COAST RUNNER CR-1

OPERATOR’S MANUAL

Version 0.9
November 1st, 2024

Table of Contents

Introduction 5
How To Read This Manual 5
Safety 6
CNC Safety 6
Electrical Safety 7
Operation Safety 8
Section 1: Setup 9
1.1: Choosing an Operating Environment 9
1.2: What's in the Package? 10
1.3: Unboxing 10
1.4: Hardware Setup 11
1.5: Software Setup 11
1.6: Updating Software and Firmware 12
1.6.1: Software Updates 12
1.6.2: Automatic Firmware Updates 13
1.6.3: Manual Firmware Updates 13
Section 2: Hardware Overview 14
2.1: Physical Overview of the Coast Runner CR-1 14
2.2: Emergency Stop 16
2.3: Voltage Selection 17
Section 3: Software Overview 19
3.1: CRWrite Overview 19
3.1.1: CRWrite Main Page 20
3.1.2: CRWrite Settings 21
3.1.3: CRWrite Support 23
3.2: grbl Overview 25
3.2.1: Communicating with grbl 25
3.2.2: grbl States 27
3.2.3: The grbl Buffer 28
Section 4: Basic Operation 29
4.1: Guided Mode 29
4.1.1: Cutcodes / .crproj Files 29
4.1.2: Launching Guided Mode 29
4.1.3: Guided Mode Interface 30

4.2: Homing 34
4.3: Autoleveling 35
4.4: Fixtures: Attaching the Workpiece 36
4.4.1: Making / Acquiring Fixtures 37
4.4.2: Fixtures and Indexing 37
4.4.3: Working with Fixtures 37
4.5: Installing Tools 39
4.6: Indexing / Probing 43
4.6.1: Edgefinding 43
4.6.2: Touchoff 44
4.6.3: Conductive Probing 45
4.6.4: Touch Probing 46
4.6.5: Probe Wizard 47
Section 5: Advanced Operation / Direct Control 48
5.1: Manual Mode 48
5.1.1: Launching Manual Mode 48
5.1.2: Manual Mode Interface 48
5.2: Writing Code: Movement 55
5.2.1: Basic Movement 55
5.2.2: Work Coordinate Systems 57
5.2.3: Spindle Control 58
5.3: Writing Code: Modal State 60
5.4: Writing Code: Indexing 61
5.5: Creating Fixtures 64
5.6: Creating crproj file 65
Section 6: Creating CAM 66
6.1: Creating the Setup 66
6.2: Creating Toolpaths 66
6.3: Exporting Your G-Code 66
Section 7: Maintenance / Disassembly 67
7.1: Basic Maintenance 67
7.1.1: Regular Cleaning 67
7.1.2: Tool Wear 68
7.2: Advanced Maintenance 70
7.2.1: Removing / Replacing Enclosure 70

7.2.2: Replacing Spindle Belt 72
7.2.3: Removing / Replacing Spindle 74
7.2.4: Replacing Limit Switch 75
7.2.5: Replacing Power Supply 77
7.2.6: Replacing Electronic Board 77
Section 8: Troubleshooting 79
Section 9: Reference 81
9.1: G-code Commands 81
9.1.1: Common Movement Commands: 81
9.1.2: Jog Commands: 84
9.1.3: Common Spindle Commands: 85
9.1.4: Common Indexing Commands: 86
9.1.5: Other Common Commands: 88
9.1.6: CRWrite-Only Commands 89
9.1.7: Uncommon Commands: 91
9.2: grbl Commands and Settings 93
9.2.1: Dollar Commands: 93
9.2.2: Parser State 96
9.2.3: Parameters 97
9.2.4: Real Time Commands 98
9.2.5: Differences From Base Grbl 100
9.3: grbl States 101
9.4: grbl Errors 102
9.5: grbl Alarms 105
9.6: .crproj formats 106
9.7: CR-1 Technical Specifications 107
9.7.1: Axis Lengths and Signs 107
9.7.2: Spindle Speed 107
9.7.3: T-Slot Table 108
9.8: Glossary 109

Introduction

Welcome to the Coast Runner CR-1 Operator's Manual. This manual will provide a thorough overview of the ownership, operation and maintenance of the CR-1 desktop CNC mill.

The CR-1 occupies a particularly interesting niche in the CNC world. It is a desktop CNC mill designed for industrial use – not a toy, not an engraver, but a genuine mill. CR-1's hardware is capable of cutting hard metals and holding industrial tolerances. Yet unlike other mills with this capability, the CR-1 is accessible to everybody, both in terms of its low price and its ability to fit and operate in any shop, garage or home.

We originally conceived the CR-1 as the best CNC mill for hobbyists available. It is that, but from the reaction we received on launch we came to understand that it is also a tool for business and industry. For many businesses, it will be their first step into automated in-house manufacturing. Even for operations already equipped with industrial CNCs, the CR-1 plays a role as a highly portable rapid prototyping mill.

The CR-1 project is more than just the mill hardware. It is also the software, which makes the mill easier to operate than any other. It is our unique education program, introducing people of all ages and backgrounds to CNC milling. It is the community of machinists sharing milling files, advice and knowledge with each other at CoastCAD.com.

While our scope and goals have grown from our early hobbyist aspirations, our maker sensibilities remain. This Operator's Manual will tell you all there is to know about how to operate, maintain, modify, hack and otherwise make full use of your Coast Runner CR-1 mill.

Thank you so much for your purchase and use of our technology. We believe it will serve you well.

~ Coast Runner CNC Team

How To Read This Manual

This manual serves as both an instructional guide and as reference material. It contains useful information no matter your experience level.

All new CR-1 owners should read **Sections 1, 2, 3** and **7**. These cover important details for the CR-1 mill.

If you are brand new to milling, and want to learn the basics of machining, we recommend proceeding to **Section 4**. As you grow, we invite you to learn **Section 5** and **6**.

Once you are writing your own G-Code, **Section 9** will be very useful for reference purposes.

Throughout this manual you will likely encounter terms you aren't familiar with or don't fully understand. We encourage you to refer to the **Glossary** at the end in these circumstances.

Safety

CNC mills can be dangerous, but it is not hard to operate them safely**.** We ask that all CR-1 owners follow our safety advice below at all times.

Please read this section closely. We put this at the very beginning of the manual for a reason.

NOTICE: Coast Runner CR-1 is not a consumer device. It is the user's responsibility to operate CR-1 per OSHA 1910.212 - Milling Machine, ANSI B11.8-1983, and OSHA 3067, as amended.

CNC Safety

The following safety procedures apply whenever you are operating any CNC mill, or any machine equipment generally.

Crush Hazard	The CR-1's table, gantry and spindle can crush, pinch, mutilate and destroy objects, including body parts. • **To avoid injury, never insert tools or body parts into the machine while the machine is moving**.
Cutting Hazard	CNC mills use very sharp cutting tools. When spinning, these tools can gouge and lacerate objects, including body parts. • **Never insert tools or body parts into the machine while the spindle is spinning.** The cutting tools themselves can also deliver nasty cuts if mishandled. • **Always take care when installing, handling and removing cutting tools.**
Heat Hazard	Milling operations produce significant heat. Workpieces, tools, chips and the CNC mill itself can become very hot during operation. The mill's motors in particular can heat up during operation, and retain heat for some time when stopped. • **Touching hot parts can produce minor burns.** • **Let parts cool before handling, or use gloves.**
Swarf Hazard	Cutting operations produce "swarf", including chips, shavings and dust. These chips can be very sharp and can cut or embed themselves in body parts, especially hands and feet. Airborne dust can cause respiratory damage when inhaled. • **Take all possible measure to contain swarf.** • **Install the magnetic chip cover during cutting operations.**

	• **Place mill in shipping foam or other "base" to contain swarf.** • **Locate mill in a low-traffic area with low consequences for swarf contamination (e.g. garage, workshop, etc.)** • **Wear gloves when handling freshly-cut parts.**
Swarf Buildup	Buildup of swarf occurs over time. • **Regularly clean swarf using a vacuum.** • **Wipe swarf off machine parts and workpieces with a cloth.**
Projectile Hazard	Mishaps during operations (including tool breakage, improperly secured tools, part breakage, etc.) can create high-speed projectiles exiting the machine. These can damage property and body parts, especially eyeballs. • **Always ensure tool is secured tight in spindle** • **Install the magnetic chip cover during cutting operations.** • **Wear safety glasses during operation**
Noise Warning	Milling operations can be very loud. **Wear hearing protection during operation**

Electrical Safety

The CR-1 contains high voltage electrical components. Please follow these safety rules to avoid dangerous – possibly fatal – electric shocks.

Grounding Warning	The CR-1's electrical components, including its power supply, are inside its main bay. This means that swarf and chips can build up and possibly cause electric shorts. For this reason, the CR-1 absolutely must be properly grounded. • **Always ground the CR-1 via the grounding pin ("third pin") on its power plug.** • **NEVER remove or otherwise bypass the grounding pin.**
Voltage Selector	The CR-1's power supply can be configured to accept either 120V or 240V input voltage. If the power supply is not configured for the correct voltage, the power supply and / or mill electronics may be damaged, or will fail to work. • **Always ensure that your CR-1's voltage is configured properly before plugging it in.** • **Review our Voltage Selector information in Section 1.7 for more details**

Operation Safety

Please take the following steps and precautions every time you run the CR-1.

Update Software and Firmware	Always ensure that you are running the most up-to-date version of the CR-1 software and firmware. Running old versions of the software or firmware may cause degraded or improper performance. • **Review Section 1.9 for instructions on updating the software and firmware.**
Ensure Latest Cutcode Version	If you are running a third-party cutcode / crproj file, please ensure that the file you are about to run is the latest version. Newer versions of a file will contain bugfixes and new features; it is generally desirable to always use the latest version. • **If the third-party file is published on CoastCAD.com, please check CoastCAD to see if there is a newer version.**
Start With Machine Empty	Unless you know what you are doing, always start a new cutcode or milling operation with the machine empty – no tools installed, no fixtures installed on the bed. Unfamiliar cutcodes may move the spindle in unexpected ways, which may cause a crash if your machine is not empty.
Stay By Mill During Operation	We recommend that you always stand by your mill during the first run of a cutcode. Be prepared at all times to hit the mill's emergency stop button if unexpected performance occurs. • **If you wish to walk away from your mill during a long cut, use your best judgement whether it is safe to do so. We recommend you always stay by your mill the entire time the first time through a cut**.

Section 1: Setup

In this section we will walk you through the initial unpacking and setup of your Coast Runner CR-1.
We recommend that all new users read this thoroughly before unpacking your machine!

As we proceed through the setup, we'll also provide an overview of the CNC mill itself.

1.1: Choosing an Operating Environment

Before you unpack (or even receive) your CR-1, it would be wise to choose an area where you can set up your machine! Here are a few things to consider:

- **Swarf:** the CR-1 generates considerable amounts of swarf (chips, dust, etc.) during operation. When using a chip cover and base, swarf is generally well-contained, but some chips may escape during operation. Swarf can also inadvertently exit the enclosure during cleaning.

- **Noise:** the CR-1 can be quite loud. This noise may be obnoxious or disruptive to yourself and / or your neighbors.

- **Access:** the CR-1 is a power tool and is therefore dangerous. Children, pets or anyone unfamiliar with safe handling of the machine may inadvertently hurt themselves if given unrestricted access to the machine.

With these considerations in mind, we recommend an operating environment with the following qualities:

1. Choose a location **away from high-traffic areas** to allow for easy access control, reduce noise exposure and prevent people walking through swarf on the floor and tracking it away.

2. Place the machine with **at least one wall** (more is better) between the CR-1 and areas that may be sensitive to noise (e.g. bedrooms, show floors, etc.) Even one wall will reduce noise considerably.

3. Place the CR-1 on a **hard surface**, and above a hard floor, to facilitate easy swarf cleanup (avoid running the CR-1 on or near carpet – chips are very hard to clean out of carpet!)

4. Try to locate the CR-1 **at least one meter away from electronics**, which may be damaged or shorted by swarf. This includes the operating computer – if you can, place the computer a full USB cable's length away from the mill!

5. Place the CR-1 such that it sits **at least at waist height**, on a table or desk. Milling requires frequent interaction (tool changes, orientation changes, etc.) – place the mill in a location convenient for this interaction. Avoid inconvenient locations such as the ground.

Keeping your CR-1 clean will help keep your operating environment clean! **Review Section 7.1** for information about cleaning the CR-1 and its operating area.

1.2: What's in the Package?

Your CR-1 will arrive containing at least the following:

- 17mm combination wrench
- 12mm combination wrench
- 3/16" Allen key
- 3mm Allen key
- ER-11 collet nut
- Data cable, USB-A to USB-B
- Power cable, NEMA 5-15P to C13
- Flash drive containing latest cutcodes and software
- Shipping foam
- The cardboard box itself (make sure to save it!)

Depending on your purchase, more may be included. Review the product listings at coastrunner.net to learn more about precisely what is contained in your mill bundle / starter kit.

1.3: Unboxing

Please follow these steps to ensure a pleasant unboxing experience. **The CR-1 is a heavy machine** – follow these instructions to remove it from its shipping container without damaging either the mill or yourself.

1. Cut the shipping tape to open the box. Don't stick the knife deeply inside while cutting.

2. Remove the smaller cardboard accessory box and set aside for now.

3. Remove the four shipping foam corners

4. You will now be looking down at an upside-down CR-1. Observe that it has two handles on the left and right sides. Insert your fingers into these handles and then carefully pull the CR-1 up out of its shipping foam

    a. The handles have a rubber guard to protect your fingers when lifting. However, the guard will be on the top of the handles, which is not the side you're grabbing right now. As such, we recommend wearing work gloves when picking the CR-1 up out of its box, to avoid discomfort.

5. Place the CR-1 (still upside-down) on a level work surface

6. Remove the large shipping foam base from the cardboard box and place it aside.

7. Retrieve and open the smaller cardboard accessory box. Review the contents.

The CR-1 is now fully unpacked! Move on to the next section to set up your new CR-1.

Do not discard any shipping foam or the cardboard shipping box. We recommend you place the four corner foam pieces in the shipping box and store the box somewhere (box can be broken down and folded up). Keeping these items will be convenient for transporting the CR-1, or for returning the mill to the factory for service.

1.4: Hardware Setup

We recommend setting up your CR-1 in the following way:

1. Take the large shipping foam base and place it in the intended location for your CR-1.

2. Carefully turn the CR-1 right-side-up and lift / place it in the shipping foam, with the bay facing you.

3. **VERY IMPORTANT:** properly set the voltage to match your household power settings (see **section 2.3** for details)

4. Place the magnetic chip cover on the machine to cover its open bay.

5. Check the red emergency stop button to ensure it is not set. Twist the button in the reset direction – if the button is set, it will pop out as it resets.

6. Connect the included NEMA power cable, or another **properly grounded** power cable, to the CR-1's power socket. Plug the other end of the cable into a standard power socket.

   Upon plugging the CR-1 in, the lights on the mill's LED board will turn on, and you will hear the mill's fans begin spinning. **If these things do not happen, and you are sure that your power outlet is providing power, please contact Coast Runner support for help**.

7. Connect the USB-B end of the USB cable to the USB port on the back of the CR-1. Connect the other end of the USB cable (USB-A) to your operating computer.

1.5: Software Setup

Now that your CR-1 is set up on your desk, it's time to install the software we can use to run it.

1. Remove the USB drive from the cardboard accessory box and plug the drive in to your computer.

2. On the drive, find the CRWrite installer appropriate to your operating system (Windows or Mac). Double-click to run the installer. The wizard will install CRWrite as normal.
   (If the installer files are not on the drive, check out https://coastrunner.net/downloads)

3. Once installed, launch CRWrite. Wait for the software to complete its first-time load (which may include installing Arduino drivers, if you do not already have them.)

4. Once the software has fully loaded, ensure the USB-A cable is connected to the computer. If it is, the connection icon in the bottom left corner (labeled CR Status) should turn to green and say "connected."

5. If the software does not connect to the CR-1, close the software and unplug the USB cable from the computer. Then re-open the software, wait until it is fully open, and then reconnect the cable.

1.6: Updating Software and Firmware

Now that CRWrite is installed and connected, it is a good time to confirm that your software and firmware are the latest versions.

If you purchased the CR-1 directly from the factory, then it should arrive with the latest firmware installed. That said, it is a good idea to regularly check for new firmware versions.

Fortunately, CRWrite will tell you via popup if there's a new software release, and will even give you the ability to automatically download and apply the update. If you ever get this message, you should update.

To upgrade the firmware, check the firmware settings under Settings > Software. If you see a new firmware listed in the "latest firmware" box, we recommend that you upgrade to the new firmware immediately.

1.6.1: Software Updates

It is important to always use the latest version of CRWrite. Using the latest software will help avoid bugs and make new features available to you. Some crproj files will refuse to run if your CRWrite version is too old.

At this time, you can update CRWrite by downloading the latest version from our websites. The latest version can be found here:

- https://coastrunner.net/downloads/
- https://github.com/CoastRunnerCNC/crwrite-v1

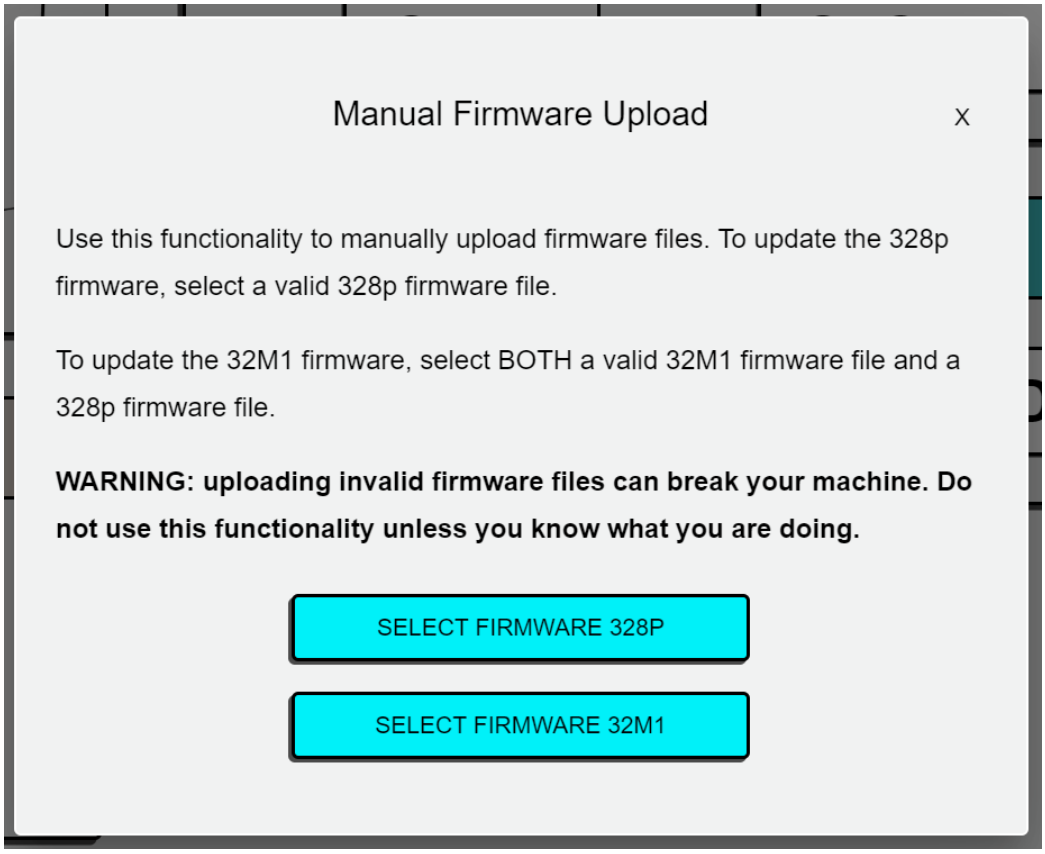
1.6.2: Automatic Firmware Updates

Automated firmware updates will be supported in a future version of CRWrite.

1.6.3: Manual Firmware Updates

You can update the firmware manually by choosing “Manual Firmware Upload” on the Settings > Software screen. Clicking this will open a popup where you can select both the ***328P Firmware*** and the ***32M1 Firmware***.

- The **328P firmware** is *grbl*, the control software which accepts G-code commands and moves the mill.
- The **32M1 firmware** controls the VFD spindle, holding it to a specific spindle speed under load.



Click the button to open a filepicker and select the corresponding firmware. Be careful not to select the 32M1 firmware for the 328P button, or vice versa!

Upon selecting the correct firmware(s), click the **Upload Firmware** button to perform the firmware update. Command boxes may flash or pop up on your screen during this time. Eventually the update will complete and you will receive a success message.

Note: you can update the 328P firmware by itself without selecting a 32M1 firmware. However, in order to update the 32M1 firmware you must always select an accompanying 328P firmware.

Section 2: Hardware Overview

In this section we will provide an overview of the Coast Runner CR-1 hardware. We will cover some basic configuration, such as the voltage selector. Hardware maintenance and disassembly will be covered in **Section 7.**

2.1: Physical Overview of the Coast Runner CR-1

This section will review the major components of the Coast Runner CR-1 mill.

Overview

We'll first perform a high-level overview of the mill's major parts.

- **Spindle**: holds the cutting tool and spins. Uses ER-11 collet system, accepts tools up to 8mm (~5/16") in diameter. The spindle can plunge and retract, creating the Z axis.
- **Gantry**: holds the spindle. Moves left and right, creating the Y axis.
- **Table:** used to secure the workpiece. Moves up and down, creating the X axis. Contains two t-slots, with which fixtures or workpieces can be secured.
- **Probe connector:** you can connect an (optional) probe cable here to utilize CR-1's electrical probing. A green LED is located to the right of the connector. The LED lights up when the probe is either tripped or disconnected.
- **Emergency Stop**: immediately stops all motion when pressed. After pressing, you must twist clockwise to reset the switch and restore the machine's functionality.

CR-1 Details

- **Frame:** the strong, rigid aluminum frame.
- **Spindle:** belt-driven BLDC. Inside of spindle is roughly 16mm / 0.6" deep – refer to this when calculating stickout.
- **Gantry**: mounts BLDC motor and Z-axis stepper motor. Z-axis component rides on two precision shafts and is controlled by a belt-driven ballscrew. Y-axis subassembly similarly rides on two precision shafts and a belt-driven ballscrew. Y stepper motor is mounted on right side of the frame.
- **T-slot table:** driven by two ballscrews (one on either side) and guided by precision shafts on each side. X-axis stepper motors are mounted at the top of the mill, beneath the chip guards, and directly drive the ballscrews. T-slot profile defined in **Section 9.7.3.**

- **Optical Limit Switches:** limit switches used to home mill. One for each stepper motor at the "home" position. Red light indicates switch is active and not tripped.

- **Reinforcement:** the CR-1 has several reinforcement pieces made from both steel and aluminum bolted on to its frame. These include the X-brace and two reinforcement pieces bolted onto the X-arms.

- **Chip guards:** chip guards covering electronics, power supply, X ballscrews, spindle.

- **Power Supply:** Located on right side of machine. Configurable for 120V or 240V household power. Cooling fan located on right side – do not block.

- **Electronics (Arduino Uno, Shield):** Arduino Uno Rev3 with custom shield. Located on left side of machine, with cooling fan – do not block port. Accessible by removing enclosure. Powered solely by USB cable – power cable does not provide power.

- **LED Strip:** Up underneath lip of the bay. Mounts probe connector, probe indicator light, several bay lights. Probe indicator light (and probe wire) receive power from Arduino, but lights receive power from power supply. When the lights turn on, your CR-1's motors are powered.

- **Enclosure:** sheet steel. Keeps chips in and covers things up. Provides no real structural rigidity (all rigidity comes from aluminum frame.)

2.2: Emergency Stop

The **emergency stop** button is one of the most important safety features on your **CR-1 mill**. This button will allow you to immediately stop the mill from damaging your workpiece, itself, or you. It is critical to familiarize yourself with this button.

The **emergency stop (e-stop)** button is located on the outside of the mill, on the left side. It's a big, red, obvious button. Pressing it down will lock it in place. The only way to release it when locked in place is to twist the button clockwise until it pops out.

The following things occur when the **e-stop** is pressed:

- **All motors** (both the **steppers** responsible for linear X/Y/Z travel and the **spindle motor**) are immediately powered down. All motion stops.
- The **grbl mill control firmware** is placed into reset, and will reject / ignore all sent commands.
- The **CRWrite software** likely locks up / freezes and must be closed and reset.
- The **LED strip** and **electronics cooling fan** (on the left side) remain on
- The **current cutting operation** will cease immediately and irrecoverably. You will need to restart the current operation.

Never be afraid to engage the e-stop. Doing so will not hurt the CR-1, and may save your workpiece or prevent damage to the machine or to yourself. The worst consequence is needing to reset your milling job – a small price compared to protecting your tools, workpiece and limbs.

When you rotate the e-stop button and cause it to pop out, the following things will happen:

- **Motor power** (both **stepper** and **spindle motors**) will be restored. You may hear a small noise as the motors power up. In some unusual circumstances, the mill may attempt to move / spin the spindle briefly.
- **Grbl** will remain in its alarm state and will have to be reset
- **CRWrite** may unlock and restore interactive functionality

Once you have recovered from the e-stop, you can start milling operations again. The mill **can start moving any time** after the e-stop is reset, so ensure that the mill is safe and clear before resetting the e-stop button!

Is your mill not responding to commands, especially if this is your first time using it? Check the e-stop. These buttons commonly get inadvertently pressed during shipping, setup, or anytime the mill is moved around.

2.3: Voltage Selection

Worldwide, there are two common voltages that standard household power runs at: **120V** and **240V**. The table below shows which countries use which voltages:

120V Standard	240V Standard
United States of America	Europe (most countries)
Canada	Asia (most countries)
Japan	Africa (most countries)
Some Central and South American countries	South America (most countries)

Note: this table is not exhaustive. Please double-check your country's power standards before operating Coast Runner.

The **Coast Runner CR-1** can be configured to operate at both **120V** and **240V** input power by setting a switch on the power supply.

Warning: never plug in or attempt to operate your CR-1 without confirming that it is set for the proper voltage. Failure to do so may break your power supply, or injure you!

By default, CR-1 is shipped from the factory set to the **120V setting**. If your household power operates at 120V, then you do not need to change anything.

If your household power operates at **240V**, **you must change your CR-1 configuration before plugging in or operating the CR-1**.

How to Configure the CR-1 for 240V:

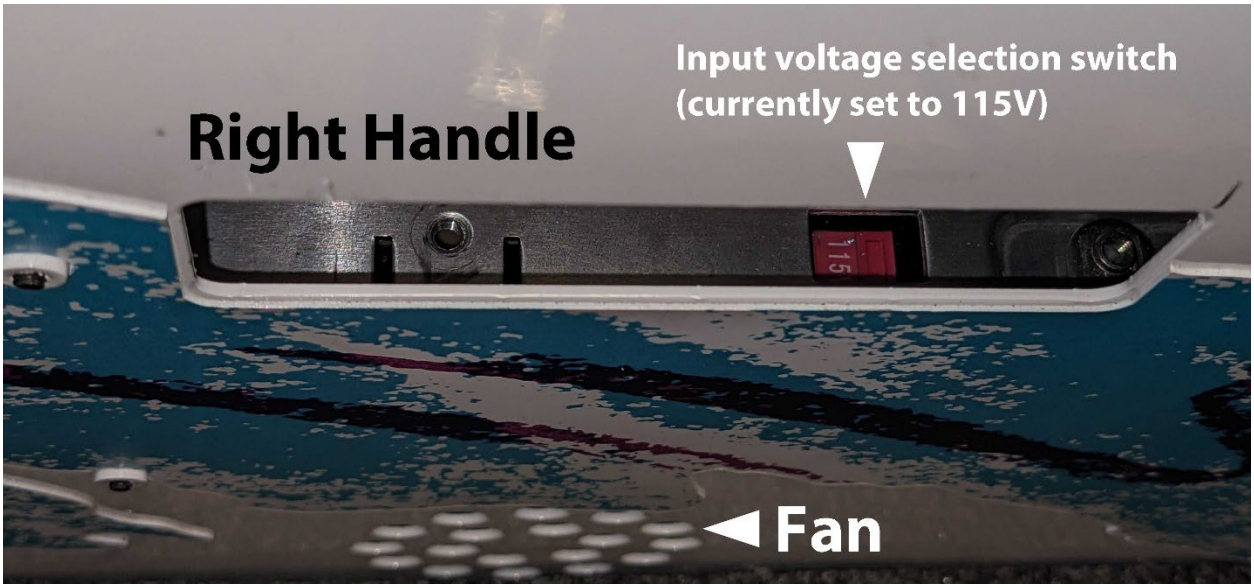
1. Ensure that the CR-1 is not plugged in (both power and USB data cables must be unplugged). Ensure that the cooling fans are not spinning.

2. Identify the **red power supply voltage configuration switch**. It can be seen through the **right handle** (see photo below)

3. Insert a long flathead screwdriver into the handle and use this to slide the switch to the **230V** position. You should read "230V" on the switch. **This configures the CR-1 for 230V power**.

4. You will also need to acquire a power cable or adapter to connect the **CR-1** to your power outlet. Use either an adapter / cable to the **C13** socket on the mill's power supply, or find an adapter to connect the **NEMA 15-5P plug** on the included power cable to your household socket.

<u>Warning</u>: never cut off, bypass or otherwise defeat the safety ground prong (third prong) on your power cable. Stray chips can cause a fatal electric shock on an ungrounded mill!



This photo shows the location of the input voltage selection switch, accessible through the right-side handle of the CR-1

Note: if desired, you can always return your **CR-1** to the **120V** configuration simply by sliding the switch back to the **115V** setting.

Section 3: Software Overview

In this section we will provide an overview of the software included with the **CR-1** mill – namely, **CRWrite** and **grbl**. To start, here is a brief description of each:

- **CRWrite** is a control software that you install and run on your computer. When **CR-1** is connected to this computer, **CRWrite** can connect to and read from the CR-1 mill. CRWrite handles walking users through milling operations, including sending G-code milling commands to CR-1.

- **grbl** (which we pronounce "gerbil", others "grr-ble" or "garble") is lower-level control firmware that runs on an Arduino Uno board installed on the **CR-1**. grbl receives G-code milling commands from control software (often, but not necessarily, CRWrite) and then plans and executes these commands by moving the CNC's motors. grbl also tracks and reports on the machine's status.

3.1: CRWrite Overview

CRWrite is a control interface created by Coast Runner CNC to help users interact with and perform milling operations on the **CR-1**. In this section we will provide an overview of the CRWrite software.

The repository for CRWrite may be found here: https://github.com/CoastRunnerCNC/crwrite-v1

Things CRWrite Does:	Notes:
Creates a serial connection to the CR-1	When CRWrite is running, plugging in the CR-1 will automatically open and maintain a connection to the mill.
Tracks the CR-1's status and internal state	CRWrite tracks and displays all data returned from grbl, including status (Idle, Alarm, etc.) position, modal state and more.
Guided Mode: loads and provides an interface for running prepackaged "cutcodes" ("crproj files")	See **Section 4.1** to learn more about Guided Mode.
Manual Mode: provides a serial terminal, jogging controls and a DRO for direct mill control	See **Section 5.1** to learn more about Manual Mode.

Facilitates updating the CR-1's firmware	See **Section 1.6** to learn more about firmware updates.
Allows the use of special "macros" such as M100, M101, M102	These special macros are only supported in CRWrite. See **Section 9.1.6** to learn more about them.
Implements a "Probing Wizard" to facilitate the creation and execution of probing sequences	See **section 4.6.5** to learn more about the Probing Wizard
Implements a File Creator to facilitate the creation and editing of crproj files.	File Creator will be detailed in a future version of the manual

Things CRWrite Doesn't Do:	**Notes**
Create G-code from a 3D model	Use CAD / CAM software such as Fusion360 to generate toolpaths and G-code for a 3D model

3.1.1: CRWrite Main Page

The CRWrite main page is the "homepage" that appears when you first launch CRWrite. It provides direct links to several

1. **Connection Status:** indicates whether the mill is connected or not. When CRWrite is open, connecting the CR-1's USB cable to the computer will automatically create a serial connection, which will update the connection status.

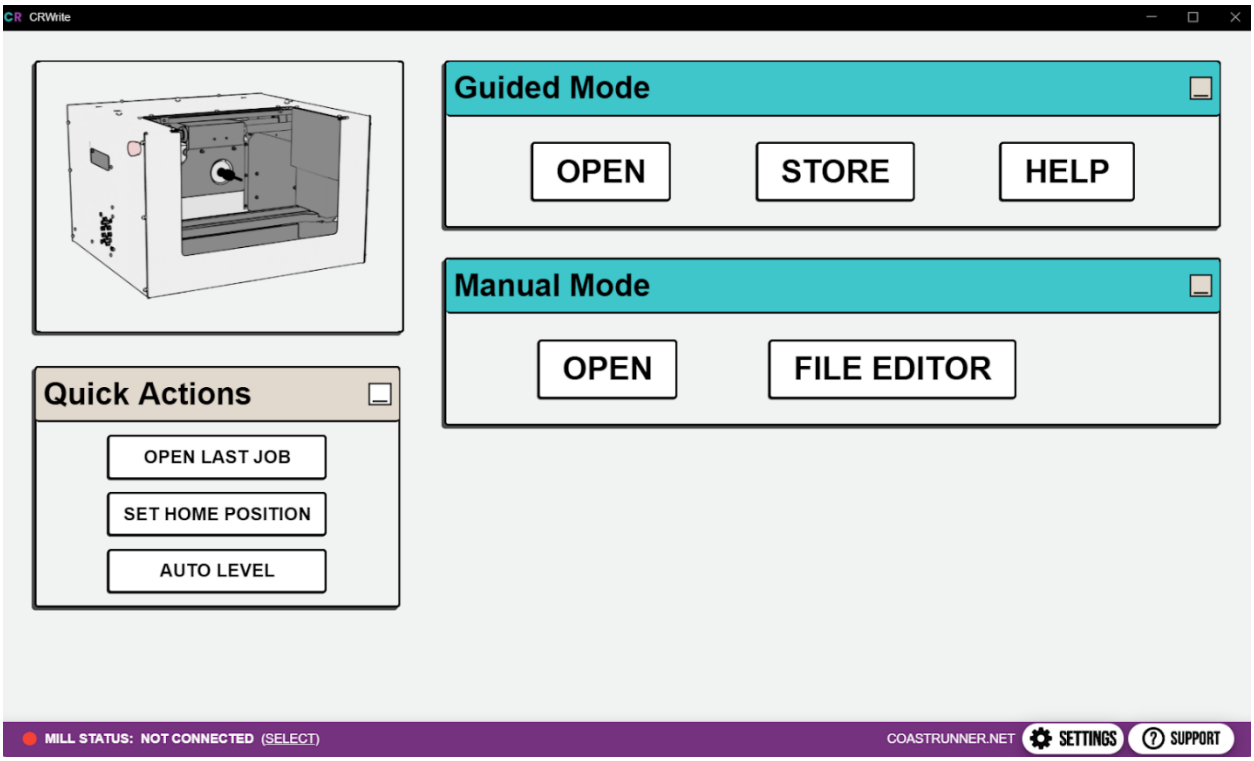
2. **Open Guided Mode**: opens a filepicker with which you can select a valid crproj file. Selecting a crproj file will launch the guided mode interface (see **Section 4.1**)

3. **Open Manual Mode:** opens the manual mode interface (see **Section 5.1**)

4. **Open Manual Mode**: manual mode can be opened at any time using this footer link, including from the Guided Mode interface.

5. **Settings**: opens the settings window, within which you can configure settings for CRWrite and the CR-1 itself

6. **Support**: opens the support menu, which you can use to get help with your milling project.



3.1.2: CRWrite Settings

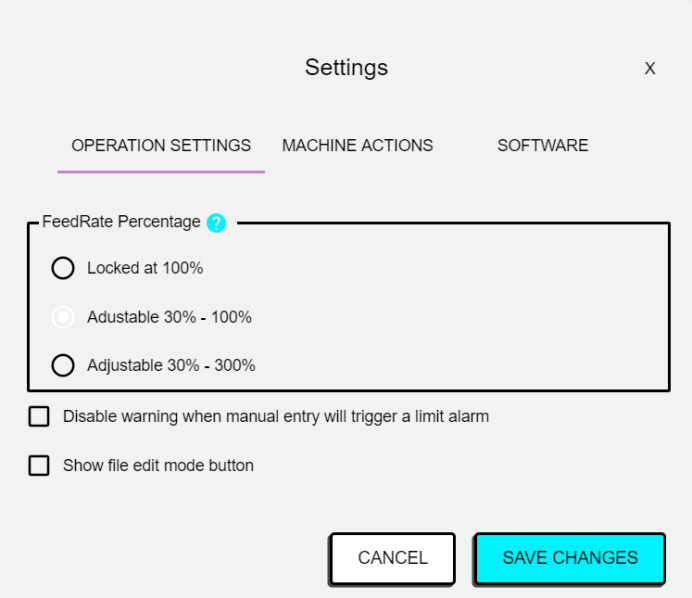
There are three sections within the Settings window: Operation Settings, Machine Actions and Software.

Machine Actions: consists of settings that affect guided mode or manual mode operation.

- **Feedrate Percentage**: enables or disables the feedrate override slider and determines the range that slider can travel. More information about the feedrate slider will be provided in **Section 4.1.**

- **Disable Warning when Manual Entry Will Trigger a Limit Alarm:** when unchecked, CRWrite will attempt to intercept manual commands that it thinks will trigger a soft limit; a popup will appear confirming whether the user really wants to attempt command execution. Checking the box disables this warning.

- **Show File Edit Mode Button**: only available for some CRWrite installations. When checked, enables an edit mode functionality for crproj files.

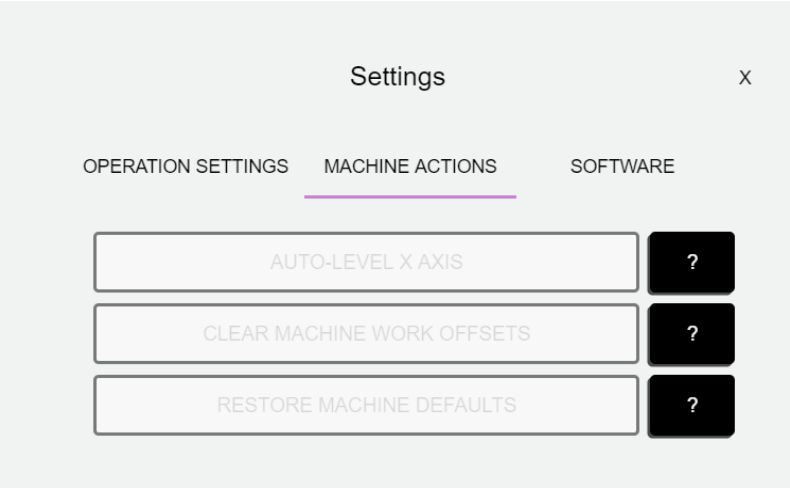


Machine Actions: allows you to perform certain actions related to the CR-1. Note that if CRWrite is not connected to the CR-1 mill, these buttons are disabled.

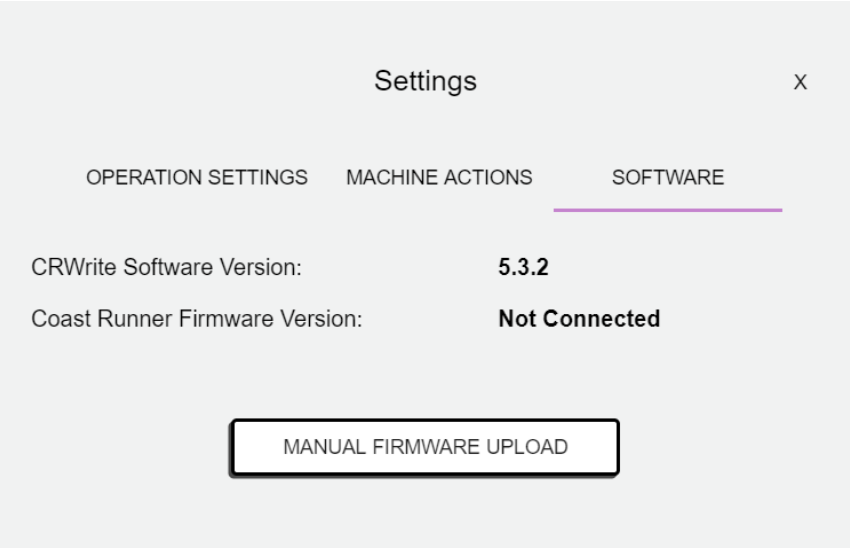
- **Auto-Level X Axis:** runs a command ($L) to perform a bed auto-leveling sequence.

- **Clear Machine Work Offsets:** runs a command ($RST=#) to zero out all work coordinate offset (WCS) registers. Learn more about WCS in **section 5.2.2** and about the $RST=# command in **Section 9.2.1**.

- **Restore Machine Defaults**: runs a command ($RST=*) to reset all grbl modal parameters to their default values. Also zeros out the WCS. Learn more about grbl modal state in **Section 5.3**, and about the $RST=* command in **Section 9.2.1**.



Software: contains information and functionality related to the CRWrite software and grbl firmware. Firmware information and functionality is only available if CRWrite is connected to the CR-1 mill.



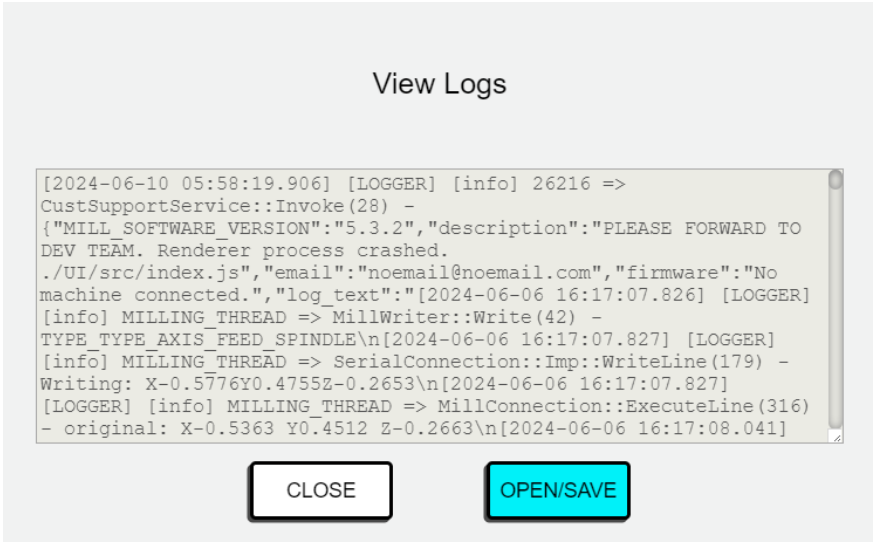
- **CRWrite Software Version:** shows the current version of the CRWrite software.
- **Coast Runner Firmware Version**: shows the current version parameters of the current CR-1 firmware.
- **Manual Firmware Upload**: allows you to manually update the 328p and / or the 32M1 firmwares. For more information on updating the firmware, see **section 1.6.**

3.1.3: CRWrite Support

There are two options in the Support menu:

How-To Walkthrough: launches a set of walkthrough popups that takes the user through the basics of the CRWrite interface. This walkthrough will automatically run the first time the user opens CRWrite.

Show Logs: opens a popup containing the current CRWrite log file. CRWrite generates logs related to sending and receiving G-Code, and to CRWrite operations generally. You can click “Open / Save” to dump the log file to a location of your choice.


View Logs
[2024-06-10 05:58:19.906] [LOGGER] [info] 26216 =>
CustSupportService::Invoke(28) -
{"MILL_SOFTWARE_VERSION":"5.3.2","description":"PLEASE FORWARD TO
DEV TEAM. Renderer process crashed.
./UI/src/index.js","email":"noemail@noemail.com","firmware":"No
machine connected.","log_text":"[2024-06-06 16:17:07.826] [LOGGER]
[info] MILLING_THREAD => MillWriter::Write(42) -
TYPE_TYPE_AXIS_FEED_SPINDLE\n[2024-06-06 16:17:07.827] [LOGGER]
[info] MILLING_THREAD => SerialConnection::Imp::WriteLine(179) -
Writing: X-0.5776Y0.4755Z-0.2653\n[2024-06-06 16:17:07.827]
[LOGGER] [info] MILLING_THREAD => MillConnection::ExecuteLine(316)
- original: X-0.5363 Y0.4512 Z-0.2663\n[2024-06-06 16:17:08.041]
CLOSE
OPEN/SAVE

3.2: grbl Overview

When we say that CRWrite "talks to CR-1" or something to this effect, what we are actually saying is that CRWrite is *talking to grbl*.

CRWrite's communication with grbl consists of sending commands via the USB cable to the grbl software and receiving responses. The commands sent by CRWrite do things like move the mill or request status updates; the responses report successful movements, errors, or mill status.

Note that CR-1's version of grbl is derived from GPLv3-licensed third-party software, the repository for which can be found here: https://github.com/gnea/grbl.

Coast Runner's version of grbl is forked from grbl1.1g, and may be found here: https://github.com/CoastRunnerCNC/coastrunner_cr1_firmware_328p

To learn more about **grbl**, we recommend reading the following resources:

- The remainder of Section **3.2** (this section!)
- Sections **9.1** and **9.2** of this manual
- The **grbl wiki:** https://github.com/gnea/grbl/wiki ***- reading and understanding is the really the best way to learn grbl***

Note: this section touches only lightly on a relatively advanced subject. The information presented in this section is limited to that which directly affects your experience in CRWrite.

Understanding grbl is necessary if you wish to write your own code, or if you wish to operate the CR-1 beyond guided mode, but we do not attempt to cover grbl it in-depth here. **Use the other resources (especially the grbl wiki) to learn more about it.**

3.2.1: Communicating with grbl

The easiest way to send commands to grbl is to let CRWrite do it for you in **Guided Mode**. You can also communicate with grbl more directly in **Manual Mode**, sending commands either one-at-a-time or in bulk by uploading a G-Code file.

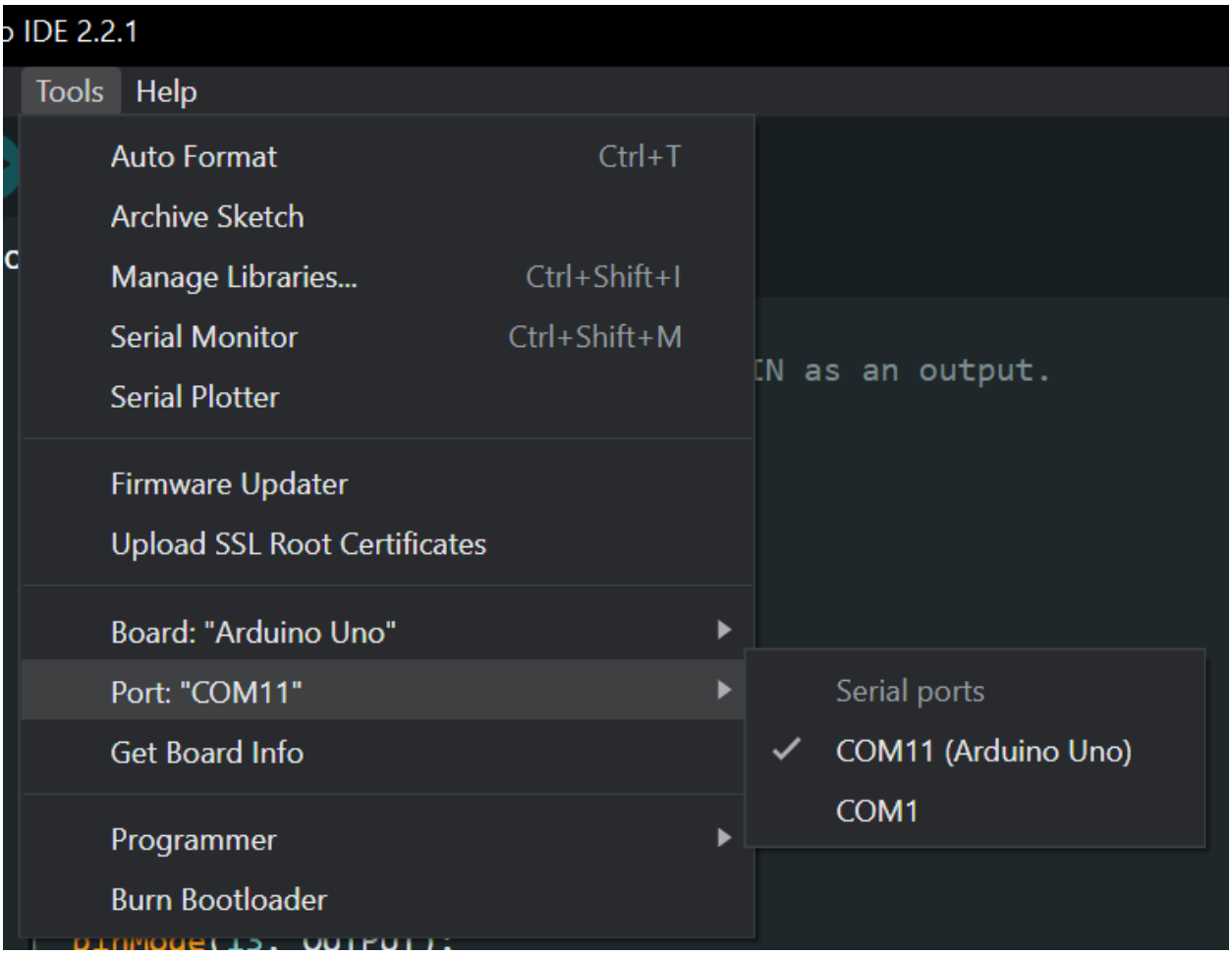
This said, you can also bypass CRWrite entirely and connect directly to grbl using your own serial terminal!

When you connect the CR-1 to your computer via a USB cable, your computer will create a virtual COM port to handle communications. There are several serial terminal softwares that can handle sending and receiving communication over this virtual COM port. One of the easier softwares to use for this is **Arduino Studio**.

Follow the instructions below to connect to grbl using Arduino Studio:
Note: these instructions are for Arduino Studio 2.X. If you are on a 1.X version, we recommend you update.

1. Connect the CR-1 to your computer via the USB cable.

2. Open Arduino Studio and ensure that the **Board** and **Port** options are correctly configured under the **Tools** menu option.



Normally these should auto-configure to their correct values, but if they don't, configure them manually.

3. Choose **Tools > Serial Monitor** from the main menu to launch the serial terminal, which will open at the bottom of the software.

   The baud rate must be set to **115200**. If it isn't set to this, you won't get proper communication.

4. Once the **serial monitor** is open and the baud rate is set properly, you should automatically see grbl's welcome message, which is pictured above.

   You can then begin sending commands to grbl by typing them in the Messages window. Review **Sections 9.1** and **9.2** to learn more about which commands you can send to grbl.

Note that most of the info in this section (states, buffer, etc.) is handled automatically by CRWrite in guided mode. This information is necessary to know only for users wanting to interact more directly with the mill, e.g. as part of new code development.

3.2.2: grbl States

grbl is stateful and can be in one of several different states at any given time. The state both reflects what is currently happening and determines what commands, if any, grbl will accept.

grbl's state can be determined by sending "**?**" real time command, which will return the state in the status string.

When grbl starts up (i.e. a new serial connection is made), it always begins in an **Alarm** state, in which no commands will be accepted except the **home ($H)** and **unlock ($X)** commands. This **Alarm** state can only be exited by running one of these two commands. The idea is that, on startup, you should be required to **home** the machine so it can return to a known location before starting to mill.

Other pertinent states include:

- **Idle**: the machine is idle, awaiting commands
- **Cycle**: the machine is running code, but is accepting new commands.

There are other states as well. See **Section 9.3** for details.

3.2.3: The grbl Buffer

Under normal circumstances, when a command is sent to grbl, you will immediately receive a response indicating either an **error** (due to e.g. a typo, etc.) or an **"ok"** message. The **"ok"** message indicates that grbl has *accepted* the command, and has placed it in its ***look-ahead planning buffer***.

When grbl is not in an error, alarm or pause state, it will execute each command in its planning buffer sequentially in the order received. As soon as one command has finished execution, the next command will be popped off the buffer and execution will initiate.

This buffer can hold up to **16 G-Code commands**. Tools like CRWrite have algorithms to keep the planning buffer "full" – when executing a G-Code file, CRWrite tracks how many commands it has sent and how many responses it has received, and based on this information the software "knows" when it must wait before sending additional commands.

When communicating with grbl directly via serial terminal, it is also possible to place multiple commands on the buffer, although this is unlikely unless you are executing many slow commands.

The fact that grbl contains an internal buffer which CRWrite attempts to keep full explains certain behaviors you may encounter, such as the fact that feedrate override changes in CRWrite do not apply immediately. These changes apply only to new commands entering the buffer, and depending on what commands are already *in* the buffer, it may be some time before the overridden commands start executing.

Section 4: Basic Operation

This section covers the basics of operating the CR-1. Primarily, this covers the use of Guided Mode, and the basic operations that you'll be asked to perform in Guided Mode.

4.1: Guided Mode

"Guided Mode" is a mode of operating the mill in which a user can execute defined "cutcodes" (also called "crproj files") in order to perform a particular milling job. Guided mode will walk a user through every step of performing a milling operation, including installing the tools, installing the workpiece, indexing, running cutting code, reorienting a workpiece, and so on. Running cutcodes in guided mode is a great way for new machinists to get familiar with milling, and for machinists of any experience level to perform milling as easily as 3D printing.

4.1.1: Cutcodes / .crproj Files

A "crproj file" is a specially formatted file with a ".crproj" file extension (for example, Universal_Clamp.crproj) which can be loaded and executed in CRWrite's Guided Mode. We also use the term "cutcode" to refer to these files.

A cutcode is defined for a specific milling operation – for example, the creation of a universal clamp. Each cutcode can contain the following:

- An organized set of steps to walk a user through every step of performing the milling operations
- Descriptive text for each step, telling the user exactly what to do or expect for this step
- Images illustrating each step
- The actual G-Code files to perform mill operations such as cutting, probing, moving to tool installation positions, etc.
- Additional files (such as 3D printable fixtures) to facilitate the milling operation.

Cutcodes created by Coast Runner, or by the Coast Runner community, can be downloaded from coastcad.com. We encourage you to visit this website and check out the projects!

4.1.2: Launching Guided Mode

Guided Mode can be launched in one of two ways:

1. Click the "Open" button in the Guided Mode section on the CRWrite homepage. This will open a filepicker. Selecting a valid .crproj file in this filepicker will launch the Job Selection popup, and upon selecting a job, Guided Mode will launch for that project and job.

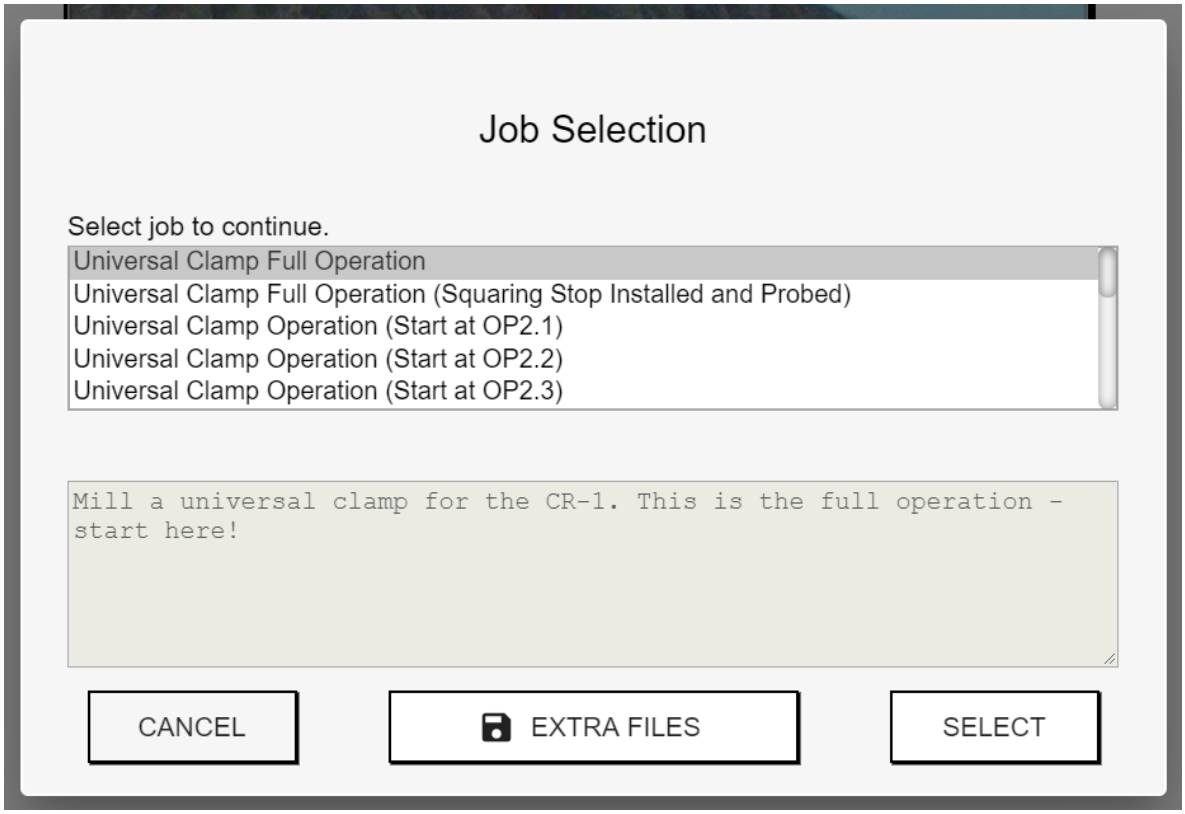
2. You can also launch Guided Mode simply by double-clicking a .crproj file's icon in your file explorer. This will launch CRWrite (if it isn't already running) and immediately open the Job Selection popup.

4.1.3: Guided Mode Interface

In this section we will cover the various interfaces that make up Guided Mode.

4.1.3.1: Job Selection Screen

The **Job Selection Screen** is the first section that comes up when you open a .crproj file. It allows you to choose between one or more different "jobs" defined within the file. You can select any one of these jobs and then press the "Select" button to initiate execution.



Regarding Jobs

A **job** should be thought of as a conceptually discrete set of steps to perform a certain milling task, or set of milling tasks. Here are some examples of how **jobs** can be used:

- The most basic **crproj** file contains only a single **job** that accomplishes the entire project represented by that crproj file. For example, a crproj file that mills a Turner's Cube could have a single job to perform the entire cube milling process.

- **Jobs** can be used to provide variants of different behaviors. For example, a **crproj** file that engraves different designs into a Zippo lighter may have five different jobs, each of which engraves a different design.

- Finally, **jobs** can be used to provide different "**entry points**" into a larger (and longer) project. As seen in the screenshot above, there is one job for the entire Universal Clamp milling process – that job contains all milling steps.

  Other jobs start that full operation later in the process – the idea being that, if you have to stop milling at some point, you can later come back, open up the crproj file again, and choose a job close to where you left off.

Each **job** has a **name** and a d**escription**. The description is displayed in the textbox beneath the job list.

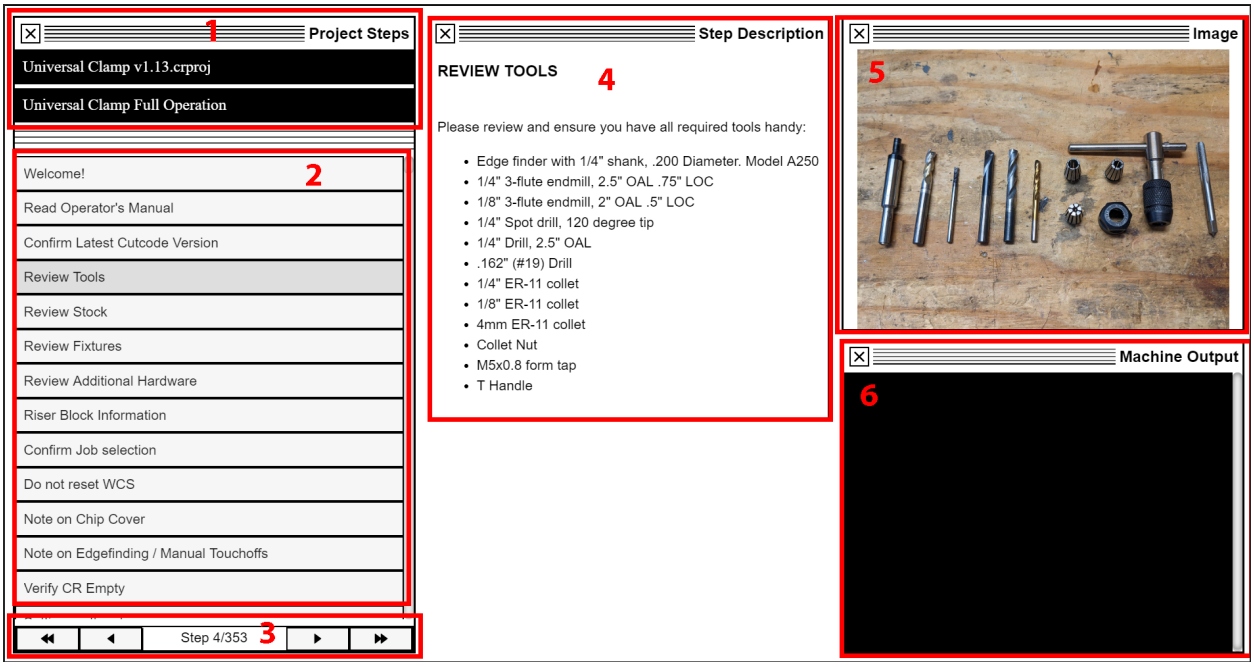
Extra Files

Some **crproj** files are bundled with additional files, such as 3D models that can be printed to create parts or fixtures that you can use during milling. When these "additional files" are present, the **Extra Files** button will be displayed, as seen in the screenshot above.

Clicking this button will open a filepicker to allow you to select where to extract the files to. Once extracted, you can review and use the bundled files.

4.1.3.2: Guided Mode Execution Screen

The **Guided Mode Execution Screen** is loaded after you select a **job.** It contains several sections that will assist you as you walk through a milling project. The screenshot below illustrates what this screen looks like. We have highlighted several different sections.



4.1.3.3: Job Information

The section labeled **1** is the **Job Information** window. It shows the **crproj filename** and **job name** that you are currently executing.

4.1.3.4: List of Steps

The section labeled **2** is the set of steps contained within this **job.** The step you are currently on is highlighted. You can scroll up or down this list to see what is coming – but be aware, **you cannot select a step to jump to it**. You can only move between steps using the **step controls**.

4.1.3.5: Step Controls

The section labeled **3** is the **step controls**. The controls are as follows, from left to right:

- **Exit** (Rewind): clicking this will give you the option to exit the job or crproj file.
- **Back:** clicking this will take you back one step. You cannot go back to a milling step, meaning that you can only go as far back as the step following the last milling step.
- **Step Info:** the step info section (in the center) displays which step you are on and how many steps there are in the job
- **Next:** clicking this will take you forward one step. If you are on a milling step, clicking this will show a warning (seen below) confirming you are ready to run the cutting code. Clicking "Start" in this warning will cause the mill to execute G-Code.
- **Skip Forward** (Fast Forward): clicking this will skip ahead to the next milling step. Milling steps cannot be skipped, but you can skip all the info steps in between.

This warning box will pop up every time the mill is about to run code.

Order of operations is critical in CNC. For a job to run successfully, all of its milling operations must run in the exact order specified by the developer who created the job.

This is why we place tight controls over how you can move through a crproj file – we do not allow you to jump back and repeat previous milling steps, to run steps out of order, or to skip milling steps as you advance.

4.1.3.6: Step Details

The section labeled **4** contains details about the step we are currently on. This includes the **step title** and **step description**.

Be sure to read these thoroughly! Each step contains important information which you must follow.

4.1.3.7: Step Image

The section labeled **5** contains the **step image**, an illustrative image providing details about the current step.

Not all steps have images.

4.1.3.8: Machine Output

The section labeled **6** contains the **machine output**. If the present step is a milling step, once you begin executing code in that step, the output will display in this section.

Nothing will display in this window until code starts executing. Obviously, if the current step is not a milling step, then nothing will display here.

4.2: Homing

A CNC mill is a precision tool, and the CR-1 is capable of holding to 0.001" of precision. This means that its movement is very accurate.

In order to achieve this accuracy, however, the mill must know *where its spindle is located in the first place*. This is what **homing** is for.

Homing is an automatic process in which the mill moves all three of its axes until a **limit switch** on each axis detects that the axis has reached its extreme end.

CR-1's homing procedure is as follows:

1. The mill will begin by homing the **Z axis**. The spindle will retract all the way to the back of the mill. Once it reaches the back, it will plunge slightly and then retract again.

2. The mill will then simultaneously home the **X** and **Y axes**. The table will rise all the way to the top, and the gantry will move all the way to the right. Both of these, upon reaching the end, will briefly move in the opposite direction before returning to the extreme end.

The mill "knows" it has reached the end of each axis due to a metal tab mounted on the axis tripping a limit switch for that axis. The brief "adjustment" at the end of each axis involves the mill both "double checking" its home and then positioning the axis a small distance (0.5mm) away from its actual extreme end.

Upon completing homing, the mill will assign the home coordinate position (X: -78.5, Y: -0.5, Z: -0.5) to the spindle's physical location. All future movement will therefore be based off this position.

The order of operations in this procedure – retracting the tool first, and only afterwards homing X and Y – ensures that, so long as no tools are installed beyond the recommended length of 3", the homing procedure cannot crash the tool.

How to Home

In **guided mode**, homing is almost always performed at the beginning of each job, as well as multiple times throughout the job. **If you are running guided mode, you don't really have to worry about homing.**

You can manually perform a homing cycle in **manual mode** by either clicking the **Home button** in the **Position Preset** section, or by sending a **$H** command to grbl.

You can also home **each individual axis** by issuing a special home command:

- **$HX** – homes the X axis
- **$HY** – homes the Y axis
- **$HZ** – homes the Z axis

4.3: Autoleveling

A critical part of milling is the concept of **squareness**. In order for your cuts to be accurate, your mill must be **square** – all three axes must be exactly orthogonal to one another.

The CR-1's **Y axis** (parallel to the **bed**) is its longest axis, and therefore has the greatest potential for getting out of square. Unlike the other axes, the table is also controlled by two stepper motors.

We therefore added an **autoleveling function** to the table to enable it to **square itself** (or return to square).

Autoleveling is automatic:

- In **guided mode**, an autolevel procedure is usually performed at the start of each cutcode, as well as intermittently throughout the code.

  If you see the table "moving up and down" as part of the early steps of a cutcode, this is almost certainly an autolevel procedure.

- In **manual mode**, an autolevel procedure can be executed by issuing a **$L** command to grbl.

Because this process is automatic, you don't need to know how it works, but if you're interested here is how it happens:

1. A **$HZ** home is performed to move the tool out of the way
2. The table is raised until one of the two limit switches triggers.
3. The stepper motor whose limit switch triggered is disabled; the other stepper motor continues raising its own side of the table until its own limit switch triggers.
4. An offset is calculated between the two triggering positions.
5. This offset is compared to an offset calculated at the factory and stored in grbl's memory.
6. Based on the comparison, the table's level is adjusted slightly.
7. This is repeated two more times, each time theoretically producing increasing accuracy.

grbl will report each of the three calculated offsets as it proceeds. You can see these in the logs.

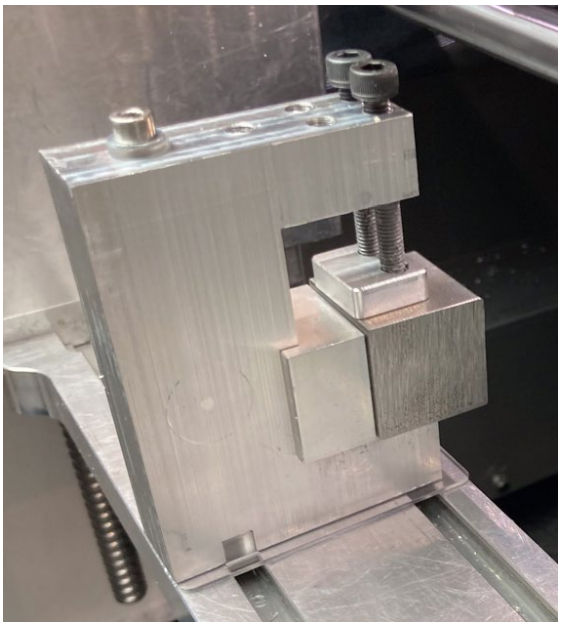
The **factory offset** can be modified by the end user. This is an advanced procedure and will not be covered in this section. More detail is provided in **section 9.2.1**.

4.4: Fixtures: Attaching the Workpiece

Your workpiece must be securely fastened (or "fixed") to the CR-1's bed during cutting operations. Failure to do so may result in the workpiece shifting during operation, which may lead to inaccurate cuts, poor cut quality, bogging and other undesirable behavior.

Any equipment used to help secure the workpiece to the bed is called a *fixture*.

Note: while *fixture* is the correct term for this equipment, the term *jig* is commonly used in the machining world to refer to these things as well. This is technically not the correct term - *jig* correctly refers to any equipment that facilitates *moving/guiding* a tool or workpiece during cutting operations. This said, the incorrect use of *jig* is ubiquitous enough to warrant mentioning it here.



In this image we see both a jig and a fixture. The aluminum universal clamp is a fixture*, holding the steel cube in place and preventing it from moving while milling. The aluminum "parallel" sitting between the steel cube and the clamp throat is a* jig *– it helps position the cube properly, but it does nothing to hold the cube in place.*

Examples of fixtures include:

- **Basic C-clamps**: one of the simplest fixtures. Uses clamping force in the X dimension to hold a part in place directly against the bed. Provides no electrical insulation by itself, so unless additional measures are taken, workpieces fixed with C-clamps will need to be probed manually.

- **Coast Runner universal clamp**: bolts to the t-slot plate, uses clamping force in the X dimension to hold a part in place. Part can be squared against the clamp throat and is elevated off the bed a small amount. Sized for use with the CR-1. Clamps (and part) are electrically isolated from the bed.

- **Coast Runner Jig Plate**: a “sub-plate” that sits between the bed and the part, in this configuration the plate is bolted “flat” to the t-slot plate. Threaded holes can accept additional clamping systems (tiedowns, toe clamps, etc.) to provide different clamping systems as needed.

- **Coast Runner Jig Plate (tombstone)**: the Coast Runner Jig Plate can also be turned 90 degrees and “stood up”, effectively turning the “bed” 90 degrees in relation to the spindle. This is called a “tombstone” and is highly useful for milling flats.

- **Machinist’s Vise:** the vise is secured to the bed / t-slot plate and then provides clamping force in the Y dimension.

- **Custom Clamshell:** oddly shaped parts (or parts that must be held at an unusual angle) can be secured in a custom “clamshell”, which is itself inserted into another fixture (universal clamp, etc.)

4.4.1: Making / Acquiring Fixtures

The selection and creation of appropriate fixtures is one of the most important parts of milling. Fixtures can be purchased from Coast Runner or from third parties. Coast Runner also provides several cutcodes for milling fixtures on the CR-1 itself, and we encourage end users to consider making their own fixtures for their particular needs.

Fixtures should be strong enough to provide adequate clamping force, should be rigid enough to not bend / warp during cutting, and should be resistant to the heat generated during milling. For this reason, it is often appropriate to make fixtures out of metal or industrial plastic (e.g. Delrin, etc.).

Under some circumstances, some fixtures (or parts of fixtures) can be 3D-printed. Fixtures made from 3D-printed plastic are inherently weaker than those made of stronger materials, but these can still provide successful results

4.4.2: Fixtures and Indexing

In addition to securing your workpiece, fixtures can also be used for locating / indexing (see **section 4.6** to learn what indexing is). If a fixture holds or otherwise locates a workpiece in a repeatable position, then the fixture can be probed as a way to index the workpiece; similarly, a fixture can provide a repeatable index point that need not change between operations / workpiece orientations.

(Image: show how you can probe a fixture to probe the workpiece)
(Image: show how a squaring stop can create a repeatable index point. Text: “after probing this fixture edge, we can reuse that index point for all future stock installations.)

4.4.3: Working with Fixtures

When using guided mode, CRWrite will inform you when and how to install your workpiece in a fixture. Specific steps will be provided depending on the fixtures, but usually the following steps will apply:

- **Attaching the fixture to the bed:** if your fixture engages with the Coast Runner's t-slot plate on the bed, you will need to install t-slot nuts and screw the fixture down into those nuts

- **Positioning the fixture:** often you will be asked to position the fixture on a specific place on the bed. **Pinstop** operations are used to help with this.

- **Positioning / securing stock in fixture:** once the fixture is secured to the bed, you will need to install the stock in the fixture. There may be information provided on specifically how or where the stock should be secured – keep an eye out for info on the desired orientation and position in the fixture.

4.5: Installing Tools

You must use the right tool for the job for all mill operations, including indexing, cutting, pinstop and more. The **CR-1** does not have an automatic tool changer (ATC) and therefore you will have to install or change tools manually.

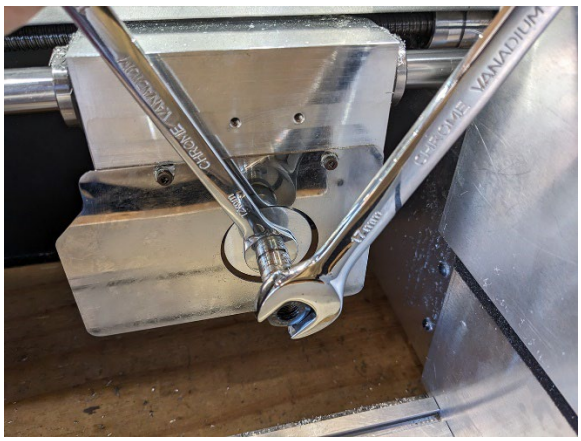
When running in Guided Mode, you will be instructed how to install or change tools. In Manual Mode, you'll need to change tools on your own.

The following steps detail how to install or change tools:

Remove Old Tools

If a tool is installed in the spindle already, we must remove it.

1. Pull the **chip fan** off the spindle, if present.
2. Place the **12mm wrench** on the slots in the spindle, and place the 17mm wrench on the collet nut.
3. Use the **17mm wrench** to loosen the nut; hold onto the 12mm wrench to prevent the spindle from turning.
4. Once the nut is loosened, you can unthread it with your fingers.
5. Pull the tool out of the loose collet, then fully unthread and remove the collet and nut.



Install Collet in Nut

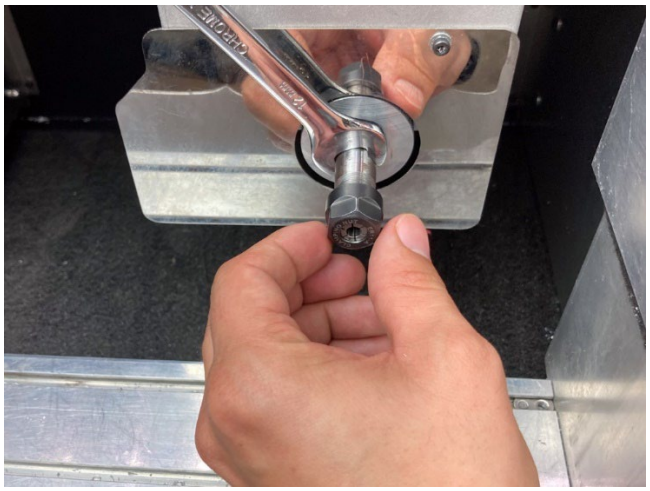
The CR-1 uses ER-11 collets. ER-11 collets have a long tapered end and a wider flat end. The wider flat end should be inserted into the hollow of the collet nut and snapped into place, like in this photo:

It is very important to properly snap the collet nut in. Insert the nut at a slight angle and then push it until it clicks. **Failure to snap the collet nut in properly can cause the tool to walk out during the cut, which will cause cutting problems and tool breakage.**

SERIOUSLY, MAKE SURE YOU INSTALL THE COLLET PROPERLY!
The most common problems we see are due to improperly installed collet nuts.
MAKE SURE IT SNAPS INTO PLACE. Turn the collet upside down and try to shake it out – if it falls out, it wasn't installed properly.

Loosely Thread Collet Nut

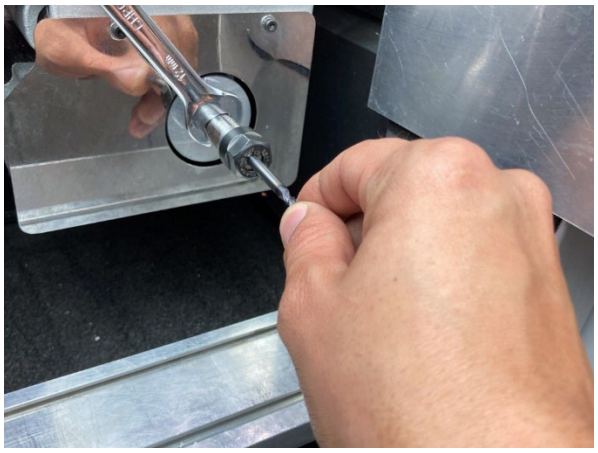
Insert the collet nut (with collet installed) onto the spindle and thread it by hand a few turns. You don't need to thread it any more than this.



Install Tool and Finger Tighten

Take your tool and place it into the collet. *Make sure to insert it such that the cutting end sticks out.*

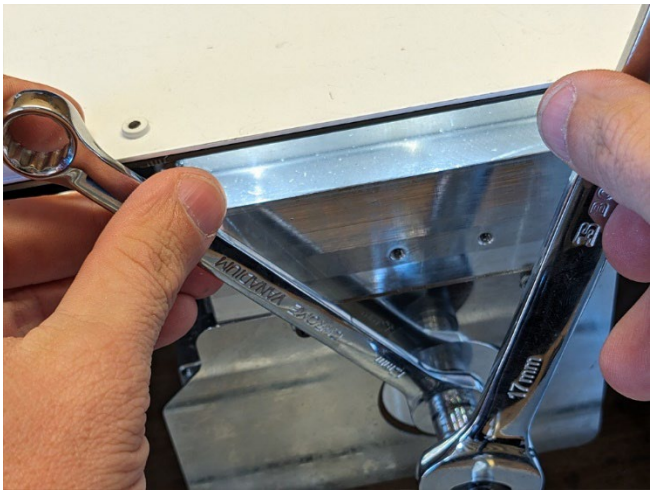
Push the tool all the way to the back of the collet. This is about 0.9” deep. It’s important that the tool is fully seated to ensure maximum holding strength.



While pushing the tool fully back into the collet, **finger tighten the collet nut** until you can’t tighten it any further. This should get the tool snug enough that you can’t pull it out.

Tighten with Wrenches

In most (but not all) cases you’ll want to tighten the nut even further using the wrenches. Place them in the same spot as before and tighten the collet nut until you cannot easily tighten it further.



How tight should I go? We say that you should tighten it until you cannot easily tighten it any further. This means: if you can still tighten it without excessive force, you should do so.

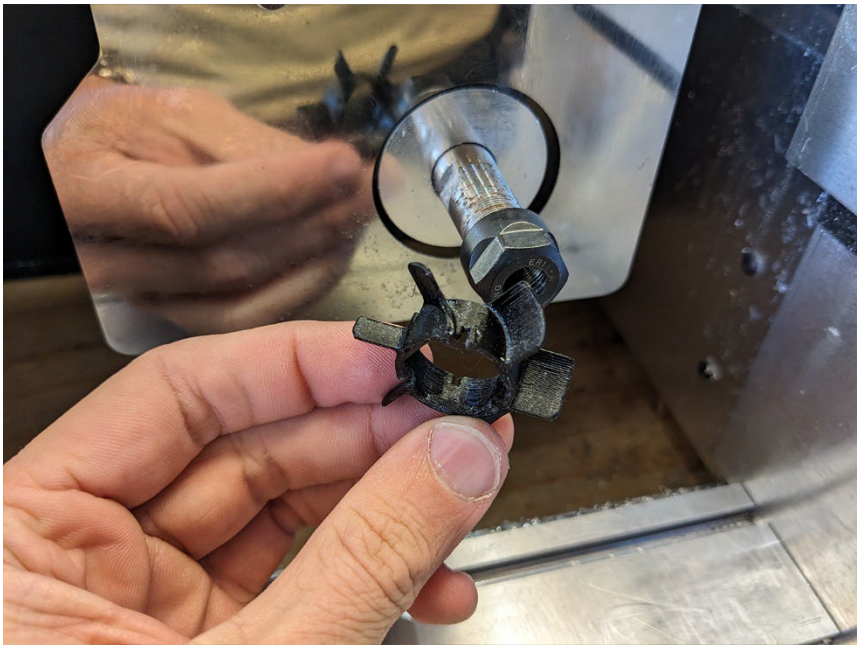
However, if you feel that any tighter would require serious force, don't go any further. While it is hard to damage the spindle, it isn't impossible, so don't do it.

Install Chip Fan

Normally you'll want to install the chip fan over the collet nut. The chip fan helps to blow chips away during milling, which will help improve cutting performance.

The chip fan hole has a hexagon shape on one end and a circle shape over another. The hexagon should face *out* when you install it. This hexagon interfaces with the hexagon-shaped collet nut.

Slide the fan all the way onto the nut, as deep as you can. It should stop automatically when you've slid it fully in.



4.6: Indexing / Probing

In order for your CNC mill to cut properly, the mill must "know" where the part it is cutting is. The mill "learns" this information via a process called "indexing" or "probing". Indexing must be performed every time a part is installed (or its orientation is changed), and often every time a tool changes.

In this section we will simply cover common methods of how indexing is performed via CRWrite. To learn more about the theory of indexing, check out **section 5.4**.

When it comes time to index a part in guided mode, one of the four following methods will (almost always) be used. You will be instructed on how to do this as part of guided mode, but they are described here as well.

All four of the following methods ultimately achieve the same goal: position the indexing tool such that it is *just barely* touching the edge or surface of the fixture or part being probed. The main differences between methods are how this is achieved. Each method is more or less suitable for different sorts of circumstances.

4.6.1: Edgefinding

Edgefinding involves using a tool called an **edgefinder** to touch the side of a part or fixture. The edgefinder tool we use (Fisher Model A250) is spun during the edgefinding operation, and this spinning causes the tool to make loud clicking noises when it makes contact with an edge. This audible indicator makes it very easy to know when the edgefinder is touching the edge.

Edgefinding is useful because it requires no electrical insulation or conductivity, making it usable in more circumstances. We've also found it to be slightly more accurate than conductive probing. However, it is much slower and more tedious, and cannot be automated.

Edgefinding involves the following steps:

1. Move the edgefinder near the edge you wish to probe
2. Begin jogging towards this edge in **1mm** increments
3. Once the edgefinder touches the edge, it will make a loud clicking noise.
4. Pull the edgefinder back from the edge 1mm, then switch jogging increments to **0.1mm**.
5. Begin jogging towards the edge in **0.1mm** increments. Continue until you touch the edge and hear clicking again.
6. Pull the edgefinder back from the edge 0.1mm, then switch jogging increments to **0.01mm**.
7. Begin jogging towards the edge in **0.01mm** increments. Continue until you touch the edge and hear clicking again.

8. The side of the edgefinder is now within 0.01mm – less than 1/1000" – from the edge. This is accurate enough for our needs.

9. Write the edgefinder's current location into memory by running a G10 command in the manual entry window.

   For example, if you are finding the left edge (left from the perspective of the spindle), use this command to write the location to the first register, G54:

   G10 L20 P1 Y2.54

   Notice the offset. The A250 edgefinder has a 0.2" diameter tip; half of this is 0.1" (2.54mm). We are edgefinding the left side (with respect to the spindle) which is the *positive* direction in Y, so the offset is positive.

Note: because edgefinders only work if used radially, they can only be used to index in **X** and **Y.** For this reason, edgefinding is almost always paired with **touchoff** indexing.

4.6.2: Touchoff

Touchoff pinches a piece of paper between the tool and a surface to tell us when the tool is properly touching the edge. As you carefully approach the surface with the tool, you wiggle the paper back and forth – once the paper gets pinched, you can no longer wiggle it, and as such you know you are touching the edge.

Touchoff requires no electrical insulation or conductivity. It can be performed with an actual cutting tool, allowing you to properly index your cutter without marring the part. However, it is very slow and tedious, and cannot be automated.

When using a cutting tool, touchoff is only accurate when probing Z – probing X or Y via touchoff is inherently inaccurate due to the irregular outer diameter of a cutting tool.

While you can perform touchoff in X and Y using a smooth rod, there isn't much reason to do so given that you could just use the edgefinder instead. Given all this, we recommend using touchoff only with cutters and only for Z.

Touchoff involves the following steps:

1. **Do not spin the tool.** Touchoff is performed with the tool not spinning.

2. Jog the cutting tool close to the surface you wish to touch off.

3. Begin jogging close to that surface in **1mm increments**. Stop jogging once you are within 1-2mm of the surface.

4. Insert a piece of paper between the tool tip and the surface. Begin wiggling the paper back and forth.

5. Switch jogging distance to **0.1mm increments** and begin jogging until the paper gets pinched. You can tell the paper is being pinched because you cannot easily keep wiggling the paper.

6. If the paper is fully pinched, back off 0.1mm. If it's only a little pinched (you can still wiggle a little bit) you don't need to back off here.

7. Switch to **0.01mm increments** and jog forward again while wiggling the paper until you cannot move the paper any further. Once the paper stops wiggling you don't need to keep pinching it further – stop just after you become unable to move the paper.

8. Write the tool tip's current location into memory by running a G10 command in the manual entry window.

   For example, use this command to write the location to the first register, G54:

   G10 L20 P1 Z0

   Note that no offset is necessary here – unlike when probing X and Y, there is no radius to worry about.

4.6.3: Conductive Probing

Conductive probing is one of the special features that makes the CR-1 convenient. In conductive probing, a probe wire is connected to the part or fixture being probed. The tool is then given a slight electric charge. This setup means that, when the tool touches the edge or surface, the circuit completes (the tool electrifies the part or fixture, which is sensed by the probe wire) and is automatically detected by the mill.

Conductive probing is highly convenient, as it is automatic, and when set up correctly is quite accurate. Conductive probing can also probe in all three axes.

However, it can only be used in limited circumstances:

- The part or fixture being probed must be conductive (or at the very least, a conductive path must be created between the probe wire and the surface being probed.)

- The part or fixture must be electrically isolated from the CR-1's bed

- If probing X or Y with a cutter, the cutter should be spun in order to present a relatively uniform radial surface. This means that, when the cutter touches the side of the part, slight marring will occur. This can be mitigated by spinning the tool backwards (M4).

Conductive probing is performed as follows.

Note that most of the steps below will be handled automatically in guided mode. We have highlighted all the steps that guided mode will handle for you in grey – you will only need to do anything for the non-grey steps.

Regardless, we present the entire process here for reference.

1. Plug in the probe wire to the LED board and connect the wire's spade terminal to the fixture or part being probed.

2. Position the tool near the surface being probed.

3. If probing X or Y, **spin the tool backwards**. No need to spin the tool if you are probing Z.

4. Check that the probe wire is not currently shorted to the bed. When shorted, the green LED on the LED board (next to the probe jack) will light up. It should not be lit up right now.

5. Execute a G38.2 command to begin moving the tool towards the surface.
   This command might look like: G38.2 G91 Y-20 F20
   *You can also use G38.3, G38.4 or G38.5. Refer to **Section 9.1.4** and **9.1.7** for more details.*

6. Assuming the probe trips, the tool will stop automatically and will now be just touching the edge.

7. Execute a **G10 command** to write current location to memory. Note that, just like with edgefinding, if probing X or Y you must apply an offset to your G10 to compensate for the tool's radius.

8. Pull the tool away from the surface and stop it from spinning.

Often, several different edges or surfaces are performed as part of a conductive probing step in guided mode. Sit back and watch the mill probe!

4.6.4: Touch Probing

Touch probing combines the "manual" probing techniques (edgefinding and touchoff) with conductive probing and achieves the benefits or both. A tool called a "touch probe" is used – this tool has a long "wand" which, when it touches an object, creates an electrical signal that can be sent into the CR-1's controller.

The touch here is purely physical – no conductivity or other method is used, meaning that the touch probe does not rely on conductive material or electrical isolation. Yet it is still automatic, providing the benefits of conductive probing.

At time of writing we have not created a touch probe for the CR-1, but we have one in the works and hope to offer support for it soon!

4.6.5: Probe Wizard

In **Guided Mode**, you will always be instructed on how to probe, and most probing tasks will be handled for you.

In the event that you wish to probe on your own, there are two ways you can do it:

1. Write your own **probe code**, which is covered in **Section 5.4**.
2. Use the **Probing Wizard**

The probing wizard has a convenient UI, which is why we have placed this information in the **Basic Operation** section.

Probing wizard details to be added in future versions of the manual.

Section 5: Advanced Operation / Direct Control

This section covers more advanced operation of the CR-1 – namely, creating your own code to perform cutting operations on the mill. It will focus on Manual Mode, and on the sorts of code that you can write "by hand" and run in Manual Mode.

Cutting code written using CAM software (the preferred way to generate cutting code) will be covered in **Section 6**. This software-generated code will also be run using Manual Mode. It is very uncommon to write actual cutting code "by hand" in the way we will describe in this section; "hand-written" code is instead used to "fill gaps" between cutting operations, e.g. for probing operations, tool installation, etc.

5.1: Manual Mode

Manual mode is the CRWrite interface that allows for more direct, traditional CNC mill control. It consists of a serial terminal, a DRO, control buttons, a jogging interface, a feedrate override slider, position presets, several sections representing mill modes and status, and functionality to upload and run G-Code files.

5.1.1: Launching Manual Mode

Manual Mode can be opened by either:

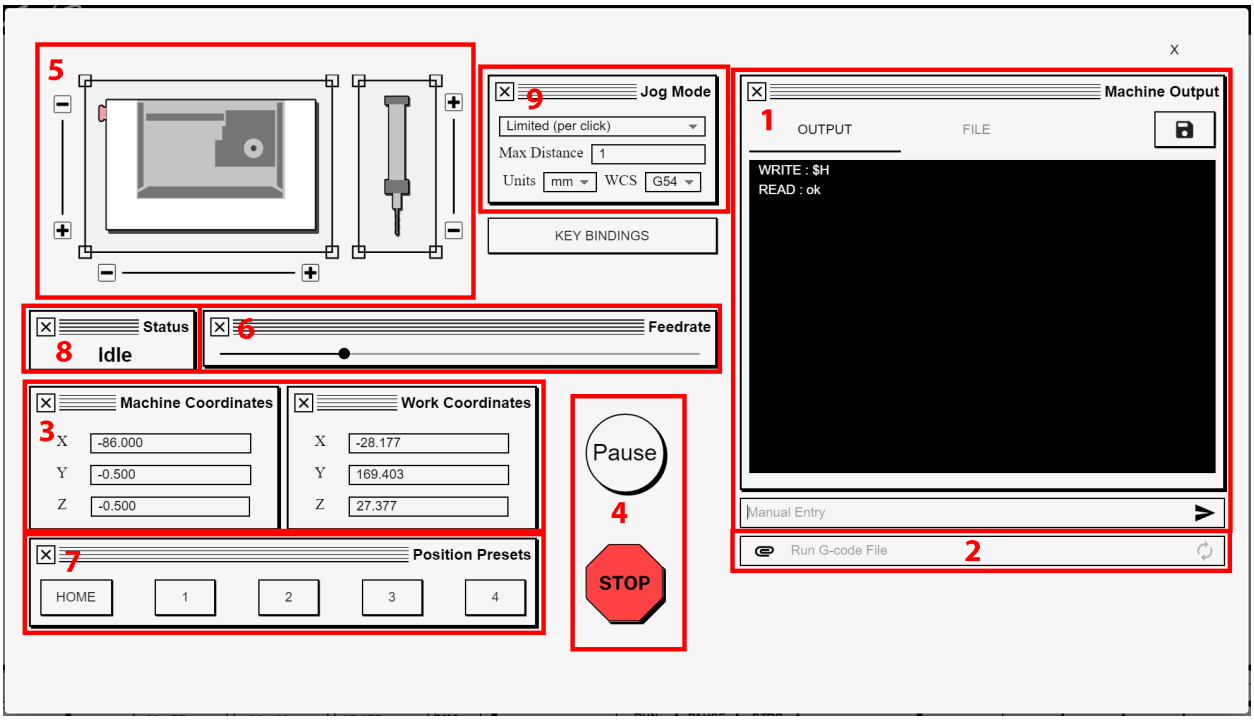
1. Clicking the "Open" button in the Manual Mode section on the CRWrite main page
2. Clicking the "joystick" icon (next to the Settings button) in the CRWrite footer. This icon will appear only when CRWrite is connected to a CNC mill.

In either case, Manual Mode will open as an overlay on the current screen.

If the connected CNC mill is in an ALARM state following an initial connection or reset, manual mode will provide a popup asking the user to home the machine.

5.1.2: Manual Mode Interface

This section will review the various parts of the Manual Mode interface.



5.1.2.1: Serial Terminal

This section is labeled **1** in the diagram above. The serial terminal provides for direct communication with grbl.

Operators can enter commands in the "manual entry" textbox directly beneath the large serial terminal display. Clicking the "send" arrow or pressing the Enter key will send this command to grbl.

The serial terminal display records all commands sent to grbl, and the responses received. It may also display additional logging information. Only information generated after opening the manual operations window will be displayed in the serial terminal display - commands, logs, etc. generated prior to opening the window are not shown.

5.1.2.2: G-Code File Upload

This section is labeled **2** in the diagram above. This interface allows you to select and submit a sequence of G-Code commands, contained in a single text file, to grbl. This interface is located directly beneath the "manual entry" textbox used for the serial terminal.

Clicking the paper clip icon opens a filepicker with which you can select a G-Code file. CRWrite allows you to select any file here - it does not limit you based on file extensions. As such, it is your responsibility to ensure that you select correctly formatted G-Code files.

Upon selecting a file, clicking the submit button (formatted as two arrows in a circle) immediately initiates sending the G-Code commands in the selected file to grbl.

These commands (and their responses and some related logs) will be recorded in the serial terminal display, just as if you were manually entering each one in the manual entry field.

5.1.2.3: DRO

This section is labeled **3** in the diagram above. The DRO (digital readout) displays coordinate information related to the spindle's current location. There are two sets of X/Y/Z coordinates: Machine Coordinates and Work Coordinates.

Machine Coordinates shows the coordinates in relation to the mill's home position. Work Coordinates shows the coordinates in relation to the zero position of the currently selected work coordinate system (WCS).

Both of these will update in real time as the mill operates.

5.1.2.4: Control Buttons

This section is labeled **4** in the diagram above. The control buttons allow you to quickly pause or stop the mill's movement. There are two buttons:

STOP: A stop button. Clicking it will immediately issue a real-time soft reset command, which will stop all motor movement (both X/Y/Z and spindle.)

After pressing the software stop button, grbl automatically enters an ALARM mode which can only be exited by either homing the mill or by issuing a $X (unlock) command.

Warning: do not rely on the software stop button for dangerous situations such as tool bogging, impending crashes or injury.

Software delay or slowdown may prevent the soft reset command from executing immediately. Use the physical hardware estop button for true emergencies; the software stop button should be used when the situation is not time-sensitive (example: running the wrong file, etc.)

PAUSE: The Pause button issues a feed hold command. Feed hold will stop X/Y/Z motion (but not spindle motion) and will hold the spindle in place until a resumption command is issued. All queued commands will remain in the execution buffer, and any incomplete command will be preserved.

While paused, no new commands will be accepted from grbl.

Clicking the button will change its label from "Pause" to "Run." Clicking the "Run" button will issue a resume command, and all queued grbl will process as normal.

5.1.2.5: Jogging Interface

This section is labeled **5** in the diagram above. The jogging interface allows you to issue jogging commands to the CR-1. Jogging is a special type of movement in grbl that is analogous to "remote control". It is a useful way to quickly move the spindle to (or near to) a desired location.

The jogging interface is located in the top-left part of the manual entry window. The main part of this interface is an illustration of the CR-1 with three arrows, each representing one of the three axes of movement:

- The arrow labeled 1 is the X axis, which moves the table up and down
- The arrow labeled 2 is the Y axis, which moves the gantry left and right
- The arrow labeled 3 is the Z axis, which plunges and retracts the spindle

Each arrow has a button at both of its extreme ends. The button is labeled with a plus sign (+) or a minus sign (-) representing the sign of movement in this direction.

Clicking either button on an axis arrow will issue a jog command for that action, in the direction corresponding to the button pressed.

5.1.2.6: Jog Modes

You can switch between **jogging modes** by changing the dropdown in the section labeled **9** above.

There are two jogging modes:

1. **Continuous Mode:** When jogging in continuous mode, motion will continue for as long as the button is held down, or until the mill reaches the end of its travel distance. Releasing the button will immediately halt motion.

2. **Limited Distance:** When jogging in limited distance mode, each jog command will move the machine for a specified distance, determined by the Max Distance (and units) inputs. Upon traveling exactly that distance, the jog will stop. Releasing the button early will immediately halt motion, even if the mill hasn't completed its specified distance.

5.1.2.7: Feedrate Slider

This section is labeled **6** in the diagram above. The feedrate slider allows you to apply a percent override to the mill's feedrate. By default it can be set between 30% and 300%.

Applying a percent override modifies the current feedrate (and all future feedrates) by that amount. For example, if the slider is set to 200%, then a command to set the feedrate to F1000 will be intercepted and changed to F2000. A future command that sets the feedrate to F500 will again be intercepted and changed to F600.

The feedrate slider is a useful way to slightly adjust the speed at which the mill is travelling.

5.1.2.8: Position Presets

This section is labeled **7** in the diagram above. The Position Presets box contains five buttons: a home button and four buttons labeled 1 through 4.

Home Button: Clicking the Home button issues a $H command to home the machine.

1-4 Buttons: these represent saved position data. Clicking a button with no position currently saved will give you the option (via popup) to save the current spindle position as a preset value.

Once set, the preset will be written to long-term memory. Clicking the button any time after that will home the machine and then move the spindle to the saved location - first in X/Y, and only after in Z.

If you wish to overwrite a saved preset, right-click on the button to bring up the same popup as that which you used to originally set the preset.

Presets are a useful way to store common positions that you may return to, such as tool change positions.

5.1.2.9: Mill Mode and Status

These items are located in the areas labeled **8** and **9** in the diagram above. There are several status indicators and dropdowns positioned around the manual operation screen. These report, and sometimes enable you to change, grbl's operating modes and modal parameters.

The indicators are as follows:

1. grbl Status: reports grbl's overall status. Some possible values are:
   - Idle
   - Cycle
   - Alarm
   - Jog
   - Hold

Review **Section 9.3** for more details on grbl statuses and what they mean.

2. Units: a dropdown that can be switched between mm and inches. Controls the format that units are displayed in CRWrite, and the way in which G-Code commands will be interpreted

   Changing the Units dropdown will issue either a G20 or G21 command.

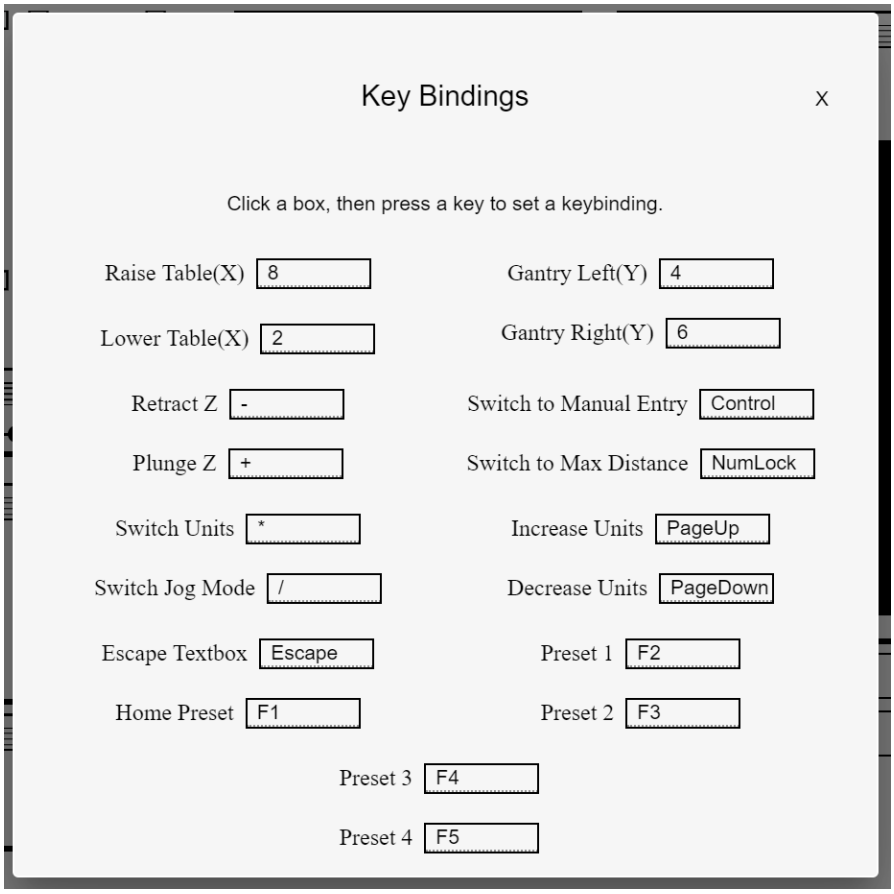
3. WCS: a dropdown that switches the active work coordinate system. Changing the dropdown will change which coordinates are displayed in the "Work Coordinates" subsection of the DRO.

   Changing the WCS dropdown will issue a G5[4-9] command corresponding to the WCS register

5.1.2.10: Hotkeys

The manual mode interface provides several **hotkeys** to facilitate ease of operation. Note that these hotkeys are only available when Manual Mode is open.

You can view the current bindings and rebind the keys by clicking the "**Keybindings**" button to open the Hotkeys popup. Within this popup, you can rebind the keys as desired.



The keybinding interface. Note that the keybindings shown here may differ from the defaults

The following hotkeys are available:

Jogging Movement Hotkeys:

- **Raise Table:** issue a jog command to move the table in the negative direction (up)
- **Lower Table:** issue a jog command to move the table in the positive direction (down)
- **Gantry Left:** issue a jog command to move the gantry in the negative direction (left)
- **Gantry Right:** issue a jog command to move the gantry in the positive direction (right)
- **Plunge Spindle:** issue a jog command to move the spindle in the negative direction (plunge)
- **Retract Spindle:** issue a jog command to move the spindle in the positive direction (retract)

Mode Control

- **Switch Units:** toggle between the units (mm or inches)
- **Switch Jog Mode:** toggle between jog modes (limited or continuous)
- **Increase Units:** increase the "max distance" units by one order of magnitude – e.g. go from 0.1 to 1. This is useful when performing edgefinding.
- **Decrease Units:** decrease the "max distance" units by one order of magnitude – e.g. go from 0.1 to 0.01. This is useful when performing edgefinding.

Interface Control:

- **Switch to Manual Entry:** force the cursor focus into the manual entry textbox
- **Switch to Max Distance:** force the cursor focus into the max distance textbox
- **Escape Textbox:** remove the focus from any particular textbox. This is useful if you wish to jog immediately following the entry of a manual entry command – jog commands will be interpreted as text characters if you are still focused on the textbox, so you'll have to remove focus on the textbox before you can jog. Hitting this hotkey allows you to do so without removing your hands from the keyboard.

Position Presets:
Note that you can only issue preset move commands – to rebind a preset you must still right-click the preset button.

- **Home Preset:** issue a home command
- **Preset 1:** moves to the first preset
- **Preset 2:** moves to the second preset
- **Preset 3:** moves to the third preset
- **Preset 4:** moves to the fourth preset

5.2: Writing Code: Movement

In this section we will cover basic G-Code operations. This will be from the perspective of writing G-Code manually, which is very useful for small operations like tool changes, pinstop, etc. When writing G-Code for actual cutting operations, we recommend using CAM software, which we cover in **Section 6**.

Any G-Code that you wish to execute manually can be executed in **Manual Mode**, either via single-command entry or via uploading a G-Code file. Review **Section 5.1** for more details on this.

This section of the manual is not meant to be a comprehensive tutorial for G-Code. For a fuller treatment, check out the **grbl wiki:** https://github.com/gnea/grbl/wiki

5.2.1: Basic Movement

We will begin by covering basic movement of the spindle.

5.2.1.1: G53 Movement

The easiest way to move the spindle in the **CR-1** is via **G53** commands. These commands perform **absolute movement** in the **machine coordinate system,** which is centered at the **machine home**.

A **G53 movement command** looks like the following:

G53 G0 Y-20

This command will rapidly move the gantry to coordinate -20 in the Y axis. Assuming we are operating in millimeter mode (G21), this is 20mm to the left of machine zero (20mm from the right of the home position).

There are three components to the command:

- *G53*: tells us that we are moving in **absolute machine coordinates**
- *G0:* tells us that we are moving rapidly (and sets modal state to G0)
- *Y-20*: tells the machine what coordinate to move to – in this case, -20 in the Y dimension

You can move in multiple axes at the same time:

G53 G0 X-70 Y-20

This will cause the gantry and table to move at the same time.

While it is possible to move in X, Y and Z simultaneously, we recommend when writing manual G-Code to always move in X and Y *first*, and only afterwards move in Z. This helps prevent crashing.

The machine coordinate system is defined as follows:

Axis	Min. Coordinate (Machine Home)	Max. Coordinate
X	-86.5mm	0mm
Y	0mm	-241.5mm
Z	0mm	-78.5mm

You can issue a **G53 move command** to move the spindle to any location in between the min and max coordinates for the axes. Attempting to move to a coordinate outside these limits will throw a soft limit alarm.

These axes use the following signs when moving:

Z Axis: Spindle Plunge is **NEGATIVE**, Spindle Retract is **POSITIVE**
Y Axis: Gantry Left is **NEGATIVE**, Gantry Right is **POSITIVE**
X Axis: Table Up (Spindle "Down") is **POSITIVE**, Table Down (Spindle "Up") is **NEGATIVE**

G53 moves are very useful for positioning the spindle / tool somewhere that is the same every single time – moves in G53 will always go to the same place (i.e. they always refer to an immutable zero position.) This makes them quite useful for **tool changes**, **pinstop operations** and **setup steps**.

For the same reason, however, we virtually never write cutting code in G53, because cutting code is almost always written in reference to a user-defined zero (probe point.)

Some technical notes on G53:

- **G53** movement is, by definition, always **G90** (absolute) movement
- Despite the above, running **G53** will not switch the modal state to **G90**. If modal state is **G91**, and you run **G53**, modal state will remain **G91** after the **G53** move.
- **G53** (unlike **G54, G55, etc.**) is not modal. You must include the **G53** verb with every intended movement command.

5.2.1.2: Relative Movement

If, instead of wanting to move to a specific coordinate defined in a coordinate system, you wish to simply "move left 20mm" or something like this, you can use relative movement to achieve this.

The following command will perform the relative motion described above:

G91 G0 Y-20

There are three components to the command:

- G91: tells us that we are moving in **relative motion** (and sets modal state to G91)
- *G0:* tells us that we are moving rapidly (and sets modal state to G0)
- *Y-20*: tells the machine to move 20mm left in Y

As with other movement commands, you can move in multiple axes at once:

G91 G0 X10 Y-10

This command will move the table down 10mm and the gantry right 10mm.

Relative moves are useful in certain circumstances, such as **pinstop** or **probing**. However, cutting code is almost never written using them.

G91 is modal, meaning that after running your first **G91** commands, all movement commands afterwards that lack a G90 component (except G53) will be interpreted as relative

5.2.1.3: Movement Speed

There are two "speed modes" that a G-Code move command can run in:

- **G0:** moves will be "rapid", e.g. all performed at a specific (fast) speed. G0 moves are meant for relocating the tool, not for actual cutting

- **G1:** movement will proceed at a user-specified feedrate (specified in an **F** command, e.g. **F1000**). G1 moves are generally used for cutting, since feedrate matters for good cutting performance.

You will notice that in our example commands so far we have always included *G0*, to ensure that the mill moves rapidly. If we instead wanted to perform a G1 move, we could do:

G53 G1 Z-20 F1000 – move to machine coordinate -20 in Z at a rate of 1000 mm/min

G91 G1 Y20 F20 – move 20 units to the right of the current position at a rate of 20 mm/min

The **G0** and **G1** commands are both modal, meaning that once you execute one of them, grbl will maintain that "speed mode" until told otherwise. This means you don't need to include a G0 / G1 with every movement command – only for the first one in a series.

5.2.2: Work Coordinate Systems

When we discussed **G53 moves** (**section 5.2.1.1**) we presented the concept of moving in absolute coordinates with respect to machine zero. But machine zero is not the only coordinate system we can move in.

In truth, we can set a "zero point" anywhere in the addressable coordinate space and use this to define a new coordinate system. A coordinate system defined in this way is called a "work coordinate system" or **WCS**.

The most common purpose for using a **WCS** is to define a coordinate system with respect to a zero point located on either the *stock being cut* or the *fixture* – for example, the top-left corner of a clamp, or the midpoint on the stock surface.

When generating cutting code using CAM software (see **section 6**) it is expected that you'll be using a WCS. The idea is that the mill needs to "know" where a part is in order to cut it properly. Defining a WCS zero point on the part (or fixture) provides this information.

In order to define a **WCS**, you usually use an **indexing** technique, described in **section 5.4**. After indexing, the WCS zero point is stored in one of our **WCS registers**, of which we have six: **G54, G55, G56, G57, G59** and **G59**.

Once a WCS register is populated with a zero point, you can issue moves with respect to that WCS zero by including the register name in the move command:

G54 G0 X-20 – move to coordinate -20 in X with respect to the G54 WCS

G56 G0 X30 Y-50 – move to coordinate +30, -50 in the G56 coordinate space

These commands are essentially the same as **G53 moves**, the only difference being which zero point is being referred to.

There is one additional difference to **G53 moves** – unlike **G53**, moves in **G54 – G59** are **modal**, meaning that after issuing a move in (say) G54, all future moves will also be in G54 until a different register is selected.

5.2.3: Spindle Control

There are two types of spindle operations we can perform:

- Causing the spindle to spin (or stopping it from spinning)
- Changing the spindle's speed (revolutions per minute)

5.2.3.1: Spindle Activation / Deactivation

The following commands are used to spin up (or spin down) the spindle:

- **M3:** causes the spindle to spin clockwise (with respect to the spindle) – this is normally the direction that tools cut in

- **M4:** causes the spindle to spin anticlockwise (with respect to the spindle) – this is often used when you want to touch a part with a spinning tool without actually cutting (e.g. conductive probing)

- **M5:** this stops the spindle from spinning

These commands are all **modal**, meaning that once set, the spindle will remain in this mode until a new **M3, M4** or **M5** command is sent.

5.2.3.2: Spindle Speed

Spindle speed is controlled with an **S** command, which looks like:

S5000 – sets speed to 5000 RPM

S1500 – sets speed to 1500 RPM

CR-1 has a minimum spindle speed of **1500RPM** and a maximum speed of **8000RPM**. While you can issue commands above or below these (e.g. S500, S9000, etc.) they will not have the expected effect; issuing a command beneath 1500 will not spin the spindle at all, and a command above 8000 will simply spin the spindle at the maximum 8000RPM.

Spindle speed commands are also **modal**. Once set, they remain in place until otherwise changed.

Often you will issue an **activation / deactivation** command and a **speed** command at the same time, e.g.:

M4 S1500 – begin spinning backwards at 1500RPM

M3 S5000 – begin spinning in the cutting direction at 5000 RPM

M5 S0 – stop the spindle entirely

5.3: Writing Code: Modal State

We have referred several times now to the concept of **modal commands**. What does this mean?

grbl has several internal modes which affects how commands are interpreted. **Modal commands** are commands that affect these modes. For example, consider **G20 / G21**, which control whether movement commands are interpreted as inches or mm

G91 G0 G20 Y-2 – moves two *inches* to the left

G91 G0 G21 Y-2 – moves two *millimeters* to the left

When issuing a modal command, the mode is now set internally and will remain set until either:

- A new modal command is issued to change the mode
- grbl resets

For example, after running *G91 G0 G20 Y-2*, all future movement commands will be interpreted in inches as well. As such, a command like this will move left 3 inches, even though we didn't specify G20 in the command.

G91 G0 Y-3

Almost all modal commands can be run independently, without needing to tie them to a move or other command. For example, running this will simply change the internal mode to millimeters.

G21

Upon startup, grbl has a default value for each internal mode, which is:

G0 G54 G17 G21 G91 G94 M5 M9 T0 F0 S0

It is not uncommon to start a G-Code file with a set of modal commands to ensure that all future commands are interpreted as expected.

Here are several common modes and their corresponding commands:

Mode	Description	Examples
Motion Mode	Whether we move rapidly or at a specified feedrate	G0, G1
Move Type	Whether we move in absolute or relative space	G90, G91
WCS Selected	Which WCS is currently active	G54, G55, G56, G57, G58, G59
Feed Rate	What the current feed rate is	F1000
Spindle Speed	What the current spindle speed is	S5000
Spindle State	Whether the spindle is on or not	M3, M4, M5
Input Units	How to interpret move units	G20, G21

For an exhaustive treatment for modal commands, please review **Section 9.1**.

5.4: Writing Code: Indexing

We first covered indexing / probing in **Section 4.6**, so if you haven't already, we suggest you read that section.

At a high level, indexing is performed either by:

1. Touching a part (or fixture) with the spindle
2. Touching a part (or fixture) against another part (or fixture) whose position is known

The first method is possible because the CNC always knows where its own spindle is (assuming it has been properly homed). The second method requires the reference part (sometimes referred to as an "index") to have itself been indexed in some form.

In either case, there are two aspects that you may need to write code for:

- Moving the spindle so it touches the part or fixture
- Recording the location into a WCS register

Moving the Spindle:

When performing **edgefinding** or **touchoff** styles or probing, the only code you need to write is the code that moves the spindle / tool to its pre-probing position. From there, the user must simply jog the spindle until it touches the edge / pinches the paper.

However, when performing **conductive** probing, there is a special command we can use. This command will look something like this:

G38.2 G91 Y-20 F20

This appears to be a normal relative mode, except for the addition of **G38.2**. Adding this component performs this movement in *probe mode*, which will monitor the probe circuit during the movement and stop moving immediately if that circuit closes.

Other valid probe commands:

G38.2 G91 X10 F20

G38.2 G91 Z-5 F20

You can apply **G38.2** to any movement and have that movement act as a probe movement – for example, *G38.2 G53 G1 Y-20 F20* will work. However, it is more common to probe using relative movement.

Similarly, while you *can* probe in G0, this is not recommended. Given that you are intentionally crashing the tool into a part, it makes sense to move very slowly with G1 so that you get an accurate probe result, and you don't damage your part or tool.

If probing in X or Y, we recommend that you probe while spinning the tool backwards at the minimum RPM. This command is:

M4 S1500

You want to spin the tool when probing in X or Y because you are trying to touch the part with the side of the tool, and the tool (when still) presents an uneven surface. Spinning the tool approximates probing with a perfect cylinder, improving accuracy. We spin *backwards* with **M4** so that you minimize damage to the surface of the part when the tool touches.

When probing in Z, there is no need to spin the tool.

Recording the Location

Once your probing technique has touched the tool to the part, you must write the location of the spindle to your WCS register.

This command will write a location into WCS:

G10 L20 P1 Z0

Let's break this command down:

- **G10:** this is the main "write to WCS" command
- **L20:** this affects how the write location will be interpreted. L20 means "write with reference to spindle location", which is what we almost always want.
- **P1:** this controls which WCS we are writing to. P1 writes to G54, P2 writes to G55, etc.
- **Z0:** this controls the data we are writing – in this case, we are writing the current Z position of the spindle

You can write only one axis to WCS at once (as above) or multiple:

G10 L20 P3 X0 Y0 – writes the current position of X and Y to G56

When recording Z position, you normally always record Z0, since the tool tip will always be directly touching the surface. However, when recording X and Y, you often want to record an *offset* to the current position, because it is the *edge* of the tool that is touching the edge, and not the *spindle center*.

To account for this, you should record with an offset equal to the radius of the tool. For example, if probing with a ¼" endmill:

G10 L20 P1 Y-3.175

This will record the current Y position to G54, offset by 1/8", or half of a ¼" endmill.

The sign for the offset matters. The sign you want reflects the *direction from the actual edge to the spindle's actual current location*.

- If we are probing the *right* side of a cube (with respect to the mill) then the direction from the cube's edge to the tool is in the *negative* direction, hence Y-3.175

- If we are probing the *bottom* of a cube, then the direction from the cube's edge to the tool is in the *positive* direction, so we'd use something like X3.175

The above commands will be sufficient for probing edges and surfaces, but there are some cases where you may want to find *midpoints*. The CR-1 offers a few different methods for calculating midpoint data and writing it to WCS.

One method is using **M100**. You begin by probing the two sides that the midpoint lays between, e.g. the two edges. Store these in two different WCS registers (for example, G55 and G56), and then run:

M100 G55Y G56Y G54Y

This will record the midpoint between the G55 probe point and the G56 probe point and store it in G54. Note that it will only do so for the Y component - you can replace the Y's with X's to get the X component, and so on.

You can also use **M102** to perform arbitrary math to get a midpoint:

M102 G54Y ((G55Y + G56Y)/2)

To learn more about **G10**, **M100** and **M102** commands, check out **Section 9.**

5.5: Creating Fixtures

This section to be completed in a future version of the manual.

5.6: Creating crproj file

This section to be completed in a future version of the manual.

Section 6: Creating CAM

This section to be completed in a future version of the manual.

6.1: Creating the Setup

This section to be completed in a future version of the manual.

6.2: Creating Toolpaths

This section to be completed in a future version of the manual.

6.3: Exporting Your G-Code

This section to be completed in a future version of the manual.

Section 7: Maintenance / Disassembly

The CR-1 is a rugged and hardy tool, but that doesn't mean it doesn't require care and maintenance. In this section we will go over maintenance tasks, as well as common disassembly tasks.

7.1: Basic Maintenance

The following items should be performed regularly – at the very least, we recommend performing these tasks after each major milling job is completed. Caring for your CR-1 will make it last longer!

7.1.1: Regular Cleaning

"Cleaning the CR-1" refers almost entirely to removing swarf / chip debris. This debris can take several forms, from large chips that must be vacuumed to small chip dust that should be wiped away.

Follow the best practices below to keep the CR-1 free of accumulated swarf.

DO NOT USE COMPRESSED AIR TO CLEAN THE CR-1

While it may be tempting to use compressed air to blow chips off and away, *do not use it*. Blowing chips could work their way past internal guards and seals and into the electronics and power supply, which could cause a short and break your CR-1.

Only remove chips by vacuuming or wiping – never by compressed air!

1. **Vacuum Chips**

   Use a shop vac / Dustbuster to regularly vacuum up chip accumulation inside, on and around the CR-1. You should be able to insert a vacuum hose into the front of the CR-1 and reach most of the areas where chip accumulation is problematic – the table, the fixtures, on and around the X-motors.

   Obviously, do not vacuum while the machine is running!

   We recommend cleaning chips every time you switch between operations – e.g. every time you perform a tool change, orientation change, etc.

2. **Wipe and Lubricate Shafts, Ballscrews and Surfaces**

   Vacuuming chips will remove many of the large chips, but some chips and smaller swarf may remain attached to various surfaces within the mill. This is especially true if you are cutting with lubricant or other liquid that can build up and cause debris to stick to internal surfaces.

   Between milling sessions we recommend you wipe down all available surfaces with a rag to remove accumulated swarf and lubricant. This includes the walls, table, mirrored spindle guard, etc.

   You should also wipe down all easily accessible shafts, ballscrews and bearings. The sealed bearings may accumulate chips around their edges – use a cotton swab to gently wipe these away.

   After wiping the chips away from the shafts and ballscrews, apply a light coating of oil.

3. Check Belt Condition

The CR-1 has **three belts:**

- One connecting the spindle to the spindle motor
- One connecting the Y ballscrew to the Y motor
- One connecting the Z ballscrew to the Z motor

When performing other maintenance, look around for signs of deterioration of these belts. The clearest sign of deterioration is black rubber debris indicating belt wear or shredding. This will accumulate in the back or right side of the mill, near where the belts sit.

If you turn the mill upside-down, you can also examine the belts themselves. **If you notice belt deterioration, we recommend replacing the belt** (see the next section on how to do this.)

7.1.2: Tool Wear

When used for cutting operations, endmills accumulate wear. Wear can take several forms:

- Dulling – the cutting edges of the tool lose sharpness
- Chipping – small pieces of the tool break or chip off
- Breakage – the endmill breaks entirely

There are many problems associated with worn tools:

- Poorer cut quality – including worse surface finish and less accurate cuts
- Louder noise when cutting
- Chipwelding – the phenomenon when a worn tool produces higher friction when cutting, melting chips and causing them to stick to the tool

Wear accumulates differently depending on several factors:

- How long the tool has been used for
- The hardness of the material the tool is cutting
- The sorts of operations the tool is performing
- The feeds and speeds used when cutting

For example, a tool chat cuts primarily aluminum (a softer metal) will not wear as quickly as one that primarily cuts steel.

Here are some tips for tracking and dealing with wear:

1. Pay attention to cut performance, noise, chip color, etc. during cutting operations. If the cut is louder than normal, or if you see darker colored chips, this is likely an indicator that the tool is excessively warn.

2. Try to keep track of how much wear a tool has accumulated – how long it's been used and what it's been used for. Proactively replace a tool if you feel that it's been used for too long.

3. Configure your cuts to “spread wear evenly” across the tool as best you can. For example, it’s often better to do larger stepdowns and smaller stepovers to “spread the wear” across more of the flute length.

7.2: Advanced Maintenance

If you encounter problems beyond the info described above, or if you simply wish to investigate or modify the hardware on your machine, read this section to learn about machine disassembly and part replacement.

7.2.1: Removing / Replacing Enclosure

For almost all advanced maintenance operations, removing the enclosure is the first step (and replacing it is the last step). To do this, follow the instructions below.

The following tools are required to remove / replace the enclosure:

1. 3.0mm screwdriver or Allen key
2. 2.5mm screwdriver or Allen key
3. 2.5mm screwdriver or Allen key with *long shaft* – at least 8" (12" is recommended.)
    a. The Bondhus 25445 Allen key set is a good option for this and is available at Amazon at time of writing

Removing Enclosure

1. **Unplug both power and USB cable**

2. **Vacuum all metal chips and clean machine as best you can**

3. **Turn machine upside-down on table**

4. **Unplug cable from LED PCB.**
   This cable connects to the LED PCB near the probe wire connector. Pull it backwards towards the back of the machine to unplug it.
   We recommend you take a photograph of the plug's orientation before unplugging to help you plug it back in with the right orientation.

5. **Remove X chip guards**
   Remove the two bolts at the bottom of the two chip guards covering the X shafts / ballscrews (four bolts in total.) This will require the long-handled 2.5mm Allen key.

   Once removed, set the chip guards aside.

6. **Unplug emergency stop button connectors**
   Removing the chip guards exposes the back of the estop button.
   There are three spade connectors attached to the estop button. Pull all three out.

   We recommend you take a photograph of the connectors before unplugging to help you plug them back in with the right way. *You may also want to mark the wires with a marker or tape to help*

identify them.

7. **Remove bolts from electronics bay cover**
   Using a 2.5mm Allen key, remove the seven bolts holding the electronics bay cover in place. This can be found on the side of the machine with the estop button.

8. **Remove bolts from power supply cover**
   Using a 2.5mm Allen key, remove the six bolts holding the electronics bay cover in place. This can be found on the side of the machine opposite the estop button.

9. **Turn machine over / right side up**

10. **Remove bolts on the exterior of the mill**
    Using a 2.5mm Allen key, remove three bolts from the left side of the mill, two from the top and three from the right side. All of these are at the back of their respective sides.
    Using a 3mm Allen key, remove three bolts from the top of the mil.

11. **Slide Enclosure Off**
    Now that all bolts have been removed, you can slide the enclosure forward and off the mill.
    The backplate will still be attached, which is fine.

Now that the enclosure is off, the mill is fully supported via its internal frame. While strong, this frame is partially aligned by the enclosure, meaning that with the enclosure off the frame can skew and sit at weird angles.

Do not apply undue force or pressure to the "naked" machine. If you must move it, lift it by the top plate or brace. Do not apply force to thin sheet metal parts (the chip guards).

Replacing Enclosure

To replace the enclosure, simply follow the instructions above in reverse order.

A few notes to help with reassembly:

- The emergency stop cables should be routed such that they sit along the enclosure wall. Do not run these cables in front of any of the shafts or ballscrews.
- Some of the screws (i.e. those on the X chip guards) are rather difficult to place in their holes without falling off the long-handled Allen key. Using a magnetized Allen key or heavy grease to help retain the screws should help.

After replacing the enclosure, we recommend you run $L to autolevel the bed.

Reconnecting Emergency Stop Cables

It can be tricky to reconnect the emergency stop cables, and if you do it wrong your machine will act like the estop is permanently down.

Use this process to ensure that you connect the cables right.

There are three terminals on the estop button:

- One labeled **NO**
- One labeled **NC**
- One labeled **C**

Plug in the USB cable to power the electronics and then measure the voltage between the X table and each cable spade. The cables should connect to the terminals in the following way:

- Cable measuring ~**0V** between spade and bed: connect to **NO** terminal
- Cable measuring **~4.7V** between spade and bed: connect to **NC** terminal
- Cable measuring **~0.7V** between spade and bed: connects to **C** terminal

Reconnecting LED Board Cable

The LED board cable can be reconnected in one of two orientations. To confirm you've connected it right:

1. Plug in the USB cable to power the electronics.
2. If the probe wire is unplugged from the LED board, the probe LED should be lit a solid green.
3. Plug in the probe wire and the LED should turn off
4. Touch the probe wire's metal end to the X table. The LED should light up green again.
5. Removing the probe wire from the X table should turn the LED light off again.

If the above behavior is not observed, unplug the LED board cable, flip it upside down and try it in the new position.

7.2.2: Replacing Spindle Belt

After a long period of mill use, the spindle belt will start to wear out. Fortunately it is quite easy to replace it.

For our spindle belts we recommend **Gates brand, model 172-2MGT-06**. Our testing shows that these can last 20 times longer than other belts.

The following tools are required to remove / replace the spindle belt:

1. 4.0mm screwdriver or Allen key
2. 4.0mm screwdriver or Allen key with *long shaft* – at least 8" (12" is recommended.)
    a. The Bondhus 25445 Allen key set is a good option for this and is available at Amazon
3. 8mm wrench

The following instructions describe how to replace the belt:

1. **Ensure the mill is empty and that no tools or fixtures are installed**

2. **Move the CR-1 spindle to a convenient position**
   You can run this code:
   $H
   G53 G90 G21 X0 Y-120 Z-78.5

   Alternatively, you can just the gantry / table / spindle by hand. It's okay to push these parts along their axes with the power off.

3. **Unplug both power and USB cable**

4. **Vacuum all metal chips and clean machine as best you can**

5. **Turn machine upside-down on table**

6. **Identify the BLDC motor next to the spindle**
   This is a large, rectangular black motor sitting right next to the spindle and linked to the spindle via a belt. You'll see four bolts / nuts holding the BLDC motor in place.

7. **Loosen the four nuts holding the BLDC motor**
   Do not remove these nuts – just loosen them.
   Use the 3.0mm Allen key to hold the bolts in place while using the 8mm wrench to loosen each nut. One of the bolts is hard to see / reach, but you should be able to come in from the front of the mill with the long-handled Allen key.

8. **With the nuts loosened, the motor can now wiggle slightly in its housing. Tilt it such that you can remove the belt**

9. **Install the new belt over the motor / spindle sheaves**

10. **Tension the belt by pulling the BLDC away from the spindle. An easy way to do this is to insert a metal bar or screwdriver between the spindle and BLDC motor and levering it to push the motor away.**

11. **Re-tighten all four of the nuts securing the motor**

12. **Test the belt tension by plucking it – it should make a note like a guitar string**

7.2.3: Removing / Replacing Spindle

Removing the CR-1's spindle is a semi-common task for more advanced maintenance, either because the maintenance involves the spindle itself, or because you must remove the spindle to gain access to other parts.

The following tools are required to remove / replace the spindle:

1. 2.5mm screwdriver or Allen key
2. 4.0mm screwdriver or Allen key
3. 3.0mm screwdriver or Allen key with *long shaft* – at least 8" (12" is recommended.)
    a. The Bondhus 25445 Allen key set is a good option for this and is available at Amazon at time of writing
4. 4.0mm screwdriver or Allen key with *long shaft* – at least 8" (12" is recommended.)
    a. Same Bondhus 25445 kit recommended
5. 8mm wrench

Removing Spindle

To remove the spindle, follow the directions below:

1. **Ensure the mill is empty and that no tools or fixtures are installed**

2. **Remove the two bolts securing the mirrored spindle guard with a 2.5mm Allen key. Remove spindle guard and place aside.**

3. **Run through steps 1-8 of Section 7.2.2 "Replacing Spindle Belt" in order to take the belt off.**

4. **Remove the wiper mounted next to the spindle by removing the two M4 screws mounting it to the spindle mount. Pull the wiper out of the spindle assembly.**

5. **Loosen the three M4 bolts on the upper spindle mount to unclamp the spindle from this mount.**

6. **Pull the spindle forward and out of its mounts.**

Removing the spindle will grant you access to certain parts beneath the spindle.

Replacing Spindle

When you're ready to replace the spindle, follow these instructions:

1. **Slide the spindle back into its mounts**

2. **Reinstall the wiper into the spindle and bolt it in place**

3. **Tighten the three bolts on the upper spindle mount to re-clamp the spindle**

4. **Reinstall the belt using the instructions in Section 7.2.2.**

After replacing the spindle and enclosure, we recommend you run $L to autolevel the bed. You should also test spinning the spindle and listen closely to ensure it spins smoothly.

7.2.4: Replacing Limit Switch

The CR-1 uses optical limit switches to assist with homing and autoleveling procedures. Broken limit switches mean your mill cannot home. If you notice problems where the mill refuses to home properly, a broken limit switch may be the culprit.

The CR-1 uses this limit switch:
https://www.robotdigg.com/product/155/Optical-Endstop-w/-Lead-Wires-for-3D-Printers

We also use a custom wire different from the stock wire. You can purchase both the switch and the wire from Coast Runner.

Here are the locations of the CR-1's limit switches:

- X limit switches: mounted to the top of the internal frame behind the chip guards for the X ballscrews. Accessible by removing the chip guards
- Y limit switch: on right wall of the mill behind the Y stepper motor. Accessible by removing enclosure
- Z limit switch: underneath the spindle BLDC, accessed by removing spindle

Identifying a Bad Switch

If we're seeing homing problems, how do we know *which switch* is bad?

You can determine which limit switch is failing by running individual home commands for each axis:

$HX *– homes X axis*
$HY – *homes Y axis*
$HZ *– homes Z axis*

If one of these fails or throws an error, that axis' limit switch is the likely culprit.

You can also visually inspect the switches. Functional switches should have a red LED light illuminated when the switch is powered (via the USB cable) and not tripped. Check inside the mill when the USB cable is plugged in – if a switch is not tripped, but there is no red LED light, that's a good sign the switch has failed.

Replacing a Bad Switch

Follow these steps to replace the limit switch:

1. **Ensure the mill is empty and that no tools or fixtures are installed**

2. **Remove the enclosure by following the steps in Section 7.2.1.**

3. **Identify the location of your limit switch**
   Now that you have the enclosure off you should be able to get a good look at where the limit switch you want to remove is located. Using the location list above, find the switch.
   Identify how its cable is routed. You'll want to route the new switch's cable in the same way.

4. **Unplug the switch from the electronics board**
   Now that the cover is off you have access to the electronics bay on the left side of the machine (same side as the estop button). Find where the limit switch is plugged in and unplug it.

5. **If removing the Y limit switch, you can remove it immediately.**
   Unbolt it from the right side of the mill. Let it hang off the side for now.

6. **If removing an X limit switch, unbolt the chip guards and then remove the switch.**
   You've already taken off two of the X chip guards; unbolt the remaining two (they're bolted to the X table.) Now you can access the X limit switch and remove it. Let it hang off the side for now.

7. **If removing the Z limit switch, remove the spindle and then remove the switch.**
   The Z limit switch is buried beneath the spindle assembly. Remove the spindle by following the steps in Section 7.2.3.
   Once the spindle is removed, you can get at the switch. Unbolt it and let it hang for now.

8. **Pull the switch out gently. Watch how the cable is routed as you pull**.
   The cable will run through a cable bundle, which will be routed in a specific way through the mill.

9. **Run the new switch's cable back through the same cable bundle / routing**

10. **Bolt the new switch into its location**

11. **Plug the new switch into the electronics board**

12. **Plug in the USB cable to apply power and test the switch**
    If the red LED lights up on the new switch, then the new switch is good to go.

13. **Reassemble all parts of the mill**
    Replace the spindle (if removed), chip guards (if removed) and enclosure.

7.2.5: Replacing Power Supply

In some unfortunate circumstances (such as improper use of compressed air) a power supply can fail. If this happens, you'll have to replace the supply.

Follow these steps to replace it:

1. **Ensure the mill is empty and that no tools or fixtures are installed**

2. **ENSURE THAT THE POWER CORD AND USB CORDS ARE UNPLUGGED**.
   Never attempt to work with or touch the power supply when the power cord is plugged in.

3. **Remove the enclosure by following the steps in Section 7.2.1.**

4. **Identify the power supply on the right side of the machine**
   The PSU should be relatively obvious – it's a large rectangular block with a visible fan

5. **There are five wires connecting the PSU to the mill.**
   Take a photo of the current wiring configuration

6. **Unscrew the five wires from the PSU**

7. **Unscrew the PSU from the mill**
   There are four screws accessible on the inside of the mill that hold the PSU on. Unscrew all four and remove the old PSU.

8. **Screw in the new PSU**

9. **Rewire the PSU**
   Use the photo you took to ensure the wires are reconnected properly

10. **Reassemble the mill**

7.2.6: Replacing Electronic Board

CR-1 uses three electronics boards:

- **Arduino Uno Rev3:** this is our main microcontroller
- **Custom Control Board:** this board controls additional electronics to handle the spindle and motors
- **Custom LED Board:** this board mounts a series of LEDs and also mounts the jack for the probe wire. *We will not cover how to replace this board*.

It's rather unusual for an electronics board to fail, but it can happen. Sometimes a board might get bricked following a botched manual firmware update.

In any case, if you need to replace the board, here's how to do it:

1. **Ensure the mill is empty and that no tools or fixtures are installed**

2. **ENSURE THAT THE POWER CORD AND USB CORDS ARE UNPLUGGED**.
   Never attempt to work with or touch the electronics when the cables are plugged in.

3. **Remove the enclosure by following the steps in Section 7.2.1.**

4. **Identify the electronics boards on the left side of the machine**
   The electronics boards will be oriented such that the custom control board is mounted on top of the Arduino.

5. **Take a photo of how the boards are oriented**
   Since you'll need to unplug and replug everything, take this opportunity to photograph the board with everything plugged in right. You can use it as reference later.

6. **Unplug all cables from the board you need to replace**

7. **Remove the old board and set it aside**

8. **Plug the new cables into the new board**

9. **Once everything is plugged back in, plug in the USB cable (*not the power cable*) and see if you can connect to the mill using CRWrite.**
   If you're able to achieve connection / communication, you've likely replaced the boards successfully.

10. **Replace the enclosure**

11. **Update the firmware**
    It's possible the new board doesn't have the latest firmware. Make sure to flash the firmware with the latest values.

Section 8: Troubleshooting

Mill won't connect

This manifests as the Mill Status indicator in CRWrite not changing to "Connected". Sometimes CRWrite will only display a red "Not Connected" or "Connection Failed" message, other times it will change to the yellow "Connecting" status but will never connect.

If this happens, try the following:

1. Ensure the red emergency stop button on the side of the mill is not pushed down. If it is, twist it counter-clockwise until it pops out to reset.
2. Ensure that no other programs are open that may be monopolizing the serial connection. This can include other serial terminals (such as that in Arduino Studio), and we have also observed 3D slicer softwares (like Cura) causing this behavior. Please close them all.
3. Ensure that the correct Arduino drivers are installed. CRWrite is supposed to do this upon first launch, but if it fails to do so, installing Arduino Studio should add those drivers.
4. Unplug the mill's USB cable and close CRWrite. Then open CRWrite again, and once CRWrite is open, plug the USB cable back in.

CRWrite displays blank white screen at launch

On some operating systems and under some configurations, CRWrite may take a long time to initially load and render, especially when first launching the software. Usually the software will eventually render, even though it may take 30 to 60 seconds. Future versions of CRWrite will improve this performance.

Opening a .crproj file takes a long time

If a crproj file is large (>100MB) it may take several seconds for CRWrite to load the file and open the subsequent job selection screen. Please be patient while it loads. Future versions of CRWrite will improve this performance.

Mill won't home / motors crash during homing

This is likely due to an issue with a limit switch, especially if accompanied by Alarms 8 or 9. Check **section 7.2.4** for information about diagnosing and replacing limit switches.

Alarm / Error Thrown in Guided Mode

.crproj files are tested before release, and so alarms / errors are unexpected and likely indicate either a hardware problem with your mill, or possibly a user mistake that has led to the error.

We recommend you look up the particular error or alarm you receive in **Section 9.4 / 9.5** and based on this perform further diagnosis / troubleshooting. If you cannot figure it out, please contact support@coastrunner.net for more information.

If you are writing your own code and running it in manual mode, alarms / errors are far more expected :) If you're clever enough to write your own code, we expect you to be clever enough to troubleshoot it using the information in this manual – but always feel free to reach out to support!

"Emergency stop detected" error thrown when no estop performed

If you get an "estop detected" message when you didn't actually press the estop button, this indicates that for some reason CRWrite lost communication with grbl. This can happen in a few scenarios:

- Your computer went to sleep
- Your computer went into a "lower power mode" that caused a disconnection
- Your USB hub went into a "low power mode" that caused a disconnection (yes, we have seen this happen)
- The CRWrite software glitched out

We recommend double-checking your power settings and sleep settings and trying again. If you have done this and are still experiencing these problems, contact support@coastrunner.net for assistance.

Section 9: Reference

9.1: G-code Commands

These tables show some of the commands that the CR-1 can execute. The tables are not exhaustive. Commands are grouped by their similarity and / or modal state.

Excepting the commands in **7.1.6,** and M17/M18, Coast Runner's G-code interpreter conforms to generally accepted industry syntax. When discrepancies exist between various G-code standards, Coast Runner follows LinuxCNC's methodologies. See https://linuxcnc.org/documents/

9.1.1: Common Movement Commands:

These commands are commonly used to initiate, stop, or modify X/Y/Z motion.

Name	G-Code	Description	Modal State	Example	Example Notes
Basic Movement	X/Y/Z(-)*n*	Basic command to move one or more axes. Nature of move and destination will depend on modal state. Movement will be interpolated, e.g. a straight line from current position to destination, in all specified axes. Numbers are interpreted as either inches or mm depending on **Input Units** modal state.	N/A	X-70	Move in the negative direction in one axis
				X-70 Y10	Move two axes simultaneously, negative in X and positive in Y
				X10 Z-20	Move two axes simultaneously, positive in X and negative in Y
Rapid Move	G0	Run movements at a fixed "rapid" speed of 4,500 units/min (defined in mill EEPROM). Can be combined with **Basic Movement** to initiate move.	Motion Mode	G0	Set **Motion Mode** in parser state to G0.
				G53 G90 G0 Y-5	Move rapidly to Y-5 in absolute machine coords
				G0 X-10	Move rapidly to X-10. Destination affected by **WCS Selected, Move Type** modes
Linear Move	G1	Run movements at a user-specified speed (specified by an **F*n*** command). Can be combined with **Basic Movement** to initiate move. Attempting a G1 move without specifying a feedrate throws an error.	Motion Mode	G1	Set **Motion Mode** in parser state to G1.
				G54 G90 G1 Y-5 F1000	Move at 1000 units/min to Y-5 in absolute machine coords.
				G1 Y-5 F5000	Move at 500 units/min to Y-5. Destination affected by **WCS Selected, Move Type** modes.

Absolute Move	G90	**Basic Movement** commands are interpreted as absolute coordinates in reference to a zero point (either machine zero for G53 moves or user-defined WCS zero for G54-59 moves)	Move Type	G90	Switch **Move Type** modal state to G90
				G54 G90 G0 X10 Y-5	Move rapidly to X, Y coord (10, -5) in reference to zero point defined in G54.
				G90 Y-5	Move to Y coord -5. **WCS Selected** and **Motion Mode** determined by modal state.
Relative Move	G91	**Basic Movement** commands are interpreted as absolute coordinates in reference to a zero point (either machine zero for G53 moves or user-defined WCS zero for G54-59 moves)	Move Type	G91	Switch **Move Type** modal state to G91
				G91 G0 Y-5	Move rapidly left 5 units from current position
				G54 G91 Y10	Move right 10 units from current position. **Motion Mode** determined by modal state. G54's presence does not affect this command, but including it does set **WCS Selected** to G54.
Machine Zero	G53	Use with **Basic Movement** command to perform **absolute move** in relation to machine zero (the home position)	N/A	G53 Y-5	Move to absolute Y coord -5, in reference to machine zero.
				G53 G91 Y-5	Equivalent to absolute move command above. G91 is always ignored for G53 moves.
				G53	Does nothing. G53 does not set a modal state – only way to move in G53 is to specify G53 in G-code command.
WCS Zero	G54 G55 G56 G57 G58 G59	Defines which WCS we are operating in. When combined with an absolute **Basic Movement** command, initiates a move in reference to that particular WCS.	WCS Selected	G54	Switch **WCS Selected** modal state to G54
				G55 G90 G0 Y-5	Perform rapid absolute move to Y coord -5 with regards to G55 zero point

				G56 X10	Assuming G90 set, move to X coord +10. **Motion Mode** determined by modal state. (If G91 set, move 10 units up – G56 is ignored)
Set Feed Rate	F*n*	Sets the current feedrate. Numbers are interpreted as either inches or mm depending on **Input Units** modal state.	Feed Rate	F1000	Set **Feed Rate** modal state to 1000 units/min
				G53 G1 Y-5 F200	Move to Y coord -5 in absolute machine coords at 200 units/min (slow)
				G53 G0 X-30 F500	Move to X coord -30 in absolute machine coords at rapid pace. F500 does not affect move speed due to G0, although **Feed Rate** modal state is set to 500.

9.1.2: Jog Commands:

Jogging is a special kind of movement that works a little differently from normal movement. This table covers jog commands. Note that jog commands are not technically G-Code, but are instead grbl commands; regardless, we put them in this section.

For more information about jogging, visit https://github.com/gnea/grbl/wiki/Grbl-v1.1-Jogging

G-Code	Description	Example	Example Notes
$J=*(x/y/z move)* F*n*	Sets the current feedrate. Numbers are interpreted as either inches or mm depending on **Input Units** modal state. Command always starts with $J= followed by a normal move command. Like other move commands, grbl reads from the parser state to determine the nature of the move, but with some exceptions: 1. The parser state is not altered by the jog command. 2. All jogs are in G1 and the feedrate must be specified – parser state feedrate is ignored. 3. Jog commands that would violate a soft limit rule are ignored instead of throwing an alarm When executing a jog command, grbl's internal status changes to "Jog". Multiple jog commands can be queued up at one time. When jogging is being performed, no other move commands can be queued up.	$J=G91 Y-20 F100	Jog left 20 units at 100 units per minute. Does *not* write G91 or F100 to parser state, these apply only to the jog command.
		$J=G90 X-10 F50	Jog to X coord -10 in whatever WCS is currently active, at 50 units per minute
		$J=G53 Z-20 F300	Jog to Z coord -20 in absolute machine coords at 300 units / min
! *0x85*	An in-progress jog command can be cancelled by sending either of the two jog cancel commands. Sending either will stop machine motion and flush any queued jog commands. ! normally acts as a feed hold command, but when jogging is active, it acts instead as a jog cancel command. Executing it to cancel a jog will not activate feed hold. *0x85* is an extended ASCII character which cannot be typed on the keyboard, but which can be sent programmatically. Sending this command does not activate feed hold.	!	Cancels any in-progress jog. Does not activate feed hold.

9.1.3: Common Spindle Commands:

These commands are commonly used to initiate, stop, or modify spindle motion.

Name	G-Code	Description	Modal State	Example	Example Notes
Spindle Clockwise	M3	Enables the spindle to spin clockwise, the standard direction for cutting operations. Spindle will only actually spin if a valid **spindle speed** is stored in the modal state.	Spindle State	M3	Sets **spindle state** modal state to clockwise. May start spinning if **spindle speed** was previously set.
				M3 S5000	Starts spindle spinning clockwise at 5000 RPM
Spindle Counterclockwise	M4	Enables the spindle to spin counter-clockwise, which is "backwards" for most cutting tools. Tool will not cut – useful for probing operations. Spindle will only actually spin if a valid **spindle speed** is stored in modal state.	Spindle State	M4	Sets **spindle state** modal state to counter-clockwise. May start spinning if **spindle speed** was previously set.
				M4 S5000	Starts spindle spinning counter-clockwise at 5000 RPM
Spindle Stop	M5	Stops spindle from spinning.	Spindle State	M5	Sets **spindle state** modal state to "stop". If spindle is currently spinning, will stop the spindle.
				M5 S5000	Same as the above – spindle will stop despite the speed command. Note though that the speed will still be set in the modal state.
Spindle Speed	S*n*	Sets the spindle speed. Numbers are interpreted as revolutions per minute (RPM)	Spindle Speed	S5000	Sets the **spindle speed** modal state to 5000 RPM.
				M3 S5000	Starts spindle spinning clockwise at 5000 RPM
				M3 S100	Spindle stops spinning. CR's spindle minimum is 1500 RPM
				M3 S10000	Spindle starts spinning at 8000 RPM. Any speed set above 8000 is bumped down to 8000.

9.1.4: Common Indexing Commands:

These commands perform, or are related to performing, indexing operations.

Name	G-Code	Description	Example	Example Notes
Probe Until Contact, Alarm If No Contact	G38.2 X/Y/Z(-)*n* F*x*	Performs specified move while monitoring electrical probe circuit. If circuit trips during movement, movement immediately stops. Also sets **Motion Mode** to G38.2. Movement can be **G90 (absolute)** or **G91 (relative)**, and can involve multiple axes. Probe moves are always **G1 (linear)**, so feedrate required to be included or defined in modal state. Failure to trip probe after movement completes throws **ALARM:5**. Must start with probe in untripped state, throws **ALARM:4** if probe tripped when called.	G38.2 G91 Y-5 F200	Probes left 5 units at 200 units/min. Sets **Motion Mode** to G38.2.
			G38.2 G54 G90 Y-5 F200	Probes while moving in Y from current position to coordinate -5 (with reference to G54 WCS). Movement is 200 units/min.
			G38.2 Y-5	Determines the type of move from modal state (G90 vs G91) and probes while performing that move. Feedrate read from modal state (error thrown if modal state empty).
			G38.2	Invalid – command requires a **basic movement** command.
Probe Until Contact, Proceed If No Contact	G38.3 X/Y/Z(-)*n* F*x*	Performs specified move while monitoring electrical probe circuit. If circuit trips during movement, movement immediately stops. Also sets **Motion Mode** to G38.2. Movement can be **G90 (absolute)** or **G91 (relative)**, and can involve multiple axes. Probe moves are always **G1 (linear)**, so feedrate required to be included or defined in modal state. If probe does not trip during movement, grbl progresses to next statement (no alarm.) Must start with probe in untripped state, throws **ALARM:4** if probe tripped when called.	G38.3 G91 Y-5 F200	Probes left 5 units at 200 units/min.
			G38.3 G54 G90 Y-5 F200	Probes while moving in Y from current position to coordinate -5 (with reference to G54 WCS). Movement is 200 units/min.
			G38.3	Invalid – command requires a **basic movement** command.

Set WCS (Absolute)	G10 L2 P*n*	Writes specified X/Y/Z data to the WCS register specified by *n*. *n* can be 1-6, corresponding to G54-59. Written values are based on machine origin, modified by the provided X/Y/Z offset. Another way to think of this: the subsequent zero point will match the provided offsets *exactly* (see examples for details)	G10 L2 P1 Y-10	Sets G54's Y value to 10 units to the left of the machine origin – e.g. Y zero is now -10 units.
			G10 L2 P2 X-70 Y-10	Sets G55's X and Y zero point coords to machine coords -70 and -10 respectively.
Set WCS (Relative)	G10 L20 P*n*	Writes specified X/Y/Z data to the WCS register specified by *n*. *n* can be 1-6, corresponding to G54-59. Written values are selected such that current spindle position is equal to the new zero point we're writing if the provided X/Y/Z offsets are applied. (see examples for details)	G10 L20 P1 X5 Y-10 G4 P2 G91 Y-4	Writes new zero point to G54 that is 10 units to the right of current spindle position, and 5 units above.

9.1.5: Other Common Commands:

Name	G-Code	Description	Modal State	Example	Example Notes
Input Inches	G20	Configures **input units** modal state to expect **inches**. Future commands with coords or distances will interpret those numbers as **inches**. Note that this does not affect the **display units** that grbl returns.	Input Units	G20	Switch **Input Units** modal state to G21
				G91 G20 Y-1	Moves one inch to the left from current position and sets **Input Units** modal state to G20. The units change takes effect immediately and is applied to the current command.
Input Millimeters	G21	Configures **input units** modal state to expect **millimeters**. Future commands with coords or distances will interpret those numbers as **millimeters**. Note that this does not affect the **display units** that grbl returns.	Input Units	G21	Switch **Input Units** modal state to G21
				G91 G20 Y-1	Moves one inch to the left from current position and sets **Input Units** modal state to G20. The units change takes effect immediately and is applied to the current command.
Dwell	G4 P*n*	Pauses execution for a set number of seconds. No commands after this will be executed until G4 finishes, so can serve as a useful "synchronizing" command to allow grbl to "catch up". Example: execute between G10 and M102 command to ensure EEPROM write is completed before M102 attempts to read.	N/A	G4 P1	Dwell for 1 second
				G4	Throws error. Cannot issue G4 command without related P*n* command
				G4 P2 G91 Y-4	Dwells for 2 seconds and then moves 4 units to the left. G4 is executed first.

9.1.6: CRWrite-Only Commands

These commands are not "true" G-Code commands. Sending them directly to grbl will cause an error. They are special "macro" commands that can be sent to CRWrite (via either Guided Mode or Manual Mode) where they are intercepted, interpreted and executed. This may involve sending one or more subcommands to grbl.

Due to the complexity of these commands, extra attention will be paid to some of them.

Name	G-Code	Example	Example Notes
Find and Store Midpoint	M100	M100 G54X G55X G56X	Averages G54X and G55X and writes to G56X
		M100 G54X G55X G54X	Averages G54X and G55X and writes to G54X. One of the input arguments can also be used as the output argument.
		M100 G54X G55Y G56Y	Averages G54X and G55X (not G55Y – because the two input arguments differed on axis, the first argument's axis was used) and writes to G56Y.
Takes three arguments, each consisting of the format G5(4-9)(X,Y,Z). The WCS values referred to by the second two arguments are averaged and then written to the WCS component specified by the first argument. The first two (averaged) arguments are expected to have the same axis. If they differ, the first argument's axis will be used for both. However, the final argument can be a different axis if desired.			
Check Tolerance	M101	M101 G54 G55 G56 X0.1	Throws an alarm if any of the differences between the X components of G54, G55 and G56 are greater than 0.1 units.
		M101 G56 G59 G54 Y1	Throws an alarm if any of the differences between the Y components of G54, G56 and G59 are greater than 1 unit.
Compares three WCS registers and reports whether the absolute differences between them exceeds a certain value. Takes four arguments: the first three are WCS register names in the form of G5(4-9) and the last argument takes the form of (X,Y,Z)*n*. This last argument defines both the axis we are checking, and the tolerance requirement. If the absolute difference between any of the two passed WCS registers is greater than the tolerance requirement, throws an alarm.			
WCS Math	M102	M102 G54Y ((G55Y + G56Y)/2)	Calculates midpoint between the Y values of G55 and G56 and stores the result in G54.
		M102 G54X (G54X – ((.25 / 2) * 25.4))	Adjusts G54X by applying an offset equivalent to the radius of a ¼" endmill, in mm.
		M102 G55Y 10	Writes 10 directly to the Y component of G55. This command allows you to treat these registers like variables, which enables some interesting programming capability.

M102 allows you to perform arbitrary mathematical expressions within G-code and to store the results in a WCS register. Other WCS registers can be used as input variables as well. This can be highly useful for doing things like finding the midpoint between two measured offsets, or adjusting an offset based on a previous measurement.

The command syntax is: *M102 G5(4-9)(X,Y,Z) <exprtk expression using variables of form G5(4-9)(X,Y,Z)>*

The first WCS register specifies the output register. Everything following the first register is treated as an exprtk expression. Exprtk is a C++ math library whose details can be found here: http://www.partow.net/programming/exprtk/. The expression passed in an M102 command must be well formatted and can take arbitrarily many input register variables that match the format described above.

Note that both the input registers and the output registers are *one* component of a WCS register - the X, the Y or the Z component. Only one component is operated on at a time. Valid registers are G54 - 59 and you can select the X, Y or Z component.

The expert expression is evaluated and the result of *expression.value()* is written to the output register. Nothing is written to the input registers during operation - only the specified output register is modified in an M102 command.

The M102 command is surprisingly powerful and should be used only by users who know what they are doing. Review the exprtk link to see the variety of supported features in the exprtk language.

WCS Comparison	M106	M106 G54X < 0 Error: G54X is too low	Will display the alarm “Error: G54X is too low” if G54X is negative.
		M106 G54X > G55X Error: G54X should be greater than G55X	Will display the alarm message “Error: G54X should be greater than G55X” if G54X is less than G55X.
		M106 0 > G54X Invalid	Will throw a syntax error. The item on the left side of the comparison must always be a WCS variable.

M106 performs a comparison between a WCS register and either another WCS register, or with a constant. If that comparison is false, an alarm is thrown.

The command syntax is: *M106 G5(4-9)(X,Y,Z) [comparison] [G5(4-9)(X,Y,Z), const] [Error message]*.

- ***G5(4-9)(X,Y,Z)****:* the first argument must always be a component of a WCS register.
- ***[comparison]****:* the comparison operator. This can be == (equals), <> (not equals), > (greater than), < (less than), >= (greater than or equal to), or <= (less than or equal to)
- ***[G5(4-9)(X,Y,Z), const]:*** the second argument can be either another WCS register (with one or more subcomponents) or a constant.
- ***[Error message]:*** The custom error message that will be displayed if the comparison fails.

One to One Comparisons: In its most basic case, we can do a one-to-one comparison between a single WCS component and a constant. Here are valid one to one comparison: M106 G54X < 0 Error: G54X is too low

Like to Like Comparisons: You can also compare two WCS registers. For example, M106 G54X > G55X Error: G54X should be greater than G55X

This can be very useful for checking assumptions or tolerances, or for any other use case requiring a comparison.

9.1.7: Uncommon Commands:

The following commands *are* supported by the CR-1, but are not commonly used. Descriptions will not be as thorough as the common commands, and examples will not be given; if you wish to learn more about these commands, study more in-depth grbl / G-code resources.

Name	G-Code	Description	Modal State
Clockwise Arc	G2	Move spindle in a clockwise arc (actually a series of straight lines) based on parameters provided. Arc moves in X/Y; adding Z will not interpolate a curve, but will create a helical movement. Params are checked to ensure they are valid. Sets **Motion Mode** to G2.	Motion Mode
Counter-clockwise Arc	G3	Move spindle in a counter-clockwise arc (actually a series of straight lines) based on parameters provided. Arc moves in X/Y; adding Z will not interpolate a curve, but will create a helical movement. Params are checked to ensure they are valid. Sets **Motion Mode** to G3.	Motion Mode
Set X/Y Plane	G17	Causes the arc movements in G2/G3 to be performed in the machine's X/Y plane	Plane Select
Set X/Z Plane	G18	Causes the arc movements in G2/G3 to be performed in the machine's X/Z plane	Plane Select
Set Y/Z Plane	G19	Causes the arc movements in G2/G3 to be performed in the machine's Y/Z plane	Plane Select
Set Reference Position #1	G28.1	Write spindle's current absolute position (in machine coords) to EEPROM as the "reference position." This value is persistent across power cycles.	N/A
Move to Reference Position #1	G28	Move rapidly (G0) to the reference position, possibly after first moving to an intermediate position. If intermediate position is specified, movement to stored position is performed ONLY in provided axes. Intermediate position can be determined either relative to current position (G91) or relative to WCS zero (G90)	N/A
Set Reference Position #2	G30.1	Same as G29.1, but with a different memory location. Allows storage of a second reference point.	N/A
Move to Reference Position #2	G30	Same as G28, but with a different memory location. Allows use of a second reference point.	N/A
Probe Until Clear, Alarm If No Clear	G38.4 X/Y/Z(-)*n* F*x*	Performs specified move while monitoring electrical probe circuit. Probe must start with circuit closed (tripped). If circuit opens during movement, movement immediately stops. Behaves very similarly to G38.2. Sets **Motion Mode** to G38.4.	N/A

Probe Until Clear, Proceed If No Clear	G38.5 X/Y/Z(-)*n* F*x*	Performs specified move while monitoring electrical probe circuit. Probe must start with circuit closed (tripped). If circuit opens during movement, movement immediately stops. Behaves very similarly to G38.3. Sets **Motion Mode** to G38.5.	N/A
Tool Length Offset	G43.1 Z*n*	Applies Z offset to all motions to account for a change in tool length. The idea is that, if you probe with one tool and then change to a different tool, you can use G43.1 to avoid re-probing. Writes to same EEPROM as WCS registers, so persists across power cycles.	N/A
Clear Tool Length Offset	G49	Clears the tool length offset by setting it to 0 in EEPROM.	N/A
Coordinate System Offset	G92	Allows you to set current spindle position to a specific coordinate value, thereby moving the zero point for the current coordinate system (and all other coordinate systems, including G53) to a new position. This can be used to store an additional offset beyond G54-59. Clears on reset.	N/A
Clear Coord Systm Offset	G92.1	Clears any current offset set by G92	N/A
Feedrate Minutes / Unit	G93	Causes F*n* feedrate change commands to be interpreted as “minutes per unit.”	Feedrate Mode
Feedrate Units / Minute	G94	Causes F*n* feedrate change commands to be interpreted as “units per minute.” This is the default value.	Feedrate Mode
Line Number	N*n*	Can be run alone or included with any other G-code command. Has no effect on movement, but is reported in status reports. Useful for debugging.	N/A
Tool Change	T*n*	Automatic tool change, but CR-1 has no tool changer, so code is ignored.	N/A
Pause Program	M2	Pauses execution until the resume ~ command is set. Commands can still be queued up, and will be executed upon resumption. Behaves similarly as the “feed hold” ! command, except that it will not “interrupt” queued motion.	N/A
End Program	M2	Returns [MSG: Pgm End] and sets modal state to: G1, G54, G17, G90, G94, M5. No other modal state values are changed. Does not actually stop queued execution.	N/A
Stepper High Power Mode	M17	Special CR-1 M-command. Places all steppers into high power mode, starting at the next motion commands, and maintains high power mode until the steppers next become idle. Will quickly brownout the mill or overheat the motors. Intended use is to free a stuck axis.	N/A
Turn Off Steppers	M18	Special CR-1 M-command. Turns off all steppers until next motion command. Intended use is to allow manual manipulation of steppers without unplugging mill. Next motion command automatically re-enables steppers.	N/A

9.2: grbl Commands and Settings

These tables show special non-G-code commands that can be issued to grbl. Generally these have to do with exerting special machine-specific control, or retrieving machine status.

9.2.1: Dollar Commands:

These commands allow you to either retrieve information about the state of grbl and / or the CR-1 it is running on, or to perform a special operation like homing. These commands can only be run when grbl is otherwise idle – i.e. not during movement. They are called "dollar commands" because they all start with a $.

Name	Command	Description	Response
Home	$H	Finds the machine origin using the limit switches. The Z axis is homed first, followed by the X and Y axes homing simultaneously. Homing is one way to unlock grbl after a powerup or reset.	None
Home Single Axis	$HX $HY $HZ	Finds the machine origin using the limit switch on the specified origin only. This will also unlock grbl after a powerup or reset – be careful not to rely on the position of the other two axes before homing them as well.	None
Unlock	$X	Unlocks grbl after a powerup or reset, without requiring a homing cycle. Unlocking grbl without previously homing arbitrarily sets the machine origin to the machine's current position.	None
Autolevel X Table	$L	Uses the two X motors and limit switches to return the table to its programmed level state. Homes Z before leveling. Performs three leveling adjustment operations back-to-back. Reports the detected skew from each one. Note that "level" here means "parallel to the gantry's YZ plane"	Example: [0.120mm] [0.050mm] [0.002mm]
Store X Table as Level	$LS	Detects the X table's current state of level (by moving it up once and detecting where each of the two limit switches trip) and then writes the difference to EEPROM as the new "level" state. Should not be used unless you know what you're doing.	None
Jog	$J=*jog cmd*	Issues a special movement command called a "jog". Jog commands are always in G1, do not affect the parser state and can be cancelled in real-time by a cancellation command. As these commands are more like move commands, they are covered in more detail in section **9.1.2.**	None

Report Version Info	$I	Returns information about the current hardware, based on values programmed into the control board at the factory. Format is: • Grbl:<version> - specifies grbl version • CR:<version> - specifies the CR hardware version • PCB:<version> - specifies the PCB version • YMD:<date> - specifies the 328p firmware release date	Example: [grbl:1.1g CR:3A PCB:3B YMD:20240318]
Display EEPROM	$E	Dumps a map of the EEPROM	Example: x0 11 0 ... x16 69 0 ... ...
Report Modal State	$G	Returns a list of the currently active parser state. Refer to section **9.2.2** to review the parser state.	Example: [GC:G0 G54 G17 G21 G91 G94 M5 M9 T0 F0 S0]
Check G-Code	$C	Places grbl in “check code mode”. All standard G-Code commands are accepted but not actually run (machine does not move.) Commands with syntax errors return errors. Useful for testing code for typos. Toggle-type command – run again to turn the mode back off.	[MSG:$C:ON]
Report Offsets	$#	Shows WCS offsets, G28/G30 offsets, G92 offsets, and last probe result	Example: [G54:10.000,10.000,20.000] [G55:10.000,5.000,30.000] ... [G59:0.000,0.000,0.000] [G28:0.000,0.000,0.000] [G30:0.000,0.000,0.000] [G92:0.000,0.000,0.000]
Report Parameters	$$	Shows all stored parameters. Review table **9.2.3** to see what each parameter means.	Example: $30=8500 (rpmMax) $31=1360 (rpmMin) $100=400.000 (x:stp/mm) ...

Display Startup Lines	$N	Shows two possible lines stored to run upon every power-up or reset.	Example: $N0= $N1= (No startup lines defined)
Store Startup Line	$N*k*=*value*	Store a line of G-Code which runs on every power-up or reset. *k* is either 0 or 1; only two lines allowed to run on startup. Run "$N*k*=" to assign an empty line to memory, i.e. clear the line.	None
Store Parameter	$*n*=*value*	Write *value* to parameter *n*. See table **9.2.3** for a list of parameters and what values they will accept.	None
Zero Offsets	$RST=#	Clears all offsets reported in $# - G5[4-9], G28, G30, G92. Also resets all parser state values to their default	[MSG:Restore:defaults]
Restore Parameters	$RST=$	Reset all parameters reported in $$ to their factory defaults.	[MSG:Restore:defaults]
Restore All	$RST=*	Calls both $RST=# and $RST=$ to perform a full factory reset	[MSG:Restore:defaults]

9.2.2: Parser State

This table lists each "modal group" tracked by grbl. The bolded item in each list is the default value for that group. To learn more about these commands, review section **9.1**.

Modal Group	Group Values	Notes
Motion Mode	**G0**, G1, G2, G3, G38.2, G38.3, G38.4, G38.5	Determines how **Basic Movement Commands** will be interpreted
WCS Selected	**G54**, G55, G56, G57, G58, G59	Determines what zero point absolute movement commands will move in reference to
Plane Select	**G17**, G18, G19	Determines arc plane. Rarely used.
Input Units	G20, **G21**	Determines how input numbers are interpreted (inches or mm)
Move Type	**G90,** G91	Determines whether move commands are interpreted as absolute or relative moves.
Feed Rate Mode	G93, **G94**	Determines whether feed rate commands are interpreted as units / minute or minutes / unit.
Spindle State	M3, M4, **M5**	Determines whether the spindle is spinning or not, and in what direction
Coolant State	**M9**	CR-1 does not use coolant, always returns M9
Tool Number	**T*n***	Determines what tool is active. Has no effect.
Feed Rate	**F*n***	Determines current feed rate
Spindle Speed	**S*n***	Determines current spindle speed

9.2.3: Parameters

These parameters are values written to the Arduino's EEPROM and are used by grbl to track / configure machine operation. This table covers many (but not all) of the parameters. The bolded item in each list is the default value for that parameter.

Name	Parameter	Description	Possible Values
Idle Delay	$1	Two possible values: 100 allows steppers to enter power-saving mode when idle. 255 blocks power saving mode.	**100**, 255
Unit Report	$13	Determines in what measurement system units are reported. Does not affect how incoming units are *interpreted*.	**0**: display units in mm 1: display units in inches
Soft Limits Enable	$20	Enables or disables soft limits. When active, soft limits throw alarm when a submitted G-Code command would go outside the soft limit. When off, machine will happily move until it hits hard limit and crashes. See $130, $131, $132 for add'l details.	0: soft limits disabled **1:** soft limits enabled
Hard Limits Enable	$21	Enables or disables *hard limits*. When disabled, limit switches are ignored except when homing. This means that machine will happily crash and grind away against the edge of their axis.	0: hard limits disabled **1:** hard limits enabled
Require Unlock	$22	When disabled, grbl will accept commands immediately after power cycle / reset without requiring unlock ($X o r $H). When set, one of these commands is required before G-Code will be accepted	0: unlock requirement disabled **1:** unlock requirement enabled
Esteps	$100 $101 $102	Configures the esteps per mm for each axis ($100 = X, $101 = Y, $102 = Z). Set at the factory, recommended you do not modify.	*Decimal*
Max Velocity	$110 $111 $112	Configure the maximum velocity allowed for each axis ($110 = X, $111 = Y, $112 = Z). Unit is mm/min. Default values are slightly low to ensure mill does not lose steps during normal operation.	*Decimal*, 100 - 4500
Max Acceleration	$120 $121 $122	Configure the maximum acceleration allowed for each axis ($120 = X, $121 = Y, $122 = Z). Unit is mm/min^2. Default values are slightly low to ensure mill does not lose steps during normal operation.	*Decimal*, 10 - 1000
Max Travel (Soft Limit)	$130 $131 $132	Defines the maximum travel distance – i.e. the soft limit – for each axis. Unit is mm.	*Decimal*,

9.2.4: Real Time Commands

These special commands will always return a result and / or perform an action in real time, regardless of whether grbl is executing other queued commands or not. They tend to report on or control the internal state of grbl itself.

Name	Command	Description	Response
Soft Reset	| *(pipe)* *0x18*	Resets grbl's internal state, EXCEPT for current machine position, to its default power-up state. Can be used to recover from an alarm. Either the | or the extended ASCII *0x18* can be sent to trigger the soft reset.	None
Status Report	?	Provides an immediate report on the mill. Contents include: • Mill status (Idle, Alarm, Jog, etc.) • Current machine coordinates (X,Y,Z) • Grbl buffer status, two numbers. First number reports number of lines in progress or queued (max 14), second number reports number of chars available in receive buffer (max 127). • Line number: current line number being executed (if line has a number assigned) • PXYZ: each char is 0 by default, is changed to its corresponding character if that particular switch is tripped. P = probe, XYZ = limit switches • WCS offset: displays stored offset values for the current WCS. Displayed once for each ten '?' requests	Example: <Run|M:-86.000,-2.015,-0.500|B:10,127|L:|0000>
Feed Hold	!	X/Y/Z axes all decelerate to stop. Spindle status does not change. Queued commands remain in planning buffer, and new commands can be enqueued, but no linear movement commands execute until Resume command sent. Has no effect when homing or leveling bed.	None
Resume	~	Restores movement after a feed hold. Command that was in progress when feed hold was applied will complete, followed by any other commands in the buffer.	None

Help	$	Lists several of the available grbl commands.	[? status] [$H home] [$X unlock] [$G state] [$I version] [$L levelX] [$C check] [$# offsets] [$$ settings] [$_=_ set]

9.2.5: Differences From Base Grbl

Coast Runner CR-1's grbl is a forked version of grbl 1.1g (https://github.com/gnea/grbl). From an interfacing perspective, CR-1's grbl is identical to the base grbl, except as noted below:

Condition	grbl1.1g Behavior	CR-1 Behavior	Notes
Stepper Motor Power Levels	Either on or off	Either high, normal, low or off	Level is automatically selected. M17 / M18 can manually change stepper behavior.
ERROR Messages	Code only (e.g. ERROR:9)	Code (e.g. ERROR:9) plus human readable text (e.g. [MSG:$H])	See section **9.4** for list of errors and what they mean.
ALARM Messages	Code only (e.g. ALARM:1)	Code (e.g. ALARM:1) plus human readable text (e.g. [MSG: LIM X])	See section **9.5** for list of alarms and what they mean.
Alarm Behavior	Auto-resets (except limits)	Requires manual soft reset	
Soft Reset	Only extended ASCII *0x18*	*0x18* or | *(pipe)*	Allows user to easily perform soft reset
Syntax Error Response	Only reports ERROR:2	Reports ERROR:2 plus an [echo:_____] message echoing input and a [MSG:_____] message guessing error.	
Auto-Level	Not present	Run $L to autolevel X	
Probing Trip Method	Uses polling method	Uses interrupt-driven method	CR-1's interrupt-based probe trip is more reliable
Coolant	M7/M8/M9 supported	M7/M8/M9 allowed but ignored	CR-1 has no coolant, so no action occurs
$I Formatting	grbl version plus options	grbl version plus CR-1 specific information	See table **9.2.1** for details on $I
? Formatting	Review https://github.com/gnea/grbl/wiki/Grbl-v1.1-Interface#real-time-status-reports	Same as grbl, except probe and limit status is either 0 or the corresponding letter P, X, Y or Z	

9.3: grbl States

The following table describes the various states grbl may be in.

State Name	State Description	Notes
Idle	Waiting for any command	When interacting directly with grbl, most of the time grbl will be "idle"
Alarm	Either initial startup or alarm encountered. Home or unlock to resume.	grbl always initiates in an alarm state, and this state can also be reached due to an ALARM triggering. In either case, when grbl is in an "Alarm" state, no commands will be accepted except for a $H (home) or $X unlock.
Homing	Performing a homing cycle, won't accept new commands until complete.	When the CR-1 is running a homing cycle, grbl enters this special state and refuses to respond to any messages until homing is complete.
Jog	Performing jog motion, no new commands until complete, except Jog commands.	When a jog is being performed, grbl enters this state. The only commands that will be accepted are either new jog commands or jog cancel commands.
Check	Check mode is enabled; all commands accepted but will only be parsed, not executed.	"Code check" mode is entered by running $C. All commands entered in this state will be evaluated but not actually executed. This is a good way to test a program for typos / errors.
Cycle	Running G-Code commands, all commands accepted, will go to Idle when commands are complete.	When G-Code commands are actively executing (machine is moving) grbl enters this state. New commands can be issued.
Hold	Pause is in operation, resume to continue.	When grbl is set to feed hold (!) the state changes to "hold." Some new commands will be accepted, but the CR-1 will not move until the pause is released (~).

9.4: grbl Errors

Error #	Error Message	Description	Example
1	G-code missing Let	Encountered a number without a corresponding letter	10
2	G-code missing num	Encountered a letter without a corresponding number	X
3	$UNK	grbl '$' command not recognized or supported	$K
4	-#	Invalid negative value provided.	F-5000
5		Homing, not enabled via settings, did not home.	*None*
6		Minimum step pulse time must be greater than 3 microseconds.	*None*
7		EEPROM read failed. Auto-restoring affected EEPROM to default values	*None*
8	Not idle	'$' command cannot be used unless grbl is 'Idle'.	G91 G1 Y-5 F20 $H
9	$H disabled	All commands locked out during alarm or jog state. Home or unlock required.	Any command on start before $H or $X
10		Soft limits cannot be enabled unless homing is also enabled	*None*
11	2long	Max characters per line exceeded. Maximum chars is 79	(Any command longer than 79 chars)
12	*None*	grbl '$' setting value caused the step rate to exceed 30kHz, which is not allowed.	$100=5000
13	*N/A*	Safety door detected as opened and door state initiated (not applicable for CR-1)	*N/A*
14		Build info or startup line exceeded EEPROM line length limit. Line not stored.	*None*
15	jogLIM	Jog target exceeds machine travel.	$J=G91 X-500 F100

16		Jog command missing '=' or contains prohibited G-code	*None*
17	*N/A*	Setting disabled, laser mode requires PWM output (not applicable for CR-1)	*N/A*
20	G-code bad	Unsupported or invalid G-code command found	G1000
21	G-code conflict	More than one G-code command from same modal group found	G0 G1
22	G-code F?	Feed rate not set or undefined	(Before setting feed) G1 G91 Y-5
23	G-code missing num	G-code requires integer value	G59.2
24	G-code conflict	Line contains two commands that both specify movement (or other axis operation)	G0 G0 X-10
25	G-code conflict	Repeated g-code word found in line	F100F1000
26	G-code missing axis	G-code command missing expected movement command	G38.2 G91 F100
27	G-code missing Ln	Invalid line number	N1000000000
28	G-code missing word	Missing a required value, word or number	G4
29	G-code G59	Some G-code interpreters allow the use of WCS registers above G59 (sometimes labeled G59.2, G59.3, etc.) but grbl does not	G10 L20 P7 Z0
30	G-code missing G0|G1	G53 only allowed with G0 / G1 motions	G53 G3
31		Unexpected axis word found in block when no command or current modal state requires them	*None*
32	G-code missing word	G2 / G3 arcs require at least one in-plane axis word	G2 Z10 F10
33	G-code missing targ	Motion command target is invalid	G2 X-1000 Y-1000 I-10 J-10 F20
34	G-code missing R	Arc radius value is invalid	G2 X-10 Y-10 R-10 F10
35	G-code missing word	G2 / G3 arcs require at least one in-plane offset word	G2 Y-10 F10

36	G-code word	Unused value words found in block	L1000
37	G-code G43	G43.1 dynamic too length offset is not assigned to configured tool length offset	G43.1 X10
38	*None*	Tool number greater than supported value	T1000

A grbl error occurs when grbl receives a command that it cannot understand, or does not know how to respond to. The most common sorts of errors are typos or syntax errors, but errors also occur when a required command component is missing, and in other situations.

An error is detected and returned when a command is first received, before it is inserted into the planning buffer. When an error occurs, the following things happen:

1. grbl discards the entire line and does not attempt to execute it
2. grbl responds with three lines:
    a. An “echo” line which repeats the command. Example: “[echo: G35]”
    b. A message which describes the error. Example: “[MSG:G-code bad]”
    c. The actual error code. Example: “error:20”
3. grbl will continue to operate as normal

Errors are not fatal to grbl, but they do stop execution in CRWrite if encountered when processing a G-Code file (for example, in guided mode, or when running a G-Code file in manual mode.) This is because an error in a G-Code file indicates that something has gone wrong, and continuing in the face of this could produce undesirable behavior.

9.5: grbl Alarms

Alarm #	Alarm Message	Description	Position Guaranteed	Example
1	Limit	Hard limit error – an axis ran into a limit switch	No	Hard to recreate via G-Code.
2	Soft Lim	Soft limit. Requested motion would exceed machine travel.	Yes	G53 G0 X-100
3	Reset	Reset occurred while machine was moving	No	G91 G1 Y-10 F10 |
4	Probe	Probe is not in expected starting state	Yes	G38.4 G91 Y-10 F10
5	Probe	Probe did not change state during travel	Yes	G38.2 G91 Y-1 F100
6	Home	Reset occurred during homing routine	No	$H |
7	N/A	Homing fail due to safety door opening. Not pertinent for CR-1	No	N/A
8		Limit switch did not reset during homing routine	No	*None*
9		Limit switch did not trip during homing routine	No	*None*

A grbl alarm occurs when grbl receives a command that it can understand, but either refuses to execute, or attempts to execute but fails due to external conditions.

Alarms only occur when grbl actually attempts to execute the offending command. When an alarm occurs, the following things happen:

1. grbl immediately halts all motion, including spindle motion.
2. grbl responds with three lines:
    a. A message which describes the alarm. Example: “[MSG:limit Y]”
    b. The actual alarm code. Example: “ALARM:1”
    c. The message “[MSG: Reset to cont]”
3. grbl enters the Alarm state, within which it will refuse all commands until a soft reset is issued.

Because alarms unexpectedly stop motion, loss of steps may occur. This is why grbl enters the alarm state – the only safe move after possibly losing steps is re-homing, which must be done to exit the alarm state. This said, some alarms do guarantee that current machine position is preserved, as noted by the “Position Guaranteed” column in the table above. For these sorts of alarms, it is safe to unlock ($X) the mill to exit the alarm state instead of homing

9.6: .crproj formats

To be provided in a future version of the manual

9.7: CR-1 Technical Specifications

9.7.1: Axis Lengths and Signs

The CR-1's axes have the following travel limitations:

Axis	Min. Coordinate (Machine Home)	Max. Coordinate
X	-86.5mm	0mm
Y	0mm	-241.5mm
Z	0mm	-78.5mm

These axes use the following signs when moving:

Z Axis: Spindle Plunge is **NEGATIVE**, Spindle Retract is **POSITIVE**
Y Axis: Gantry Left is **NEGATIVE**, Gantry Right is **POSITIVE**
X Axis: Table Up (Spindle "Down") is **POSITIVE**, Table Down (Spindle "Up") is **NEGATIVE**

These signs are consistent with milling standards, which always moves in negative space.

Note that X is dissimilar to the other two axes. This is because X does not move the *spindle*, but the *table*, and as such operates "backwards".

This "backwards operation" can be understood when one realizes that moving the table in one direction is equivalent to moving the spindle in the *opposite* direction.

Notes on Travel Buffers:

When homing, the CR-1 finds its minimum coordinate, but then parks the gantry 0.5mm away from that minimum coordinate in all axes (-86, -0.5, -0.5).

Similarly, when jogging, CRWrite only allows jogging up to 1mm away from the extreme end of each axis – meaning, for example, that CR-1 can only jog to -240.5mm at the extreme end of Y.

If you wish to position the spindle beyond these "buffers", you can do so with direct G-Code commands (e.g. *G53 G0 Y-241.5*)

9.7.2: Spindle Speed

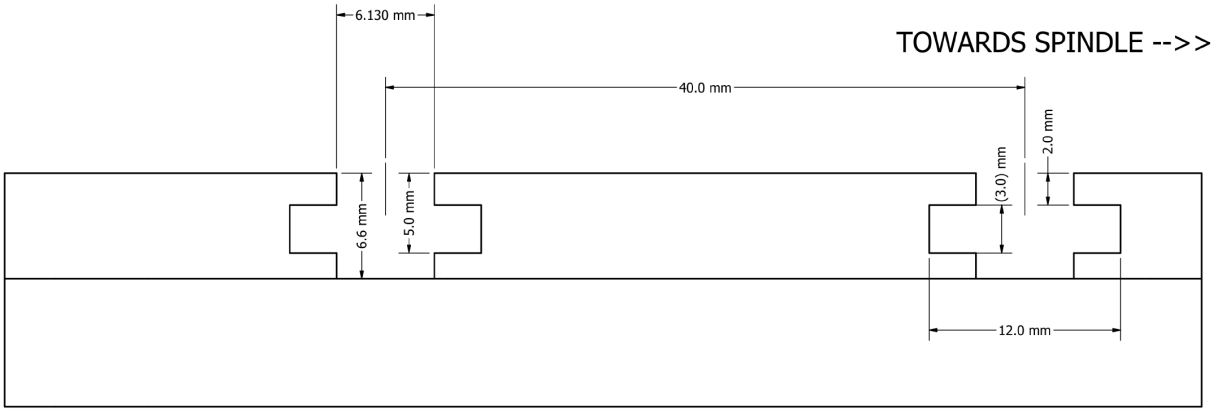
CR-1's spindle speed is defined as follows:

Minimum Speed	Maximum Speed
1500 RPM	8000 RPM

9.7.3: T-Slot Table

The CR-1's table has two T-slot rails which can be used for attaching fixtures.

This table's profile can be seen below:



We use T-Slot nuts from Fairway Fasteners (model JPPNM04016).

Details:

- M4 thread
- 11mm x 3.14mm x 15.87mm

Equivalent models include model 25-1941 and model 25-1951 from 80/20 Store.

9.8: Glossary

Term	Definition
328p	The Arduino control board uses a 328p microprocessor as its main CPU. When we use the term 328p, we are referring either to the Arduino Uno itself, or to the grbl firmware that runs on the 328p.
32M1	We have a custom shield attached to the 328p which, among other things, controls the spindle. This custom shield contains a 32M1 microprocessor dedicated to spindle control, and so we use the term "32M1" to refer either to the custom shield, or to the firmware that runs on the 32M1.
Absolute	The term "absolute", as in "absolute coordinates" or "absolute movement", is used to refer to moving the CR-1's spindle (or measuring the position of this spindle) within a coordinate system centered at a zero point that does not change with movement. "Absolute movement" means that, if we tell the CR-1 to move to a specific coordinate (an "absolute coordinate") then the CR-1 will move to the exact same point in space regardless of where the spindle was located when we started movement. The **G90** modal state enforces absolute movement – see **section 9.1.1.** This is in contrast to "relative movement" – see the entry for "relative" below. See also the entries for "machine coordinates" or "work coordinates", both of which are absolute systems of measurement / movement. The fact that CNCs may have different "absolutes" depending on the coordinate system we are using may be confusing. There is no "one true absolute" for a CNC mill – we use the term solely to refer to the fact that we're using a coordinate system with a specified zero position in the first place.
Alarm	An alarm is a condition detected and reported by grbl in which it either refuses, or is prevented in some way, from attempting to carry out a valid command. For example, a soft limit alarm occurs when grbl refuses to attempt to move an axis past its limits. Another example is when a probe sequence does not trip – in this case, an alarm is thrown due to an unexpected outcome. See **section 9.5** to learn more about alarms.
Bed	Sometimes the table is referred to as the "bed". See entry for *table*.
CAD	"Computer Aided Design" (or "Drafting") is software that allows you to create, represent and manipulate models, schematics or designs. For our purposes, these are three-dimensional models. Often, though not always, CAD software is paired with CAM software. We also sometimes use the term "CAD" to refer to a file containing data representing a 3D model, such as a STEP file.

CAM	"Computer Aided Manufacturing" (or "Machining") is software that allows you to generate code that controls machine tools – almost always G-Code. CAM software allows you to take a 3D model and create G-Code to fabricate parts or all of the model from stock. Sometimes CAM software is paired with CAD software. We also sometimes use the term "CAM" to refer to the "source" file containing the 3D model and associated toolpaths and parameters, such as an F3D file. See **Section 6** to learn more about CAM.
Collet	A collet is a small, round "cone" with a hole through the center. The collet is inserted into the CR-1's spindle, and then the shank of a tool is inserted into the collet. When the collet is tightened with the collet nut, it grips the tool's shank securely and perfectly centers it I n the spindle.
Collet Nut	A collet nut threads onto the threaded CR-1 spindle and, when tightened, causes the collet inside the spindle to tightly grip the installed tool. A collet must be installed ("snapped in") to the collet nut.
Control Board	We use this term to refer to the Arduino Uno that runs grbl and controls the CR-1. Sometimes we use the term to group the Uno and its 32M1 shield together, but usually we refer solely to the Arduino Uno itself.
Control Software	We use the term "control software" to refer to software that runs on a user-accessible computer and facilitates the user's operation and control of the CNC mill, almost always by helping that user to send (or automating the sending of) G-Code commands. CRWrite is the control software for the CR-1. Technically, grbl could also be referred to as "control software", but for us the term "control software" always refers to CRWrite.
crproj File	A "crproj File" is a specifically formatted zip file (renamed to have a "crproj" file extension) that contains a specific folder structure and manifest.yml file. These files define a set of jobs and steps that can be displayed and run in CRWrite's Guided Mode. See **section 4.1** to learn more about the Guided Mode interface, and see **section 9.6** to learn more about the crproj file format.
Cutcode	We use the term "cutcode" to refer to a crproj file. Rarely and where necessary, we use the term to refer to the collection of G-Code files (as well as related necessary information about which tools to install when, orientation changes, etc.) that make up a project, even if this information has not been organized into a crproj file, but almost always the term "cutcode" refers to the organized crproj file containing this information.
DRO / Digital Readout	A DRO (stands for "digital readout") is a visual interface showing information about a machine tool – in this case, showing the location coordinates of the CR-1. The CR-1 DRO shows both the "machine coordinates" and the "work system coordinates" for the CR-1. See **section 5.1.2** to learn more about CRWrite's DRO.

E-Stop / Emergency Stop	An emergency stop refers to user action taken to abort execution of in-progress G-Code. There are two types: soft estops and hard estops. See those entries to learn more.
Error	An error is a condition detected and reported by grbl in which grbl cannot understand, or has insufficient information, to execute a submitted command. The most common example of this is a typo, but errors can also occur if an incomplete command is submitted, and in other circumstances. See section 9.4 to learn more about errors.
Feed / Feedrate	Feed refers here to the rate at which one of the moving parts of the machine (the gantry, table, or spindle) moves in X/Y/Z. It can be measured in either mm/min or in/min depending on which units we are operating in.
Firmware	Firmware is software that runs on an embedded controller, like an Arduino Uno, and which facilitates control of or operates hardware attached to said controller. There are two firmwares
Finished Product	The "finished product" (or "project", etc.) is a workpiece that has been machined to completion. The goal of all machining projects is to produce a finished product.
Fixture	A fixture is any implement or mechanism that holds a workpiece in place, preventing it from moving or flexing while cutting. Fixtures can also facilitate holding the workpiece at the right angle / orientation. Examples include clamps, vises, jig plates, etc. The term *jig* is often used interchangeably with *fixture*, although this is not technically correct, and Coast Runner strives to use *fixture* whenever it is correct and possible. See **section 4.4** to learn more about fixtures.
G90 / G91 / G53 / G54 / etc., "Moving In"	When we say "moving in G91" or "moving in G53" we mean that the mill is executing movement involving this particular modal state specified. "Moving in G91" means that we are moving relatively. "Moving in G53" means that we are moving absolutely with respect to the machine zero. "Moving in G54" means that we are moving absolutely with respect to G54's WCS zero. "Moving in G0" means we are moving rapidly. You can make your own version of this phraseology with any other modal state command.
Gantry	The gantry is the part of the CR-1 which holds the spindle and translates in the Y axis (left and right). It rides on two shafts and is driven by a single belt-driven ballscrew.
G-Code	G-Code is a computer language that encodes commands to manipulate and control machine tools. There are many different "dialects" of G-Code – for our purposes, when we refer to G-Code, we are talking about "grbl-flavored G-Code", the dialect understood by the CR-1. G-Code can be written manually, or generated using a CAM software such as Fusion360. G-Code is always spelled with an uppercase G and C, and a dash in between.
G-Code File	We use the term "G-Code File" to refer to a text file consisting of G-Code commands. These files can be "uploaded" to Manual Mode and executed, at which point each command in the file will be sent to the mill one after the other. This is far more convenient than manually entering each command in the serial terminal. See **section 5.1.2** to learn more about CRWrite's G-Code file upload functionality.

grbl	grbl (always spelled in all lowercase) is firmware that runs on the 328p. It accepts G-Code commands via serial communication and returns information along the same connection. See **Sections 3.2, ???** and **9.2** to learn more about grbl.
Hard Limit	A "hard limit" is the physical limitation preventing an axis from moving any further – attempting to move it any further causes a physical crash, and likely loss of steps. The grbl firmware uses soft limits to prevent the machine from hitting a hard limit, though hard limit errors are certainly still possible under some circumstances.
Hardware E-Stop	A hardware estop occurs when a user hits the red button on the left side of the CR-1 mill. This cuts power to the mill's motors and interrupts communication between the computer and grbl. We recommend using the hardware estop in critical situations – e.g. the tool is bogging down, or is about to crash. After a hardware estop, the software will usually lock up and will need to be restarted.
Home / Home Position	The home position is the position that the CR-1's spindle occupies after running a homing sequence. This is the "zero point" for the machine coordinate system.
Homing	Homing is an automatic sequence that allows the mill to "acquire" knowledge of its own position. Homing should almost always be performed at the beginning of every milling session. See **section 4.2** for more information about homing.
Indexing	We use the term "Indexing" to refer to any process that allows the CR-1 to accurately "know" where a workpiece is located. Almost always this is done via probing, although the term can also correctly be used to refer to techniques where a fixture (which has itself likely been probed) that serves as a reference surface. We often use the term "indexing" to refer to manual probing, although the term is not exclusive and can also refer to electrical probing as well.
Jig	A jig's technical definition is any implement or mechanism which facilitates the correct position or movement of a tool or workpiece during a cut. So, a plate containing holes to help you position your drill is a jig, as is a tool that helps you manually move your workpiece during cutting operations. The term "jig" has taken on a meaning synonymous to "fixture" in the machining community at large, but Coast Runner tries to only use the term in its correct context, preferring instead "fixture" to refer to implements whose sole job is to secure an item in stock.
Job	In the context of a crproj file, a job is a set of steps. A job can be "the full project", e.g. all the steps that are required to mill the project from start to finish, or it can be a subset of "the full project". Different jobs can also represent different variations or options for how the project is milled.
Leveling	"Leveling" refers in this case to ensuring the table is square across the entire Y axis. An autoleveling sequence can be run with $L to re-level the table to its factory settings.

Limit	The term “limit” usually refers to the extreme ends of the three axes in which the mill can move. Attempting to move past these extreme ends is referred to “hitting a limit” and can cause a “limit alarm.” CR-1 has both “soft” and “hard” limits – see the entries for both these terms.
Limit Switch	Limit switches are, in this case, sensors which detect when the mill has moved up to one of its axis limits. CR-1’s limit switches are optical switches – when the mill moves up to the limit switch, a small metal tab enters the switch and is detected. The Y and Z axis each have a single limit switch, and the X axis has two switches, which are used for both standard limit detection and for leveling.
Machine Coordinates	Refers to a coordinate system whose zero point at the mill’s home position. Moving in “machine coordinates” means to move in this coordinate system. The **DRO** in manual mode always displays the current machine coordinates. See the **G53** entry in **section 9.1.1** to learn more about how to move in machine coordinates.
Model	We use the term “model” to refer to a 3D representation of an object. Can also refer to any file that contains model data, including STEPs, STLs and others.
Offset	The term “offset” is in two circumstances: 1. It can refer to the modifier to an axis in certain G-Code commands, e.g. G10 L20 P1 X-5 has an offset on the X axis of -5. 2. It can also refer to the contents of a WCS register, e.g. we might say you “write your offsets to WCS”. This is because the actual data written to the WCS is the distance from machine zero to WCS zero.
Probe Point	A “probe point” is almost always synonymous with a “zero point”, though the use of this terms carries a confirmation that this zero point was found through probing (which it almost always is).
Probing	Probing is the process of using the tool to locate and save the position of a workpiece, fixture, or something else in the mill. We often use the term “probing” only to refer to *electrical probing*, although it can technically refer to manual probing as well.
Project	We use the term “project” to refer to everything involved in taking a piece of stock and using engineering operations to transform that stock into a completed product. This most commonly refers to the overall collection of tools, code and knowledge that is required to perform these operations.
Register	A “register” is a memory location – in this context, this term almost always refers to one of the six writable “WCS registers” (labeled G54, G55, G56, G57 and G58) to which a zero point can be written, and based on which a WCS is defined.

Relative	The term “relative”, as in “relative movement”, refers to movement specified in relation to the spindle’s current position, and not to any external fixed zero position. The **G91** modal state enforces absolute movement – see **section 9.1.1.** Executing a relative movement command multiple times will produce movement each time, because we are saying not “move to this specific location” but “move in this direction X units”.
Serial Terminal	A “serial terminal” is a software interface that facilitates the writing and reading of serial data to a remote unit – in this case, sending G-Code to (and receiving responses from) the CR-1. Almost always, serial terminals are text-based and this is the case here as well. See **section 5.1.2** to learn more about CRWrite’s serial terminal.
Shank	The shank is the non-cutting end of a tool. Sometimes the shank will have a different diameter than the cutting end of the tool.
Soft Limit	A soft limit refers to a software-enforced limit beyond which grbl attempts to prevent the spindle from travelling. The soft limits are generally a small distance “inside” the hard limits. Issuing a command that attempts to make the move past the soft limit will trigger a soft limit alarm.
Software E-stop	A “software estop” refers to the user clicking the “Stop” button during code execution in either guided or manual mode. Clicking this will issue a reset command (|) which will stop execution in an easily recoverable way. We recommend using the software estop in non-critical situations.
Speed	Speed refers here to the rate at which the spindle is spinning. It is measured in revolutions per minute (RPM). Speed does *not* refer to the rate that any part of the mill moves in X/Y/Z – for this measurement, see *feed*.
Spindle	The spindle is the part of the mill that holds the tool, can spin clockwise or counterclockwise, and can plunge or retract in the Z axis. The spindle’s plunge / retract mechanism rides on two shafts and is driven by a single belt-driven ballscrew. The spindle itself is spun via a BLDC motor which is, again, linked to the spindle itself by a belt. Note that because the spindle is the “business end” of the mill, we often refer to “the spindle” as travelling when we discuss any sort of mill movement, even if the spindle does not itself literally move (for example, when the table moves, we still may refer to this as being “spindle movement” – and from the point of view of a workpiece attached to the table, the spindle does indeed move.)
Squaring Boss	A squaring boss is a protrusion on the bottom of a part that mounts on the table. This protrusion slots into the t-slot and, in doing so, forces the part to mount in a certain orientation. Normally this orientation will be “square”, and so we call it a squaring boss.
Stock	“Stock” is a piece of physical material (metal, plastic, wood, etc.) that we intend to fixture in the machine and cut into a certain shape. Stock is often used to refer to this material in its totally unaltered state (once a cut has been performed, it becomes a *workpiece*) but this distinction is not consistently enforced.

T-Slot	A “t-slot” is a slot in a table which has a geometry shaped like a cross, or a lowercase t. This geometry is useful for mounting things to the table – a t-slot nut is inserted into the horizontal part of the t, and bolts (in our case, M4 bolts) can be screwed into the nut. Squaring bosses can also slot into the top part of the t.
T-Slot Nut	This is a nut that slides into the T-slot on the table. It contains a threaded M4 hole into which an M4 bolt can be threaded, thereby clamping something to the table.
T-Slot Plate	Sometimes the table is referred to as the “t-slot plate.” See entry for *table*.
Table	The table is the part of the mill which can move “up and down” (or “raise and lower”) in the X axis, and upon which parts are fixtured. Our table contains two t-slots to facilitate mounting parts.
Toolpath	We tend to use the term “toolpath” to refer to a single operation in a CAM software that produces tool movement to machine a particular section of the finished product. For example, a single Adaptive cut defined in Fusion360 is a toolpath. When exported, the toolpath becomes “G-Code” or “a G-Code file”
WCS / Work Coordinate System	WCS stands for “Work Coordinate System.” A WCS is essentially a user-defined zero point, enabling absolute movement in a coordinate system other than the machine’s own home position. Almost all cutting code is run with a WCS. WCS’ are defined via indexing.
WCS Coordinates	This term is usually used in contrast to “machine coordinates” – e.g. we are measuring or moving in a coordinate system defined in a WCS.
WCS Register	*See “register”*.
Workpiece	Sometimes used synonymously with “stock”, the workpiece is the material that is being cut into the finished product. Often we only use the term “workpiece” when we are operating on something that has already had some cutting operations applied to it – an intermediary state as it were – although this is not consistently enforced, and we will sometimes refer to raw stock as a “workpiece.”
Work Coordinates	*See “WCS coordinates”*
Zero Point	A “zero point” is a location in absolute space around which a coordinate system is centered – e.g. it is the point (0, 0, 0) in that system. A zero point is defined either according to the data written to a WCS register (for WCS coordinates) or according to the machine’s home position (for machine coordinates.)