Automatic/GRBL CNC User Manual

Multi-Axis Positioning System for Water Blaster Testing

Fort Lewis College Senor Design Team

Spring 2025

Alijah Konikowski

Table of Contents

1 - System Overview

The automated system is driven by a Nucleo F411RE microcontroller running GRBL-Advanced, a fork of the open-source GRBL CNC firmware optimized for STM32 microcontrollers. This firmware interprets G-code commands and translates them into precise motion instructions for each axis of the system.

Motion is executed using Teknic ClearPath closed-loop servo motors, which integrate internal linear encoders that provide accurate feedback on motor position. These motors communicate motion status through the High-Level Feedback (HLFB) signal, enabling multiple safety interlocks within GRBL and precise movement tracking.

Through the graphical user interface (GUI), users can access core features such as homing, probing, toolhead/nozzle control, and spindle management. The GUI also supports manual jogging, macro execution, and real-time visualization of toolpaths. To generate G-code,

many programs are supported, but only the most efficient programs for this machine are highlighted.

The key advantages of this control method include-

Repeatable, consistent tool motion across successive tests.

Complex path execution for high-precision nozzle testing.

Scalable integration with nozzle/spindle controls.

Extensive safety features designed to prevent collisions before and during travel.

2 - Safety Information

- Always maintain a 15 ft buffer between operators and the nozzle per WJTA standards.
- Use the emergency stop switch to cut power and water instantly.
- Ensure water diverter valves and whip restraints are installed before running.
- Be sure all the wiring is clear from pinch points that may tear the harnesses.
- If travel moves are commanded outside the set axis limits, a soft limit will halt progress to prevent gantry collision.
- If soft limits fail to prevent collisions, the system will immediately stop when it hits a hard surface, preventing system damage.

2 – GSender/ LaserGRBL/Universal Gcode Sender GUI Features

2.1 - Startup

Double-check all power cords connected and plug a usb cable from a PC/Laptop into the type b connection exposed through the side of the electrical box. Bootup the preferred GUI and connect to board. Despite the multitude of GUI options available- GSender, LaserGRBL, and Universal Gcode Sender seem to offer the best balance of user-friendly controls with a powerful interface, suited for this application.

2.2 - Axis Mapping

To maintain compatibility with standard G-code generation programs, most of which assume a cartesian X-Y plane, the physical Z-axis of this system has been remapped to the logical Y-axis within GRBL firmware and the Nucleo pin configuration. This adjustment allows tools like Inkscape, JScut, and standard CAM software to generate toolpaths without requiring post-processing or reconfiguration. This allows users to interact with the system as if it were an X-Y machine with the intuition that actual motion occurs in the X-Z plane instead. As a result, when using a GUI like gSender, all Z-axis movements must be made using the Y-axis controls.

This remapping does not impact motion accuracy or system behavior, but it is essential to keep in mind that any "Y" axis inputs will be translated into "Z" coordinates while the established "Z" movements of any executed Gcode will be ignored.

2.3 - Jog Controls

- Use GUI arrows or keyboard to move in X/Z.
- Configure travel distances and move speed for each input.
- Hold shift for rapid movement.

2.4 - Gcode Execution

- Load .gcode file from local PC
- Preview and simulate toolpath
- Click "Run Job" to execute

2.5 - Macros

- Create reusable commands or save common test procedures here.
- To create a reusable macro without generating Gcode first- open a new console output, jog the axes at the preferred speed for the required travel distances, then copy/paste and save the Gcode output as a new macro once testing is completed.

2.6 - Console

Send commands such as-

- $X Unlock GRBL after reset or alarm state
- $H Home all axes using homing cycle
- ?# Show real-time machine status
- ! Pause the current job

- ~ Resume a paused program

  Monitor Gcode outputs, travel motion, and any errors generated in real time.

3 – Gcode Generation Tools

Several G-code generation options are available, depending on application complexity and desired workflow automation.

The following tools are validated for use with the GRBL-Advanced firmware setup, supporting both simple and advanced motion patterns without the need for extensive post-processing.

3.1 - Inkscape + JScut for developing an SVG and processing into Gcode

Best for:

- Complex/Irregular line paths
- Circular motion patterns (supports G2/G3 command outputs)
- Execution of travel motion using existing 2D object files

Workflow:

1. **Create drawing** in Inkscape by uploading almost any 2D file or drawing the vectors directly.
2. **Save as Plain SVG** (important- no metadata).
3. **Upload SVG into JScut** (https://jscut.org) or download all Gcode extensions available for Inkscape on GitHub, allowing Gcode generation and path optimization within the GUI.
4. **Configure tool settings** (tool diameter, travel height, feedrate).
5. **Generate GRBL-compatible G-code** and download.
6. **Load the .gcode file into GSender** for preview and execution.

Notes:

- All CAM software used expects a traditional X-Y plane- the Gcode generated will move the Z-axis with Y-commands, and any Z-commands in the Gcode will be ignored.

- Set "Cut Depth per Pass" and "Final Depth" to zero, limiting unnecessary Z-commands in the generated Gcode.
- Inkscape can export Gcode directly to LaserGRBL, minimizing the number of steps throughout this process.

3.1 - Custom Python Gcode Generation and Logging

Scripts Used:

- GcodeGenerationandLogging.py
- Logging_Gcode.py

Best for:

- Complex mathematical motion paths (e.g., sinusoidal sweeps, parabolic curves)
- Simulates and logs dynamic acceleration, position, and feedrates.
- Generates visual outputs showing time vs position of each system axis.

Workflow:

1. Modify motion parameters inside the CONFIG dictionary (start/end points, velocities, sample density, axis constraints, and type of path to take).
2. Run the script to generate:
    - A .gcode toolpath file
    - A .csv log file containing time-resolved kinematics
3. Review plots of velocity, position, and acceleration automatically generated by the script, double check no system constraints are being exceeded.
4. Load .gcode into GSender for execution.

Notes:

- Sampling resolution and feedrates must be configured based on the test goal (e.g., slow sweeps vs. rapid impulse).
- Built-in error handling prevents unrealistic acceleration or velocity violations, but especially in the case of the custom equation inputs for each axis, this script will throw errors if misconfigured, read all the docstrings before use.

4 – Motor Feedback Integration and Movement Tracking

HLFB (High-Level Feedback) Overview

Each Teknic ClearPath servo motor used in the system integrates an internal linear encoder that provides High-Level Feedback (HLFB) signals. These signals indicate motor status in real-time:

HLFB State	Meaning
HIGH (logic 1)	Motor is enabled and in-range (The system's actual position is within an allowable range of steps compared to the commanded position)
LOW (logic 0)	Motor is not enabled, has triggered an out-of-range error, or the system has hit a hard stop.

The HLFB output is wired to digital inputs on the Nucleo F411RE board, acting in place of physical limit switches. This configuration allows GRBL-Advanced to perform a homing procedure upon startup, moving the axes to set zero positions for the system, improving safety and repeatability. Any size object that will fit can be placed under the machine. During a homing or probing procedure, the nozzle will stop when it contacts that surface, setting the zero position accordingly. After the homing procedure is completed, the system relies on correctly calibrated parameters within GRBL firmware such as steps/mm and backlash compensation. When tuned precisely, the position and velocity of the system should be exactly as commanded by GRBL, making actual data collection an open loop process. It's important these values are recalibrated whenever mechanical or MSP configurations occur, even when just installing different length box rails, as the axis limits must be changed accordingly as well, or soft limits disabled.

Important Notes:

- HLFB behavior is inverted compared to standard GRBL limit switch logic- firmware settings were adjusted to accommodate this.
- Soft limits, hard limits, and homing will be enabled. Soft limits prevent gantry collisions before they occur but can become annoying to change if the box rails of the system are consistently being swapped. In which case, disable soft limits and rely on the hard limits instead, which will shut down the motors when an unexpected hard stop occurs. Homing is required for repeatable testing.

5 - Key GRBL Parameters to Monitor and Adjust

Parameter	Function	Typical Setting	Notes
$100, $101	Steps per mm (X and Y logical axes)	Based on the set pulses/rev of the motor, gearing, and drive mechanics	Must be recalibrated whenever these factors change
$110, $111	Maximum travel speed (mm/min)	Limited by motor performance and safe load conditions	Tune conservatively for high-pressure nozzle tests
$120, $121	Acceleration (mm/s^2)	Defines axis speed ramp-up and ramp-down during travel moves	Adjust to balance speed with mechanical stability
$130, $131	Maximum travel limits (mm)	Physical motion limits for each axis	Critical to update after any mechanical length change, subtract the carriage length from the new box rail
$140, $141	Backlash compensation distance (mm) for X and Y axes	Based on measured mechanical play	Enables improved positioning accuracy across direction reversals
$20	Soft limits enable	1 (enabled) or 0 (disabled)	Prevents software-commanded overtravel
$21	Hard limits enable	1 (enabled)	Required for HLFB-based crash detection- turning this off won't change motor behavior when encountering a hard stop, but the test will still appear to be running.
$22	Homing cycle enable	1 (enabled)	Must remain active for all test setups to ensure repeatability.

Notes

- There is no need to recompile firmware when making these changes, all settings are accessible within the mentioned GRBL-based Gcode senders.
- $140 (X) and $141 (Y) specify the measured mechanical backlash for each axis, defined as the distance the system must "take up the slack" after changing direction.
- These values are automatically compensated for during normal G-code execution, improving precision especially during oscillating movements, repeated reversals, or fine positioning tests.
- No additional speed control is associated with backlash compensation; the compensation moves are blended seamlessly into normal axis motion.

6 – Troubleshooting and Recovery

Even with properly configured systems, unexpected behavior or faults can occasionally occur. This section outlines common issues encountered during automatic operation and recommended steps for recovery.

Symptom	Possible Cause	Recovery Action
GSender will not connect to the board	USB cable not properly seated, board not powered, or incorrect COM port selected	Check all physical USB connections, power supplies, and select the correct COM port in the GUI- sometimes the wrong COM port will connect, but the GUI controls remain shaded. Try another COM port in this case.
Axis refuses to move after startup	GRBL is locked due to alarm state or emergency stop	Send $X unlock command via console and retry motion, this won't work when homing is needed upon startup
Homing cycle fails or axis hits hard/soft limits prematurely	HLFB wiring fault, mechanical obstruction, or incorrect $130/$131 limits	Inspect wiring integrity, verify clear travel path, and confirm limit parameters are updated to current components

Symptom	Possible Cause	Recovery Action
Soft limit alarm triggered immediately after homing	Logic is no longer inverted or the HLFB circuit has a fault	Check limit switch logic in GRBL firmware
Motor stalls or vibrates without moving	Step/Direction wiring reversed	Verify correct Step/Direction/Enable wiring
Unexpected G-code behavior (ignoring Z-axis moves)	G-code or jogging used Z-axis commands instead of Y-axis	Regenerate G-code ensuring correct X-Y output orientation per axis mapping guidelines, don't use the Z axis jogging controls
System freezes during operation	The green lights on the motors turn orange due to power or signal fluctuation, can occur at high commanded speeds due to step pulse frequency being too high	Reset system by pressing E-stop, reboot PC if necessary, and verify acceleration ($120/$121) and speed parameters are within motor capability. Lower the steps/rev of the motors using Clearpath MSP if the issue is noted at high speeds only. Check the 75v power supply is not overheating.

7 – Appendix

Firmware and Configuration Repository

All firmware used for the Nucleo F411RE GRBL-Advanced control system is based on the open-source repository-

GRBL-Advanced GitHub Repository

In addition, all custom firmware specifically developed for this project, including manual mode control and path following operation, is available at:

FLCxStoneAge Project Firmware Repository

This repository includes:

- Customized GRBL-Advanced source code tailored for the system hardware.
- Manual Mode firmware for direct operator control through buttons and LCD interface.
- Configuration files for motor steps, acceleration, and soft limit enforcement.
- System homing procedures, backlash compensation settings, and safety feature integrations.

Users seeking to modify or update the firmware should refer to the documentation provided within the repository for installation and configuration instructions.