

SPARETIMELABS

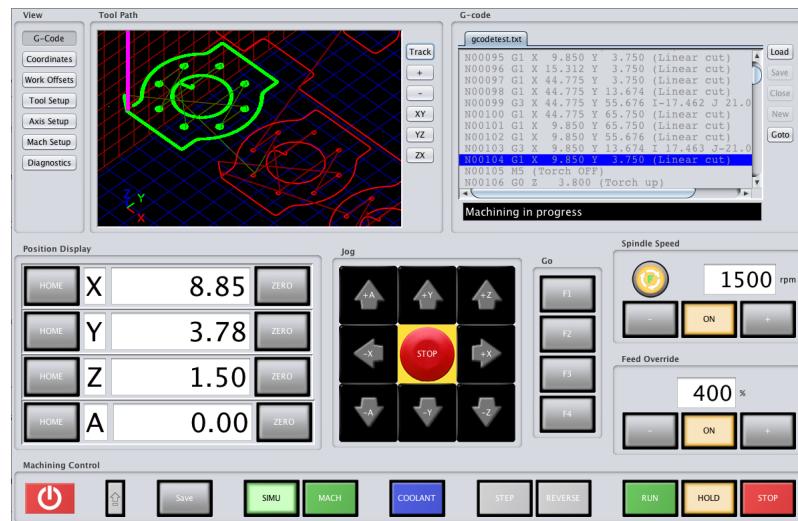
eazycnc@sparetimelabs.com

EazyCNC – Manual

this is an early access draft revision

for

EazyCNC version 0.0.0.18a



April 6, 2014

Contents

1 Safety First!	12
2 Introduction to CNC-Machining	14
2.1 Overview of a CNC Machining System	14
2.2 Understanding Real-Time	15
2.3 Understanding Stepper Motors	15
3 Hardware and Software Requirements	18
3.1 Dedicated Computer	18
3.2 Hardware	18
3.3 Operating System	19
3.4 Java	19
3.5 Anti-Virus Software	20
4 Installation	21
4.1 Getting Java	21
4.2 Getting the Application	21
4.3 Installing the Application	22
5 Overview of EazyCNC	23
5.1 Moving Around in the Program	23
6 Setting Up and Configuring	26

CONTENTS	2
6.1 Saving Your Setup	26
6.2 Setting the Communication Port Name	27
6.2.1 Using the Auto Detect port function	28
6.2.2 Installing the Windows Serial Port Driver	28
6.2.3 Finding out the virtual serial port name on Mac OS X	29
6.2.4 Finding out the virtual serial port name in Linux	29
6.3 Setting your work Units	30
6.4 Configuring Motors and Axes	31
6.4.1 Motor Config -panel	31
6.4.2 Axis Limits -panel	36
6.4.3 Jogging -panel	37
6.4.4 Test -panel	38
6.4.5 Pre-requisites	38
6.4.6 The Test run	39
6.5 Setting the Movement limits	40
6.5.1 Update Period -entry field	42
6.6 Spindle setup	43
6.6.1 Min Spee -entry field	43
6.6.2 Max Spee -entry field	44
6.7 G-Code Options	44
6.7.1 Incremental IJK -checkbox	44
6.7.2 Incremental XYZ -checkbox	45
6.7.3 G4 P in msec -checkbox	45
6.8 Shortcuts setup	45
6.9 Diagnostics screen	46
7 Operating	47

7.1	Simulation versus Cutting Metal	47
7.2	Status Display	48
7.3	G-code display	48
7.3.1	Loading G-code for execution	48
7.3.2	The Goto -button	49
7.3.3	Editing G-code	49
7.4	Toolpath display	50
7.4.1	Controlling the toolpath display	51
7.5	Coordinate displays aka DROs	51
7.6	Jogging	52
7.7	Finding your bearings i.e. coordinates	53
7.8	The easy and lazy way	54
7.9	Going pro	54
7.10	Adjusting Feed Rate	55
7.11	Controlling the Spindle	55
7.12	Machining!	56
7.12.1	Pausing the machining	56
7.12.2	Stepping and Reversing	57
7.12.3	Stopping	58
7.13	Setting up and managing the coordinate systems	58
7.13.1	Coordinate axes	58
7.13.2	Work/Fixture Coordinate System/Offsets	58
7.13.3	Selecting the active coordinate system	59
7.13.4	Changing offsets/setting up the coordinates	59
7.13.5	Setting the XY-coordinate system origin via touching	60
7.13.6	Setting the Z-coordinate system origin via touching	60

CONTENTS	4
7.14 Setting up and managing tool information	61
7.14.1 Setting the current tool	62
7.14.2 Managing the tool diameter and length	62
7.14.3 Setting the tool length via touching	62
8 Cutter compensation	64
8.1 Tool length compensation	64
8.2 Cutter diameter compensation	64
8.2.1 Cutter compensation example - cutting part's outline	65
8.2.2 Cutter compensation example - cutting holes and openings	67
9 G-code reference	68
9.1 The Basics	68
9.2 Numbers, Expressions and Parameters	69
9.2.1 Numbers	69
9.2.2 Expressions	69
9.2.3 Parameters	71
9.3 G-codes and M-codes	72
9.3.1 Length Units, G20,G21 codes	72
9.3.2 Coordinate Axes	72
9.3.3 Setting the length units – G20,G21	72
9.3.4 Feedrate – F-word	72
9.3.5 Spindle speed – S-word	73
9.3.6 Spindle on/off – M3,M4,M5 codes	73
9.3.7 Coolant on/off – M7,M8,M9 codes	73
9.3.8 Select a tool – T-word	74
9.3.9 Dwelling – G4 P*-code	74

CONTENTS	5
9.3.10 Coordinates/moving axes – XYZABC -words	74
9.3.11 Motion mode – G0,G1,G2 and G3 codes	75
9.3.12 Rapid positioning – G0 code	75
9.3.13 Linear interpolation – G1 code	75
9.3.14 Clockwise Arc interpolation – G2 code	76
9.3.15 Counter Clockwise Arc interpolation – G3 code	77
9.3.16 Perform probing move – G31	77
9.4 Coordinate systems	77
9.4.1 Scaling – G50,G51 codes	78
9.4.2 Incremental mode – G90,G91 codes	78
9.4.3 Polar coordinate mode – G15,G16 codes	79
9.4.4 Temporary coordinate system offsets – G52	79
9.4.5 Temporary coordinate system offsets – G52,G92,G92.1,G92.2,G92.3 codes	80
9.4.6 Coordinate system rotation – G68,G69 codes	80
9.4.7 Active plane – G17,G18,G19 codes	81
9.4.8 Tool length compensation – G43,G44,G49 codes	81
9.4.9 Work/fixtures offsets – G54,G55,G56,G57,G58,G59 codes	81
9.4.10 Absolute coordinates – G53 code	82
9.4.11 Cutter radius compensation – G40,G41,G4 codes	82
9.4.12 Feedrate mode – G93,G94,G95 codes	83
9.4.13 Feedrate override on/off – M48,M49 codes	83
9.4.14 Tool change – M6 code	83
9.4.15 Tool length compensation – G43,G44,G49 codes	83
9.4.16 Path mode – G61,G61.1,G64 codes	84
9.4.17 Incremental XYZ mode – G90,G91 codes	84
9.4.18 Incremental IJK mode – G90.1,G91.1 codes	85

<i>CONTENTS</i>	6
9.4.19 Set tool table – G10 L1 code	85
9.4.20 Set work/fixture offsets – G10 L2 code	85

List of Tables

6.1	Motors versus configuration jumpers	34
6.2	Jumpers versus Step Mode	35
9.1	Mathematical functions	70
9.2	Coordinate transformations	78

List of Figures

2.1	a CNC Machining System Overview	14
5.1	EazyCNC Main Screen – the G-code view	24
5.2	The view selection buttons	25
6.1	The communication port setup screen	27
6.2	The Units setup screen	30
6.3	The Axis setup screen	31
6.4	The movement setup screen	40
6.5	The spindle setup screen	44
6.6	The G-code setup screen	44
6.7	The Shortcuts setup screen	46
6.8	The Diagnostics screen	46
7.1	The operating mode control and indicator buttons	47
7.2	The status/error display	48
7.3	The G-code editor and display panel	49
7.4	The toolpath display panel	50
7.5	The Digital Readouts	52
7.6	The Jog control buttons	53
7.7	The Feed Override controls	55
7.8	The Spindle controls	56

<i>LIST OF FIGURES</i>	9
------------------------	---

7.9 The G-code execution control buttons	56
7.10 The step and reverse execution control buttons	57
7.11 The coordinate systems	58
7.12 The Work Offsets screen	59
7.13 The Tool Setup Screen	61
8.1 G-code path versus compensated cutter path	66
8.2 Cutter compensation detail	66
8.3 G-code path versus compensated cutter path	67
9.1 Coordinate axes of a 3-axis CNC System	72

Preface

This is an early access draft revision of this manual which implies that there are lots of typographical, lexical, grammar and spelling failures, not to mention factual errors and omissions, so please proceed with caution.

Disclaimer

EazyCNC is program to control the operation of a CNC-Machine Tool.

Any machine tool is potentially dangerous.

All electrical system have the potential to cause an electric shock or a fire hazard.

All motorised systems can cause serious personal injury or damage to property.

All computer programs have design or implementation flaws (bugs) some of which can cause serious malfunction of the system.

Most countries and states have regulations and standards that govern the design, construction, use, deployment and placing on the market of electrical and mechanical equipment, including the software used to control them.

No safety of design or construction or programming nor warranty is implied, instead *it is your responsibility to ensure that you understand the implications of using EazyCNC and/or TOAD4 and to comply with any legislation and codes of practice applicable to your country or state.*

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

SpareTimeLabs or Kustaa Nyholm cannot accept any responsibility resulting from the design, construction or use of EazyCNC and/or TOAD4.

All names of products and trademarks used in this manual are for example purposes only, no endorsement of them by SpareTimeLabs nor endorsement of EazycNC or TOAD4 by their respective owners is implied.

Chapter 1

Safety First!

Machine tools are dangerous!

Always keep that in mind, both when designing and setting up your system and when operating it on a daily bases.

CNC machine tools are heavy and strong machinery, moving sharp and hot cutting tools or extremely powerful plasma torches under computer control. Computers are complex systems and it is *impossible* to ensure 100% error free and safe operation in every situation. It is perfectly possible that a software design flaw, called bug, cause the system to operate unexpectedly or even run away wild.

Therefore it is very important to take appropriate precautions for such an eventuality.

Every system needs to have an Emergency switch fitted.

The emergency switch needs to be so wired that it will prevent any machine movement and stop spindle or shutdown torch arc when activated.

The emergency switch needs to be mounted to a place that is easily accessible place.

The emergency switch has to be of the latching kind in other words once activated it must stay activated until manually de-activated.

Depending on the physical layout and power of your machine movements you need to consider if you should activate the emergency switch whenever you have your hands or limbs inside the working area of your machine.

With some machine configurations it may be preferable not to activate the emergency switch if you need to pause the system for in the middle of machining, for example to change the tool bit because the axes might loose their positions and it maybe acceptable to just ensure that the spindle will not start on its own.

For that purpose a kill switch to the spindle motor controller maybe fitted that will prevent the spindle from running no matter what the control systems does.

Above does not by any means endorse any particular way of ensuring safety and no responsibility or liability is accepted by me. You need to do your own risk and safety assessment and act accordingly.

Chapter 2

Introduction to CNC-Machining

2.1 Overview of a CNC Machining System

This is probably familiar territory for you otherwise you would not be here in the first place but this section is short introduction to tell you where exactly EazyCNC and TOAD4 fits in the big picture.

Figure 2.1 tries to capture the 'big' picture of a CNC machining system.

Figure 2.1: a CNC Machining System Overview

The CNC machining process starts with a design of the part to be machined which is turned into a sequence of instructions to the computer that controls the motors, typically stepper motors, that move the cutting tool (or work piece) via series of gears, belts, pulleys and/or screws. These tool movements are typically called 'axes', for example X-axis, Y-axis etc.

The 'sequence of instructions' is called G-code and it is basically a text file with coordinate points that define the path the cutting tool will make.

G-code can be hand written but is typically generated automatically from a CAD (Computer Aided Design model) of the part using CAM (Computer Aided Manufacturing) software, either directly by the CAD/CAM program or by a program called post-processor.

The G-code file is read by a program that turns the coordinate information and other commands in the G-code file into motor control pulses in real-time observing programmed feed rates and machine parameters such as number of pulses required to move a unit distance.

This is where EazyCNC/TOAD4 comes into picture because EazyCNC is the program that does the G-code interpretation and TOAD4 is a micro controller that does the real-time motor control.

2.2 Understanding Real-Time

In common language real time means roughly 'as it happens' but in computer jargon real-time has a specific and important meaning.

Real-time here means that the pulses that control the distance and speed of movements need to be generated precisely at appropriate rate because the motors and mechanisms that move the axis have physical limits beyond which they fail to move as required.

EazyCNC runs on a personal computer such as an IBM PC compatible or a Macintosh computer. These computers use an operating systems, such as Windows, Linux or Mac OS X, that are not ideally suited to real-time control. You can easily get a feel for this when you plug in a USB-device as often the mouse cursor stops for a second or two – imaging if the system paused like that when it should be turning a corner.

There are different ways out of this difficulty. A Windows program called Mach3 uses a special 'driver' software for the real-time stuff and another program called EMC2 uses a specially 'patched' version of the Linux operating system just to mention two.

EazyCNC takes a different approach.

EazyCNC delegates the most demanding real-time tasks to the TOAD4 micro controller which is better equipped and positioned to do the precise real-time generation of motor control pulses and such because it does not have to deal with the endless variety of the PC hardware and software and because it has been designed from ground up for the very task of performing real-time control.

EazyCNC reads and interprets the G-code and breaks it into bite size chunks that the simple micro controller in the TOAD4 can process in real-time.

These bite size chunks are transferred from the PC to TOAD4 over the USB bus and put into a command queue in the TOAD4 micro controller.

EazyCNC attempts to keep the queue full at all times so that if EazyCNC or the operating system it runs on needs to 'pause' for a second or so there is still data for TOAD4 to process and machining can continue without missing a beat.

This is important because when stepper motors are run at high speed a delayed pulse is equivalent to a sudden deceleration which may cause the system to exceed the stepper motor's max torque in which case the stepper motor will not be able to 'hold' its position and accuracy is compromised.

2.3 Understanding Stepper Motors

While EazyCNC and TOAD4 can be used with Servo Motors they really are meant to be used with Stepper Motors.

Stepper Motor are motors with two or more stationary coils and a rotating permanent magnet

rotor. By energising the coils in sequence the rotor can be made to move.

Stepper Motors have some interesting and important properties.

First of all they do not 'run' if you just energise them, at most they make a single small movement called step. This makes them rather safe as a short circuited transistor in the drive system cannot make the motor run wildly.

Secondly when you apply a controlled energising sequence into the coils the motor makes precise fractional rotational movements called steps, a typical stepper motor step is 1.8° or 200 steps/revolution.

It is this second property that makes steppers very attractive for controlling precise movements.

If you 'step' a stepper motor ten times you can be pretty sure that it actually moved ten steps. So there is no need to measure position of the motor in any other way than counting the pulses we send to it, no need for expensive encoders and or position scales.

This makes stepper motors very cost effective way to implement motion control.

But the lack of position feedback is also the downside of stepper motors.

The positional control of a stepper motor based system relies on an initial position and keeping track of the steps and their direction.

The initial position can be either given manually or found automatically by using a reference switch.

Manually means that you move the motor/axis to a known position, such as to the end of the movement range, and tell the system that this is it.

If a reference switch is available then the system can move the axis/motor through its range of movements and make a note of the step number on which the reference switch is reached and in this way calibrate its position.

Keeping track of the pulse and their direction is done automatically and precisely by the software but under certain circumstances the motor cannot 'carry out the step' and the system loses track of the real physical position, this is called missing steps.

Missing steps can happen if the stepper is stepped too fast or the load exceeds what the motor can deliver. To prevent that the correct maximum speed and acceleration parameters need to be programmed into the system.

Also note that manually forcing the axes to move, if you manage to overpower the motors, will cause the system to lose its position, so all manual movements must be done via the 'jog' controls of the system.

It must also be mentioned that since we are controlling the motor position, not the cutter position, any backlash or play in the mechanism is *not* automatically compensated for.

Typically not of practical concern but good to know is the fact that stepper motors are not very 'stiff'. Even though the motor has specific torque it actually has very little torque when the magnetic poles of the coil and rotor are aligned i.e. at every full step.

You can think about this as if the rotor and stator poles were connected with springs; when the poles are aligned the spring will not pull the rotor one way or the other, they only exert force and torque when the rotor is moved into misalignment and thus there is almost always a small but measurable error between the physical and ideal step position.

There is of course a lot more to know and understand about Stepper Motors but above is the most import thing to keep in mind when working with systems based on them.

Chapter 3

Hardware and Software Requirements

This chapter gives you important information about the software and hardware requirements for a system based on EazyCNC and TOAD4.

3.1 Dedicated Computer

The computer used for CNC machining should be dedicated for the CNC system and not shared for other use.

A CNC machine should not be directly connected to internet because of the possibility of viruses and malware causing havoc in an environment where they can cause actual physical damage and injury.

If the dedicated CNC PC is connected to a local network it is important that the network is secure and runs behind a firewall and the all network connectivity on the CNC PC is kept to minimum.

No other software besides the operating system and EazyCNC should be running on the CNC PC during machining. You should especially watch out for programs that start automatically behind the scenes when the computer boots up. Utility programs to kill non required applications and processes exist.

Once the system has been set up, configured properly and tested it should be 'freezed' and *any upgrades and changes to the system should be approached with due caution and care.*

3.2 Hardware

It is impossible to give hard limits as to which kind of PC computer should be used with EazyCNC.

In a less demanding application at moderate feed rates you can get by with a less powerful computer.

It is tempting to utilise that old PC that is just lying there gathering dust, but that is not recommended.

Consider that you are building an automated machining tool and for that you want reliability and responsiveness. A suitable PC can be purchased for a few hundred euros/dollars and is well worth it considering what you are building and what your are going to do with it.

A fairly recent and decent PC with at least 1 GHz processor, good graphics card, a minimum of 2 GB RAM is recommended as the minimum. To install and run EazyCNC you need at least 200 MB of free disk space even if the application itself is only about 12 MB.

During machining all the G-code data as well as the tool path graphics is kept in memory so there is no such thing as too much memory, too fast graphics card or too fast CPU!

3.3 Operating System

The following operating systems/version have been tested:

- Mac OS X 10.8.5
- Windows XP sp 2
- Windows 7
- Linux Lubuntu 13.10

EazyCNC is built on Java which is fairly operating system independent so it is likely it will run with a wide range of version of above mentioned operating systems but of course it is not possible to guarantee that.

Also worth noting is that operating system version tend to become obsolete and unsupported in a matter of some years and while I try to maintain compatibility with older OS versions it may become impossible at anytime.

As an example, the trusty old Windows XP is well beyond its sell-by-date and no longer supported in any shape or form by Microsoft but lots of people still use it. Today EazyCNC runs on Windows XP with but this situation may change.

3.4 Java

EazyCNC requires Java Runtime Environment (JRE).

Java is an operating system independent virtual machine that allows software written on one operating system to run more or less seamlessly on other operating systems.

Java was originally developed by Sun and later acquired by Oracle and Open Sourced. Before Java was Open Sourced the GNU movement created an incomplete but Free clone of Java.

EazyCNC has been tested on the Sun/Oracle Java and you are advised to use that. Having said that it is likely that EazyCNC will also work fine with OpenJDK Java, but it will not work with GNU Java.

EazyCNC has been tested with Java version 7 but it should also run just fine on Java 6, especially on Mac OS X.

Note that the JRE version has an effect on the real-time qualities of the EazyCNC so you should not trivially and without due diligence upgrade the JRE version, even if recommended by media hype and vulnerability scares – a computer not connected to Internet is not subject to the alleged Java vulnerabilities.

In the future EazyCNC will have Java JRE built in so the Java version question will become moot.

You can download Oracle Java JRE from:

<http://java.com>

On pre Mountain Lion Macs Java is pre-installed so you do not need to and should not install a newer version from Oracle.

3.5 Anti-Virus Software

An anti-virus software is not recommended on the PC running EazyCNC as such software is very intrusive and can cause real-time violations and machining failures. Further, it is not very necessary because the CNC PC should not be connected to Internet.

It is of course important that anti-virus software or some other means are used to ensure clean operation of the computer producing or transferring the G-code file as a virus might infect the media, such as a USB memory stick, used to transfer the G-code to the CNC system.

Chapter 4

Installation

4.1 Getting Java

As a pre-requisite you need to have Java installed in your system. You can get Java from

<http://java.com>

Older Mac OS version have Java pre installed and you should not install a newer version from Oracle.

In Linux you can also install Java with one of the various package managers.

Regardless how you get it, ensure that you have Oracle Java.

In the future Java will be built in so this will become a moot point.

4.2 Getting the Application

Note that do not need to have actual hardware or any drivers installed to run and play with the software.

EazyCNC is distributed via Internet so just download the file appropriate for your operating system from the EazyCNC website at:

<http://www.sparetimelabs.com/eazycnc/downloads/downloads.php>

The download size is about 10 MB.

Depending on your operating system EazyCNC is distributed as a compressed file which you may have to un-compress with the tools in you operating system.

4.3 Installing the Application

Remember that you should have the Java Runtime Environment installed first.

EazyCNC comes as a single executable file which does not require an installer. Just copy the file to where ever you want in your computers hard disk.

If you prefer you can create a shortcut aka alias for the application and put it on the Start Menu or Desktop.

To launch the application just double click on the file.

If you have Java installed the program will start.

In Linux you may have to give the application execute permissions first. You can do that by right clicking at the application file and selecting 'Properties' and in the Dialog that appears you should be able to set the file as executable.

Chapter 5

Overview of EazyCNC

The main purpose of EazyCNC is to read machining instructions in the form of a G-code file and control accordingly the motors that move the machine tool. In addition to that it provides the necessary controls to manually move the machine axes.

5.1 Moving Around in the Program

Figure 5.1 shows the 'main screen' of the application. Note that if your screen resolution is small then the titles of the boxes aka panels are not displayed to conserve some screen real estate.

The screen is logically divided into two parts, upper and lower half. The lower half is always the same, but the upper half changes depending on which screen or view you are.

On the upper left corner, Figure 5.2 are buttons that control what is shown on the upper half. When this manual says 'go to screen' or 'in screen' it refers to these view control buttons and the different screens they bring up.

The lower half of the screen always shows the controls that are used to cause the machine to actually move.

For a typical usage the main screen, 'G-code', provides all the controls that are necessary to open a G-code file and machine the part it represents.

The other screens are for setting up coordinate systems, tool parameters and to configure various aspects of the machining system.

On the main screen the upper right quarter of the screen the current G-code file is shown with the line G-code line being executed highlighted in blue.

On the upper left quarter a 3D view of the tool path is shown with the path already executed shown in green and the path that remains to be machined shown in red.

Below those, from left to right, there are the coordinate read outs (DROs) displaying the tool position, jog controls to manually move the tool, user programmable function keys for repeated

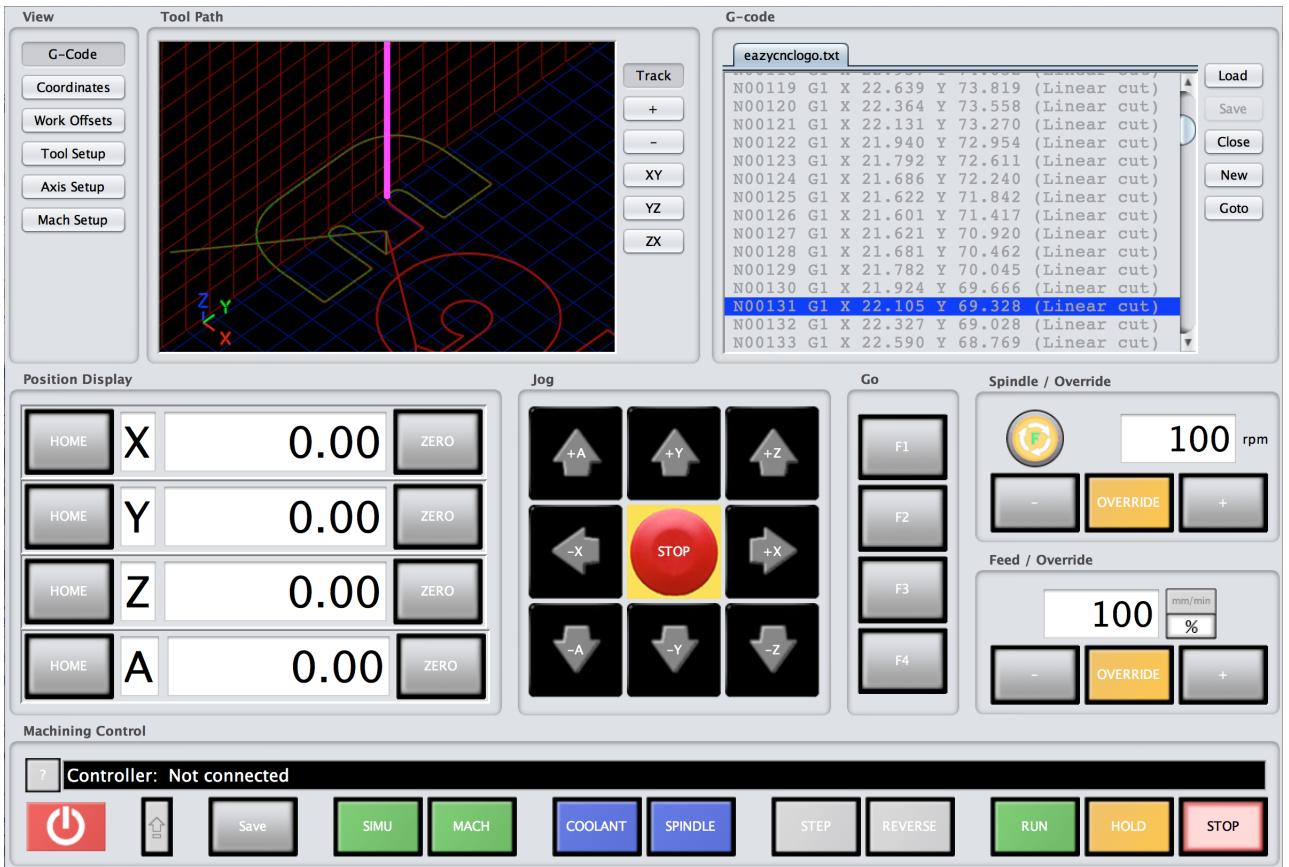


Figure 5.1: EazyCNC Main Screen – the G-code view

tasks and manual spindle controls and feed override controls.

At the bottom row there are buttons to control the actual machining and running of the G-code program, either in simulation mode or actually cutting some metal. With these controls it is also possible to temporarily pause the execution and run the G-code step by step and even backwards.

Usage and cutting metal with EazyCNC is described in detail in Chapter ??.

Setting up of EazyCNC is described in the following chapter.



Figure 5.2: The view selection buttons

Chapter 6

Setting Up and Configuring

EazyCNC should work out of the box without any setup or configuration, so you can play around and test it right away.

However before you can actually use it with TOAD4 and machine something there are a few things you need to set up and configure.

This chapter proceeds in the preferred order of setting things up, however I suggest you read it all through once before setting up the system.

The two most important things to set up are communication between your computer and you TOAD4 and configure each axis to match the motor characteristics and axis gearing.

6.1 Saving Your Setup

EazyCNC stores all setup and configuration information as well as the current machine state in file named 'EazyCNC-Mach-Config.ecnc' in a folder/directory named 'EazyCNC' in your 'home' directory.

Your 'home' directory location depends on your operating system as follows:

Mac OS X	/Users/username
Windows XP	C:\Documents and Settings\username
Windows 7	C:\Users\username
Linux	/Users/username

where *username* is the name you use when you log into your computer.

Note, EazyCNC does not ever automatically save the settings, this is to protect you from accidentally altering your carefully crafted machine setup and configuration. To save your settings or any changes you've made you need to click the 'SAVE' button.

If the file or folder does not exist EazyCNC will create them with reasonable default settings when you click the 'SAVE' button.

You do not have to 'worry' about this file but it is good to know about it as you may want to make backup copies of it or maintain several different ones for different system configurations.

The file is in a text format so it is ok to view and even edit it manually, though that is not recommended unless you know what you are doing.

6.2 Setting the Communication Port Name

Remember that you do not need to do this to play with the program or to use it to simulate machining, perhaps on a different computer from your CNC computer.

TOAD4 looks like a virtual serial port to your computers operating system.

EazyCNC uses the operating systems standard serial port programming interface and device drivers to talk to TOAD4.

To set the communication port name start EazyCNC and from the top/left corner buttons click 'Mach Setup' button and then from the top row of buttons click 'Comm', see Figure 6.1.

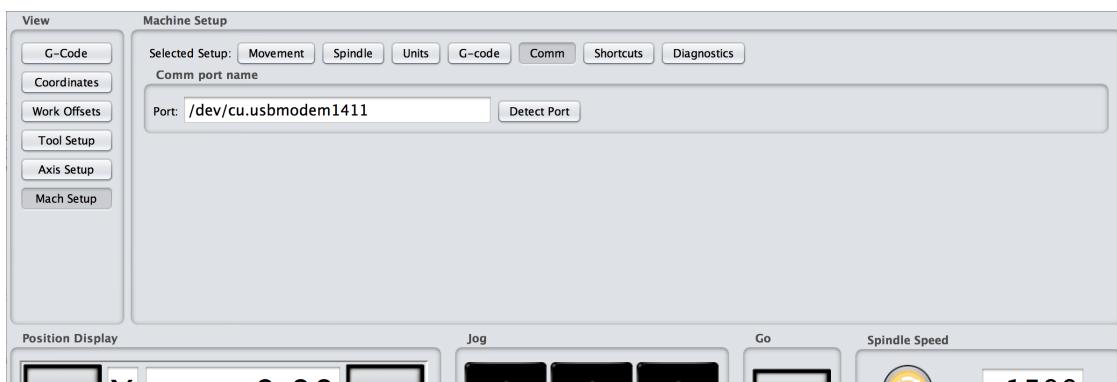


Figure 6.1: The communication port setup screen

Click on the entry field labeled "Port:" and type in the port name. To find out the port name you need to follow instruction in the following section for your operating system, or you can use the Auto Detect feature, see section ??.

Note that the serial port name changes if you plug the USB cable to a different port in your computer. In some version of Windows it sometimes seems to change for no apparent reason.

To test that the communication works click the 'MACH' button at the bottom of the screen.

If everything is correct the very wide entry field just above the bottom row of buttons should show the TOAD4 firmware version number. If it shows 'Controller: Not Connected' then EazyCNC was unable to open the serial port connection.

Don't forget to click 'SAVE' so that the port name is saved and you do not need to set it again!

6.2.1 Using the Auto Detect port function

Note that on Windows you first have to install the driver, see below section ??.

To Auto Detect the port you need to have the TOAD4 powered on and the USB cable plugged into the computer; next click the 'Auto Detect' button, and then un-plug and re-plug the USB cable pausing for about one second in-between.

If the port is successfully detected a dialog will inform you and ask if you want to use that port, click Yes and the port name will be automatically entered in to the 'Port:' field.

If the port is not found a dialog will inform you and you can try again more slowly or try to use the manual methods outlined in the following sections.

6.2.2 Installing the Windows Serial Port Driver

As said above all operating systems, even Windows, come with the serial port driver installed.

Except that on Windows you have to tell the Windows where it is and how to use it!

Installing the driver is a bit complicated process.

Further more the details of the driver installation depend on the Windows version.

Following is a brief outline how to do the driver installation on Windows 7. For better and detailed instructions I refer to google.

You need a file that contains the instructions for Windows on how to locate and use the driver it already has and you need to tell Windows where that file is.

The file is called

eazycnc.inf

and you can download it from the same place you downloaded the EazyCNC application file.

To tell Windows about this file plug in the USB cable from your TOAD4 to your computer make sure TOAD4 is powered on.

Then Go to the Windows Start -menu and from there select Printers and Devices.

From the window that appears select (somewhere at the bottom) the 'unknown device' that represents the TOAD4 at this stage, right click on it and select Properties.

From the dialog that appears select the Hardware-tab, then in that tab select, from the Device Functions list, the Driver and click Properties.

From the dialog that appears, select the Driver-tab and from that tab select Update Driver...

From the dialog that appears select ‘Browse my computer for driver software’, then click the ‘Browse...’ button and browse to the directory/folder where you have the ‘eazycnc.inf’ file and click next.

This should install the driver for you.

Once the installation is complete go back to the Printers and Devices window and make a note of the COM port name that is now show in place of the unknown device.

Enter the COM port name in the port field in the ‘Mach Setup’ screen.

6.2.3 Finding out the virtual serial port name on Mac OS X

In Mac you need to go to the Terminal program which is in the the /Applications/Utilities folder or you can use Spotlight to locate it.

Plugin TOAD4 and make sure it is powered on.

Open the terminal and type the following:

```
ls /dev
```

this will produce a long list of devices, from this list you need to look for something like:

```
cu.usbmodemNNNN
```

where NNNN is a series of digits.

If you find several, unplug TOAD4 and give the ‘ls /dev’ command again, the one device that disappeared is the one you are looking for.

Make a note of the device name and enter it in the port field in the ‘Mach Setup’ screen.

6.2.4 Finding out the virtual serial port name in Linux

In Linux you need to go to the terminal which typically can be found from the Accessories menu of the Desktop application.

Plugin TOAD4 and make sure it is powered on.

Open the terminal and type the following:

```
ls /dev
```

this will produce a long list of devices, from this list you need to look for something like:

`tty.ACMinnn`

where nnn is a series of digits.

If you find several, unplug TOAD4 and give the 'ls /dev' command again, the one device that disappeared is the one you are looking for.

Make a note of the device name and enter it in the port field in the 'Mach Setup' screen.

6.3 Setting your work Units

Now that the communication works and before we set up the motors and axis we want to select the length units you are comfortable with.

To do that go the the 'Units' screen, Figure 6.2.



Figure 6.2: The Units setup screen

EazyCNC supports working in millimeters or inches. All displays and entry fields will always show values in the selected units and accept values in these units. You can change the units at any time and it will not confuse EazyCNC, so you can use millimeters to set up things and then switch to inches; however even if EazyCNC will not get confused, you may, so it is best to select one system of units and stick with it.

Note that regardless of the units selected here the G-code file can contain coordinates that are expressed in millimeters (G21 mode) or inches (G20 mode), and this is perfectly fine, as long as the correct G20/G21 mode is specified in the G-code file.

Units -popup menu

There are two options.

'mm' – with this setting all the entry fields and DROs display and interpret values in millimeters.

'inch' – with this setting all the entry fields and DROs display and interpret values in inches.

Format -entry field

With this entry field you can control how numbers in the entry fields and DROs are displayed.

The main usage is to control the number of decimals you want displayed, to do that just enter '0.' followed by as many '0' characters as you want decimals.

For example with inches it is probably preferable to use three decimals to see the 'thous' so enter '0.000' in this field.

For the nitty gritty details of formatting those who are interested can google up the 'javadoc' for DecimalFormat.

6.4 Configuring Motors and Axes

To configure the motors and axes go to the 'Axis Setup' screen, see figure 6.3.

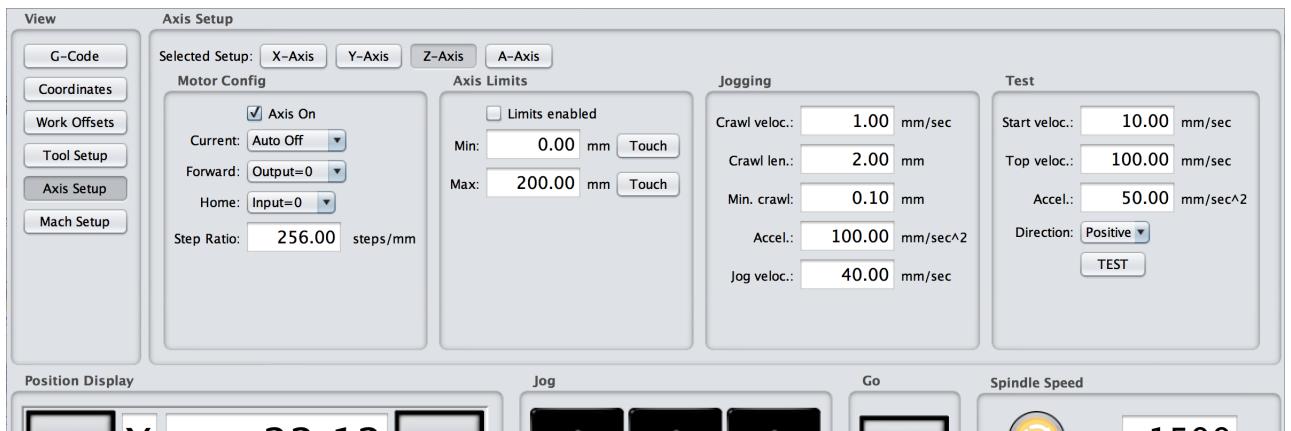


Figure 6.3: The Axis setup screen

On the top of the 'Axis Setup' panel you see four buttons that correspond to the four axes that TOAD4 can control. By clicking at those buttons you control which axis parameters are shown on the panel.

There are three group of parameters for each motor and axis.

6.4.1 Motor Config -panel

There are five parameters for each motor.

Axis On -checkbox

Axis On -parameter controls whether the G-code commands control that axis/motor or not. If the motor is not controlled by the G-code (Axis On checkbox is not 'ticked') then that motor/axis is available for manual jogging during machining or it can be controlled with EazyCNC plugin extensions.

Typically you want to use G-code to control the motors so make sure the Axis On checkbox is ticked, but if an axis is not used (say you only have a three axis set up) then un-tick the box so that you do not need to set up the motor properly.

Current -popup menu

TOAD4 supports two different drive currents for each motor, named High and Low, in addition to which the current can be totally off. The actual motor current depends on the current measurement resistors mounted to the TOAD4 board and the jumper settings on the TOAD4 board, see TOAD4 Hardware Manual for details.

The Current -popup menu controls how the three different currents are used when driving the motors.

There are four different options.

Low – with this setting the motor current is always set to Low. You might want to use this if the High current is too much for your motor.

High – with this setting the motor current is always set to High. This provides the most 'stiff' setup but means that motors will have full current applied and 'run' hot.

Auto Off – with this setting when the motor/axis is moving or the G-code program is being executed the current will be set to High, but once the movement or machining stops the motor current will be turned off completely within two seconds.

Depending on the mechanics and usage this may not be ideal as the motors may move under external forces if there is no current and thus the axis may lose its accurate position.

Auto Hold – with this setting when the motor/axis is moving or a G-code program is being executed the current will be set to High, but once the movement or machining stops the motor current will be set to Low within two seconds.

This is often the most desirable motor current setting as full current and force is used during machining but the current and heat is reduced when the motors are not being used.

Forward -popup menu

The Forward popup menu controls whether the direction output on the TOAD4 board is 1 or 0 when the motor is driven forward.

Forward means that the coordinates of the axis are increasing.

There are two options.

Output = 0 – with this setting the (internal to TOAD4 board) DIR signal is set to logic zero to when the axis/motor is driven forward.

Output = 1 – with this setting the (internal to TOAD4 board) DIR signal is set to one zero to when the axis/motor is driven forward.

You need not to care about zeros or ones, just make sure this setting is right! When you press the axis jog buttons (+X,+Y,+Z or +4) the motor should be running in the direction that you have designated as the increasing coordinate for that axis.

If the motor runs in the wrong direction just change the setting in this popup.

Home -popup menu

TOAD4 supports one home/reference position switch input for each axis.

The Home popup menu controls weather the REF input on the TOAD4 board is 1 or 0 when home/reference is switch is active.

There are three different options.

None – with this option the REF input is ignored and when you press the HOME button no movement happens, only the DRO for that axis is reset.

Input = 0 – with this setting EazyCNC expects that the REF input is a logical zero (closed) when the reference switch is active.

Input = 1 – with this setting EazyCNC expects that the REF input is a logical one (open) when the reference switch is active.

Again you should not care if the signal is active or non-active, zero or one, just make sure it works for you. If, when you press the 'HOME' button, the axis does not begin to move towards the reference switch the setting of this input is wrong. Note that you should first ensure that DIR signal is correctly configures, see previous section.

When you press the 'HOME' button for an axis EazyCNC will drive that axis until it finds the home/reference position at which point that axis DRO is automatically reset.

The way this works when you press the 'HOME' button is that if the the REF input is active the axis is driven to the positive axis direction until the signal becomes non-active. If the the signal is non-active to begin with then the axis is driven in the negative direction until the REF signal becomes active and then to the positive direction until it becomes non-active again.

This ensures that even though there is some backlash in the mechanism or hysteresis in the switch the mechanism position will always be correct.

You do not need to use a reference switch but by having one for each axis allows the system to know its absolute physical position which in turn makes it possible for EazyCNC to guard the movements against the physical limits of your system preventing crashes.

Using reference switches it is also possible to continue machining after a sudden loss of power because the absolute axis positions can be re-covered by homing the axes.

Note that the home or reference switch is no substitute for limit switches that should be installed at each end of the movements and wired to act on the emergency stop system.

The optimal placement for a reference switch is around the middle of the axis travel, but this requires that the switch is so configured that the REF signal is always on or off depending on which side of the switch the 'axis' is; it should not be possible to drive the axis 'beyond' the switch.

Step Ratio -entry field

This entry fields tells EazyCNC how many steps it takes to move the axis a unit (mm or inch) length. We call this value the step ratio.

Note that this is not an integral value and it should be entered with as many significant digits as required to achieve the desired accuracy. As rule of thumb use at least six significant digits in calculations and entry to achieve 0.01 mm accuracy over 1000 mm axis movement range.

To calculate this value you need to know following:

- *mode*, a factor dependent on TOAD4 step mode
- *steps*, the number of steps per revolution for the motor
- *pitch*, the axis movement per motor revolution (including possible gearing)

Then you calculate the step ratio as follows:

$$step_{ratio} = \frac{mode * steps}{pitch} \quad (6.1)$$

The step mode depends on the M1 and M2 jumpers for each motor on the TOAD4 board.

See Table 6.1 for the jumper labels for each motor and Table 6.2 for the jumpers that need to be installed to get the desired step mode and the mode value to use in Equation (6.1)

Table 6.1: Motors versus configuration jumpers

Motor	M2	M1
X	U29	U25
Y	U30	U26
Z	U31	U27
4	U32	U28

The driver chip supports four different step modes: full step, half step, fine step and micro step. Fine step provides eight intermediate sinusoidal current values for each full step and micro stepping provides sixteen intermediate current values.

Typically, micro stepping is preferred for its smooth ride, but sometimes speed requirements dictate the use of half or even full step.

Table 6.2: Jumpers versus Step Mode

M2	M1	mode
-	-	1 Full Step (2 phase)
-	X	2 Half Step (1-2 phase)
X	X	8 Fine Step (2W1-2 phase)
X	-	16 Micro Step (4W1-2 phase)

'-' indicates no jumper installed

'X' indicates jumper is installed

(value) refers to TB6560 excitation mode, see data sheet for details

Above may feel a bit complicated so an example maybe useful.

Most stepper motors have 200 steps or step angle of 1.8° so we have:

$$steps = 200(steps/rev)$$

For this example we assume that we want to run X-axis motor at 'Half Step' mode so from Table 6.2 we see that we need to have jumper M1 installed and from Table 6.1 we see that for X-motor M1 jumper is labeled U25 (don't forget to see the errata for the TOAD4 board, some of the jumper labels in the early board are wrong).

While looking at Table 6.2 we also note that 'mode' value for 'Micro Step' is 2 so we have:

$$mode = 2$$

To make this more interesting and life-like let's suppose we use a lead screw to move the X-axis and the screw has a pitch of 3 mm/revolution and that we use a toothed belt to drive it with a 16 tooth pulley on the motor axis and 45 pulley on the lead screw, so we have:

$$pitch = 3 * \frac{16}{45} = 1.066666(mm/rev)$$

Putting it all together we have

$$step_ratio = \frac{2 * 200}{1.066666} = 375.000(steps/mm)$$

Now would be good time to check how fast we can move the X-axis.

The maximum theoretical step rate is about 11 kHz, for jitter and other reasons the maximum recommended pulse rate is about $\frac{1}{5}$ of that, say 2000 pulses/sec. You need to divide this by the *mode* factor we looked up above so in our example the maximum step rate is 1000 steps/sec and so our max speed is

$$\frac{1000}{375} \approx 2.6 \text{mm/sec}$$

Which is not very fast so maybe we should have geared up instead of down!

6.4.2 Axis Limits -panel

EazyCNC can guard movements against set limits to prevent crashing the mechanism.

This is especially useful in Jogging where it is too easy to run too fast to an end of an axis.

However the limit checking is not fool proof and it depends on the operators (that's you!) diligence to work properly. If you for example put a large cutter into the spindle chuck but don't tell EazyCNC about it or move the axis manually by turning handles or forgot to 'home' the axes there is nothing EazyCNC can do about it.

A 'bad' a G-code move may still crash the machine and you need to visually satisfy yourself using the simulation mode that this will not happen.

Also worth remembering is that the limits checking does nothing to prevent crashing against the workpiece, fixtures or other obstacles.

The actual movement limits are based a fixed coordinate system independent of all the different G-code coordinate systems. The limits are expressed in the current unit system unscaled and unaffected by any G-code coordinate system transformations.

The origin of the limits coordinate system is at the home/ref switch position, so if the home/ref switch is not used you should not enable the limits because the position is physically undefined.

If you want to use the limits you must remember to 'home' all axes by pressing the 'HOME' buttons if the TOAD4 has been power cycled (it is TOAD4 who maintains the coordinates so it will loose track of the position if it is turned off).

Also worth remembering is that if you use the limits you should have the motors energised at all times otherwise the motors will 'lose' their positions, so don't use the 'Auto Off' current mode.

Limits Enabled -checkbox

Limits Enabled -checkbox controls weather EazyCNC enforces the limits for the axes or not.

Min/Max - entry fields / Touch -buttons

These entry field controls the minimum and maximum coordinates allowed for the axis, expressed in the current unit system.

The easiest way to set the limit is to disable the limit checking and carefully 'jog' the mechanisms to each end of the movement and press the corresponding 'Touch' button which will then set corresponding limit based on the current axis position

Remember to 'home' the axis before setting the limits and don't forget to enable the limits once you have set them.

And don't forget to verify your limits!

6.4.3 Jogging -panel

Jogging refers to manually moving the axes with either the 'jog buttons' or with the joystick.

The settings in this panel control the details of jogging.

When you 'jog' an axis, the axis first moves slowly ie crawls. This goes on for some short time after which the movement accelerates until it reaches the jog speed. Jogging then continues at that speed as long as you keep jogging or you hit the end of movement, after which the speed decelerates to a halt.

In many CNC mechanisms the axes are not created equal, some motors are by necessity stronger than others and thus the desired jogging characteristics are different and you want to set them individually for each axis.

Crawl Velocity -entry field

This entry field controls the initial ie crawl speed of jogging, you typically want to have this pretty slow.

Crawl Length -entry field

This entry field controls how long a distance the crawl will go until the acceleration starts if you keep on jogging.

Min Crawl -entry field

This entry field sets a minimum crawl distance, ie the axis will always move at least this much even if you jog very briefly.

Acceleration -entry field

This entry field control the acceleration/deceleration rate of the movement, you probably want to have this as high as possible but not so high that there is a risk of stalling the motor or the motor skipping steps.

Only experimentation can find a correct value for this, start with a small value and first find the maximum jog speed before you try to maximise the acceleration.

EazyCNC has a motor test function, see section ??), to help you to determine the limits of your motors and mechanisms.

Jog Velocity -entry field

This entry field controls the jog or maximum speed the axis will run when you jog it. You probably want to have this set as high as possible but not so high that there is a risk of stalling the motor or the motor skipping steps.

Only experimentation can find a correct value for this, start with a small value and find the maximum at which you can jog back and forth to a given DRO value watching that the tool tip hits exact same position every time.

A manual way to check that the motor has not lost any steps is to 'home' the axis and see that as the axis approaches the home position the DRO will not suddenly jump to the reset value when the reference switch is reached, this is not wholly accurate if the motor only loses a small number of steps but typically it is all or nothing with stepper motors.

6.4.4 Test -panel

The controls in this panel can be used to test and find out the maximum acceleration and velocity for an axis.

When you click the 'TEST' button EazyCNC will perform a test movement on that motor/axis and report the accuracy if you have a reference switch installed.

If you don't have a reference switch then you will need to use a Dial Test Indicator or some such to measure the accuracy.

6.4.5 Pre-requisites

Note that this test can damage your machine if not performed carefully and as intended.

The test run requires free travel of 25 mm or one inch plus the distance it takes to accelerate from the start velocity to the top velocity and back.

To use this test you must have the reference switch located at least 30 mm away from the

negative end of the axis movement.

If you don't have that 30 mm spare travel you run the risk of hitting the the end of the axis movement range on the return leg of the test run. It is acceptable to temporarily move the reference switch to a suitable location for this test or use a temporary switch if you so desire.

If you don't have a reference switch then you need to ensure that you start the test from a position on the axis from which there is enough free travel on both sides of the starting position.

6.4.6 The Test run

When you click the the 'TEST' button the test movement is performed as follows.

First EazyCNC 'homes' the axis/motor ie it moves slowly towards the reference switch and then just out of it.

EazyCNC makes an internal note of this step position.

Next the axis/motor starts to move into the positive (or negative, depending on which direction you have selected) axis direction and accelerates at the given acceleration until the motor reaches the given test speed. The movement continues for 25 mm (about 1 inch) and then it decelerates back to stand still.

Then the axis is homed again and a note of the step position at which the reference switch is detected is noted down.

The difference between the two home positions noted down is reported as the accuracy or repeatability of the movement at the given acceleration and movement velocity.

A positive value indicates that steps were lost on the way out ie during acceleration or high speed movement. A negative value indicates that steps were lost on the way back ie slow movement; this should not really ever occur.

A small non zero value is acceptable or even expected as it is unlikely that the system can be absolutely accurate all things considered, but repeated tests at given acceleration and velocity should show consistently similar values.

To guarantee that the test is valid for machining conditions the acceleration is performed step wise at the current machine Update Period just as it will when take place when executing G-codes.

Therefore you need to remember that if you change the update period it is good to re-run the test for each axis, especially if you are running close to the speed and acceleration limits of your system.

Start Velocity -entry field

In this field enter the start velocity for the test, this should be a low speed at which the motor is more or less guaranteed to work without losing steps or stalling.

Top Velocity -entry field

This is the velocity you want to test for so keep adjusting this and retesting until you have maxed out your system.

Acceleration -entry field

This is the acceleration you want to test for so keep adjusting this and retesting until you have maxed out your system.

Direction -popup menu

This selects weather the test movement direction will be along the positive direction or negative direction from home/ref position.

This is the acceleration you want to test for so keep adjusting this and retesting until you have maxed out your system.

6.5 Setting the Movement limits

The parameters in this screen Figure 6.4 tell EazyCNC how fast it is safe to accelerate and run the motors, how often the motor position and speed should be updated and how accurately you want EazyCNC to follow the tool path described by the G-code file.



Figure 6.4: The movement setup screen

Next to the individual axis/motor parameters these are the most import parameters to carefully set as if you set the speed and acceleration to too high values the steppers will lose steps

and the accuracy is ruined or the motors will completely stall.

On the other hand you will want to have the values as high as reasonable so as not to waste time in machining and with some cutters like plasma torches even the dimensions and quality of cut are dependent high enough speeds.

Note that these settings here set the maximum values, G-code programs specify the actual value which can be lower or higher than what you specify here. If the G-code specifies a lower value then that applies but if the G-code specifies a higher value then what you have set up here applies.

The only way to find out the maximum acceptable value you can use is to try progressively higher values and verify the accuracy of the motions for each trial.

Note that the values here are common to all axes. See section ??) on how to do this for each axis. Once you have found out the maximum velocity and acceleration for each axis, pick the minimum of those values and use that here as the maximum velocity and acceleration for machining.

Velocity -entry field

Enter the minimum of the maximum velocities your motors can handle.

Note that it is ok to use/have too high feed rate in G-code (the F-word) as EazyCNC will automatically limit the feed rate to the maximum velocity you have set here.

Acceleration -entry field

Enter the minimum of the maximum accelerations your motors can handle.

Path tolerance -entry field

This field tells EazyCNC how accurately during machining it should try to follow the tool path described by the G-code program.

If you enter a value of zero here then the tool path is accurately reproduced but this necessitates that the tool comes to a complete stop between cutting movements (G-codes G1,G2 and G3).

This is because to have the tool follow the path absolutely without stopping at corners would require infinite acceleration and that strong motors are hard to find.

If you enter a non-zero value then EazyCNC will try to follow the prescribed tool path to within the specified limit but using the leeway given by the tolerance to allow continuous movement and not stopping between cuts.

Rapid tool positioning (G0) always stops at the end of the movement so this parameter does

not apply.

If you use EazyCNC with a plasma torch it is important to try to minimise the speed variations as the cut width depends on the travel speed of the torch so use as large path tolerance you can accept.

6.5.1 Update Period -entry field

Setting a suitable value for the Update Period is also critical.

This value depends on the speed your computer. A faster computer allows for a faster update period, however there are limits on how fast it is acceptable to update the speed and position into TOAD4.

USB limits the update period to a minimum value (maximum update speed) of 1 msec, but typically you should aim to a value of 10 to 20 msec on a modern PC hardware. Note that this has nothing to do with step rate because we are transferring position values to the TOAD4 and the actual steps are generated on the TOAD4 board.

To understand how the update period affects things here is brief description.

TOAD4 maintains a queue of movement commands so that small pauses, interruptions or hiccups in the computer won't affect cutting movements.

If TOAD4 runs out of movement commands ie the queues run empty which happens if the computer does not send new commands fast enough on average then the cutting movements will stop until more commands arrive.

At best this is not desirable as an interrupted cut can leave a mark in the workpiece and at worst the accuracy may be lost if the movement was at such a high feed rate that the motor cannot be accurately stopped from such speed.

The queue capacity is 16 commands, so an update period of 20 msec means that there are commands for 16×20 msec or 320 msec and the system can tolerate a pause, such as Java garbage collection, for that length of time. So longer period allows for longer pauses and hiccups in the host computer system.

On the otherhand TOAD4 can only change speed at the interval of the update period so when the speed is changing, like when the machine is cutting a circular path, the speed is always partially 'wrong' for the duration of the update period.

So longer Update Period will result in an increased positional error, which fortunately is not accumulative.

The upper theoretical limit for such an error is 'feed rate x update period', for example if you are cutting at 1200 mm/min which is 20 mm/sec and your update period is 20 msec then the maximum error caused by the update period is $20 \text{ mm/sec} \times 0.02 \text{ sec}$ which is 0.4 mm. This would probably be un-acceptable for milling but such high feed rates when milling are rare and for plasma cutting where such high feed rates are common this is in the same ballpark as cutting accuracy anyway. Besides that is a worst case error unlikely to happen in practice.

6.6 Spindle setup

Whenever the spindle has been turned ON with M3 or M4 -code TOAD4 will output a voltage from the 'SPINDLE SPEED' output proportional to the S-word value.

The output voltage is relative to the voltage fed into the '+10 VREF IN' input to the TOAD4 board, the intention is to grab that voltage from the VFD from which it is typically readily available for just this purpose.

The output voltage is calculated as

$$U_{out} = \frac{Sword - MinSpeed}{MaxSpeed - MinSpeed} * Uref \quad (6.2)$$

where

Sword = The S-word value from the G-code program

Uout = 'SPINDLE SPEED' output voltage in TOAD4

Uout = 'SPINDLE SPEED' output voltage in TOAD4

MinSpeed = Value entered into the 'Min Speed' entry field

MaxSpeed = Value entered into the 'Max Speed' entry field

if above should result in a value outside the range 0..100% of *Uref* then it is clamped to that range.

Typically a VFD has parameters that are used to set the minimum and maximum spindle speed and it is precisely those same speed values that you should enter into EazyCNC here.

Figure 6.5.

6.6.1 Min Spee -entry field

Enter into this field the rpm value which corresponds to the 0 Volt voltage at SPINDLE SPEED output and VFD speed input.

6.6.2 Max Speed -entry field

Enter into this field the rpm value which corresponds to the 10 Volt (100%) voltage at SPINDLE SPEED output and VFD speed input.



Figure 6.5: The spindle setup screen

6.7 G-Code Options

G-code dates back to 1960s with the final revision RS274D approved in 1980. Over the years manufacturers of CNC system have added extension and variations to the standard.

EazyCNC tries to accommodate a common subset of the most popular systems out there by allowing you to fine tune some details of the G-code interpretation to match your G-code program.

You can change them in the G-code screen, Figure 6.6.



Figure 6.6: The G-code setup screen

6.7.1 Incremental IJK -checkbox

If this check box is ticked then the I,J and K words in the arc cutting G2 and G3 commands are interpreted relative to the start point of the arc.

If you see a lot of large erroneous arcs in the tool path graphics panel when your are previewing your G-codes then you can be pretty confident that this tick box is in the wrong state.

6.7.2 Incremental XYZ -checkbox

If this check box is ticked then the X,Y and Z words in the movement commands G0,G1,G2 and G3 are interpreted relative to the previous X,Y or Z words/positions.

6.7.3 G4 P in msec -checkbox

If this check box is ticked then the P-word value in the G4 dwell command is interpreted in milliseconds instead of seconds.

If the execution of G-codes seems to stop at G4 commands you can be pretty sure that this check box is not ticked but should be.

6.8 Shortcuts setup

EazyCNC is really designed to be used on a PC or tablet computer with a touch screen. However you can use most of the functions with convenient single key keyboard shortcuts on a conventional keyboard or with a gamepad function keys and joystick.

The key assignments are fully user definable so you can configure these as you best please.

To examine or change the key assignments go to the Shortcuts setup screen, Figure 6.7.

On the left side column you see the keys and on the right side column the corresponding assigned functionality.

To change the key, click on the left side column at the key you want to change; this will popup a dialog prompting you to press the new key you want to assign for that functionality.

To change the functionality click on the right side column of the key assignment you want to change and from the popup pick the functionality you want to assign.

To create a totally new keyboard shortcut scroll to the bottom of the list and click at the 'New Shortcut' button at the bottom of the left column.

To delete a shortcut click on the left column on the shortcut you want to delete and from the dialog that pops up select 'Delete'.

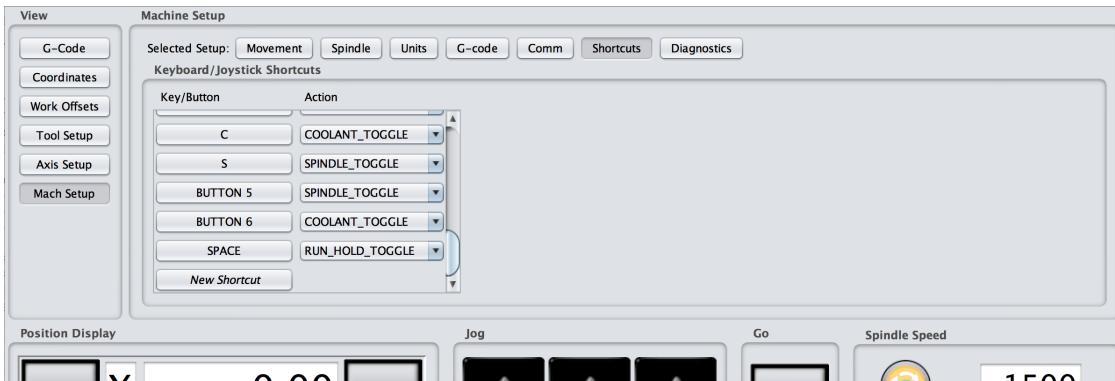


Figure 6.7: The Shortcuts setup screen

6.9 Diagnostics screen

In addition to displaying a bunch of system related information the Diagnostics screen, Figure 6.8, can be used to get insight into how consistently EazyCNC is able to communicate with the TOAD4 controller.

When the controller is connected (the 'MACH' button is active) the histogram shows in real time the distribution of actual update periods.

The middle of the histogram represents the current 'Update Period' so the access times should consistently pile up slightly to the right of that. When EazyCNC is executing the G-code program the update speed is actually double so the pile should be concentrated on the left side of the histogram.

If the update times show up all over the place then there is something in your computer setup disturbing EazyCNC, possibly an other program or process running in the background.

As time goes by the histogram represents a longer and longer history and thus new information makes little difference to it, that is why you should clear the histogram by pressing the 'Reset' button to get fresh look at the distribution of update periods.

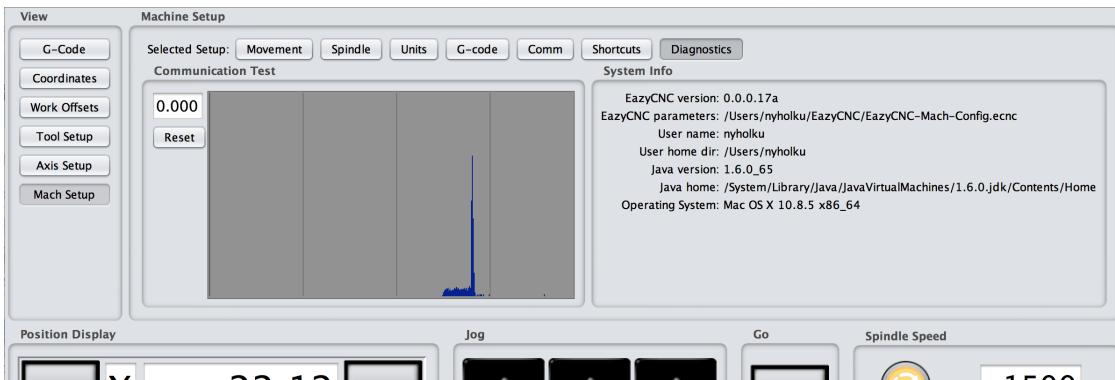


Figure 6.8: The Diagnostics screen

Chapter 7

Operating

This chapter tells you all how to operate EazyCNC to machine stuff with G-code.

G-code is covered in more detail in Chapter 9, but since the sole purpose of EazyCNC is G-code interpretation most functionality is related to G-code in some way and I cover that in this chapter.

7.1 Simulation versus Cutting Metal

EazyCNC can be in one of two operating modes, simulation mode or machining mode.

The simulation mode is handy for training or experimentation and for G-code verification before you actually machine anything. Because no TOAD4 motor controller is used or needed in the simulation mode you can install and run the EazyCNC software in any computer and use it in the comfort of your study or office.

The simulation mode attempts to be step-accurate i.e. simulate the exact movements the stepper motors will make and runs in real time (if your computer is fast enough) meaning that simulation takes as long as the actual machining will take. As this may take a long time you may speed things up when simulating by using a faster feedrate (the F-word G-code) but this will affect the small details of the cutting paths so the path generated at high speed is not 100% same as the lower speed path.

You control the operating mode with SIMU and MACH buttons at the bottom of the screen, Figure 7.1, the current operating mode is shown with the button lit.

To toggle a mode on or off click the corresponding button.



Figure 7.1: The operating mode control and indicator buttons

7.2 Status Display

At top of the panel titled 'Machine Controls' at the bottom of the screen, Figure 7.2, there is a status/error display.

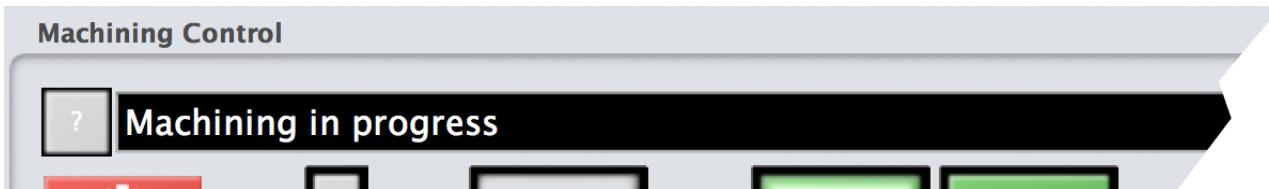


Figure 7.2: The status/error display

This display shows the general status of the system or a helpful error message in case of an error.

Most errors are transient in nature because they are the results of an operator (that's you) mistake, such as pressing the right button at the wrong time or entering an invalid value into an entry field, thus the error messages as just shown briefly accompanied with a beep sound.

If you are not able to read the error message quickly enough you can press and hold the '??' button down to re-call the last error message and read it at leisure.

During machining it is possible that the G-code contains messages to the operator and those messages are also displayed in this display.

7.3 G-code display

On the main screen the top righthand quarter of the screen is occupied by the G-code display and the associated controls, Figure 7.3.

The current G-code file is displayed there. Only one G-code program can be open at any given time, but if it calls subroutines in an other G-code file it is possible that multiple 'tabs' or files are displayed there.

The next G-code line to be executed is highlighted in blue, if there is an error in the G-code then the line containing the error is highlighted in red and the associated error message is displayed in the status/error display.

7.3.1 Loading G-code for execution

In order to execute a G-code program you need to load it into the EazyCNC first, to do that, click 'Open' button and select the G-code file to load. G-code files are pure text (7-bit ASCII) and so often have a '.txt' extension but that is not mandatory, some people and systems use '.nc'

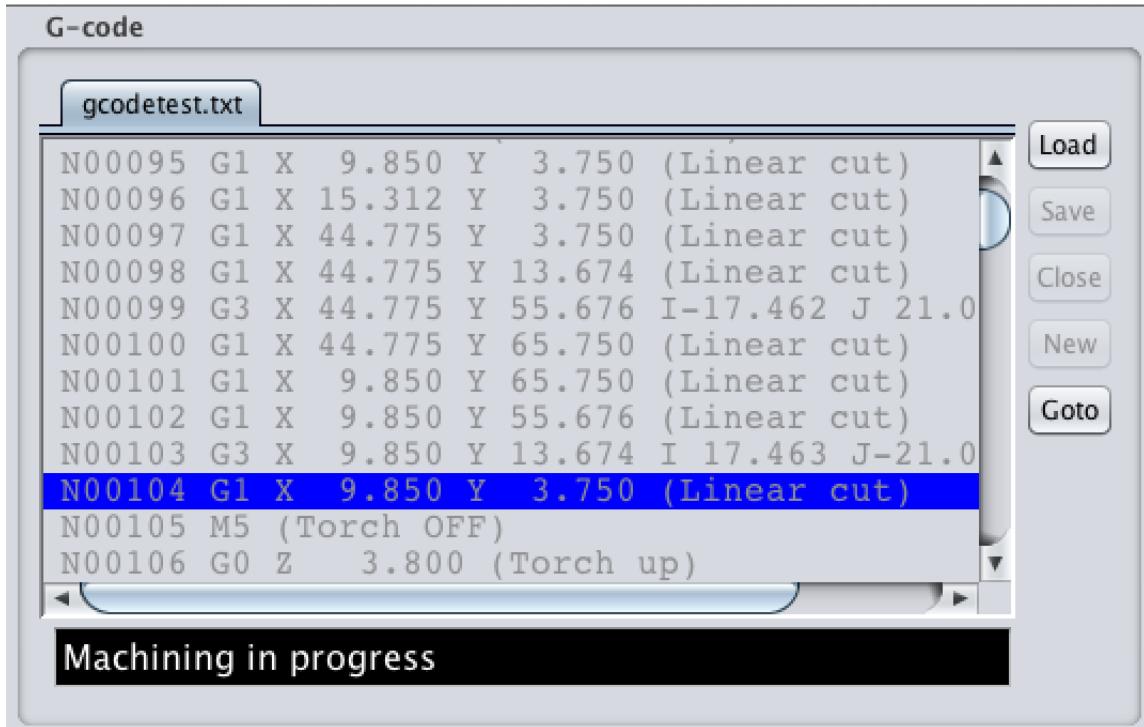


Figure 7.3: The G-code editor and display panel

7.3.2 The Goto -button

G-code programs are meant to be executed from the beginning to the end. However it is possible to start from somewhere else, for example if you need to continue execution after a black out, even if that is not recommended.

To start or continue the machining from somewhere else than the current line highlighted in blue, click on any line and then click the 'Goto' button, this will move the highlight to that line.

The caveat is that because you are skipping G-codes when you 'Goto' a specific line in the program you maybe breaking assumptions the G-code generator, be it a person or a CAM software, made when the code was written and this may have consequences.

For example a plasma cutter is typically programmed to start the cut with a piercing burn outside the actual part outline and if you skip that the arc may not be established and the metal not be pierced or the piercing may leave its mark on the part.

7.3.3 Editing G-code

It is also possible to edit and even create G-code files directly in EazyCNC, but this is recommended. EazyCNC is not a general purpose text or G-code editor, anytime you edit the code EazyCNC will go through the whole code and re-calculate the tool path and this can be slow, especially if the G-code file is very long, as it very well can be.

To edit the code ensure that EazyCNC is in the stopped state (you cannot edit the code while

machining) and just click on line you want to edit, most standard basic text editing facilities works as you would expect.

To create a new G-code file from scratch click the 'New' button.

An asterisk, '*', after a file name indicates that the file contains unsaved changes, to save the changes to the actual file click the 'Save' button.

7.4 Toolpath display

On the main screen the top lefthand quarter of the screen is occupied by the toolpath display and the associated controls, Figure 7.4.

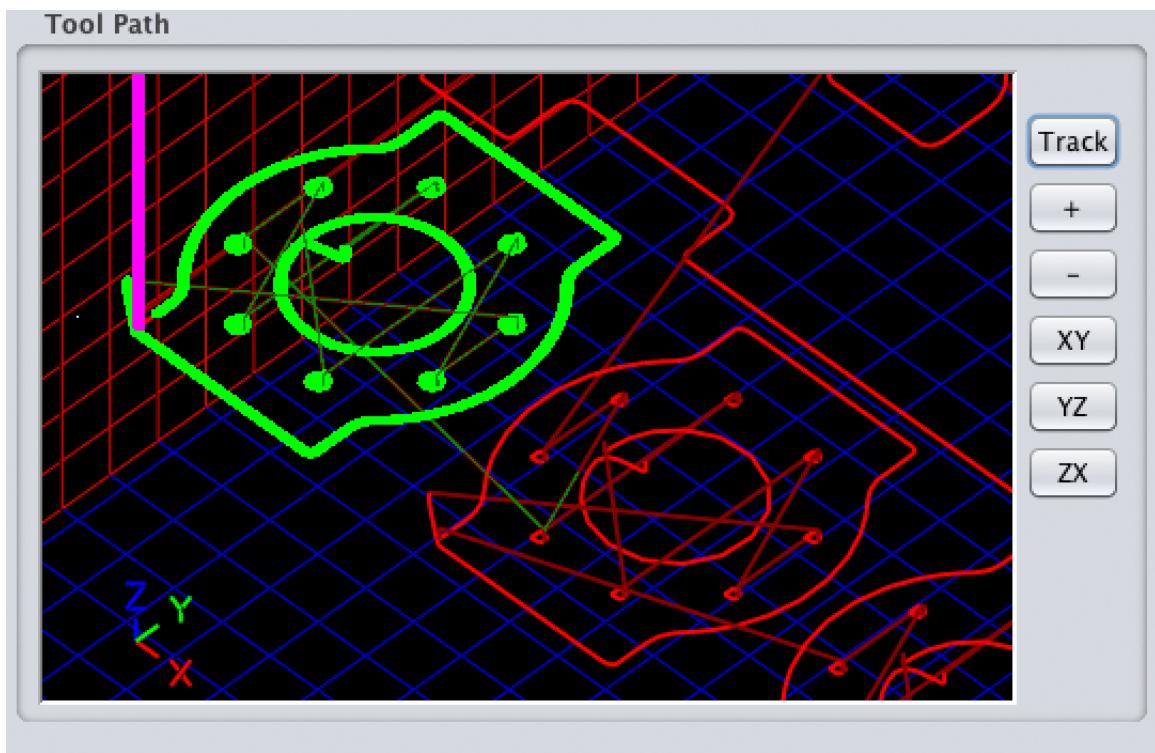


Figure 7.4: The toolpath display panel

In this panel the toolpath that the currently loaded G-code represents is illustrated in a three dimensional view.

The movement limits of the axes are illustrated as grids and the corresponding positive axis directions are displayed in associated colours.

The cutting tool is illustrated as a purple cylinder and the actual tool path is displayed as series of connected red and green lines of different intensity and line width.

The display is update in real time as the machining progresses. The toolpath machined or cut so far is displayed in green and the yet to be cut toolpath is displayed in red.

For the already machined toolpath in green the width of the line corresponds to the width

of the tool used to cut that path if the spindle was turned 'on' during the machining, otherwise the toolpath is shown as a thin line.

Whether the spindle is 'on' or 'off' for a given toolpath segment is indicated by the intensity of the color of the line, spindle 'on' is illustrated with bright red or green and spindle 'off' is illustrated with a darker shade.

When the G-code program is executing or running you can see, if you look carefully, that the green toolpath appears to precede the purple tool, this actually accurately represents the fact that the movement commands are queued in the motor controller waiting to be executed.

7.4.1 Controlling the toolpath display

If the 'Track' button is not active you can move around the three dimensional virtual space and inspect different parts of the toolpath by clicking and dragging in the toolpath display panel. If the 'Track' button is active the display is always automatically centered at the tooltip and follows it as the tool moves. You can toggle the tracking on and off by clicking the button repeatedly.

If you hold down the 'ALT' key on your keyboard while you drag with the mouse you can rotate the view. If one of the 'XY', 'YZ' or 'XZ' buttons is active then the rotation is disabled and the view is forced to be towards the corresponding coordinate plane along the 'plane normal' axis. For example if you have the XY button selected then the view is as if you were looking down from the top of the positive Z-axis towards the XY-plane.

Note that you can de-select any of the 'Track', 'XY', 'YZ' or 'XZ' by just re-clicking on it.

You can use the '+' and '-' buttons to zoom in and out to see more or less the toolpath or you can use the mouse scroll wheel.

7.5 Coordinate displays aka DROs

The coordinate displays, Figure 7.5, also known as Digital Readouts or DROs, show the tool position in the currently active coordinate system in the currently selected units.

The coordinate system can be quite complex but for most practical purpose you can just think that the DROs just show the position EazyCNC thinks the tool tip is at the moment. If the DROs says that the tool X-axis is at position 100 and the G0 -code tells it to move to X150 then the tool will travel 50 units to the right, no matter where it physically actually was.

By clicking at the DRO and typing in a number you can change the displayed value. This will not move the tool but will of course change how subsequent G-code coordinates are interpreted.

You can also zero the DRO by clicking at the 'ZERO' button.

If you press the 'HOME' button and you have a home/reference switch installed for that axis EazyCNC will drive the axis to the home switch position and then zero the DRO automatically. This establishes a repeatable absolute origin for that axis.

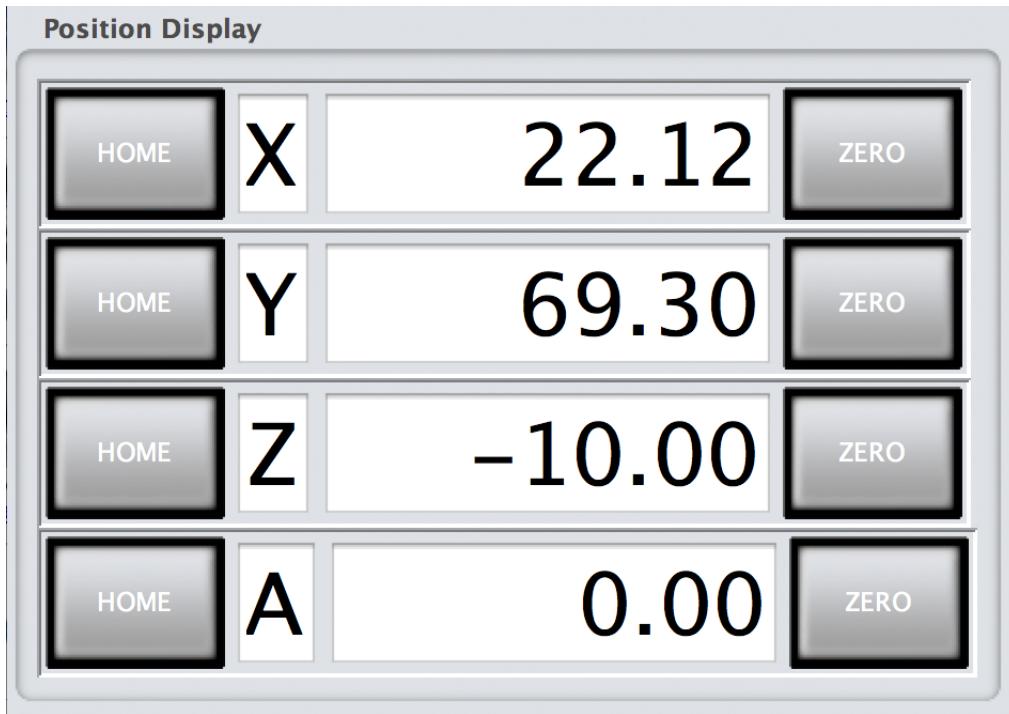


Figure 7.5: The Digital Readouts

7.6 Jogging

The Jog buttons, Figure 7.6 can be used to drive or move the motors/axes 'manually'. If you need to move the axes you should use the jog buttons and not any handles your machine possibly has for moving the axes, as if you use the manual handles EazyCNC has no way of knowing that the tool has moved and thus coordinate positions will be wrong.

Pressing a jog button will first move that axis slowly and then rapidly accelerate to the full jog speed. Pressing a jog button briefly will always move the axis a predetermined, minimal distance.

In the middle of the jog controls there is a red 'STOP' button. Pressing this will G-code execution, stop any axes movement, empty command queues and stop the spindle and coolant flow. The movement will stop abruptly so if the axis were moving at high speed it is possible that they will lose steps and thus you should consider that the system coordinates are lost if you press this button.

In that respect this button is kind of an emergency stop but this is no substitute for a real emergency stop that you need to have installed because the function of this button depends on the correct function of the software and communication between EazyCNC and to the motor controller which may not be functional if you have an emergency!

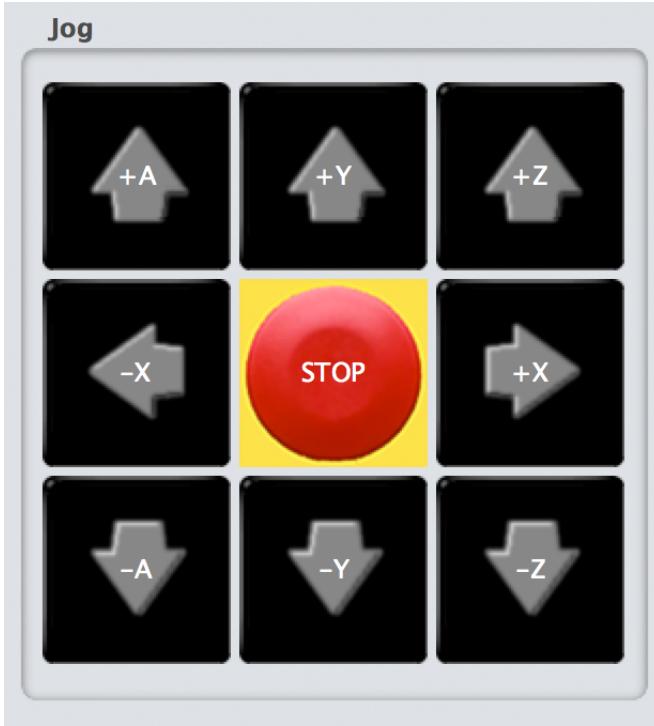


Figure 7.6: The Jog control buttons

7.7 Finding your bearings i.e. coordinates

Before you start machining you need to establish the correct relationship between the virtual part described by the coordinates in your G-code file and the work piece you are about to cut.

It is important to remember that EazyCNC does not really know where the tool is, it just keeps track of the position relying on an initial known position. Further even if EazyCNC can keep track of the tool position it knows next to nothing about the workpiece location.

There are ways to calibrate all three in relation to each other, the EazyCNC coordinate system, the actual physical axis positions and the workpiece location, but most of the time this would be an overkill.

The relationship between the physical axis position and the coordinate system only really matter for two things: if you enable the axis limits and if you need to continue machining after a power down.

In the first instance, if EazyCNC does not have the correct relationship between the physical axis position and its internal coordinate system then it will allow you to drive the axis bang against one end of the movement and will unnecessarily limit your movements at the other end.

In the second instance if you lose power, a fuse blows or the job takes so long that you can't do it all at one go, unless the system can be returned to the same axis coordinate system relationship you will not be able to continue machining accurately after the system has been powered off.

The relationship between the EazyCNC coordinate system and the workpiece really only

matters if you need to continue machining, see above, you need to remove and remount the workpiece during the course of machining, work on existing part or the work piece is very close in size to the finished part.

7.8 The easy and lazy way

The easiest way to setup the coordinate systems is just to bolt down the workpiece, jog the axes to ensure that you can reach all of the to be machined parts of it without hitting the axes limits, then drive to tool an approximate location you can name the coordinates of and set the DROs to those coordinates.

If you create your G-codes so that the finished parts smallest coordinates are at 0,0,0 i.e the part extends to the right, forward and up of the origin, preferably with a little bit of margin so that the workpiece alignment is not critical and you don't run the risk of fouling the table, then all you have to do after mounting your workpiece is to drive the tool to the bottom left corner on top of the workpiece, press 'ZERO' buttons to zero X and Y and type in the height of the work piece in the Z axis DRO and you are good to go!

For a simple system like a plasma cutter where the motors are not strong enough to cause any damage even if you hit ends of the movement above is very adequate, no home/reference or limit switches,no calibration, no nothing.

7.9 Going pro

If you need or want to align EazyCNC coordinate system and the physical axes position the easiest way is to have the home/reference switches installed. With those all you have to do after powering up the system is to press the 'HOME' buttons and the system will drive the axis to the switches and reset the coordinate positions.

For that to work accurately the switches need to perform repeatably so optical gates are the way to go.

As the axes will travel at slow speed during 'homing' it is a good idea to have the reference switch in the middle of the movement range instead of at the end as this will cut down the homing time.

It is also possible to just carefully run the axes to end stops and zero the DROs or type in a known value for that position.

Above takes care of the EazyCNC coordinate system and physical machine axis relationship.

To 'calibrate' the workpiece to machine relationship you can use whatever mechanically accurate method you want, simplest being using known sized gauge blocks or if you equip your system with a touch probe you can use that to measure the workpiece location.

7.10 Adjusting Feed Rate

??

Feedrate is controlled by the F-word in your G-code programs but can be override with the feed override controls illustrated in Figure ??.

The idea is that while the G-code programmer designs at what feedrate the part should be cut and programs that into the code it sometimes it is necessary to adjust or fine tune that while machining.

You can turn the override on and off by clicking he 'ON' button and you can adjust the override with the '+' and '-' buttons.

When the override is off the display shows the current feedrate as set by the last F-word executed, when the override is off the display shows the overriden value.

The override can be expressed either as a percentage of the F-word value or as units/minute. To change that click on the button to the right of the display field.

You can adjust the override with the '+' and '-' keys or you can type in a new value. Adjusting or setting the override value will turn on the override.

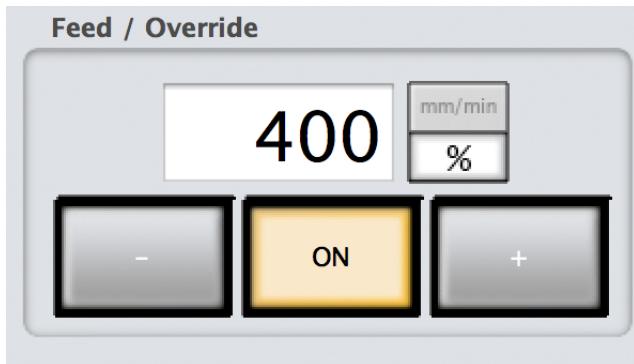


Figure 7.7: The Feed Override controls

7.11 Controlling the Spindle

Spindle is mainly controlled by your G-code program using S-word for setting the speed, M3 to turn the spindle 'on' clockwise, M4 to turning the spindle 'on' counter clockwise and M5 to turning the spindle 'off'.

You can manually control or override the G-code program commands with the spindle control illustrated in Figure ??, but as soon the G-code execution reaches next spindle control command that command will take over.

When the override is off the display shows the current spindle speed as set by the last S-word executed, when the override is off the display shows the overriden value.

You can adjust the override with the '+' and '-' keys or you can type in a new value. Adjusting or setting the override value will turn on the override.



Figure 7.8: The Spindle controls

7.12 Machining!

Finally!

Executing G-code or machining with G-code is basically very simple.

You set the operation mode, simulation or machining, with the 'SIMU' or 'MACH' button.

You load your G-code program with the 'Open' button.

Then you start the execution with the 'RUN' button, Figure 7.9.



Figure 7.9: The G-code execution control buttons

This will cause EazyCNC to interpret the G-codes line by line and control the motors and axis accordingly.

7.12.1 Pausing the machining

If you need to temporarily pause the execution, for example to clean up some swarf from the work area, you can press the 'HOLD' button.

Note that pausing the machining may slightly change the actual machined toolpath because pausing changes the speed of movements and this in turn may affect how corners are cut, see XXX.

To continue machining you press the 'RUN' button.

You can't move the tool 'manually' with the jog controls while the G-code is executing, but you can move it when the system is in the paused or 'HOLD' state. If you move the tool 'manually' EazyCNC will automatically return the tool to the position where it was before you jogged it.

In the hold state the spindle and coolant will keep running if they were 'on' when you paused the machining. You can turn them on/off manually with the respective control buttons, but they will NOT be automatically restored to their original state when you resume machining.

EazyCNC may also enter the hold state when it encounters one of the pause codes M0, M1 or M60.

7.12.2 Stepping and Reversing

Sometimes, especially when simulating and examining G-code, you may want to execute the G-code one line at a time. To do that active the 'STEP' button, Figure 7.10.



Figure 7.10: The step and reverse execution control buttons

When the 'STEP' button is active the execution of G-code will automatically enter the pause or 'HOLD' state after executing every line of G-code.

An other use for the 'STEP' feature is in combination with the 'REVERSE' button. In general you can't run G-code backwards because you can't un-machine material back to the work piece or un-flow the coolant, but sometimes you want re-run some cut or G-code line because for example the cutter broke or you had a flameout in plasma cutting in the middle of the movement.

If something like that happens you press the 'HOLD' button to pause the machining. Then you active the 'STEP' and 'REVERSE' buttons and press the 'RUN' button to run backwards as required to get to a position before where you the tool bit broke or the arc flamed out. Don't forget to deactivate 'STEP' and 'REVERSE' afterwards.

Next you stop the spindle to change the cutter and restart the spindle.

Remember to install a kill switch for the spindle/torch and always use it before touching the spindle!

Once you have the cutter changed you can continue machining by hitting the 'RUN' button.

7.12.3 Stopping

The machining will of course automatically stop when it reaches the end of the G-code file or if it encounters a stop code, M2 or M30.

If you hit the 'STOP' button the machining will stop and spindle and coolant will be turned off and the G-code is re-wound to the beginning, so that next time you hit 'RUN' it will execute from the beginning.

7.13 Setting up and managing the coordinate systems

This section discusses the coordinate systems in detail.

7.13.1 Coordinate axes

Figure 7.11 illustrates the coordinate axes. When standing in front of the machine and looking into the machine X-axis runs from left to right, Y axis from front to back and Z axis from bottom to top. This is a well established convention in CNC world.

The origin of the coordinate system where ever your home/ref switches are positioned or if you don't have a home/ref switch installed and enabled for an axis then origin is where ever that axis motor is when you press the 'HOME' switch.

The physical origin of the coordinate axis is not really relevant most of the time as you need to 'calibrate' or set your work coordinate system with respect to the the work piece anyway as explained in the next section.

Figure 7.11: The coordinate systems

7.13.2 Work/Fixture Coordinate System/Offsets

Work coordinate system, also know as fixture coordinate system or offsets, is the main thing relates the axes physical positions to the the coordinates displayed in the DROs or the G-codes coordinates.

EazyCNC supports 255 work coordinate systems, you are unlikely to ever need more than a few, most people use only one.

Mathematically a work coordinate system is simply a number for each axis that is added to the DRO or G-code value axis value to convert them to the physical axis, so basically they offset your coordinates, hence the name.

Changing the coordinate system never moves the tool or axes but it will of course change how much or to where commands move the axes after the coordinate system is changed.

The purpose of the work/fixtures coordinate systems is to allow you to set up multiple work-pieces and 'calibrate' a separate coordinate system for each and then machine them at one go instead of machining them one by one. Using multiple coordinate system is hardly ever worth the effort in a hobby installation, so I suggested you use coordinate system number one and stick with it.

You manage the coordinate systems with the Work Offsets screen, see Figure 7.12.

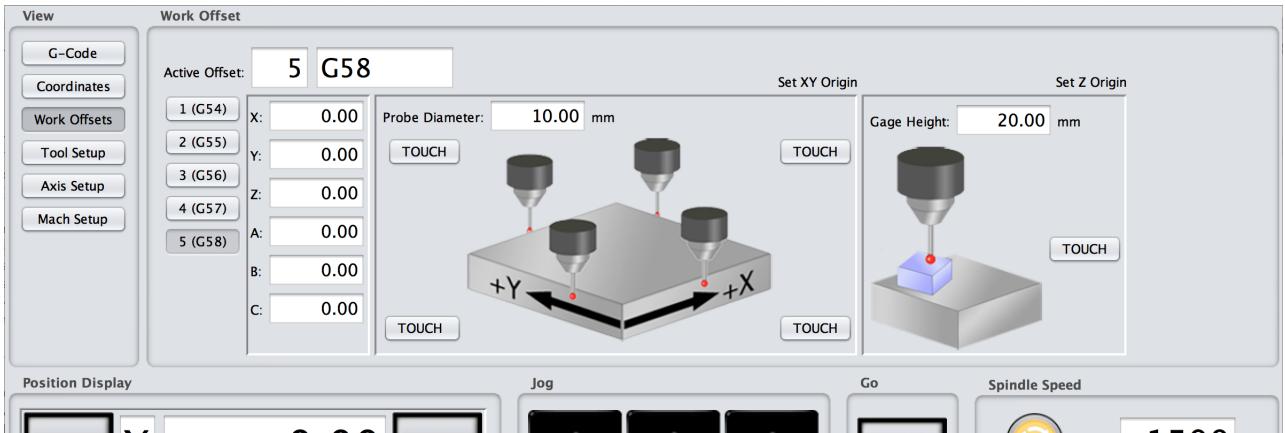


Figure 7.12: The Work Offsets screen

7.13.3 Selecting the active coordinate system

At any given time one of the coordinate systems is in effect. To switch from current coordinate system to another click on one the buttons labeled with '1 (G54)' etc or type in the number of the coordinate system you want to use into the entry field labeled 'Active Offset'. The coordinate systems are numbered from 1 to 255.

You can also, of course, change the active coordinate system in your G-code. To remind you of this and make the connection in your mind between the coordinate systems and the G-codes the corresponding G-code is shown on the button and to the right of the 'Active Offset' entry field.

7.13.4 Changing offsets/setting up the coordinates

One way to set up the coordinate system was already described in Section 7.7, that is you drive the axis to a known or desired position and set coordinates in the DROs either manually or automatically.

If you do that while you are in the 'Work Offsets' screen you will see how the offsets in the 'X:','Y:'... entry fields change. Or if you feel comfortable working with the offsets directly you can just type them into the entry fields.

A third way to set offsets and to specify the coordinate/axis origin is by 'touching' the work piece with a probe or the cutter in the tool holder.

7.13.5 Setting the XY-coordinate system origin via touching

This is illustrated and done with the controls in the 'Set XY-origin' panel. This feature assumes that you want to set XY origin to edge of the work piece, typically the left/front edge of it.

Usually a special edge finder probe or wobbler tool is held in the chuck while touching because it gives much better sensitivity and accuracy to the touching and does not mark the work piece and neither is likely to break like cutter would.

To use the touch feature simply Jog the axis to the work piece edge, ensure that the probe diameter is correctly entered into the 'Probe Diameter:' entry field and click on one of the 'Touch' buttons to indicate the edge you have driven the probe to. EazyCNC then calculates and sets the offsets so that the coordinate origin for that axis is at the indicated edge of the work piece.

7.13.6 Setting the Z-coordinate system origin via touching

This is illustrated and done with the controls in the 'Set Z-origin' panel. This feature assumes that you want to set Z origin to top milling table.

Remember that physically changing the tool or cutter requires you to re-set the Z-axis via any of the methods described here, unless you have a presettable toolholder system and keep the tool table up to date.

As setting the Z-coordinate origin via touching is dependent on the tool or cutter length mounted in the chuck you need to ensure that the correct tool is currently selected and set up as described in the following Section [7.14](#).

Touching in the Z-direction is usually done with the tool or cutter in the chuck. To use the touch feature you will need a gauge piece about 10 mm or half an inch thick.

Start by jogging down the Z-axis close to the milling table taking care not to actually hit the table with the cutter. Closer than the thickness of the gauge piece is close enough.

Next jog *up*, slowly and carefully, until your gauge piece just fits between the cutter and the table.

Now enter the thickness of your gauge piece into the 'Gauge Thickness:' field and click 'Touch'.

EazyCNC then calculates and sets the Z offset so that the coordinate origin for that axis is at the top of the workpiece.

Note that it is much safer from the tool breakage point of view to jog up i.e. first jog close to the milling table without the gauge piece in place, then jog up a little bit at a time until you can just slide the gauge piece between the table and the cutter.

7.14 Setting up and managing tool information

Certain features of EazyCNC rely on correct information of the tool i.e. the cutter diameter and length. To help you to manage this EazyCNC maintains a table of tool information for up to 256 tools.

You manage the tool table with the Tool Setup screen, see Figure 7.13.

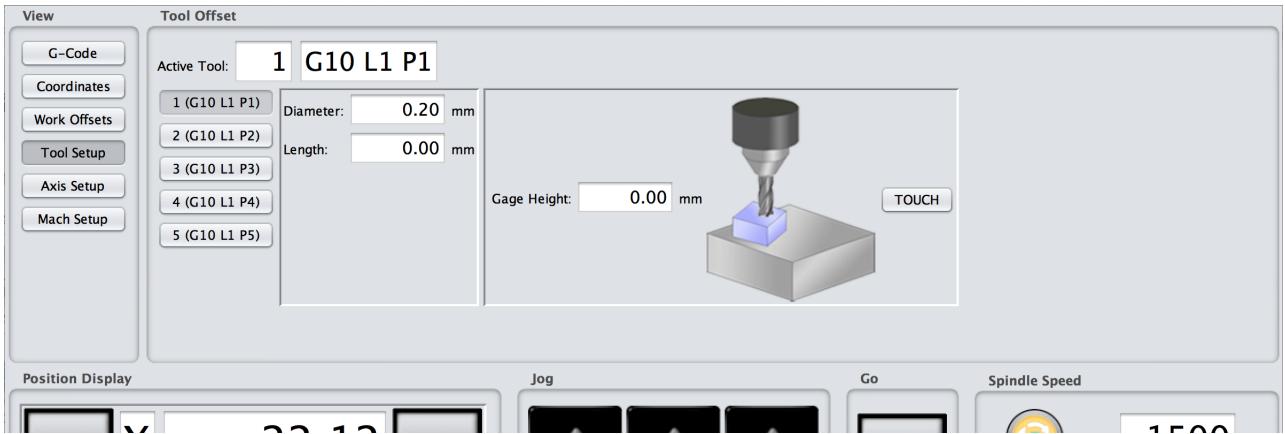


Figure 7.13: The Tool Setup Screen

The tool dimension are used in movement limit calculations and if they are not correct you may either move bang against the end of an axis movement or EazyCNC may not allow you to move the tool. This applies to both manual jogging and running G-code programs.

The tool diameter is also used in the cutter compensation calculations, and it can take advantage of the tool diameter information in the tool table or you can manage that in the G-code.

Keeping the tool information in the tool table has the advantage that the tool information can be kept up to date without touching the G-code files as cutters wear or are changed, but for some types of machines like plasma cutters where the width of the cut depends on the feed rate it maybe better managed in the G-code files themselves.

To effectively use the tool table you need a presettable toolholder system that allows you to repeatably change the tool without affecting the cutter position i.e. a system that allows you to take a tool out and put it back in later and which keeps the tool at the same height in relation to the mill spindle.

If you don't have a toolholder like that you can just set the cutter manually, hopefully with a jig or something, to a correct correct depth in the chuck and update the tool length in the table manually. With this method, as the length info is probably not accurate, you need to re-set the Z-axis coordinate system every time you change the tool.

You may also decide that keeping the tool length information correct in the tool table is too much work in which case you can just set the tool length to zero in the tool table, re-set the Z-coordinates as needed and disable the Z-axis limits, see Section ??.

You also don't have to maintain the tool diameter as this can be handled in the G-code. If

you find keeping the tool diameter correct in the tool table too much work then just set it zero and disable the XY-axis limits.

7.14.1 Setting the current tool

To set which tool in the tool table is in use, shown and manipulated on the screen click on one the buttons labeled with '1 (T1)' etc or type in a number you want to use into the entry field labeled 'Current Tool'. The tools are numbered from 1 to 256.

You can also set the current tool in your G-code program with the 'T' word. To remind you of this and make the connection in your mind between the tool table entries and the G-codes the corresponding T-word is shown on the button and to the right of the 'Current Tool' entry field.

Note that EazyCNC relies on you to keep the physical world and the software in sync. Just changing the tool number in your G-code program with the 'T' is not enough, you actually have to physically change the tool. To be able to do that you need to stop the spindle and pause the G-code execution by adding following sequence to your program.

```
M5 ; Spindle off
T 5 ; Select tool 5
M0 (MSG Load tool 5) ; Pause with a message to the operator
```

When EazyCNC encounters above sequence it sets the current tool to 5 and pauses i.e. enters the 'HOLD' state, allowing you to change the tool after which you hit 'RUN'.

Don't forget to wait for the spindle to stop and use the kill switch before touching the spindle!

If you change the tool length or diameter in the tool table when you change the tool those updated length and diameter values are *not* automatically used. Instead you need to click one of the G40,G41,G42,G43 and G49 buttons as needed if you update the tool table during machining.

7.14.2 Managing the tool diameter and length

To set or examine a tool table entry for a tool first set it as the current tool and then view or type in the tool diameter and length in the 'Diameter:' and 'Length:' fields.

You can also update the tooltable in your G-code program with **G10 L1 P_{toolno} Z _{radius} X _{length}** command; you could for example have a G-code program that updates all of your tool table.

7.14.3 Setting the tool length via touching

This is illustrated and done with the controls in the 'Set Tool Length' panel.

This feature expects that you have set your Z-coordinate system correctly, which may sound a bit odd as setting the Z-coordinates system via touching requires that the tool height is

correct! The around this problem is to first set the tool length to zero, calibrate the Z-axis with not tool in the holder and then proceed to set the tool heights via touching.

Don't forget that all this is worth the trouble if you have a re-settable toolholder, otherwise you can just always re-set the Z-axis and be done with it.

To set the tool length via touching you need a gauge piece of known thickness. The thickness is not important but about 10 mm or half an inch is good.

First enter the correct tool number into the 'Current Tool' entry field and then mount the corresponding cutter into the spindle. Then jog down the Z-axis close to the milling table taking care not to actually hit the table with the cutter. Closer than the thickness of the gauge piece is close enough.

Next jog *up*, slowly and carefully, until your gauge piece just fits between the cutter and the table.

Now enter the thickness of your gauge piece into the 'Gauge Thickness:' field and click 'Touch'.

EazyCNC then calculates and sets the Tool Length.

Chapter 8

Cutter compensation

G-codes specify the cutter or tool movement with coordinates. These coordinates specify the position of the spindle of the milling machine. Because the cutter of the milling has a non negligible size they do *not* specify the size and shape of the part you are cutting. It is the job a CAM software post processors software to turn CAD models into toolpaths that accommodate for the diameter and length of the cutter.

However G-codes specify and EazyCNC supports limited automatic cutter length and diameter compensation so that you can use the *coordinates of the part* to be machined instead of the coordinates of the spindle.

These can be useful with hand written G-code programs and with some special machine configurations such as plasma cutters and when working mainly in plane ie two dimensions.

8.1 Tool length compensation

Tool length compensation is rather simple to understand, basically it just moves the machine higher by the length of the tool. So when the G-codes specify for example Z 200 and the current tool length compensation is turned on (with G43 code) and the tool length for the current tool (set with the T word) is 50 the actually Z coordinate the machine moves to is 250.

8.2 Cutter diameter compensation

Cutter diameter compensation is not difficult to understand but there are more details to consider.

Basically if you use the diameter compensation you put the coordinates of the part your are machining in to your G-code program and when you turn on the compensation you tell EazyCNC on which side of the path you want the tool travel. The side of course depends on weather you are cutting an opening or an outline of the path and weather you are travelling the path clockwise or counter clockwise.

The cutter compensation cannot change the past, what is cut is cut, so when you turn on the compensation (G41 for left and G42 for right) the compensation will only have an effect the next move.

Therefore you need pay attention to the first move when you are planning your tool path and you need to pre-position (usually with G0 code) the tool outside of the part by at least half the tool diameter. Also consider the direction this first move-in cut will make, ideally this should be in the same direction as the first actually part outline to be cut, but for inside holes or openings this is not always possible.

8.2.1 Cutter compensation example - cutting part's outline

To get a more concrete picture of above please study the following example G-code and the Figure 8.1 which illustrates various aspects of the diameter compensation process.

```
N1 G40      ; make sure diameter compensation is off
N2 G0 X2 Y1 ; move to our starting position
N3 G42 P0.4 ; turn on compensation to the right for 0.8 unit tool diameter
N4 G1 X3 Y2 ; first move, from starting point to part perimeter
N5 G1 X8 Y2 ; cut the lower edge
N6 G1 X8 Y6 ; cut the right side
N7 G1 X3 Y3 ; cut the slanted top edge
N8 G1 X3 Y2 ; and finally back to where we started cutting the left edge
N9 G40      ; compensation is off
```

Referring to the Figure 8.1 the black line illustrates the un-compensated coordinate path ie the coordinates in the G-code file. The green lines illustrates the path of the centre of the tool and the compensated coordinates. And finally the red line illustrates the left edge of the groove the cutter actually cuts, i.e. this is the actual outline of the part we are making here, so any deviation between the black and red line (except for the initial move-in movement) is a indication of badly constructed G-code program.

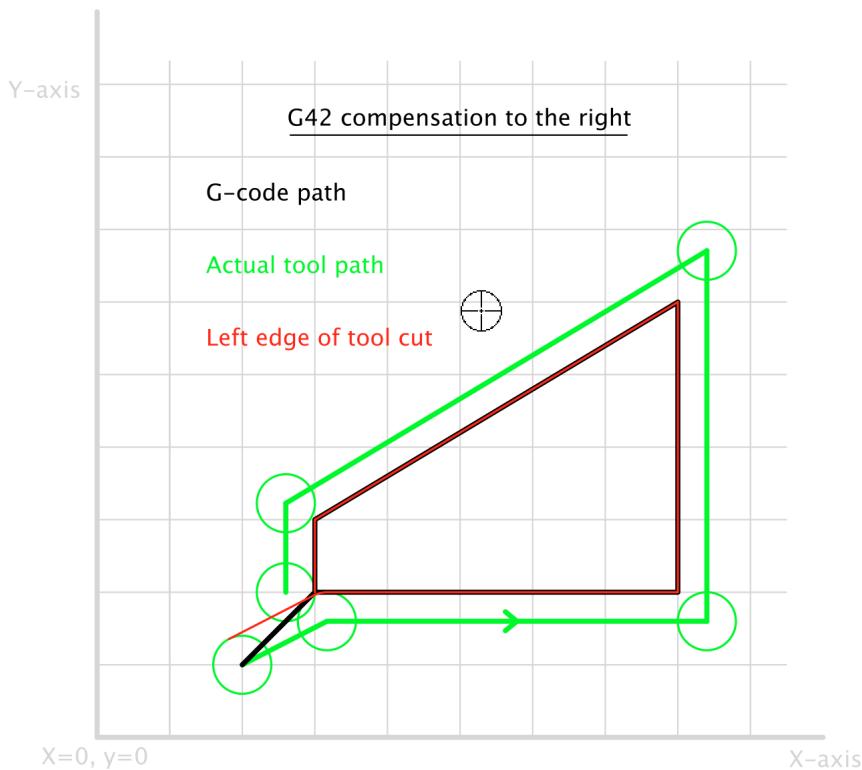


Figure 8.1: G-code path versus compensated cutter path

Overall the part outline and the cut path seem to match but if you look carefully in the blown up detail of the beginning of the cut in Figure 8.2 you see that cutting the path as specified would leave a semi circular concave notch to the part and besides, at least in theory, the part would be left hanging by a thread as the loop is not completed. These problems come from the first two actual moves (lines N2,N4 and N4) which are not parallel.

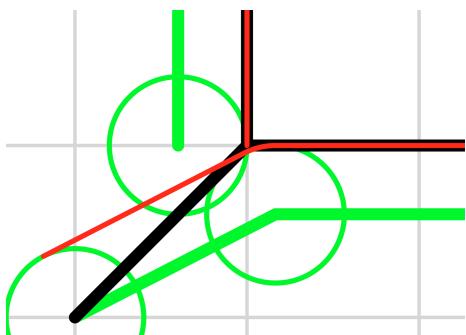


Figure 8.2: Cutter compensation detail

8.2.2 Cutter compensation example - cutting holes and openings

To illustrate cutting a hole or cutout to match the above part we can just set compensation to the left and move the starting point to the inside of the path. Following G-code does just that, the actual path is identical to the previous example only the compensation side and starting point are different. The starting point was deliberately chosen badly and the cutter is way oversize relative to the path dimensions to exaggerate the issues you may encounter when designing your paths.

```
N1 G40      ; make sure diameter compensation is off
N2 G0 X5 Y3 ; move to our starting position
N3 G41 P0.4 ; turn on compensation to the left for 0.8 unit tool diameter
N4 G1 X3 Y2 ; first move, from starting point to part perimeter
N5 G1 X8 Y2 ; cut the lower edge
N6 G1 X8 Y6 ; cut the right side
N7 G1 X3 Y3 ; cut the slanted top edge
N8 G1 X3 Y2 ; and finally back to where we started cutting the left edge
N9 G40      ; compensation is off
```

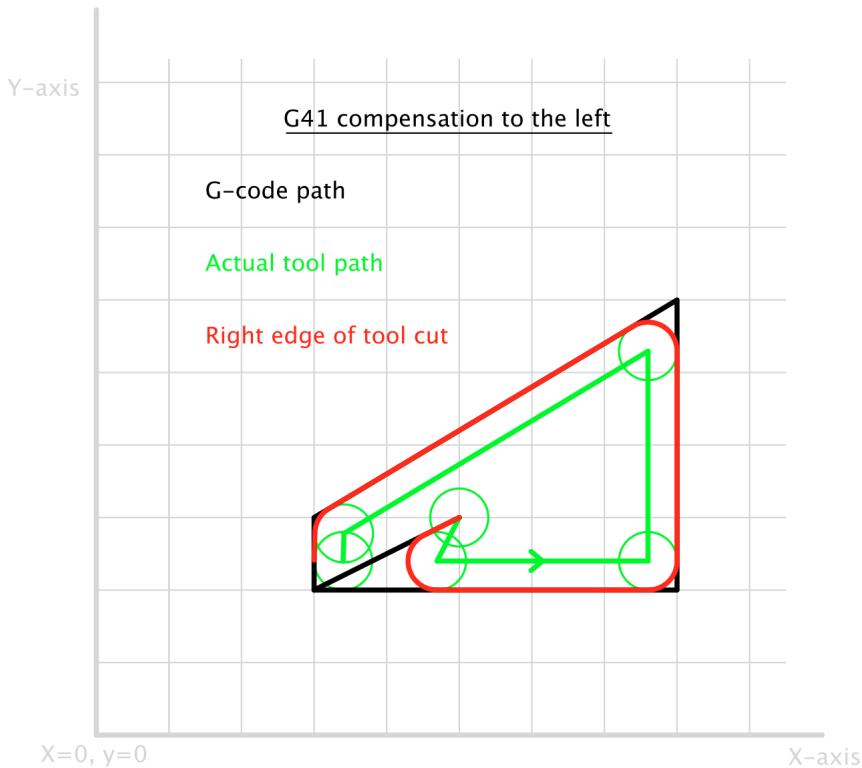


Figure 8.3: G-code path versus compensated cutter path

Chapter 9

G-code reference

G-code is ancient by computer standards, you can trace it's roots back to MIT labs in the 1950s. Wikipedia describes G-code as "a language in which people tell computerised machine tools what to make and how to make it".

Today this is not actually a good description. Rather today G-code is the standard by which computer aided design and manufacturing programs, CAD/CAM software, communicate the machining instructions to CNC machines.

G-code should not be written by hand, instead a CAD/CAM program should be used to generate it. Not only is this a lot faster but it also reduces the chances of errors and improves the quality of machining as the CAD/CAM program has a much better understanding of the machining process as a whole than the CNC machine ever can.

Therefore it is not really important to learn to write G-code, but it is handy to be able to read it to some extent and this is what this chapter tries teach you to do.

There is a EIA standard from year 1980 called RS274D that sort of defines the G-codes but over the years manufacturers have extended and interpreted it differently so that today there are many dialects.

EazyCNC tries to support a common subset of what the two most common hobby CNC controller programs, EMC2 and Mach3 support. CAM software actually produces a very simple, 'pidgin English' kind of G-code which is rather universal in order to be avoid the differences between machine. Interpretation of some of the G-codes can be configured in the Mach Setup screen, see Section [6.7](#).

9.1 The Basics

G-codes programs are text files made of lines of commands to the CNC machine. The file names often have the file extension '.nc' but this may no means required, you can use '.txt' as well.

EazyCNC reads the program line by line executing the commands.

The commands are made of single letter 'words' possibly followed by number, for example 'G0' or 'X1000'.

The words can be written in upper or lower case letters. Any number of 'space' characters can be placed anywhere on each line to improve readability; they are ignored and make no difference to the execution.

In this document an asterisk, '*' after a word is used to denote a word followed by a value, ie number or expression. These are optional arguments or parameters to the G-code in question. For example 'G0 X Y Z*' indicates that you can write 'G0 X10 Y20'. Whether parameters are optional or not is explained in the text. A '*' is used to denote a compulsory or required word.

Comments to the human reader i.e. text that EazyCNC will not try to interpret as command can be placed on lines if preceded with a semicolon ';' or enclosed in parentheses, '(' and ')'. A comment that starts with '(MSG' the rest of the comment is displayed as message to the operator in the status display, Figure Figure 7.2. Everything on after a semicolon ';' on a line is also treated as a comment.

The order of words on a line makes no difference.

No word or G-code can appear more than once on any given line.

Some G-codes are actually M-codes!

Line numbers are optional and effectively ignored but if you want you can use line numbers. Line numbers are expressed as the 'L' word followed one and up to five digits.

9.2 Numbers, Expressions and Parameters

9.2.1 Numbers

Numbers in commands are expressed as one or more digits, optionally followed by a decimal point and one or more digits. A number can optionally be preceded with a plus or minus sign.

It is also ok to leave a preceding zero out from a number.

Some valid examples:

7 3.1415 -2.0 .1

9.2.2 Expressions

Where ever a number can be used an arithmetic enclosed expression in brackets, '[' and ']', can be used.

For example the commands

X100

and

X [50 + 70]

are equivalent.

Arithmetic operators

Nine common arithmetic operations are supported.

The operations are divided to three groups and denoted as follows. Operations in a higher precedence group are executed before those with lower precedence. Operators in the same precedence level are executed from left to right. This pretty much follows the common arithmetic expression calculation order.

The highest precedence group only has the exponentiation or power operator '**'.

The second highest precedence contains the multiplication, division and modulus operators '*', '/' and 'MOD' respectively.

The lowest precedence is for addition, subtraction, i.e. '+' and '-', and the logical operators 'AND', 'OR' and 'XOR' aka exclusive or.

The logical operators work on numbers interpreting any non zero number as logical one or true value.

Arithmetic functions

Arithmetic expressions support a number of common arithmetic functions. See table Table 9.1.

Table 9.1: Mathematical functions

Operator	Function
ABS	absolute value
ROUND	round to nearest integer
FIX	round down to previous integer
FUP	round up to next integer
COS	cosine function
SIN	sine function
TAN	tangent function
ACOS	arcus cosine function
ASIN	arcus sine function
SQRT	square root function
EXP	e to the 'power of' function
LN	natural logarithm function

A function is expressed as the name of the function followed by the function argument enclosed in brackets, '[' and ']'.

For example calculate the square root of two you would write:

SQRT [2]

Note that for some weird historical reason one function, ATAN, has an exceptional format, to write the equivalent of

arctan(123/456)

you need to write

ATAN [123] / [456]

9.2.3 Parameters

Parameters are what most programming languages would call variables i.e. they storage locations for values which can be used in place of numbers in expression.

Instead of names the parameters are referred to by numbers preceded with a '#' sign.

For example

#5324

Some parameters have a special meaning, others you are free to use as you wish to make your code more useful.

You can for example create G-code programs that is parametrized to cut different sized parts by just changing a parameter or two.

To change or set a parameter you follow the parameter name with the equal sign '='. For example:

#5324 = 12.34

It is possible to both set and refer to the value of a parameter, even several times, on the same line. The parameter values are only assigned or set once the whole line has been executed. If a parameter is assigned several times on a line the last one will take effect.

So for example after executing following lines:

#22=5 #22=[#22+1] #23=#22 #22=[#22-1]

the parameter 22 has value 4 and parameter #23 has value 5.

Note that the number following the '#' sign can itself be an expression which makes for some interesting possibilities, you can for example use it to index parameter to simulate what most

program languages would call arrays.

9.3 G-codes and M-codes

9.3.1 Length Units, G20,G21 codes

Coordinates i.e. positions or distances in G-codes are expressed in either millimetres or inches.

The G20 code tells EazyCNC that following coordinates and lengths are in inches, G21 tells it that they are in millimetres.

It is recommended that no coordinates or length are specified on the same line as G20 or G21 code is used.

9.3.2 Coordinate Axes

G-code specifies the movement of the machining tool using orthogonal cartesian coordinates. (There is also polar coordinate mode, see G-codes G15 and G16.)

The coordinate system follows the right-hand rule with the positive X-axis pointing to right, Y-axis away from the operator and Z-axis pointing up, so values increase from left to right, front to back, and bottom to top, as illustrated in Figure 9.1.

Figure 9.1: Coordinate axes of a 3-axis CNC System

9.3.3 Setting the length units – G20,G21

G-codes specify the length and any related physical quantities like feedrate in either metric or imperial units. For metric system the basic unit is one millimetre and for imperial it is one inch. It is possible but not recommended to mix the units in a single G-code program, but it is perfectly feasible machine 'metric' G-code programs in a machine configure for imperial units as long as the program contains the proper G-code to set the length units.

The **G20** sets the imperial (inch) units mode and **G21** sets the metric (mm) units mode.

It is an error to have botn **G20** and **G21** on the same line.

9.3.4 Feedrate – F-word

The **F**-word sets the current feedrate for all the other movement commands than **G0** which is always performed at maximum speed.

Depending on the length units mode in effect the **F**-word value specifies the feedrate either in inches per minute or millimetres per minute.

It is an error if the feedrate is not a positive number.

9.3.5 Spindle speed – S-word

The **S**-word value sets the current spindle speed in rotations per minute (rpm).

It is ok to specify a larger rmp value than what is set up for the system, see Section 6.6, but of course the spindle can't run faster than it's maximum speed.

Setting the spindle speed does *not* turn on the spindle, you need to program **M3** or **M4** to turn on the spindle.

Note that the speed change can take several seconds to take effect so a dwell or **G4** code should be programmed right after a spindle speed change.

Also note that specifying zero speed does not guarantee that the spindle stops, on the contrary it is most likely to remain turning at some low speed. To stop the spindle use the **M5** command.

Remember never to touch the spindle unless the kill switch has been applied!

It is an error if the value is not a positive number or zero.

9.3.6 Spindle on/off – M3,M4,M5 codes

The **M3** code turns on the spindle in the clockwise a.k.a. forward direction direction.

The **M4** code turns on the spindle in the counter clockwise a.k.a. reverse direction direction.

Note that reverse running the spindle can be dangerous if the chuck (in a lathe) is screw mounted and of course trying to machine with reverse running cutter will only damage the cutter and ruin the workpiece.

The **M5** code turns off the spindle.

Note that the turning the spindle on/off can take several seconds to actually take place so a dwell or **G4** code should be programmed right after commanding the spindle on or off.

Also note that depending on your spindle drive system abruptly changing directions can damage the equipment or be dangerous.

It is an error if more than one of **M4,M5** or **M6** is programmed on the same line.

9.3.7 Coolant on/off – M7,M8,M9 codes

The **M7** (mist cooling) and **M8** (flood cooling) codes both turn on the coolant and mist/flood aspect of the codes is ignored.

The **M9** code turns off cooling.

Note that when turning coolant on it can take several seconds for the coolant pump to react so you may want to program a dwell or **G4** right after turning the coolant on.

It is an error if more than one of **M7,M8** or **M9** is programmed on the same line.

9.3.8 Select a tool – T-word

The T-word designates a tooltable entry as the current tool. Note that this alone does *not* do anything else.

To physically change the tool you need to program a pause **M0** and change the tool yourself and to use the tool length you need to use the **G43** command and to use the tool diameter info you need to program **G40** or**G41** as needed.

It is an error if the value is not a positive or is larger than the number of tools supported by EazyCNC.

9.3.9 Dwelling – G4 P*-code

The **G4**-word causes eazycnc to wait for the specified time before executing the next G-code program line, this is useful for example after turning on or changing the spindle speed.

The wait time is specified with the P-word in either seconds or milliseconds depending on which G-code interpretation options has been selected, see Section 6.7. If a G-code program seems to hang for a long time then probably that setting for the P-word interpretation is wrong.

It is an error if the value is not a positive or zero.

9.3.10 Coordinates/moving axes – XYZABC -words

Coordinates in G-code programs are expressed as an axis letter, see followed by the coordinate position either as a number or as an expression enclosed in brackets.

The presence of an axis letter in a G-code line is an implicit command for the named axis to move.

The movement is carried out in the current motion mode either **G0,G1,G2** or **G3**. This allows for more compact representation of the toolpath as most of a G-code program will be just a long list of coordinates to move the tool.

Axis letter names the axis that will move, valid letters are '**X**',**'Y'**',**'Z'**',**'A'**',**'B'**',**'C'**'.

If an axis does not need to move on a given line it is not necessary to specify the axis and its position at all.

The axis movements are co-ordinated so that the movements start and stop at the same time and the axis speeds are such that the tool will move at the specified rate.

Here is an example that will cause X and Y axis to move to the position 120,140 in current motion mode:

```
X 120.0 Y [100.0+40.0]
```

It is an error if the same axis word appears twice any given line.

Note that actual axis movements are subject to machine limits acceleration limits and if the best speed mode (**G64**) is enabled then the movements may 'cut corners'.

Also note that the coordinates specified with the axis words maybe scaled, offset and even rotated, see XXX

9.3.11 Motion mode – **G0,G1,G2 and G3** codes

As describe above specifying coordinates with axis words causes EazyCNC to move the axis according to the current motion mode.

The **G0,G1,G2** and **G3** codes are used to specify the current motion mode, which will stay in effect until and other motion mode is specified.

If the motion mode is explicitly specified with one of the above G-codes, then at least one axis word needs to specified on the same line.

It is an error to specify more than one motion mode in a single line.

Is is an error to specify a motion mode without any axis words.

9.3.12 Rapid positioning – **G0** code

Rapid positioning will move the specified axes in co-ordinated fashion as fast as possible i.e. at the Max Velocity set up in the Mach Setup, see YYY.

G0 is used to rapidly position the tool for the beginning of a cut and is not meant for machining.

9.3.13 Linear interpolation – **G1** code

For some archaic reason the machining G-codes are called interpolations.

G1 is used to tell EazyCNC to move the tool at the current feedrate, to the given position. This is the 'work horse' mode of all G-codes, most machining will take place in this mode.

9.3.14 Clockwise Arc interpolation – G2 code

G2 is used to tell EazyCNC to move the tool at the current feedrate from its current position to the given position following, a circular arc path clockwise on the active plane as set by the G17 (XY-plane), G18 (XZ-plane) or G19 (YZ-plane).

Clockwise means as if you were looking down at the arc on the active plane from the third axis i.e. if your are cutting in the XY plane and looking down at it from the positive Z axis.

The end point of the arc is specified with the 'X', 'Y', 'Z' words. It is ok to leave out axes which do not need to move, for example typically if you are cutting an arc in the XY plane then you don't specify the 'Z' coordinate.

If a movement in the direction perpendicular to the current plane is specified then the tool will actually follow a three dimensional helical path.

The curvature of the arc is specified either by giving the centre point of the arc or by specifying it's radius.

Specifying arc using centre point

G2 X Y Z I J K specifies the arc curvature using the radius method.

The centre point is specified with the 'I' and 'J' and 'K' words for the coordinates in the X,Y and Z planes respectively. The the IJK words specify the centre coordinates relative to the starting point of the arc or as coordinates in the current coordinate system.

Which interpretation is used depends on the machine setup, see XXX. If your toolpath preview shows large arcs that don't make sense then it is likely that the interpretation mode of the IJK words is wrong.

It is ok to omit any of the any but not all of the end point words (XYZ) and centre point words (IJK) in which case the last specified word values are used.

Note that by specifying the start,end and centre point of the arc you are over specifying the arc, this may result in an error message if the distance from the start point to the centre differs too much from the distance from the end point to the centre.

Specifying arc using radius

G2 X Y Z R specifies the arc curvature using the radius method.

If the centre point method of specifying the arc curvature is not used then radius of the arc must be specified with the 'R' word. The start and end points alone do not specify an unambiguous arc, mathematically for any two points and radius there are two arcs that connect, one arc is less than 180° and the other is larger.

If the radius specified with the 'R' word is positive then this is interpreted as the arc that

turn less than 180° and if the radius is negative then it is interpreted to mean the arc that makes the longer turn and the absolute value of the 'R' word is used as the radius.

Do not try to cut full or nearly full circles with this method as this makes the start and endpoints very nearly the same which means that any roundoff error in the calculations has large effect in the internal arc centre point calculation.

9.3.15 Counter Clockwise Arc interpolation – G3 code

The G3 command performs the same as G2 but the arc is 'drawn' counter clockwise.

9.3.16 Perform probing move – G31

The probing command 'G31' works the same as the in other words it programs a linear movement at current feedrate, except that this command must always be explicitly specified on a line and does not 'carry over' from line to line like the modal .

If the probe input becomes activated during the movement the movement is stopped as soon as feasible without losing any steps and the parameters DRO values at which the probe became active are copied to the parameters 2000-2005.

Note that it is important that the feedrate is low enough because the probe movement will overshoot the position at which the probe trips/triggers by an amount that depends on the feedrate and it is equally important that the probe design allows for the overshoot, otherwise the probe and/or workpiece can be damaged.

9.4 Coordinate systems

As said in Section 9.3.10, the coordinates are specified using the axis words 'X', 'Y', 'Z', 'A', 'B', 'C' and 'I', 'J', 'K' for the arc centres in G2 and G3 commands.

This is not the end of the story though.

G-codes provide number of ways to transform the axis word values before they become final physical positions of the CNC machine axes.

This section provides the nitty gritty details.

Table 9.2 lists, in order of application, all the coordinate transformations every axis word value goes through.

If the absolute mode G53 is active then none of the offsets or the rotation is applied.

Table 9.2: Coordinate transformations

Apply G51 Scaling
In in G91 mode interpret as incremental coordinates
If in G16 mode interpret as polar coordinates
Apply G52 temporary coordinate offsets
Apply G68 rotation
Apply G44 tool length offset
Apply G54 work/fixture offsets

9.4.1 Scaling – G50,G51 codes

G50 turns off scaling and sets scale factors for all axis words to 1.

G51 X* Y* Z* A* B* C* turns on scaling and specifies the scaling factor the given axes. The scale factors for axes that are not specified remain at their previous values.

The purpose of scaling is to scale the part being cut and thus it is the first operation to be applied to coordinates and so it will not affect position of the part on the workpiece which is set by the fixture offsets nor does it affect the tool length or temporary offsets are applied.

For example following will scale a 2D part design to three times it's original size.

```
G50      ; ensure all scaling is off and set to 1.0
G51 X3 Y3 ; scale 2D design by two
```

It is an error to have both G50 and G51 on the same line.

9.4.2 Incremental mode – G90,G91 codes

G91 turns on incremental coordinate mode. In incremental model all axis words are interpreted relative to the current to position tool position.

For example following will scale a 2D part design to twice it's size.

```
G91      ; turn on incremental mode
G1 X100 ; mill hundred units to the right
Y50      ; hundred units forward
X-100    ; back to start X coordinate
Y-50      ; and we are where we started from
```

defines a rectangular toolpath 100 units wide by 50 units high to the right and front of current tool position.

G90 turns off incremental coordinate mode

It is an error to have both **G90** and **G91** on the same line.

9.4.3 Polar coordinate mode – **G15,G16** codes

G16 turns on the polar coordinate mode and sets the current tool position as the origin of the polar coordinate system. In polar coordinate mode the '**X**' word is interpreted to mean the polar coordinate distance relative to the polar coordinate origin and '**Y**' word is interpreted as the angle (in degrees) of the polar coordinate.

In incremental mode '**X**' and '**Y**' are interpreted relative to the previous distance and angle, not the current cartesian **X,Y** position.

The following example defines a hexagonal tool path where each side of the hexagon is 10 units long.

```
G16      ; turn on polar mode
G91      ; turn on incremental mode
X0 Y0    ; 'reset' distance and angle for incremental mode
X10 Y60  ; Cut the 1st side
X10 Y60  ; ... and 2nd side
X10 Y60  ; ... and 3rd
X10 Y60  ; ... 4th
X10 Y60  ; ... 5th
X10 Y60  ; 6th side and back to where we started from
```

Following example defines a pentagonal tool path with radius of 20 units.

```
G16      ; turn on polar mode
G90      ; turn off incremental mode
X20 Y72  ;
X20 Y144 ;
X20 Y216 ;
X20 Y288 ;
X20 Y360 ;
```

G15 turns off the polar coordinate mode.

It is an error to have both **G15** and **G16** on the same line.

9.4.4 Temporary coordinate system offsets – **G52**

The **G52 X Y Z A B C** code sets the temporary coordinate offsets for the given axes to the given values. The offsets for axes that are not specified remain at their previous values.

9.4.5 Temporary coordinate system offsets – G52,G92,G92.1,G92.2,G92.3 codes

All these codes are legacy features that are best left unused.

As all *G52* and *G92* codes all use the same mechanism mixing them requires extreme care, one more reason to leave all this well alone!

92 X Y Z A B C sets the temporary coordinate offsets for the given axes so that the current tool position has the given coordinates.

92.1 saves the temporary offsets to G-code parameters 5211..5216.

92.2 clears the temporary offsets.

92.3 restores the temporary offsets from the parameters 5211..5216.

92.1 X Y Z A B C

9.4.6 Coordinate system rotation – G68,G69 codes

G68 A B I R sets the coordinate system rotation.

The 'A' and 'B' specify in the local coordinate system the X,Y coordinates around which the coordinate system is rotated.

The 'R' word specifies the rotation in degrees, positive values giving counter clockwise rotation when viewed down from positive Z-axis i.e. looking down at the work piece.

If the 'I' word is present then the 'R' word value is treated as an increment to the current rotation, the value of 'I' word is ignored.

Rotation is only available in the XY-plane. Rotation can only be turned on if the active plane is XY (**G17**).

G69 turns off the coordinate system rotation.

It is an error if 'A', 'A' or 'R' is not specified.

It is an error to have both **G68** and **G69** on the same line.

9.4.7 Active plane – G17,G18,G19 codes

The arc interpolation i.e. arc cutting ('G2' and 'G3') works by calculating a circular path in the active plane which is one of the the main coordinate planes XY,XZ or YZ plane.

The 'G17' sets the active plane to XY-plane.

The 'G18' sets the active plane to XZ-plane.

The 'G17' sets the active plane to YZ-plane.

It is an error to use more than one of G17,G18 or G19 on the same line.

It is an error to program G18 or G19 if the coordinate system rotation G(is on.

9.4.8 Tool length compensation – G43,G44,G49 codes

G43 H sets the tool length compensation based on the tool number specified with the 'H' word and the tool length of set up for that tool in the Tool Setup panel.

G44 H works the same as G43 but it expect that the length values in tool set up are negative, this should not be used and is provided for compatibility only.

A 'H' value of 0 can be used to turn off tool length compensation and is equivalent to the G49 command.

It is an error 'H' word is missing, is not an integer, is negative or larger than the number of tools EazyCNC supports.

It is an error more than one of G43,G44 and G49 on the same line.

G49 turns the tool length compensation off by setting it to 0.0 value.

9.4.9 Work/fixture offsets – G54,G55,G56,G57,G58,G59 codes

Any of the six first work/fixture coordinates systems/offsets can be selected with the G54...G59 codes. The G54 selects the first i.e. number one coordinate system specified in the Work Offsets panel.

G59 P* selects the work/fixture coordinate system specified with the 'P' word. The 'P' word is optional and if not given it behaves as described above.

It is an error the'P' word is given with G59, is not an integer, is smaller than 1 or larger than the number of work/fixture coordinate systems EazyCNC supports.

9.4.10 Absolute coordinates – G53 code

If a line that causes linear interpolation (i.e. implicit or explicit G0 or G1 command) contains the G53 command then all the coordinates on that line are treated as absolute coordinates without applying any offsets or rotations. Scaling and cutter radius compensation are applied regardless if they are enabled.

9.4.11 Cutter radius compensation – G40,G41,G4 codes

EazyCNC can adjust the programmed tool path automatically to compensate for the non-negligible width of the cutter. While this works the cutter compensation is best carried in the CAM software that you should be using to create the G-code program.

When you use the G-code cutter radius compensation you program your tool paths as if you were cutting the outline of the part with an infinitely thin cutter i.e. the XYZ coordinates specify the outline of the part to be cut.

Of course the real cutter has a substantial size so you need to tell EazyCNC what is the cutter diameter and on which side the part outline the cutter should cut.

Use G40[^]to indicate that the cutter should stay to left of the toolpath, use this if you are cutting the outline of a part clockwise (or if you are cutting a hole counter clockwise). Use G41 to indicate that the cutter should stay to the right.

Left and right are defined as if you were riding on the cutter and facing in the direction of the travel.

Note that when you turn on the cutter compensation it will only affect the next movement of the tool ie the tool does *not* move from where it is when you turn on the compensation, rather the next position will be offset by the tool radius and the next move will then move from the current position to that compensated position. Therefore you should always plan a move-in movement when you turn on the compensation.

To turn off the compensation use G42.

There are several ways you can specify the tool/cutter radius. Following shows them for G40 but G41 works just the same, only the compensation is taken to the right.

To turn on the compensation to the left and use the tool/cutter diameter stored in the tool table for the currently selected tool (as specified with the T -word) use G40.

To turn on the compensation to the left and use the tool/cutter diameter stored in the tool table use G40 D , where the D word specifies the number of the tool in the tool table.

To explicitly set the compensation to the left with given amount of compensation use G40 P , where the P word specifies the *radius* of the cutter.

9.4.12 Feedrate mode – G93,G94,G95 codes

The EazyCNC does *not* support the inverse time feedrate mode **G93** nor the units per ref feedrate mode **G95**.

The **G94** code programs the feed per minute mode in which axes movements are carried out so that the tool moves at the rate specified with the F-word inches or millimeters per minute depending on which length unit mode, **G20** or **G21**, is in effect.

It is an error to use **G93** or **G95** code.

G40 nor

9.4.13 Feedrate override on/off – M48,M49 codes

The feed override that the machine operator can adjust during machining, see Section ??, can be turned on or off in the G-code program, but programming it in G-code does not prevent the operator from turning it on and off again.

The idea is that the G-code programmer knows the best what feedrate the part should be cut but sometimes it is necessary to adjust or fine tune that while machining.

The actual override percentage *cannot* be set with G-codes.

The **M48** code turns the feed override on.

The **M49** code turns the feed override off.

It is an error to have both **M48** and **M49** codes on the same line.

9.4.14 Tool change – M6 code

EazyCNC does not support the **M6** tool change command but ignores it.

9.4.15 Tool length compensation – G43,G44,G49 codes

The length compensation basically works by just offsetting the Z-axis value so that there will be room for the specified tool length between the spindle/chuck and the work piece.

To turn off the tool length compensation program **G49**.

To turn on and set the tool length compensations from the length stored in the tool table use **G43 H**, where the H-word specifies the number of the tool in the tool table. If the H-word is zero i.e. specifies no tool number then compensation is set to zero and effectively turned off, in fact this equivalent to programming **G49**.

The G44 command works the same as G43 but expects that the length values in the tool table are negative. Since you can't enter negative values this is provided for compatibility with existing practice only.

It is an error if the H-word value is negative or larger than the number of tool supported.

It is an error to have more than one of these G-codes on the same line.

9.4.16 Path mode – G61,G61.1,G64 codes

In any CNC system there are basically two options how the system tries to follow the specified tool path. Either the system tries to obey the specified coordinates i.e. position or the specified speed. You can't have both at the same time, think about it: if you need to move from point A to point B at a given speed you would need infinite acceleration and deceleration at the beginning and end of travel.

The path mode along with the machine limits, Section 6.5, determine how EazyCNC calculates the actual tool path.

The set exact path mode program G61, in this mode the path follows the specified path as closely as possible which results in the axes velocities coming to a complete stop at the end of movement. This is fine when milling but slows down the machining, especially if a lot of small cuts are used. When cutting with a torch stopping at the end of the movements causes local 'burn outs' so this is not an acceptable mode for plasma cutting.

The set best speed mode program G64, in this mode the path tries still to follow the specified path but is allowed to deviate from it by as much as the 'Path tolerance', Section 6.5, allows trading accuracy for speed. Because of limited path lookahead many small cuts in a row still result in a severely limited speed, so when programming tool paths for plasma try to avoid small movements.

For all practical purposes G61.1 performs the same as G61 and this is supported for compatibility only.

It is an error to have more than one of these commands in the same line.

9.4.17 Incremental XYZ mode – G90,G91 codes

The axis words, 'X', 'Y', 'Z' etc can be interpreted either as coordinates or as a change of coordinates relative to the previous coordinates.

To treat coordinates as 'absolute' positions in the local/current coordinate system program G90. This is the usual way to specify coordinates in G-code programs.

To turn on the incremental interpretation program G91, in this mode the axis word values are treated as increments to the previous axis word values.

It is an error to have both of these codes in the same line.

9.4.18 Incremental IJK mode – G90.1,G91.1 codes

Interpretation of the IJK values in the arc interpolation codes G2 and G3 can be either absolute or incremental.

To set the incremental interpretation use G91.1, in this mode the the IJK words specify the centre coordinates relative to the starting point of the arc.

To set the absolute interpretation use G91.1, in this mode the the IJK words specify the centre coordinates in the local/current coordinate system.

Incorrect settings of this mode will usually result in large and incorrectly oriented arcs in the toolpath display.

You can also set this mode in the Mach Setup screen, see Section [6.7.1](#).

It is an error to have both of these codes in the same line.

9.4.19 Set tool table – G10 L1 code

It is possible change tool table entries with G-code commands. This makes it possible to maintain different tool sets.

G10 L1 P A Z X set the tool table entry for tool number specified by the P-word. The Z-word sets the tool height and the X-word sets the tool *radius*; the A-word, tool tip radius, is ignored but accepted for compatibility reasons.

The A,Z and X -words are all optional and it is ok to leave any or all of them out.

It is an error if the P word is missing, smaller than one or larger than the number of tools EazyCNC supports.

It is an error if the L word is missing.

9.4.20 Set work/fixtures offsets – G10 L2 code

It is possible to set the work/fixtures offsets in G-code. This makes it possible to maintain multiple different jig set ups easily.

G10 L2 P X Y Z A B C sets the work/fixtures offsets for the fixture number specified with the P word. The axis words X,Y etc specify the corresponding offsets. It is ok not specify all or none of the axis and those that are not specified are left untouched.

It is an error if the P word is missing, smaller than one or larger than the number of work/fixtures offsets EazyCNC supports.

It is an error if the L word is missing.