Physical

3v3 GND *GPIO*

Setup macros in config

```
macros:
    macro0: G90&G53G0Z-1&G0X0Y0
    macro1: $SD/Run=drill.nc
```

User Input
- Macro 0
- Macro 1
- Macro 2
- Macro 3

Machine control
- Limits
- Probe
- Safety door
- Reset
- Feed hold
- Cycle start

Macros can be used to execute some commands with the click of a button. These commands can be most gcode and FluidNC commands including running a file from the SD card. If you have more than one command, separate them with an ampersand "&". The use of "&" to split a line into multiple GCode commands only works for startup lines; it is not a general GCode feature.

Setup control pins in config

```
control:
    safety_door_pin: NO_PIN
    reset_pin: NO_PIN
    feed_hold_pin: NO_PIN
    cycle_start_pin: NO_PIN
    macro0_pin: NO_PIN
    macro1_pin: NO_PIN
    macro2_pin: NO_PIN
    macro3_pin: NO_PIN
```

- ☐ Control (Inputs)
- ☐ Homing and limit switches
- ☐ Probe
- ☐ Configuring IO pins

Physical
GPIO/I2SO

I2SO only available on controller hardware that implements them, like the 6 Pack.

☐ User outputs allow you to output digital (on/off) and analog (PWM) signals via gcode.

Analog outputs
(PWM works only on GPIO pins)

- ☐ `M67` Analog output, syncronized
- ☐ `M68` Analog output, immediate

A PWM signal is output on this pin. It is controlled by the `M67` command.
`M67 E0 Q23.87` would turn on `analog0` with a 23.87% percent duty cycle.
`M67 E0 Q0` would turn off `analog0`.

Serial Terminal

⊟ *USB*

There are two basic interaction methods with FluidNC over a serial connection:

1. The **grbl line protocol**, with which you stream gcode. Sender software like UGS use this protocol, but also MCUs or Python scripts can stream gcode. It is fully compatible with grbl.
2. **Commands and settings** mostly specific to FluidNC.

Commands are used for actions (like homing or disabling motors) or retrieve information (like gcode offsets). You can get a complete list of them with the `$CMD` command.
All standard Grbl $ commands are supported.

Settings are used to set saved values. Most settings are now in the config file, but there a a few like $Config/Filename that need to be outside the config file. You can see all of them with $S

Fluidterm is a very simple serial terminal that makes it a little easier to interact with FluidNC during setup and configuring. It is not a gcode sender or meant to be a good interface to control the machine. It is included with the releases.

Run Fluidterm
`sh fluidterm.sh`

☐ Fluidterm usage

☐ Communication with GRBL

FluidNC
Generic controller board

IN
- GPIO 1
- GPIO 2
- GPIO 3
- GPIO 4

OUT
- GPIO 5
- GPIO 6
- GPIO 7
- GPIO 8

Standard Stepper

Stepstick/ Trinamic

MOTORS

Step / Direction / Enable / Common

Check your actual pin assignments.

A+ / A- / B+ / B-

External stepper driver

A+ / A- / B+ / B-

Bipolar Stepper

Axis

RC Servo   **Solenoid**

Digital outputs

`M62` Digital output, ON, synchronized
`M63` Digital output, OFF, synchronized
`M64` Digital output, ON, immediate
`M65` Digital output, OFF, immediate

The `M62` & `M63` commands will be queued. Subsequent commands referring to the same output number will overwrite the older settings.

`M64` & `M65` happen immediately as they are received by the motion controller. They are not synchronized with movement, and they will break blending.

`M62 P0` Would turn `digital0` pin on.
`M63 P0` Would turn `digital0` pin off.
Like all output pins, you can set the active state with the `:high` or `:low` attribute.
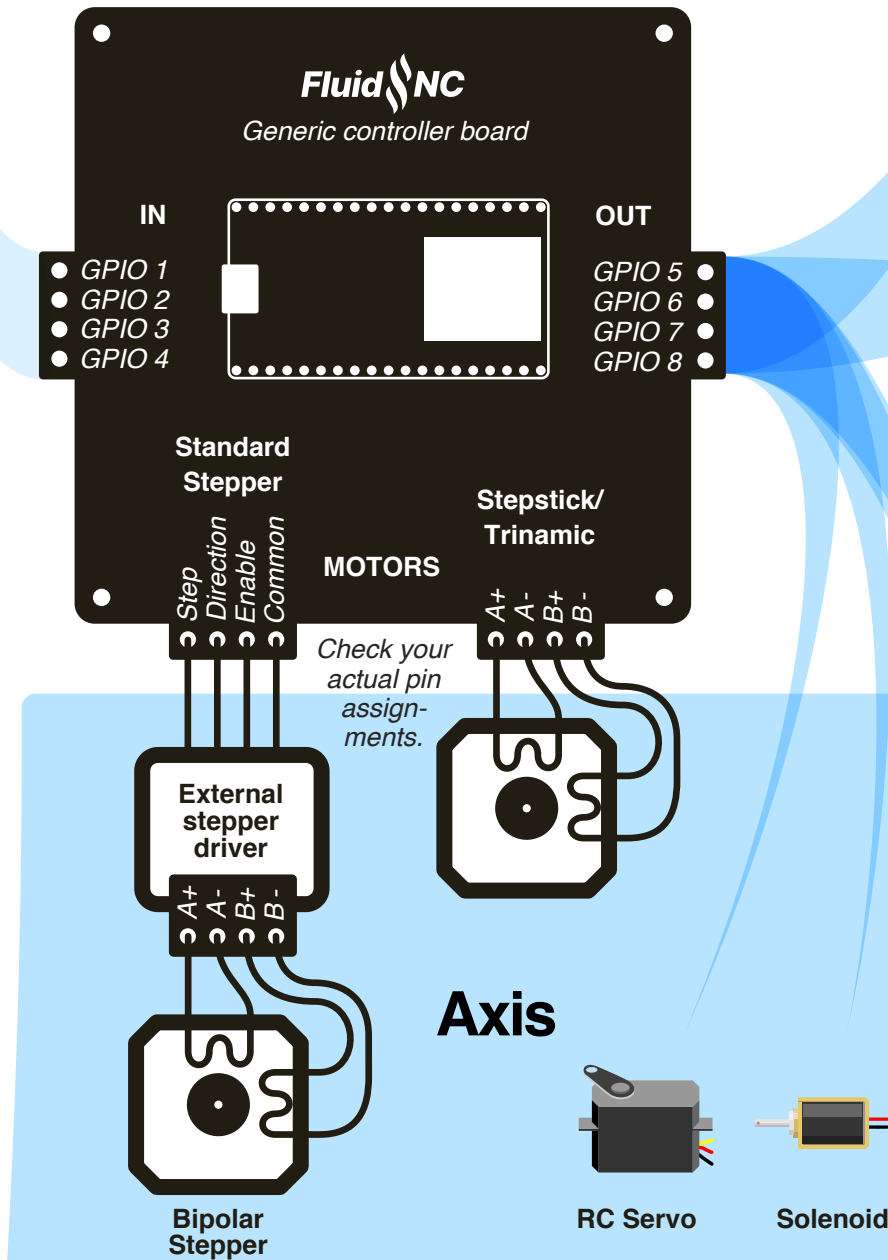
Coolant (digital)

`M7` Mist on
`M8` Flood on
`M9` Mist and flood both `off`

These outputs are traditionally called mist and flood, but many people use them for other things, like dust extraction, etc.

Serial over Telnet

⊚ *WIFI/Bluetooth*

If you have `$Telnet/Enable=True`, you can communicate via telnet with the same protocol as serial. The default port is 23 and set by `$Telnet/Port`. If enabled, you should see it in your startup messages:
`[MSG:INFO: Telnet Started on port 23]`

Open Telnet session in Terminal
`telnet <hostname>.local`

Websockets

⊚ *WIFI/Bluetooth*

if you connect via a websocket (on $http/port +1, e.g. 81), you have a streaming connection to FluidNC that behaves just like serial. You send newline-delimited lines just like you would over serial, and get back the same ok or error responses. Flow control is the same as for serial, as documented on the plain old Grbl wiki.

XModem file upload

⊟ *USB*

You can use xmodem via serial port to upload files to the localfs. This can be helpful to upload config files. you must be in the Idle or Alarm states.

Send to FluidNC
`$Xmodem/Receive=<filename>`

Download from FluidNC
`$XModem/Send=<filename>`

curl file upload

⊚ *WIFI/Bluetooth*

Upload file test.nc to the SD card at address 192.168.1.31
`curl -F upload=@test.nc http://192.168.1.31/upload`

Upload file test.nc to the ESP32 localfs (flash) at address 192.168.1.31
`curl -F file=@test.nc http://192.168.1.31/files`

Spindles

FluidNC supports multiple spindles on one machine. Spindles can be controlled by different hardware interfaces like relays, PWM, DACs, or RS485 serial interfaces to VFDs. Lasers are treated as spindles.

FluidNC I/O Cheatsheet

WebUI in Browser

⊚ *WIFI/Bluetooth*

The WebUI is the web browser based user interface. The WebUI is stored as a file, index.html.gz, on the local file system.
It allows for machine control and configuration and can be used to upload and run Gcode files. It has a tablet mode.

Set hostname via fluidterm
`$Hostname=<hostname>`

Open WebUI in browser
`<hostname>.local`

Link Colors

- FluidNC Wiki
- LinuxCNC Docs
- other

General Documentation

- ☐ G-Codes
- ☐ M-Codes
- ☐ Supported G-Codes
- ☐ ESP32 Pin reference

Useful links

- ☐ Set up FluidNC using macOS

Most texts and examples derived from the FluidNC Wiki
M-Code examples derived from the LinuxCNC Docs