,name2...]]

FUNCTIONS

conv-u32-float.N (requires a floating-point thread)

Update 'out' based on 'in'

PINS

conv-u32-float.N.in u32 in

conv-u32-float.N.out float out

LICENSE

GPL

NAME

conv_u32_s32 – Convert a value from u32 to s32

SYNOPSIS

loadrt conv_u32_s32 [count=N|names=name1[,name2...]]

FUNCTIONS

conv-u32-s32.N

Update 'out' based on 'in'

PINS

conv-u32-s32.N.in u32 in

conv-u32-s32.N.out s32 out

conv-u32-s32.N.out-of-range bit out

TRUE when 'in' is not in the range of s32

PARAMETERS

conv-u32-s32.N.clamp bit rw

If TRUE, then clamp to the range of s32. If FALSE, then allow the value to "wrap around".

LICENSE

GPL

NAME

counter – counts input pulses (**DEPRECATED**)

SYNOPSIS

loadrt counter [num_chan=N]

DESCRIPTION

counter is a deprecated HAL component and will be removed in a future release. Use the **encoder** component with encoder.X.counter-mode set to TRUE.

counter is a HAL component that provides software- based counting that is useful for spindle position sensing and maybe other things. Instead of using a real encoder that outputs quadrature, some lathes have a sensor that generates a simple pulse stream as the spindle turns and an index pulse once per revolution. This component simply counts up when a "count" pulse (phase-A) is received, and if reset is enabled, resets when the "index" (phase-Z) pulse is received.

This is of course only useful for a unidirectional spindle, as it is not possible to sense the direction of rotation.

counter conforms to the "canonical encoder" interface described in the HAL manual.

FUNCTIONS

counter.capture-position (uses floating-point)

Updates the counts, position and velocity outputs based on internal counters.

counter.update-counters

Samples the phase-A and phase-Z inputs and updates internal counters.

PINS

counter.N.phase-A bit in

The primary input signal. The internal counter is incremented on each rising edge.

counter.N.phase-Z bit in

The index input signal. When the **index-enable** pin is TRUE and a rising edge on **phase-Z** is seen, **index-enable** is set to FALSE and the internal counter is reset to zero.

counter.N.index-enable bit io

counter.N.reset bit io

counter.N.counts signed out

counter.N.position float out

counter.N.velocity float out

These pins function according to the canonical digital encoder interface.

counter.N.position-scale float rw

This parameter functions according to the canonical digital encoder interface.

counter.N.rawcounts signed ro

The internal counts value, updated from **update-counters** and reflected in the output pins at the next call to **capture-position**.

SEE ALSO

encoder(9). in the LinuxCNC documentation.

NAME

ddt – Compute the derivative of the input function

SYNOPSIS

loadrt ddt [count=N|names=name1[,name2...]]

FUNCTIONS

ddt.N (requires a floating-point thread)

PINS

ddt.N.in float in

ddt.N.out float out

LICENSE

GPL

NAME

deadzone – Return the center if within the threshold

SYNOPSIS

loadrt deadzone [count=N|names=name1[,name2...]]

FUNCTIONS

deadzone.N (requires a floating-point thread)

Update **out** based on **in** and the parameters.

PINS

deadzone.N.in float in

deadzone.N.out float out

PARAMETERS

deadzone.N.center float rw (default: *0.0*)

The center of the dead zone

deadzone.N.threshold float rw (default: *1.0*)

The dead zone is **center** \pm (**threshold**/2)

LICENSE

GPL

NAME

debounce – filter noisy digital inputs

SYNOPSIS

loadrt debounce cfg=*size1,size2,...*

Creates debounce groups with the number of filters specified by (*size*). Every filter in the same group has the same sample rate and delay. For example cfg=2,3 creates two filter groups with 2 filters in the first group and 3 filters in the second group.

DESCRIPTION

The debounce filter works by incrementing a counter whenever the input is true, and decrementing the counter when it is false. If the counter decrements to zero, the output is set false and the counter ignores further decrements. If the counter increments up to a threshold, the output is set true and the counter ignores further increments. If the counter is between zero and the threshold, the output retains its previous state. The threshold determines the amount of filtering: a threshold of 1 does no filtering at all, and a threshold of N requires a signal to be present for N samples before the output changes state.

FUNCTIONS

debounce.*G*

Sample all the input pins in group G and update the output pins.

PINS

debounce.*G.F.in* bit in

The F'th input pin in group G.

debounce.*G.F.out* bit out

The F'th output pin in group G. Reflects the last "stable" input seen on the corresponding input pin.

debounce.*G.delay* signed rw

Sets the amount of filtering for all pins in group G.

NAME

edge – Edge detector

SYNOPSIS

loadrt edge [count=N|names=name1[,name2...]]

FUNCTIONS

edge.N Produce output pulses from input edges

PINS

edge.N.in bit in

edge.N.out bit out

Goes high when the desired edge is seen on 'in'

edge.N.out-invert bit out

Goes low when the desired edge is seen on 'in'

PARAMETERS

edge.N.both bit rw (default: *FALSE*)

If TRUE, selects both edges. Otherwise, selects one edge according to in-edge

edge.N.in-edge bit rw (default: *TRUE*)

If both is FALSE, selects the one desired edge: TRUE means falling, FALSE means rising

edge.N.out-width-ns s32 rw (default: 0)

Time in nanoseconds of the output pulse

edge.N.time-left-ns s32 r

Time left in this output pulse

edge.N.last-in bit r

Previous input value

LICENSE

GPL

NAME

encoder – software counting of quadrature encoder signals

SYNOPSIS

```
loadrt encoder [num_chan=num | names=name1[,name2...]]
```

DESCRIPTION

encoder is used to measure position by counting the pulses generated by a quadrature encoder. As a software-based implementation it is much less expensive than hardware, but has a limited maximum count rate. The limit is in the range of 10KHz to 50KHz, depending on the computer speed and other factors. If better performance is needed, a hardware encoder counter is a better choice. Some hardware-based systems can count at MHz rates.

encoder supports a maximum of eight channels. The number of channels actually loaded is set by the **num_chan** argument when the module is loaded. Alternatively, specify **names=** and unique names separated by commas.

The **num_chan=** and **names=** specifiers are mutually exclusive. If neither **num_chan=** nor **names=** are specified, or if **num_chan=0** is specified, the default value is three.

encoder has a one-phase, unidirectional mode called *counter*. In this mode, the **phase-B** input is ignored; the counts increase on each rising edge of **phase-A**. This mode may be useful for counting a unidirectional spindle with a single input line, though the noise-resistant characteristics of quadrature are lost.

FUNCTIONS

encoder.update-counters (no floating-point)

Does the actual counting, by sampling the encoder signals and decoding the quadrature waveforms. Must be called as frequently as possible, preferably twice as fast as the maximum desired count rate. Operates on all channels at once.

encoder.capture-position (uses floating point)

Captures the raw counts from **update-counters** and performs scaling and other necessary conversion, handles counter rollover, etc. Can (and should) be called less frequently than **update-counters**. Operates on all channels at once.

NAMING

The names for pins and parameters are prefixed as:

encoder.N. for N=0,1,...,num-1 when using **num_chan=num**

nameN. for nameN=name1,name2,... when using **names=name1,name2,...**

The **encoder.N.** format is shown in the following descriptions.

PINS

encoder.N.counter-mode bit i/o

Enables counter mode. When true, the counter counts each rising edge of the phase-A input, ignoring the value on phase-B. This is useful for counting the output of a single channel (non-quadrature) sensor. When false (the default), it counts in quadrature mode.

encoder.N.counts s32 out

Position in encoder counts.

encoder.N.index-enable bit i/o

When true, **counts** and **position** are reset to zero on the next rising edge of **Phase-Z**. At the same time, **index-enable** is reset to zero to indicate that the rising edge has occurred.

encoder.N.phase-A bit in
 Quadrature input for encoder channel *N*.

encoder.N.phase-B bit in
 Quadrature input.

encoder.N.phase-Z bit in
 Index pulse input.

encoder.N.position float out
 Position in scaled units (see **position-scale**)

encoder.N.position-interpolated float out
 Position in scaled units, interpolated between encoder counts. Only valid when velocity is approximately constant and above **min-velocity-estimate**. Do not use for position control.

encoder.N.position-scale float i/o
 Scale factor, in counts per length unit. For example, if **position-scale** is 500, then 1000 counts of the encoder will be reported as a position of 2.0 units.

encoder.N.rawcounts s32 out
 The raw count, as determined by **update-counters**. This value is updated more frequently than **counts** and **position**. It is also unaffected by **reset** or the index pulse.

encoder.N.reset bit in
 When true, **counts** and **position** are reset to zero immediately.

encoder.N.velocity float out
 Velocity in scaled units per second. **encoder** uses an algorithm that greatly reduces quantization noise as compared to simply differentiating the **position** output. When the magnitude of the true velocity is below min-velocity-estimate, the velocity output is 0.

encoder.N.x4-mode bit i/o
 Enables times-4 mode. When true (the default), the counter counts each edge of the quadrature waveform (four counts per full cycle). When false, it only counts once per full cycle. In **counter-mode**, this parameter is ignored.

encoder.N.latch-input bit in

encoder.N.latch-falling bit in (default: **TRUE**)

encoder.N.latch-rising bit in (default: **TRUE**)

encoder.N.counts-latched s32 out

encoder.N.position-latched float out
 Update **counts-latched** and **position-latched** on the rising and/or falling edges of **latch-input** as indicated by **latch-rising** and **latch-falling**.

encoder.N.counter-mode bit rw
 Enables counter mode. When true, the counter counts each rising edge of the phase-A input, ignoring the value on phase-B. This is useful for counting the output of a single channel (non-quadrature) sensor. When false (the default), it counts in quadrature mode. **encoder.N.capture-position.tmax** s32 rw Maximum number of CPU cycles it took to execute this function.

PARAMETERS

Parameter names for num_chan= specifier are:

encoder.N.the_parameter_name

Parameter names for names= specifier are:

nameN.the_parameter_name

encoder.N.min-velocity-estimate float rw (default: 1.0)

Determine the minimum true velocity magnitude at which **velocity** will be estimated as nonzero and **position-interpolated** will be interpolated. The units of **min-velocity-estimate** are the same as the units of **velocity**. Setting this parameter too low will cause it to take a long time for **velocity**

ENCODER(9)

HAL Component

ENCODER(9)

to go to 0 after encoder pulses have stopped arriving.

NAME

encoder_ratio – an electronic gear to synchronize two axes

SYNOPSIS

```
loadrt encoder_ratio [num_chan=num | names=name1[,name2...]]
```

DESCRIPTION

encoder_ratio can be used to synchronize two axes (like an "electronic gear"). It counts encoder pulses from both axes in software, and produces an error value that can be used with a PID loop to make the slave encoder track the master encoder with a specific ratio.

This module supports up to eight axis pairs. The number of pairs is set by the module parameter **num_chan**. Alternatively, specify **names=** and unique names separated by commas.

The **num_chan=** and **names=** specifiers are mutually exclusive. If neither **num_chan=** nor **names=** are specified, the default value is one.

FUNCTIONS**encoder-ratio.sample**

Read all input pins. Must be called at twice the maximum desired count rate.

encoder-ratio.update (uses floating-point)

Updates all output pins. May be called from a slower thread.

NAMING

The names for pins and parameters are prefixed as:

encoder-ratio.N. for N=0,1,...,num-1 when using **num_chan=num**

nameN. for nameN=name1,name2,... when using **names=name1,name2,...**

The **encoder-ratio.N.** format is shown in the following descriptions.

PINS**encoder-ratio.N.master-A** bit in**encoder-ratio.N.master-B** bit in**encoder-ratio.N.slave-A** bit in**encoder-ratio.N.slave-B** bit in

The encoder channels of the master and slave axes

encoder-ratio.N.enable bit in

When the enable pin is FALSE, the error pin simply reports the slave axis position, in revolutions. As such, it would normally be connected to the feedback pin of a PID block for closed loop control of the slave axis. Normally the command input of the PID block is left unconnected (zero), so the slave axis simply sits still. However when the enable input goes TRUE, the error pin becomes the slave position minus the scaled master position. The scale factor is the ratio of master teeth to slave teeth. As the master moves, error becomes non-zero, and the PID loop will drive the slave axis to track the master.

encoder-ratio.N.error float out

The error in the position of the slave (in revolutions)

PARAMETERS**encoder-ratio.N.master-ppr** unsigned rw**encoder-ratio.N.slave-ppr** unsigned rw

The number of pulses per revolution of the master and slave axes

encoder-ratio.N.master-teeth unsigned rw

encoder-ratio.N.slave-teeth unsigned rw

The number of "teeth" on the master and slave gears.

SEE ALSO

[encoder\(9\)](#)

NAME

`estop_latch` – Software ESTOP latch

SYNOPSIS

`loadrt estop_latch [count=N|names=name1[,name2...]]`

DESCRIPTION

This component can be used as a part of a simple software ESTOP chain.

It has two states: "OK" and "Faulted".

The initial state is "Faulted". When faulted, the **out-ok** output is false, the **fault-out** output is true, and the **watchdog** output is unchanging.

The state changes from "Faulted" to "OK" when **all** these conditions are true:

- **fault-in** is false
- **ok-in** is true
- **reset** changes from false to true

When "OK", the **out-ok** output is true, the **fault-out** output is false, and the **watchdog** output is toggling.

The state changes from "OK" to "Faulted" when **any** of the following are true:

- **fault-in** is true
- **ok-in** is false

To facilitate using only a single fault source, **ok-in** and **fault-en** are both set to the non-fault-causing value when no signal is connected. For estop-latch to ever be able to signal a fault, at least one of these inputs must be connected.

Typically, an external fault or estop input is connected to **fault-in**, **iocontrol.0.user-request-enable** is connected to **reset**, and **ok-out** is connected to **iocontrol.0.emc-enable-in**.

In more complex systems, it may be more appropriate to use classicladder to manage the software portion of the estop chain.

FUNCTIONS

`estop-latch.N`

PINS

estop-latch.N.ok-in bit in (default: *true*)
estop-latch.N.fault-in bit in (default: *false*)
estop-latch.N.reset bit in
estop-latch.N.ok-out bit out (default: *false*)
estop-latch.N.fault-out bit out (default: *true*)
estop-latch.N.watchdog bit out

LICENSE

GPL

NAME

feedcomp – Multiply the input by the ratio of current velocity to the feed rate

SYNOPSIS

loadrt feedcomp [count=N|names=name1[,name2...]]

FUNCTIONS

feedcomp.N (requires a floating-point thread)

PINS

feedcomp.N.out float out

Proportionate output value

feedcomp.N.in float in

Reference value

feedcomp.N.enable bit in

Turn compensation on or off

feedcomp.N.vel float in

Current velocity

PARAMETERS

feedcomp.N.feed float rw

Feed rate reference value

NOTES

Note that if enable is false, out = in

LICENSE

GPL

NAME

flipflop – D type flip-flop

SYNOPSIS

loadrt flipflop [count=N|names=name1[,name2...]]

FUNCTIONS

flipflop.N

PINS

flipflop.N.data bit in
data input

flipflop.N.clk bit in
clock, rising edge writes data to out

flipflop.N.set bit in
when true, force out true

flipflop.N.reset bit in
when true, force out false; overrides set

flipflop.N.out bit io
output

LICENSE

GPL

NAME

`gantrykins` – A kinematics module that maps one axis to multiple joints

SYNOPSIS

`loadrt gantrykins coordinates=axisletters`

Specifying gantry joint mapping via loadrt

The **coordinates=** parameter specifies the initial gantry joint mapping. Each axis letter is mapped to a joint, starting from 0. So **coordinates=XYYZ** maps the X axis to joint 0, the Y axis to joint 1 and 2, and the Z axis to joint 3. If not specified, the default mapping is **coordinates=XYZABC**. Coordinate letters may be specified in uppercase or lowercase.

A note about joints and axes

LinuxCNC makes a distinction between joints and axes: a joint is something controlled by a motor, and an axis is a coordinate you can move via G-code. You can also jog joints or jog axes.

A gantry has two joints controlling one axis, and this requires a bit of special care.

Homing always happens in joint mode (aka Free mode). The two joints of a gantry's axis must be homed together, so they must have the same [AXIS_n]HOME_SEQUENCE in the .ini file.

Jogging of a gantry must happen in world mode (aka Teleop mode). If you jog a gantry in joint mode (Free mode), you will move just one of the joints, and the gantry will rack. In contrast, if you jog a gantry in world mode (Teleop mode), it's the axis that jogs: linuxcnc will coordinate the motion of the two joints that make up the axis, both joints will move together, and the gantry will stay square.

The Axis GUI has provisions for jogging in joint mode (Free) and in world mode (Teleop). Use the "\$" hotkey, or the View menu to switch between them.

Joint-mode (aka Free mode) supports continuous and incremental jogging. World-mode (aka Teleop mode) only supports continuous jogging.

KINEMATICS

In the inverse kinematics, each joint gets the value of its corresponding axis. In the forward kinematics, each axis gets the value of the highest numbered corresponding joint. For example, with **coordinates=XYYZ** the Y axis position comes from joint 2, not joint 1.

FUNCTIONS

None.

PINS

None.

PARAMETERS

gantrykins.joint-*N* (s32)

Specifies the axis mapped to joint *N*. The values 0 through 8 correspond to the axes XYZ-ABCUVW. It is preferable to use the "coordinates=" parameter at loadrt-time rather than setting the joint-*N* parameters later, because the gantrykins module prints the joint-to-axis mapping at loadrt-time, and having that output correct is nice.

NOTES

gantrykins must be loaded before **motion**.

SEE ALSO

Kinematics section in the LinuxCNC documentation

LICENSE

GPL

NAME

gearchange – Select from one two speed ranges

SYNOPSIS

The output will be a value scaled for the selected gear, and clamped to the min/max values for that gear. The scale of gear 1 is assumed to be 1, so the output device scale should be chosen accordingly. The scale of gear 2 is relative to gear 1, so if gear 2 runs the spindle 2.5 times as fast as gear 1, scale2 should be set to 2.5.

FUNCTIONS

gearchange.N (requires a floating-point thread)

PINS

gearchange.N.sel bit in

Gear selection input

gearchange.N.speed-in float in

Speed command input

gearchange.N.speed-out float out

Speed command to DAC/PWM

gearchange.N.dir-in bit in

Direction command input

gearchange.N.dir-out bit out

Direction output - possibly inverted for second gear

PARAMETERS

gearchange.N.min1 float rw (default: 0)

Minimum allowed speed in gear range 1

gearchange.N.max1 float rw (default: 100000)

Maximum allowed speed in gear range 1

gearchange.N.min2 float rw (default: 0)

Minimum allowed speed in gear range 2

gearchange.N.max2 float rw (default: 100000)

Maximum allowed speed in gear range 2

gearchange.N.scale2 float rw (default: 1.0)

Relative scale of gear 2 vs. gear 1 Since it is assumed that gear 2 is "high gear", **scale2** must be greater than 1, and will be reset to 1 if set lower.

gearchange.N.reverse bit rw (default: 0)

Set to 1 to reverse the spindle in second gear

LICENSE

GPL

NAME

gladevcp – displays Virtual control Panels built with GTK / GLADE

SYNOPSIS

```
loadusr gladevcp [-c componentname0xN] [-g WxH+Xoffset+Yoffset0xN] [-H halcmdfile] [-x windowid] gladefile.glade
```

DESCRIPTION

gladevcp parses a glade file and displays the widgets in a window. Then calls gladevcp_makepins which again parses the gladefile looking for specific HAL widgets then makes HAL pins and sets up updating for them. The HAL component name defaults to the basename of the glade file. The -x option directs gladevcp to reparent itself under this X window id instead of creating its own toplevel window. The -H option passes an input file for halcmd to be run after the gladevcp component is initialized. This is used in Axis when running gladevcp under a tab with the EMBED_TAB_NAME/EMBED_TAB_COMMAND ini file feature.

gladevcp supports gtkbuilder or libglade files though some widgets are not fully supported in gtkbuilder yet.

ISSUES

For now system links need to be added in the glade library folders to point to our new widgets and catalog files. look in lib/python/gladevcp/READ_ME for details

NAME

gray2bin – convert a gray-code input to binary

SYNOPSIS

loadrt gray2bin [count=N|names=name1[,name2...]]

DESCRIPTION

Converts a gray-coded number into the corresponding binary value

FUNCTIONS

gray2bin.N

PINS

gray2bin.N.in u32 in

gray code in

gray2bin.N.out u32 out

binary code out

AUTHOR

andy pugh

LICENSE

GPL

NAME

hm2_7i43 – LinuxCNC HAL driver for the Mesa Electronics 7i43 EPP Anything IO board with HostMot2 firmware.

SYNOPSIS

```
loadrt hm2_7i43 [ioaddr=N[,N...]] [ioaddr_hi=N[,N...]] [epp_wide=N[,N...]] [config="str[,str...]"]
[debug_epp=N[,N...]]
```

ioaddr [default: 0x378]
 The base address of the parallel port.

ioaddr_hi [default: 0]
 The secondary address of the parallel port, used to set EPP mode. 0 means to use ioaddr + 0x400.

epp_wide [default: 1]
 Set to zero to disable the "wide EPP mode". "Wide" mode allows a 16- and 32-bit EPP transfers, which can reduce the time spent in the read and write functions. However, this may not work on all EPP parallel ports.

config [default: ""]
 HostMot2 config strings, described in the hostmot2(9) manpage.

debug_epp [default: 0]
 Developer/debug use only! Enable debug logging of most EPP transfers.

DESCRIPTION

hm2_7i43 is a device driver that interfaces the Mesa 7i43 board with the HostMot2 firmware to the LinuxCNC HAL. Both the 200K and the 400K FPGAs are supported.

The driver talks with the 7i43 over the parallel port, not over USB. USB can be used to power the 7i43, but not to talk to it. USB communication with the 7i43 will not be supported any time soon, since USB has poor real-time qualities.

The driver programs the board's FPGA with firmware when it registers the board with the hostmot2 driver. The old bfload(1) firmware loading method is not used anymore. Instead the firmware to load is specified in the **config** modparam, as described in the hostmot2(9) manpage, in the *config modparam* section.

Some parallel ports require special initialization before they can be used. LinuxCNC provides a kernel driver that does this initialization called `probe_parport`. Load this driver before loading `hm2_7i43`, by putting "loadrt `probe_parport`" in your .hal file.

Jumper settings

To send the FPGA configuration from the PC, the board must be configured to get its firmware from the EPP port. To do this, jumpers W4 and W5 must both be down, ie toward the USB connector.

The board must be configured to power on whether or not the USB interface is active. This is done by setting jumper W7 up, ie away from the edge of the board.

Communicating with the board

The 7i43 communicates with the LinuxCNC computer over EPP, the Enhanced Parallel Port. This provides about 1 MBps of throughput, and the communication latency is very predictable and reasonably low.

The parallel port must support EPP 1.7 or EPP 1.9. EPP 1.9 is preferred, but EPP 1.7 will work too. The EPP mode of the parallel port is sometimes a setting in the BIOS.

Note that the popular "NetMOS" aka "MosChip 9805" PCI parport cards **do not work**. They do not meet the EPP spec, and cannot be reliably used with the 7i43. You have to find another card, sorry.

EPP is very reliable under normal circumstances, but bad cabling or excessively long cabling runs may

cause communication timeouts. The driver exports a parameter named `hm2_7i43.<BoardNum>.io_error` to inform HAL of this condition. When the driver detects an EPP timeout, it sets `io_error` to True and stops communicating with the 7i43 board. Setting `io_error` back to False makes the driver start trying to communicate with the 7i43 again.

Access to the EPP bus is not threadsafe: only one realtime thread may access the EPP bus.

SEE ALSO

`hostmot2(9)`

LICENSE

GPL

NAME

hm2_7i90 – LinuxCNC HAL driver for the Mesa Electronics 7i90 EPP Anything IO board with HostMot2 firmware.

SYNOPSIS

```
loadrt hm2_7i90 [ioaddr=N[,N...]] [ioaddr_hi=N[,N...]] [epp_wide=N[,N...]] [debug_epp=N[,N...]]
```

ioaddr [default: 0x378]

The base address of the parallel port.

ioaddr_hi [default: 0]

The secondary address of the parallel port, used to set EPP mode. 0 means to use ioaddr + 0x400.

epp_wide [default: 1]

Set to zero to disable the "wide EPP mode". "Wide" mode allows a 16- and 32-bit EPP transfers, which can reduce the time spent in the read and write functions. However, this may not work on all EPP parallel ports.

debug_epp [default: 0]

Developer/debug use only! Enable debug logging of most EPP transfers.

DESCRIPTION

hm2_7i90 is a device driver that interfaces the Mesa 7i90 board with the HostMot2 firmware to the LinuxCNC HAL.

The 7i90 firmware is fixed, it is not programmed by the driver at load time.

The driver talks with the 7i90 over the parallel port, via EPP.

Some parallel ports require special initialization before they can be used. LinuxCNC provides a kernel driver that does this initialization called `probe_parport`. Load this driver before loading hm2_7i90, by putting "loadrt probe_parport" in your .hal file.

The hm2_7i90 driver requires an in-kernel EPP communications API. This is provided by the '`epp`' driver. Load this driver before loading hm2_7i90, by putting 'loadrt epp' in your .hal file.

Communicating with the board

The 7i90 communicates with the LinuxCNC computer over EPP, the Enhanced Parallel Port. This provides about 1 MBps of throughput, and the communication latency is very predictable and reasonably low.

The parallel port must support EPP 1.7 or EPP 1.9. EPP 1.9 is preferred, but EPP 1.7 will work too. The EPP mode of the parallel port is sometimes a setting in the BIOS.

Note that the popular "NetMOS" aka "MosChip 9805" PCI parport cards **do not work**. They do not meet the EPP spec, and cannot be reliably used with the 7i90. You have to find another card, sorry.

EPP is very reliable under normal circumstances, but bad cabling or excessively long cabling runs may cause communication timeouts. The driver exports a parameter named `hm2_7i90.<BoardNum>.io_error` to inform HAL of this condition. When the driver detects an EPP timeout, it sets `io_error` to True and stops communicating with the 7i90 board. Setting `io_error` back to False makes the driver start trying to communicate with the 7i90 again.

Access to the EPP bus is not threadsafe: only one realtime thread may access the EPP bus.

SEE ALSO

`hostmot2(9)`

LICENSE
GPL

NAME

hm2_pci – LinuxCNC HAL driver for the Mesa Electronics PCI-based Anything IO boards, with HostMot2 firmware.

SYNOPSIS

```
loadrt hm2_pci [config="str[,str...]" ]
```

config [default: ""]

HostMot2 config strings, described in the hostmot2(9) manpage.

DESCRIPTION

hm2_pci is a device driver that interfaces Mesa's PCI and PC-104/Plus based Anything I/O boards (with the HostMot2 firmware) to the LinuxCNC HAL.

The supported boards are: the 5i20, 5i21, 5i22, 5i23, 5i24, and 5i25 (all on PCI); the 4i65, 4i68, and 4i69 (on PC-104/Plus), and the 3x20 (using a 6i68 or 7i68 carrier card) and 6i25 (on PCI Express).

The driver optionally programs the board's FPGA with firmware when it registers the board with the hostmot2 driver. The firmware to load is specified in the **config** modparam, as described in the hostmot2(9) manpage, in the *config modparam* section.

SEE ALSO

hostmot2(9)

LICENSE

GPL

NAME

hostmot2 – LinuxCNC HAL driver for the Mesa Electronics HostMot2 firmware.

SYNOPSIS

See the config modparam section below for Mesa card configuration. Typically hostmot2 is loaded with no parameters unless debugging is required.

```
loadrt hostmot2 [debug_idrom=N] [debug_module_descriptors=N] [debug_pin_descriptors=N]
[debug_modules=N]

debug_idrom [default: 0]
Developer/debug use only! Enable debug logging of the HostMot2 IDROM header.

debug_module_descriptors [default: 0]
Developer/debug use only! Enable debug logging of the HostMot2 Module Descriptors.

debug_pin_descriptors [default: 0]
Developer/debug use only! Enable debug logging of the HostMot2 Pin Descriptors.

debug_modules [default: 0]
Developer/debug use only! Enable debug logging of the HostMot2 Modules used.
```

DESCRIPTION

hostmot2 is a device driver that interfaces the Mesa HostMot2 firmware to the LinuxCNC HAL. This driver by itself does nothing, the boards that actually run the firmware require their own drivers before anything can happen. Currently drivers are available for the 5i20, 5i22, 5i23, 5i25, 3x20, 4i65, and 4i68 (all using the hm2_pci module) and the 7i43 (using the hm2_7i43 module).

The HostMot2 firmware provides modules such as encoders, PWM generators, step/dir generators, and general purpose I/O pins (GPIOs). These things are called "Modules". The firmware is configured, at firmware compile time, to provide zero or more instances of each of these Modules.

Board I/O Pins

The HostMot2 firmware runs on an FPGA board. The board interfaces with the computer via PCI, PC-104/Plus, or EPP, and interfaces with motion control hardware such as servos and stepper motors via I/O pins on the board.

Each I/O pin can be configured, at board-driver load time, to serve one of two purposes: either as a particular I/O pin of a particular Module instance (encoder, pwmgen, stepgen etc), or as a general purpose digital I/O pin. By default all Module instances are enabled, and all the board's pins are used by the Module instances.

The user can disable Module instances at board-driver load time, by specifying a hostmot2 config string modparam. Any pins which belong to Module instances that have been disabled automatically become GPIOs.

All IO pins have some HAL presence, whether they belong to an active module instance or are full GPIOs. GPIOs can be changed (at run-time) between inputs, normal outputs, and open drains, and have a flexible HAL interface. IO pins that belong to active Module instances are constrained by the requirements of the owning Module, and have a more limited interface in HAL. This is described in the General Purpose I/O section below.

config modparam

All the board-driver modules (hm2_pci and hm2_7i43) accept a load-time modparam of type string array, named "config". This array has one config string for each board the driver should use. Each board's config string is passed to and parsed by the hostmot2 driver when the board-driver registers the board.

The config string can contain spaces, so it is usually a good idea to wrap the whole thing in double-quotes (the " character).

The comma character (,) separates members of the config array from each other.

For example, if your control computer has one 5i20 and one 5i23 you might load the hm2_pci driver with a HAL command (in halcmd) something like this:

```
loadrt hm2_pci config="firmware=hm2/5i20/SVST8_4.BIT num_encoders=3 num_pwmgens=3 num_stepgens=3,firmware=hm2/5i20/SVST8_4.BIT num_encoders=3 num_pwmgens=3 num_stepgens=3,firmware=hm2/5i20/SVST8_4.BIT num_encoders=3 num_pwmgens=3 num_stepgens=3"
```

Note: this assumes that the hm2_pci driver detects the 5i20 first and the 5i23 second. If the detection order does not match the order of the config strings, the hostmot2 driver will refuse to load the firmware and the board-driver (hm2_pci or hm2_7i43) will fail to load. To the best of my knowledge, there is no way to predict the order in which PCI boards will be detected by the driver, but the detection order will be consistent as long as PCI boards are not moved around. Best to try loading it and see what the detection order is.

The valid entries in the format string are:

```
[firmware=F]
[num_encoders=N]
[ssi_chan_N=abc%onq]
[biss_chan_N=abc%onq]
[fanuc_chan_N=abc%onq]
[num_resolvers=N]
[num_pwmgens=N]
[num_3pwmgens=N]
[num_stepgens=N]
[stepgen_width=N]
[sserial_port_0=00000000]
[num_leds=N]
[enable raw]
```

firmware [optional]

Load the firmware specified by F into the FPGA on this board. If no "**firmware=F**" string is specified, the FPGA will not be re-programmed but may continue to run a previously downloaded firmware.

The requested firmware F is fetched by udev. udev searches for the firmware in the system's firmware search path, usually /lib/firmware. F typically has the form "hm2/<BoardType>/file.bit"; a typical value for F might be "hm2/5i20/SVST8_4.BIT". The hostmot2 firmware files are supplied by the hostmot2-firmware packages, available from linuxcnc.org and can normally be installed by entering the command "sudo apt-get install hostmot2-firmware-5i23" to install the support files for the 5i23 for example.

The 5i25 / 6i25 come pre-programmed with firmware and no "firmware=" string should be used with these cards. To change the firmware on a 5i25 or 6i25 the "mesaflash" utility should be used (available from Mesa). It is perfectly valid and reasonable to load these cards with no config string at all.

num_dlls [optional, default: -1]

The hm2dpll is a phase-locked loop timer module which may be used to trigger certain types of encoder. This parameter can be used to disable the hm2dpll by setting the number to 0. There is only ever one module of this type, with 4 timer channels, so the other valid numbers are -1 (enable all) and 1, both of which end up meaning the same thing.

num encoders [optional, default: -1]

Only enable the first N encoders. If N is -1, all encoders are enabled. If N is 0, no encoders are enabled. If N is greater than the number of encoders available in the firmware, the board will fail to register.

ssi_chan_N [optional, default: ""]

Specifies how the bit stream from a Synchronous Serial Interface device will be interpreted. There should be an entry for each device connected. Only channels with a format specifier will be enabled. (as the software can not guess data rates and bit lengths)

biss_chan_N [optional, default: ""]

As for ssi_chan_N, but for BiSS devices

fanuc_chan_N [optional, default: ""]

Specifies how the bit stream from a Fanuc absolute encoder will be interpreted. There should be an entry for each device connected. Only channels with a format specifier will be enabled. (as the software can not guess data rates and bit lengths)

num_resolvers [optional, default: -1]

Only enable the first N resolvers. If N = -1 then all resolvers are enabled. This module does not work with generic resolvers (unlike the encoder module which works with any encoder). At the time of writing the Hostmot2 Resolver function only works with the Mesa 7i49 card.

num_pwmgens [optional, default: -1]

Only enable the first N pwmgens. If N is -1, all pwmgens are enabled. If N is 0, no pwmgens are enabled. If N is greater than the number of pwmgens available in the firmware, the board will fail to register.

num_3pwmgens [optional, default: -1]

Only enable the first N Three-phase pwmgens. If N is -1, all 3pwmgens are enabled. If N is 0, no pwmgens are enabled. If N is greater than the number of pwmgens available in the firmware, the board will fail to register.

num_stepgens [optional, default: -1]

Only enable the first N stepgens. If N is -1, all stepgens are enabled. If N is 0, no stepgens are enabled. If N is greater than the number of stepgens available in the firmware, the board will fail to register.

stepgen_width [optional, default: 2]

Used to mask extra, unwanted, stepgen pins. Stepper drives typically require only two pins (step and dir) but the Hostmot2 stepgen can drive up to 8 output pins for specialised applications (depending on firmware). This parameter applies to all stepgen instances. Unused, masked pins will be available as GPIO.

sserial_port_N (N = 0 .. 3) [optional, default: 00000000 for all ports]

Up to 32 Smart Serial devices can be connected to a Mesa Anything IO board depending on the firmware used and the number of physical connections on the board. These are arranged in 1-4 ports of 1 to 8 channels.

Some Smart Serial (SSLBP) cards offer more than one load-time configuration, for example all inputs, or all outputs, or offering additional analogue input on some digital pins.

To set the modes for port 0 use, for example **sserial_port_0=0120xxxx**. A '0' in the string sets the corresponding port to mode 0, 1 to mode 1, and so on up to mode 9. An "x" in any position disables that channel and makes the corresponding FPGA pins available as GPIO.

The string can be up to 8 characters long, and if it defines more modes than there are channels on the port then the extras are ignored. Channel numbering is left to right so the example above would set sserial device 0.0 to mode 0, 0.2 to mode2 and disable channels 0.4 onwards.

The sserial driver will auto-detect connected devices, no further configuration should be needed. Unconnected channels will default to GPIO,

but the pin values will vary semi-randomly during boot when card-detection runs, so it is best to actively disable any channel that is to be used for GPIO.

num_bspis [optional, default: -1]

Only enable the first N Buffered SPI drivers. If N is -1 then all the drivers are enabled. Each BSPI driver can address 16 devices.

num_leds [optional, default: -1]

Only enable the first N of the LEDs on the FPGA board. If N is -1, then HAL pins for all the LEDs will be created. If N=0 then no pins will be added.

enable_raw [optional]

If specified, this turns on a raw access mode, whereby a user can peek and poke the firmware from HAL. See Raw Mode below.

dpll

The hm2dpll module has pins like "hm2_<BoardType>.<BoardNum>.dpll" It is likely that the pin-count will decrease in the future and that some pins will become parameters. This module is a phase-locked loop that will synchronise itself with the thread in which the hostmot2 "read" function is installed and will trigger other functions that are allocated to it at a specified time before or after the "read" function runs. This can currently only be applied to the three absolute encoder types and is intended to ensure that the data is ready when needed, and as fresh as possible.

Pins:

(float, in) hm2_<BoardType>.<BoardNum>.dpll.NN.timer-us

This pin sets the triggering offset of the associated timer. There are 4 timers numbered 01 to 04, represented by the NN digits in the pin name. The units are micro-seconds. Negative numbers indicate that the trigger should occur prior to the main hostmot2 write. It is anticipated that this value will be calculated from the known bit-count and data-rate of the functions to be triggered. Alternatively you can just keep making the number more negative until the over-run error bit in the encoder goes false. The default value is set to 100uS, enough time for approximately 50 bits to be transmitted at 500kHz. For very critical systems it may be worth reducing this until errors appear, and for very long bit-length or slow encoders it will need to be increased.

(float, in) hm2_<BoardType>.<BoardNum>.dpll.base-freq-khz

This pin sets the base frequency of the phase-locked loop. by default it will be set to the nominal frequency of the thread in which the PLL is running and will not normally need to be changed.

(float, out) hm2_<BoardType>.<BoardNum>.dpll.phase-error-us

Indicates the phase error of the DPLL. If the number cycles by a large amount it is likely that the PLL has failed to achieve lock and adjustments will need to be made.

(u32, in) hm2_<BoardType>.<BoardNum>.dpll.time-const"

The filter time-constant for the PLL. Default 40960 (0xA000)

(u32, in) hm2_<BoardType>.<BoardNum>.dpll.plimit"

Sets the phase adjustment limit of the PLL. If the value is zero then the PLL will free-run at the base frequency independent of the servo thread rate. This is probably not what you want. Default 4194304 (0x4000000) Units not known...

(u32, out) hm2_<BoardType>.<BoardNum>.dpll.ddsize
 Used internally by the driver, likely to disappear.

(u32, in) hm2_<BoardType>.<BoardNum>.dpll.prescale
 Prescale factor for the rate generator. Default 1.

encoder

Encoders have names like "hm2_<BoardType>.<BoardNum>.encoder.<Instance>". "Instance" is a two-digit number that corresponds to the HostMot2 encoder instance number. There are 'num_encoders' instances, starting with 00.

So, for example, the HAL pin that has the current position of the second encoder of the first 5i20 board is: hm2_5i20.0.encoder.01.position (this assumes that the firmware in that board is configured so that this HAL object is available)

Each encoder uses three or four input IO pins, depending on how the firmware was compiled. Three-pin encoders use A, B, and Index (sometimes also known as Z). Four-pin encoders use A, B, Index, and Index-mask.

The hm2 encoder representation is similar to the one described by the Canonical Device Interface (in the HAL General Reference document), and to the software encoder component. Each encoder instance has the following pins and parameters:

Pins:

(s32 out) count
 Number of encoder counts since the previous reset.

(float out) position
 Encoder position in position units (count / scale).

(float out) velocity
 Estimated encoder velocity in position units per second.

(bit in) reset
 When this pin is TRUE, the count and position pins are set to 0. (The value of the velocity pin is not affected by this.) The driver does not reset this pin to FALSE after resetting the count to 0, that is the user's job.

(bit in/out) index-enable
 When this pin is set to True, the count (and therefore also position) are reset to zero on the next Index (Phase-Z) pulse. At the same time, index-enable is reset to zero to indicate that the pulse has occurred.

(s32 out) rawcounts
 Total number of encoder counts since the start, not adjusted for index or reset.

Parameters:

(float r/w) scale
 Converts from 'count' units to 'position' units.

(bit r/w) index-invert
 If set to True, the rising edge of the Index input pin triggers the Index event (if index-enable is True). If set to False, the falling edge triggers.

(bit r/w) index-mask
 If set to True, the Index input pin only has an effect if the Index-Mask input pin is True (or False, depending on the index-mask-invert pin below).

(bit r/w) index-mask-invert
 If set to True, Index-Mask must be False for Index to have an effect. If set to False, the Index-Mask pin must be True.

(bit r/w) counter-mode
 Set to False (the default) for Quadrature. Set to True for Step/Dir (in which case Step is on the A pin and Dir is on the B pin).

(bit r/w) filter
 If set to True (the default), the quadrature counter needs 15 clocks to register a change on any of the three input lines (any pulse shorter than this is rejected as noise). If set to False, the quadrature counter needs only 3 clocks to register a change. The encoder sample clock runs at 33 MHz on the PCI AnyIO cards and 50 MHz on the 7i43.

(float r/w) vel-timeout
 When the encoder is moving slower than one pulse for each time that the driver reads the count from the FPGA (in the hm2_read() function), the velocity is harder to estimate. The driver can wait several iterations for the next pulse to arrive, all the while reporting the upper bound of the encoder velocity, which can be accurately guessed. This parameter specifies how long to wait for the next pulse, before reporting the encoder stopped. This parameter is in seconds.

Synchronous Serial Interface (SSI)

(Not to be confused with the Smart Serial Interface)

One pin is created for each SSI instance regardless of data format: (bit, in) hm2_XiXX.NN.ssi.MM.data-incomplete This pin will be set "true" if the module was still transferring data when the value was read. When this problem exists there will also be a limited number of error messages printed to the UI. This pin should be used to monitor whether the problem has been addressed by config changes. Solutions to the problem depend on whether the encoder read is being triggered by the hm2dppl phase-locked-loop timer (described above) or by the trigger-encoders function (described below).

The names of the pins created by the SSI module will depend entirely on the format string for each channel specified in the loadrt command line. A typical format string might be

ssi_chan_0=error%1bposition%24g

This would interpret the LSB of the bit-stream as a bit-type pin named "error" and the next 24 bits as a Gray-coded encoder counter. The encoder-related HAL pins would all

begin with "position".

There should be no spaces in the format string, as this is used as a delimiter by the low-level code.

The format consists of a string of alphanumeric characters that will form the HAL pin names, followed by a % symbol, a bit-count and a data type. All bits in the packet must be defined, even if they are not used. There is a limit of 64 bits in total.

The valid format characters and the pins they create are:

p: (Pad). Does not create any pins, used to ignore sections of the bit stream that are not required.

b: (Boolean).

(bit, out) hm2_XiXX.N.ssi.MM.<name>. If any bits in the designated field width are non-zero then the HAL pin will be "true".

(bit, out) hm2_XiXX.N.ssi.MM.<name>-not. An inverted version of the above, the HAL pin will be "true" if all bits in the field are zero.

u: (Unsigned)

(float, out) hm2_XiXX.N.ssi.MM.<name>. The value of the bits interpreted as an unsigned integer then scaled such that the pin value will equal the scalemax parameter value when all bits are high. (for example if the field is 8 bits wide and the scalmax parameter was 20 then a value of 255 would return 20, and 0 would return 0).

s: (Signed)

(float, out) hm2_XiXX.N.ssi.MM.<name>. The value of the bits interpreted as a 2s complement signed number then scaled similarly to the unsigned variant, except symmetrical around zero.

f: (bitField)

(bit, out) hm2_XiXX.N.ssi.MM.<name>-NN. The value of each individual bit in the data field. NN starts at 00 up to the number of bits in the field.

(bit, out) hm2_XiXX.N.ssi.MM.<name>-NN-not. An inverted version of the individual bit values.

e: (Encoder)

(s32, out) hm2_XiXX.N.ssi.MM.<name>.count. The lower 32 bits of the total encoder counts. This value is reset both by the ...reset and the ...index- enable pins.

(s32, out) hm2_XiXX.N.ssi.MM.<name>.rawcounts. The lower 32 bits of the total encoder counts. The pin is not affected by reset and index.

(float, out) hm2_XiXX.N.ssi.MM.<name>.position. The encoder position in machine units. This is calculated from the full 64-bit buffers so will show a true value even after the counts pins have wrapped. It is zeroed by reset and index enable.

(bit, IO) hm2_XiXX.N.ssi.MM.<name>.index-enable. When this pin is set "true" the module will wait until the raw encoder counts next passes through an integer multiple of the number of counts specified by counts-per-rev parameter and then it will zero the counts and position pins, and set the index-enable pin back to "false" as a signal to the system that "index" has been passed. this pin is used for spindle-synchronised motion and index-homing.

(bit, in) (bit, out) hm2_XiXX.N.ssi.MM.<name>.reset. When this pin is set high the counts and position pins are zeroed.

h: (Split encoder, high-order bits)

Some encoders (Including Fanuc) place the encoder part-turn counts and full-turn counts in separate, non-contiguous fields. This tag defines the high-order bits of such an encoder module. There can be only one h and one l tag per channel, the behaviour with multiple such channels will be undefined.

l: (Split encoder, low-order bits)

Low order bits (see "h")

g: (Gray-code). This is a modifier that indicates that the following

format string is gray-code encoded. This is only valid for encoders (e, h l) and unsigned (u) data types.

Parameters:

Two parameters is universally created for all SSI instances

(float r/w) hm2_XiXX.N.ssi.MM.frequency-khz

This parameter sets the SSI clock frequency. The units are kHz, so 500 will give a clock frequency of 500,000 Hz.

(u32 r/w) hm2_XiXX.N.ssi.MM.timer-num

This parameter allocates the SSI module to a specific hm2dpll timer instance. This pin is only of use in firmwares which contain a hm2dpll function and will default to 1 in cases where there is such a function, and 0 if there is not. The pin can be used to disable reads of the encoder, by setting to a nonexistent timer number, or to 0.

Other parameters depend on the data types specified in the config string.

p: (Pad) No Parameters.

b: (Boolean) No Parameters.

u: (Unsigned)

(float, r/w) hm2_XiXX.N.ssi.MM.<name>.scalemax. The scaling factor for the channel.

s: (Signed)

(float, r/w) hm2_XiXX.N.ssi.MM.<name>.scalemax. The scaling factor for the channel.

f: (bitField): No parameters.

e: (Encoder):

(float, r/w) hm2_XiXX.N.ssi.MM.<name>.scale: (float, r/w) The encoder scale in counts per machine unit.

(u32, r/w) hm2_XiXX.N.ssi.MM.<name>.counts-per-rev (u32, r/w) Used to emulate the index behaviour of an incremental+index encoder. This would normally be set to the actual counts per rev of the encoder, but can be any whole number of revs. Integer divisors or multimpilers of the true PPR might be useful for index-homing. Non-integer factors might be appropriate where there is a synchronous drive ratio between the encoder and the spindle or ballscrew.

BiSS

BiSS is a bidirectional variant of SSI. Currently only a single direction is supported by LinuxCNC (encoder to PC).

One pin is created for each BiSS instance regardless of data format:

(bit, in) hm2_XiXX.NN.biss.MM.data-incomplete This pin will be set "true" if the module was still transferring data when the value was read. When this problem exists there will also be a limited number of error messages printed to the UI. This pin should be used to monitor whether the problem has been addressed by config changes. Solutions to the problem depend on whether the encoder read is being triggered by the hm2dpll phase-locked-loop timer (described above) or by the trigger-encoders function (described below)

The names of the pins created by the BiSS module will depend entirely on the format string for each channel specified in the loadrt command line and follow closely the format defined above for SSI. Currently data packets of up to 96 bits are supported by the LinuxCNC driver, although the Mesa Hostmot2 module can handle 512 bit packets. It should be possible to extend the number of packets supported by the driver if there is a requirement to do so.

Fanuc encoder.

The pins and format specifier for this module are identical to the SSI module described above, except that at least one pre-configured format is provided. A modparam of fanuc_chan_N=AA64 (case sensitive) will configure the channel for a Fanuc Aa64 encoder. The pins created are:

hm2_XiXX.N.fanuc.MM.batt	indicates battery state
hm2_XiXX.N.fanuc.MM.batt-not	inverted version of above
hm2_XiXX.N.fanuc.MM.comm	The 0-1023 absolute output for motor communication
hm2_XXiX.N.fanuc.MM.crc	The CRC checksum. Currently HAL has no way to use this
hm2_XiXX.N.fanuc.MM.encoder.count	Encoder counts
hm2_XiXX.N.fanuc.MM.encoder.index-enable	Simulated index. Set by counts-per-rev parameter
hm2_XiXX.N.fanuc.MM.encoder.position	Counts scaled by the ...scale parameter
hm2_XiXX.N.fanuc.MM.encoder.rawcounts	Raw counts, unaffected by reset or index
hm2_XiXX.N.fanuc.MM.encoder.reset	If high/true then counts and position = 0
hm2_XiXX.N.fanuc.MM.valid	Indicates that the absolute position is valid
hm2_XiXX.N.fanuc.MM.valid-not	Inverted version

resolver

Resolvers have names like hm2_<BoardType>.<BoardNum>.resolver.<Instance>. <Instance> is a 2-digit number, which for the 7i49 board will be between 00 and 05. This function only works with the Mesa Resolver interface boards (of which the 7i49 is the only example at the time of writing). This board uses an SPI interface to the FPGA card, and will only work with the correct firmware. The pins allocated will be listed in the dmesg output, but are unlikely to be usefully probed with HAL tools.

Pins:

(float, out) angle

This pin indicates the angular position of the resolver. It is a number between 0 and 1 for each electrical rotation.

(float, out) position

Calculated from the number of complete and partial revolutions since startup, reset, or index-reset multiplied by the scale parameter.

(float, out) velocity

Calculated from the rotational velocity and the velocity-scale parameter. The default scale is electrical rotations per second.

(s32, out) count

This pin outputs a simulated encoder count at 2^{24} counts per rev (16777216 counts).

(s32, out) rawcounts

This is identical to the counts pin, except it is not reset by the 'index' or 'reset' pins. This is the pin which would be linked to the bldc HAL component if the resolver was being used to commutate a motor.

(bit, in) reset

Resets the position and counts pins to zero immediately.

(bit, in/out) index-enable

When this pin is set high the position and counts pins will be reset the next time the resolver passes through the zero position. At the same time the pin is driven low to indicate to connected modules that the index has been seen, and that the counters have been reset.

(bit, out) error

Indicates an error in the particular channel. If this value is "true" then the reported position and velocity are invalid.

Parameters:**(float, read/write) scale**

The position scale, in machine units per resolver electrical revolution.

(float, read/write) velocity-scale

The conversion factor between resolver rotation speed and machine velocity. A value of 1 will typically give motor speed in rps, a value of 0.01666667 will give (approximate) RPM.

(u32, read/write) index-divisor (default 1)

The resolver component emulates an index at a fixed point in the sin/cos cycle. Some resolvers have multiple cycles per rev (often related to the number of pole-pairs on the attached motor). LinuxCNC requires an index once per revolution for proper threading etc. This parameter should be set to the number of cycles per rev of the resolver. CAUTION: Which pseudo-index is used will not necessarily be consistent between LinuxCNC runs. Do not expect to re-start a thread after restarting LinuxCNC. It is not appropriate to use this parameter for index-homing of axis drives.

(float, read/write) excitation-khz

This pin sets the excitation frequency for the resolver. This pin is module-level rather than instance-level as all resolvers share the same excitation frequency.

Valid values are 10 (~10kHz), 5 (~5kHz) and 2.5 (~2.5kHz). The actual frequency depends on the FPGA frequency, and they correspond to CLOCK_LOW/5000, CLOCK_LOW/10000 and CLOCK_LOW/20000 respectively. The parameter will be set to the closest available of the three frequencies.

A value of -1 (the default) indicates that the current setting should be retained.

pwmgen

pwmgens have names like "hm2_<BoardType>.<BoardNum>.pwmgen.<Instance>". "Instance" is a two-digit number that corresponds to the HostMot2 pwmgen instance number. There are 'num_pwmgens' instances, starting with 00.

So, for example, the HAL pin that enables output from the fourth pwmgen of the first 7i43 board is: hm2_7i43.0.pwmgen.03.enable (this assumes that the firmware in that board is configured so that this HAL object is available)

In HM2, each pwmgen uses three output IO pins: Not-Enable, Out0, and Out1.

The function of the Out0 and Out1 IO pins varies with output-type parameter (see below).

The hm2 pwmgen representation is similar to the software pwmgen component. Each pwmgen instance has the following pins and parameters:

Pins:

(bit input) enable

If true, the pwmgen will set its Not-Enable pin false and output its pulses. If 'enable' is false, pwmgen will set its Not-Enable pin true and not output any signals.

(float input) value

The current pwmgen command value, in arbitrary units.

Parameters:

(float rw) scale

Scaling factor to convert 'value' from arbitrary units to duty cycle: dc = value / scale. Duty cycle has an effective range of -1.0 to +1.0 inclusive, anything outside that range gets clipped. The default scale is 1.0.

(s32 rw) output-type

This emulates the output_type load-time argument to the software pwmgen component. This parameter may be changed at runtime, but most of the time you probably want to set it at startup and then leave it alone. Accepted values are 1 (PWM on Out0 and Direction on Out1), 2 (Up on Out0 and Down on Out1), 3 (PDM mode, PDM on Out0 and Dir on Out1), and 4 (Direction on Out0 and PWM on Out1, "for locked antiphase").

In addition to the per-instance HAL Parameters listed above, there are a couple

of HAL Parameters that affect all the pwmgen instances:

(u32 rw) `pwm_frequency`

This specifies the PWM frequency, in Hz, of all the pwmgen instances running in the PWM modes (modes 1 and 2). This is the frequency of the variable-duty-cycle wave. Its effective range is from 1 Hz up to 193 kHz. Note that the max frequency is determined by the ClockHigh frequency of the Anything IO board; the 5i20 and 7i43 both have a 100 MHz clock, resulting in a 193 kHz max PWM frequency. Other boards may have different clocks, resulting in different max PWM frequencies. If the user attempts to set the frequency too high, it will be clipped to the max supported frequency of the board. Frequencies below about 5 Hz are not terribly accurate, but above 5 Hz they're pretty close. The default `pwm_frequency` is 20,000 Hz (20 kHz).

(u32 rw) `pdm_frequency`

This specifies the PDM frequency, in Hz, of all the pwmgen instances running in PDM mode (mode 3). This is the "pulse slot frequency"; the frequency at which the pdm generator in the AnyIO board chooses whether to emit a pulse or a space. Each pulse (and space) in the PDM pulse train has a duration of 1/`pdm_frequency` seconds. For example, setting the `pdm_frequency` to 2e6 (2 MHz) and the duty cycle to 50% results in a 1 MHz square wave, identical to a 1 MHz PWM signal with 50% duty cycle. The effective range of this parameter is from about 1525 Hz up to just under 100 MHz. Note that the max frequency is determined by the ClockHigh frequency of the Anything IO board; the 5i20 and 7i43 both have a 100 MHz clock, resulting in a 100 MHz max PDM frequency. Other boards may have different clocks, resulting in different max PDM frequencies. If the user attempts to set the frequency too high, it will be clipped to the max supported frequency of the board. The default `pdm_frequency` is 20,000 Hz (20 kHz).

3ppwmgen

Three-Phase PWM generators (3pwmgens) are intended for controlling the high-side and low-side gates in a 3-phase motor driver. The function is included to support the Mesa motor controller daughter-cards but can be used to control an IGBT or similar driver directly. 3pwmgens have names like "`hm2_<BoardType>.<BoardNum>.3pwmgen.<Instance>`" where `<Instance>` is a 2-digit number. There will be num_3pwmgens instances, starting at 00. Each instance allocates 7 output and one input pins on the Mesa card connectors. Outputs are: PWM A, PWM B, PWM C, /PWM A, /PWM B, /PWM C, Enable. The first three pins are the high side drivers, the second three are their complementary low-side drivers. The enable bit is intended to control the servo amplifier. The input bit is a fault bit, typically wired to over-current detection. When set the PWM generator is disabled. The three phase duty-cycles are individually controllable from -Scale to +Scale. Note that 0 corresponds to a 50% duty cycle and this is the initialization value.

Pins:

(float input) A-value, B-value, C-value: The PWM command value for each phase, limited to +/- "scale". Defaults to zero which is 50% duty cycle on high-side and low-side-pins (but see the "deadtime" parameter)

(bit input) enable

When high the PWM is enabled as long as the fault bit is not set by the external fault input pin. When low the PWM is disabled, with both high- side and low-

side drivers low. This is not the same as 0 output (50% duty cycle on both sets of pins) or negative full scale (where the low side drivers are "on" 100% of the time)

(bit output) fault

Indicates the status of the fault bit. This output latches high once set by the physical fault pin until the "enable" pin is set to high.

Parameters:

(u32 rw) deadtime

Sets the dead-time between the high-side driver turning off and the low-side driver turning on and vice-versa. Deadtime is subtracted from on time and added to off time symmetrically. For example with 20 kHz PWM (50 uSec period), 50% duty cycle and zero dead time, the PWM and NPWM outputs would be square waves (NPWM being inverted from PWM) with high times of 25 uS. With the same settings but 1 uS of deadtime, the PWM and NPWM outputs would both have high times of 23 uS (25 - (2X 1 uS), 1 uS per edge). The value is specified in nS and defaults to a rather conservative 5000nS. Setting this parameter to too low a value could be both expensive and dangerous as if both gates are open at the same time there is effectively a short circuit across the supply.

(float rw) scale

Sets the half-scale of the specified 3-phase PWM generator. PWM values from -scale to +scale are valid. Default is +/- 1.0

(bit rw) fault-invert

Sets the polarity of the fault input pin. A value of 1 means that a fault is triggered with the pin high, and 0 means that a fault is triggered when the pin is pulled low. Default 0, fault = low so that the PWM works with the fault pin unconnected.

(u32 rw) sample-time

Sets the time during the cycle when an ADC pulse is generated. 0 = start of PWM cycle and 1 = end. Not currently useful to LinuxCNC. Default 0.5.

In addition the per-instance parameters above there is the following parameter that affects all instances

(u32 rw) frequency

Sets the master PWM frequency. Maximum is approx 48kHz, minimum is 1kHz. Defaults to 20kHz.

stepgen

steppergen have names like "hm2_<BoardType>.<BoardNum>.steppergen.<Instance>". "Instance" is a two-digit number that corresponds to the HostMot2 steppergen instance number. There are 'num_steppergen' instances, starting with 00.

So, for example, the HAL pin that has the current position feedback from the first steppergen of the second 5i22 board is: hm2_5i22.1.steppergen.00.position-fb (this assumes that the

firmware in that board is configured so that this HAL object is available)

Each stepgen uses between 2 and 6 IO pins. The signals on these pins depends on the step_type parameter (described below).

The stepgen representation is modeled on the stepgen software component. Each stepgen instance has the following pins and parameters:

Pins:

(float input) position-cmd

Target position of stepper motion, in arbitrary position units. This pin is only used when the stepgen is in position control mode (control-type=0).

(float input) velocity-cmd

Target velocity of stepper motion, in arbitrary position units per second. This pin is only used when the stepgen is in velocity control mode (control-type=1).

(s32 output) counts

Feedback position in counts (number of steps).

(float output) position-fb

Feedback position in arbitrary position units. This is similar to "counts/position_scale", but has finer than step resolution.

(float output) velocity-fb

Feedback velocity in arbitrary position units per second.

(bit input) enable

This pin enables the step generator instance. When True, the stepgen instance works as expected. When False, no steps are generated and velocity-fb goes immediately to 0. If the stepgen is moving when enable goes false it stops immediately, without obeying the maxaccel limit.

(bit input) control-type

Switches between position control mode (0) and velocity control mode (1). Defaults to position control (0).

Parameters:

(float r/w) position-scale

Converts from counts to position units. position = counts / position_scale

(float r/w) maxvel

Maximum speed, in position units per second. If set to 0, the driver will always use the maximum possible velocity based on the current step timings and position-scale. The max velocity will change if the step timings or position-scale changes. Defaults to 0.

(float r/w) maxaccel

Maximum acceleration, in position units per second per second. Defaults to 1.0. If set to 0, the driver will not limit its acceleration at all - this requires that the position-cmd or velocity-cmd pin is driven in a way that does not exceed the machine's capabilities. This is probably what you want if you're going to be using the LinuxCNC trajectory planner to jog or run G-code.

(u32 r/w) steplen

Duration of the step signal, in nanoseconds.

(u32 r/w) stepspace

Minimum interval between step signals, in nanoseconds.

(u32 r/w) dirsetup

Minimum duration of stable Direction signal before a step begins, in nanoseconds.

(u32 r/w) dirhold

Minimum duration of stable Direction signal after a step ends, in nanoseconds.

(u32 r/w) step_type

Output format, like the step_type modparam to the software stegen(9) component. 0 = Step/Dir, 1 = Up/Down, 2 = Quadrature, 3+ = table-lookup mode. In this mode the step_type parameter determines how long the step sequence is. Additionally the stepgen_width parameter in the loadrt config string must be set to suit the number of pins per stepgen required. Any stepgen pins above this number will be available for GPIO. This mask defaults to 2. The maximum length is 16. Note that Table mode is not enabled in all firmwares but if you see GPIO pins between the stepgen instances in the dmesg/log hardware pin list then the option may be available.

In Quadrature mode (step_type=2), the stepgen outputs one complete Gray cycle (00 → 01 → 11 → 10 → 00) for each "step" it takes. In table mode up to 6 IO pins are individually controlled in an arbitrary sequence up to 16 phases long.

(u32 r/w) table-data-N

There are 4 table-data-N parameters, table-data-0 to table-data-3. These each contain 4 bytes corresponding to 4 stages in the step sequence. For example table-data-0 = 0x00000001 would set stepgen pin 0 (always called "Step" in the dmesg output) on the first phase of the step sequence, and table-data-4 = 0x20000000 would set stepgen pin 6 ("Table5Pin" in the dmesg output) on the 16th stage of the step sequence.

Smart Serial Interface

The Smart Serial Interface allows up to 32 different devices such as the Mesa 8i20 2.2kW 3-phase drive or 7i64 48-way IO cards to be connected to a single FPGA card. The driver auto-detects the connected hardware port, channel and device type. Devices can be connected in any order to any active channel of an active port. (see the config modparam definition above).

For full details of the smart-serial devices see **man sserial**.

BSPI

The BSPI (Buffered SPI) driver is unusual in that it does not create any HAL pins. Instead the driver exports a set of functions that can be used by a sub -driver for the attached hardware. Typically these would be written in the "comp" pre-processing language: see http://linuxcnc.org/docs/html/hal_comp.html or man comp for further details. See man mesa_7i65 and the source of mesa_7i65.comp for details of a typical sub-driver. See man hm2_bspi_setup_chan, man hm2_bspi_write_chan, man hm2_tram_add_bspi_frame, man hm2_allocate_bspi_tram, man hm2_bspi_set_read_function and man hm2_bspi_set_write_function for the exported functions.

The names of the available channels are printed to standard output during the driver loading process and take the form hm2_<board name>.<board index>.bspi.<index> For example hm2_5i23.0.bspi.0

UART

The UART driver also does not create any HAL pins, instead it declares two simple read/write functions and a setup function to be utilised by user-written code. Typically this would be written in the "comp" pre-processing language: see http://linuxcnc.org/docs/html/hal_comp.html or man comp for further details. See man mesa_uart and the source of mesa_uart.comp for details of a typical sub-driver. See man hm2_uart_setup_chan, man hm2_uart_send, man hm2_uart_read and man hm2_uart_set-up.

The names of the available uart channels are printed to standard output during the driver loading process and take the form hm2_<board name>.<board index>.uart.<index> For example hm2_5i23.0 uart.0

General Purpose I/O

I/O pins on the board which are not used by a module instance are exported to HAL as "full" GPIO pins. Full GPIO pins can be configured at run-time to be inputs, outputs, or open drains, and have a HAL interface that exposes this flexibility. IO pins that are owned by an active module instance are constrained by the requirements of the owning module, and have a restricted HAL interface.

GPIOs have names like "hm2_<BoardType>.<BoardNum>.gpio.<IONum>". IONum is a three-digit number. The mapping from IONum to connector and pin-on-that-connector is written to the syslog when the driver loads, and it's documented in Mesa's manual for the Anything I/O boards.

So, for example, the HAL pin that has the current inverted input value read from GPIO 012 of the second 7i43 board is: hm2_7i43.1 gpio.012.in-not (this assumes that the firmware in that board is configured so that this HAL object is available)

The HAL parameter that controls whether the last GPIO of the first 5i22 is an input or an output is: hm2_5i22.0 gpio.095.is_output (this assumes that the firmware in that board is configured so that this HAL object is available)

The hm2 GPIO representation is modeled after the Digital Inputs and Digital Outputs described in the Canonical Device Interface (part of the HAL General Reference document). Each GPIO can have the following HAL Pins:

(bit out) in & in_not: State (normal and inverted) of the hardware input pin. Both full GPIO pins and IO pins used as inputs by active module instances have these pins.

(bit in) out

Value to be written (possibly inverted) to the hardware output pin. Only full GPIO pins have this pin.

Each GPIO can have the following Parameters:

(bit r/w) is_output

If set to 0, the GPIO is an input. The IO pin is put in a high-impedance state (weakly pulled high), to be driven by other devices. The logic value on the IO pin is available in the "in" and "in_not" HAL pins. Writes to the "out" HAL pin have no effect. If this parameter is set to 1, the GPIO is an output; its behavior then depends on the "is_opendrain" parameter. Only full GPIO pins have this parameter.

(bit r/w) is_opendrain

This parameter only has an effect if the "is_output" parameter is true. If this parameter is false, the GPIO behaves as a normal output pin: the IO pin on the connector is driven to the value specified by the "out" HAL pin (possibly inverted), and the value of the "in" and "in_not" HAL pins is undefined. If this parameter is true, the GPIO behaves as an open-drain pin. Writing 0 to the "out" HAL pin drives the IO pin low, writing 1 to the "out" HAL pin puts the IO pin in a high-impedance state. In this high-impedance state the IO pin floats (weakly pulled high), and other devices can drive the value; the resulting value on the IO pin is available on the "in" and "in_not" pins. Only full GPIO pins and IO pins used as outputs by active module instances have this parameter.

(bit r/w) invert_output

This parameter only has an effect if the "is_output" parameter is true. If this parameter is true, the output value of the GPIO will be the inverse of the value on the "out" HAL pin. Only full GPIO pins and IO pins used as outputs by active module instances have this parameter.

led

Creates HAL pins for the LEDs on the FPGA board.

Pins:

(bit in) CR<NN>

The pins are numbered from CR01 upwards with the name corresponding to the PCB silkscreen. Setting the bit to "true" or 1 lights the led.

Watchdog

The HostMot2 firmware may include a watchdog Module; if it does, the hostmot2 driver will use it. The HAL representation of the watchdog is named "hm2_<Board-Type>.<BoardNum>.watchdog".

The watchdog starts out asleep and inactive. Once you access the board the first time by running any the hm2 HAL functions read(), write(), or pet_watchdog() (see below), the watchdog wakes up. From them on it must be patted periodically or it will bite. Pet the watchdog by running the pet_watchdog() HAL function.

When the watchdog bites, all the board's I/O pins are disconnected from their Module

instances and become high-impedance inputs (pulled high), and all communication with the board stops. The state of the HostMot2 firmware modules is not disturbed (except for the configuration of the IO Pins). Encoder instances keep counting quadrature pulses, and pwm- and step-generators keep generating signals (which are *not* relayed to the motors, because the IO Pins have become inputs).

Resetting the watchdog (by clearing the has_bit pin, see below) resumes communication and resets the I/O pins to the configuration chosen at load-time.

If the firmware includes a watchdog, the following HAL objects will be exported:

Pins:

(bit in/out) has_bit

True if the watchdog has bit, False if the watchdog has not bit. If the watchdog has bit and the has_bit bit is True, the user can reset it to False to resume operation.

Parameters:

(u32 read/write) timeout_ns

Watchdog timeout, in nanoseconds. This is initialized to 5,000,000 (5 milliseconds) at module load time. If more than this amount of time passes between calls to the pet_watchdog() function, the watchdog will bite.

Functions:

pet_watchdog(): Calling this function resets the watchdog timer (postponing the watchdog biting until timeout_ns nanoseconds later).

Raw Mode

If the "enable_raw" config keyword is specified, some extra debugging pins are made available in HAL. The raw mode HAL pin names begin with "hm2_<Board-Type>.<BoardNum>.raw".

With Raw mode enabled, a user may peek and poke the firmware from HAL, and may dump the internal state of the hostmot2 driver to the syslog.

Pins:

(u32 in) read_address

The bottom 16 bits of this is used as the address to read from.

(u32 out) read_data

Each time the hm2_read() function is called, this pin is updated with the value at .read_address.

(u32 in) write_address

The bottom 16 bits of this is used as the address to write to.

(u32 in) write_data

This is the value to write to .write_address.

(bit in) write_strobe

Each time the hm2_write() function is called, this pin is examined. If it is True, then value in .write_data is written to the address in .write_address, and .write_strobe is set back to False.

(bit in/out) dump_state

This pin is normally False. If it gets set to True the hostmot2 driver will write its representation of the board's internal state to the syslog, and set the pin back to False.

Setting up Smart Serial devices

See man setserial for the current way to set smart-serial eeprom parameters.

FUNCTIONS**hm2_<BoardType>.<BoardNum>.read**

This reads the encoder counters, stepgen feedbacks, and GPIO input pins from the FPGA.

hm2_<BoardType>.<BoardNum>.write

This updates the PWM duty cycles, stepgen rates, and GPIO outputs on the FPGA. Any changes to configuration pins such as stepgen timing, GPIO inversions, etc, are also effected by this function.

hm2_<BoardType>.<BoardNum>.pet-watchdog

Pet the watchdog to keep it from biting us for a while.

hm2_<BoardType>.<BoardNum>.read_gpio

Read the GPIO input pins. Note that the effect of this function is a subset of the effect of the .read() function described above. Normally only .read() is used. The only reason to call this function is if you want to do GPIO things in a faster-than-servo thread. (This function is not available on the 7i43 due to limitations of the EPP bus.)

hm2_<BoardType>.<BoardNum>.write_gpio

Write the GPIO control registers and output pins. Note that the effect of this function is a subset of the effect of the .write() function described above. Normally only .write() is used. The only reason to call this function is if you want to do GPIO things in a faster-than-servo thread. (This function is not available on the 7i43 due to limitations of the EPP bus.)

hm2_<BoardType>.<BoardNum>.trigger-encoders

This function will only appear if the firmware contains a BiSS, Fanuc or SSI encoder module and if the firmware does not contain a hm2dpll module (qv) or if the modparam contains num_dplls=0. This function should be inserted first in the thread so that the encoder data is ready when the main **hm2_XiXX.NN.read** function runs. An error message will be printed if the encoder read is not finished in time. It may be possible to avoid this by increasing the data rate. If the problem persists and if "stale" data is acceptable then the function may be placed later in the thread, allowing a full servo cycle for the data to be transferred from the devices. If available it is better to use the synchronous hm2dpll triggering function.

SEE ALSO

hm2_7i43(9)

hm2_pci(9)

Mesa's documentation for the Anything I/O boards, at <<http://www.mesonet.com>>

HOSTMOT2(9)

HAL Component

HOSTMOT2(9)

LICENSE

GPL

NAME

hypot – Three-input hypotenuse (Euclidean distance) calculator

SYNOPSIS

loadrt hypot [count=N|names=name1[,name2...]]

FUNCTIONS

hypot.N (requires a floating-point thread)

PINS

hypot.N.in0 float in

hypot.N.in1 float in

hypot.N.in2 float in

hypot.N.out float out

out = sqrt(in0^2 + in1^2 + in2^2)

LICENSE

GPL

NAME

ilowpass – Low-pass filter with integer inputs and outputs

SYNOPSIS

loadrt ilowpass [count=N|names=name1[,name2...]]

DESCRIPTION

While it may find other applications, this component was written to create smoother motion while jogging with an MPG.

In a machine with high acceleration, a short jog can behave almost like a step function. By putting the **ilowpass** component between the MPG encoder **counts** output and the axis jog-counts input, this can be smoothed.

Choose **scale** conservatively so that during a single session there will never be more than about $2e9/\text{scale}$ pulses seen on the MPG. Choose **gain** according to the smoothing level desired. Divide the axis.*N.jog-scale* values by **scale**.

FUNCTIONS

ilowpass.N (requires a floating-point thread)

Update the output based on the input and parameters

PINS

ilowpass.N.in s32 in

ilowpass.N.out s32 out

out tracks **in*scale** through a low-pass filter of **gain** per period.

PARAMETERS

ilowpass.N.scale float rw (default: *1024*)

A scale factor applied to the output value of the low-pass filter.

ilowpass.N.gain float rw (default: *.5*)

Together with the period, sets the rate at which the output changes. Useful range is between 0 and 1, with higher values causing the input value to be tracked more quickly. For instance, a setting of 0.9 causes the output value to go 90% of the way towards the input value in each period

AUTHOR

Jeff Epler <jepler@unpythonic.net>

LICENSE

GPL

NAME

integ – Integrator with gain pin and windup limits

SYNOPSIS

loadrt integ [count=N|names=name1[,name2...]]

FUNCTIONS

integ.N (requires a floating-point thread)

PINS

integ.N.in float in

integ.N.gain float in (default: *1.0*)

integ.N.out float out

The discrete integral of 'gain * in' since 'reset' was deasserted

integ.N.reset bit in

When asserted, set out to 0

integ.N.max float in (default: *1e20*)

integ.N.min float in (default: *-1e20*)

LICENSE

GPL

NAME

invert – Compute the inverse of the input signal

SYNOPSIS

The output will be the mathematical inverse of the input, ie **out** = $1/\text{in}$. The parameter **deadband** can be used to control how close to 0 the denominator can be before the output is clamped to 0. **deadband** must be at least 1e-8, and must be positive.

FUNCTIONS

invert.N (requires a floating-point thread)

PINS

invert.N.in float in
Analog input value

invert.N.out float out
Analog output value

PARAMETERS

invert.N.deadband float rw
The **out** will be zero if **in** is between **-deadband** and **+deadband**

LICENSE

GPL

NAME

joyhandle – sets nonlinear joypad movements, deadbands and scales

SYNOPSIS

loadrt joyhandle [count=N|names=name1[,name2...]]

DESCRIPTION

The component **joyhandle** uses the following formula for a non linear joypad movements:

$$y = (\text{scale} * (\text{a} * x^{\text{power}} + \text{b} * x)) + \text{offset}$$

The parameters a and b are adjusted in such a way, that the function starts at (deadband,offset) and ends at (1,scale+offset).

Negative values will be treated point symmetrically to origin. Values -deadband < x < +deadband will be set to zero.

Values x > 1 and x < -1 will be skipped to $\pm(\text{scale}+\text{offset})$. Invert transforms the function to a progressive movement.

With power one can adjust the nonlinearity (default = 2). Default for deadband is 0.

Valid values are: power ≥ 1.0 (reasonable values are 1.x .. 4-5, take higher power-values for higher deadbands (>0.5), if you want to start with a nearly horizontal slope), $0 \leq \text{deadband} < 0.99$ (reasonable 0.1).

An additional offset component can be set in special cases (default = 0).

All values can be adjusted for each instance separately.

FUNCTIONS

joyhandle.N (requires a floating-point thread)

PINS

joyhandle.N.in float in
joyhandle.N.out float out

PARAMETERS

joyhandle.N.power float rw (default: 2.0)
joyhandle.N.deadband float rw (default: 0.)
joyhandle.N.scale float rw (default: 1.)
joyhandle.N.offset float rw (default: 0.)
joyhandle.N.inverse bit rw (default: 0)

LICENSE

GPL

NAME

kins – kinematics definitions for LinuxCNC

SYNOPSIS

```
loadrt trivkins
loadrt rotatekins
loadrt tripodkins
loadrt genhexkins
loadrt maxkins
loadrt genserkins
loadrt pumakins
loadrt scarakins
```

DESCRIPTION

Rather than exporting HAL pins and functions, these components provide the forward and inverse kinematics definitions for LinuxCNC.

trivkins – Trivial Kinematics

There is a 1:1 correspondence between joints and axes. Most standard milling machines and lathes use the trivial kinematics module.

rotatekins – Rotated Kinematics

The X and Y axes are rotated 45 degrees compared to the joints 0 and 1.

tripodkins – Tripod Kinematics

The joints represent the distance of the controlled point from three predefined locations (the motors), giving three degrees of freedom in position (XYZ)

```
tripodkins.Bx
tripodkins.Cx
tripodkins.Cy
```

The location of the three motors is (0,0), (Bx,0), and (Cx,Cy)

genhexkins – Hexapod Kinematics

Gives six degrees of freedom in position and orientation (XYZABC). The location of the motors is defined at compile time.

maxkins – 5-axis kinematics example

Kinematics for Chris Radek's tabletop 5 axis mill named 'max' with tilting head (B axis) and horizontal rotary mounted to the table (C axis). Provides UVW motion in the rotated coordinate system. The source file, maxkins.c, may be a useful starting point for other 5-axis systems.

genserkins – generalized serial kinematics

Kinematics that can model a general serial-link manipulator with up to 6 angular joints.

The kinematics use Denavit-Hartenberg definition for the joint and links. The DH definitions are the ones used by John J Craig in "Introduction to Robotics: Mechanics and Control" The parameters for the manipulator are defined by hal pins.

```
genserkins.A-N
genserkins.ALPHA-N
genserkins.D-N
```

Parameters describing the Nth joint's geometry.

pumakins – kinematics for puma typed robots

Kinematics for a puma-style robot with 6 joints

pumakins.A2
pumakins.A3
pumakins.D3
pumakins.D4

Describe the geometry of the robot

scarakins – kinematics for SCARA-type robots

scarakins.D1

Vertical distance from the ground plane to the center of the inner arm.

scarakins.D2

Horizontal distance between joint[0] axis and joint[1] axis, ie. the length of the inner arm.

scarakins.D3

Vertical distance from the center of the inner arm to the center of the outer arm. May be positive or negative depending on the structure of the robot.

scarakins.D4

Horizontal distance between joint[1] axis and joint[2] axis, ie. the length of the outer arm.

scarakins.D5

Vertical distance from the end effector to the tooltip. Positive means the tooltip is lower than the end effector, and is the normal case.

scarakins.D6

Horizontal distance from the centerline of the end effector (and the joints 2 and 3 axis) and the tooltip. Zero means the tooltip is on the centerline. Non-zero values should be positive, if negative they introduce a 180 degree offset on the value of joint[3].

SEE ALSO

Kinematics section in the LinuxCNC documentation

NAME

knob2float – Convert counts (probably from an encoder) to a float value

SYNOPSIS

loadrt knob2float [count=N|names=name1[,name2...]]

FUNCTIONS

knob2float.N (requires a floating-point thread)

PINS

knob2float.N.counts s32 in
Counts

knob2float.N.enable bit in
When TRUE, output is controlled by count, when FALSE, output is fixed

knob2float.N.scale float in
Amount of output change per count

knob2float.N.out float out
Output value

PARAMETERS

knob2float.N.max-out float rw (default: *1.0*)
Maximum output value, further increases in count will be ignored

knob2float.N.min-out float rw (default: *0.0*)
Minimum output value, further decreases in count will be ignored

LICENSE

GPL

NAME

latencybins – comp utility for scripts/latencyhistogram

SYNOPSIS

Usage:

Read availablebins pin for the number of bins available.

Set the maxbinnumber pin for the number of +/- bins.

Ensure maxbinnumber <= availablebins

For maxbinnumber = N, the bins are numbered:

-N ... 0 ... + N bins

(the -0 bin is not populated)

(total effective bins = 2*maxbinnumber +1)

Set nsbinsize pin for the binsize (ns)

Iterate:

Set index pin to a bin number: 0 <= index <= maxbinnumber.

Read check pin and verify that check pin == index pin.

Read pbvalue,nbvalue,pextra,nextra pins.

(pbvalue is count for bin = +index)

(nbvalue is count for bin = -index)

(pextra is count for all bins > maxbinnumber)

(nextra is count for all bins < maxbinnumber)

If index is out of range (index < 0 or index > maxbinnumber)
then pbvalue = nbvalue = -1.

The reset pin may be used to restart.

The latency pin outputs the instantaneous latency.

Maintainers note: hardcoded for MAXBINNUMBER==1000

FUNCTIONS

latencybins.N

PINS

latencybins.N.maxbinnumber s32 in (default: 1000)

latencybins.N.index s32 in

latencybins.N.reset bit in

latencybins.N.nsbinsize s32 in

latencybins.N.check s32 out

latencybins.N.latency s32 out

latencybins.N.pbvalue s32 out

latencybins.N.nbvalue s32 out

latencybins.N.pextra s32 out

latencybins.N.nextra s32 out

latencybins.N.availablebins s32 out (default: 1000)

LICENSE

GPL

NAME

lcd – Stream HAL data to an LCD screen

SYNOPSIS

```
loadrt lcd fmt_strings=""Plain Text %4.4f\nAnd So on|Second Page, Next Inst""
```

FUNCTIONS

lcd (requires a floating-point thread). All LCD instances are updated by the same function.

PINS

lcd.NN.out (u32) out

The output byte stream is sent via this pin. One character is sent every thread invocation. There is no handshaking provided.

lcd.NN.page.PP.arg.NN (float/s32/u32/bit) in

The input pins have types matched to the format string specifiers.

lcd.NN.page_num (u32) in

Selects the page number. Multiple layouts may be defined, and this pin switches between them.

lcd.NN.contrast (float) in

Attempts to set the contrast of the LCD screen using the byte sequence ESC C and then a value from 0x20 to 0xBF. (matching the Mesa 7i73). The value should be between 0 and 1.

PARAMETERS

lcd.NN.decimal-separator (u32) rw

Sets the decimal separator used for floating point numbers. The default value is 46 (0x2E) which corresponds to ". ". If a comma is required then set this parameter to 44 (0x2C).

DESCRIPTION

lcd takes format strings much like those used in C and many other languages in the printf and scanf functions and their variants.

The component was written specifically to support the Mesa 7i73 pendant controller, however it may be of use streaming data to other character devices and, as the output format mimics the ADM3 terminal format, it could be used to stream data to a serial device. Perhaps even a genuine ADM3. The strings contain a mixture of text values which are displayed directly, "escaped" formatting codes and numerical format descriptors. For a detailed description of formatting codes see: <http://en.wikipedia.org/wiki/Printf>

The component can be configured to display an unlimited number of differently-formatted pages, which may be selected with a HAL pin.

Escaped codes

\n Inserts a clear-to-end, carriage return and line feed character. This will still linefeed and clear even if an automatic wrap has occurred (lcd has no knowledge of the width of the lcd display.) To print in the rightmost column it is necessary to allow the format to wrap and omit the \n code.

\t Inserts a tab (actually 4 spaces in the current version rather than a true tab.)

\NN inserts the character defined by the hexadecimal code NN.

\\\ Insert a literal \.

Numerical formats

lcd differs slightly from the standard printf conventions. One significant difference is that width

limits are strictly enforced to prevent the LCD display wrapping and spoiling the layout. The field width includes the sign character so that negative numbers will often have a smaller valid range than positive. Numbers that do not fit in the specified width are displayed as a line of asterisks (*****).

Each format begins with a "%" symbol. (For a literal % use "%%"). Immediately after the % the following modifiers may be used:

" " (space) Pad the number to the specified width with spaces. This is the default and is not strictly necessary.

"0" Pad the number to the specified width with the numeral 0.

"+" Force display of a + symbol before positive numbers. This (like the - sign) will appear immediately to the left of the digits for a space-padded number and in the extreme left position for a 0-padded number.

"1234567890" A numerical entry (other than the leading 0 above) defines the total number of characters to display including the decimal separator and the sign. Whilst this number can be as many digits as required the maximum field width is 20 characters. The inherent precision of the "double" data type means that more than 14 digits will tend to show errors in the least significant digits. The integer data types will never fill more than 10 decimal digits.

Following the width specifier should be the decimal specifier. This can only be a full-stop character (.) as the comma (,) is used as the instance separator. Currently lcd does not access the locale information to determine the correct separator and the **decimal-separator** parameter should be used.

Following the decimal separator should be a number that determines how many places of decimals to display. This entry is ignored in the case of integer formats.

All the above modifiers are optional, but to specify a decimal precision the decimal point must precede the precision. For example %.3f.

The default decimal precision is 4.

The numerical formats supported are:

%f %F (for example, %+09.3f) These create a floating-point type HAL pin. The example would be displayed in a 9-character field, with 3 places of decimals, . as a decimal separator, padded to the left with 0s and with a sign displayed for both positive and negative. Conversely a plain %f would be 6 digits of decimal, variable format width, with a sign only shown for negative numbers. both %f and %F create exactly the same format.

%i %d (For example %+ 4d) Creates a signed (s32) HAL pin. The example would display the value at a fixed 4 characters, space padded, width including the + giving a range of +999 to -999. %i and %d create identical output.

%u (for example %08u) Creates an unsigned (u32) HAL pin. The example would be a fixed 8 characters wide, padded with zeros.

%x, %X Creates an unsigned (u32) HAL pin and displays the value in Hexadecimal. Both %x and %X display capital letters for digits ABCDEF. A width may be specified, though the u32 HAL type is only 8 hex digits wide.

%o Creates an unsigned (u32) pin and displays the value in Octal.

%c Creates a u32 HAL pin and displays the character corresponding to the value of the pin. Values less than 32 (space) are suppressed. A width specifier may be used, for example %20c might be used to create a complete line of one character.

%b This specifier has no equivalent in printf. It creates a bit (boolean) type HAL pin. The b should be followed by two characters and the display will show the first of these when the pin is true, and the second when false. Note that the characters follow, not precede the "b", unlike the case with other formats. The characters may be "escaped" Hex values. For example "%b\FF" will display a solid black block if true, and a space if false and "%b\7F\7E" would display right-arrow for false and left-arrow for true. An unexpected value of 'E' indicates a formatting error.

Pages The page separator is the "|" (pipe) character. (if the actual character is needed then \7C may be used). A "Page" in this context refers to a separate format which may be displayed on the same display.

Instances The instance separator is the comma. This creates a completely separate lcd instance, for example to drive a second lcd display on the second 7i73. The use of comma to separate instances is built in to the modparam reading code so not even escaped commas "\", can be used. A comma may be displayed by using the \2C sequence.

AUTHOR

Andy Pugh

LICENSE

GPL

NAME

limit1 – Limit the output signal to fall between min and max

SYNOPSIS

loadrt limit1 [count=N|names=name1[,name2...]]

FUNCTIONS

limit1.N (requires a floating-point thread)

PINS

limit1.N.in float in
limit1.N.out float out

PARAMETERS

limit1.N.min float rw (default: *-1e20*)
limit1.N.max float rw (default: *1e20*)

LICENSE

GPL

NAME

limit2 – Limit the output signal to fall between min and max and limit its slew rate to less than maxv per second. When the signal is a position, this means that position and velocity are limited.

SYNOPSIS

```
loadrt limit2 [count=N|names=name1[,name2...]]
```

FUNCTIONS

limit2.N (requires a floating-point thread)

PINS

limit2.N.in float in

limit2.N.out float out

limit2.N.load bit in

When TRUE, immediately set **out** to **in**, ignoring maxv

PARAMETERS

limit2.N.min float rw (default: -1e20)

limit2.N.max float rw (default: 1e20)

limit2.N.maxv float rw (default: 1e20)

LICENSE

GPL

NAME

limit3 – Limit the output signal to fall between min and max, limit its slew rate to less than maxv per second, and limit its second derivative to less than maxa per second squared. When the signal is a position, this means that the position, velocity, and acceleration are limited.

SYNOPSIS

```
loadrt limit3 [count=N|names=name1[,name2...]]
```

FUNCTIONS

limit3.N (requires a floating-point thread)

PINS

limit3.N.in float in

limit3.N.out float out

limit3.N.load bit in

When TRUE, immediately set **out** to **in**, ignoring maxv and maxa

limit3.N.min float in (default: -1e20)

limit3.N.max float in (default: 1e20)

limit3.N.maxv float in (default: 1e20)

limit3.N.maxa float in (default: 1e20)

LICENSE

GPL

NAME

lincurve – one-dimensional lookup table

SYNOPSIS

loadrt lincurve [count=N|names=name1[,name2...]] [personality=P,P,...]

DESCRIPTION

This component can be used to map any floating-point input to a floating-point output. Typical uses would include linearisation of thermocouples, defining PID gains that vary with external factors or to substitute for any mathematical function where absolute accuracy is not required.

The component can be thought of as a 2-dimensional graph of points in (x,y) space joined by straight lines. The input value is located on the x axis, followed up until it touches the line, and the output of the component is set to the corresponding y-value.

The (x,y) points are defined by the x-val-NN and y-val-NN parameters which need to be set in the HAL file using "setp" commands.

The maximum number if (x,y) points supported is 16.

For input values less than the x-val-00 breakpoint the y-val-00 is returned. For x greater than the largest x-val-NN the yval corresponding to x-max is returned (ie, no extrapolation is performed.)

Sample usage: loadrt lincurve count=3 personality=4,4,4 for a set of three 4-element graphs.

FUNCTIONS

lincurve.N (requires a floating-point thread)

PINS

lincurve.N.in float in
The input value

lincurve.N.out float out
The output value

lincurve.N.out-io float io
The output value, compatible with PID gains

PARAMETERS

lincurve.N.x-val-MM float rw (MM=00..personality)
axis breakpoints

lincurve.N.y-val-MM float rw (MM=00..personality)
output values to be interpolated

AUTHOR

Andy Pugh

LICENSE

GPL

NAME

logic – LinuxCNC HAL component providing configurable logic functions

SYNOPSIS

loadrt logic [count=N|names=name1[,name2...]] [personality=P,P,...]

DESCRIPTION

General ‘logic function’ component. Can perform ‘and’, ‘or’ and ‘xor’ of up to 16 inputs.

Determine the proper value for ‘personality’ by adding the inputs and outputs then convert to hex:

- The number of input pins, usually from 2 to 16
- 256 (0x100) if the ‘and’ output is desired
- 512 (0x200) if the ‘or’ output is desired
- 1024 (0x400) if the ‘xor’ (exclusive or) output is desired

Outputs can be combined, for example $2 + 256 + 1024 = 1282$ converted to hex would be 0x502 and would have two inputs and have both ‘xor’ and ‘and’ outputs.

FUNCTIONS

logic.N

PINS

logic.N.in-MM bit in (MM=00..personality & 0xff)

logic.N.and bit out [if personality & 0x100]

logic.N.or bit out [if personality & 0x200]

logic.N.xor bit out [if personality & 0x400]

LICENSE

GPL

NAME

lowpass – Low-pass filter

SYNOPSIS

loadrt lowpass [count=N|names=name1[,name2...]]

FUNCTIONS

lowpass.N (requires a floating-point thread)

PINS

lowpass.N.in float in

lowpass.N.out float out

out += (in - out) * gain

lowpass.N.load bit in

When TRUE, copy **in** to **out** instead of applying the filter equation.

PARAMETERS

lowpass.N.gain float rw

NOTES

The effect of a specific **gain** value is dependent on the period of the function that **lowpass.N** is added to

LICENSE

GPL

NAME

lut5 – Arbitrary 5-input logic function based on a look-up table

SYNOPSIS

loadrt lut5 [count=N|names=name1[,name2...]]

DESCRIPTION

lut5 constructs a logic function with up to 5 inputs using a look-up table. The value for **function** can be determined by writing the truth table, and computing the sum of **all** the **weights** for which the output value would be TRUE. The weights are hexadecimal not decimal so hexadecimal math must be used to sum the weights. A wiki page has a calculator to assist in computing the proper value for function.

<http://wiki.linuxcnc.org/cgi-bin/wiki.pl?Lut5>

Note that LUT5 will generate any of the 4,294,967,296 logical functions of 5 inputs so **AND**, **OR**, **NAND**, **NOR**, **XOR** and every other combinatorial function is possible.

Example Functions

A 5-input *and* function is TRUE only when all the inputs are true, so the correct value for **function** is **0x80000000**.

A 2-input *or* function would be the sum of **0x2 + 0x4 + 0x8**, so the correct value for **function** is **0xe**.

A 5-input *or* function is TRUE whenever any of the inputs are true, so the correct value for **function** is **0xffffffffe**. Because every weight except **0x1** is true the function is the sum of every line except the first one.

A 2-input *xor* function is TRUE whenever exactly one of the inputs is true, so the correct value for **function** is **0x6**. Only **in-0** and **in-1** should be connected to signals, because if any other bit is **TRUE** then the output will be **FALSE**.

Weights for each line of truth table					
Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Weight
0	0	0	0	0	0x1
0	0	0	0	1	0x2
0	0	0	1	0	0x4
0	0	0	1	1	0x8
0	0	1	0	0	0x10
0	0	1	0	1	0x20
0	0	1	1	0	0x40
0	0	1	1	1	0x80
0	1	0	0	0	0x100
0	1	0	0	1	0x200
0	1	0	1	0	0x400
0	1	0	1	1	0x800
0	1	1	0	0	0x1000
0	1	1	0	1	0x2000
0	1	1	1	0	0x4000
0	1	1	1	1	0x8000
1	0	0	0	0	0x10000
1	0	0	0	1	0x20000
1	0	0	1	0	0x40000
1	0	0	1	1	0x80000
1	0	1	0	0	0x100000
1	0	1	0	1	0x200000
1	0	1	1	0	0x400000
1	0	1	1	1	0x800000
1	1	0	0	0	0x1000000
1	1	0	0	1	0x2000000
1	1	0	1	0	0x4000000
1	1	0	1	1	0x8000000
1	1	1	0	0	0x10000000
1	1	1	0	1	0x20000000
1	1	1	1	0	0x40000000
1	1	1	1	1	0x80000000

FUNCTIONS**lut5.N****PINS**

lut5.N.in-0 bit in
lut5.N.in-1 bit in
lut5.N.in-2 bit in
lut5.N.in-3 bit in
lut5.N.in-4 bit in
lut5.N.out bit out

PARAMETERS**lut5.N.function** u32 rw**LICENSE**

GPL

NAME

maj3 – Compute the majority of 3 inputs

SYNOPSIS

loadrt maj3 [count=N|names=name1[,name2...]]

FUNCTIONS

maj3.N

PINS

maj3.N.in1 bit in

maj3.N.in2 bit in

maj3.N.in3 bit in

maj3.N.out bit out

PARAMETERS

maj3.N.invert bit rw

LICENSE

GPL

NAME

match8 – 8-bit binary match detector

SYNOPSIS

loadrt match8 [count=N|names=name1[,name2...]]

FUNCTIONS

match8.N

PINS

match8.N.in bit in (default: *TRUE*)

cascade input - if false, output is false regardless of other inputs

match8.N.a0 bit in

match8.N.a1 bit in

match8.N.a2 bit in

match8.N.a3 bit in

match8.N.a4 bit in

match8.N.a5 bit in

match8.N.a6 bit in

match8.N.a7 bit in

match8.N.b0 bit in

match8.N.b1 bit in

match8.N.b2 bit in

match8.N.b3 bit in

match8.N.b4 bit in

match8.N.b5 bit in

match8.N.b6 bit in

match8.N.b7 bit in

match8.N.out bit out

true only if in is true and a[m] matches b[m] for m = 0 thru 7

LICENSE

GPL

NAME

`matrix_kb` – Convert integers to HAL pins. Optionally scan a matrix of IO ports to create those integers.

SYNOPSIS

loadrt matrix_kb config=RxCs,RxCs... names=name1,name2...

Creates a component configured for R rows and N columns of matrix keyboard.

If the **s** option is specified then a set of output rows will be cyclically toggled, and a set of input columns will be scanned.

The **names** parameter is optional, but if used then the HAL pins and functions will use the specified names rather than the default ones. This can be useful for readability and 2-pass HAL parsing.

There must be no spaces in the parameter lists.

DESCRIPTION

This component was written to convert matrix keyboard scancodes into HAL pins. However, it might also find uses in converting integers from 0 to N into N HAL pins.

The component can work in two ways, and the HAL pins created vary according to mode.

In the default mode the component expects to be given a scan code from a separate driver but could be any integer from any source. Most typically this will be the keypad scancode from a Mesa 7i73. The default codes for keyup and keydown are based on the Mesa 7i73 specification with 0x40 indicating a keydown and 0x80 a keyup event.

If using the 7i73 it is important to match the keypad size jumpers with the HAL component. Valid configs for the 7i73 are 4x8 and 8x8. Note that the component will only work properly with the version 12 (0xC) 7i73 firmware. The firmware version is visible on the component parameters in HAL.

In the optional scan-generation mode the **matrix_kb.N.keycode** pin changes to an output pin and a set of output row pins and input column pins are created. These need to be connected to physical inputs and outputs to scan the matrix and return values to HAL. Note the **negative-logic** parameter described below, this will need to be set on the most common forms of inputs which float high when unconnected.

In both modes a set of HAL output pins are created corresponding to each node of the matrix.

FUNCTIONS

matrix_kb.N

Perform all requested functions. Should be run in a slow thread for effective debouncing.

PINS

matrix_kb.N.col-CC-in bit in

The input pin corresponding to column C.

matrix_kb.N.key,rC bit out

The pin corresponding to the key at row R column C of the matrix.

matrix_kb.N.keycode unsigned in or out depending on mode.

This pin should be connected to the scancode generator if hardware such as a 7i73 is being used.

In this mode it is an input pin. In the internally-generated scanning mode this pin is an output, but will not normally be connected. **matrix_kb.N.row-RR-out** bit out The row scan drive pins. Should be connected to external hardware pins connected to the keypad.

PARAMETERS

matrix_kb.N.key_rollover unsigned r/w (default 2)

With most matrix keyboards the scancodes are only unambiguous with 1 or 2 keys pressed. With more keys pressed phantom keystrokes can appear. Some keyboards are optimised to reduce this problem, and some have internal diodes so that any number of keys may be pressed simultaneously. Increase the value of this parameter if such a keyboard is connected, or if phantom keystrokes are more acceptable than only two keys being active at one time.

matrix_kb.N.negative-logic bit r/w (default 1) only in scan mode

When no keys are pressed a typical digital input will float high. The input will then be pulled low by the keypad when the corresponding poll line is low. Set this parameter to 0 if the IO in use requires one row at a time to be high, and a high input corresponds to a button press.

NAME

mesa_7i65 – Support for the Mesa 7i65 Octuple Servo Card

SYNOPSIS

loadrt mesa_7i65

DESCRIPTION

The component takes parameters in the form of a comma-separated list of bspi (buffered SPI) instance names, for example:

loadrt mesa_7i65 bspi_chans=hm2_5i23.0.bspi.0, hm2_5i23.0.bspi.1

The BSPI instances are printed to the dmesg buffer during the Hostmot2 setup sequence, one for each bspi instance included in the bitfile loaded to each installed card during the Hostmot2 setup sequence. Type "dmesg" at the terminal prompt to view the output.

PINS

mesa-7i65.N.analogue.M.out float in (M=0..7)

Analogue output values. The value will be limited to a -1.0 to +1.0 range

mesa-7i65.N.analogue.M.in float out (M=0..7)

Analogue outputs read by the 7i65 (in Volts)

mesa-7i65.N.digital.M.in bit out (M=0..3)

Miscellaneous Digital Inputs

mesa-7i65.N.enable.M.out bit in (M=0..7)

Amplifier-enable control pins

mesa-7i65.N.watchdog.has-bit bit out

Indicates the status of the 7i65 Watchdog (which is separate from the FPGA card watchdog)

PARAMETERS

mesa-7i65.N.scale-M float rw (M=0..7) (default: 10)

Analogue output scale factor. For example if the scale is 7 then an input of 1.0 will give 7V on the output terminals

mesa-7i65.N.is-bipolar-M bit rw (M=0..7) (default: 1)

Set this value to TRUE for a plus/minus "scale" output. Set to 0 for a 0-"scale" output

AUTHOR

Andy Pugh / Cliff Blackburn

LICENSE

GPL

NAME

mesa_uart – An example component demonstrating how to access the Hostmot2 UART

SYNOPSIS

```
loadrt mesa_uart [count=N|names=name1[,name2...]]
```

DESCRIPTION

This component creates 16 input and 16 output pins. It transmits {name}.N.tx-bytes on the selected UART every thread cycle and reads up to 16 bytes each cycle out of the receive FIFO and writes the values to the associated output pins. {name}.rx-bytes indicates how many pins have been written to. (pins > rx-bytes simply hold their previous value)

This module uses the names= mode of loadrt declaration to specify which UART instances to enable. A check is included to ensure that the count= option is not used instead.

The component takes parameters in the form of a comma-separated list of UART instance names, for example:

```
loadrt mesa_uart names=hm2_5i23.0 uart.0,hm2_5i23.0 uart.7
```

Note that no spaces are allowed in the string unless it is delimited by double quotes.

The UART instance names are printed to the dmesg buffer during the Hostmot2 setup sequence, one for each bspi instance included in the bitfile loaded to each installed card during the Hostmot2 setup sequence. Type "dmesg" at the terminal prompt to view the output.

The component exports two functions, send and receive, which need to be added to a realtime thread.

The above example will output data on UART channels 0 and 7 and the pins will have the names of the individual UARTS. (they need not be on the same card, or even the same bus).

Read the documents on "comp" for help with writing realtime components: <http://www.linux-cnc.org/docview/html/hal/comp.html>

FUNCTIONS

mesa uart.N.send (requires a floating-point thread)

mesa uart.N.receive (requires a floating-point thread)

PINS

mesa uart.N.tx-data-MM u32 in (MM=00..15)

 Data to be transmitted

mesa uart.N.rx-data-MM u32 out (MM=00..15)

 Data received

mesa uart.N.tx-bytes s32 in

 Number of bytes to transmit

mesa uart.N.rx-bytes s32 out

 Number of Bytes received

AUTHOR

Andy Pugh andy@bodgesoc.org

LICENSE

GPL

NAME

message – Display a message

SYNOPSIS

loadrt message [count=N|names=name1[,name2...]] [messages=N]

messages

The messages to display. These should be listed, comma-delimited, inside a single set of quotes. See the "Description" section for an example. If there are more messages than "count" or "names" then the excess will be ignored. If there are fewer messages than "count" or "names" then an error will be raised and the component will not load.

DESCRIPTION

Allows HAL pins to trigger a message. Example hal commands:

```
loadrt message names=oillow,oilpressure,inverterfail messages="Slideway oil low,No oil pressure,Spindle
inverter fault"
addf oillow servo-thread
addf oilpressure servo-thread
addf inverterfail servo-thread
```

```
setp oillow.edge 0 #this pin should be active low
net no-oil classicladder.0.out-21 oillow.trigger
net no-pressure classicladder.0.out-22 oilpressure.trigger
net no-inverter classicladder.0.out-23 inverterfail.trigger
```

When any pin goes active, the corresponding message will be displayed.

FUNCTIONS**message.*N***

Display a message

PINS**message.*N*.trigger** bit in (default: *FALSE*)

signal that triggers the message

message.*N*.force bit in (default: *FALSE*)

A *FALSE*->*TRUE* transition forces the message to be displayed again if the trigger is active

PARAMETERS**message.*N*.edge** bit rw (default: *TRUE*)

Selects the desired edge: *TRUE* means falling, *FALSE* means rising

LICENSE

GPL v2

NAME

minmax – Track the minimum and maximum values of the input to the outputs

SYNOPSIS

loadrt minmax [count=N|names=name1[,name2...]]

FUNCTIONS

minmax.N (requires a floating-point thread)

PINS

minmax.N.in float in

minmax.N.reset bit in

When reset is asserted, 'in' is copied to the outputs

minmax.N.max float out

minmax.N.min float out

LICENSE

GPL

NAME

motion – accepts NML motion commands, interacts with HAL in realtime

SYNOPSIS

```
loadrt motmod [base_period_nsec=period] [base_thread_fp=0 or 1] [servo_period_nsec=period]
[traj_period_nsec=period] [num_joints={0-9}] ([num_dio={1-64}] [num_aio={1-16}])
```

DESCRIPTION

By default, the base thread does not support floating point. Software stepping, software encoder counting, and software pwm do not use floating point. **base_thread_fp** can be used to enable floating point in the base thread (for example for brushless DC motor control).

These pins and parameters are created by the realtime **motmod** module. This module provides a HAL interface for LinuxCNC's motion planner. Basically **motmod** takes in a list of waypoints and generates a nice blended and constraint-limited stream of joint positions to be fed to the motor drives.

Optionally the number of Digital I/O is set with num_dio. The number of Analog I/O is set with num_aio. The default is 4 each.

Pin names starting with "**axis**" are actually joint values, but the pins and parameters are still called "**axis.N**". They are read and updated by the motion-controller function.

PINS**axis.N.amp-enable-out** OUT BIT

TRUE if the amplifier for this joint should be enabled

axis.N.amp-fault-in IN BIT

Should be driven TRUE if an external fault is detected with the amplifier for this joint

axis.N.home-sw-in IN BIT

Should be driven TRUE if the home switch for this joint is closed

axis.N.homing OUT BIT

TRUE if the joint is currently homing

axis.N.index-enable IO BIT

Should be attached to the index-enable pin of the joint's encoder to enable homing to index pulse

axis.N.is-unlocked IN BIT

If the axis is a locked rotary the unlocked sensor should be connected to this pin

axis.N.jog-counts IN S32

Connect to the "counts" pin of an external encoder to use a physical jog wheel.

axis.N.jog-enable IN BIT

When TRUE (and in manual mode), any change to "jog-counts" will result in motion. When false, "jog-counts" is ignored.

axis.N.jog-scale IN FLOAT

Sets the distance moved for each count on "jog-counts", in machine units.

axis.N.jog-vel-mode IN BIT

When FALSE (the default), the jogwheel operates in position mode. The axis will move exactly jog-scale units for each count, regardless of how long that might take. When TRUE, the wheel operates in velocity mode - motion stops when the wheel stops, even if that means the commanded motion is not completed.

axis.N.joint-pos-cmd OUT FLOAT

The joint (as opposed to motor) commanded position. There may be several offsets between the joint and motor coordinates: backlash compensation, screw error compensation, and home offsets.

axis.N.joint-pos-fb OUT FLOAT

The joint feedback position. This value is computed from the actual motor position minus joint offsets. Useful for machine visualization.

axis.N.motor-pos-cmd OUT FLOAT

The commanded position for this joint.

axis.N.motor-pos-fb IN FLOAT

The actual position for this joint.

axis.N.neg-lim-sw-in IN BIT

Should be driven TRUE if the negative limit switch for this joint is tripped.

axis.N.pos-lim-sw-in IN BIT

Should be driven TRUE if the positive limit switch for this joint is tripped.

axis.N.unlock OUT BIT

TRUE if the axis is a locked rotary and a move is commanded.

motion.adaptive-feed IN FLOAT

When adaptive feed is enabled with M52 P1, the commanded velocity is multiplied by this value. This effect is multiplicative with the NML-level feed override value and motion.feed-hold.

motion.analog-in-NN IN FLOAT

These pins are used by M66 Enn wait-for-input mode.

motion.analog-out-NN OUT FLOAT

These pins are used by M67-68.

motion.coord-error OUT BIT

TRUE when motion has encountered an error, such as exceeding a soft limit

motion.coord-mode OUT BIT

TRUE when motion is in "coordinated mode", as opposed to "teleop mode"

motion.current-vel OUT FLOAT
Current cartesian velocity

motion.digital-in-NN IN BIT
These pins are used by M66 Pnn wait-for-input mode.

motion.digital-out-NN OUT BIT
These pins are controlled by the M62 through M65 words.

motion.distance-to-go OUT FLOAT
Distance remaining in the current move

motion.enable IN BIT
If this bit is driven FALSE, motion stops, the machine is placed in the "machine off" state, and a message is displayed for the operator. For normal motion, drive this bit TRUE.

motion.feed-hold IN BIT
When Feed Stop Control is enabled with M53 P1, and this bit is TRUE, the feed rate is set to 0.

motion.feed-inhibit IN BIT
When this bit is TRUE, the feed rate is set and held to 0. This will be delayed during spindle synch moves till the end of the move.

motion.in-position OUT BIT
TRUE if the machine is in position (ie, not currently moving towards the commanded position).

motion.probe-input IN BIT
G38.x uses the value on this pin to determine when the probe has made contact. TRUE for probe contact closed (touching), FALSE for probe contact open.

motion.program-line OUT S32

motion.requested-vel OUT FLOAT
The requested velocity with no adjustments for feed override

motion.spindle-at-speed IN BIT
Motion will pause until this pin is TRUE, under the following conditions: before the first feed move after each spindle start or speed change; before the start of every chain of spindle-synchronized moves; and if in CSS mode, at every rapid->feed transition.

motion.spindle-brake OUT BIT
TRUE when the spindle brake should be applied

motion.spindle-forward OUT BIT
TRUE when the spindle should rotate forward

motion.spindle-index-enable I/O BIT
For correct operation of spindle synchronized moves, this signal must be hooked to the index-enable pin of the spindle encoder.

motion.spindle-inhibit IN BIT

When TRUE, the spindle speed is set and held to 0.

motion.spindle-on OUT BIT

TRUE when spindle should rotate

motion.spindle-reverse OUT BIT

TRUE when the spindle should rotate backward

motion.spindle-revs IN FLOAT

For correct operation of spindle synchronized moves, this signal must be hooked to the position pin of the spindle encoder.

motion.spindle-speed-in IN FLOAT

Actual spindle speed feedback in revolutions per second; used for G96 (constant surface speed) and G95 (feed per revolution) modes.

motion.spindle-speed-out OUT FLOAT

Desired spindle speed in rotations per minute

motion.spindle-speed-out-abs OUT FLOAT

Desired spindle speed in rotations per minute, always positive regardless of spindle direction.

motion.spindle-speed-out-rps OUT float

Desired spindle speed in rotations per second

motion.spindle-speed-out-rps-abs OUT float

Desired spindle speed in rotations per second, always positive regardless of spindle direction.

motion.spindle-orient-angle OUT FLOAT

Desired spindle orientation for M19. Value of the M19 R word parameter plus the value of the [RS274NGC]ORIENT_OFFSET ini parameter.

motion.spindle-orient-mode OUT BIT

Desired spindle rotation mode. Reflects M19 P parameter word.

motion.spindle-orient OUT BIT

Indicates start of spindle orient cycle. Set by M19. Cleared by any of M3,M4,M5. If spindle-orient-fault is not zero during spindle-orient true, the M19 command fails with an error message.

motion.spindle-is-oriented IN BIT

Acknowledge pin for spindle-orient. Completes orient cycle. If spindle-orient was true when spindle-is-oriented was asserted, the spindle-orient pin is cleared and the spindle-locked pin is asserted. Also, the spindle-brake pin is asserted.

motion.spindle-orient-fault IN S32

Fault code input for orient cycle. Any value other than zero will cause the orient cycle to abort.

motion.spindle-locked OUT BIT

Spindle orient complete pin. Cleared by any of M3,M4,M5.

motion.teleop-mode OUT bit**motion.tooloffset.x** OUT FLOAT**motion.tooloffset.y** OUT FLOAT**motion.tooloffset.z** OUT FLOAT**motion.tooloffset.a** OUT FLOAT**motion.tooloffset.b** OUT FLOAT**motion.tooloffset.c** OUT FLOAT**motion.tooloffset.u** OUT FLOAT**motion.tooloffset.v** OUT FLOAT**motion.tooloffset.w** OUT FLOAT

Current tool offset in all 9 axes.

DEBUGGING PINS

Many of the pins below serve as debugging aids, and are subject to change or removal at any time.

axis.N.active OUT BIT

TRUE when this joint is active

axis.N.backlash-corr OUT FLOAT

Backlash or screw compensation raw value

axis.N.backlash-filt OUT FLOAT

Backlash or screw compensation filtered value (respecting motion limits)

axis.N.backlash-vel OUT FLOAT

Backlash or screw compensation velocity

axis.N.coarse-pos-cmd OUT FLOAT**axis.N.error** OUT BIT

TRUE when this joint has encountered an error, such as a limit switch closing

axis.N.f-error OUT FLOAT

The actual following error

axis.N.f-error-lim OUT FLOAT

The following error limit

axis.N.f-errored OUT BIT

TRUE when this joint has exceeded the following error limit

axis.N.faulted OUT BIT**axis.N.free-pos-cmd** OUT FLOAT

The "free planner" commanded position for this joint.

axis.N.free-tp-enable OUT BIT

TRUE when the "free planner" is enabled for this joint

axis.N.free-vel-lim OUT FLOAT

The velocity limit for the free planner

axis.N.homed OUT BIT

TRUE if the joint has been homed

axis.N.in-position OUT BIT

TRUE if the joint is using the "free planner" and has come to a stop

axis.N.joint-vel-cmd OUT FLOAT

The joint's commanded velocity

axis.N.kb-jog-active OUT BIT**axis.N.neg-hard-limit** OUT BIT

The negative hard limit for the joint

axis.N.pos-hard-limit OUT BIT

The positive hard limit for the joint

axis.N.wheel-jog-active OUT BIT**motion.motion-enabled** OUT BIT**motion.motion-type** OUT S32

These values are from src/emc/nml_intf/motion_types.h

1: Traverse

2: Linear feed

3: Arc feed

4: Tool change

5: Probing

6: Rotary axis indexing

motion.on-soft-limit OUT BIT

motion.program-line OUT S32

motion.teleop-mode OUT BIT

TRUE when motion is in "teleop mode", as opposed to "coordinated mode"

PARAMETERS

Many of the parameters serve as debugging aids, and are subject to change or removal at any time.

motion-command-handler.time

motion-command-handler.tmax

motion-controller.time

motion-controller.tmax

Show information about the execution time of these HAL functions in CPU cycles

motion.debug-*

These values are used for debugging purposes.

motion.servo.last-period

The number of CPU cycles between invocations of the servo thread. Typically, this number divided by the CPU speed gives the time in seconds, and can be used to determine whether the realtime motion controller is meeting its timing constraints

motion.servo.overruns

By noting large differences between successive values of motion.servo.last-period, the motion controller can determine that there has probably been a failure to meet its timing constraints. Each time such a failure is detected, this value is incremented.

FUNCTIONS

Generally, these functions are both added to the servo-thread in the order shown.

motion-command-handler

Processes motion commands coming from user space

motion-controller

Runs the LinuxCNC motion controller

BUGS

This manual page is horribly incomplete.

MOTION(9)

HAL Component

MOTION(9)

SEE ALSO

[iocontrol\(1\)](#)

NAME

mult2 – Product of two inputs

SYNOPSIS

loadrt mult2 [count=N|names=name1[,name2...]]

FUNCTIONS

mult2.N (requires a floating-point thread)

PINS

mult2.N.in0 float in

mult2.N.in1 float in

mult2.N.out float out

out = in0 * in1

LICENSE

GPL

NAME

multiclick – Single-, double-, triple-, and quadruple-click detector

SYNOPSIS

```
loadrt multiclick [count=N|names=name1[,name2...]]
```

DESCRIPTION

A click is defined as a rising edge on the 'in' pin, followed by the 'in' pin being True for at most 'max-hold-ns' nanoseconds, followed by a falling edge.

A double-click is defined as two clicks, separated by at most 'max-space-ns' nanoseconds with the 'in' pin in the False state.

I bet you can guess the definition of triple- and quadruple-click.

You probably want to run the input signal through a debounce component before feeding it to the multiclick detector, if the input is at all noisy.

The '*-click' pins go high as soon as the input detects the correct number of clicks.

The '*-click-only' pins go high a short while after the click, after the click separator space timeout has expired to show that no further click is coming. This is useful for triggering halui MDI commands.

FUNCTIONS**multiclick.N**

Detect single-, double-, triple-, and quadruple-clicks

PINS**multiclick.N.in** bit in

The input line, this is where we look for clicks.

multiclick.N.single-click bit out

Goes high briefly when a single-click is detected on the 'in' pin.

multiclick.N.single-click-only bit out

Goes high briefly when a single-click is detected on the 'in' pin and no second click followed it.

multiclick.N.double-click bit out

Goes high briefly when a double-click is detected on the 'in' pin.

multiclick.N.double-click-only bit out

Goes high briefly when a double-click is detected on the 'in' pin and no third click followed it.

multiclick.N.triple-click bit out

Goes high briefly when a triple-click is detected on the 'in' pin.

multiclick.N.triple-click-only bit out

Goes high briefly when a triple-click is detected on the 'in' pin and no fourth click followed it.

multiclick.N.quadruple-click bit out

Goes high briefly when a quadruple-click is detected on the 'in' pin.

multiclick.N.quadruple-click-only bit out

Goes high briefly when a quadruple-click is detected on the 'in' pin and no fifth click followed it.

multiclick.N.state s32 out**PARAMETERS****multiclick.N.invert-input** bit rw (default: *FALSE*)

If FALSE (the default), clicks start with rising edges. If TRUE, clicks start with falling edges.

multiclick.N.max-hold-ns u32 rw (default: 250000000)

If the input is held down longer than this, it's not part of a multi-click. (Default 250,000,000 ns, 250 ms.)

multiclick.N.max-space-ns u32 rw (default: 250000000)

If the input is released longer than this, it's not part of a multi-click. (Default 250,000,000 ns, 250 ms.)

multiclick.N.output-hold-ns u32 rw (default: 100000000)

Positive pulses on the output pins last this long. (Default 100,000,000 ns, 100 ms.)

LICENSE

GPL

NAME

multiswitch – This component toggles between a specified number of output bits

SYNOPSIS

loadrt multiswitch personality=*P* [cfg=*N*]

cfg cfg should be a comma-separated list of sizes for example cfg=2,4,6 would create 3 instances of 2, 4 and 6 bits respectively.
Ignore the "personality" parameter, that is auto-generated

FUNCTIONS

multiswitch.*N* (requires a floating-point thread)

PINS

multiswitch.*N.up* bit in (default: *false*)

Receives signal to toggle up

multiswitch.*N.down* bit in (default: *false*)

Receives signal to toggle down

multiswitch.*N.bit-MM* bit out (MM=00..personality) (default: *false*)

Output bits

PARAMETERS

multiswitch.*N.top-position* u32 rw

Number of positions

multiswitch.*N.position* s32 rw

Current state (may be set in the HAL)

AUTHOR

ArcEye schooner30@tiscali.co.uk / Andy Pugh andy@bodgesoc.org

LICENSE

GPL

NAME

mux16 – Select from one of sixteen input values

SYNOPSIS

loadrt mux16 [count=N|names=name1[,name2...]]

FUNCTIONS

mux16.N (requires a floating-point thread)

PINS

mux16.N.use-graycode bit in

This signifies the input will use Gray code instead of binary. Gray code is a good choice when using physical switches because for each increment only one select input changes at a time.

mux16.N.suppress-no-input bit in

This suppresses changing the output if all select lines are false. This stops unwanted jumps in output between transitions of input. but make in00 unavailable.

mux16.N.debounce-time float in

sets debounce time in seconds. eg. .10 = a tenth of a second input must be stable this long before outputs changes. This helps to ignore 'noisy' switches.

mux16.N.selM bit in (M=0..3)

Together, these determine which **inN** value is copied to **out**.

mux16.N.out-f float out

mux16.N.out-s s32 out

Follows the value of one of the **inN** values according to the four **sel** values and whether use-graycode is active. The s32 value will be truncated and limited to the max and min values of signed values.

sel3=FALSE, sel2=FALSE, sel1=FALSE, sel0=FALSE

out follows **in0**

sel3=FALSE, sel2=FALSE, sel1=FALSE, sel0=TRUE

out follows **in1**

etc.

mux16.N.inMM float in (MM=00..15)

array of selectable outputs

PARAMETERS

mux16.N.elapsed float r

Current value of the internal debounce timer
for debugging.

mux16.N.selected s32 r

Current value of the internal selection variable after conversion
for debugging

LICENSE

GPL

NAME

mux2 – Select from one of two input values

SYNOPSIS

loadrt mux2 [count=N|names=name1[,name2...]]

FUNCTIONS

mux2.N (requires a floating-point thread)

PINS

mux2.N.sel bit in

mux2.N.out float out

Follows the value of in0 if sel is FALSE, or in1 if sel is TRUE

mux2.N.in1 float in

mux2.N.in0 float in

LICENSE

GPL

NAME

mux4 – Select from one of four input values

SYNOPSIS

loadrt mux4 [count=N|names=name1[,name2...]]

FUNCTIONS

mux4.N (requires a floating-point thread)

PINS

mux4.N.sel0 bit in

mux4.N.sel1 bit in

Together, these determine which **inN** value is copied to **out**.

mux4.N.out float out

Follows the value of one of the **inN** values according to the two **sel** values

sel1=FALSE, sel0=FALSE

out follows **in0**

sel1=FALSE, sel0=TRUE

out follows **in1**

sel1=TRUE, sel0=FALSE

out follows **in2**

sel1=TRUE, sel0=TRUE

out follows **in3**

mux4.N.in0 float in

mux4.N.in1 float in

mux4.N.in2 float in

mux4.N.in3 float in

LICENSE

GPL

NAME

mux8 – Select from one of eight input values

SYNOPSIS

loadrt mux8 [count=N|names=name1[,name2...]]

FUNCTIONS

mux8.N (requires a floating-point thread)

PINS

mux8.N.sel0 bit in

mux8.N.sel1 bit in

mux8.N.sel2 bit in

Together, these determine which **inN** value is copied to **out**.

mux8.N.out float out

Follows the value of one of the **inN** values according to the three **sel** values

sel2=FALSE, sel1=FALSE, sel0=FALSE

out follows **in0**

sel2=FALSE, sel1=FALSE, sel0=TRUE

out follows **in1**

sel2=FALSE, sel1=TRUE, sel0=FALSE

out follows **in2**

sel2=FALSE, sel1=TRUE, sel0=TRUE

out follows **in3**

sel2=TRUE, sel1=FALSE, sel0=FALSE

out follows **in4**

sel2=TRUE, sel1=FALSE, sel0=TRUE

out follows **in5**

sel2=TRUE, sel1=TRUE, sel0=FALSE

out follows **in6**

sel2=TRUE, sel1=TRUE, sel0=TRUE

out follows **in7**

mux8.N.in0 float in

mux8.N.in1 float in

mux8.N.in2 float in

mux8.N.in3 float in

mux8.N.in4 float in

mux8.N.in5 float in

mux8.N.in6 float in

mux8.N.in7 float in

LICENSE

GPL

NAME

mux_generic – choose one from several input values

SYNOPSIS

```
loadrt mux_generic config="bb8,fu12...."
```

FUNCTIONS

mux-gen.NN Depending on the data types can run in either a floating point or non-floating point thread.

PINS

mux-gen.NN.suppress-no-input bit in

This suppresses changing the output if all select lines are false. This stops unwanted jumps in output between transitions of input. but makes in00 unavailable.

mux-gen.NN.debounce-us unsigned in

sets debounce time in microseconds. eg. 100000 = a tenth of a second. The selection inputs must be stable this long before the output changes. This helps to ignore 'noisy' switches.

mux-gen.NN.sel-bitMMM bit in (M=0..N)

mux-gen.NN.sel-intMMM unsigned in

Together, these determine which **inN** value is copied to **output**. The bit pins are interpreted as binary bits, and the result is simply added on to the integer pin input. It is expected that either one or the other would normally be used. However, the possibility exists to use a higher-order bit to "shift" the values set by the integer pin. The sel-bit pins are only created when the size of the mux_gen component is an integer power of two. This component (unlike mux16) does not offer the option of decoding gray-code, however the same effect can be achieved by arranging the order of the input values to suit.

mux-gen.NN.out-[bit/float/s32/u32] variable-type out

Follows the value of one of the **inN** values according to the selection bits and/or the selection number. Values will be converted/truncated according to standard C rules. This means, for example that a float input greater than 2147483647 will give an S32 output of -2147483648.

mux-gen.NN.in-[bit/float/s32/u32]-MM variable-type in

The possible output values that are selected by the selection pins.

PARAMETERS

mux-gen.N.elapsed float r

Current value of the internal debounce timer for debugging.

mux-gen.N.selected s32 r

Current value of the internal selection variable after conversion for debugging. Possibly useful for setting up gray-code switches.

DESCRIPTION

This component is a more general version of the other multiplexing components. It allows the creation of arbitrary-size multiplexers (up to 1024 entries) and also supports differing data types on the input and output pins. The configuration string is a comma-separated list of code-letters and numbers, such as "bb4,fu12". This would create a 4-element bit-to-bit mux and a 12-element float-to-unsigned mux. The code letters are b = bit, f = float, s = signed integer, u = unsigned integer. The first letter code is the input type, the second is the output type. The codes are not case-sensitive. The order of the letters is significant but the position in the string is not. Do not insert any spaces in the config string. Any non-zero float value will be

converted to a "true" output in bit form. Be wary that float datatypes can be very, very, close to zero and not actually be equal to zero.

Each mux has its own HAL function and must be added to a thread separately. If neither input nor output is of type float then the function is base-thread (non floating-point) safe. Any mux_generic with a floating point input or output can only be added to a floating-point thread.

LICENSE

GPL

AUTHOR

Andy Pugh

NAME

near – Determine whether two values are roughly equal.

SYNOPSIS

loadrt near [count=N|names=name1[,name2...]]

FUNCTIONS

near.N (requires a floating-point thread)

PINS

near.N.in1 float in

near.N.in2 float in

near.N.out bit out

out is true if **in1** and **in2** are within a factor of **scale** (i.e., for **in1** positive, **in1/scale** <= **in2** <= **in1*scale**), OR if their absolute difference is no greater than **difference** (i.e., $|in1-in2| \leq difference$). **out** is false otherwise.

PARAMETERS

near.N.scale float rw (default: *I*)

near.N.difference float rw (default: *O*)

LICENSE

GPL

NAME

not – Inverter

SYNOPSIS

loadrt not [count=N|names=name1[,name2...]]

FUNCTIONS

not.N

PINS

not.N.in bit in

not.N.out bit out

LICENSE

GPL

NAME

offset – Adds an offset to an input, and subtracts it from the feedback value

SYNOPSIS

loadrt offset [count=N|names=name1[,name2...]]

FUNCTIONS

offset.N.update-output (requires a floating-point thread)

Updated the output value by adding the offset to the input

offset.N.update-feedback (requires a floating-point thread)

Update the feedback value by subtracting the offset from the feedback

PINS

offset.N.offset float in

The offset value

offset.N.in float in

The input value

offset.N.out float out

The output value

offset.N.fb-in float in

The feedback input value

offset.N.fb-out float out

The feedback output value

LICENSE

GPL

NAME

oneshot – one-shot pulse generator

SYNOPSIS

loadrt oneshot [count=N|names=name1[,name2...]]

DESCRIPTION

creates a variable-length output pulse when the input changes state. This function needs to run in a thread which supports floating point (typically the servo thread). This means that the pulse length has to be a multiple of that thread period, typically 1mS. For a similar function that can run in the base thread, and which offers higher resolution, see "edge".

FUNCTIONS

oneshot.N (requires a floating-point thread)

 Produce output pulses from input edges

PINS

oneshot.N.in bit in
 Trigger input

oneshot.N.out bit out
 Active high pulse

oneshot.N.out-not bit out
 Active low pulse

oneshot.N.width float in (default: 0)
 Pulse width in seconds

oneshot.N.time-left float out
 Time left in current output pulse

PARAMETERS

oneshot.N.retriggerable bit rw (default: *TRUE*)
 Allow additional edges to extend pulse

oneshot.N.rising bit rw (default: *TRUE*)
 Trigger on rising edge

oneshot.N.falling bit rw (default: *FALSE*)
 Trigger on falling edge

LICENSE

GPL

NAME

opto_ac5 – Realtime driver for opto22 PCI-AC5 cards

SYNOPSIS

loadrt opto_ac5 [portconfig0=0xN] [portconfig1=0xN]

DESCRIPTION

These pins and parameters are created by the realtime **opto_ac5** module. The portconfig0 and portconfig1 variables are used to configure the two ports of each card. The first 24 bits of a 32 bit number represent the 24 i/o points of each port. The lowest (rightmost) bit would be HAL pin 0 which is header connector pin 47. Then next bit to the left would be HAL pin 1, header connector pin 45 and so on, until bit 24 would be HAL pin 23 , header connector pin 1. "1" bits represent output points. So channel 0..11 as inputs and 12..23 as outputs would be represented by (in binary) 111111111110000000000000 which is 0xffff000 in hexadecimal. That is the number you would use Eg. loadrt opto_ac5 portconfig0=0xffff000

If no portconfig variable is specified the default configuration is 12 inputs then 12 outputs.

Up to 4 boards are supported. Boards are numbered starting at 0.

Portnumber can be 0 or 1. Port 0 is closest to the card bracket.

PINS

opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER] OUT bit

opto_ac5.[BOARDNUMBER].port[PORTNUMBER].in-[PINNUMBER]-not OUT bit

Connect a hal bit signal to this pin to read an i/o point from the card. The PINNUMBER represents the position in the relay rack. Eg. PINNUMBER 0 is position 0 in a opto22 relay rack and would be pin 47 on the 50 pin header connector. The **-not** pin is inverted so that LOW gives TRUE and HIGH gives FALSE.

opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER] IN bit

Connect a hal bit signal to this pin to write to an i/o point of the card. The PINNUMBER represents the position in the relay rack.Eg. PINNUMBER 23 is position 23 in a opto22 relay rack and would be pin 1 on the 50 pin header connector.

opto_ac5.[BOARDNUMBER].led[NUMBER] OUT bit

Turns one of the on board LEDS on/off. LEDS are numbered 0 to 3.

PARAMETERS

opto_ac5.[BOARDNUMBER].port[PORTNUMBER].out-[PINNUMBER]-invert W bit

When TRUE, invert the meaning of the corresponding **-out** pin so that TRUE gives LOW and FALSE gives HIGH.

FUNCTIONS

opto_ac5.0.digital-read

Add this to a thread to read all the input points.

opto_ac5.0.digital-write

Add this to a thread to write all the output points and LEDS.

BUGS

All boards are loaded with the same port configurations as the first board.

SEE ALSO

<http://wiki.linuxcnc.org/cgi-bin/wiki.pl?OptoPciAc5>

NAME

or2 – Two-input OR gate

SYNOPSIS

loadrt or2 [count=N|names=name1[,name2...]]

FUNCTIONS

or2.N

PINS

or2.N.in0 bit in

or2.N.in1 bit in

or2.N.out bit out

out is computed from the value of **in0** and **in1** according to the following rule:

in0=FALSE in1=FALSE

out=FALSE

Otherwise,

out=TRUE

LICENSE

GPL

NAME

orient – Provide a PID command input for orientation mode based on current spindle position, target angle and orient mode

SYNOPSIS

```
loadrt orient [count=N|names=name1[,name2...]]
```

DESCRIPTION

This component is designed to support a spindle orientation PID loop by providing a command value, and fit with the motion spindle-orient support pins to support the M19 code.

The spindle is assumed to have stopped in an arbitrary position. The spindle encoder position is linked to the **position** pin. The current value of the position pin is sampled on a positive edge on the **enable** pin, and **command** is computed and set as follows: floor(number of full spindle revolutions in the **position** sampled on positive edge) plus **angle**/360 (the fractional revolution) +1/-1/0 depending on **mode**.

The **mode** pin is interpreted as follows:

0: the spindle rotates in the direction with the lesser angle, which may be clockwise or counterclockwise.

1: the spindle rotates always rotates clockwise to the new angle.

2: the spindle rotates always rotates counterclockwise to the new angle.

HAL USAGE

On **motion.spindle-orient** disconnect the spindle control and connect to the orient-pid loop:

```
loadrt orient names=orient
loadrt pid names=orient-pid
net orient-angle motion.spindle-orient-angle orient.angle
net orient-mode motion.spindle-orient-mode orient.mode
net orient-enable motion.spindle-orient orient.enable orient-pid.enable
net spindle-pos ...encoder..position orient.position orient-pid.feedback
net orient-command orient.command orient-pid.command
```

FUNCTIONS

orient.N (requires a floating-point thread)

Update **command** based on **enable**, **position**, **mode** and **angle**.

PINS

orient.N.enable bit in
enable angular output for orientation mode

orient.N.mode s32 in
0: rotate - shortest move; 1: always rotate clockwise; 2: always rotate counterclockwise

orient.N.position float in
spindle position input, unit 1 rev

orient.N.angle float in
orient target position in degrees, $0 \leq \text{angle} < 360$

orient.N.command float out
target spindle position, input to PID command

ORIENT(9)

HAL Component

ORIENT(9)

orient.N.poserr float out
in degrees - aid for PID tuning

AUTHOR

Michael Haberler

LICENSE

GPL

NAME

pcl720 – Driver for the Advantech PCL 720 card.

SYNOPSIS

loadrt pcl720 [ioaddr=N]

ioaddr Base address of card. Separate each card base address with a comma but no space to load more than one card. eg loadrt pcl720 ioaddr=0x200,0x200. use 0xNNN to define addresses in Hex

DESCRIPTION

This driver supports the Advantech PCL720 ISA card. It might work with the PCI version too, but this is untested.

It creates hal pins corresponding to the digital inputs and outputs, but does not support the counters/timers.

FUNCTIONS

pcl720.N.read

Reads each of the digital inputs and updates the HAL pins

pcl720.N.write

Writes the values of the output HAL pins to the digital IO

pcl720.N.reset

Waits for the length of time specified by the **reset-time** parameter and resets any pins for which the **reset** parameter has been set. This can be used to allow step generators to make a step every thread rather than every other thread. This function must be added to the thread after the "write" function.

Do not use this function if you do not wish to reset any pins.

the stepgen **step-space** parameter should be set to 0 to make use of this function.

PINS

pcl720.N.pin-MM-out bit in (MM=00..31)

Output pins

pcl720.N.pin-MM-in bit out (MM=00..31)

Input pins

pcl720.N.pin-MM-in-not bit out (MM=00..31)

Inverted version of each input pin

pcl720.N.wait-clocks u32 out

PARAMETERS

pcl720.N.reset-time u32 rw (default: 5000)

The time in nanoseconds after the write function has run to reset the pins for which the "reset" parameter is set.

pcl720.N.pin-MM-reset bit rw (MM=00..31)

specifies if the pin should be reset by the "reset" function

pcl720.N.pin-MM-out-invert bit rw (MM=00..31)

Set to true to invert the sense of the output pin

AUTHOR

Andy Pugh

LICENSE

GPL

NAME

pid – proportional/integral/derivative controller

SYNOPSIS

```
loadrt pid [num_chan=num | names=name1[,name2...]] [debug=dbg]
```

DESCRIPTION

pid is a classic Proportional/Integral/Derivative controller, used to control position or speed feedback loops for servo motors and other closed-loop applications.

pid supports a maximum of sixteen controllers. The number that are actually loaded is set by the **num_chan** argument when the module is loaded. Alternatively, specify **names=** and unique names separated by commas.

The **num_chan=** and **names=** specifiers are mutually exclusive. If neither **num_chan=** nor **names=** are specified, the default value is three. If **debug** is set to 1 (the default is 0), some additional HAL parameters will be exported, which might be useful for tuning, but are otherwise unnecessary.

NAMING

The names for pins, parameters, and functions are prefixed as:

pid.N. for N=0,1,...,num-1 when using **num_chan=num**

nameN. for nameN=name1,name2,... when using **names=name1,name2,...**

The **pid.N.** format is shown in the following descriptions.

FUNCTIONS

pid.N.do-pid-calc (uses floating-point) Does the PID calculations for control loop *N*.

PINS

pid.N.command float in

The desired (commanded) value for the control loop.

pid.N.Pgain float in

Proportional gain. Results in a contribution to the output that is the error multiplied by **Pgain**.

pid.N.Igain float in

Integral gain. Results in a contribution to the output that is the integral of the error multiplied by **Igain**. For example an error of 0.02 that lasted 10 seconds would result in an integrated error (**errorI**) of 0.2, and if **Igain** is 20, the integral term would add 4.0 to the output.

pid.N.Dgain float in

Derivative gain. Results in a contribution to the output that is the rate of change (derivative) of the error multiplied by **Dgain**. For example an error that changed from 0.02 to 0.03 over 0.2 seconds would result in an error derivative (**errorD**) of 0.05, and if **Dgain** is 5, the derivative term would add 0.25 to the output.

pid.N.feedback float in

The actual (feedback) value, from some sensor such as an encoder.

pid.N.output float out

The output of the PID loop, which goes to some actuator such as a motor.

pid.N.command-deriv float in

The derivative of the desired (commanded) value for the control loop. If no signal is connected then the derivative will be estimated numerically.

pid.N.feedback-deriv float in

The derivative of the actual (feedback) value for the control loop. If no signal is connected then the derivative will be estimated numerically. When the feedback is from a quantized position

source (e.g., encoder feedback position), behavior of the D term can be improved by using a better velocity estimate here, such as the velocity output of encoder(9) or hostmot2(9).

pid.N.error-previous-target bit in

Use previous invocation's target vs. current position for error calculation, like the motion controller expects. This may make torque-mode position loops and loops requiring a large I gain easier to tune, by eliminating velocity-dependent following error.

pid.N.error float out

The difference between command and feedback.

pid.N.enable bit in

When true, enables the PID calculations. When false, **output** is zero, and all internal integrators, etc, are reset.

pid.N.index-enable bit in

On the falling edge of **index-enable**, pid does not update the internal command derivative estimate. On systems which use the encoder index pulse, this pin should be connected to the index-enable signal. When this is not done, and FF1 is nonzero, a step change in the input command causes a single-cycle spike in the PID output. On systems which use exactly one of the **-deriv** inputs, this affects the D term as well.

pid.N.bias float in

bias is a constant amount that is added to the output. In most cases it should be left at zero. However, it can sometimes be useful to compensate for offsets in servo amplifiers, or to balance the weight of an object that moves vertically. **bias** is turned off when the PID loop is disabled, just like all other components of the output. If a non-zero output is needed even when the PID loop is disabled, it should be added with an external HAL sum2 block.

pid.N.FF0 float in

Zero order feed-forward term. Produces a contribution to the output that is **FF0** multiplied by the commanded value. For position loops, it should usually be left at zero. For velocity loops, **FF0** can compensate for friction or motor counter-EMF and may permit better tuning if used properly.

pid.N.FF1 float in

First order feed-forward term. Produces a contribution to the output that is **FF1** multiplied by the derivative of the commanded value. For position loops, the contribution is proportional to speed, and can be used to compensate for friction or motor CEMF. For velocity loops, it is proportional to acceleration and can compensate for inertia. In both cases, it can result in better tuning if used properly.

pid.N.FF2 float in

Second order feed-forward term. Produces a contribution to the output that is **FF2** multiplied by the second derivative of the commanded value. For position loops, the contribution is proportional to acceleration, and can be used to compensate for inertia. For velocity loops, it should usually be left at zero.

pid.N.deadband float in

Defines a range of "acceptable" error. If the absolute value of **error** is less than **deadband**, it will be treated as if the error is zero. When using feedback devices such as encoders that are inherently quantized, the deadband should be set slightly more than one-half count, to prevent the control loop from hunting back and forth if the command is between two adjacent encoder values. When the absolute value of the error is greater than the deadband, the deadband value is subtracted from the error before performing the loop calculations, to prevent a step in the transfer function at the edge of the deadband. (See **BUGS**.)

pid.N.maxoutput float in

Output limit. The absolute value of the output will not be permitted to exceed **maxoutput**, unless **maxoutput** is zero. When the output is limited, the error integrator will hold instead of integrating, to prevent windup and overshoot.

pid.N.maxerror float in

Limit on the internal error variable used for P, I, and D. Can be used to prevent high **Pgain** values from generating large outputs under conditions when the error is large (for example, when the command makes a step change). Not normally needed, but can be useful when tuning non-linear systems.

pid.N.maxerrorD float in

Limit on the error derivative. The rate of change of error used by the **Dgain** term will be limited to this value, unless the value is zero. Can be used to limit the effect of **Dgain** and prevent large output spikes due to steps on the command and/or feedback. Not normally needed.

pid.N.maxerrorI float in

Limit on error integrator. The error integrator used by the **Igain** term will be limited to this value, unless it is zero. Can be used to prevent integrator windup and the resulting overshoot during/after sustained errors. Not normally needed.

pid.N.maxcmdD float in

Limit on command derivative. The command derivative used by **FF1** will be limited to this value, unless the value is zero. Can be used to prevent **FF1** from producing large output spikes if there is a step change on the command. Not normally needed.

pid.N.maxcmdDD float in

Limit on command second derivative. The command second derivative used by **FF2** will be limited to this value, unless the value is zero. Can be used to prevent **FF2** from producing large output spikes if there is a step change on the command. Not normally needed.

pid.N.saturated bit out

When true, the current PID output is saturated. That is,

$$\text{output} = \pm \text{maxoutput}$$

pid.N.saturated-s float out**pid.N.saturated-count** s32 out

When true, the output of PID was continually saturated for this many seconds (**saturated-s**) or periods (**saturated-count**).

PARAMETERS**pid.N.errorI** float ro (only if debug=1)

Integral of error. This is the value that is multiplied by **Igain** to produce the Integral term of the output.

pid.N.errorD float ro (only if debug=1)

Derivative of error. This is the value that is multiplied by **Dgain** to produce the Derivative term of the output.

pid.N.commandD float ro (only if debug=1)

Derivative of command. This is the value that is multiplied by **FF1** to produce the first order feed-forward term of the output.

pid.N.commandDD float ro (only if debug=1)

Second derivative of command. This is the value that is multiplied by **FF2** to produce the second order feed-forward term of the output.

BUGS

Some people would argue that deadband should be implemented such that error is treated as zero if it is within the deadband, and be unmodified if it is outside the deadband. This was not done because it would cause a step in the transfer function equal to the size of the deadband. People who prefer that behavior are welcome to add a parameter that will change the behavior, or to write their own version of **pid**. However, the default behavior should not be changed.

Negative gains may lead to unwanted behavior. It is possible in some situations that negative FF gains

make sense, but in general all gains should be positive. If some output is in the wrong direction, negating gains to fix it is a mistake; set the scaling correctly elsewhere instead.

NAME

pluto_servo – Hardware driver and firmware for the Pluto-P parallel-port FPGA, for use with servo machines.

SYNOPSIS

loadrt pluto_servo [ioaddr=N] [ioaddr_hi=N] [epp_wide=N] [watchdog=N] [test_encoder=N]

ioaddr [default: 0x378]

The base address of the parallel port.

ioaddr_hi [default: 0]

The secondary address of the parallel port, used to set EPP mode. 0 means to use ioaddr + 0x400. -1 means there is no secondary address. The secondary address is used to set the port to EPP mode.

epp_wide [default: 1]

Set to zero to disable the "wide EPP mode". "Wide" mode allows a 16- and 32-bit EPP transfers, which can reduce the time spent in the read and write functions. However, this may not work on all EPP parallel ports.

watchdog [default: 1]

Set to zero to disable the "hardware watchdog". "Watchdog" will tristate all outputs approximately 6ms after the last execution of **pluto-servo.write**, which adds some protection in the case of LinuxCNC crashes.

test_encoder [default: 0]

Internally connect dout0..2 to QA0, QB0, QZ0 to test quadrature counting

DESCRIPTION

Pluto_servo is a LinuxCNC software driver and associated firmware that allow the Pluto-P board to be used to control a servo-based CNC machine.

The driver has 4 PWM channels, 4 quadrature channels with index pulse, 18 digital outputs (8 shared with PWM), and 20 digital inputs (12 shared with quadrature).

Encoders

The encoder pins and parameters conform to the ‘canonical encoder’ interface described in the HAL manual. It operates in ‘x4 mode’.

The sample rate of the encoder is 40MHz. The maximum number quadrature rate is 8191 counts per LinuxCNC servo cycle. For correct handling of the index pulse, the number of encoder counts per revolution must be less than 8191.

PWM

The PWM pins and parameters conform to the ‘canonical analog output’ interface described in the HAL manual. The output pins are ‘up/down’ or ‘pwm/dir’ pins as described in the documentation of the ‘pwm-gen’ component.

Internally the PWM generator is based on a 12-bit, 40MHz counter, giving 4095 duty cycles from -100% to +100% and a frequency of approximately 19.5kHz. In PDM mode, the duty periods are approximately 100ns long.

Digital I/O

The digital output pins conform to the ‘canonical digital output’ interface described in the HAL manual.

The digital input pins conform to the ‘canonical digital input’ interface described in the HAL manual.

FUNCTIONS

pluto-servo.read (requires a floating-point thread)
 Read all the inputs from the pluto-servo board

pluto-servo.write (requires a floating-point thread)
 Write all the outputs on the pluto-servo board

PINS

pluto-servo.encoder.M.count s32 out (M=0..3)
pluto-servo.encoder.M.position float out (M=0..3)
pluto-servo.encoder.M.velocity float out (M=0..3)
pluto-servo.encoder.M.reset bit in (M=0..3)
pluto-servo.encoder.M.index-enable bit io (M=0..3)

encoder.*M* corresponds to the pins labeled QAM, QBM, and QZM on the pinout diagram

pluto-servo.pwm.M.value float in (M=0..3)
pluto-servo.pwm.M.enable bit in (M=0..3)

pwm.*M* corresponds to the pins labeled UPM and DNM on the pinout diagram

pluto-servo.dout.MM bit in (MM=00..19)

dout.*MM* corresponds to the pin labeled OUT*M* on the pinout diagram. Other pins are shared with the PWM function, as follows:

Pin	Shared
Label	with
dout.10	UP0
dout.10	UP0
dout.12	UP1
dout.14	UP2
dout.18	UP3
dout.11	DOWN0
dout.13	DOWN1
dout.15	DOWN2
dout.19	DOWN3

pluto-servo.din.MM bit out (MM=00..19)

pluto-servo.din.MM-not bit out (MM=00..19)

For M=0 through 7, din.*MM* corresponds to the pin labeled IN*M* on the pinout diagram. Other pins are shared with the encoder function, as follows:

Pin	Shared
Label	with
din.8	QZ0
din.9	QZ1
din.10	QZ2
din.11	QZ3
din.12	QB0
din.13	QB1
din.14	QB2
din.15	QB3
din.16	QA0

din.17	QA1
din.18	QA2
din.19	QA3

PARAMETERS**pluto-servo.encoder.M.scale** float rw (M=0..3) (default: *I*)**pluto-servo.encoder.z-polarity** bit rw

Set to TRUE if the index pulse is active low, FALSE if it is active high. Affects all encoders.

pluto-servo.pwm.M.offset float rw (M=0..3)**pluto-servo.pwm.M.scale** float rw (M=0..3) (default: *I*)**pluto-servo.pwm.M.max-dc** float rw (M=0..3) (default: *I*)**pluto-servo.pwm.M.min-dc** float rw (M=0..3) (default: 0)**pluto-servo.pwm.M.pwmdir** bit rw (M=0..3) (default: 0)

Set to TRUE use PWM+direction mode. Set to FALSE to use Up/Down mode.

pluto-servo.pwm.is-pdm bit rw

Set to TRUE to use PDM (also called interleaved PWM) mode. Set to FALSE to use traditional PWM mode. Affects all PWM outputs.

pluto-servo.dout.MM-invert bit rw (MM=00..19)If TRUE, the output on the corresponding **dout.MM** is inverted.**pluto-servo.communication-error** u32 rw

Incremented each time pluto-servo.read detects an error code in the EPP status register. While this register is nonzero, new values are not being written to the Pluto-P board, and the status of digital outputs and the PWM duty cycle of the PWM outputs will remain unchanged. If the watchdog is enabled, it will activate soon after the communication error is detected. To continue after a communication error, set this parameter back to zero.

pluto-servo.debug-0 s32 rw**pluto-servo.debug-1** s32 rw

These parameters can display values which are useful to developers or for debugging the driver and firmware. They are not useful for integrators or users.

SEE ALSOThe *pluto_servo* section in the HAL User Manual, which shows the location of each physical pin on the pluto board.**LICENSE**

GPL

NAME

pluto_step – Hardware driver and firmware for the Pluto-P parallel-port FPGA, for use with stepper machines.

SYNOPSIS

```
loadrt pluto_step ioaddr=addr ioaddr_hi=addr epp_wide=[0/1]
```

ioaddr [default: 0x378]

The base address of the parallel port.

ioaddr_hi [default: 0]

The secondary address of the parallel port, used to set EPP mode. 0 means to use ioaddr + 0x400. -1 means there is no secondary address.

epp_wide [default: 1]

Set to zero to disable "wide EPP mode". "Wide" mode allows 16- and 32-bit EPP transfers, which can reduce the time spent in the read and write functions. However, this mode may not work on all EPP parallel ports.

watchdog [default: 1]

Set to zero to disable the "hardware watchdog". "Watchdog" will tristate all outputs approximately 6ms after the last execution of **pluto-step.write**, which adds some protection in the case of LinuxCNC crashes.

speedrange [default: 0]

Selects one of four speed ranges:

0: Top speed 312.5kHz; minimum speed 610Hz

1: Top speed 156.25kHz; minimum speed 305Hz

2: Top speed 78.125kHz; minimum speed 153Hz

3: Top speed 39.06kHz; minimum speed 76Hz

Choosing the smallest maximum speed that is above the maximum for any one axis may give improved step regularity at low step speeds.

DESCRIPTION

Pluto_step is a LinuxCNC software driver and associated firmware that allow the Pluto-P board to be used to control a stepper-based CNC machine.

The driver has 4 step+direction channels, 14 dedicated digital outputs, and 16 dedicated digital inputs.

Step generators

The step generator takes a position input and output.

The step waveform includes step length/space and direction hold/setup time. Step length and direction set-up/hold time is enforced in the FPGA. Step space is enforced by a velocity cap in the driver.

(*all the following numbers are subject to change*) In *speedrange=0*, the maximum step rate is 312.5kHz. For position feedback to be accurate, the maximum step rate is 512 pulses per servo cycle (so a 1kHz servo cycle does not impose any additional limitation). The maximum step rate may be lowered by the step length and space parameters, which are rounded up to the nearest multiple of 1600ns.

In successive speedranges the maximum step rate is divided in half, as is the maximum steps per servo cycle, and the minimum nonzero step rate.

Digital I/O

The digital output pins conform to the ‘canonical digital output’ interface described in the HAL manual.

The digital input pins conform to the ‘canonical digital input’ interface described in the HAL manual.

FUNCTIONS

- pluto-step.read** (requires a floating-point thread)
 - Read all the inputs from the pluto-step board
- pluto-step.write** (requires a floating-point thread)
 - Write all the outputs on the pluto-step board

PINS

- pluto-step.stepgen.M.position-cmd** float in (M=0..3)
- pluto-step.stepgen.M.velocity-fb** float out (M=0..3)
- pluto-step.stepgen.M.position-fb** float out (M=0..3)
- pluto-step.stepgen.M.counts** s32 out (M=0..3)
- pluto-step.stepgen.M.enable** bit in (M=0..3)
- pluto-step.stepgen.M.reset** bit in (M=0..3)
 - When TRUE, reset position-fb to 0
- pluto-step.dout.MM** bit in (MM=00..13)
 - dout.MM corresponds to the pin labeled OUT M on the pinout diagram.
- pluto-step.din.MM** bit out (MM=00..15)
- pluto-step.din.MM-not** bit out (MM=00..15)
 - din.MM corresponds to the pin labeled IN M on the pinout diagram.

PARAMETERS

- pluto-step.stepgen.M.scale** float rw (M=0..3) (default: 1.0)
- pluto-step.stepgen.M.maxvel** float rw (M=0..3) (default: 0)
- pluto-step.stepgen.step-polarity** bit rw
- pluto-step.stepgen.steplen** u32 rw
 - Step length in ns.
- pluto-step.stepgen.stepspace** u32 rw
 - Step space in ns
- pluto-step.stepgen.dirtime** u32 rw
 - Dir hold/setup in ns. Refer to the pdf documentation for a diagram of what these timings mean.
- pluto-step.dout.MM-invert** bit rw (MM=00..13)
 - If TRUE, the output on the corresponding **dout.MM** is inverted.
- pluto-step.communication-error** u32 rw
 - Incremented each time pluto-step.read detects an error code in the EPP status register. While this register is nonzero, new values are not being written to the Pluto-P board, and the status of digital outputs and the PWM duty cycle of the PWM outputs will remain unchanged. If the hardware watchdog is enabled, it will activate shortly after the communication error is detected by Linux-CNC. To continue after a communication error, set this parameter back to zero.
- pluto-step.debug-0** s32 rw
- pluto-step.debug-1** s32 rw
- pluto-step.debug-2** float rw (default: .5)
- pluto-step.debug-3** float rw (default: 2.0)
 - Registers that hold debugging information of interest to developers

SEE ALSO

The *pluto_step* section in the HAL User Manual, which shows the location of each physical pin on the pluto board.

PLUTO_STEP(9)

HAL Component

PLUTO_STEP(9)

LICENSE

GPL

NAME

pwmgen – software PWM/PDM generation

SYNOPSIS

```
loadrt pwmgen output_type=type0[,type1...]
```

DESCRIPTION

pwmgen is used to generate PWM (pulse width modulation) or PDM (pulse density modulation) signals. The maximum PWM frequency and the resolution is quite limited compared to hardware-based approaches, but in many cases software PWM can be very useful. If better performance is needed, a hardware PWM generator is a better choice.

pwmgen supports a maximum of eight channels. The number of channels actually loaded depends on the number of *type* values given. The value of each *type* determines the outputs for that channel.

type 0: single output

A single output pin, **pwm**, whose duty cycle is determined by the input value for positive inputs, and which is off (or at **min-dc**) for negative inputs. Suitable for single ended circuits.

type 1: pwm/direction

Two output pins, **pwm** and **dir**. The duty cycle on **pwm** varies as a function of the input value. **dir** is low for positive inputs and high for negative inputs.

type 2: up/down

Two output pins, **up** and **down**. For positive inputs, the PWM/PDM waveform appears on **up**, while **down** is low. For negative inputs, the waveform appears on **down**, while **up** is low. Suitable for driving the two sides of an H-bridge to generate a bipolar output.

FUNCTIONS

pwmgen.make-pulses (no floating-point)

Generates the actual PWM waveforms, using information computed by **update**. Must be called as frequently as possible, to maximize the attainable PWM frequency and resolution, and minimize jitter. Operates on all channels at once.

pwmgen.update (uses floating point)

Accepts an input value, performs scaling and limit checks, and converts it into a form usable by **make-pulses** for PWM/PDM generation. Can (and should) be called less frequently than **make-pulses**. Operates on all channels at once.

PINS

pwmgen.N.enable bit in

Enables PWM generator *N* - when false, all **pwmgen.N** output pins are low.

pwmgen.N.value float in

Commanded value. When **value** = 0.0, duty cycle is 0%, and when **value** = +/-**scale**, duty cycle is +/- 100%. (Subject to **min-dc** and **max-dc** limitations.)

pwmgen.N.pwm bit out (output types 0 and 1 only)

PWM/PDM waveform.

pwmgen.N.dir bit out (output type 1 only)

Direction output: low for forward, high for reverse.

pwmgen.N.up bit out (output type 2 only)

PWM/PDM waveform for positive input values, low for negative inputs.

pwmgen.N.down bit out (output type 2 only)

PWM/PDM waveform for negative input values, low for positive inputs.

PARAMETERS

pwmgen.N.curr-dc float ro

The current duty cycle, after all scaling and limits have been applied. Range is from -1.0 to +1.0.

pwmgen.N.max-dc float rw

The maximum duty cycle. A value of 1.0 corresponds to 100%. This can be useful when using transistor drivers with bootstrapped power supplies, since the supply requires some low time to recharge.

pwmgen.N.min-dc float rw

The minimum duty cycle. A value of 1.0 corresponds to 100%. Note that when the pwm generator is disabled, the outputs are constantly low, regardless of the setting of **min-dc**.

pwmgen.N.scale float rw

pwmgen.N.offset float rw

These parameters provide a scale and offset from the **value** pin to the actual duty cycle. The duty cycle is calculated according to $dc = (value/scale) + offset$, with 1.0 meaning 100%.

pwmgen.N.pwm-freq float rw

PWM frequency in Hz. The upper limit is half of the frequency at which **make-pulses** is invoked, and values above that limit will be changed to the limit. If **dither-pwm** is false, the value will be changed to the nearest integer submultiple of the **make-pulses** frequency. A value of zero produces Pulse Density Modulation instead of Pulse Width Modulation.

pwmgen.N.dither-pwm bit rw

Because software-generated PWM uses a fairly slow timebase (several to many microseconds), it has limited resolution. For example, if **make-pulses** is called at a 20KHz rate, and **pwm-freq** is 2KHz, there are only 10 possible duty cycles. If **dither-pwm** is false, the commanded duty cycle will be rounded to the nearest of those values. Assuming **value** remains constant, the same output will repeat every PWM cycle. If **dither-pwm** is true, the output duty cycle will be dithered between the two closest values, so that the long-term average is closer to the desired level. **dither-pwm** has no effect if **pwm-freq** is zero (PDM mode), since PDM is an inherently dithered process.

NAME

sample_hold – Sample and Hold

SYNOPSIS

loadrt sample_hold [count=N|names=name1[,name2...]]

FUNCTIONS

sample-hold.N

PINS

sample-hold.N.in s32 in

sample-hold.N.hold bit in

sample-hold.N.out s32 out

LICENSE

GPL

NAME

sampler – sample data from HAL in real time

SYNOPSIS

loadrt sampler depth=depth1[,depth2...] **cfg=string1[,string2...]**

DESCRIPTION

sampler and **halsampler(1)** are used together to sample HAL data in real time and store it in a file. **sampler** is a realtime HAL component that exports HAL pins and creates a FIFO in shared memory. It then begins sampling data from the HAL and storing it to the FIFO. **halsampler** is a user space program that copies data from the FIFO to stdout, where it can be redirected to a file or piped to some other program.

OPTIONS

depth=depth1[,depth2...]

sets the depth of the realtime->user FIFO that **sampler** creates to buffer the realtime data. Multiple values of *depth* (separated by commas) can be specified if you need more than one FIFO (for example if you want to sample data from two different realtime threads).

cfg=string1[,string2...]

defines the set of HAL pins that **sampler** exports and later samples data from. One *string* must be supplied for each FIFO, separated by commas. **sampler** exports one pin for each character in *string*. Legal characters are:

F, f (float pin)

B, b (bit pin)

S, s (s32 pin)

U, u (u32 pin)

FUNCTIONS

sampler.N

One function is created per FIFO, numbered from zero.

PINS

sampler.N.pin.M input

Pin for the data that will wind up in column *M* of FIFO *N* (and in column *M* of the output file). The pin type depends on the config string.

sampler.N.curr-depth s32 output

Current number of samples in the FIFO. When this reaches *depth* new data will begin overwriting old data, and some samples will be lost.

sampler.N.full bit output

TRUE when the FIFO *N* is full, FALSE when there is room for another sample.

sampler.N.enable bit input

When TRUE, samples are captured and placed in FIFO *N*, when FALSE, no samples are acquired. Defaults to TRUE.

PARAMETERS

sampler.N.overruns s32 read/write

The number of times that **sampler** has tried to write data to the HAL pins but found no room in the FIFO. It increments whenever **full** is true, and can be reset by the **setup** command.

sampler.N.sample-num s32 read/write

A number that identifies the sample. It is automatically incremented for each sample, and can be reset using the **setp** command. The sample number can optionally be printed in the first column of the output from **halsampler**, using the **-t** option. (see **man 1 halsampler**)

SEE ALSO

halsampler(1) streamer(9) halstreamer(1)

HISTORY**BUGS****AUTHOR**

Original version by John Kasunich, as part of the LinuxCNC project. Improvements by several other members of the LinuxCNC development team.

REPORTING BUGS

Report bugs to jmkasunich AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2006 John Kasunich.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

scale – LinuxCNC HAL component that applies a scale and offset to its input

SYNOPSIS

loadrt scale [count=N|names=name1[,name2...]]

FUNCTIONS

scale.N (requires a floating-point thread)

PINS

scale.N.in float in
scale.N.gain float in
scale.N.offset float in
scale.N.out float out
out = in * gain + offset

LICENSE

GPL

NAME

select8 – 8-bit binary match detector

SYNOPSIS

loadrt select8 [count=N|names=name1[,name2...]]

FUNCTIONS

select8.N

PINS

select8.N.sel s32 in

The number of the output to set TRUE. All other outputs will be set FALSE

select8.N.outM bit out (M=0..7)

Output bits. If enable is set and the sel input is between 0 and 7, then the corresponding output bit will be set true

PARAMETERS

select8.N.enable bit rw (default: *TRUE*)

Set enable to FALSE to cause all outputs to be set FALSE

LICENSE

GPL

NAME

serport – Hardware driver for the digital I/O bits of the 8250 and 16550 serial port.

SYNOPSIS

loadrt serport io=addr[,addr...]

The pin numbers refer to the 9-pin serial pinout. Keep in mind that these ports generally use rs232 voltages, not 0/5V signals.

Specify the I/O address of the serial ports using the module parameter **io=addr[,addr...]**. These ports must not be in use by the kernel. To free up the I/O ports after bootup, install setserial and execute a command like:

```
sudo setserial /dev/ttys0 uart none
```

but it is best to ensure that the serial port is never used or configured by the Linux kernel by setting a kernel commandline parameter or not loading the serial kernel module if it is a modularized driver.

FUNCTIONS

serport.N.read

serport.N.write

PINS

serport.N.pin-1-in bit out

Also called DCD (data carrier detect); pin 8 on the 25-pin serial pinout

serport.N.pin-6-in bit out

Also called DSR (data set ready); pin 6 on the 25-pin serial pinout

serport.N.pin-8-in bit out

Also called CTS (clear to send); pin 5 on the 25-pin serial pinout

serport.N.pin-9-in bit out

Also called RI (ring indicator); pin 22 on the 25-pin serial pinout

serport.N.pin-1-in-not bit out

Inverted version of pin-1-in

serport.N.pin-6-in-not bit out

Inverted version of pin-6-in

serport.N.pin-8-in-not bit out

Inverted version of pin-8-in

serport.N.pin-9-in-not bit out

Inverted version of pin-9-in

serport.N.pin-3-out bit in

Also called TX (transmit data); pin 2 on the 25-pin serial pinout

serport.N.pin-4-out bit in

Also called DTR (data terminal ready); pin 20 on the 25-pin serial pinout

serport.N.pin-7-out bit in

Also called RTS (request to send); pin 4 on the 25-pin serial pinout

PARAMETERS

serport.N.pin-3-out-invert bit rw

serport.N.pin-4-out-invert bit rw

serport.N.pin-7-out-invert bit rw

SERPORT(9)

HAL Component

SERPORT(9)

serport.N.ioaddr u32 r

LICENSE
GPL

NAME

setserial - a utility for setting Smart Serial NVRAM parameters.

SYNOPSIS

```
loadrt setserial cmd="set hm2_8i20.001f.nvmaxcurrent 750"
```

FUNCTIONS

None

PINS

None

USAGE

```
loadrt setserial cmd="{command} {parameter/device} {value/filename}"
```

Commands available are **set** and **flash**.

This utility should be used under halcmd, without LinuxCNC running or any realtime threads running.

A typical command sequence would be:

```
halrun  
loadrt hostmot2 use_serial_numbers=1  
loadrt hm2_pci config="firmware=hm2/5i23/svss8_8.bit"  
show param  
loadrt setserial cmd="set hm2_8i20.001f.nvmaxcurrent 750"  
exit
```

This example uses the option to have the hal pins and parameters labelled by the serial number of the remote. This is not necessary but can reduce the scope for confusion. (The serial number is normally on a sticker on the device.)

The next line loads the hm2_pci driver in the normal way. The hm2_7i43 driver should work equally well, as should any future 7i80 driver. If the card has already been started up and a firmware has been loaded, then the config string may be omitted.

"show param" is optional, but provides a handy list of all the devices and parameters. It also shows the current values of the parameters which can be useful for determining scaling. u32 pin values are always shown in hex, but new values can be entered in decimal or hex. Use the Ox123ABC format to enter a hex value.

The next line invokes setserial. This is run in a slightly strange way in order to have kernel-level access to a live Hostmot2 config. It is basically a HAL module that always fails to load. This may lead to error messages being printed to the halcmd prompt. These can often be ignored. All the real feedback is via the dmesg command. It is suggested to have a second terminal window open to run dmesg after each command.

On exiting there will typically be a further error message related to the driver failing to unload setserial. This can be ignored.

The parameter changes will not show up until the drivers are reloaded. //TODO// Add a "get" command to avoid this problem.

Flashing Firmware To flash new firmware to an FPGA card such as the 5i25 or 5i20 the "mesaflash" utility should be used. Setserial is only useful for changing/updating the firmware on smart-serial remote such as the 8i20. The firmware should be placed somewhere in the /lib/firmware/hm2 tree, where the Linux firmware loading macros can find it.

The flashing routine operates in a realtime thread, and can only send prompts to the user through the kernel

log (dmesg). It is most convenient to open two terminal windows, one for command entry and one to monitor progress.

In the first terminal enter

```
tail -f /var/log/kern.log
```

This terminal will now display status information.

The second window will be used to enter the commands. It is important that LinuxCNC and/or HAL are not already loaded when the process is started. To flash new firmware it is necessary to move a jumper on the smart-serial remote drive and to switch smart-serial communication to a slower baudrate.

A typical command sequence is then

```
halrun  
loadrt hostmot2 sserial_baudrate=115200  
loadrt hm2_pci config="firmware=hm2/5i23/svss8_8.bit"  
loadrt setserial cmd="flash hm2_5i23.0.8i20.0.1 hm2/8i20/8i20T.BIN"  
exit
```

It is not necessary (or useful) to specify a config string in a system using the 5i25 or 6i25 cards.

Note that it is necessary to exit halrun and unload the realtime environment before flashing the next card (exit)

The correct sserial channel name to use can be seen in the dmesg output in the feedback terminal after the loadrt hm2_pci step of the sequence.

LICENSE

GPL

NAME

siggen – signal generator

SYNOPSIS

```
loadrt siggen [num_chan=num | names=name1[,name2...]]
```

DESCRIPTION

siggen is a signal generator that can be used for testing and other applications that need simple waveforms. It produces sine, cosine, triangle, sawtooth, and square waves of variable frequency, amplitude, and offset, which can be used as inputs to other HAL components.

siggen supports a maximum of sixteen channels. The number of channels actually loaded is set by the **num_chan** argument when the module is loaded. Alternatively, specify **names=** and unique names separated by commas.

The **num_chan=** and **names=** specifiers are mutually exclusive. If neither **num_chan=** nor **names=** are specified, the default value is one.

NAMING

The names for pins, parameters, and functions are prefixed as:

siggen.N. for $N=0,1,\dots,num-1$ when using **num_chan=num**

nameN. for $nameN=name1,name2,\dots$ when using **names=name1,name2,...**

The **siggen.N.** format is shown in the following descriptions.

FUNCTIONS

siggen.N.update (uses floating-point)

Updates output pins for signal generator N . Each time it is called it calculates a new sample. It should be called many times faster than the desired signal frequency, to avoid distortion and aliasing.

PINS

siggen.N.frequency float in

The output frequency for signal generator N , in Hertz. The default value is 1.0 Hertz.

siggen.N.amplitude float in

The output amplitude for signal generator N . If **offset** is zero, the outputs will swing from **-amplitude** to **+amplitude**. The default value is 1.00.

siggen.N.offset float in

The output offset for signal generator N . This value is added directly to the output signal. The default value is zero.

siggen.N.clock bit out

The clock output. Bit type clock signal output at the commanded frequency.

siggen.N.square float out

The square wave output. Positive while **triangle** and **cosine** are ramping upwards, and while **sine** is negative.

siggen.N.sine float out

The sine output. Lags **cosine** by 90 degrees.

siggen.*N*.cosine float out

The cosine output. Leads **sine** by 90 degrees.

siggen.*N*.triangle float out

The triangle wave output. Ramps up while **square** is positive, and down while **square** is negative. Reaches its positive and negative peaks at the same time as **cosine**.

siggen.*N*.sawtooth float out

The sawtooth output. Ramps upwards to its positive peak, then instantly drops to its negative peak and starts ramping again. The drop occurs when **triangle** and **cosine** are at their positive peaks, and coincides with the falling edge of **square**.

PARAMETERS

None

NAME

sim_encoder – simulated quadrature encoder

SYNOPSIS

```
loadrt sim_encoder [num_chan=num | names=name1[,name2...]]
```

DESCRIPTION

sim_encoder can generate quadrature signals as if from an encoder. It also generates an index pulse once per revolution. It is mostly used for testing and simulation, to replace hardware that may not be available. It has a limited maximum frequency, as do all software based pulse generators.

sim_encoder supports a maximum of eight channels. The number of channels actually loaded is set by the **num_chan=** argument when the module is loaded. Alternatively, specify **names=** and unique names separated by commas.

The **num_chan=** and **names=** specifiers are mutually exclusive. If neither **num_chan=** nor **names=** are specified, the default value is one.

FUNCTIONS

sim-encoder.make-pulses (no floating-point)

Generates the actual quadrature and index pulses. Must be called as frequently as possible, to maximize the count rate and minimize jitter. Operates on all channels at once.

sim-encoder.update-speed (uses floating-point)

Reads the **speed** command and other parameters and converts the data into a form that can be used by **make-pulses**. Changes take effect only when **update-speed** runs. Can (and should) be called less frequently than **make-pulses**. Operates on all channels at once.

NAMING

The names for pins and parameters are prefixed as:

sim-encoder.N. for N=0,1,...,num-1 when using **num_chan=num**

nameN. for nameN=name1,name2,... when using **names=name1,name2,...**

The **sim-encoder.N.** format is shown in the following descriptions.

PINS

sim-encoder.N.phase-A bit out

One of the quadrature outputs.

sim-encoder.N.phase-B bit out

The other quadrature output.

sim-encoder.N.phase-Z bit out

The index pulse.

sim-encoder.N.speed float in

The desired speed of the encoder, in user units per second. This is divided by **scale**, and the result is used as the encoder speed in revolutions per second.

PARAMETERS

sim-encoder.*N*.ppr u32 rw

The pulses per revolution of the simulated encoder. Note that this is pulses, not counts, per revolution. Each pulse or cycle from the encoder results in four counts, because every edge is counted. Default value is 100 ppr, or 400 counts per revolution.

sim-encoder.*N*.scale float rw

Scale factor for the **speed** input. The **speed** value is divided by **scale** to get the actual encoder speed in revolutions per second. For example, if **scale** is set to 60, then **speed** is in revolutions per minute (RPM) instead of revolutions per second. The default value is 1.00.

NAME

sim_spindle – Simulated spindle with index pulse

SYNOPSIS

loadrt sim_spindle [count=N|names=name1[,name2...]]

FUNCTIONS

sim-spindle.N (requires a floating-point thread)

PINS

sim-spindle.N.velocity-cmd float in
Commanded speed

sim-spindle.N.position-fb float out
Feedback position, in revolutions

sim-spindle.N.index-enable bit io
Reset **position-fb** to 0 at the next full rotation

PARAMETERS

sim-spindle.N.scale float rw (default: *1.0*)
factor applied to **velocity-cmd**.

The result of '**velocity-cmd * scale**' be in revolutions per second. For example, if **velocity-cmd** is in revolutions/minute, **scale** should be set to 1/60 or 0.016666667.

LICENSE

GPL

NAME

sphereprobe – Probe a pretend hemisphere

SYNOPSIS

loadrt sphereprobe [count=N|names=name1[,name2...]]

FUNCTIONS

sphereprobe.N

update probe-out based on inputs

PINS

sphereprobe.N.px s32 in

sphereprobe.N.py s32 in

sphereprobe.N.pz s32 in

rawcounts position from software encoder

sphereprobe.N.cx s32 in

sphereprobe.N.cy s32 in

sphereprobe.N.cz s32 in

Center of sphere in counts

sphereprobe.N.r s32 in

Radius of hemisphere in counts

sphereprobe.N.probe-out bit out

AUTHOR

Jeff Epler

LICENSE

GPL

NAME

hostmot2 - Smart Serial LinuxCNC HAL driver for the Mesa Electronics HostMot2 Smart-Serial remote cards

SYNOPSIS

The Mesa Smart-Serial interface is a 2.5Mbs proprietary interface between the Mesa Anything-IO cards and a range of subsidiary devices termed "smart-serial remotes". The remote cards perform a variety of functions, but typically they combine various classes of IO. The remot cards are self-configuring, in that they tell the main LinuxCNC Hostmot2 driver what their pin functions are and what they should be named.

Many sserial remotes offer different pinouts depending on the mode they are started up in. This is set using the sserial_port_N= option in the hm2_pci modparam. See the hostmot2 manpage for further details.

It is likely that this documentation will be permanently out of date.

Each Anything-IO board can attach up to 8 sserial remotes to each header (either the 5-pin headers on the 5i20/5i22/5i23/7i43 or the 25-pin connectors on the 5i25, 6i25 and 7i80). The remotes are grouped into "ports" of up to 8 "channels". Typically each header will be a single 8 channel port, but this is not necessarily always the case.

PORTS

In addition to the per-channel/device pins detailed below there are three per-port pins and three parameters.

Pins:

(bit, in) .sserial.port-N.run: Enables the specific Smart Serial module. Setting this pin low will disable all boards on the port and puts the port in a pass-through mode where device parameter setting is possible. This pin defaults to TRUE and can be left unconnected. However, toggling the pin low-to-high will re-enable a faulted drive so the pin could usefully be connected to the iocontrol.0.user-enable-out pin.

(u32, ro) .run_state: Shows the state of the sserial communications state-machine. This pin will generally show a value of 0x03 in normal operation, 0x07 in setup mode and 0x00 when the "run" pin is false.

(u32, ro) .error-count: Indicates the state of the Smart Serial error handler, see the parameters sections for more details.

Parameters:

(u32 r/w) .fault-inc: Any over-run or handshaking error in the SmartSerial communications will increment the .fault-count pin by the amount specified by this parameter. Default = 10.

(u32 r/w) .fault-dec: Every successful read/write cycle decrements the fault counter by this amount. Default = 1.

(u32 r/w) .fault-lim: When the fault counter reaches this threshold the Smart Serial interface on the corresponding port will be stopped and an error printed in dmesg. Together these three pins allow for control over the degree of fault-tolerance allowed in the interface. The default values mean that if more than one transaction in ten fails, more than 20 times, then a hard error will be raised. If the increment were to be set to zero then no error would ever be raised, and the system would carry on regardless. Conversely setting decrement to zero, threshold to 1 and limit to 1 means that absolutely no errors will be tolerated. (This structure is copied directly from vehicle ECU practice)

DEVICES

The other pins and parameters created in HAL depend on the devices detected. The following list of Smart Serial devices is by no means exhaustive.

- 8i20** The 8i20 is a 2.2kW three-phase drive for brushless DC motors and AC servo motors. 8i20 pins and parameters have names like "hm2_<BoardType>.<BoardNum>.8i20.<PortNum>.<Chan-Num>.<Pin>", for example "hm2_5i23.0.8i20.1.3.current" would set the phase current for the drive connected to the fourth channel of the second sserial port of the first 5i23 board. Note that the sserial ports do not necessarily correlate in layout or number to the physical ports on the card.

Pins:

(float in) angle

The rotor angle of the motor in fractions of a full **phase** revolution. An angle of 0.5 indicates that the motor is half a turn / 180 degrees / π radians from the zero position. The zero position is taken to be the position that the motor adopts under no load with a poitive voltage applied to the A (or U) phase and both B and C (or V and W) connected to -V or 0V. A 6 pole motor will have 3 zero positions per physical rotation. Note that the 8i20 drive automatically adds the phase lead/lag angle, and that this pin should see the raw rotor angle. There is a HAL module (bldc) which handles the complexity of differing motor and drive types.

(float, in) current

The phase current command to the drive. This is scaled from -1 to +1 for forwards and reverse maximum currents. The absolute value of the current is set by the max_current parameter.

(float, ro) voltage

The drive bus voltage in V. This will tend to show 25.6V when the drive is unpowered and the drive will not operate below about 50V.

(float, ro) temp

The temperature of the driver in degrees C.

(u32, ro) comms

The communication status of the drive. See the manual for more details.

(bit, ro) status and fault.

The following fault/status bits are exported. For further details see the 8i20 manual. fault.U-current / fault.U-current-not fault.V-current / fault.V-current-not fault.W-current / fault.W-current-not fault.bus-high / fault.bus-high-not fault.bus-overv / fault.bus-overv-not fault.bus-underv / fault.bus-underv-not fault.framingr / fault.framingr-not fault.module / fault.module-not fault.no-enable / fault.no-enable-not fault.overcurrent / fault.overcurrent-not fault.overrun / fault.overrun-not fault.overtemp / fault.overtemp-not fault.watchdog / fault.watchdog-not

status.brake-old / status.brake-old-not status.brake-on / status.brake-on-not status.bus-underv / status.bus-underv-not status.current-lim / status.current-lim-no status.ext-reset / status.ext-reset-not status.no-enable / status.no-enable-not status.pid-on / status.pid-on-not status.sw-reset / status.sw-reset-not status.wd-reset / status.wd-reset-not

Parameters:

The following parameters are exported. See the pdf documentation downloadable from Mesa for further details

```

hm2_5i25.0.8i20.0.1.angle-maxlim
hm2_5i25.0.8i20.0.1.angle-minlim
hm2_5i25.0.8i20.0.1.angle-scalemax
hm2_5i25.0.8i20.0.1.current-maxlim
hm2_5i25.0.8i20.0.1.current-minlim
hm2_5i25.0.8i20.0.1.current-scalemax
hm2_5i25.0.8i20.0.1.nvbrakeoffv
hm2_5i25.0.8i20.0.1.nvbrakeonv
hm2_5i25.0.8i20.0.1.nvbusoverv
hm2_5i25.0.8i20.0.1.nvbusundervmax
hm2_5i25.0.8i20.0.1.nvbusundervmin
hm2_5i25.0.8i20.0.1.nvkdihi
hm2_5i25.0.8i20.0.1.nvkdl
hm2_5i25.0.8i20.0.1.nvkdiло
hm2_5i25.0.8i20.0.1.nvkdp
hm2_5i25.0.8i20.0.1.nvkqihi
hm2_5i25.0.8i20.0.1.nvkqil
hm2_5i25.0.8i20.0.1.nvkqilo
hm2_5i25.0.8i20.0.1.nvkqp
hm2_5i25.0.8i20.0.1.nvmaxcurrent
hm2_5i25.0.8i20.0.1.nvrembaudrate
hm2_5i25.0.8i20.0.1.swrevision
hm2_5i25.0.8i20.0.1.unitnumber

```

(float, rw) max_current

Sets the maximum drive current in Amps. The default value is the maximum current programmed into the drive EEPROM. The value must be positive, and an error will be raised if a current in excess of the drive maximum is requested.

(u32, ro) serial_number

The serial number of the connected drive. This is also shown on the label on the drive.

7i64 The 7i64 is a 24-input 24-output IO card. 7i64 pins and parameters have names like "hm2_<BoardType>.<BoardNum>.7i64. <PortNum>.<ChanNum>.<Pin>", for example hm2_5i23.0.7i64.1.3.output-01

Pins: (bit, in) 7i64.0.0.output-NN: Writing a 1 or TRUE to this pin will enable output driver NN. Note that the outputs are drivers (switches) rather than voltage outputs. The LED adjacent to the connector on the board shows the status. The output can be inverted by setting a parameter.

(bit, out) 7i64.0.0.input-NN: The value of input NN. Note that the inputs are isolated and both pins of each input must be connected (typically to signal and the ground of the signal. This need not be the ground of the board.)

(bit, out) 7i64.0.0.input-NN-not: An inverted copy of the corresponding input.

(float, out) 7i64.0.0.analog0 & 7i64.0.0.analog1: The two analogue inputs (0 to 3.3V) on the board.

Parameters: (bit, rw) 7i64.0.0.output-NN-invert: Setting this parameter to 1 / TRUE will invert the output value, such that writing 0 to .gpio.NN.out will enable the output and vice-versa.

7i76 The 7i76 is not only a smart-serial device. It also serves as a breakout for a number of other Hostmot2 functions. There are connections for 5 step generators (for which see the main hostmot2 manpage). The stepgen pins are associated with the 5i25 (hm2_5i25.0.stepgen.00....) whereas the smart-serial pins are associated with the 7i76 (hm2_5i25.0.7i76.0.0.output-00).

Pins:

(float out) .7i76.0.0.analogN (modes 1 and 2 only) Analogue input values.

(float out) .7i76.0.0.fieldvoltage (mode 2 only) Field voltage monitoring pin.

(bit in) .7i76.0.0.spindir: This pin provides a means to drive the spindle VFD direction terminals on the 7i76 board.

(bit in) .7i76.0.0.spinena: This pin drives the spindle-enable terminals on the 7i76 board.

(float in) .7i76.0.0.spinout: This controls the analogue output of the 7i76. This is intended as a speed control signal for a VFD.

(bit out) .7i76.0.0.output-NN: (NN = 0 to 15). 16 digital outputs. The sense of the signal can be set via a parameter

(bit out) .7i76.0.0.input-NN: (NN = 0 to 31) 32 digital inputs.

(bit in) .7i76.0.0.input-NN-not: (NN = 0 to 31) An inverted copy of the inputs provided for convenience. The two complementary pins may be connected to different signal nets.

Parameters:

(u32 ro) .7i76.0.0.nvbaudrate: Indicates the vbaud rate. This probably should not be altered, and special utils are needed to do so.

(u32 ro) .7i76.0.0.nvunitnumber: Indicates the serial number of the device and should match a siticker on the card. This can be useful for wokring out which card is which.

(u32 ro) .7i76.0.0.nvwatchdogtimeout: The sserial remote watchdog timeout. This is separate from the Anything-IO card timeout. This is unlikley to need to be changed.

(bit rw) .7i76.0.0.output-NN-invert: Invert the sense of the corresponding output pin.

(bit rw) .7i76.0.0.spindir-invert: Invert the senseof the spindle direction pin.

(bit rw) .7i76.0.0.spinena-invert: Invert the sense of the spindle-enable pin.

(float rw) .7i76.0.0.spinout-maxlim: The maximum speed request allowable

(float rw) .7i76.0.0.spinout-minlim: The minimum speed request.

(float rw) .7i76.0.0.spinout-scalemax: The spindle speed scaling. This is the speed request which would correspond to full-scale output from the spindle control pin. For example with a 10V drive voltage and a 10000rpm scalemax a value of 10,000 rpm on the spinout pin would produce 10V output. However, if spinout-maxlim were set to 5,000 rpm then no voltage above 5V would be output.

(u32 ro) .7i76.0.0.swrevision: The onboard firmware revision number. Utilities exist to update and change this firmware.

- 7i77** The 7i77 is an 6-axis servo control card. The analogue outputs are smart-serial devices but the encoders are conventional hostmot2 encoders and further details of them may be found in the hostmot2 manpage.

Pins: (bit out) .7i77.0.0.input-NN: (NN = 0 to 31) 32 digital inputs.

(bit in) .7i77.0.0.input-NN-not: (NN = 0 to 31) An inverted copy of the inputs provided for convenience. The two complementary pins may be connected to different signal nets.

(bit out) .7i77.0.0.output-NN: (NN = 0 to 15). 16 digital outputs. The sense of the signal can be set via a parameter

(bit in) .7i77.0.0.spindir: This pin provides a means to drive the spindle VFD direction terminals on the 7i76 board.

(bit in) .7i77.0.0.spinena: This pin drives the spindle-enable terminals on the 7i76 board.

(float in) .7i77.0.0.spinout: This controls the analog output of the 7i77. This is intended as a speed control signal for a VFD.

(bit in) .7i77.0.1.analogena: This pin drives the analog enable terminals on the 7i77 board.

(float in) .7i77.0.1.analogoutN: (N = 0 to 5) This controls the analog output of the 7i77.

Parameters: (bit rw) .7i77.0.0.output-NN-invert: Invert the sense of the corresponding output pin.

(bit rw) .7i77.0.0.spindir-invert: Invert the senseof the spindle direction pin.

(bit rw) .7i77.0.0.spinena-invert: Invert the sense of the spindle-enable pin.

(float rw) .7i77.0.0.spinout-maxlim: The maximum speed request allowable

(float rw) .7i77.0.0.spinout-minlim: The minimum speed request.

(float rw) .7i77.0.0.spinout-scalemax: The spindle speed scaling. This is the speed request which would correspond to full-scale output from the spindle control pin. For example with a 10V drive voltage and a 10000rpm scalemax a value of 10,000 rpm on the spinout pin would produce 10V output. However, if spinout-maxlim were set to 5,000 rpm then no voltage above 5V would be output.

(float rw) .7i77.0.0.analogoutN-maxlim: (N = 0 to 5) The maximum speed request allowable

(float rw) .7i77.0.0.analogoutN-minlim: (N = 0 to 5) The minimum speed request.

//// ***** CHECK ME ***** I'm not sure about the description on analogoutN-scalemax ////

(float rw) .7i77.0.0.analogoutN-scalemax: (N = 0 to 5) The analog speed scaling. This is the speed request which would correspond to full-scale output from the spindle control pin. For example with a 10V drive voltage and a 10000rpm scalemax a value of 10,000 rpm on the spinout pin would produce 10V output. However, if spinout-maxlim were set to 5,000 rpm then no voltage above 5V would be output.

- 7i69** The 7i69 is a 48 channel digital IO card. It can be configured in four different modes: Mode 0 B 48 pins bidirectional (all outputs can be set high then driven low to work as inputs)
 Mode 1 48 pins, input only
 Mode 2 48 pins, all outputs
 Mode 3 24 inputs and 24 outputs.

Pins: (bit in) .7i69.0.0.output-NN: Digital output. Sense can be inverted with the corresponding Parameter

(bit out) .7i69.0.0.input-NN: Digital input

(bit out) .7i69.0.0.input-NN-not: Digital input, inverted.

Parameters:

(u32 ro) .7i69.0.0.nvbaudrate: Indicates the vbaud rate. This probably should not be altered, and special utils are needed to do so.

(u32 ro) .7i69.0.0.nvunitnumber: Indicates the serial number of the device and should match a siticker on the card. This can be useful for wokring out which card is which.

(u32 ro) .7i69.0.0.nvwatchdogtimeout: The sserial remote watchdog timeout. This is separate from the Anything-IO card timeout. This is unlikley to need to be changed.

(bit rw) .7i69.0.0.output-NN-invert: Invert the sense of the corresponding output pin.

(u32 ro) .7i69.0.0.swrevision: The onboard firmware revision number. Utilities exist to update and change this firmware.

7i70

The 7I70 is a remote isolated 48 input card. The 7I70 inputs sense positive inputs relative to a common field ground. Input impedance is 10K Ohms and input voltage can range from 5VDC to 32VDC. All inputs have LED status indicators. The input common field ground is galvanically isolated from the communications link.

The 7I70 has three software selectable modes. These different modes select different sets of 7I70 data to be transferred between the host and the 7I70 during real time process data exchanges. For high speed applications, choosing the correct mode can reduced the data transfer sizes, resulting in higher maximum update rates.

MODE 0 Input mode (48 bits input data only)

MODE 1 Input plus analog mode (48 bits input data plus 6 channels of analog data)

MODE 2 Input plus field voltage

Pins:

(float out) .7i70.0.0.analogN (modes 1 and 2 only) Analogue input values.

(float out) .7i70.0.0.fieldvoltage (mode 2 only) Field voltage monitoring pin.

(bit out) .7i70.0.0.input-NN: (NN = 0 to 47) 48 digital inputs.

(bit in) .7i70.0.0.input-NN-not: (NN = 0 to 47) An inverted copy of the inputs provided for

convenience. The two complementary pins may be connected to different signal nets.

Parameters:

(u32 ro) .7i70.0.0.nvbaudrate: Indicates the vbaud rate. This probably should not be altered, and special utils are needed to do so.

(u32 ro) .7i70.0.0.nvunitnumber: Indicates the serial number of the device and should match a siticker on the card. This can be useful for wokring out which card is which.

(u32 ro) .7i70.0.0.nvwatchdogtimeout: The sserial remote watchdog timeout. This is separate from the Anything-IO card timeout. This is unlikley to need to be changed.

(u32 ro) .7i69.0.0.swrevision: The onboard firmware revision number. Utilities exist to update and change this firmware.

7i71

The 7I71 is a remote isolated 48 output card. The 48 outputs are 8VDC to 28VDC sourcing drivers (common + field power) with 300 mA maximum current capability. All outputs have LED status indicators.

The 7I71 has two software selectable modes. For high speed applications, choosing the correct mode can reduced the data transfer sizes, resulting in higher maximum update rates

MODE 0 Output only mode (48 bits output data only)

MODE 1 Outputs plus read back field voltage

Pins:

(float out) .7i71.0.0.fieldvoltage (mode 2 only) Field voltage monitoring pin.

(bit out) .7i71.0.0.output-NN: (NN = 0 to 47) 48 digital outputs. The sense may be inverted by the invert parameter.

Parameters:

(bit rw) .7i71.0.0.output-NN-invert: Invert the sense of the corresponding output pin.

(u32 ro) .7i71.0.0.nvbaudrate: Indicates the vbaud rate. This probably should not be altered, and special utils are needed to do so.

(u32 ro) .7i71.0.0.nvunitnumber: Indicates the serial number of the device and should match a siticker on the card. This can be useful for wokring out which card is which.

(u32 ro) .7i71.0.0.nvwatchdogtimeout: The sserial remote watchdog timeout. This is separate from the Anything-IO card timeout. This is unlikley to need to be changed.

(u32 ro) .7i69.0.0.swrevision: The onboard firmware revision number. Utilities exist to update and change this firmware.

7i73 The 7I73 is a remote real time pendant or control panel interface.

The 7I73 supports up to four 50KHz encoder inputs for MPGs, 8 digital inputs and 6 digital outputs and up to a 64 Key keypad. If a smaller keypad is used, more digital inputs and outputs become available. Up to eight 0.0V to 3.3V analog inputs are also provided. The 7I73 can drive a 4 line 20 character LCD for local DRO applications.

The 7I73 has 3 software selectable process data modes. These different modes select different sets of 7I73 data to be transferred between the host and the 7 I73 during real time process data exchanges. For high speed applications, choosing the correct mode can reduce the data transfer sizes, resulting in higher maximum update rates

MODE 0 I/O + ENCODER

MODE 1 I/O + ENCODER +ANALOG IN

MODE 2 I/O + ENCODER +ANALOG IN FAST DISPLAY

Pins:

(float out) .7i73.0.0.analoginN: Analogue inputs. Up to 8 channels may be available dependant on software and hardware configuration modes. (see the pdf manual downloadable from www.mesonet.com)

(u32 in) .7i73.0.1.display (modes 1 and 2). Data for LCD display. This pin may be conveniently driven by the HAL "lcd" component which allows the formatted display of the values any number of HAL pins and textual content.

(u32 in) .7i73.0.1.display32 (mode 2 only). 4 bytes of data for LCD display. This mode is not supported by the HAL "lcd" component.

(s32 out) .7i73.0.1.encN: The position of the MPG encoder counters.

(bit out) .7i73.0.1.input-NN: Up to 24 digital inputs (dependent on config)

(bit out) .7i73.0.1.input-NN-not: Inverted copy of the digital inputs

(bit in) .7i73.0.1.output-NN: Up to 22 digital outputs (dependent on config)

Parameters:

(u32 ro) .7i73.0.1.nvanalogfilter:
 (u32 ro) .7i73.0.1.nvbaudrate
 (u32 ro) .7i73.0.1.nvcontrast
 (u32 ro) .7i73.0.1.nvdispmode
 (u32 ro) .7i73.0.1.nvencmode0
 (u32 ro) .7i73.0.1.nvencmode1
 (u32 ro) .7i73.0.1.nvencmode2
 (u32 ro) .7i73.0.1.nvencmode3
 (u32 ro) .7i73.0.1.nvkeytimer
 (u32 ro) .7i73.0.1.nvunitnumber
 (u32 ro) .7i73.0.1.nvwatchdogtimeout
 (u32 ro) .7i73.0.1.output-00-invert

The above parameters are only settable with utility software, for further details of their use see the Mesa manual.

(bit rw) .7i73.0.1.output-01-invert: Invert the corresponding output bit.

(s32 ro) .7i73.0.1.swrevision: The version of firmware installed.

TODO: Add 7i77, 7i66, 7i72, 7i83, 7i84, 7i87.

NAME

stepgen – software step pulse generation

SYNOPSIS

```
loadrt stepgen step_type=type0[,type1...] [ctrl_type=type0[,type1...]] [user_step_type=#,#...]
```

DESCRIPTION

stepgen is used to control stepper motors. The maximum step rate depends on the CPU and other factors, and is usually in the range of 5KHz to 25KHz. If higher rates are needed, a hardware step generator is a better choice.

stepgen has two control modes, which can be selected on a channel by channel basis using **ctrl_type**. Possible values are "p" for position control, and "v" for velocity control. The default is position control, which drives the motor to a commanded position, subject to acceleration and velocity limits. Velocity control drives the motor at a commanded speed, again subject to accel and velocity limits. Usually, position mode is used for machine axes. Velocity mode is reserved for unusual applications where continuous movement at some speed is desired, instead of movement to a specific position. (Note that velocity mode replaces the former component **freqgen**.)

stepgen can control a maximum of 16 motors. The number of motors/channels actually loaded depends on the number of *type* values given. The value of each *type* determines the outputs for that channel. Position or velocity mode can be individually selected for each channel. Both control modes support the same 16 possible step types.

By far the most common step type is '0', standard step and direction. Others include up/down, quadrature, and a wide variety of three, four, and five phase patterns that can be used to directly control some types of motor windings. (When used with appropriate buffers of course.)

Some of the stepping types are described below, but for more details (including timing diagrams) see the **stepgen** section of the HAL reference manual.

type 0: step/dir

Two pins, one for step and one for direction. **make-pulses** must run at least twice for each step (once to set the step pin true, once to clear it). This limits the maximum step rate to half (or less) of the rate that can be reached by types 2-14. The parameters **steplen** and **stepspace** can further lower the maximum step rate. Parameters **dirsetup** and **dirhold** also apply to this step type.

type 1: up/down

Two pins, one for 'step up' and one for 'step down'. Like type 0, **make-pulses** must run twice per step, which limits the maximum speed.

type 2: quadrature

Two pins, phase-A and phase-B. For forward motion, A leads B. Can advance by one step every time **make-pulses** runs.

type 3: three phase, full step

Three pins, phase-A, phase-B, and phase-C. Three steps per full cycle, then repeats. Only one phase is high at a time - for forward motion the pattern is A, then B, then C, then A again.

type 4: three phase, half step

Three pins, phases A through C. Six steps per full cycle. First A is high alone, then A and B together, then B alone, then B and C together, etc.

types 5 through 8: four phase, full step

Four pins, phases A through D. Four steps per full cycle. Types 5 and 6 are suitable for use with unipolar steppers, where power is applied to the center tap of each winding, and four open-collector transistors drive the ends. Types 7 and 8 are suitable for bipolar steppers, driven by two H-bridges.

types 9 and 10: four phase, half step

Four pins, phases A through D. Eight steps per full cycle. Type 9 is suitable for unipolar drive, and type 10 for bipolar drive.

types 11 and 12: five phase, full step

Five pins, phases A through E. Five steps per full cycle. See HAL reference manual for the patterns.

types 13 and 14: five phase, half step

Five pins, phases A through E. Ten steps per full cycle. See HAL reference manual for the patterns.

type 15: user-specified

This uses the waveform specified by the **user_step_type** module parameter, which may have up to 10 steps and 5 phases.

FUNCTIONS

stepgen.make-pulses (no floating-point)

Generates the step pulses, using information computed by **update-freq**. Must be called as frequently as possible, to maximize the attainable step rate and minimize jitter. Operates on all channels at once.

stepgen.capture-position (uses floating point)

Captures position feedback value from the high speed code and makes it available on a pin for use elsewhere in the system. Operates on all channels at once.

stepgen.update-freq (uses floating point)

Accepts a velocity or position command and converts it into a form usable by **make-pulses** for step generation. Operates on all channels at once.

PINS

stepgen.N.counts s32 out

The current position, in counts, for channel N. Updated by **capture-position**.

stepgen.N.position-fb float out

The current position, in length units (see parameter **position-scale**). Updated by **capture-position**. The resolution of **position-fb** is much finer than a single step. If you need to see individual steps, use **counts**.

stepgen.N.enable bit in

Enables output steps - when false, no steps are generated.

stepgen.N.velocity-cmd float in (velocity mode only)

Commanded velocity, in length units per second (see parameter **position-scale**).

stepgen.N.position-cmd float in (position mode only)

Commanded position, in length units (see parameter **position-scale**).

stepgen.N.step bit out (step type 0 only)

Step pulse output.

stepgen.N.dir bit out (step type 0 only)

Direction output: low for forward, high for reverse.

stepgen.N.up bit out (step type 1 only)

Count up output, pulses for forward steps.

stepgen.N.down bit out (step type 1 only)

Count down output, pulses for reverse steps.

stepgen.N.phase-A thru **phase-E** bit out (step types 2-14 only)

Output bits. **phase-A** and **phase-B** are present for step types 2-14, **phase-C** for types 3-14, **phase-D** for types 5-14, and **phase-E** for types 11-14. Behavior depends on selected stepping type.

PARAMETERS**stepgen.N.frequency** float roThe current step rate, in steps per second, for channel *N*.**stepgen.N.maxaccel** float rw

The acceleration/deceleration limit, in length units per second squared.

stepgen.N.maxvel float rwThe maximum allowable velocity, in length units per second. If the requested maximum velocity cannot be reached with the current combination of scaling and **make-pulses** thread period, it will be reset to the highest attainable value.**stepgen.N.position-scale** float rw

The scaling for position feedback, position command, and velocity command, in steps per length unit.

stepgen.N.rawcounts s32 roThe position in counts, as updated by **make-pulses**. (Note: this is updated more frequently than the **counts** pin.)**stepgen.N.steplen** u32 rw

The length of the step pulses, in nanoseconds. Measured from rising edge to falling edge.

stepgen.N.stepspace u32 rw (step types 0 and 1 only) The minimumspace between step pulses, in nanoseconds. Measured from falling edge to rising edge. The actual time depends on the step rate and can be much longer. If **stepspace** is 0, then **step** can be asserted every period. This can be used in conjunction with **hal_parport**'s auto-resetting pins to output one step pulse per period. In this mode, **steplen** must be set for one period or less.**stepgen.N.dirsetup** u32 rw (step type 0 only)

The minimum setup time from direction to step, in nanoseconds periods. Measured from change of direction to rising edge of step.

stepgen.N.dirhold u32 rw (step type 0 only)

The minimum hold time of direction after step, in nanoseconds. Measured from falling edge of step to change of direction.

stepgen.N.dirdelay u32 rw (step types 1 and higher only)

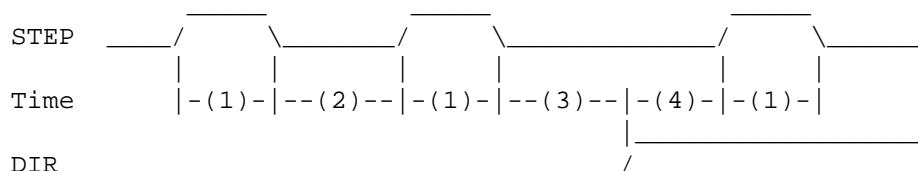
The minimum time between a forward step and a reverse step, in nanoseconds.

TIMING

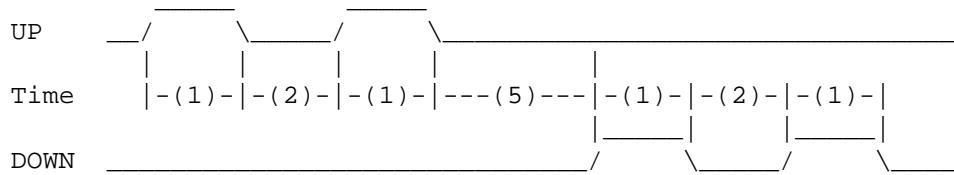
There are five timing parameters which control the output waveform. No step type uses all five, and only those which will be used are exported to HAL. The values of these parameters are in nano-seconds, so no recalculation is needed when changing thread periods. In the timing diagrams that follow, they are identified by the following numbers:

- (1) **stepgen.n.steplen**
- (2) **stepgen.n.stepspace**
- (3) **stepgen.n.dirhold**
- (4) **stepgen.n.dirsetup**
- (5) **stepgen.n.dirdelay**

For step type 0, timing parameters 1 thru 4 are used. The following timing diagram shows the output waveforms, and what each parameter adjusts.



For step type 1, timing parameters 1, 2, and 5 are used. The following timing diagram shows the output waveforms, and what each parameter adjusts.



For step types 2 and higher, the exact pattern of the outputs depends on the step type (see the HAL manual for a full listing). The outputs change from one state to another at a minimum interval of **steplen**. When a direction change occurs, the minimum time between the last step in one direction and the first in the other direction is the sum of **steplen** and **dirdelay**.

SEE ALSO

The HAL User Manual.

NAME

steptest – Used by Stepconf to allow testing of acceleration and velocity values for an axis.

SYNOPSIS

loadrt steptest [count=N|names=name1[,name2...]]

FUNCTIONS

steptest.N (requires a floating-point thread)

PINS

steptest.N.jog-minus bit in

Drive TRUE to jog the axis in its minus direction

steptest.N.jog-plus bit in

Drive TRUE to jog the axis in its positive direction

steptest.N.run bit in

Drive TRUE to run the axis near its current position_fb with a trapezoidal velocity profile

steptest.N.maxvel float in

Maximum velocity

steptest.N.maxaccel float in

Permitted Acceleration

steptest.N.amplitude float in

Approximate amplitude of positions to command during 'run'

steptest.N.dir s32 in

Direction from central point to test: 0 = both, 1 = positive, 2 = negative

steptest.N.position-cmd float out

steptest.N.position-fb float in

steptest.N.running bit out

steptest.N.run-target float out

steptest.N.run-start float out

steptest.N.run-low float out

steptest.N.run-high float out

steptest.N.pause s32 in (default: 0)

pause time for each end of run in seconds

PARAMETERS

steptest.N.epsilon float rw (default: .001)

steptest.N.elapsed float r

Current value of the internal timer

LICENSE

GPL

NAME

streamer – stream file data into HAL in real time

SYNOPSIS

loadrt streamer depth=depth1[,depth2...] **cfg=string1[,string2...]**

DESCRIPTION

streamer and **halstreamer(1)** are used together to stream data from a file into the HAL in real time. **streamer** is a realtime HAL component that exports HAL pins and creates a FIFO in shared memory. **hal_streamer** is a user space program that copies data from stdin into the FIFO, so that **streamer** can write it to the HAL pins.

OPTIONS

depth=depth1[,depth2...]

sets the depth of the user->realtime FIFO that **streamer** creates to receive data from **halstreamer**. Multiple values of *depth* (separated by commas) can be specified if you need more than one FIFO (for example if you want to stream data from two different realtime threads).

cfg=string1[,string2...]

defines the set of HAL pins that **streamer** exports and later writes data to. One *string* must be supplied for each FIFO, separated by commas. **streamer** exports one pin for each character in *string*. Legal characters are:

F, f (float pin)

B, b (bit pin)

S, s (s32 pin)

U, u (u32 pin)

FUNCTIONS

streamer.N

One function is created per FIFO, numbered from zero.

PINS

streamer.N.pin.M output

Data from column *M* of the data in FIFO *N* appears on this pin. The pin type depends on the config string.

streamer.N.curr-depth s32 output

Current number of samples in the FIFO. When this reaches zero, new data will no longer be written to the pins.

streamer.N.empty bit output

TRUE when the FIFO *N* is empty, FALSE when valid data is available.

streamer.N.enable bit input

When TRUE, data from FIFO *N* is written to the HAL pins. When false, no data is transferred. Defaults to TRUE.

streamer.N.underruns s32 read/write

The number of times that **sampler** has tried to write data to the HAL pins but found no fresh data in the FIFO. It increments whenever **empty** is true, and can be reset by the **setp** command.

SEE ALSO

halstreamer(1) **sampler(9)** **halsampler(1)**

HISTORY**BUGS**

Should an **enable** HAL pin be added, to allow streaming to be turned on and off?

AUTHOR

Original version by John Kasunich, as part of the LinuxCNC project. Improvements by several other members of the LinuxCNC development team.

REPORTING BUGS

Report bugs to jmkasunich AT users DOT sourceforge DOT net

COPYRIGHT

Copyright © 2006 John Kasunich.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

NAME

sum2 – Sum of two inputs (each with a gain) and an offset

SYNOPSIS

loadrt sum2 [count=N|names=name1[,name2...]]

FUNCTIONS

sum2.N (requires a floating-point thread)

PINS

sum2.N.in0 float in

sum2.N.in1 float in

sum2.N.out float out

out = in0 * gain0 + in1 * gain1 + offset

PARAMETERS

sum2.N.gain0 float rw (default: 1.0)

sum2.N.gain1 float rw (default: 1.0)

sum2.N.offset float rw

LICENSE

GPL

NAME

supply – set output pins with values from parameters (obsolete)

SYNOPSIS

```
loadrt supply num_chan=num
```

DESCRIPTION

supply was used to allow the inputs of other HAL components to be manipulated for testing purposes. When it was written, the only way to set the value of an input pin was to connect it to a signal and connect that signal to an output pin of some other component, and then let that component write the pin value. **supply** was written to be that "other component". It reads values from parameters (set with the HAL command **setp**) and writes them to output pins.

Since **supply** was written, the **setp** command has been modified to allow it to set unconnected pins as well as parameters. In addition, the **sets** command was added, which can directly set HAL signals, as long as there are no output pins connected to them. Therefore, **supply** is obsolete.

supply supports a maximum of eight channels. The number of channels actually loaded is set by the **num_chan** argument when the module is loaded. If **numchan** is not specified, the default value is one.

FUNCTIONS

supply.N.update (uses floating-point)

Updates output pins for channel *N*.

PINS

supply.N.q bit out

Output bit, copied from parameter **supply.N.d**.

supply.N._q bit out

Output bit, inverted copy of parameter **supply.N.d**.

supply.N.variable float out

Analog output, copied from parameter **supply.N.value**.

supply.N._variable float out

Analog output, equal to -1.0 times parameter **supply.N.value**.

supply.N.d bit rw

Data source for **q** and **_q** output pins.

supply.N.value bit rw

Data source for **variable** and **_variable** output pins.

NAME

thc – Torch Height Control

SYNOPSIS

loadrt thc

DESCRIPTION

Torch Height Control Mesa THC > Encoder > LinuxCNC THC component

The Mesa THC sends a frequency based on the voltage detected to the encoder. The velocity from the encoder is converted to volts with the velocity scale parameter inside the THC component.

The THCAD card sends a frequency at 0 volts so the scale offset parameter is used to zero the calculated voltage.

Component Functions If enabled and torch is on and X + Y velocity is within tolerance of set speed allow the THC to offset the Z axis as needed to maintain voltage.

If enabled and torch is off and the Z axis is moving up remove any correction at a rate not to exceed the rate of movement of the Z axis.

If enabled and torch is off and there is no correction pass the Z position and feed back untouched.

If not enabled pass the Z position and feed back untouched.

Physical Connections

Plasma Torch Arc Voltage Signal => 6 x 487k 1% resistors => THC Arc Voltage In

THC Frequency Signal => Encoder #0, pin A (Input)

Plasma Torch Arc OK Signal => input pin

output pin => Plasma Torch Start Arc Contacts

HAL Plasma Connections

encoder.nn.velocity => thc.encoder-vel (tip voltage)

motion.spindle-on => output pin (start the arc)

thc.arc-ok <= motion.digital-in-00 <= input pin (arc ok signal)

HAL Motion Connections

thc.requested-vel <= motion.requested-vel

thc.current-vel <= motion.current-vel

FUNCTIONS

thc (requires a floating-point thread)

PINS

thc.encoder-vel float in

Connect to hm2_5i20.0.encoder.00.velocity

thc.current-vel float in

Connect to motion.current-vel

thc.requested-vel float in

Connect to motion.requested-vel

thc.volts-requested float in

Tip Volts current_vel >= min_velocity requested

thc.vel-tol float in
Velocity Tolerance (Corner Lock)

thc.torch-on bit in
Connect to motion.spindle-on

thc.arc-ok bit in
Arc OK from Plasma Torch

thc.enable bit in
Enable the THC, if not enabled Z position is passed through

thc.z-pos-in float in
Z Motor Position Command in from axis.n.motor-pos-cmd

thc.z-pos-out float out
Z Motor Position Command Out

thc.z-fb-out float out
Z Position Feedback to Axis

thc.volts float out
The Calculated Volts

thc.vel-status bit out
When the THC thinks we are at requested speed

PARAMETERS

thc.vel-scale float rw
The scale to convert the Velocity signal to Volts

thc.scale-offset float rw
The offset of the velocity input at 0 volts

thc.velocity-tol float rw
The deviation percent from planned velocity

thc.voltage-tol float rw
The deviation of Tip Voltage before correction takes place

thc.correction-vel float rw
The amount of change in user units per period to move Z to correct

AUTHOR

John Thornton

LICENSE

GPLv2 or greater

NAME

thcud – Torch Height Control Up/Down Input

SYNOPSIS

loadrt thcud

DESCRIPTION

Torch Height Control This THC takes either an up or a down input from a THC

If enabled and torch is on and X + Y velocity is within tolerance of set speed allow the THC to offset the Z axis as needed to maintain voltage.

If enabled and torch is off and the Z axis is moving up remove any correction at a rate not to exceed the rate of movement of the Z axis.

If enabled and torch is off and there is no correction pass the Z position and feed back untouched.

If not enabled pass the Z position and feed back untouched.

Physical Connections typical paraport.0.pin-12-in <= THC controller Plasma Up paraport.0.pin-13-in <= THC controller Plasma Down parport.0.pin-15-in <= Plasma Torch Arc Ok Signal parport.0.pin-16-out => Plasma Torch Start Arc Contacts

HAL Plasma Connections thc.torch-up <= paraport.0.pin-12-in thc.torch-down <= paraport.0.pin-13-in motion.spindle-on => parport.0.pin-16-out (start the arc) thc.arc-ok <= motion.digital-in-00 <= parport.0.pin-15-in (arc ok signal)

HAL Motion Connections thc.requested-vel <= motion.requested-vel thc.current-vel <= motion.current-vel

Pyvcp Connections In the xml file you need something like:

```
<checkbutton>
<text>"THC Enable"</text>
<halpin>"thc-enable"</halpin> </checkbutton> <spinbox>
<width>"5"</width>
<halpin>"vel-tol"</halpin>
<min_>.01</min_>
<max_>1</max_>
<resolution>0.01</resolution>
<initval>0.2</initval>
<format>"1.2f"</format>
<font>("Arial",10)</font> </spinbox>
```

Connect the Pyvcp pins in the postgui.hal file like this:

net thc-enable thcud.enable <= pyvcp.thc-enable

FUNCTIONS

thcud (requires a floating-point thread)

PINS

thcud.torch-up bit in

Connect to an input pin

thcud.torch-down bit in
Connect to input pin

thcud.current-vel float in
Connect to motion.current-vel

thcud.requested-vel float in
Connect to motion.requested-vel

thcud.torch-on bit in
Connect to motion.spindle-on

thcud.arc-ok bit in
Arc Ok from Plasma Torch

thcud.enable bit in
Enable the THC, if not enabled Z position is passed through

thcud.z-pos-in float in
Z Motor Position Command in from axis.n.motor-pos-cmd

thcud.z-pos-out float out
Z Motor Position Command Out

thcud.z-fb-out float out
Z Position Feedback to Axis

thcud.cur-offset float out
The Current Offset

thcud.vel-status bit out
When the THC thinks we are at requested speed

thcud.removing-offset bit out
Pin for testing

PARAMETERS

thcud.velocity-tol float rw
The deviation percent from planned velocity

thcud.correction-vel float rw
The Velocity to move Z to correct

AUTHOR

John Thornton

LICENSE

GPLv2 or greater

NAME

threads – creates hard realtime HAL threads

SYNOPSIS

```
loadrt threads name1=name period1=period [fp1=<0|1>] [<thread-2-info>] [<thread-3-info>]
```

DESCRIPTION

threads is used to create hard realtime threads which can execute HAL functions at specific intervals. It is not a true HAL component, in that it does not export any functions, pins, or parameters of its own. Once it has created one or more threads, the threads stand alone, and the **threads** component can be unloaded without affecting them. In fact, it can be unloaded and then reloaded to create additional threads, as many times as needed.

threads can create up to three realtime threads. Threads must be created in order, from fastest to slowest. Each thread is specified by three arguments. **name1** is used to specify the name of the first thread (thread 1). **period1** is used to specify the period of thread 1 in nanoseconds. Both *name* and *period* are required. The third argument, **fp1** is optional, and is used to specify if thread 1 will be used to execute floating point code. If not specified, it defaults to 1, which means that the thread will support floating point. Specify 0 to disable floating point support, which saves a small amount of execution time by not saving the FPU context. For additional threads, **name2**, **period2**, **fp2**, **name3**, **period3**, and **fp3** work exactly the same. If more than three threads are needed, unload threads, then reload it to create more threads.

FUNCTIONS

None

PINS

None

PARAMETERS

None

BUGS

The existence of **threads** might be considered a bug. Ideally, creation and deletion of threads would be done directly with **halcmd** commands, such as "**newthread** *name* *period*", "**delthread** *name*", or similar. However, limitations in the current HAL implementation require thread creation to take place in kernel space, and loading a component is the most straightforward way to do that.

NAME

threadtest – LinuxCNC HAL component for testing thread behavior

SYNOPSIS

loadrt threadtest [count=N|names=name1[,name2...]]

FUNCTIONS

threadtest.N.increment

threadtest.N.reset

PINS

threadtest.N.count u32 out

LICENSE

GPL

NAME

time – Time on in Hours, Minutes, Seconds

SYNOPSIS

loadrt time [count=N|names=name1[,name2...]]

DESCRIPTION

Time

When the time.N.start bit goes true the cycle timer resets and starts to time until time.N.start goes false. If you connect time.N.start to halui.is-running as a cycle timer it will reset during a pause. See the example connections below to keep the timer timing during a pause.

Time returns the hours, minutes, and seconds that time.N.start is true.

Sample pyVCP code to display the hours:minutes:seconds.

```
<pyvcp>
<hbox>
<label>
  <text>"Cycle Time"</text>
  <font>("Helvetica",14)</font>
</label>
<u32>
  <halpin>"time-hours"</halpin>
  <font>("Helvetica",14)</font>
  <format>"2d"</format>
</u32>
<label>
  <text>"."</text>
  <font>("Helvetica",14)</font>
</label>
<u32>
  <halpin>"time-minutes"</halpin>
  <font>("Helvetica",14)</font>
  <format>"2d"</format>
</u32>
<label>
  <text>"."</text>
  <font>("Helvetica",14)</font>
</label>
<u32>
  <halpin>"time-seconds"</halpin>
  <font>("Helvetica",14)</font>
  <format>"2d"</format>
</u32>
</hbox> </pyvcp>
```

In your post-gui.hal file you might use the following to connect it up

```
loadrt time
loadrt not
addf time.0 servo-thread
addf not.0 servo-thread
net prog-running not.0.in <= halui.program.is-idle
net cycle-timer time.0.start <= not.0.out
```

TIME(9)

HAL Component

TIME(9)

net cycle-seconds pyvcp.time-seconds <= time.0.seconds
net cycle-minutes pyvcp.time-minutes <= time.0.minutes
net cycle-hours pyvcp.time-hours <= time.0.hours

FUNCTIONS

time.N (requires a floating-point thread)

PINS

time.N.start bit in
 Timer On

time.N.seconds u32 out
 Seconds

time.N.minutes u32 out
 Minutes

time.N.hours u32 out
 Hours

AUTHOR

John Thornton

LICENSE

GPL

NAME

timedelay – The equivalent of a time-delay relay

SYNOPSIS

loadrt timedelay [count=N|names=name1[,name2...]]

FUNCTIONS

timedelay.N (requires a floating-point thread)

PINS

timedelay.N.in bit in

timedelay.N.out bit out

Follows the value of **in** after applying the delays **on-delay** and **off-delay**.

timedelay.N.on-delay float in (default: 0.5)

The time, in seconds, for which **in** must be **true** before **out** becomes **true**

timedelay.N.off-delay float in (default: 0.5)

The time, in seconds, for which **in** must be **false** before **out** becomes **false**

timedelay.N.elapsed float out

Current value of the internal timer

AUTHOR

Jeff Epler, based on works by Stephen Wille Padnos and John Kasunich

LICENSE

GPL

NAME

timedelta – LinuxCNC HAL component that measures thread scheduling timing behavior

SYNOPSIS

loadrt timedelta [count=N|names=name1[,name2...]]

FUNCTIONS

timedelta.N

PINS

timedelta.N.out s32 out
timedelta.N.err s32 out (default: 0)
timedelta.N.min s32 out (default: 0)
timedelta.N.max s32 out (default: 0)
timedelta.N.jitter s32 out (default: 0)
timedelta.N.avg-err float out (default: 0)
timedelta.N.reset bit in

LICENSE

GPL

NAME

toggle – 'push-on, push-off' from momentary pushbuttons

SYNOPSIS

loadrt toggle [count=N|names=name1[,name2...]]

FUNCTIONS

toggle.N

PINS

toggle.N.in bit in
button input

toggle.N.out bit io
on/off output

PARAMETERS

toggle.N.debounce u32 rw (default: 2)
debounce delay in periods

LICENSE

GPL

NAME

toggle2nist – toggle button to nist logic

SYNOPSIS

loadrt toggle2nist [count=N|names=name1[,name2...]]

DESCRIPTION

toggle2nist can be used with a momentary push button connected to a toggle component to control a device that has separate on and off inputs and has an is-on output. If in changes states via the toggle output
If is-on is true then on is false and off is true.
If is-on is false the on true and off is false.

FUNCTIONS

toggle2nist.N (requires a floating-point thread)

PINS

toggle2nist.N.in bit in
toggle2nist.N.is-on bit in
toggle2nist.N.on bit out
toggle2nist.N.off bit out

LICENSE

GPL

NAME

tristate_bit – Place a signal on an I/O pin only when enabled, similar to a tristate buffer in electronics

SYNOPSIS

```
loadrt tristate_bit [count=N|names=name1[,name2...]]
```

FUNCTIONS

tristate-bit.N

If **enable** is TRUE, copy **in** to **out**.

PINS

tristate-bit.N.in bit in

Input value

tristate-bit.N.out bit io

Output value

tristate-bit.N.enable bit in

When TRUE, copy in to out

LICENSE

GPL

NAME

tristate_float – Place a signal on an I/O pin only when enabled, similar to a tristate buffer in electronics

SYNOPSIS

loadrt tristate_float [count=N|names=name1[,name2...]]

FUNCTIONS

tristate-float.N (requires a floating-point thread)

If **enable** is TRUE, copy **in** to **out**.

PINS

tristate-float.N.in float in

Input value

tristate-float.N.out float io

Output value

tristate-float.N.enable bit in

When TRUE, copy in to out

LICENSE

GPL

NAME

updown – Counts up or down, with optional limits and wraparound behavior

SYNOPSIS

loadrt updown [count=N|names=name1[,name2...]]

FUNCTIONS

updown.N

Process inputs and update count if necessary

PINS

updown.N.countup bit in

Increment count when this pin goes from 0 to 1

updown.N.countdown bit in

Decrement count when this pin goes from 0 to 1

updown.N.reset bit in

Reset count when this pin goes from 0 to 1

updown.N.count s32 out

The current count

PARAMETERS

updown.N.clamp bit rw

If TRUE, then clamp the output to the min and max parameters.

updown.N.wrap bit rw

If TRUE, then wrap around when the count goes above or below the min and max parameters.
Note that wrap implies (and overrides) clamp.

updown.N.max s32 rw (default: *0xFFFFFFFF*)

If clamp or wrap is set, count will never exceed this number

updown.N.min s32 rw

If clamp or wrap is set, count will never be less than this number

LICENSE

GPL

NAME

watchdog – monitor multiple inputs for a "heartbeat"

SYNOPSIS

loadrt watchdog num_inputs=N

You must specify the number of inputs, from 1 to 32. Each input has a separate timeout value.

FUNCTIONS**process**

Check all input pins for transitions, clear the **ok-out** pin if any input has no transition within its timeout period. This function does not use floating point, and should be added to a fast thread.

set-timeouts

Check for timeout changes, and convert the float timeout inputs to int values that can be used in **process**. This function also monitors **enable-in** for false to true transitions, and re-enables monitoring when such a transition is detected. This function does use floating point, and it is appropriate to add it to the servo thread.

PINS**watchdog.input-n** bit in

Input number n. The inputs are numbered from 0 to **num_inputs**-1.

watchdog.enable-in bit in (default: *FALSE*)

If TRUE, forces out-ok to be false. Additionally, if a timeout occurs on any input, this pin must be set FALSE and TRUE again to re-start the monitoring of input pins.

watchdog.ok-out bit out (default: *FALSE*)

OK output. This pin is true only if enable-in is TRUE and no timeout has been detected. This output can be connected to the enable input of a **charge_pump** or **stepgen** (in v mode), to provide a heartbeat signal to external monitoring hardware.

PARAMETERS**watchdog.timeout-n** float in

Timeout value for input number n. The inputs are numbered from 0 to **num_inputs**-1. The timeout is in seconds, and may not be below zero. Note that a timeout of 0.0 will likely prevent **ok-out** from ever becoming true. Also note that excessively long timeouts are relatively useless for monitoring purposes.

LICENSE

GPL

NAME

wcomp – Window comparator

SYNOPSIS

loadrt wcomp [count=N|names=name1[,name2...]]

FUNCTIONS

wcomp.N (requires a floating-point thread)

PINS

wcomp.N.in float in

Value being compared

wcomp.N.min float in

Low boundary for comparison

wcomp.N.max float in

High boundary for comparison

wcomp.N.out bit out

True if **in** is strictly between **min** and **max**

wcomp.N.under bit out

True if **in** is less than or equal to **min**

wcomp.N.over bit out

True if **in** is greater than or equal to **max**

NOTES

If **max** <= **min** then the behavior is undefined.

LICENSE

GPL

NAME

weighted_sum – convert a group of bits to an integer

SYNOPSIS

loadrt weighted_sum wsum_sizes=size[,size,...]

Creates weighted sum groups each with the given number of input bits (*size*).

DESCRIPTION

This component is a "weighted summer": Its output is the offset plus the sum of the weight of each TRUE input bit. The default value for each weight is 2^n where n is the bit number. This results in a binary to unsigned conversion.

There is a limit of 8 weighted summers and each may have up to 16 input bits.

FUNCTIONS

process_wsums (requires a floating point thread)

Read all input values and update all output values.

PINS

wsum.N.bit.M.in bit in

The *m*'th input of weighted summer *n*.

wsum.N.hold bit in

When TRUE, the *sum* output does not change. When FALSE, the *sum* output tracks the *bit* inputs according to the weights and offset.

wsum.N.sum signed out

The output of the weighted summer

wsum.N.bit.M.weight signed rw

The weight of the *m*'th input of weighted summer *n*. The default value is 2^m .

wsum.N.offset signed rw

The offset is added to the weights corresponding to all TRUE inputs to give the final sum.

NAME

wj200_vfd – Hitachi wj200 modbus driver

SYNOPSIS

wj200_vfd

PINS

wj200-vfd.N.commanded-frequency float in

Frequency of vfd

wj200-vfd.N.reverse bit in

1 when reverse 0 when forward

wj200-vfd.N.run bit in

run the vfd

wj200-vfd.N.enable bit in

1 to enable the vfd. 0 will remote trip the vfd, thereby disabling it.

wj200-vfd.N.is-running bit out

1 when running

wj200-vfd.N.is-at-speed bit out

1 when running at assigned frequency

wj200-vfd.N.is-ready bit out

1 when vfd is ready to run

wj200-vfd.N.is-alarm bit out

1 when vfd alarm is set

wj200-vfd.N.watchdog-out bit out

Alternates between 1 and 0 after every update cycle. Feed into a watchdog component to ensure vfd driver is communicating with the vfd properly.

PARAMETERS

wj200-vfd.N.mbslaveaddr u32 rw

Modbus slave address

LICENSE

GPLv2 or greater

NAME

xhc_hb04_util – xhc-hb04 convenience utility

SYNOPSIS

```
loadrt xhc_hb04_util [count=N|names=name1[,name2...]]
```

DESCRIPTION

Provides logic for a start/pause button and an interface to halui.program.is_paused,is_idle,is_running to generate outputs for halui.program.pause,resume,run.

Includes 4 simple lowpass filters with coef and scale pins. The coef value should be $0 \leq \text{coef} \leq 1$, smaller coef values slow response. Note: the xhc_hb04 component includes smoothing so these values can usually be left at 1.0

FUNCTIONS

xhc-hb04-util.N (requires a floating-point thread)

PINS

xhc-hb04-util.N.start-or-pause bit in
xhc-hb04-util.N.is-paused bit in
xhc-hb04-util.N.is-idle bit in
xhc-hb04-util.N.is-running bit in
xhc-hb04-util.N.pause bit out
xhc-hb04-util.N.resume bit out
xhc-hb04-util.N.run bit out
xhc-hb04-util.N.in0 s32 in
xhc-hb04-util.N.in1 s32 in
xhc-hb04-util.N.in2 s32 in
xhc-hb04-util.N.in3 s32 in
xhc-hb04-util.N.out0 s32 out
xhc-hb04-util.N.out1 s32 out
xhc-hb04-util.N.out2 s32 out
xhc-hb04-util.N.out3 s32 out
xhc-hb04-util.N.scale0 float in (default: *1.0*)
xhc-hb04-util.N.scale1 float in (default: *1.0*)
xhc-hb04-util.N.scale2 float in (default: *1.0*)
xhc-hb04-util.N.scale3 float in (default: *1.0*)
xhc-hb04-util.N.coef0 float in (default: *1.0*)
xhc-hb04-util.N.coef1 float in (default: *1.0*)
xhc-hb04-util.N.coef2 float in (default: *1.0*)
xhc-hb04-util.N.coef3 float in (default: *1.0*)

LICENSE

GPL

NAME

xor2 – Two-input XOR (exclusive OR) gate

SYNOPSIS

loadrt xor2 [count=N|names=name1[,name2...]]

FUNCTIONS

xor2.N

PINS

xor2.N.in0 bit in

xor2.N.in1 bit in

xor2.N.out bit out

out is computed from the value of **in0** and **in1** according to the following rule:

in0=TRUE in1=FALSE

in0=FALSE in1=TRUE

out=TRUE

Otherwise,

out=FALSE

LICENSE

GPL

Appendix D: Visual Studio 2013 on the 826i

How to Create a New Project in Visual Studio 2013 on the 826 board

Creating a Program in C

Our group is doing all of our coding in C, and since VS doesn't have C as one of its default languages, special care should be taken when setting up a new project to be sure that it will compile in C.

1. Open VS
2. Select File -> New -> Project
3. When the New Project dialog box appears, select Visual C++ in the left pane.
4. In the Project window, select Win32 Console Application.
5. Name to Project.
6. When the Win32 Application Wizard box appears, click Next on the Welcome Page.
7. On the Application Settings, make sure the following are selected:
 - a. Application Type: Console Application
 - b. Additional Options: Empty Project
8. Click Finish

You now have a C project. Now we need to make the C files:

1. If Solution Explorer is not visible, go to View -> Solution Explorer.
2. Right click the Source Files folder in the Solution Explorer and select Add -> New Item
3. The New Item dialog box should appear.
4. Select C++ File(.cpp) and give it a name, but be sure to add the .c extension (for example, your file name might be 826controller.c)
5. Your source file is now in C, and you can start programming.

Compiling the Program

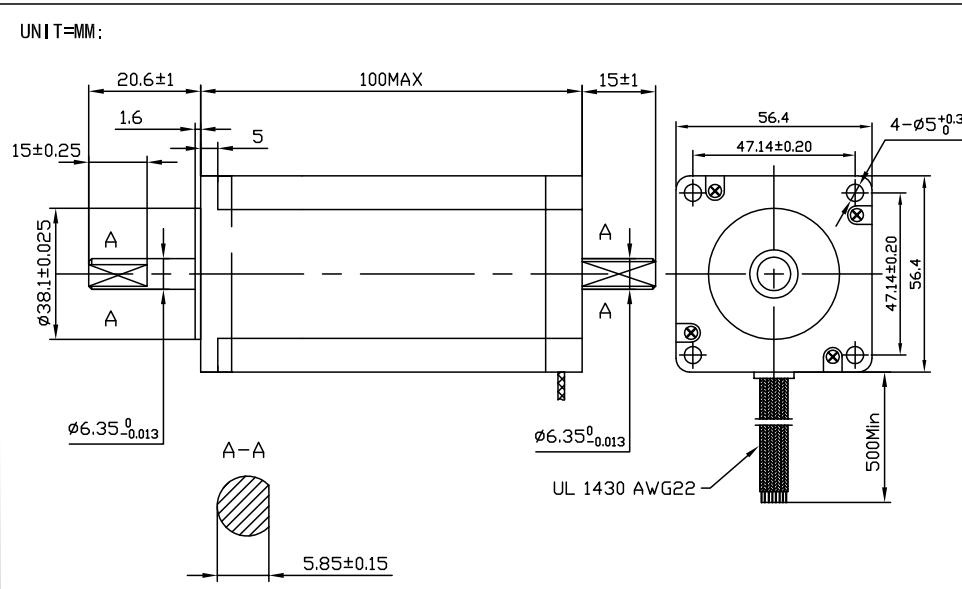
The only thing that needs to be done differently than compiling a normal program in VS is you need to add the DLL .lib file so VS can call on all the board's functions.

1. Click on Debug from the menubar and select "Project Name" Properties
2. When the dialog box opens, expand Linker on the left pane
3. Click on General
 - a. On the right hand side find Additional File Directories and click the down arrow on the far right to edit.
 - b. Type (or paste) in the address to the file where the .lib and .dll files are located.
4. Click on Input on the left pane
 - a. Select Additional Dependencies and click the down arrow to edit
 - b. Type (or paste) in the address of the .lib file
5. Close the Properties dialog box.
6. Your program is now ready to compile and talk to the 826 board

Appendix E: Stepper Motor and Driver

Hybrid Stepper Motor

KL23H2100-35-4B



PHASE	STEP ANGLE	RATED VOLTAGE	CURRENT	RESISTANCE	INDUCTANCE	HOLDING TORQUE	WEIGHT
	DEG/STEP	V	A	ohms	mH	OZ-IN	Kg
2	1.8	2.55	3.5	0.73	2.8	381	1.5

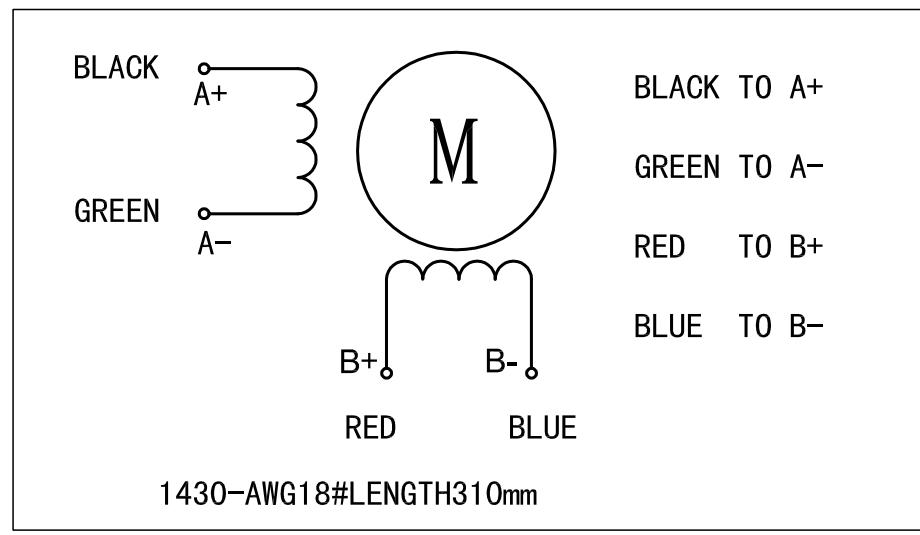


Table of Contents

KL-5056D

Fully Digital Stepping Driver

Attention: Please read this manual carefully before using the driver!

1.	Introduction, Features and Applications.....	1
	Introduction	1
	Features	1
	Applications.....	2
2.	Specifications	2
	Electrical Specifications	2
	Mechanical Specifications.....	2
	Elimination of Heat	3
	Operating Environment and other Specifications.....	3
3.	Pin Assignment and Description	3
	Connector P1 Configurations	3
	Selecting Active Pulse Edge and Control Signal Mode.....	4
	Connector P2 Configurations	4
4.	Control Signal Connector (P1) Interface.....	4
5.	Connecting the Motor.....	5
	Connections to 4-lead Motors	5
	Connections to 6-lead Motors	6
	Half Coil Configurations	6
	Full Coil Configurations.....	6
	Connections to 8-lead Motors	6
	Series Connections	6
	Parallel Connections.....	7
6.	Power Supply Selection.....	7
	Regulated or Unregulated Power Supply	7
	Multiple Drivers	8
	Selecting Supply Voltage.....	8
7.	Selecting Microstep Resolution and Driver Output Current	8
	Microstep Resolution Selection.....	9
	Current Settings.....	10

Contents

Dynamic current setting	10
Standstill current setting	10
8. Wiring Notes.....	11
9. Typical Connection.....	11
10. Sequence Chart of Control Signals.....	12
11. Protection Functions.....	12
Over-current Protection.....	12
Over-voltage Protection.....	13
Phase Error Protection.....	13
Protection Indications.....	13
12. Frequently Asked Questions.....	13
Problem Symptoms and Possible Causes	14
13. Professional Tuning Software ProTuner.....	15
Introduction	15
Software Installation.....	15
Connections and Testing.....	19
RS232 Interface Connection.....	20
Testing the Stepping System.....	20
Software Introduction.....	20
ProTuner Main Window	20
Com Config Window.....	21
Tuning.....	21
Anti-Resonance Introduction.....	24
Procedure for Achieving Optimum Performance	26
APPENDIX	Error! Bookmark not defined.
Twelve Month Limited Warranty	Error! Bookmark not defined.
Exclusions	Error! Bookmark not defined.
Obtaining Warranty Service	Error! Bookmark not defined.
Warranty Limitations	Error! Bookmark not defined.
Contact Us	Error! Bookmark not defined.

1. Introduction, Features and Applications

Introduction

The KL-5056D is a versatility fully digital stepping driver based on a DSP with advanced control algorithm. The KL-5056D is the next generation of digital stepping motor controls. It brings a unique level of system smoothness, providing optimum torque and nulls mid-range instability. Motor self-test and parameter auto-setup technology offers optimum responses with different motors and easy-to-use. The driven motors can run with much smaller noise, lower heating, smoother movement than most of the drivers in the markets. Its unique features make the KL-5056D an ideal solution for applications that require low-speed smoothness.

Features

- Anti-Resonance, provides optimum torque and nulls mid-range instability
- Motor self-test and parameter auto-setup technology, offers optimum responses with different motors
- Multi-Stepping allows a low resolution step input to produce a higher microstep output for smooth system performance
- Microstep resolutions programmable, from full-step to 102,400 steps/rev
- Supply voltage up to +50 VDC
- Output current programmable, from 0.5A to 5.6A
- Pulse input frequency up to 200 KHz
- TTL compatible and optically isolated input
- Automatic idle-current reduction
- Suitable for 2-phase and 4-phase motors
- Support PUL/DIR and CW/CCW modes
- Over-voltage, over-current, phase-error protections

Applications

Suitable for a wide range of stepping motors, from NEMA frame size 17 to 34. It can be used in various kinds of machines, such as laser cutters, laser markers, high precision X-Y tables, labeling machines, and so on. Its unique features make the KL-5056D an ideal solution for applications that require both low-speed smoothness and high speed performances.

2. Specifications

Electrical Specifications ($T_j = 25^\circ\text{C}/77^\circ\text{F}$)

Parameters	KL-5056D			
	Min	Typical	Max	Unit
Output current	0.5	-	5.6 (4.0 RMS)	A
Supply voltage	+20	+36	+50	VDC
Logic signal current	7	10	16	mA
Pulse input frequency	0	-	200	kHz
Isolation resistance	500			MΩ

Mechanical Specifications (unit: mm [inch])

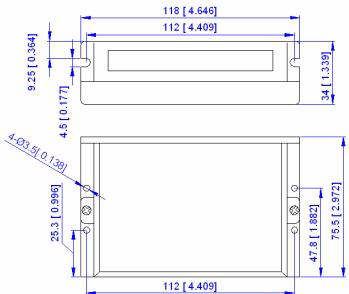


Figure 1: Mechanical specifications

Elimination of Heat

- Driver's reliable working temperature should be $<70^\circ\text{C}(158^\circ\text{F})$, and motor working temperature should be $<80^\circ\text{C}(176^\circ\text{F})$;
- It is recommended to use automatic idle-current mode, namely current automatically reduce to 60% when motor stops, so as to reduce driver heating and motor heating;
- It is recommended to mount the driver vertically to maximize heat sink area. Use forced cooling method to cool the system if necessary.

Operating Environment and other Specifications

Cooling	Natural Cooling or Forced cooling	
	Environment	Avoid dust, oil fog and corrosive gases
Operating Environment	Ambient Temperature	$0^\circ\text{C} - 50^\circ\text{C}(32^\circ\text{F} - 122^\circ\text{F})$
	Humidity	40%RH - 90%RH
	Operating Temperature	$70^\circ\text{C}(158^\circ\text{F})$ Max
	Vibration	5.9m/s^2 Max
Storage Temperature	$-20^\circ\text{C} - 65^\circ\text{C}(-4^\circ\text{F} - 149^\circ\text{F})$	
Weight	Approx. 280g (10 oz)	

3. Pin Assignment and Description

The KL-5056D has two connectors, connector P1 for control signals connections, and connector P2 for power and motor connections. The following tables are brief descriptions of the two connectors. More detailed descriptions of the pins and related issues are presented in section 4, 5, 9.

Connector P1 Configurations

Pin Function	Details
PUL+	<u>Pulse signal:</u> In single pulse (pulse/direction) mode, this input represents pulse signal, each rising or falling edge active (software configurable); 4-5V when PUL-HIGH, 0-0.5V when PUL-LOW. In double pulse mode (pulse/pulse), this input represents clockwise (CW) pulse, active both at high level and low level (software configurable). For reliable response, pulse width should be longer than 2.5us. Series connect resistors for current-limiting when +12V or +24V used. The same as DIR and ENA signals.
PUL-	<u>DIR signal:</u> In single-pulse mode, this signal has low/high voltage levels, representing two directions of motor rotation; in double-pulse mode (software configurable), this signal is counter-clock (CCW) pulse, active both at high level and low level (software configurable). For reliable motion response, DIR signal should be ahead of PUL signal by 5us at least. 4-5V when DIR-HIGH, 0-0.5V when DIR-LOW. Please note that rotation direction is also related to motor-driver wiring match. Exchanging the connection of two wires for a coil to the driver will reverse motion direction.
DIR+	<u>Enable signal:</u> This signal is used for enabling/disabling the driver. High level (NPN control signal, PNP and Differential control signals are on the contrary, namely Low level for enabling.) for enabling the driver and low level for disabling the driver. Usually left UNCONNECTED (ENABLED).
DIR-	
ENA+	
ENA-	

Contents

Selecting Active Pulse Edge and Control Signal Mode

The KL-5056D supports PUL/DIR and CW/CCW modes and pulse actives at rising or falling edge. See more information about these settings in Section 13. Default setting is PUL/DIR mode and rising edge active (NPN, and PNP control signal is on the contrary).

Connector P2 Configurations

Pin Function	Details
+Vdc	Power supply, 20-50 VDC, Including voltage fluctuation and EMF voltage.
GND	Power Ground.
A+, A-	Motor Phase A
B+, B-	Motor Phase B

4. Control Signal Connector (P1) Interface

The KL-5056D can accept differential and single-ended inputs (including open-collector and PNP output). The KL-5056D has 3 optically isolated logic inputs which are located on connector P1 to accept line driver control signals. These inputs are isolated to minimize or eliminate electrical noises coupled onto the drive control signals. Recommend use line driver control signals to increase noise immunity of the driver in interference environments. In the following figures, connections to open-collector and PNP signals are illustrated.

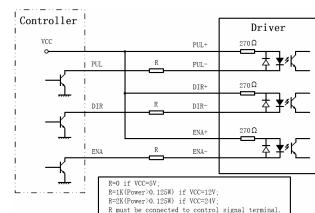


Figure 2: Connections to open-collector signal (common-anode)

Contents

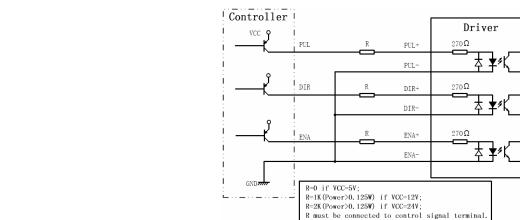


Figure 3: Connection to PNP signal (common-cathode)

5. Connecting the Motor

The KL-5056D can drive any 2-pahse and 4-pahse hybrid stepping motors.

Connections to 4-lead Motors

4 lead motors are the least flexible but easiest to wire. Speed and torque will depend on winding inductance. In setting the driver output current, multiply the specified phase current by 1.4 to determine the peak output current.

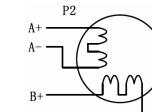


Figure 4: 4-lead Motor Connections

Connections to 6-lead Motors

Like 8 lead stepping motors, 6 lead motors have two configurations available for high speed or high torque operation. The higher speed configuration, or half coil, is so described because it uses one half of the motor's inductor windings. The higher torque configuration, or full coil, uses the full windings of the phases.

Half Coil Configurations

As previously stated, the half coil configuration uses 50% of the motor phase windings. This gives lower inductance, hence, lower torque output. Like the parallel connection of 8 lead motor, the torque output will be more stable at higher speeds. This configuration is also referred to as half chopper. In

Contents

setting the driver output current multiply the specified per phase (or unipolar) current rating by 1.4 to determine the peak output current.

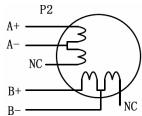


Figure 5: 6-lead motor half coil (higher speed) connections

Full Coil Configurations

The full coil configuration on a six lead motor should be used in applications where higher torque at lower speeds is desired. This configuration is also referred to as full copper. In full coil mode, the motors should be run at only 70% of their rated current to prevent over heating.

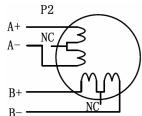


Figure 6: 6-lead motor full coil (higher torque) connections

Connections to 8-lead Motors

8 lead motors offer a high degree of flexibility to the system designer in that they may be connected in series or parallel, thus satisfying a wide range of applications.

Series Connections

A series motor configuration would typically be used in applications where a higher torque at lower speeds is required. Because this configuration has the most inductance, the performance will start to degrade at higher speeds. In series mode, the motors should also be run at only 70% of their rated current to prevent over heating.

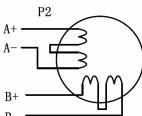


Figure 7: 8-lead motor series connections

Contents

Parallel Connections

An 8 lead motor in a parallel configuration offers a more stable, but lower torque at lower speeds. But because of the lower inductance, there will be higher torque at higher speeds. Multiply the per phase (or unipolar) current rating by 1.96, or the bipolar current rating by 1.4, to determine the peak output current.

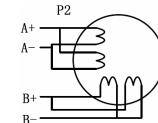


Figure 8: 8-lead motor parallel connections

NEVER disconnect or connect the motor while the power source is energized.

6. Power Supply Selection

The KL-5056D can match medium and small size stepping motors (from NEMA frame size 14 to 34) made by Keling or other motor manufactures around the world. To achieve good driving performances, it is important to select supply voltage and output current properly. Generally speaking, supply voltage determines the high speed performance of the motor, while output current determines the output torque of the driven motor (particularly at lower speed). Higher supply voltage will allow higher motor speed to be achieved, at the price of more noise and heating. If the motion speed requirement is low, it's better to use lower supply voltage to decrease noise, heating and improve reliability.

Regulated or Unregulated Power Supply

Both regulated and unregulated power supplies can be used to supply the driver. However, unregulated power supplies are preferred due to their ability to withstand current surge. If regulated power supplies (such as most switching supplies.) are indeed used, it is important to have large current output rating to avoid problems like current clamp, for example using 4A supply for 3A motor-driver operation. On the other hand, if unregulated supply is used, one may use a power supply of lower current rating than that of motor (typically 50% ~ 70% of motor current). The reason is that the driver draws current from the power supply capacitor of the unregulated supply only during the ON duration of the PWM cycle, but not during the OFF duration. Therefore, the average current withdrawn from power supply is considerably less than motor current. For example, two 3A motors can be well supplied by one power supply of 4A rating.

Contents

Multiple Drivers

It is recommended to have multiple drivers to share one power supply to reduce cost, if the supply has enough capacity. To avoid cross interference, **DO NOT** daisy-chain the power supply input pins of the drivers. Instead, please connect them to power supply separately.

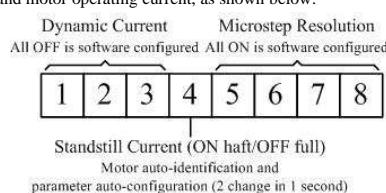
Selecting Supply Voltage

The power MOSFETs inside the KL-5056D can actually operate within +20 ~ +50VDC, including power input fluctuation and back EMF voltage generated by motor coils during motor shaft deceleration. Higher supply voltage can increase motor torque at higher speeds, thus helpful for avoiding losing steps. However, higher voltage may cause bigger motor vibration at lower speed, and it may also cause over-voltage protection or even driver damage. Therefore, it is suggested to choose only sufficiently high supply voltage for intended applications, and it is suggested to use power supplies with theoretical output voltage of +20 ~ +46VDC, leaving room for power fluctuation and back-EMF.

7. Selecting Microstep Resolution and Driver Output Current

Microstep resolutions and output current are programmable, the former can be set from full-step to 102,400 steps/rev and the latter can be set from 0.5A to 5.6A. See more information about **Microstep and Output Current Setting** in Section 13.

However, when it's not in software configured mode, this driver uses an 8-bit DIP switch to set microstep resolution, and motor operating current, as shown below:



Microstep Resolution Selection

When it's not in software configured mode, microstep resolution is set by SW5, 6, 7, 8 of the DIP switch as shown in the following table:

Contents

Microstep	Steps/rev.(for 1.8°motor)	SW5	SW6	SW7	SW8
1 to 512	Default/Software configured	ON	ON	ON	ON
2	400	OFF	ON	ON	ON
4	800	ON	OFF	ON	ON
8	1600	OFF	OFF	ON	ON
16	3200	ON	ON	OFF	ON
32	6400	OFF	ON	OFF	ON
64	12800	ON	OFF	OFF	ON
128	25600	OFF	OFF	OFF	ON
5	1000	ON	ON	ON	OFF
10	2000	OFF	ON	ON	OFF
20	4000	ON	OFF	ON	OFF
25	5000	OFF	OFF	ON	OFF
40	8000	ON	ON	OFF	OFF
50	10000	OFF	ON	OFF	OFF
100	20000	ON	OFF	OFF	OFF
125	25000	OFF	OFF	OFF	OFF

Current Settings

For a given motor, higher driver current will make the motor to output more torque, but at the same time causes more heating in the motor and driver. Therefore, output current is generally set to be such that the motor will not overheat for long time operation. Since parallel and serial connections of motor coils will significantly change resulting inductance and resistance, it is therefore important to set driver output current depending on motor phase current, motor leads and connection methods. Phase current rating supplied by motor manufacturer is important in selecting driver current, however the selection also depends on leads and connections.

When it's not in software configured mode, the first three bits (SW1, 2, 3) of the DIP switch are used to set the dynamic current. Select a setting closest to your motor's required current.

Contents

Dynamic current setting

Peak Current	RMS Current	SW1	SW2	SW3
Default/Software configured (0.5 to 5.6A)		OFF	OFF	OFF
2.1A	1.5A	ON	OFF	OFF
2.7A	1.9A	OFF	ON	OFF
3.2A	2.3A	ON	ON	OFF
3.8A	2.7A	OFF	OFF	ON
4.3A	3.1A	ON	OFF	ON
4.9A	3.5A	OFF	ON	ON
5.6A	4.0A	ON	ON	ON

Notes: Due to motor inductance, the actual current in the coil may be smaller than the dynamic current setting, particularly under high speed condition.

Standstill current setting

SW4 is used for this purpose. OFF meaning that the standstill current is set to be half of the selected dynamic current, and ON meaning that standstill current is set to be the same as the selected dynamic current.

The current automatically reduced to 60% of the selected dynamic current one second after the last pulse. Theoretically, this will reduce motor heating to 36% (due to $P=I^2 \cdot R$) of the original value. If the application needs a different standstill current, please contact Keling.

8. Wiring Notes

- In order to improve anti-interference performance of the driver, it is recommended to use twisted pair shield cable.
- To prevent noise incurred in PUL/DIR signal, pulse/direction signal wires and motor wires should not be tied up together. It is better to separate them by at least 10 cm, otherwise the disturbing signals generated by motor will easily disturb pulse direction signals, causing motor position error, system instability and other failures.
- If a power supply serves several drivers, separately connecting the drivers is recommended instead of daisy-chaining.
- It is prohibited to pull and plug connector P2 while the driver is powered ON, because there is high current flowing through motor coils (even when motor is at standstill). Pulling or plugging

Contents

connector P2 with power on will cause extremely high back-EMF voltage surge, which may damage the driver.

9. Typical Connection

A complete stepping system should include stepping motor, stepping driver, power supply and controller (pulse generator). A typical connection is shown as figure 9.

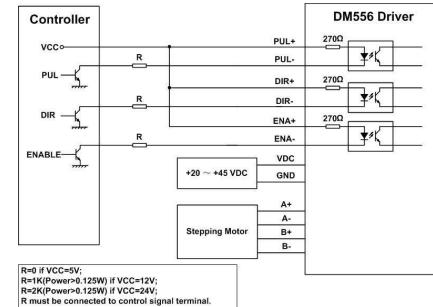


Figure 9: Typical connection

10. Sequence Chart of Control Signals

In order to avoid some fault operations and deviations, PUL, DIR and ENA should abide by some rules, shown as following diagram:

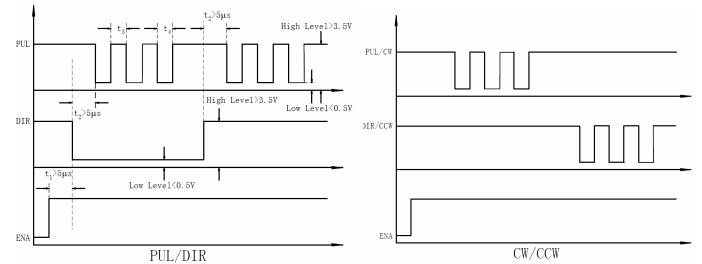


Figure 10: Sequence chart of control signals

Remark:

- a) t1: ENA must be ahead of DIR by at least 5 μ s. Usually, ENA+ and ENA- are NC (not connected). See "Connector P1 Configurations" for more information.
- b) t2: DIR must be ahead of PUL active edge by 5 μ s to ensure correct direction;
- c) t3: Pulse width not less than 2.5 μ s;
- d) t4: Low level width not less than 2.5 μ s.

11. Protection Functions

To improve reliability, the driver incorporates some built-in protection functions. The KL-5056D uses one RED LED to indicate what protection has been activated. The periodic time of RED is 3 s (seconds), and how many times the RED turns on indicates what protection has been activated. Because only one protection can be displayed by RED LED, so the driver will decide what error to display according to their priorities. See the following **Protection Indications** table for displaying priorities.

Over-current Protection

Over-current protection will be activated when continuous current exceeds 16A or in case of short circuit between motor coils or between motor coil and ground, and RED LED will turn on once within each periodic time (3 s).

Over-voltage Protection

When power supply voltage exceeds 52±1 VDC, protection will be activated and RED LED will turn on twice within each periodic time (3 s).

Phase Error Protection

Motor power lines wrong & not connected will activate this protection. RED LED will turn on four times within each periodic time (3 s).

Attention: When above protections are active, the motor shaft will be free or the LED will turn red. Reset the driver by repowering it to make it function properly after removing above problems. Since there is no protection against power leads (+, -) reversal, it is critical to make sure that power supply leads correctly connected to driver. Otherwise, the driver will be damaged instantly.

Protection Indications

Priority	Time(s) of ON	Sequence wave of RED LED	Description
1 st	1		Over-current protection
2 nd	2		Over-voltage protection
3 rd	4		Phase error protection

12. Frequently Asked Questions

In the event that your driver doesn't operate properly, the first step is to identify whether the problem is electrical or mechanical in nature. The next step is to isolate the system component that is causing the problem. As part of this process you may have to disconnect the individual components that make up your system and verify that they operate independently. It is important to document each step in the troubleshooting process. You may need this documentation to refer back to at a later date, and these details will greatly assist our Technical Support staff in determining the problem should you need assistance.

Many of the problems that affect motion control systems can be traced to electrical noise, controller software errors, or mistake in wiring.

Problem Symptoms and Possible Causes

Symptoms	Possible Problems
Motor is not rotating	No power
	Microstep resolution setting is wrong
	DIP switch current setting is wrong
	Fault condition exists
	The driver is disabled
Motor rotates in the wrong direction	Motor phases may be connected in reverse
The driver in fault	DIP switch current setting is wrong
	Something wrong with motor coil

Contents

	Control signal is too weak
	Control signal is interfered
	Wrong motor connection
	Something wrong with motor coil
Erratic motor motion	Current setting is too small, losing steps
	Current setting is too small
	Motor is undersized for the application
	Acceleration is set too high
	Power supply voltage too low
Motor stalls during acceleration	
	Inadequate heat sinking / cooling
Excessive motor and driver heating	Automatic current reduction function not being utilized
	Current is set too high

Contents

13. Professional Tuning Software ProTuner

Introduction

This section will provide an overview of connection and basic setup instructions for Keling's digital stepping driver KL-5056D using the **ProTuner** software. These instructions will walk you through the following steps necessary to start up your driver and motor. This section is intended for setting up the driver with the **ProTuner**.

Software Installation

The **ProTuner** is windows based setup software for tuning Keling's digital stepper driver KL-5056D. It can run in windows systems, including Win95/Win98/WindowsNT/ Windows 2000/Windows XP. And the selected PC should have 1 serial port at least for communicating with the driver.

Double click “**ProTuner_All_Setup_V1.0.exe**” to begin installing the **ProTuner**. See Figure 11. Click **Next** to enter the “License Agreement” window. See Figure 12.



Figure 11: Begin to install the ProTuner

Contents



Figure 12: License agreement

Choose "I agree to the terms of this license agreement" and click **Next** to continue installation. The user can enter user's information in the following window. See Figure 13. After entering the user's information, click **Next** to select installation folder, where you would like to install the **ProTuner**. See Figure 14.

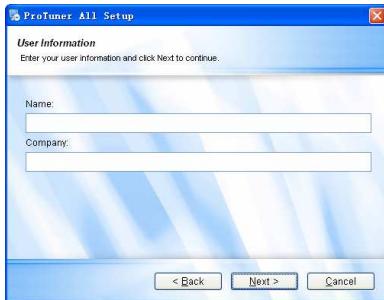


Figure 13: User's information settings

Contents

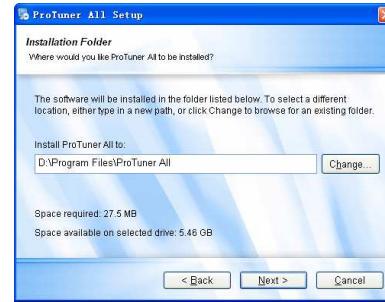


Figure 14: Installation folder settings

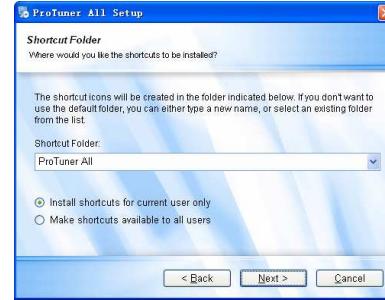


Figure 15: Shortcut folder setting

Set the "Shortcut Folder" in Figure 15 and continue to install the **ProTuner** by following Figure 16 and Figure 17. An **Installation Successful** window will appear if the **ProTuner** is installed successfully. See Figure 18.

Contents



Figure 16: Installation information summarization

Contents



Figure 18: Finish installation

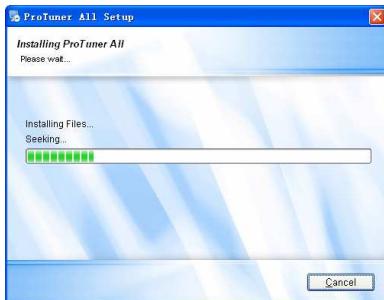


Figure 17: Installing the ProTuner

Connections and Testing

Connect the stepping system according to the contents in previous sections and connect the PC to the driver as the following figure.

RS232 Interface Connection

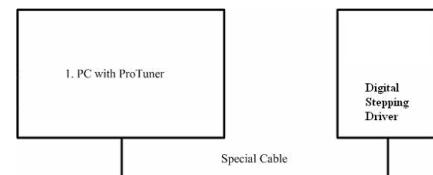


Figure 19: RS232 interface connection

Testing the Stepping System

Turn on the power supply, the green (Power) LED will light. The KL-5056D has default parameters stored in the driver. If the system has no hardware and wirings problem, the motor should be locked and the driver should be ready.

If the red LED immediately turns on (flickers), then check power supply, the motor, motor wirings

Contents

and try again. Open the tuning software **ProTuner** and check driver status by clicking **Err_check**. If it's **Phase Error**, check the motor, motor wirings and try again. If it still doesn't work after you followed all of the previous steps, please contact us at kelinginc@kelinginc.net

If the RED LED is off and the motor is normal, then you can start to tune the servo with **ProTuner**. However, we recommend you see the following contents before starting tuning.

Software Introduction

ProTuner Main Window

➤ Option

The user can choose three drop-down menus by clicking "**Option**", including **Com Config**, **SaveToDriver** and **Exit**.

- **Com Config:** Configure Com communication interface.
- **SaveToDriver:** Download the current parameter settings to the driver.
- **Exit:** Exit the **ProTuner**.

Com Config Window

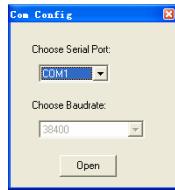


Figure 21: RS232 communication configuration window

Serial Port: Select the serial communication port to which the driver is connected. The factory default setting is COM1.

Baud Rate: Select the communication baud rate. The factory default setting is 38400.

Click **Open** button to establish a connection with the specified settings. When connecting, you can

Contents

choose **SaveToDrive** to download the current parameter settings to the driver, or to upload the stored driver settings into the **ProTuner** by clicking **Tuning > Position Loop** on the menu bar.

Tuning

The user can choose one or two drop-down menu(s) by clicking **Tuning**, including **CurrentLoop** and **SystemConfig**.

- **CurrentLoop:** In Current Tuning window, the user can tune the **Kp (Proportional Gain)** and **Ki (Integral Gain)** of driver's current loop to optimize responses with different motors. Start/Restart a Step Response test to get an optimum response.

Kp: Proportional Gain. Proportional Gain determines the response of the driver to current setting command. Low Proportional Gain provides a stable system (doesn't oscillate), has low stiffness, and large current error, causing poor performances in tracking current setting command in each step like Figure 23. Too large Proportional Gain values will cause oscillations and unstable systems.

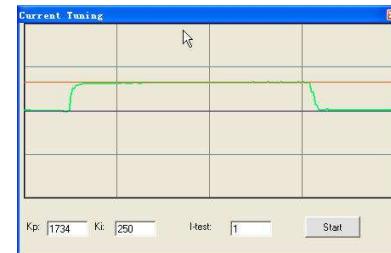


Figure 22: Current Tuning window

Ki: Integral Gain. Integral Gain helps the driver to overcome static current errors. A low or zero value for the Integral Gain may have current errors at rest. Increasing the Integral Gain can reduce the error. If the Integral Gain is too large, the systems may "hunt" (oscillate) about the desired position.

Start button: The user can start a Step Response test by clicking this button. Start/Restart a Step

Contents

Response test to get an optimum response like Figure 22, and remember to save the settings to the driver when finish tuning. See Figure 24.



Figure 23: Kp=2604, Ki=0 (poor performances)

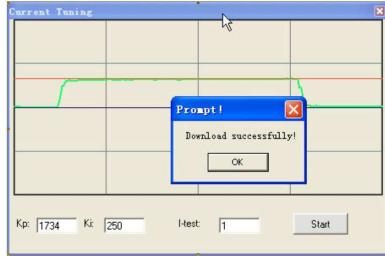


Figure 24: Finish tuning and save setting to the driver

Notes:

However, if the user does not want to tune the current loop after changing a different stepping motor, then **Motor auto-identification and parameter auto-configuration** technology of the KL-5056D can replace manual tuning the driver with **ProTuner**. Just changes SW4 two times in 1 second, and then the driver will auto-identify the new motor and auto-configure related control parameters for optimum responses. **Recommend** use this function after changing the driven motor.

- **SystemConfig:**

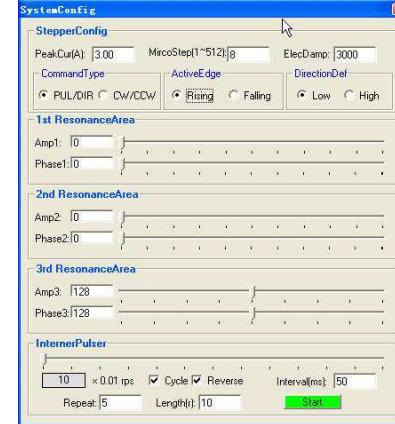
Contents

In **SystemConfig** window, the user can configure Peak Current, Microstep, Command Type, Active Edge, and eliminate motor resonance. A built-in pulse generator can be used for test during tuning. See Picture 25.

PeakCur: Peak Current. The value is the peak current to the selected motor and can be set from 0.5 to 5.6 A. The user can set the peak current with **ProTuner** or DIP switches, see more information about setting output current of the driver in section 5 “**Connecting the Motor**” and section 7 “**Selecting Microstep Resolution and Driver Output Current**”.

MicroStep: Microstep Resolution. The value is driver’s microstep resolution setting and can be set from 1 to 512. The user can set the microstep with **ProTuner** or DIP switches, See more information about setting output current of the driver in section 7 “**Selecting Microstep Resolution and Driver Output Current**”.

ElecDamp: Electronic Damping Coefficient. The electronic damping restrain resonance of the system and prevent amplitude of the oscillation from increasing to the extend that it makes the motor out of control. The optimal value depends on the system, and the default value is 3000.



Contents

Figure 25: SystemConfig window

CommandType: **Command Type** of control signal, including PUL/DIR and CW/CCW. Set this parameter according to **Command Type** of motion controller.

ActiveEdge: **Active Edge.** The user can set the triggered edge of pulse command signal in this panel. When the driver works in CW/CCW mode, no matter what level is at fixed level terminal, the driver can work properly.

DirectionDef: **Direction Definition.** Relate the default running direction to a **HIGH** level input in DIR or **Low** level input in DIR. This panel is used for PUL/DIR command type only. Please note that the default direction is also related to motor coil connections.

Anti-Resonance Introduction

Step motors are highly resonant, which results in vibration and ringing. The ringing utilizes a large fraction of the motor's available torque – thereby wasting performance. Furthermore, at mid-range velocities, the resonance can become so severe that the motor loses synchronization and stalls. The KL-5056D driver provides robust anti-resonance control to stop the vibrations and maintain equilibrium. This feature requires that the driver be configured with respect to the total inertia in the system. If set improperly, the effectiveness of the feature may be diminished.

The user can invoke or disable the feature by setting **Amp** and **Phase** values in **SystemConfig** window. **Amp** and **Phase** values all zero is to disable the feature, otherwise is to invoke the feature. It should be enabled unless the system configuration either does not need it or cannot tolerate it. A system with loose couplings or viscous loading generally does not need this feature. If a system has compliant (springy) coupling and is absent appreciably viscosity, it may not respond well to the active, anti-resonant loop in the drive. The anti-resonant feature is not designed to damp such a 4th order system. If the application of anti-resonance results in degradation or instability, it should be disabled.

1st ResonanceArea: Parameters for 1st resonance area. Usually between 0.6rps and 1.2rps.

Amp1 is Amplitude adjustment for 1st resonance area.

Phase1 is Phase adjustment for 1st resonance area. The user can enter a value directly in the text box or move the slider bar back and forth to get an optimum value.

Contents

2nd ResonanceArea: Parameters for 2nd resonance area. Usually between 1.2rps and 2.4rps. Default **Amp2** and **Phase2** values are zero.

3rd ResonanceArea: Parameters for 3rd resonance area. Usually between 2.4rps and 4.8rps. Default **Amp3** and **Phase3** values are 128.

InternerPulser: There is an internal pulse generator designed for driver self-testing and anti-resonance tuning. You can issue a motion by this simple controller.

Cycle check box: The motion will repeat if this box is checked.

Reverse check box: The motor shaft will reverse direction if this box is checked.

Interval edit box: The stop time between each cycle, unit is **millisecond**.

Repeat edit box: Total motion cycles.

Length edit box: Move distance of each cycle, unit is **revolution**.

Start/Stop button: The user can Start/Stop a motion test by clicking this button.

Procedure for Achieving Optimum Performance

Step 1: Start the motion test by clicking **Start/Stop** button. Find a resonance speed by slightly moving the slider bar of internal pulse generator back and forth. See Figure 26.

Step 2: Run the motor at the resonance speed and verify the motor smoothness. You may find a better smoothing value by slightly moving the slider bars of **AMP(s)** and **Phase(s)** back and forth.

It is very important to make the **AMP(s)** and **Phase(s)** adjustments at the proper test speeds with an unloaded motor. Running at an incorrect test speed will not excite the motor at its peak resonance, making it more difficult to find proper adjustment values. Optimum **AMP(s)** and **Phase(s)** values may be a little different between running the tests with an unloaded motor and a load motor.

Please remember to click **SavetoDrive** to download the final parameter settings to the driver when finish tuning. See Figure 27.

Contents

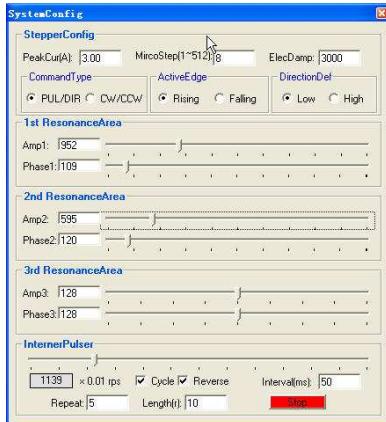


Figure 26: Anti-resonance tuning

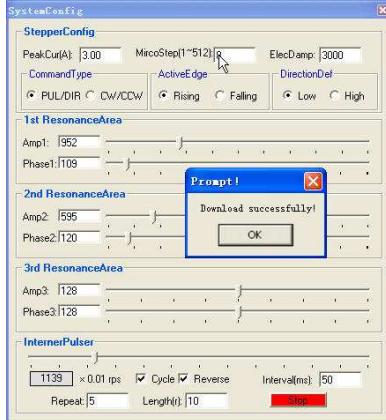


Figure 27: Finishing tuning and download parameter settings to the driver

Contents

➤ **Err_check**

- **Error Check:** This window shows both the present status of each error event and their history. Current error event(s) can be reset by clicking **Erase Current Err!** button, and all error events can be reset by clicking **Erase All!** button. List of the last ten drive faults. #0 being the most recent, #9 is the oldest. See Figure 28.

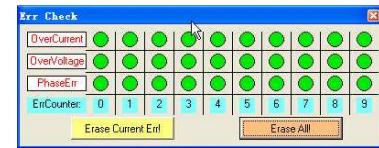


Figure 28: Error check window

OverCurrent: **Over-current Protection.** Protection will be activated when continuous current exceeds 16A.

OverVoltage: **Over-voltage Protection.** When power supply voltage exceeds 52 ± 1 VDC, protection will be activated.

PhaseErr: **Phase Error Protection.** Motor power lines wrong & not connected will activate this protection.

ErrCounter: Displays current error(s) and current error history.

Erase Current Err!: **Erase Current Err** button. The user can clear current error(s) by clicking this button.

Erase All!: **Erase All!** button. The user can clear all error(s) including error history by clicking this button.

➤ **About**

The user can choose two drop-down menus by clicking "About", including **Product Information** and

Contact Us.

- **Product Information** window: Shows some product information about ProTuner.

Appendix F: Optical Shaft Encoder

Description

The **S5** series optical shaft encoder is a non-contacting rotary to digital converter. Useful for position feedback or manual interface, the encoder converts real-time shaft angle, speed, and direction into TTL-compatible quadrature outputs with or without index. The encoder utilizes a mylar disk, metal shaft and bushing, LED light source, and monolithic electronics. It operates from a single +5VDC supply.

Three shaft torque versions are available. The standard torque version has a sleeve bushing lubricated with a viscous motion control gel to provide torque and feel that is ideal for front panel human interface applications.

The no torque added option has a sleeve bushing and a low viscosity lubricant (that does not intentionally add torque) for low RPM applications where a small amount of torque is acceptable.

The ball bearing version uses miniature precision ball bearings that are suitable for high speed and ultra low torque applications.

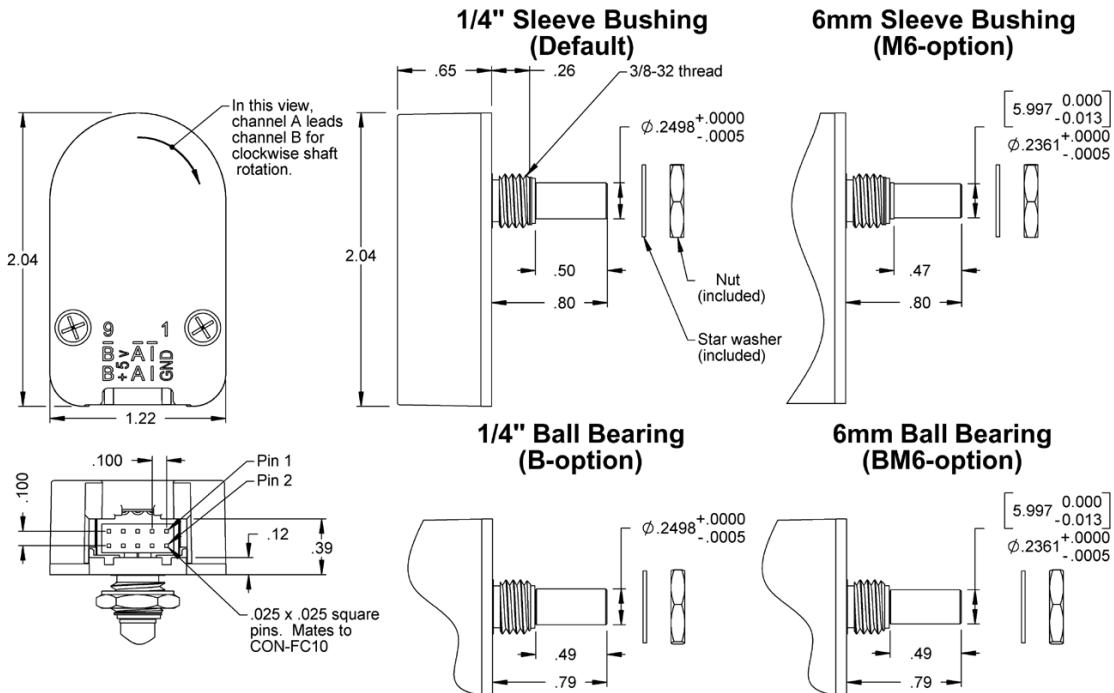
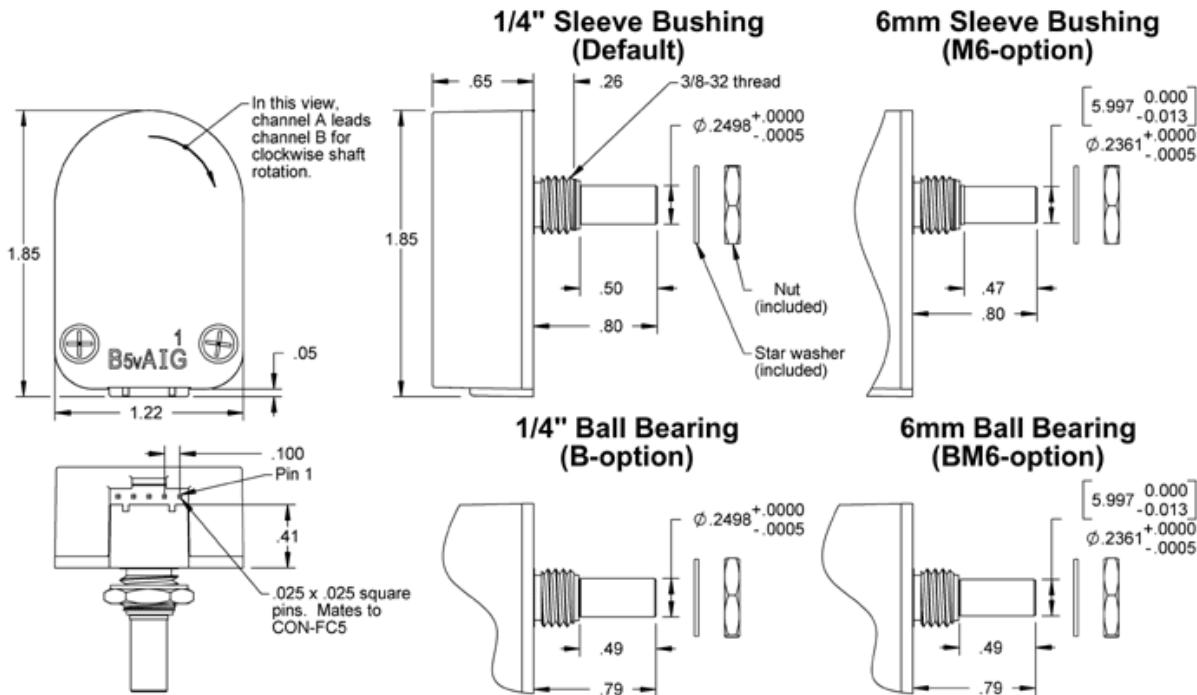
A secure connection to the **S5** series encoder is made through a 5-pin (single-ended version) or 10-pin (differential version) finger-latching connector (sold separately). The mating connectors are available from US Digital with several cable options and lengths.

For differential version: the internal differential line driver (26C31) can source and sink 20mA at TTL levels. The recommended receiver is industry standard 26C32. Maximum noise immunity is achieved when the differential receiver is terminated with a $150\ \Omega$ resistor in series with a $.0047\ \mu F$ capacitor placed across each differential pair. The capacitor simply conserves power; otherwise power consumption would increase by approximately 20mA per pair, or 60mA for 3 pairs.



Features

- Small size
- Low cost
- Optional differential / line-driver output
- Positive finger-latching connector
- 2-channel quadrature, TTL squarewave outputs
- 3rd channel index option
- Ball bearing option tracks to 10,000 RPM
- -25 to +100C operating temperature
- Single +5VDC supply

 **Differential**

 **Single-Ended**


 **Environmental**

Parameter	Value	Units
Operating Temperature, CPR < 2000	-40 to 100	C
Operating Temperature, CPR ≥ 2000	-25 to 100	C
Vibration (5Hz to 2kHz)	20	G
Electrostatic Discharge, Human Body Model	± 4	kV

 **Mechanical**

Parameter	Sleeve Bushing	Ball Bearing
Max. Acceleration	250000 rad/sec ²	250000 rad/sec ²
Max. Shaft Speed	100 rpm	10000 rpm
Max. Shaft Torque	0.5 ± 0.2 in-oz 0.3 in-oz (N -option)	0.05 in-oz
Max. Shaft Loading	2 lbs. dynamic 20 lbs. static	1 lb.
Bearing Life	> 1000000 revolutions	$L_{10} = (19.3/F_r)^3$ * Where L_{10} = bearing life in millions of revs, and F_r = radial shaft loading in pounds
Weight		
Single-ended	1.01 oz.	1.15 oz.
Differential	1.28 oz.	1.42 oz.
Max. Shaft Total Indicated Runout	0.0015 in.	0.0015 in.
Max. Panel Nut Tightening Torque	20 in-lbs	20 in-lbs
Technical Bulletin TB1001 - Shaft and Bore Tolerances		Download

* only valid with negligible axial shaft loading.

 **Phase Relationship**

B leads A for clockwise shaft rotation, and A leads B for counterclockwise rotation viewed from the shaft side of the encoder (see the EM1 page).

 **Single-ended Electrical**

- Specifications apply over entire operating temperature range.
- Typical values are specified at V_{cc} = 5.0Vdc and 25 ° C.
- For complete details, see the EM1 or EM2 product pages.

Parameter	Min.	Typ.	Max.	Units	Conditions
Supply Voltage	4.5	5.0	5.5	V	
Supply Current	27	33	mA		CPR < 500, no load
	54	62	mA		CPR ≥ 500 and < 2000, no load
	72	85	mA		CPR ≥ 2000, no load
Low-level Output		0.5	V		IOL = 8mA max., CPR < 2000
		0.5	V		IOL = 5mA max., CPR ≥ 2000
		0.25	V		no load, CPR ≥ 2000
High-level Output	2.0		V		IOH = -8mA max. and CPR < 2000
	2.0		V		IOH = -5mA max. and CPR ≥ 2000
	4.8		V		no load and CPR < 2000
	3.5		V		no load and CPR ≥ 2000
Output Current Per Channel	-8	8	mA		CPR < 2000
	-5	5	mA		CPR ≥ 2000
Output Rise Time	110		nS		CPR < 2000
	50		nS		CPR ≥ 2000, ± 5mA load
Output Fall Time	100		nS		CPR < 2000
	50		nS		CPR ≥ 2000, ± 5mA load

Differential Electrical

- Specifications apply over entire operating temperature range.
- Typical values are specified at Vcc = 5.0Vdc and 25 °C.
- For complete details, see the EM1 product page.

Parameter	Min.	Typ.	Max.	Units	Conditions
Supply Voltage	4.5	5.0	5.5	V	
Supply Current	29	36	mA		CPR < 500, no load
	57	65	mA		CPR ≥ 500 and < 2000, no load
	73	88	mA		CPR ≥ 2000, no load
Low-level Output		0.2	0.4	V	IOL = 20mA max.
High-level Output	2.4	3.4		V	IOH = -20mA max.
Differential Output Rise/Fall Time			15	nS	

Pin-outs

5-pin Single-ended: (1)



1400 NE 136th Avenue
Vancouver, Washington 98684, USA

info@usdigital.com
www.usdigital.com

Local: 360.260.2468
Toll-free: 800.736.0194



S5 Optical Shaft Encoder

Page 5 of 7



Pin	Description
1	Ground
2	Index
3	A channel
4	+5VDC power
5	B channel

10-pin Differential Standard: (2)

Pin	Description
1	Ground
2	Ground
3	Index-
4	Index+
5	A- channel
6	A+ channel
7	+5VDC power
8	+5VDC power
9	B- channel
10	B+ channel

- (1) 5-pin single-ended mating connector is CON-FC5.
(2) 10-pin differential mating connector is CON-FC10.



1400 NE 136th Avenue
Vancouver, Washington 98684, USA

info@usdigital.com
www.usdigital.com

Local: 360.260.2468
Toll-free: 800.736.0194

 **Ordering Information**

S5 - - - - -

CPR

32 =

50 =

96 =

100 =

192 =

200 =

250 =

256 =

360 =

400 =

500 =

512 =

540 =

720 =

900 =

1000 =

1024 =

1250 =

2000 =

2048 =

2500 =

4000 =

4096 =

5000 =

Shaft236 = *Metric 6mm diameter shaft*250 = *1/4" diameter***Index**NE = *No Index*IE = *Index***Output**S = *Single-ended*D = *Differential***Torque**D = *Default*B = *Ball Bearing*N = *No torque added***Notes**

- Cables and connectors are not included and must be ordered separately.
- For ordering information please see the Compatible Cables / Connectors section above.
- US Digital warrants its products against defects in materials and workmanship for two years. See complete warranty for details.

**S5****Optical Shaft Encoder**
Page 7 of 7

Base Pricing

Quantity	Price
1	\$87.50
5	\$64.85
10	\$55.87

For volume discounts, please contact us at sales@usdigital.com or 800.736.0194.

- Add 11% per unit for **CPR** of , , , or
- Add \$1.00 per unit for **Shaft** of Metric 6mm diameter shaft
- Add 23% per unit for **Output** of Differential
- Add \$5.80 per unit for **Torque** of Ball Bearing
- Add 17% per unit for **Index** of IE or **CPR** greater than or equal to 1000.



1400 NE 136th Avenue
Vancouver, Washington 98684, USA

info@usdigital.com
www.usdigital.com

Local: 360.260.2468
Toll-free: 800.736.0194