

# EBB 2209 with Octopus Board Setup Guide

I want to start this off with a disclaimer, this is a compilation of information found throughout 4 other guides and a bit of trial and error on my part. If you use this guide, you are doing so at your own risk. This is what worked for me and a couple of others I have helped. If there are any differences in builds this information may need to be altered for your build.

Please read over everything before starting and make sure to follow the steps carefully!

The first thing you will need to do is install Mainsail on your Raspberry Pi.

Here is a link to the official Voron Documentation for installing Mainsail:

[https://docs.vorondesign.com/build/software/installing\\_mainsail.html](https://docs.vorondesign.com/build/software/installing_mainsail.html)

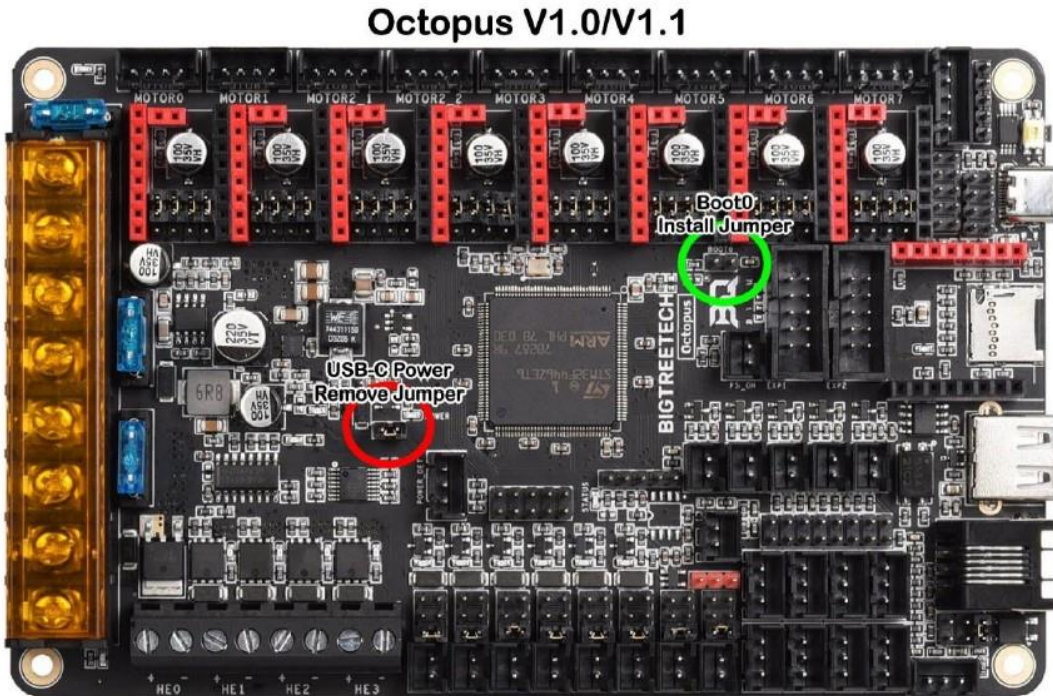
Once this is completed, you should now have a working install of mainsail with all your updates. We are ready to start building and flashing firmware. This will all be done through Putty or another SSH terminal.

Please keep in mind many linux commands are case sensitive, usually they are all lower case. If a command fails to run probably saying cannot find command, make sure it is lowercase.

These instructions will be using DFU mode for all flashing.

1. You will need to be able to install things from the web, therefore you will need to install the git command to the Pi. In many cases it is already there, but we will run this to be sure it is there.  
`sudo apt-get install git -y`
2. You will want to have canboot installed on your pi. Run the following command and allow it to finish running before doing anything else.  
`git clone https://github.com/Arksine/CanBoot`
3. Power everything down.
4. Make sure your MCU is connected to your Pi via USB at the Pi to USB-C at the octopus board (make sure this cable is a data cable, not just power). If you are powering your Pi from the octopus board, you will also need to have a cable connected USB at the octopus to USB-C at the Pi for power.

5. Remove the jumper from the center of the board (circled in red), if it is present, and move it to the Boot0 Install/DFU mode jumper (circled in green) near the reset button as shown in the image below.



7. With your jumper moved, power the MCU and Pi back up.
8. Firmly press the Reset button on the side of the octopus board near the USB-C port.



Your board should be in DFU mode now.

9. Log back into Putty or what ever SSH terminal you are using.

10. Enter:

```
cd ~/klipper/
```

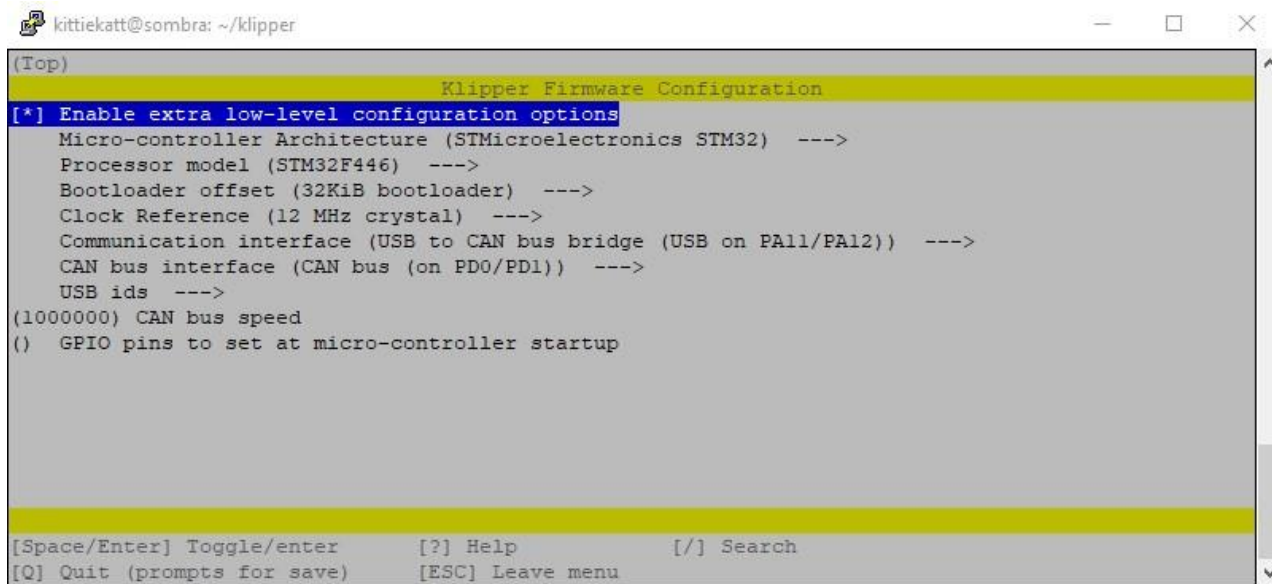
11. Now enter:

```
make menuconfig
```

You should now have a window with a series of settings. You will want to adjust your settings to match. (Again this is the settings for octopus to ebb 2209, if you are using different boards these configuration settings may be different). Make sure you verify the process model of your MCU board and use the proper values or you will get magic smoke!!!! The example is specifically for an octopus board with a STM32F446 processor!!

\*note some prefer to run their canbus speed lower to avoid possible interference. I have had no issues at 1m for this value. If you wish to lower this you may just make sure you use the same value in all 3 locations (octopus firmware compiling, ebb 2209 firmware compiling, and canbus network file) or you will get errors! Most common values used are 1m and 500k.

Write your canbus speed down now, this way you can reference this later!!!!!!!!

A screenshot of a terminal window titled 'kittiekatt@sombra: ~/klipper'. The terminal displays the 'Klipper Firmware Configuration' menu. The menu is a list of configuration options, each followed by a right-pointing arrow. The first option, '[\*] Enable extra low-level configuration options', is highlighted in blue. Below it are options for 'Micro-controller Architecture (STMicroelectronics STM32)', 'Processor model (STM32F446)', 'Bootloader offset (32KiB bootloader)', 'Clock Reference (12 MHz crystal)', 'Communication interface (USB to CAN bus bridge (USB on PA11/PA12))', 'CAN bus interface (CAN bus (on PD0/PD1))', 'USB ids', '(1000000) CAN bus speed', and '() GPIO pins to set at micro-controller startup'. At the bottom of the terminal, there is a footer with navigation instructions: '[Space/Enter] Toggle/enter', '[?] Help', '[/] Search', '[Q] Quit (prompts for save)', and '[ESC] Leave menu'.

```
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
Micro-controller Architecture (STMicroelectronics STM32) --->
Processor model (STM32F446) --->
Bootloader offset (32KiB bootloader) --->
Clock Reference (12 MHz crystal) --->
Communication interface (USB to CAN bus bridge (USB on PA11/PA12)) --->
CAN bus interface (CAN bus (on PD0/PD1)) --->
USB ids --->
(1000000) CAN bus speed
() GPIO pins to set at micro-controller startup

[Space/Enter] Toggle/enter    [?] Help    [/] Search
[Q] Quit (prompts for save)  [ESC] Leave menu
```

12. Once all your value are set properly, you can press “Q” to exit. Then select “Yes” when asked to save the configuration.

13. To compile the firmware we want to run the command:

```
make
```

14. Once this is done we will enter:

```
lsusb
```

This should give you a listing of all usb devices on your machine, and look something like this.

```
pi@fluidpi:~$ lsusb
Bus 001 Device 005: ID 0483:df11 STHicroolactronics STM Device in DFU Mode
Bus 001 Device 004: ID 1d50:6061 OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 001 Device 003: ID 0424:0c00 Microchip Technology, Inc. ( formerly SMSC ) SMC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Microchip Technology, Inc. ( formerly SMSC ) SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@fluidpi:~$
```

You are looking for the device listed as DFU Mode and will need the address to the left of the text name.

15. You will enter the following command using the address of your device if it is different from the example.  
make flash FLASH\_DEVICE=0483:df11

You should get text saying it is doing things. Most of the way down through this should be a prompt saying “File Downloaded successfully”. There will more than likely be an error message after that, dfu-util:error during download get status. As long as it says downloaded successfully, it should have worked and you should be fine.

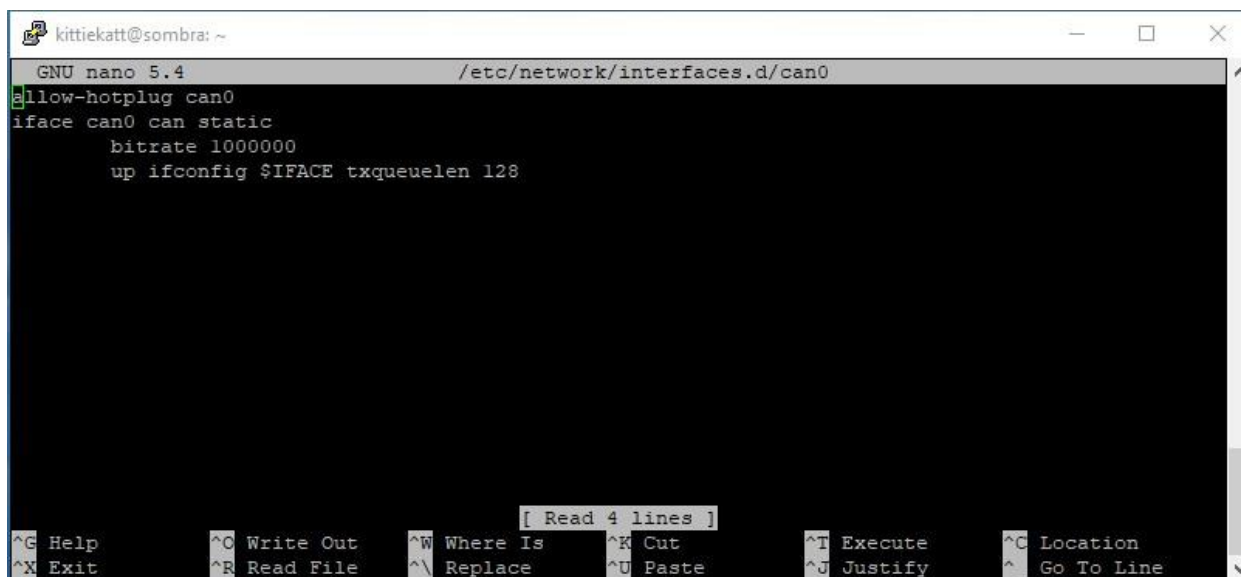
Now that our firmware should be written we need create the can network and add the MCU.

16. sudo nano /etc/network/interfaces.d/can0

This will open up a text editor. You will want to copy & paste the following text into this window:

```
allow-hotplug can0
iface can0 can static
    bitrate 1000000
    up ifconfig $IFACE txqueuelen 128
```

It should look like this.



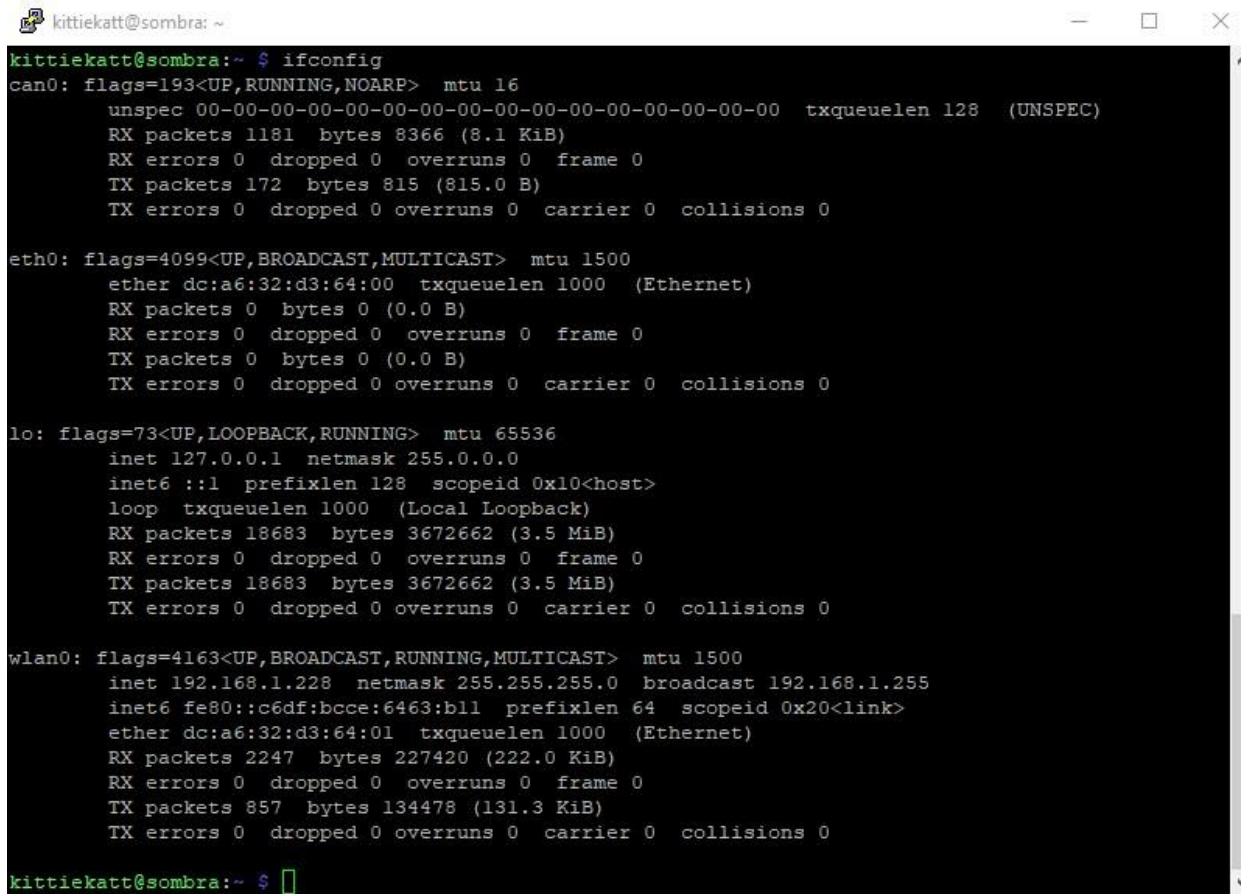
```
kittiekatt@sombra: ~
GNU nano 5.4 /etc/network/interfaces.d/can0
allow-hotplug can0
iface can0 can static
    bitrate 1000000
    up ifconfig $IFACE txqueuelen 128
[ Read 4 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

Note: if you are using 500k or another value for your bitrate/canbus speed in your firmware flashing make sure it matches here as well!

17. Once this all looks correct, you will: Ctrl + X to exit. Key “Y” for yes. And Hit Enter. This will save and exit.

18. Once this is created we want to make sure it took. Run the following command to see your networks:  
ifconfig

You should get a listing of all your networks on the Pi. You are looking for the Can0 network we created.



```
kittiekatt@sombra: ~  
kittiekatt@sombra:~ $ ifconfig  
can0: flags=193<UP,RUNNING,NOARP> mtu 16  
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 128  (UNSPEC)  
    RX packets 1181  bytes 8366 (8.1 KiB)  
    RX errors 0  dropped 0  overruns 0  frame 0  
    TX packets 172  bytes 815 (815.0 B)  
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0  
  
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    ether dc:a6:32:d3:64:00 txqueuelen 1000  (Ethernet)  
    RX packets 0  bytes 0 (0.0 B)  
    RX errors 0  dropped 0  overruns 0  frame 0  
    TX packets 0  bytes 0 (0.0 B)  
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000  (Local Loopback)  
    RX packets 18683  bytes 3672662 (3.5 MiB)  
    RX errors 0  dropped 0  overruns 0  frame 0  
    TX packets 18683  bytes 3672662 (3.5 MiB)  
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0  
  
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.1.228 netmask 255.255.255.0 broadcast 192.168.1.255  
    inet6 fe80::c6df:bcce:6463:b11 prefixlen 64 scopeid 0x20<link>  
    ether dc:a6:32:d3:64:01 txqueuelen 1000  (Ethernet)  
    RX packets 2247  bytes 227420 (222.0 KiB)  
    RX errors 0  dropped 0  overruns 0  frame 0  
    TX packets 857  bytes 134478 (131.3 KiB)  
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0  
  
kittiekatt@sombra:~ $
```

If you do not see a can0 network, go back to step 14 and open your can network file we made and ensure everything matches the screen shot exactly. If everything does match restart you pi and do another ifconfig.

Once you see your can network, we can now start with adding our MCU to this network and your printer config file.

19. cd ~/CanBoot/scripts

20. python3 flash\_can.py -i can0 -q

If these scripts fail go back and complete step 1 of the manual to install CanBoot and try steps 17 and 18 again.

You should be able to see your MCU in the list given (it should be the only one device, unfortunately I forgot to screen shot this during my build so had to use example that shows 2 results). You will want to copy and paste that UID somewhere safe, like a text document on your computer, as we will use this again and once added it will not show here



again. It is also a good idea to keep your UID list around until you have had everything running for a bit and have a printer config backed up to your pc.

```
pi@klipper:~/CanBoot/scripts $ python3 flash_can.py -i can0 -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 127081e7e3c6, Application: CanBoot
Detected UUID: 463b35222d7b, Application: Klipper
Query Complete
pi@klipper:~/CanBoot/scripts $
```

21. Open your mainsail in your browser.
22. Go to the Machine section.
23. Ensure you looking at "Config Files" if not select "config" in the "Root" box.
24. Click "Printer.cfg" to open it.
25. Near the top you should have a section for mcu this will be shown as [mcu]. Just under that line you will put your id as follows:  
Canbus\_uuid: (UID you copied earlier)

It should look something like this:

```
[mcu]
canbus_uuid: 5b465cc7b0ef
```

26. You can now click "Save and Restart" in the top right corner.

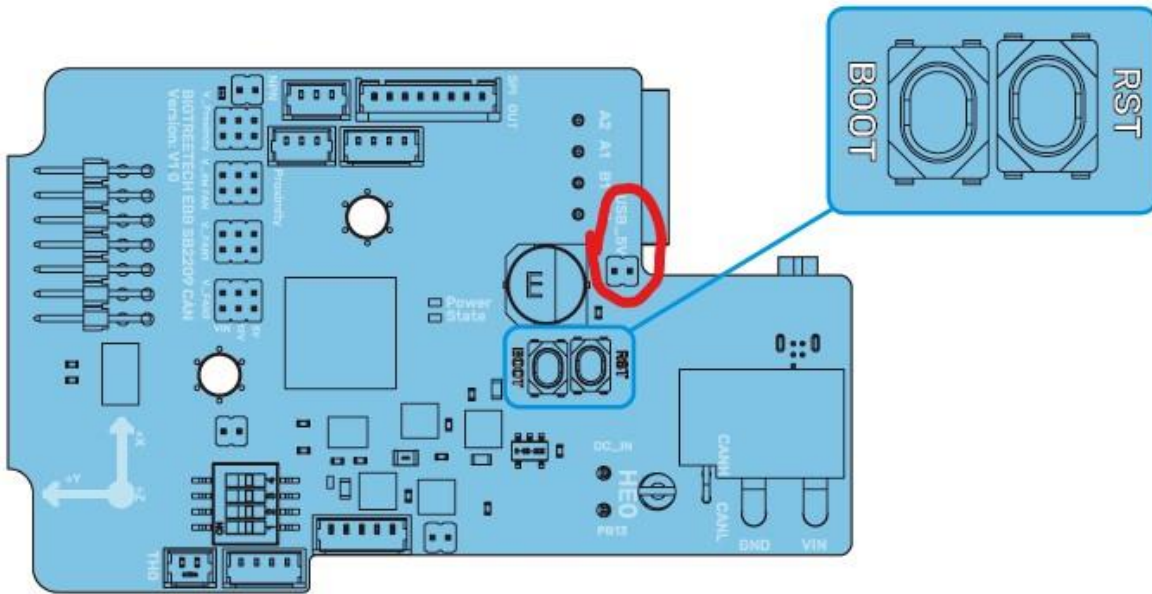
Your MCU should now be in your can network and seen by mainsail.

27. Once you have verified your MCS is showing in mainsail (you will still have errors as our ebb 2209 has not been added yet) we can shut everything down.
28. Remove your jumper from the MCU board we put on the boot jumper, if you had this on USB power jumper you can put this back on there at this time.

We are now going to start working on the EBB 2209. Make sure you have a clear area near your pi where you can set the board and nothing can short it out.

29. Make sure all jumpers are removed from the ebb 2209.

30. Place one jumper on the USB\_5v jumper.



31. Connect the the ebb 2209 to the raspberry pi via USB-C at the 2209 and USB at the pi. (Make sure this is a data cable as we are sending data over this connection!)

32. Power everything up.

33. Once the pi is done booting, press and hold the boot button. While holding, press and release the RST button. After that is released, you can release the boot button. This should put your board in DFU mode.

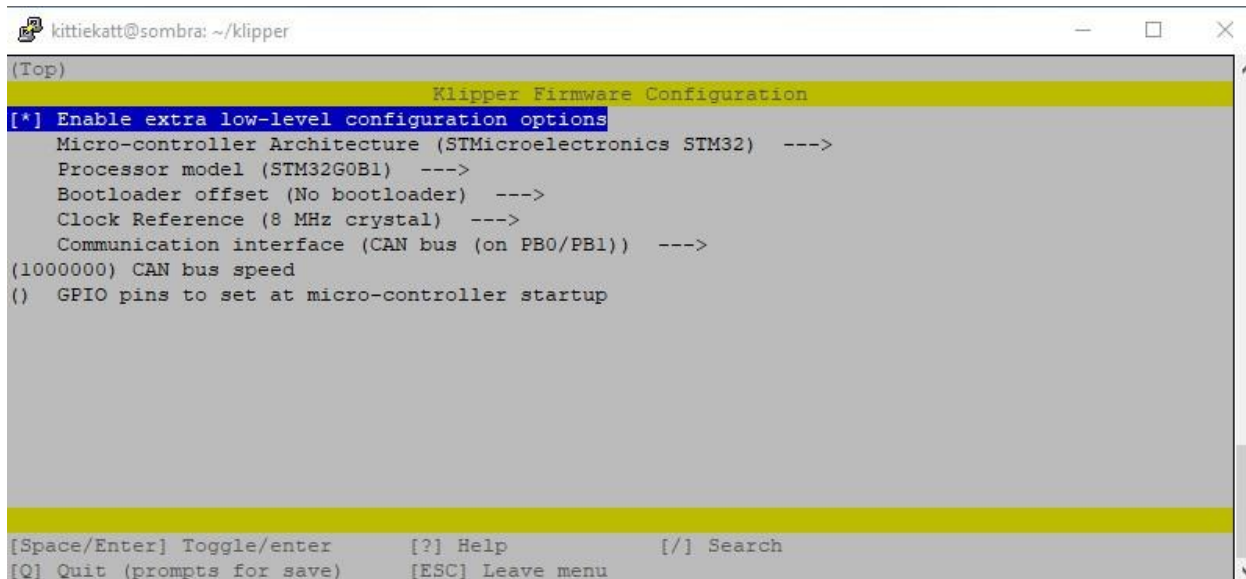
34. On your computer we will log in via Putty or what ever SSH tool you are using.

35. Enter:  
`cd ~/klipper/`

36. Now enter:  
`make menuconfig`

You should now have a window with a series of settings. You will want to adjust your settings to match. (Again this is the settings for octopus to ebb 2209, if you are using different boards these configuration settings may be different)

\*note make sure you match your canbus speeds to what you used in prior steps.



```
kittiekatt@sombra: ~/klipper
(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
  Micro-controller Architecture (STMicroelectronics STM32) --->
  Processor model (STM32G0B1) --->
  Bootloader offset (No bootloader) --->
  Clock Reference (8 MHz crystal) --->
  Communication interface (CAN bus (on PB0/PB1)) --->
(1000000) CAN bus speed
() GPIO pins to set at micro-controller startup

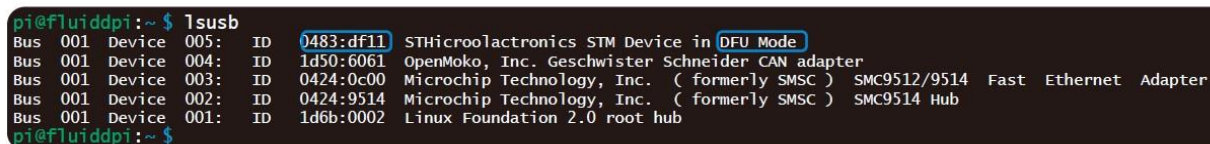
[Space/Enter] Toggle/enter    [?] Help    [/] Search
[Q] Quit (prompts for save)  [ESC] Leave menu
```

37. Once all your value are set properly, you can press “Q” to exit. Then select “Yes” when asked to save the configuration.

38. To compile the firmware we want to run the command:  
make

39. Once this is done we will enter:  
lsusb

This should give you a listing of all usb devices on your machine, and look something like this.



```
pi@fluidpi:~$ lsusb
Bus 001 Device 005: ID 0483:df11 STHicroolactronics STM Device in DFU Mode
Bus 001 Device 004: ID 1d50:6061 OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 001 Device 003: ID 0424:0c00 Microchip Technology, Inc. ( formerly SMSC ) SMC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Microchip Technology, Inc. ( formerly SMSC ) SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@fluidpi:~$
```

You are looking for the device listed as DFU Mode and will need the address to the left of the text name.

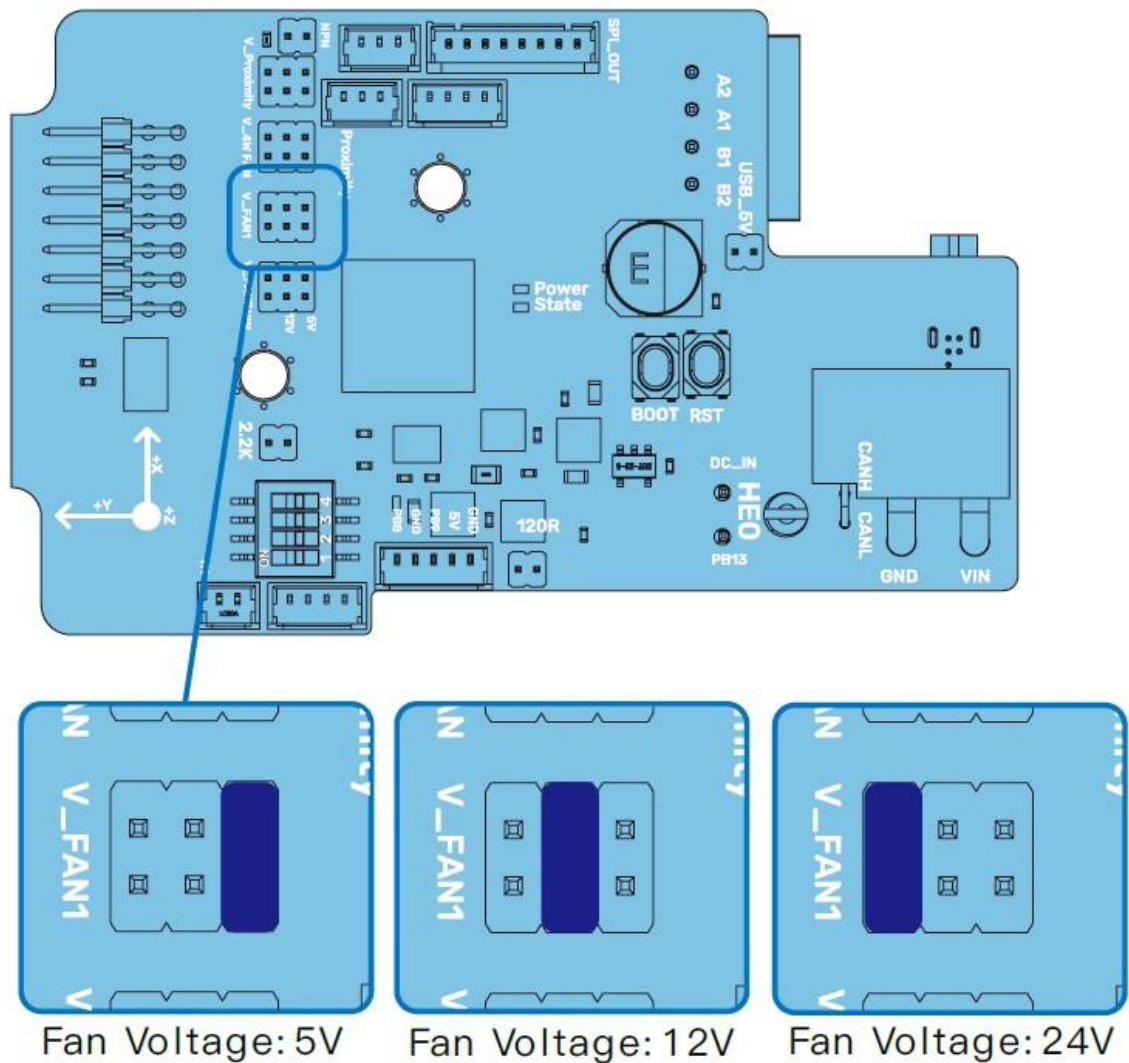
40. You will enter the following command using the address of your device if it is different from the example.  
make flash FLASH\_DEVICE=0483:df11

You should get text saying it is doing things. Most of the way down through this should be a prompt saying “File Downloaded successfully”. There will more than likely be an error message after that, dfu-util:error during download get status. As long as it says downloaded successfully, it should have worked and you should be fine.

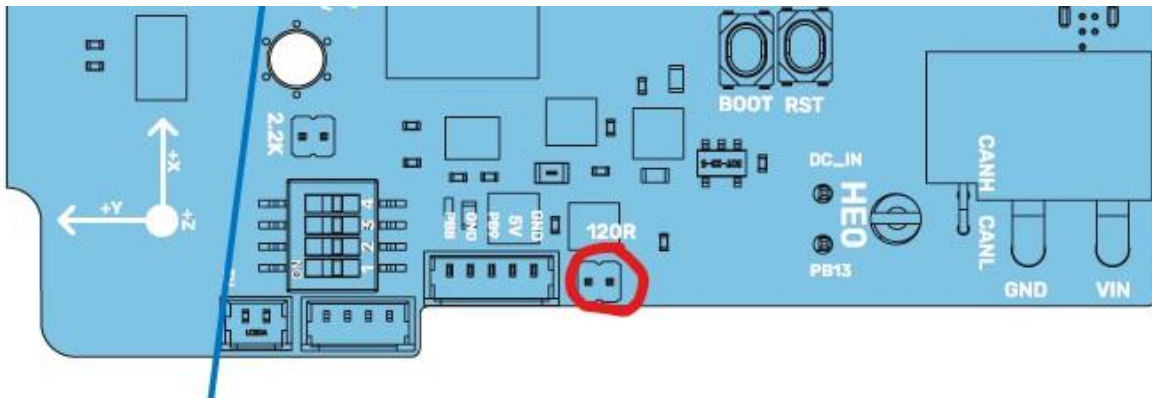
Now that our firmware should be written we need to add it to the network.



41. Shut everything down again.
42. Remove the USB connections from the pi and ebb 2209 board.
43. Remove the jumper from the USB\_5v jumper.
44. You will now add jumpers to your fan voltage selections. Standard is 24v but be sure to check the voltage of the fans you are using! You will want to set voltage pins on all 3 fan jumpers as well as the Proximity if you are using that.



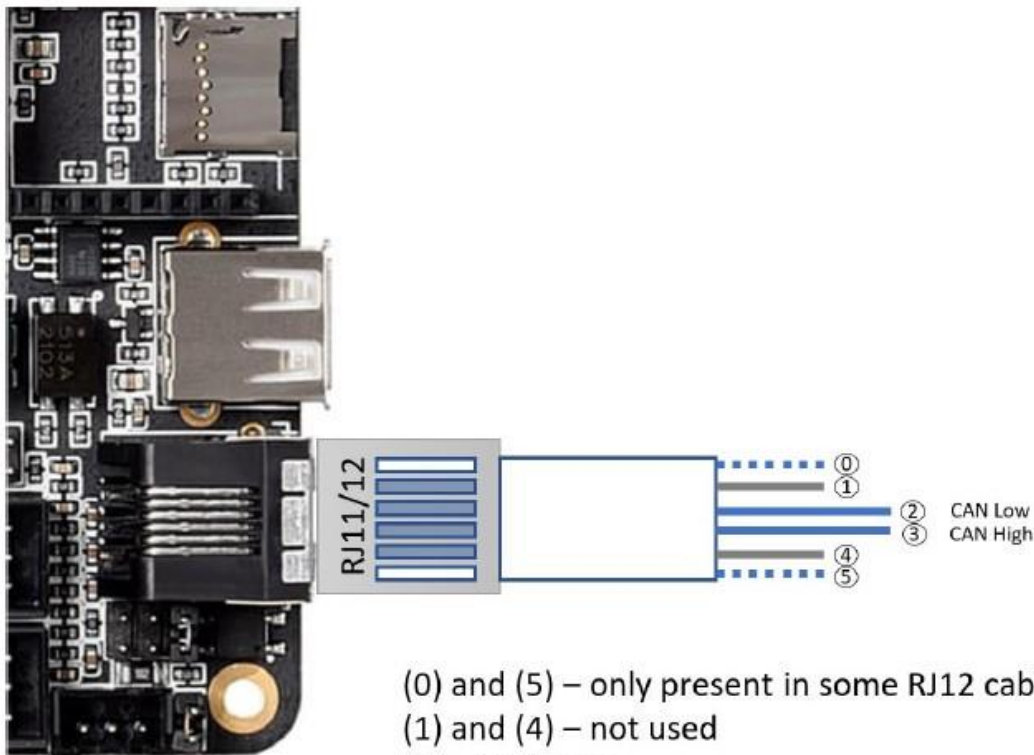
45. You will also need to add a jumper to the 120R (120 ohms resistance) jumper. This is at the bottom of the board centered.



46. Next we will connect the Canbus wiring to the EBB 2209 board and run this as desired.

47. To connect this to our Octopus board we are going to use the rj11 phone jack port. You can either crimp a rj11 connector or use a screw terminal version. (Since I have network patch cable making tools I prefer a crimped connection as it takes up far less space and is more secure.)

If you are not sure which wire is your high and your low check the user manual for your board. At the time of this manual when using the EBB 2209, the can low is green and can high is yellow.



(0) and (5) – only present in some RJ12 cables, not used  
(1) and (4) – not used  
(2) – CAN Low  
(3) – CAN High

48. Our power will be directly wired to the power supply. Positive to positive, negative to negative.

We are not going to connect the face to the print head or add any other wiring at this time. We want nothing else connected while we ensure we can add this to the canbus network.

49. Once we have our canbus cable connected to the power supply and octopus board as well as the EBB 2209, we are ready to power everything back up.

If everything was wired correctly and jumpers in place there should be no magic smoke.

50. Log into putty or what ever SSH terminal you are using.

51. `cd CanBoot/scripts`

52. `python3 flash_can.py -i can0 -q`

You should be able to see your EBB 2209 in the list given (it should be the only one device, unfortunately I forgot to screen shot this during my build so had to use example that shows 2 results). You will want to copy and paste that UID somewhere safe as we will use this again and once added it will not show here again.

```
pi@klipper:~/CanBoot/scripts $ python3 flash_can.py -i can0 -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 127081e7e3c6, Application: CanBoot
Detected UUID: 463b35222d7b, Application: Klipper
Query Complete
pi@klipper:~/CanBoot/scripts $
```

53. Open your mainsail in your browser.

54. Go to the Machine section.

55. Ensure you looking at "Config Files" if not select "config" in the "Root" box.

56. Click "Printer.cfg" to open it.

57. Near the top you should have a section for the EBB 2209 this will be shown as [mcu EBBcan]. Just under that line you will put your id as follows:  
Canbus\_uuid: (UID you copied earlier)

It should look something like this:

```
[mcu EBBCan]  
canbus_uuid: 9093dd0ffa56
```

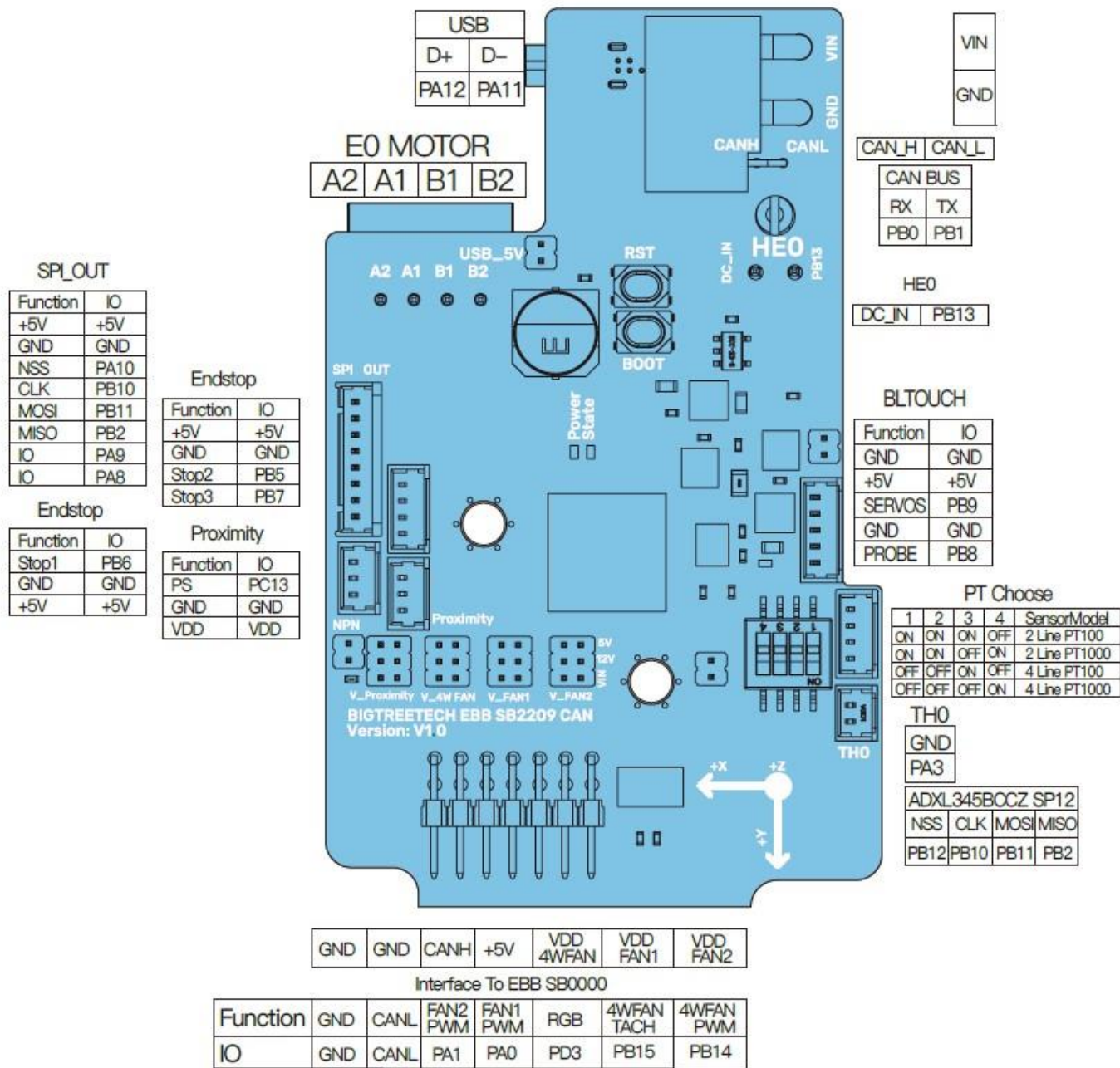
58. You can now click “Save and Restart” in the top right corner.

Your EBB 2209 should now be in your can network and seen by mainsail. You will still have errors as the tool head thermister and such have not been connected.

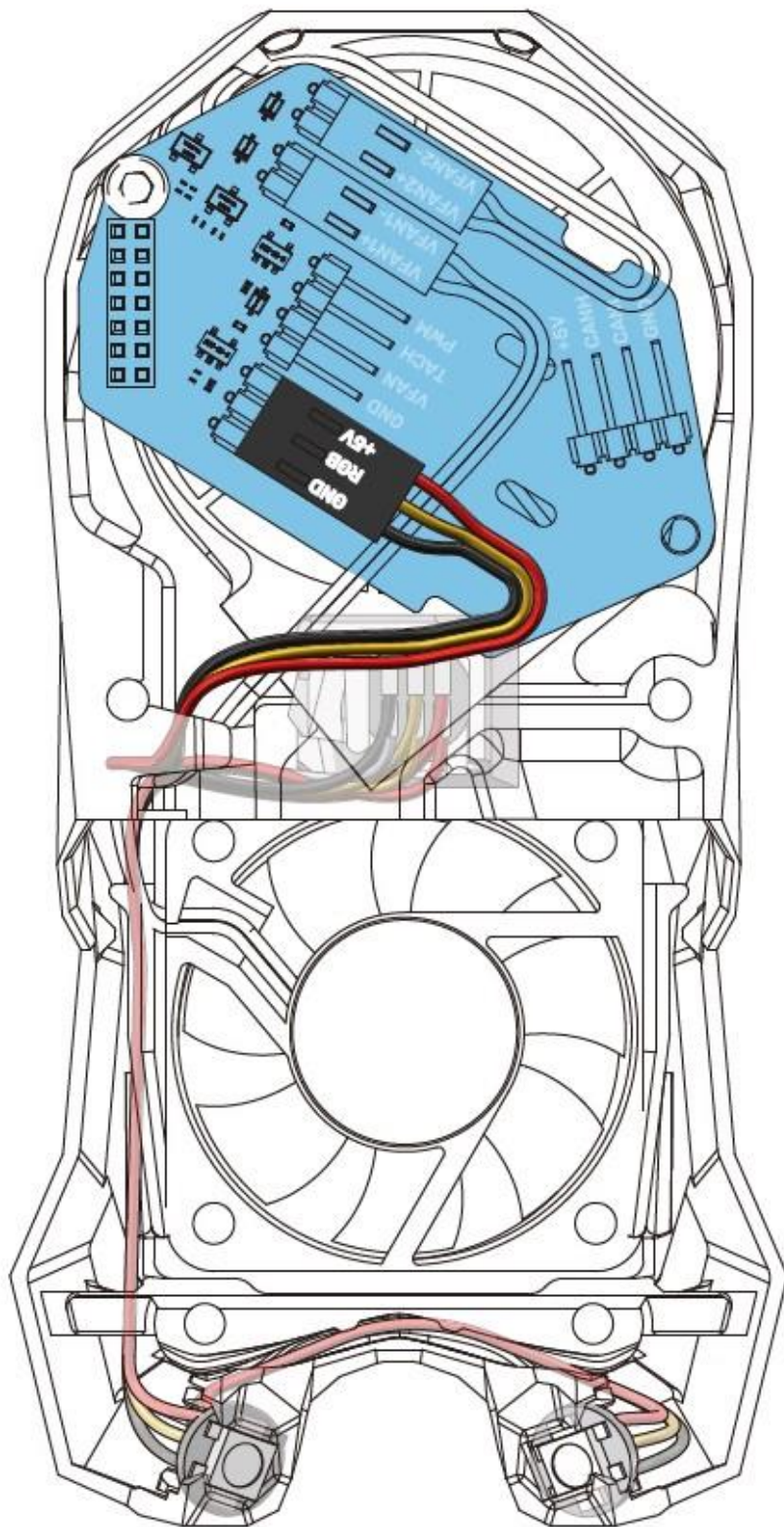
59. Once you have verified this is in place you can shut everything down again.

60. You can now wire your toolhead as needed for your needs.

Here is a pinout of the main board as well as the smaller board for your toolhead face (stealth burner example).







61. Once you have everything wired and the tool head face in place (be VERY CAREFUL to get the pins lined up properly or you can fry your boards!!)
62. Once you have double checked all your wiring and ensured everything is set correctly you can turn everything back on.

63. You will probably still have errors and need to update your pins in your printer config. The EBB 2209 also has built in input shaping, so we can just add the code for that in our printer config as well.

```
#####  
##  Input Shaping  
#####
```

```
[adxl345]  
cs_pin: EBBCan: PB12  
spi_software_sclk_pin: EBBCan: PB10  
spi_software_mosi_pin: EBBCan: PB11  
spi_software_miso_pin: EBBCan: PB2  
axes_map: x,y,z
```

```
[resonance_tester]  
accel_chip: adxl345  
probe_points:  
    150, 150, 10
```

```
#####  
##  Input Shaping  
#####  
  
[adxl345]  
cs_pin: EBBCan: PB12  
spi_software_sclk_pin: EBBCan: PB10  
spi_software_mosi_pin: EBBCan: PB11  
spi_software_miso_pin: EBBCan: PB2  
axes_map: x,y,z  
  
[resonance_tester]  
accel_chip: adxl345  
probe_points:  
    150, 150, 10  
  
[input_shaper]  
shaper_freq_x: 59.0  
shaper_type_x: zv  
shaper_freq_y: 41.0  
shaper_type_y: zv
```

The last section of input\_shaper will be based on your results when you run input shaping.

All pins for your EBB 2209 will have the same format of EBBCan: pin#. If you are unsure of the pin number, you can check the pinout image above as it has the pin numbers. You can also check the github as it has example printer config for using the EBB 2209.

<https://github.com/bigtreotech/EBB/commit/839b482818f9ad40cb180c485b1aa7aecf6e57de#diff-a1f9ab970f527271e4965aa8d0dad3a24606ad7d89c9164a163673b9ed5149f5>

64. Once you have all your pins updated you should have no errors left. If you are still getting errors trouble shoot these one by one correcting them until none remain.
65. At this point you are free to add klipper screen, crowsnest (cam controller if not already working), and other plug ins as needed.

After your printer has run about 20 hours or so it should be ok to run input shaping. This is a simple process.

1. Open mainsail.
2. In the console code input box enter:  
TEST\_RESONANCES AXIS=X  
Allow this to run until complete.
3. Enter:  
TEST\_RESONANCES AXIS=y  
Allow this to run until complete.
4. Open Putty or what ever SSH terminal you are using and enter:  
~/klipper/scripts/calibrate\_shaper.py /tmp/resonances\_x\_\*.csv -o /tmp/shaper\_calibrate\_x.png  
Allow this to run until complete.
5. Enter:  
~/klipper/scripts/calibrate\_shaper.py /tmp/resonances\_y\_\*.csv -o /tmp/shaper\_calibrate\_y.png  
Allow this to run until complete.

You should now have a screen that looks similar to this.

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep 16 08:09:45 2023 from 192.168.1.223
kittiekatt@sombra:~ $ ~/klipper/scripts/calibrate_shaper.py /tmp/resonances_x_*.
csv -o /tmp/shaper_calibrate_x.png
Fitted shaper 'zv' frequency = 59.0 Hz (vibrations = 1.0%, smoothing ~= 0.051)
To avoid too much smoothing with 'zv', suggested max_accel <= 13600 mm/sec^2
Fitted shaper 'mzv' frequency = 59.4 Hz (vibrations = 0.0%, smoothing ~= 0.058)
To avoid too much smoothing with 'mzv', suggested max_accel <= 10400 mm/sec^2
Fitted shaper 'ei' frequency = 71.0 Hz (vibrations = 0.0%, smoothing ~= 0.064)
To avoid too much smoothing with 'ei', suggested max_accel <= 9400 mm/sec^2
Fitted shaper '2hump_ei' frequency = 88.2 Hz (vibrations = 0.0%, smoothing ~= 0.
069)
To avoid too much smoothing with '2hump_ei', suggested max_accel <= 8600 mm/sec^
2
Fitted shaper '3hump_ei' frequency = 105.4 Hz (vibrations = 0.0%, smoothing ~= 0
.074)
To avoid too much smoothing with '3hump_ei', suggested max_accel <= 8100 mm/sec^
2
Recommended shaper is zv @ 59.0 Hz
kittiekatt@sombra:~ $ ~/klipper/scripts/calibrate_shaper.py /tmp/resonances_y_*.csv -o /tmp/shaper
_calibrate_y.png
Fitted shaper 'zv' frequency = 41.0 Hz (vibrations = 1.1%, smoothing ~= 0.096)
To avoid too much smoothing with 'zv', suggested max_accel <= 6600 mm/sec^2
Fitted shaper 'mzv' frequency = 41.0 Hz (vibrations = 0.0%, smoothing ~= 0.121)
To avoid too much smoothing with 'mzv', suggested max_accel <= 5000 mm/sec^2
Fitted shaper 'ei' frequency = 49.2 Hz (vibrations = 0.0%, smoothing ~= 0.133)
To avoid too much smoothing with 'ei', suggested max_accel <= 4500 mm/sec^2
Fitted shaper '2hump_ei' frequency = 61.4 Hz (vibrations = 0.0%, smoothing ~= 0.143)
To avoid too much smoothing with '2hump_ei', suggested max_accel <= 4200 mm/sec^2
Fitted shaper '3hump_ei' frequency = 74.0 Hz (vibrations = 0.0%, smoothing ~= 0.150)
To avoid too much smoothing with '3hump_ei', suggested max_accel <= 4000 mm/sec^2
Recommended shaper is zv @ 41.0 Hz
kittiekatt@sombra:~ $ █
```

You will want to look at the bottom lines for each section. It tells you the recommended shaper to use. In this case it is ZV at 59.0 Hz for the X axis. So we will want to look at the max accel for that result above. This being 13600 mm/sec. We will do this again for our Y axis. (Your Y axis commonly has a lower accel result.) So for this we will be doing ZV at 41.0 Hz with a max accel of 6600 mm/sec.

There are two ways to approach these results. Some put in the very max of the lower number into their printer config. Personally I entered mine a little under, so in this case I used 6000 as my max accel.

6. You can input your results into the [input\_shaper] (shaper and frequency) and [printer] (accel) sections of your printer config.

```

[printer]
kinematics: corexy
max_velocity: 350
max_accel: 6000          #Max 4000
max_z_velocity: 50       #Max 15 for 12V TMC Drivers, can increase for 24V
max_z_accel: 350
square_corner_velocity: 10.0

#####
##   Input Shaping
#####

[adxl345]
cs_pin: EBBCan: PB12
spi_software_sclk_pin: EBBCan: PB10
spi_software_mosi_pin: EBBCan: PB11
spi_software_miso_pin: EBBCan: PB2
axes_map: x,y,z

[resonance_tester]
accel_chip: adxl345
probe_points:
    150, 150, 10

[input_shaper]
shaper_freq_x: 59.0
shaper_type_x: zv
shaper_freq_y: 41.0
shaper_type_y: zv

```

7. Save and Restart.



Helpful Links while building your printer:

<https://www.klipper3d.org/CANBUS.html>

<https://github.com/bigtreetech/docs/blob/master/docs/EBB%202240%202209%20CAN.md>

Helpful links once your printer is working:

<https://ellis3dp.com/Print-Tuning-Guide/>

<https://www.simplify3d.com/resources/print-quality-troubleshooting/>