

BTT MANTA M8P-CB1 - EBB SB2209 COMBO CANBUS SETUP AND TESTING GUIDE

Rev 1.2.0

BY westy_pity_da_fool September 2023

Hardware needed:

BTT Manta M8P mainboard

BTT CB1 – Raspberry Pi replacement clip in board – both compact and saves the need for an extra 5V PSU. Runs mainsail, klipper, moonraker etc. Needs to be flashed with Linux package.

BTT EBB SB 2209 Canbus master control board – Controller for the tool head controls voltage to the tool head cans etc.

BTT SB0000 CAN – CanBus slave pinout board. (no programming needed)

Class 10 Min 16GB SD micro card

SC card to USB reader/adaptor – to allow all the software to be written from the PC and install in the M8P Manta soc .

2 x USB to C cables (These are only needed for pre installation configuration stage only – not for installation in the printer)

1 X CANBUS CABLE - during the config stage, we are only using the Can L and Can H cables to transfer data.

PRE ASSEMBLY

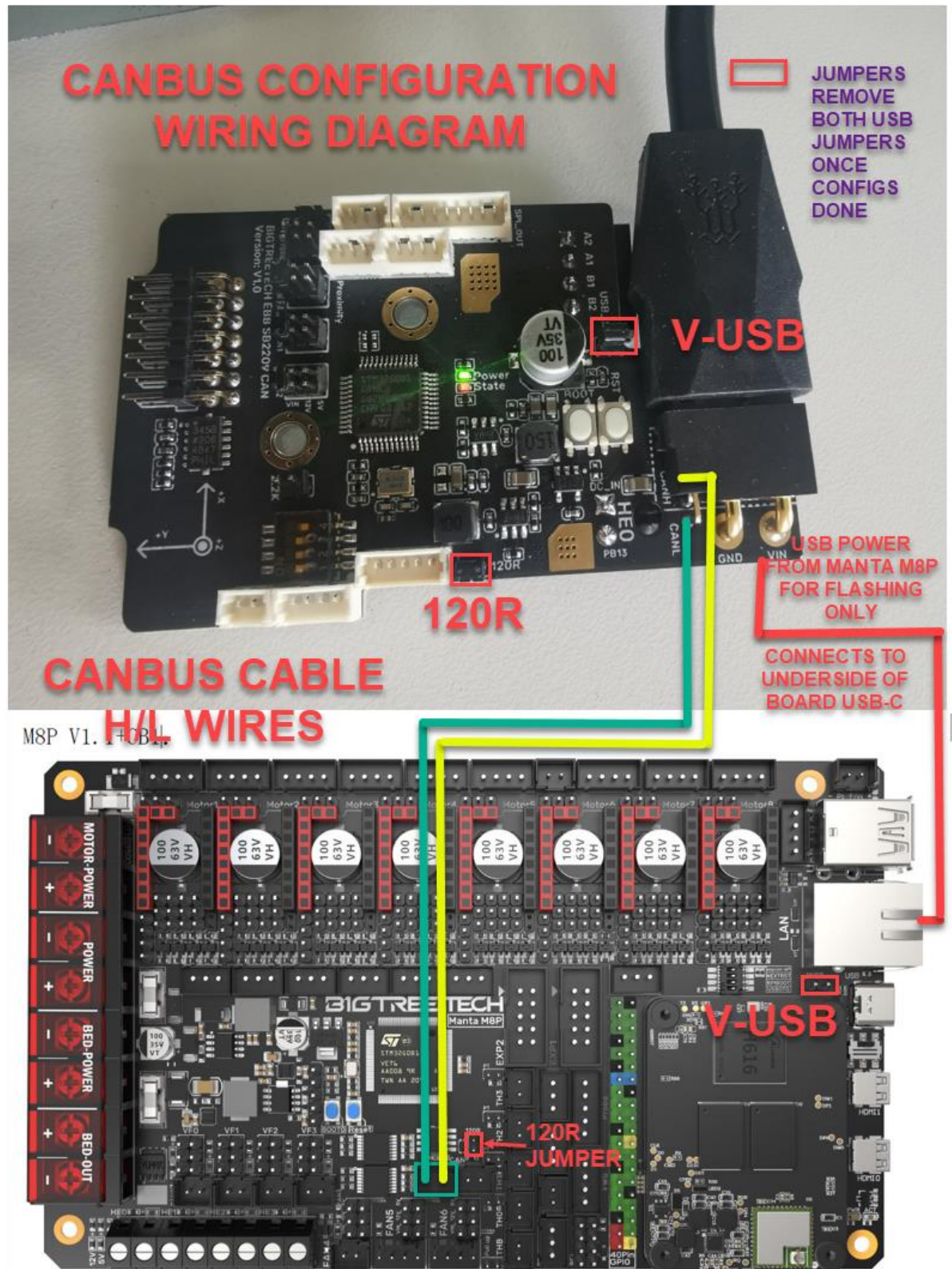
Install 4 x jumpers (SEE PIC BELOW FOR WHERE THESE ARE)

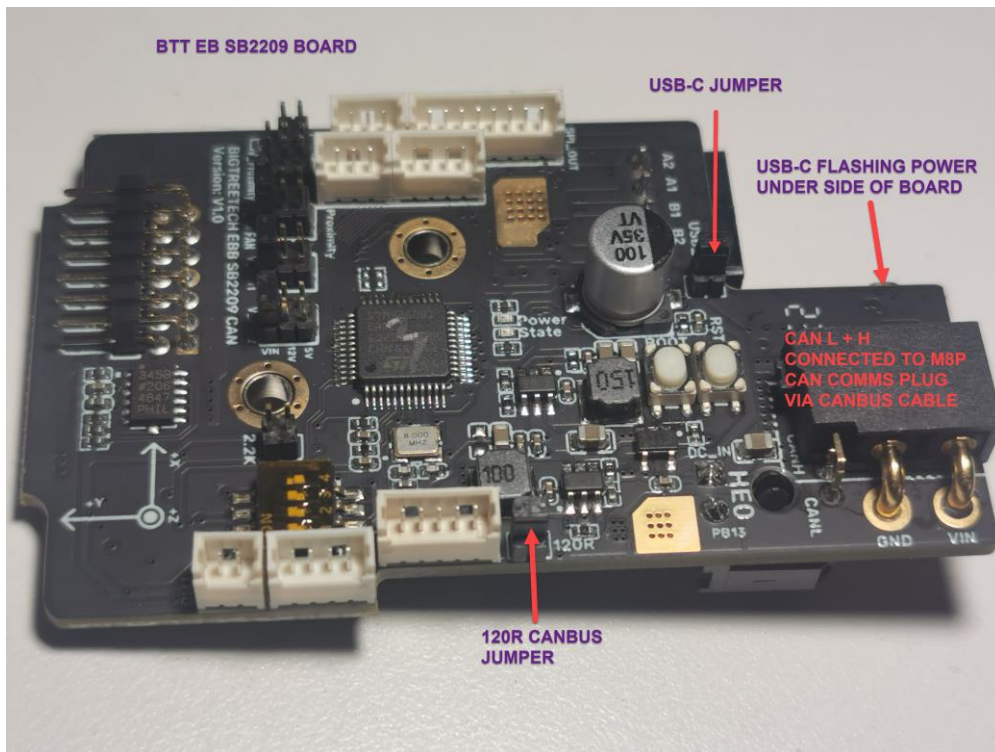
- **MANTA M8P & EBB SB2209** INSTALL JUMBER ON USB 5V
Note: TEMPORARILY WHILE PROGRAMMING ONLY. – BOTH GET REMOVED ONCE CANBUS AND FLASHING IS COMPILED, CONFIGURED AND TESTED.

These USB jumpers allow the boards to be powered up VIA the USB ports. The main USB is from a computer or power supply 5V USB and connects to the BTT MANTA M8P MAIN USB C PORT.

The second is when we power up the BTT EBB SB2209 board later on, which comes from the USB port on your Manta M9P and will plug into the EBB SB2209 USB-C connection socket under the board.

- **MANTA M8P & EBB SB2209** Jumper on 120R Canbus resistor jumper. These stay installed permanently.





PRE INSTALLATION SOFTWARE TOOLS NEEDED:

- **MOBAXTERM** <https://mobaxterm.mobatek.net/download-home-edition.html>
This is what we will be logging into the hardware to configure and compile each device using SSH
- **ADVANCED IP SCANNER** <https://www.advanced-ip-scanner.com/>
This is used to scan your network to find your device once it is online
- **BALENA ETCHER** <https://etcher.balena.io/>
This is needed to create imaged drive for your CB1 CPU device (replaces raspberry Pi and 5V PSU)

Firmware packages

CB1 – Linux package latest <https://github.com/bigtreetech/CB1/releases>

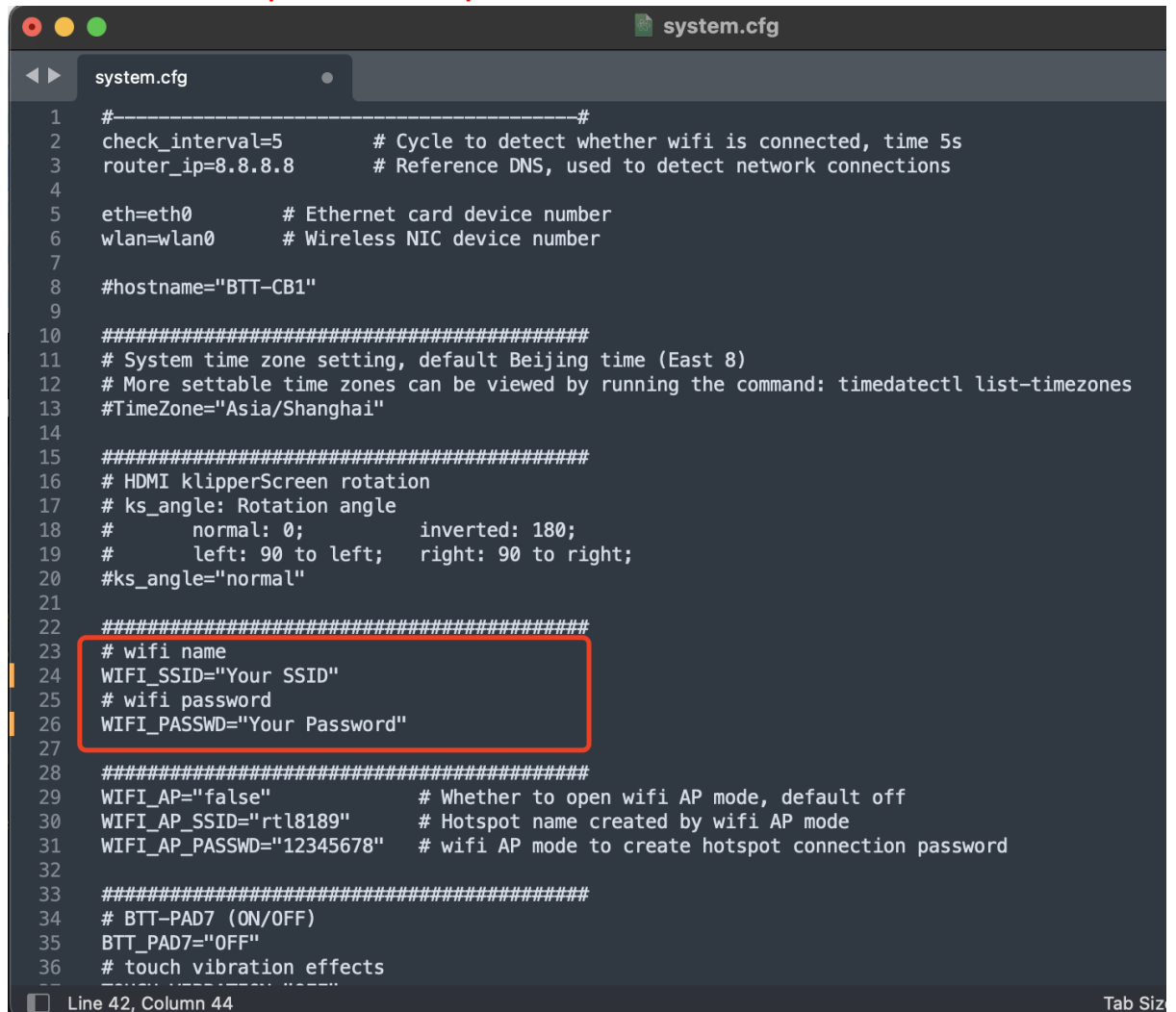
SOFTWARE INSTALLATION – Install Klipper to CB1 (Raspberry Pi replacement)

1. Download (Links above) Install Balena Etcher, Advanced IP scanner and MobaXterm onto your PC
2. Download the CB1 Linux package specially designed for the CB1 from the Github link
<https://github.com/bigtreetech/CB1/releases>
3. Open balena etcher and flash the CB1 debian package onto your SD card.

Follow this link to see how (insert link here later)

4. Remove SD reader and re insert it. This will show you a new SUB drive called "Boot"
5. Go into the boot drive file system and open the system.cfg file.
6. Change the Wireless section to match your SSID and Password on your network.

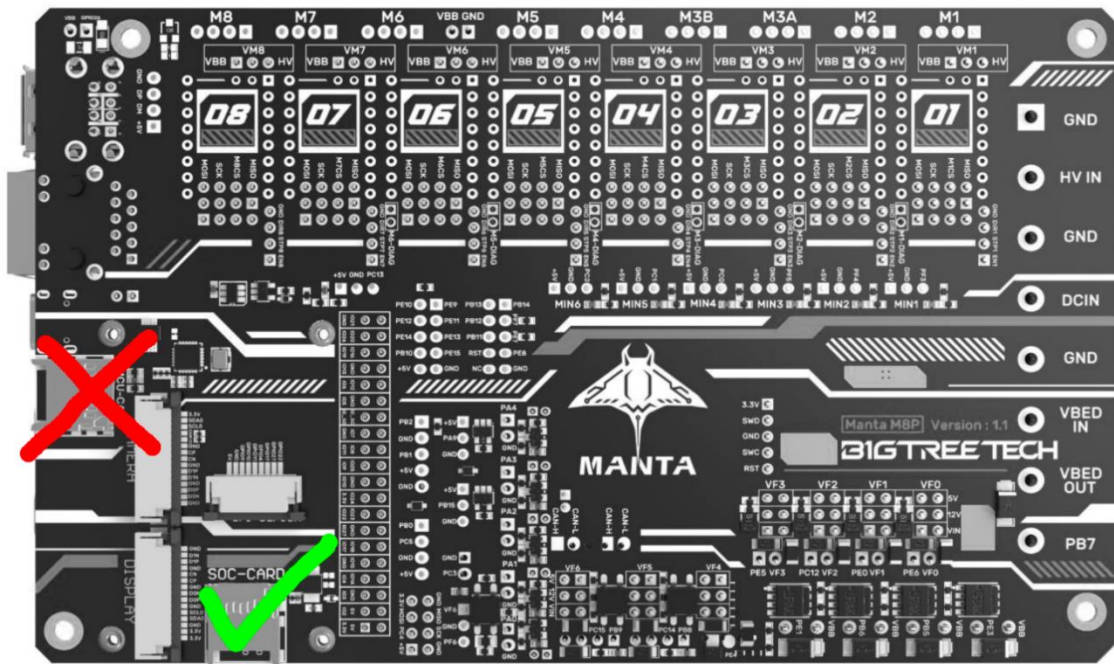
IMPORTANT: LEAVE "Speech Marks" in place – see below



```
1  #-----#
2  check_interval=5      # Cycle to detect whether wifi is connected, time 5s
3  router_ip=8.8.8.8     # Reference DNS, used to detect network connections
4
5  eth=eth0             # Ethernet card device number
6  wlan=wlan0           # Wireless NIC device number
7
8  #hostname="BTT-CB1"
9
10 #####
11 # System time zone setting, default Beijing time (East 8)
12 # More settable time zones can be viewed by running the command: timedatectl list-timezones
13 #TimeZone="Asia/Shanghai"
14
15 #####
16 # HDMI klipperScreen rotation
17 # ks_angle: Rotation angle
18 #     normal: 0;          inverted: 180;
19 #     left: 90 to left;   right: 90 to right;
20 #ks_angle="normal"
21
22 #####
23 # wifi name
24 WIFI_SSID="Your SSID"
25 # wifi password
26 WIFI_PASSWD="Your Password"
27
28 #####
29 WIFI_AP="false"        # Whether to open wifi AP mode, default off
30 WIFI_AP_SSID="rtl8189"  # Hotspot name created by wifi AP mode
31 WIFI_AP_PASSWD="12345678" # wifi AP mode to create hotspot connection password
32
33 #####
34 # BTT-PAD7 (ON/OFF)
35 BTT_PAD7="OFF"
36 # touch vibration effects
37 -----
```

Line 42, Column 44

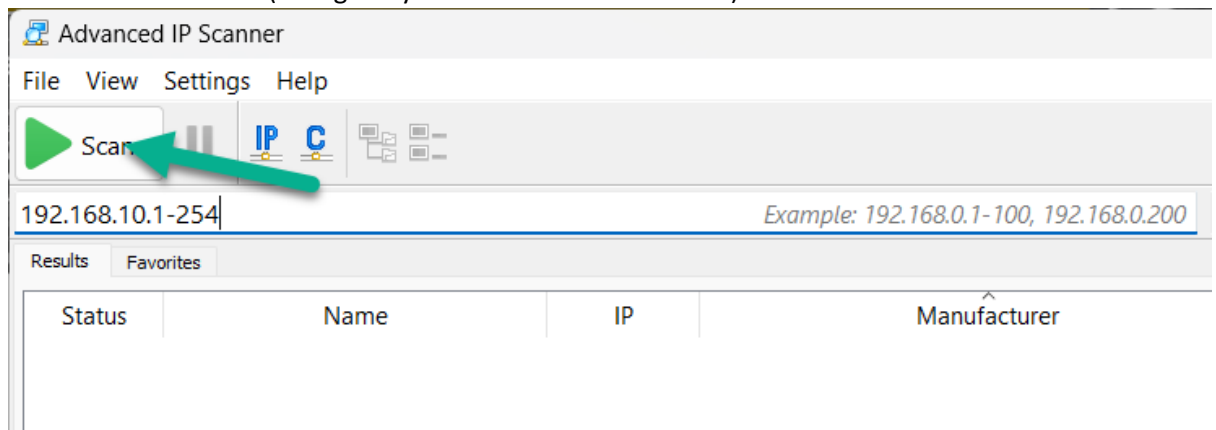
7. Save and close config.bin file.
8. Remove USB sd card drive and insert SD only into the SOC CARD socket and not the MCU socket.



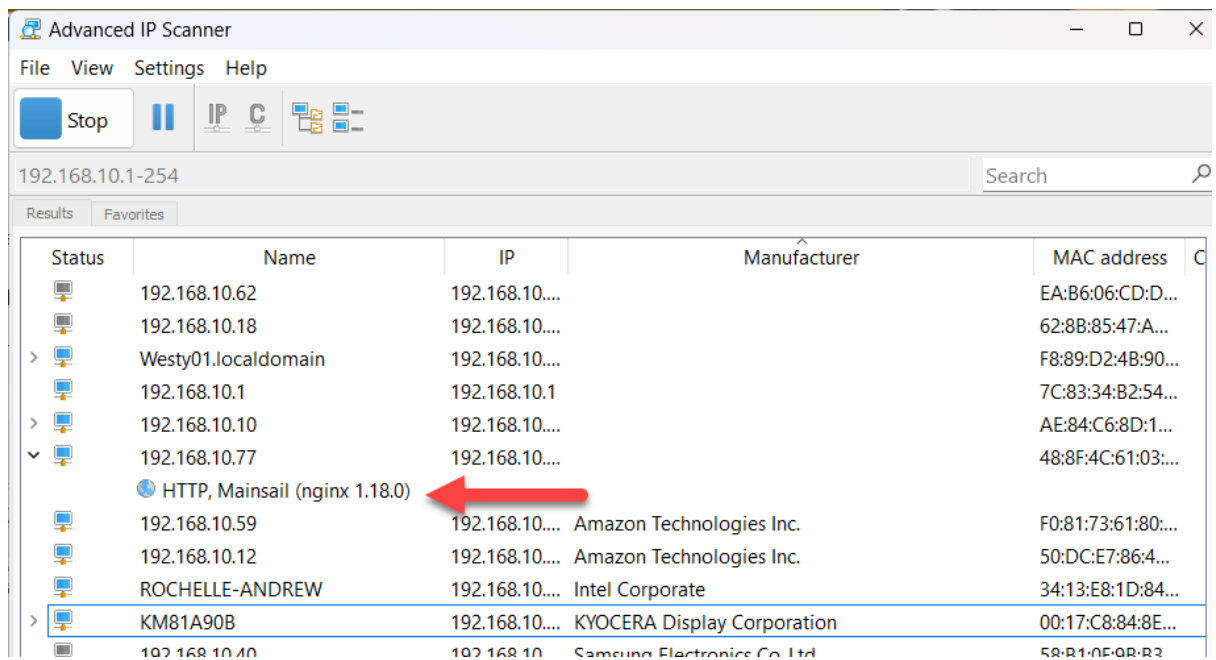
9. **MANTA M8P** - Plug your usb into a power supply either from your PC or a 5v power socket and connected the C end into the C power port on,
10. Wait about 30-60 seconds.
11. Make sure your PC is on the same network as your wireless network configured that your CB1 is booting from.

Finding your CB1 device on the network

1. Open Advanced IP Scanner and scan your network for the device. Put your network and start at .1 to 254 as shown (change to your own subnet as needed)

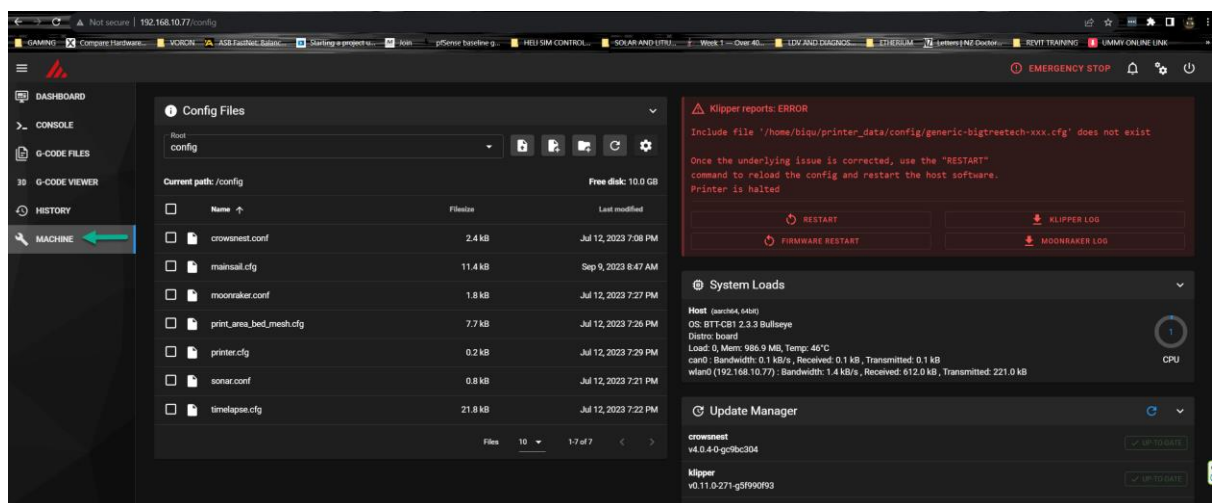


2. Select your IP Device (you may need to select the drop down in the list to find yours).

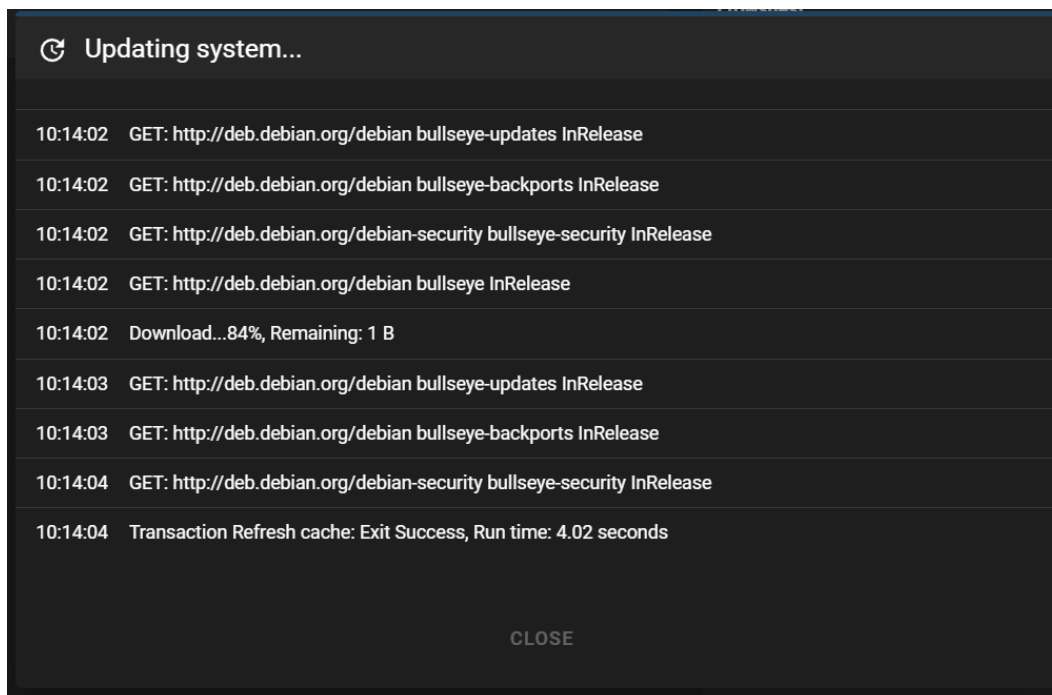
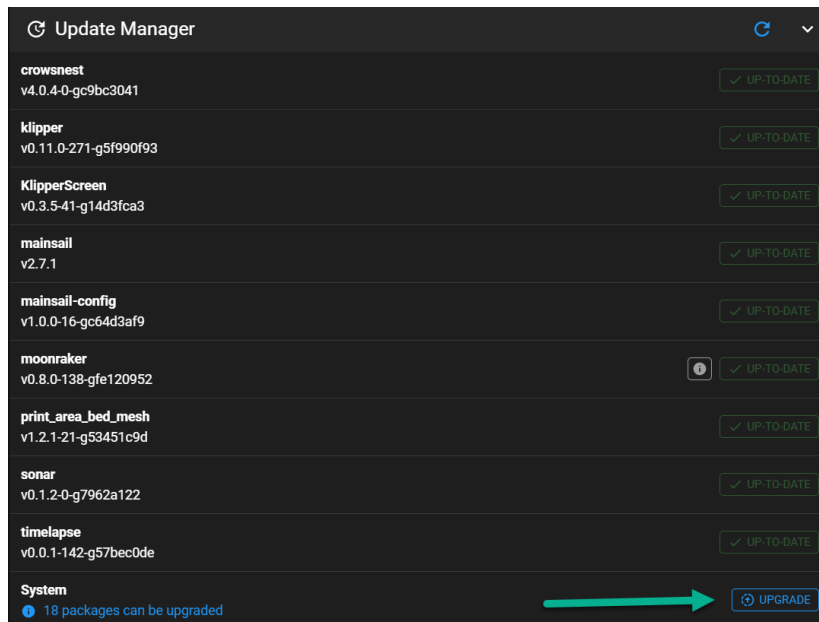


3. Double click on the HTTP. Mainsail arrowed above, this will open a browser page

Updating KLIPPER

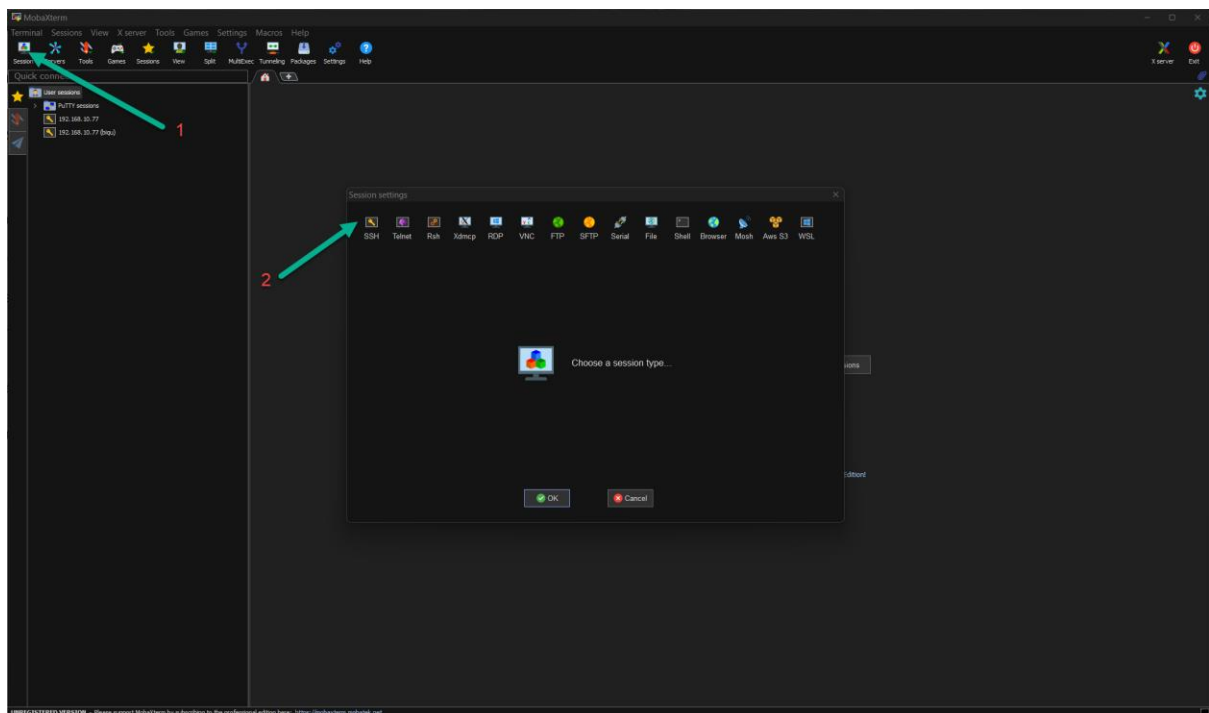


4. If you cannot see the page above, select Machine tab on the left as shown above.
5. Update all packages using the Upgrade or Update Buttons on the update manager.



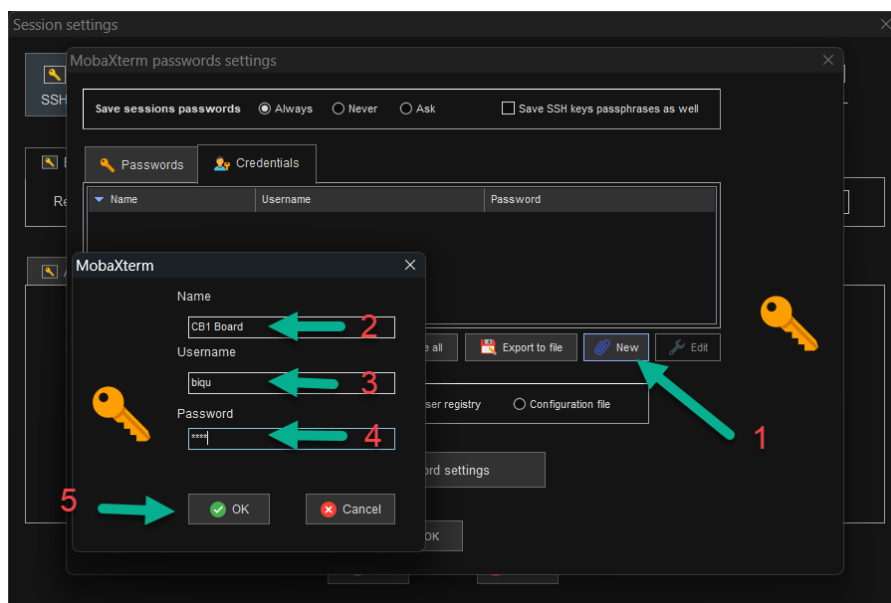
CONNECT TO THE CB1 VIA SSH

1. Open MobaXterm
2. Select session and then SSH

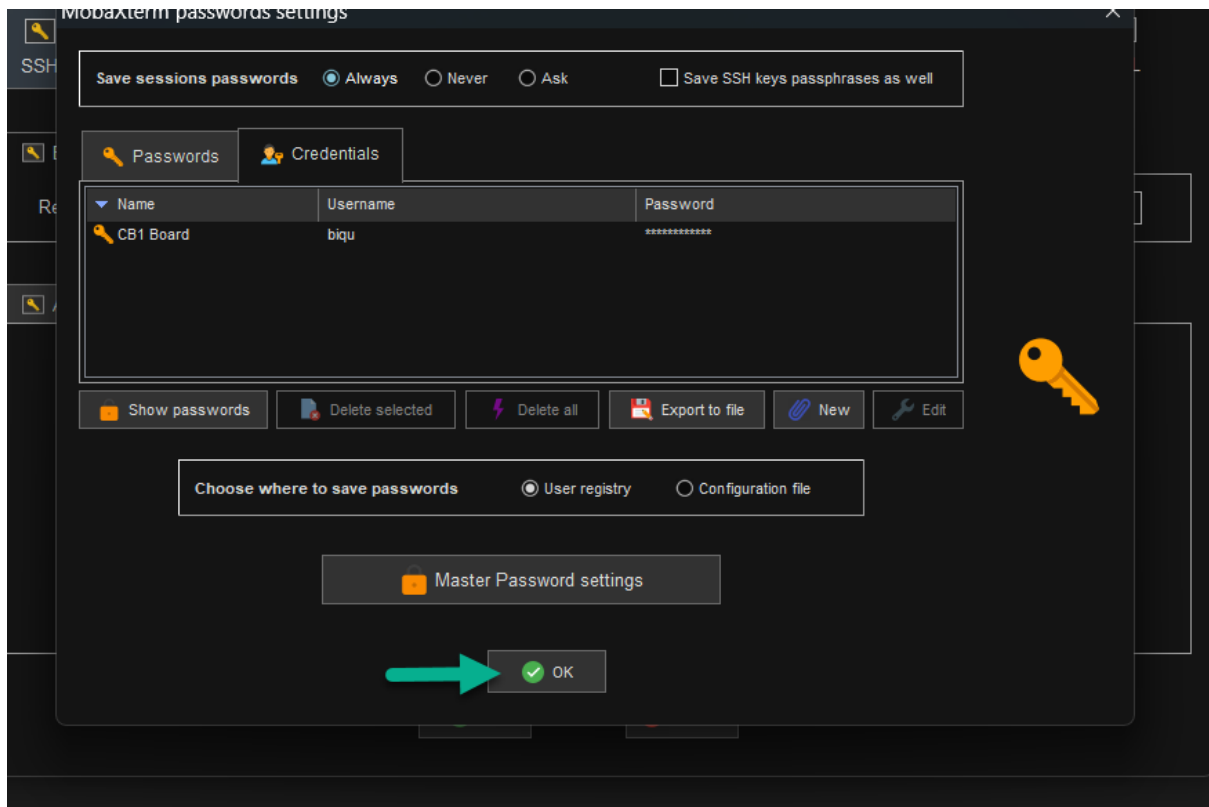


3. In the remote host section insert the IP address from your IP scanner as above and you can either hit OK... or to remember it for next time you connect (recommended) you can click on the little PERSON icon next to the Specify username drop box and do the following.

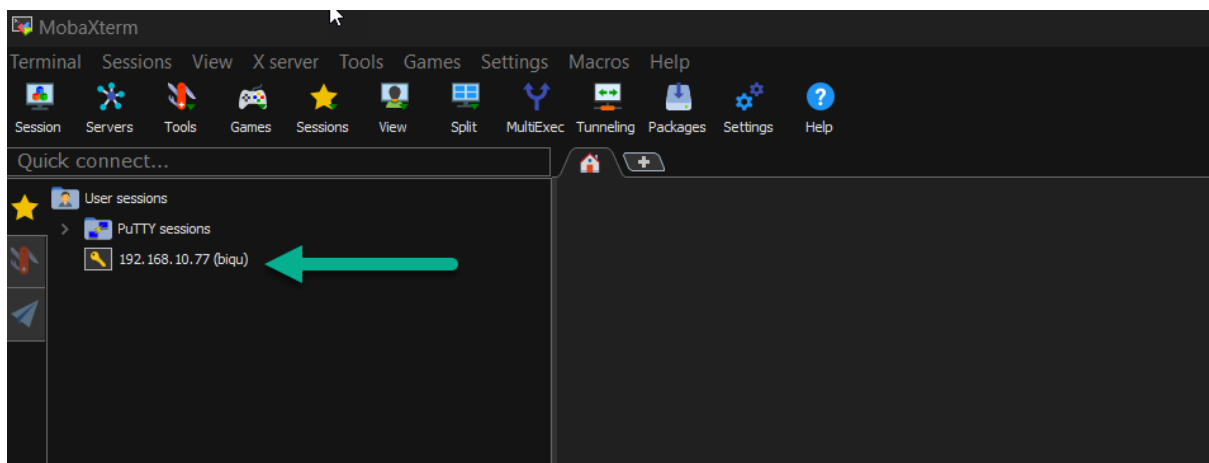
NOTE: Use **biqu** for user and **biqu** for password. All lower case.



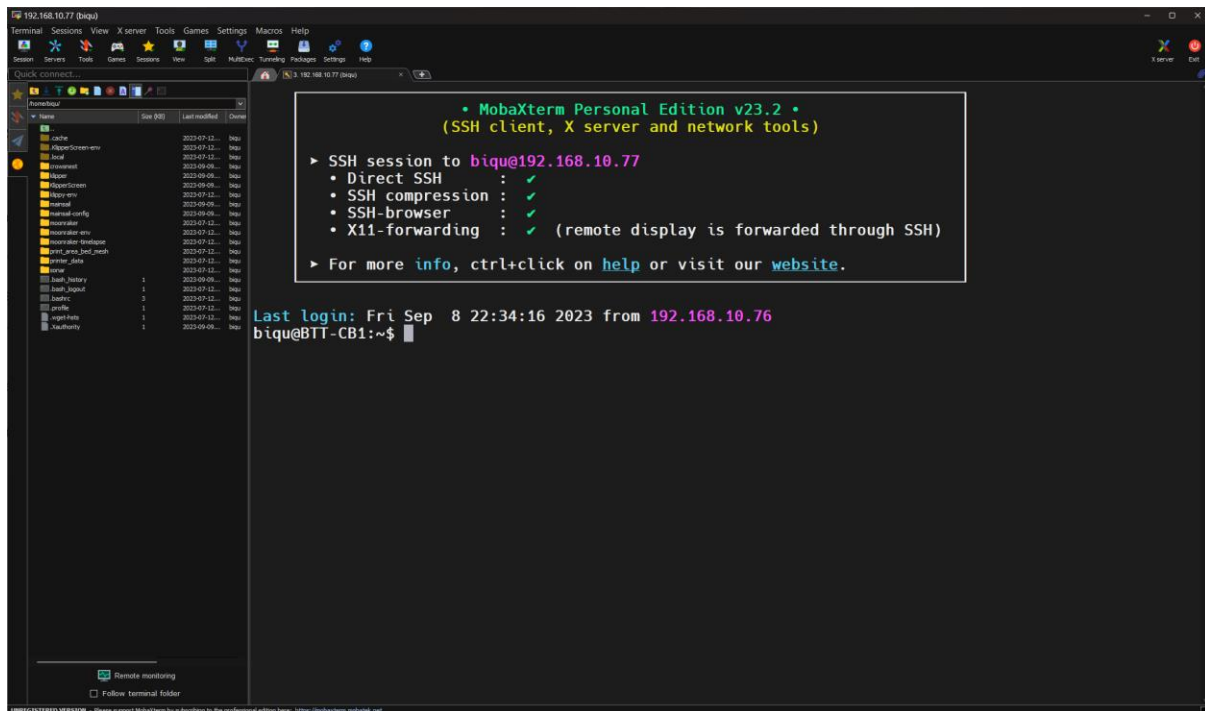
4. Select ok again.



5. You will not see your new permanent link file in the left pane.



6. Double click the new profile you have set up and it will set up an SSH session and auto log into your CB1 device.



Now we can configure the system. The nice thing about the MobaXterm is it shows all the files within the device and you can copy, paste, delete etc all here in one program.

What are we doing?

First step is to establish the main CAN Network Adaptor, Just like a network card or USB dongle that need to be configured with correct speeds, modes etc to get the network up and running.

We will be using Klipper and setting it up into “USB-CAN-Bridge Mode”. Use this to configure the rest of the network and devices and then change it back into Can mode to just communicate with the network.

First we should see what is on the network. We will look at 2 commands to do this.

1. Type `lsusb`

You should see something like this

```
biqu@BTT-CB1:~$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 003: ID 1d50:606f OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 002 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
biqu@BTT-CB1:~$
```

2. Type ifconfig

This will give you a more in depth look at your interfaces and packet loss, status etc. like below

```
Last login: Sat Sep  9 03:51:49 2023 from 192.168.10.76
biqu@BTT-CB1:~$ 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 1024  (UNSPEC)
    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

eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    ether 06:5b:91:02:15:62  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
    device interrupt 33

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 4630  bytes 2158453 (2.0 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 4630  bytes 2158453 (2.0 MiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.10.77  netmask 255.255.255.0  broadcast 192.168.10.255
    inet6 fe80::1446:efe2:c55c:6cf7  prefixlen 64  scopeid 0x20<link>
    ether 48:8f:4c:61:03:cf  txqueuelen 1000  (Ethernet)
    RX packets 6762  bytes 1461463 (1.3 MiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 1763  bytes 516371 (504.2 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

INSTALL DEPENDANCIES

1. TYPE

```
sudo apt update
sudo apt upgrade
sudo apt install python3 python3-pip python3-can
pip3 install pyserial
```

CB1 CONFIGURATION

1. Set CB1 Can Bridge Bitrate.

Copy and paste the following.

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

```
GNU nano 5.4 /etc/network/interfaces.d/can0
allow-hotplug can0
iface can0 can static
    bitrate 1000000
up ifconfig $IFACE txqueuelen 1024
```

2. Change bitrate to 1000000 (normally it is set to 500000 as standard).
3. Press CTRL X and Y to save on exit (save will only be offered if changing any parameters).
4. Once back in the normal terminal screen, type
sudo reboot

This will reboot the device. (You will need to reconnect your SSH session to it. Simply press R and the connection will reconnect).

```
• MobaXterm Personal Edition v23.2 •
(SSh client, X server and network tools)

> SSH session to biqu@192.168.10.77
• Direct SSH : ✓
• SSH compression : ✓
• SSH-browser : ✓
• X11-forwarding : ✓ (remote display is forwarded through SSH)

> For more info, ctrl+click on help or visit our website.

Last login: Fri Sep 8 22:34:16 2023 from 192.168.10.76
biqu@BTT-CB1:~$ sudo nano /etc/network/interfaces.d/can0
biqu@BTT-CB1:~$ reboot
biqu@BTT-CB1:~$
Remote side unexpectedly closed network connection

Session stopped
- Press <Return> to exit tab
- Press R to restart session
- Press S to save terminal output to file
```

FLASH M8P

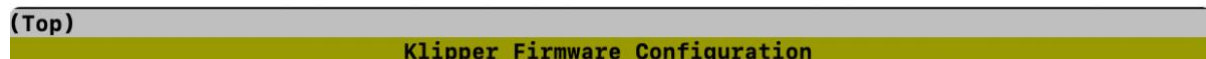
SSH back into the CB1 – ONCE CONNECTED TO THE FOLLOWING.

1. Type the following and then enter to execute the commands

```
cd ~/klipper
```

```
make menuconfig
```

This opens up the Klipper Firmware Configuration sub menu.



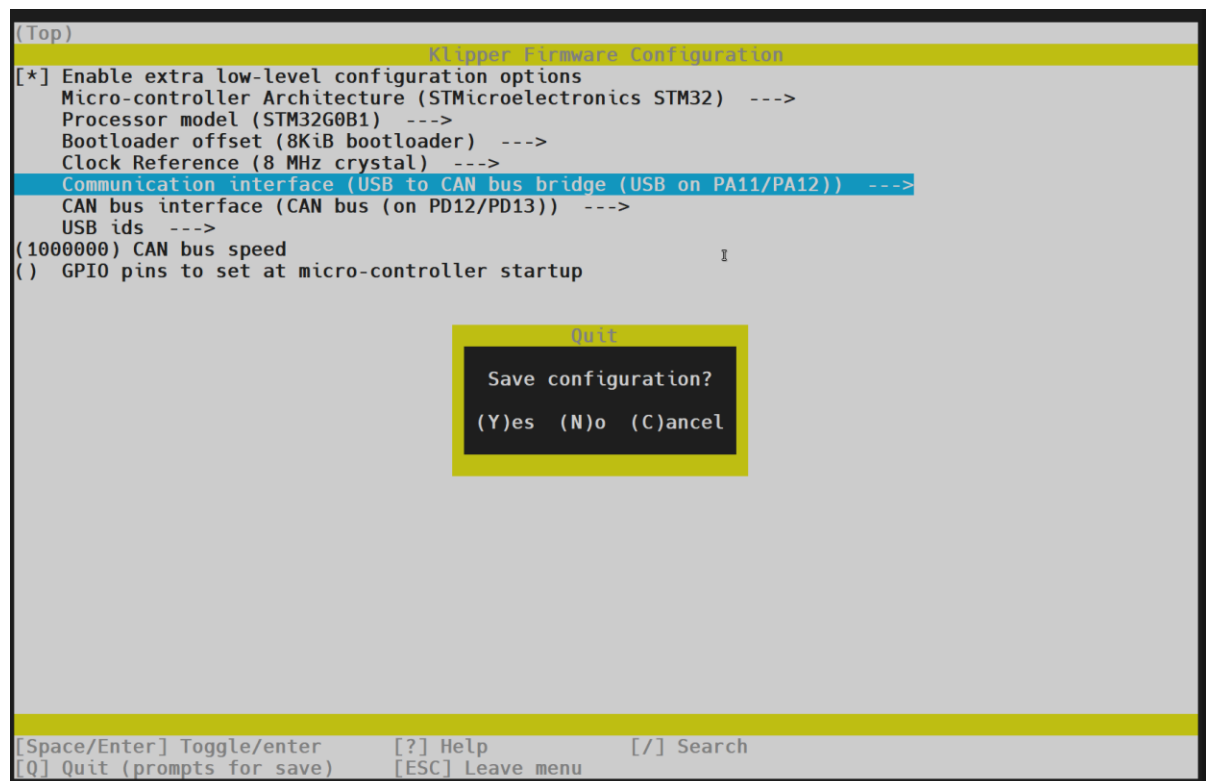
NAVIGATION:

- arrow keys to move up and down the menu
 - space bar to enable a section and add a star into it as selected
 - enter to select and go back to main screen of config menu
2. Match the menu items exactly as below.

NOTE: Make sure you set up the following line to open the Can Bus Interface line.

Communication interface (USB to CAN bus bridge (USB on PA11/PA12)) --->

Once setup – press Q to quit and Y to save.



3. Type make

This creates a file of klipper firmware file out/klipper.bin


```
► SSH session to biqu@192.168.10.77
  • Direct SSH      : ✓
  • SSH compression : ✓
  • SSH-browser     : ✓
  • X11-forwarding  : ✓ (remote display is forwarded through SSH)

► For more info, ctrl+click on help or visit our website.

BTT-001

Welcome to BTT-CB1 2.3.3 Bullseye with Linux 5.16.17-sun50iw9

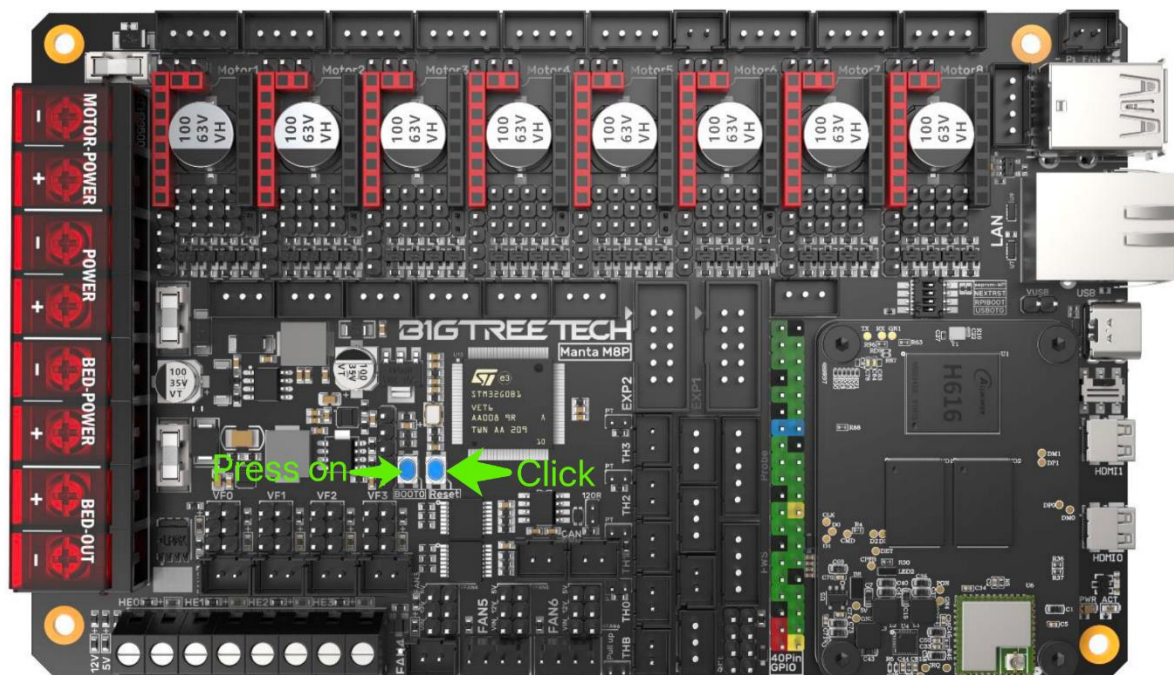
System load: 16%      Up time: 40 min
Memory usage: 20% of 986M  IP: 192.168.10.77
CPU temp: 45°C      Usage of /: 29% of 14G

Last login: Fri Sep  8 23:02:19 2023 from 192.168.10.76
biqu@BTT-CB1:~$ cd ~/klipper
biqu@BTT-CB1:~/klipper$ make
  Creating symbolic link out/board
  Building out/autoconf.h
  Compiling out/src/sched.o
  Compiling out/src/command.o
  Compiling out/src/basecmd.o
  Compiling out/src/debugcmds.o
  Compiling out/src/initial_pins.o
  Compiling out/src/gpiocmds.o
  Compiling out/src/stepper.o
```

SET MANTA M8P INTO DFU MODE

Now we need to place the Manta M8P into DFU mode.

1. PRESS (AND HOLD) BOOT0 and click RESET button. – release BOOT0



Now we do a check to see what the system sees.

2. Type `lsusb`

NOTE: the below will come up – or similar (if you only see something like, ID 1d50:606f OpenMoko, Inc. Geschwister Schneider CAN adapter – it is not in DFU mode. Try again. Repeat step 2.

```
biqu@BTT-CB1:~/klipper$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 005: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 002 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
biqu@BTT-CB1:~/klipper$
```

There should be a DFU Mode device found, copy this address ID to a txt file for later, normally it's 0483:df11

3. Type `make flash FLASH_DEVICE=0483:df11`
(enter your address if this was different to the above.)

This will load the files to flash the device. NOT IT IS OK IF YOU SEE AND ERROR 255 AS BELOW

```
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

dfu-util: Invalid DFU suffix signature
dfu-util: A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0483:df11
Run-time device DFU version 011a
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 1024
DfuSe interface name: "Internal Flash"
Downloading to address = 0x08002000, size = 30716
Download [=====] 100% 30716 bytes
Download done.
File downloaded successfully
dfu-util: Error during download get_status

Failed to flash to 0483:df11: Error running dfu-util

If the device is already in bootloader mode it can be flashed with the
following command:
    make flash FLASH_DEVICE=0483:df11
    OR
    make flash FLASH_DEVICE=1209:beba

If attempting to flash via 3.3V serial, then use:
    make serialflash FLASH_DEVICE=0483:df11

make: *** [src/stm32/Makefile:111: flash] Error 255
biqu@BTT-CB1:~/klipper$
```

4. Press and then release the RESET button only on the M8P – this will reboot the Manta M8P only and leave the CB1 still operational. – Should also take the Manta M8P out of MTU mode.

CONGRATULATION FLASHING MANTA M8P DONE! – JUST ONE THING TO DO.

CHECK MANTA M8P CAN uuid address

1. Type `python3 lib/canboot/flash_can.py -q`

```
biqu@BTT-CB1:~/klipper$ python3 lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes
Detected UUID: 8d8160cc0a82, Application: Klipper
Query Complete
biqu@BTT-CB1:~/klipper$
```

2. Copy the Can address in your txt file as klipper

UUID: 8d8160cc0a82, Application: Klipper (YOURS WILL BE DIFFERENT)

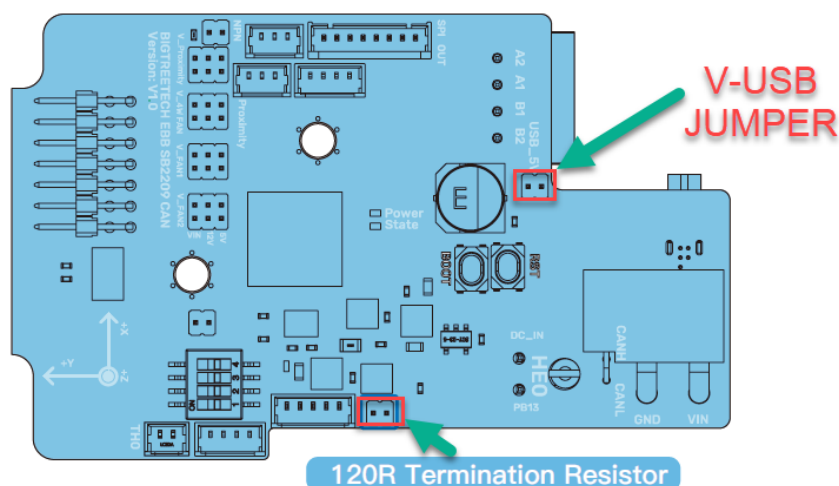
INSTALL CANBOOT

NOTE: BEFORE FLASHING OR POWERING UP THIS DEVICE, YOU MUST INSTALL THE 2 MICRO JUMPERS IN THE LOCATIONS AS SHOWN. – USB JUMPER WILL BE REMOVED ONCE PROGRAMMING IS COMPLETED FOR INSTALLATION.

CAN BUS 120R TERMINATION RESISTOR

WWW.BIGTREE-TECH.COM

When the EBB SB2240/2209 CAN device uses CAN bus communication, if it is the final device in the CAN bus chain, you must plug a jumper at the 120R position.



1. With the Manta M8P and CB1 combination up and running and SSH connected as we left it above, type (copy and paste all at once and hit enter)

```
cd ~  
git clone https://github.com/Arksine/CanBoot  
cd CanBoot
```

This will change directories, search and download the latest repository of CanBoot (yep it is case sensitive), change directory to CanBoot and make the menu config
MATCH THE BELOW SETTINGS

```
Run-time device DFU version 011a  
Claiming USB DFU Interface...  
Setting Alternate Setting #0 ...  
Determining device status: state = dfuIDLE, status = 0  
dfuIDLE, continuing  
DFU mode device DFU version 011a  
Device returned transfer size 1024  
DfuSe interface name: "Internal Flash"  
Downloading to address = 0x08002000, size = 30716  
Download [=====] 100% 30716 bytes  
Download done.  
File downloaded successfully  
dfu-util: Error during download get_status  
  
Failed to flash to 0483:df11: Error running dfu-util  
  
If the device is already in bootloader mode it can be flashed with the  
following command:  
make flash FLASH_DEVICE=0483:df11  
OR  
make flash FLASH_DEVICE=1209:beba  
  
If attempting to flash via 3.3V serial, then use:  
make serialflash FLASH_DEVICE=0483:df11  
  
make: *** [src/stm32/Makefile:111: flash] Error 255  
biqu@BTT-CB1:~/klipper$ python3 lib/canboot/flash_can.py -q  
Resetting all bootloader node IDs...  
Checking for canboot nodes...  
Detected UUID: 8d8160cc0a82, Application: Klipper  
Query Complete  
biqu@BTT-CB1:~/klipper$ cd ~  
git clone https://github.com/Arksine/CanBoot  
cd CanBoot  
make menuconfig
```

IF YOU SEE AN ERROR LIKE THE ABOVE, THIS IS OK AS LONG AS YOU SEE FILE LOADED SUCCESSFULLY.

```

Downloading to address = 0x08002000, size = 30716
Download [=====] 100% 30716 bytes
Download done.
File downloaded successfully
dfu-util: Error during download get_status
Failed to flash to 0483:df11: Error running dfu-util
If the device is already in bootloader mode it can be flashed with the
following command:
make flash FLASH_DEVICE=0483:df11
OR
make flash FLASH_DEVICE=1209:beba
If attempting to flash via 3.3V serial, then use:
make serialflash FLASH_DEVICE=0483:df11
make: *** [src/stm32/Makefile:111: flash] Error 255
biqu@BIT-CB1:~/klipper$ python3 lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 8d8160cc0a82, Application: Klipper
Query Complete
biqu@BIT-CB1:~/klipper$ cd ~
git clone https://github.com/Arksine/CanBoot
cd CanBoot
make menuconfig
Cloning into 'CanBoot'...
remote: Enumerating objects: 1629, done.
remote: Counting objects: 100% (325/325), done.
remote: Compressing objects: 100% (133/133), done.
remote: Total 1629 (delta 231), reused 222 (delta 191), pack-reused 1304
Receiving objects: 100% (1629/1629), 3.85 MiB | 2.17 MiB/s, done.
Resolving deltas: 100% (1083/1083), done.

```

You will end up with the yellow image at the bottom (not it is different from the KLIPPER CONFIG YELLOW MENU)

MATCH THE SETTINGS AS SHOWN BELOW.

```

(Top)
Katapult Configuration v0.0.1-57-gabd1545
Micro-controller Architecture (STMicroelectronics STM32) --->
Processor model (STM32G0B1) --->
Build Katapult deployment application (Do not build) --->
Clock Reference (8 MHz crystal) --->
Communication interface (CAN bus (on PB0/PB1)) --->
Application start offset (8KiB offset) --->
(1000000) CAN bus speed
() GPIO pins to set on bootloader entry
[*] Support bootloader entry on rapid double click of reset button
[ ] Enable bootloader entry on button (or gpio) state
[*] Enable Status LED
(PA13) Status LED GPIO Pin

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

```

WATCH VIDEO HERE – add link

2. Once done press Q and y to save you will end up here once it exits. (NOTE THIS IS ALL DONE WITHOUT CONNECTING THE EBB SB2209 BOARD)

YOU END UP HERE.


```

make menuconfig
Cloning into 'CanBoot'...
remote: Enumerating objects: 1629, done.
remote: Counting objects: 100% (325/325), done.
remote: Compressing objects: 100% (133/133), done.
remote: Total 1629 (delta 231), reused 222 (delta 191), pack-reused 1304
Receiving objects: 100% (1629/1629), 3.85 MiB | 2.17 MiB/s, done.
Resolving deltas: 100% (1083/1083), done.
Using default symbol values (no '/home/biqu/CanBoot/.config')
Configuration saved to '/home/biqu/CanBoot/.config'
  Creating symbolic link out/board
Loaded configuration '/home/biqu/CanBoot/.config'
Configuration saved to '/home/biqu/CanBoot/.config'
biqu@BTT-CB1:~/CanBoot$

```

3. Type `make`

4.

```

biqu@BTT-CB1:~/CanBoot$ make
  Creating symbolic link out/board
  Building out/autoconf.h
  Compiling out/src/sched.o
  Compiling out/src/bootentry.o
  Compiling out/src/command.o
  Compiling out/src/flashcmd.o
  Compiling out/src/initial_pins.o
  Compiling out/src/led.o
  Compiling out/src/generic/armcm_canboot.o
  Compiling out/src/stm32/gpio.o
  Compiling out/src/stm32/flash.o
  Compiling out/src/stm32/clockline.o
  Compiling out/src/stm32/dfu_reboot.o
  Compiling out/src/generic/armcm_irq.o

```

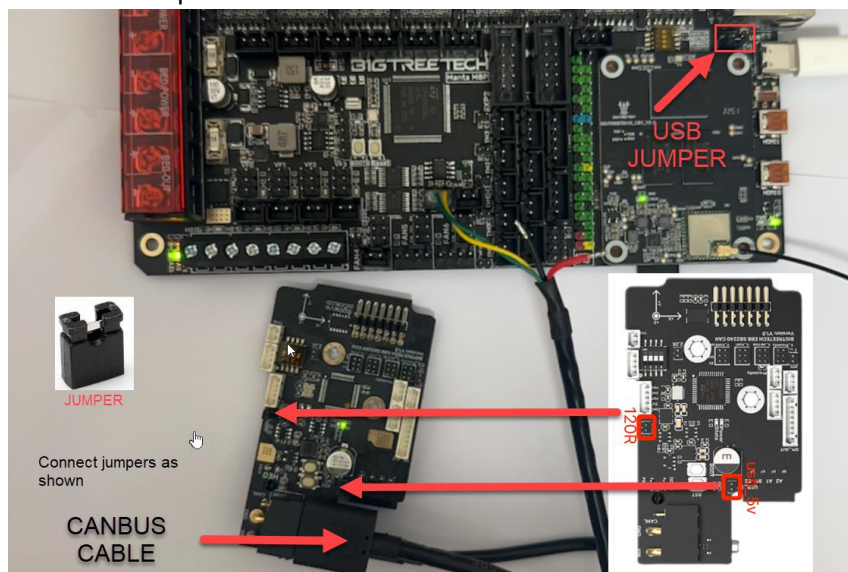
5.

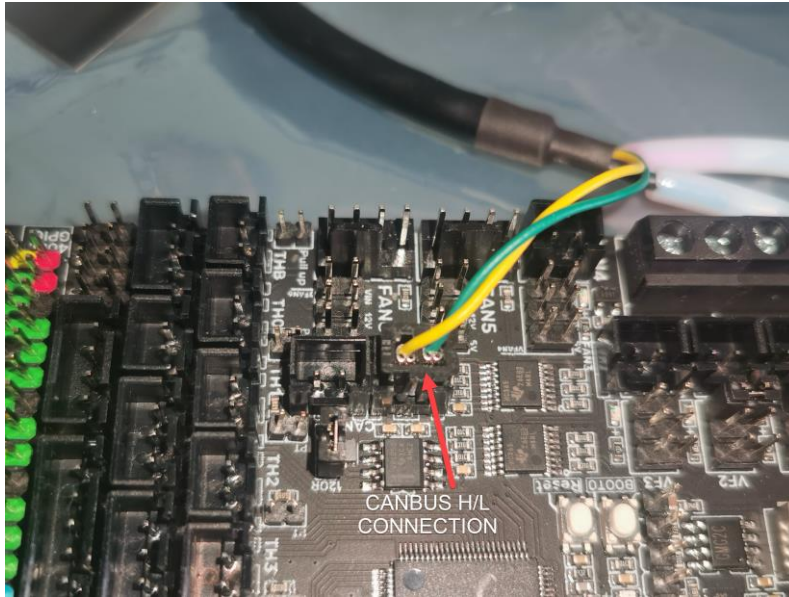
This builds the file set needed.

POWER UP EBB SB2209

We need to connect up the wiring, confirm jumpers and put the sb2209 into DFU mode so we can re flash it.

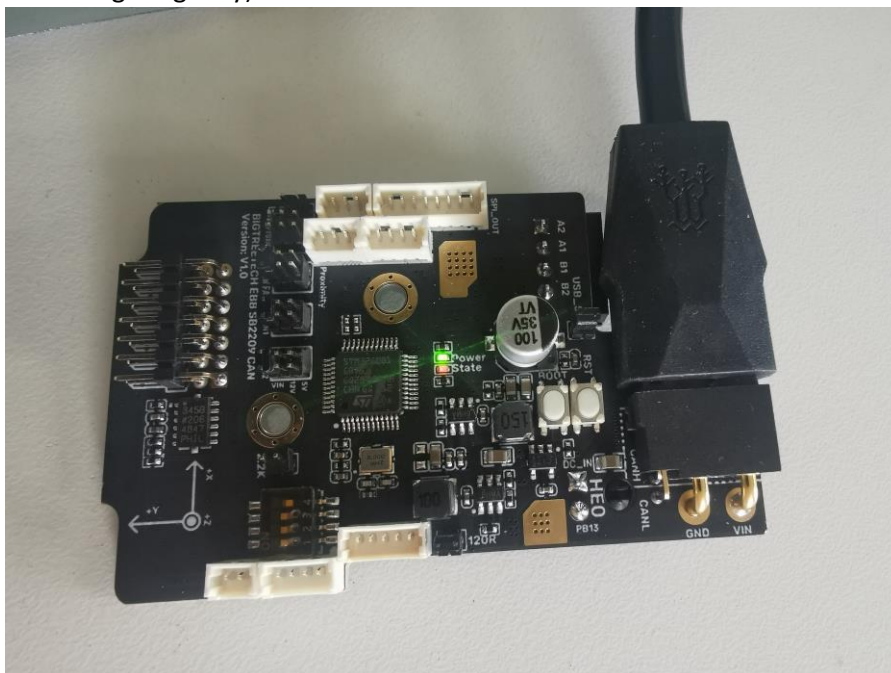
1. Connect up CABLES AS BELOW 2 IMAGES





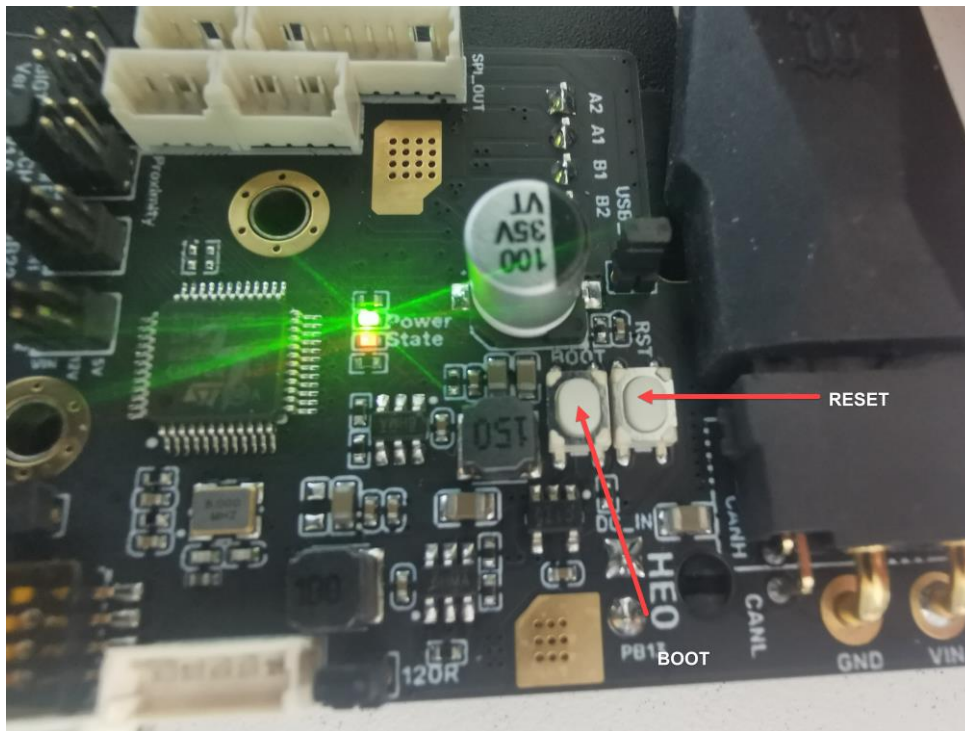
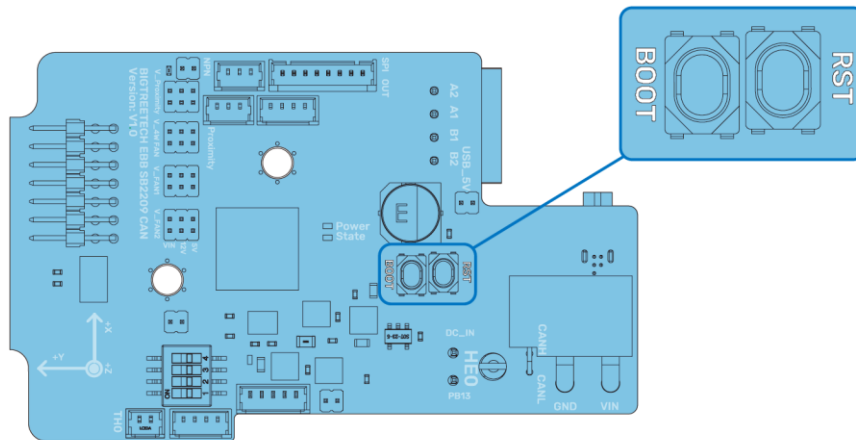
With Manta M8P and CB1 Up and running as before

2. Plug your USB-C into the SB2209 and then connect up the USB into the Manta USB output plug. This will power up the SB2209 board. (note, you cannot see the USB C cable connected underneath the board- this is powered from the Manta USB normal port for configuring only).



3. **EBB SB2209** PRESS AND HOLD BOOT BUTTON AND RESET THEN RELEASE BOOT.

4.2. Press and hold the Boot button, and then click the Reset button to enter the DFU mode.



4. TYPE lsusb

This will show all devices and you will see a device in DFU mode

```
biqu@BTT-CB1:~/CanBoot$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 007: ID 1d50:606f OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 002 Device 009: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 002 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
biqu@BTT-CB1:~/CanBoot$ make
```

We can see 0483:df11 is the address in DFU mode.

Now we can flash it

5. Type

`dfu-util -a 0 -d 0483:df11 --dfuse-address 0x08000000 -D ~/CanBoot/out/canboot.bin`

```
biqu@BTT-CB1:~/CanBoot$ dfu-util -a 0 -d 0483:df11 --dfuse-address 0x08000000 -D ~/CanBoot/out/canboot.bin
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

dfu-util: Invalid DFU suffix signature
dfu-util: A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0483:df11
Run-time device DFU version 011a
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 1024
DfuSe interface name: "Internal Flash"
Downloading to address = 0x08000000, size = 4468
Download [=====] 100% 4468 bytes
Download done.
File downloaded successfully
biqu@BTT-CB1:~/CanBoot$
```

SUCCESS! – EBB SB2209 is now flashed. – PAT YOURSELF ON THE BACK!

FLASH KLIPPER BY USB – switch from bridge mode to can mode

1. TYPE AND ENTER

`cd ~/klipper`

`make menuconfig`

change to match this


```

(Top)
Klipper Firmware Configuration
[*] Enable extra low-level configuration options
  Micro-controller Architecture (STMicroelectronics STM32) --->
  Processor model (STM32G0B1) --->
  Bootloader offset (8KiB 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

```

2. Press q and then Y to save
3. TYPE make

SET EBB SB2209 INTO DFU MODE AGAIN.

1. Hold BOOT BUTTON AND PRESS AND RELEASE RESET – THEN RELEASE BOOT BUTTON
2. Type lsusb

```

biqu@BTT-CB1:~/klipper$ lsusb
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 003: ID 1d50:606f OpenMoko, Inc. Geschwister Schneider CAN adapter
Bus 002 Device 004: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
Bus 002 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
biqu@BTT-CB1:~/klipper$
biqu@BTT-CB1:~/klipper$

```

You will see the DFU mode device.

Now flash it

3. Type dfu-util -a 0 -d 0483:df11 --dfuse-address 0x08002000 -D out/klipper.bin

Once that process is complete. We need to do some checks.

CHECKS

Canboot check

1. On the EBB SB2209 do a quick double press of the Reset button (not the boot) and the red LED should blink 1 per second.

2. Type


```
cd ~/klipper
python3 lib/canboot/flash_can.py -q
```

If you get the below response, you can successfully flashed canboot

```
File downloaded successfully
biqu@BTT-CB1:~/klipper$ cd ~/klipper
python3 lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 8d8160cc0a82, Application: Klipper
Detected UUID: b57b8507b46f, Application: CanBoot
Query Complete
biqu@BTT-CB1:~/klipper$ █
```

KLIPPER CHECK

On THE EBB SB2209

1. Press the RESET once. This should stop the blinking red LED.
2. Type `python3 lib/canboot/flash_can.py -q`

```
biqu@BTT-CB1:~/klipper$ python3 lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 8d8160cc0a82, Application: Klipper
Detected UUID: b57b8507b46f, Application: Klipper
Query Complete
biqu@BTT-CB1:~/klipper$ █
```

```
python3 lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 8d8160cc0a82, Application: Klipper
Detected UUID: b57b8507b46f, Application: CanBoot
Query Complete
biqu@BTT-CB1:~/klipper$ python3 lib/canboot/flash_can.py -q
Resetting all bootloader node IDs...
Checking for canboot nodes...
Detected UUID: 8d8160cc0a82, Application: Klipper
Detected UUID: b57b8507b46f, Application: Klipper
Query Complete
biqu@BTT-CB1:~/klipper$ █
```

That is all the flashing completed now

UUID: 8d8160cc0a82 = Klipper – save this for your printer.cfg file

[mcu]

canbus_uuid:8d8160cc0a82

UUID: b57b8507b46f, CanBoot toolhead

[mcu EBBCan]

canbus_uuid: b57b8507b46f

These of course will be unique and yours will be different.

Finally :

REMOVE USB JUMPER OFF EBB SB 2209 & MANTA M8P

Below are some diagnostic lines that can be useful when troubleshooting faults.

ip -details -statistics link show can0 shows interface statistics, including error counters

canbusload can0@1000000 -b -c shows the estimated load on the CAN bus. Specify the baud rate after the @. If CAN bridge mode is used then these values will be wrong

candump -d -e can0 -H -t a dumps raw traffic on the CAN bus

For the last two commands the package can-utils is required, install using"

apt install can-utils

All commands may require a sudo in front of them, depending on which permissions your user has.