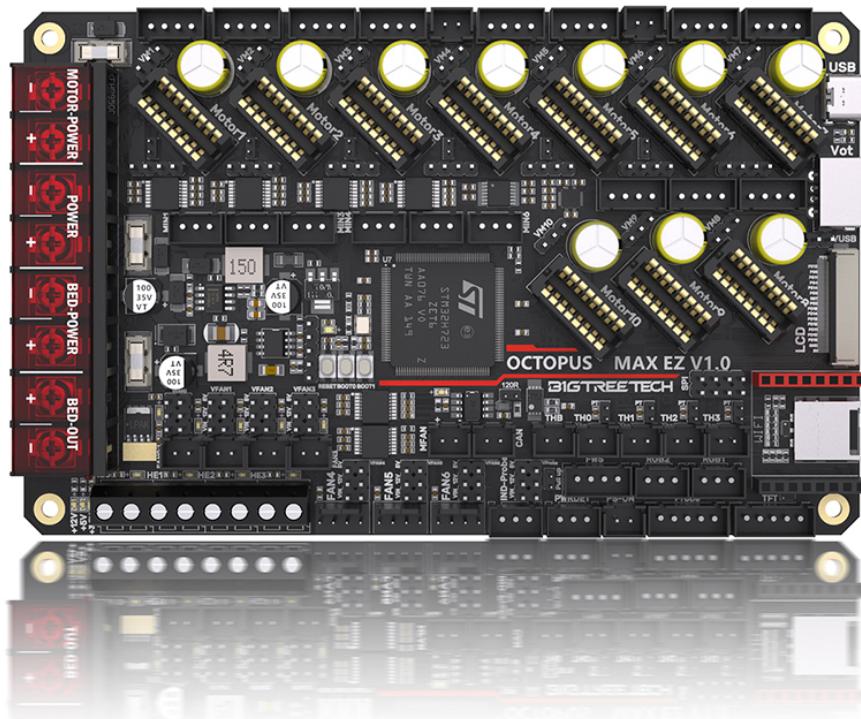


BIGTREE TECH

Octopus MAX EZ V1.0

User Manual



Revision Log

Version	Date	Revisions
v1.00	6th October 2022	Initial Version
v1.01	27th September 2023	Instructions for updating Klipper firmware via DFU.

CONTENTS

Revision Log	2
Product Profile.....	5
Feature Highlights	5
Specifications	6
Dimensions	7
Peripheral Port	8
Connector Diagram.....	8
Pinout Diagram	8
Connection Description	9
USB Power Supply.....	9
Stepper Motor Driver.....	9
UART/SPI Mode of Driver	9
TMC Driver DIAG (Sensorless Homing).....	9
Driver Voltage Selection.....	9
Voltage Selection for CNC Fan.....	10
100K NTC or PT1000 Setting.....	10
BLTouch Wiring	11
Auto Power Off (Relay V1.2) Wiring	11
Connecting with MINI12864/TFT Screen.....	12
RGB Wiring	12
Filament Sensor Wiring	13
Proximity Switch Wiring.....	13
Wiring of 4 pins CNC Fan and Water Cooling System.....	15
Marlin	16
Install Compiling Environment.....	16
Download Marlin Firmware.....	16
Configure Firmware.....	16
Open Marlin Project	16
Compiling Environment.....	16

Configure Motherboard and Serial Port.....	17
Configure Stepper Driver.....	18
Sensorless Homing	19
100K NTC or PT1000.....	20
BLTouch	20
Auto Power Off(Relay V1.2)	23
Power Loss Recovery	23
RGB	24
Filament Sensor.....	25
Smart Filament Sensor (SFS V1.0)	26
ESP3D	27
Compile Firmware	28
Klipper.....	29
Preparation	29
Download OS Image	29
Download and Install Raspberry Pi Imager.....	29
Write Image.....	30
WiFi Setting.....	32
SSH Connect to Raspberry Pi	32
Compile Firmware	34
Configure Klipper	35
Firmware Updates.....	37
Updating via microSD.....	37
Updating Klipper via DFU.....	37
Precautions	38

Product Profile

BIGTREETECH Octopus MAX EZ is an optimized 32-bit 3D printer control board based on the Octopus Pro, using self-developed stepper motor driver sockets to enhance safety and user experience, with many new features not found on Octopus Pro to increase DIY potential.

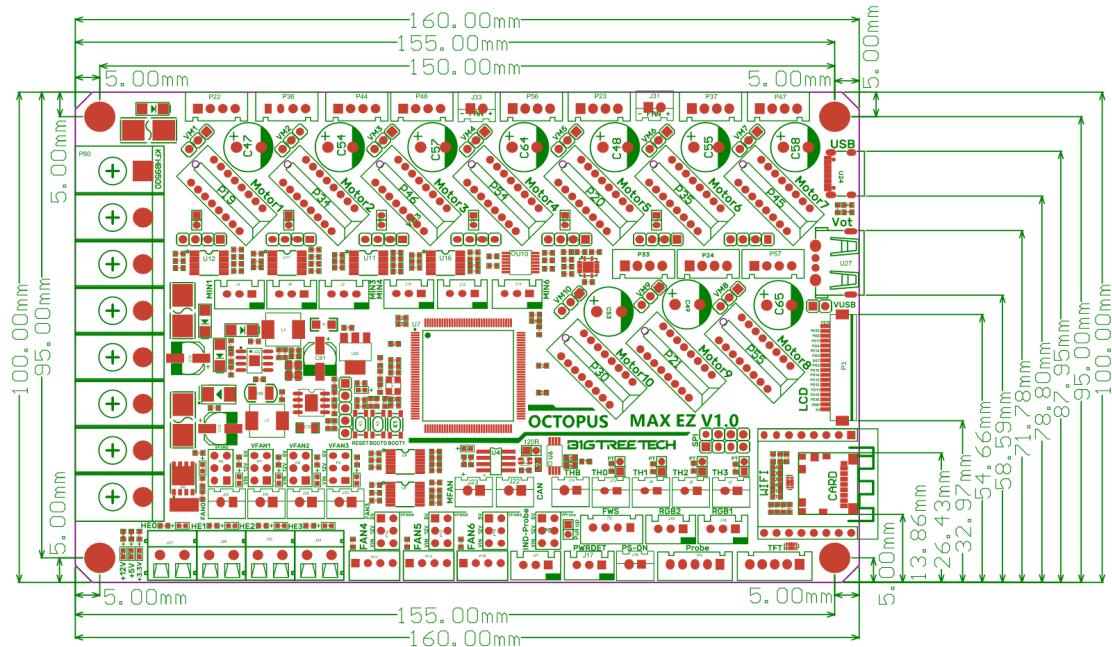
Feature Highlights

- 32 bit 550 MHz ARM Cortex-M7 series STM32H723ZET6 MCU;
- Onboard BOOT button to enable DFU mode to update bootloader;
- The thermistor circuit is protected to prevent MCU damage from shorted heated bed and heater cartridge connections;
- Selectable voltage (24V, 12V, 5V) for CNC fan, eliminating the need for external buck modules, thereby reducing the likelihood of motherboard damage.
- Upgraded with eFuse protection, which responds faster with strong protection, effectively protecting the motherboard from being damaged caused by short circuits, over-current, electric spark, etc.
- MCU firmware can be upgraded via SD card, or use DFU via Klipper's make flash command;
- 10 EZ driver sockets, working with pinless driver, safer to use; Onboard SPI and UART, can be used by simply setting in the firmware, no need for a jumper.
- Support power loss recovery, filament runout sensor, CAN, auto power-off, BLTouch, RGB, etc;
- Replaceable fuse for easy maintenance;
- 3 x 4 pins fan ports, also for connecting water cooling system;
- Onboard proximity switch port, supports NPN and PNP types, 24V, 12V, 5V voltage selectable;
- Onboard SPI interface for connecting acceleration sensor to enable Klipper's input shaping.

Specifications

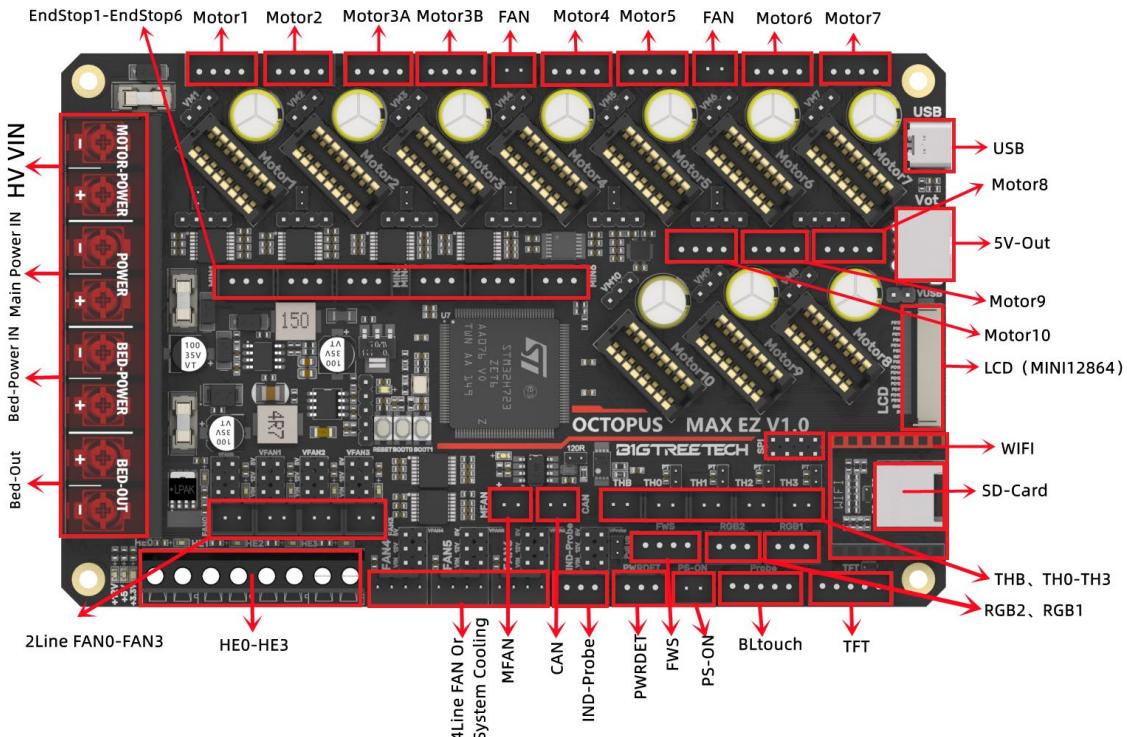
Dimensions	160mm x 100mm for details please refer to BIGTREETECH Octopus MAX EZ V1.0-SIZE.pdf
Mounting Size	Please refer to BIGTREETECH Octopus MAX EZ V1.0-SIZE.pdf
MCU	ARM Cortex-M7 STM32H723ZET6 550MHz
Driver	
Input Voltage	24V, HV(≤56V) Selectable
Motherboard Input Voltage	VIN=DC12V or DC24V
Heated Bed Input Voltage	BED IN=DC12V or DC24V
Logic Voltage	DC 3.3V
Heater Connection	Heated Bed (HB), Heater Cartridge (HE0, HE1, HE2, HE3)
HB Port Max Current	10A Continuous, 12A Instantaneous
Heater Cartridge Max Current	5.5A Continuous, 6A Instantaneous
Fan Port	2 pins CNC Fan (FAN0, FAN1, FAN2, FAN3), 4 pins CNC Fan (FAN4, FAN5, FAN6), Always On (24V FAN x 2). CNC Fan and MFAN Voltage Selectable (5/12/24V)
Fan Port Max Current	1A Continuous, 1.5A Instantaneous
Overall Max Current (Heater Cartridge+Driver+All Fans)	< 12A
Expansion Port	BLTouch (Servos, Probe), PS-ON, FWS, PWRDET, RGBx2, SPI, IND-Probe, CAN, WIFI, TFT
Motor Driver	Support EZ5160, EZ2209, EZ2225, EZ2226, EZ2208, EZ2130...
Driver Mode	SPI, UART
Motor Socket	Motor1, Motor2, Motor3 (Dual Motor Sockets), Motor4, Motor5, Motor6, Motor7, Motor8, Motor9, Motor10 10 Channels in Total
Thermistor	5 x 100K NTC, four of which are selectable for NTC and PT1000
Display	MINI12864 (FPC Connection), TFT Serial
PC Connection	Type-C
Supported Kinematics	Cartesian, Delta, Kossel, Ultimaker, CoreXY
Recommended Slicer/Console	Cura, Simplify3D, Pronterface, Repetier-host, Makerware

Dimensions

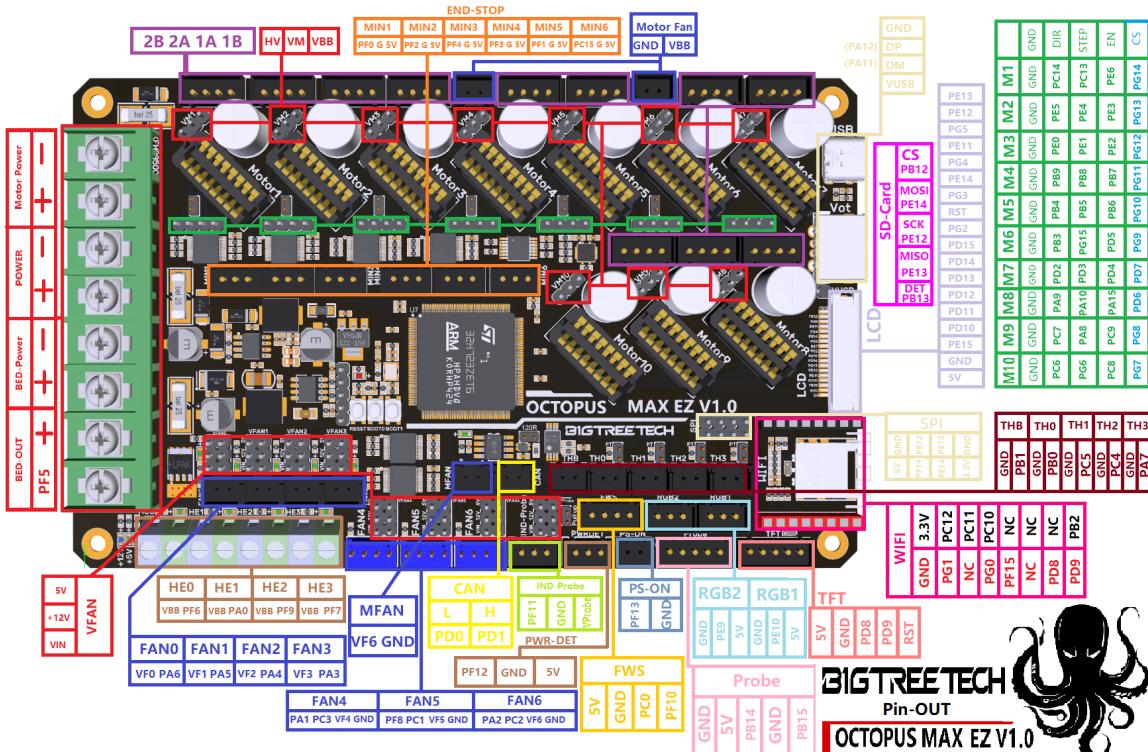


Peripheral Port

Connector Diagram



Pinout Diagram



Connection Description

USB Power Supply

After the Octopus MAX EZ has been powered, the Red light D32 on the left side of the MCU will light up, indicating power on. When using only USB to power the board or to supply power via USB, please insert the jumper cap onto the VUSB.



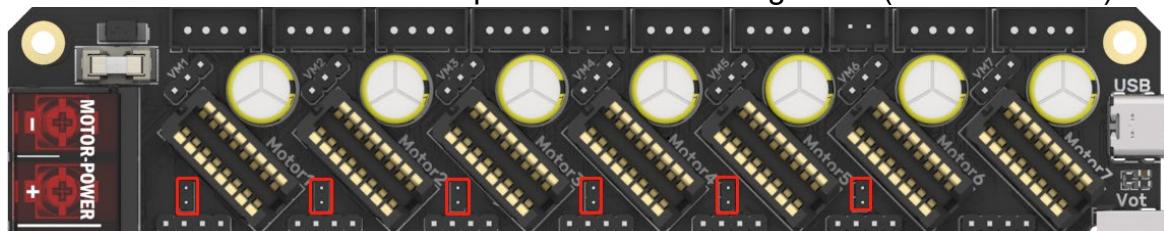
Stepper Motor Driver

UART/SPI Mode of Driver

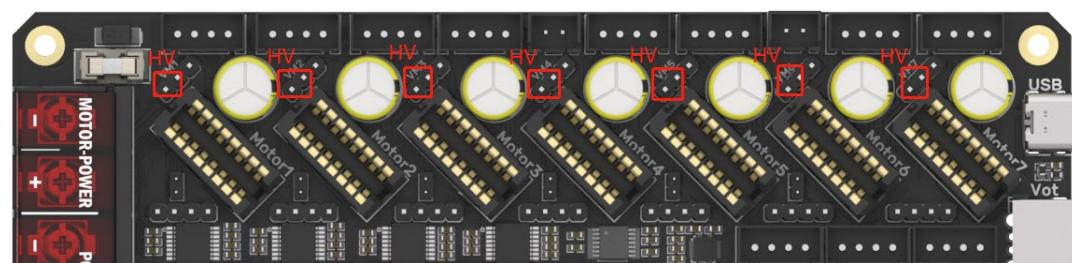
Set in the firmware, no need for a jumper.

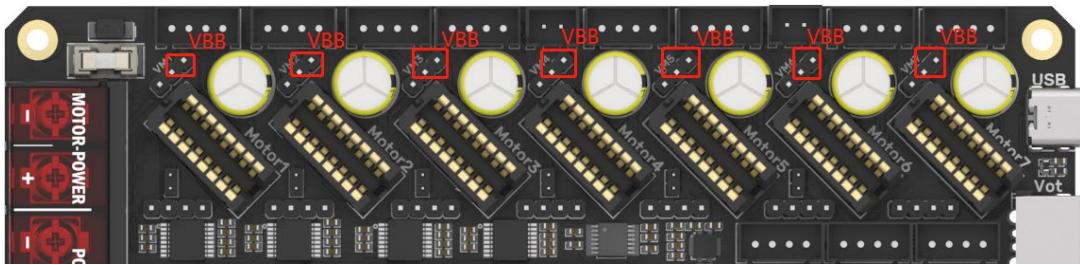
TMC Driver DIAG (Sensorless Homing)

When using sensorless homing, place jumpers according to the diagram below, there is no need to cut the DIAG pin off when not being used. (Motor1-Motor6).



Driver Voltage Selection

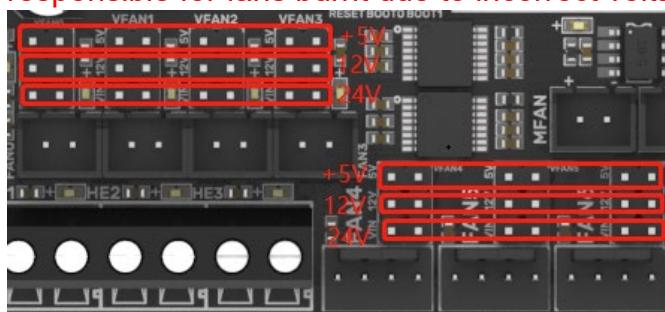




Voltage Selection for CNC Fan

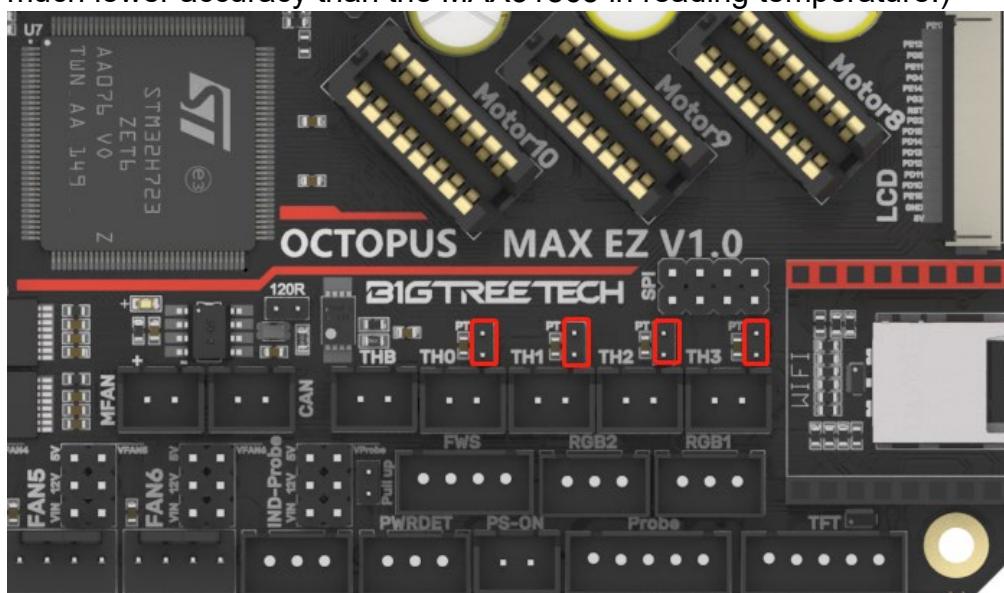
The output voltage can be set to 5V, 12V or 24V through a jumper. (**MFAN** and **FAN6** share the power supply **VFAN6**).

Note: Verify the fan's rated voltage before selecting to avoid damage. We are not responsible for fans burnt due to incorrect voltage selection.

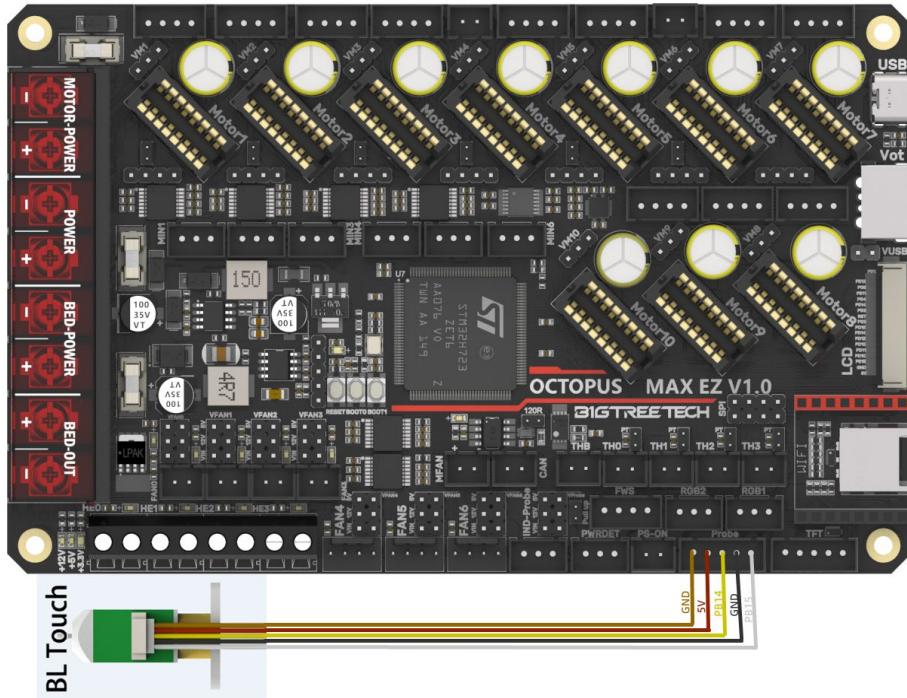


100K NTC or PT1000 Setting

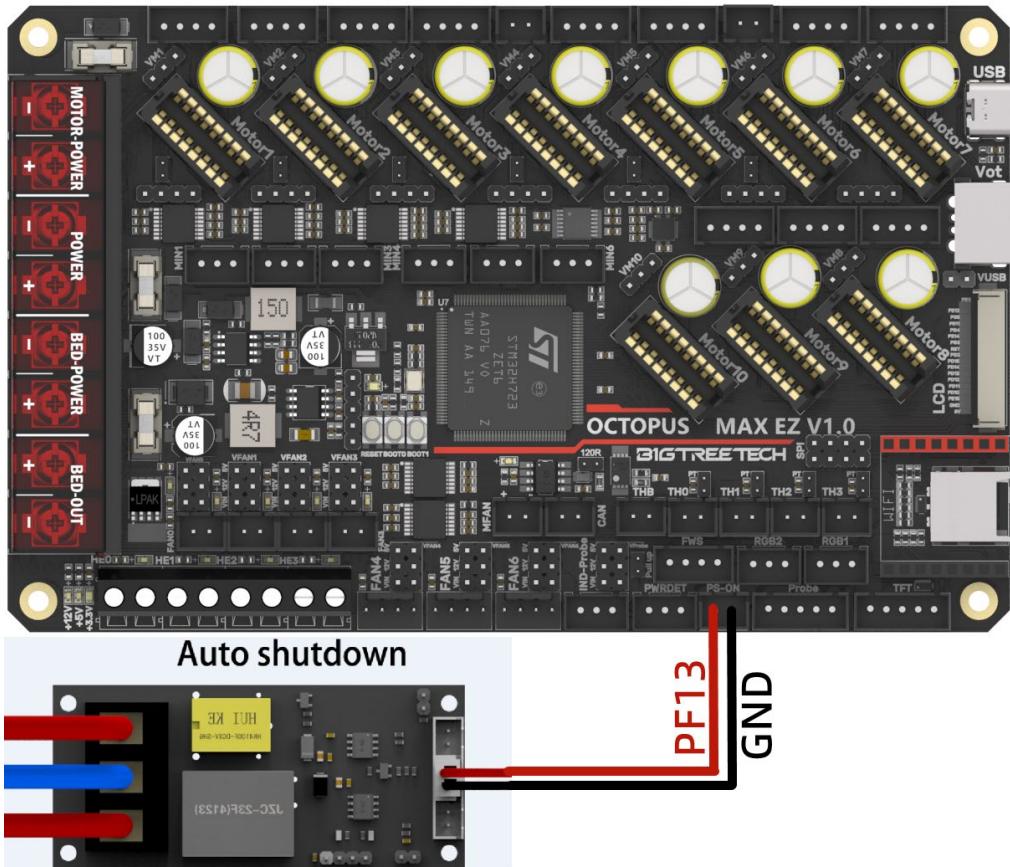
When using 100K NTC, no jumpers need to be connected, the pull-up resistance of TH0-TH3 is 4.7K 0.1%. When using PT1000, the pins indicated in the picture below need to be connected via jumpers, parallel connection of 4.12K 0.1% resistors, the pull-up resistance of TH0-TH1 is 2.2K. (**Note:** this method has a much lower accuracy than the MAX31865 in reading temperature.)



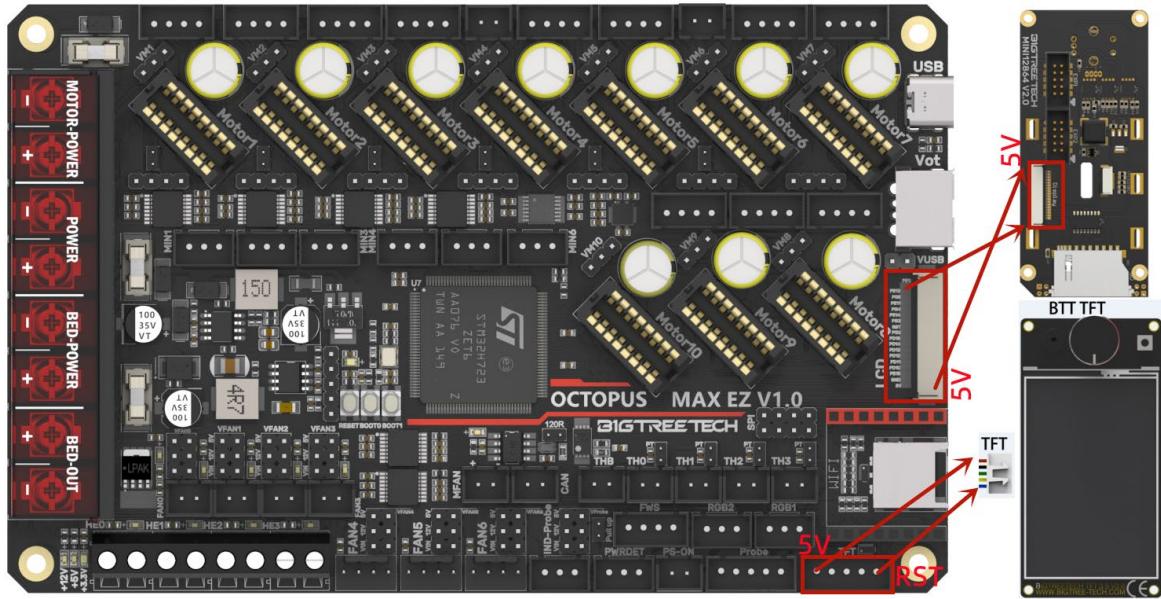
BLTouch Wiring



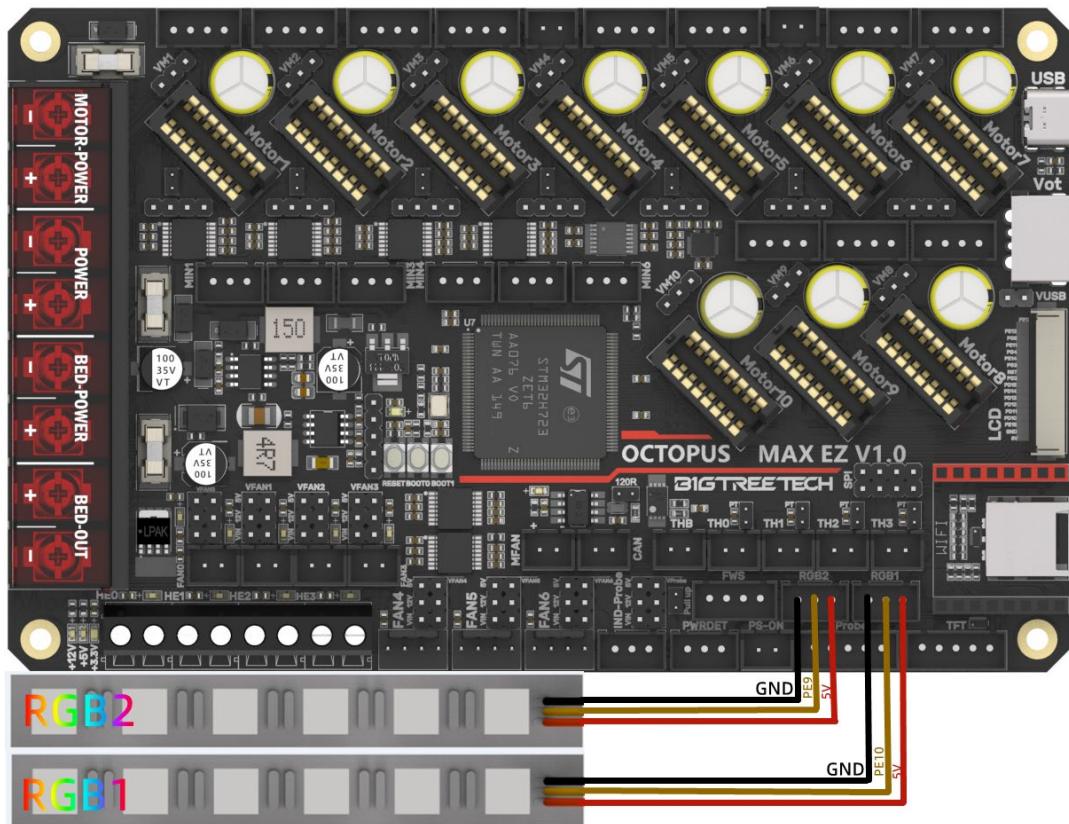
Auto Power Off (Relay V1.2) Wiring



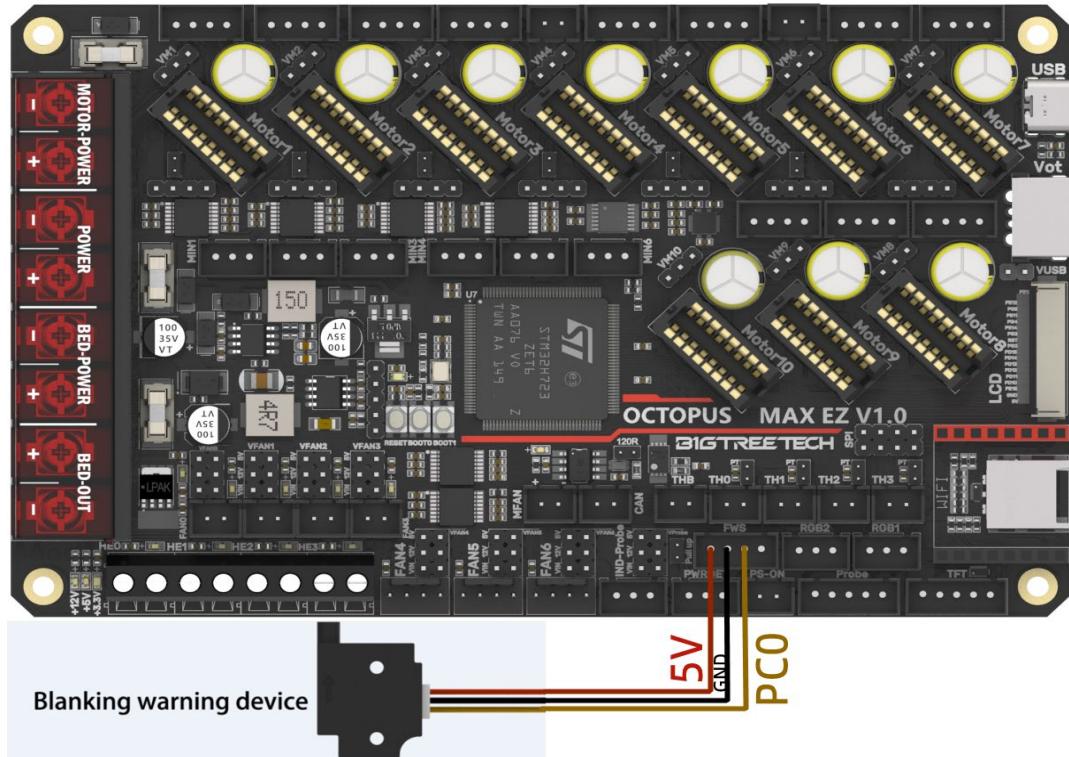
Connecting with MINI12864/TFT Screen



RGB Wiring

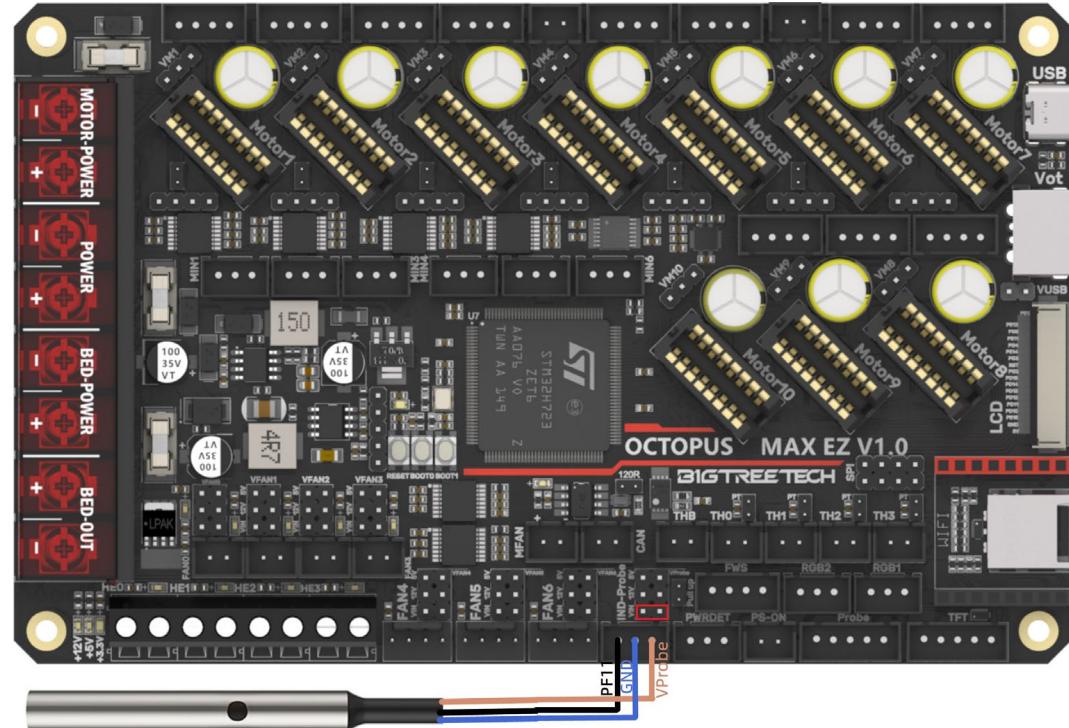


Filament Sensor Wiring

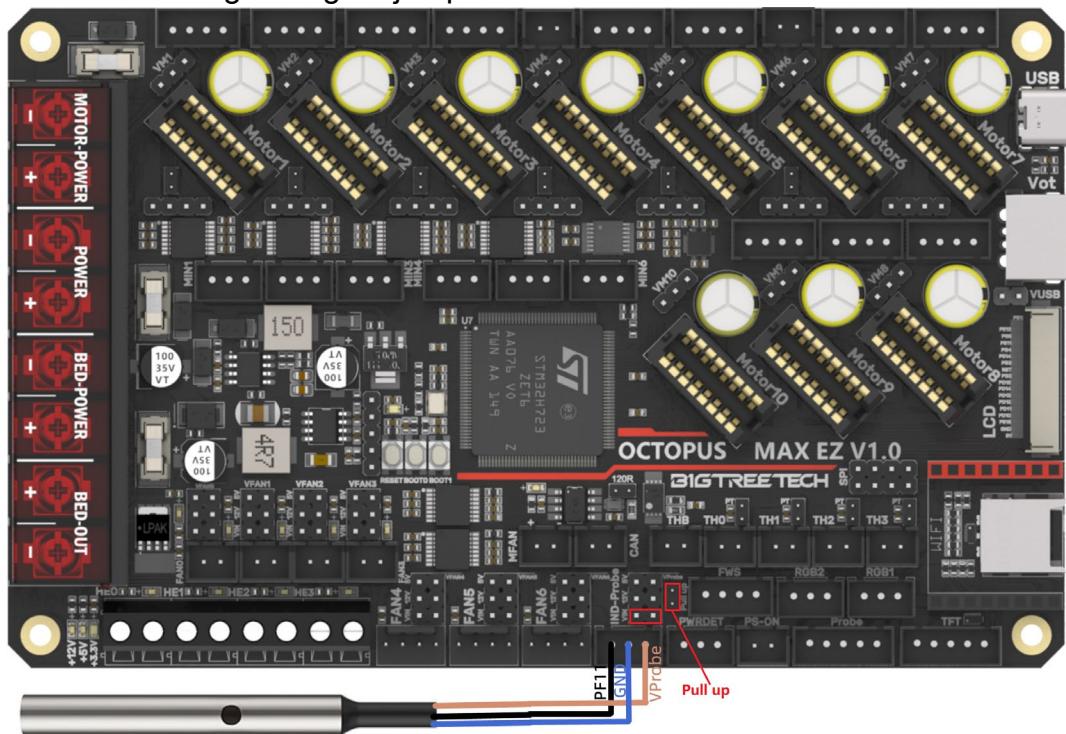


Proximity Switch Wiring

As shown in the figure below, 24V as an example, normally open (NPN type), no need for shorting through a jumper:

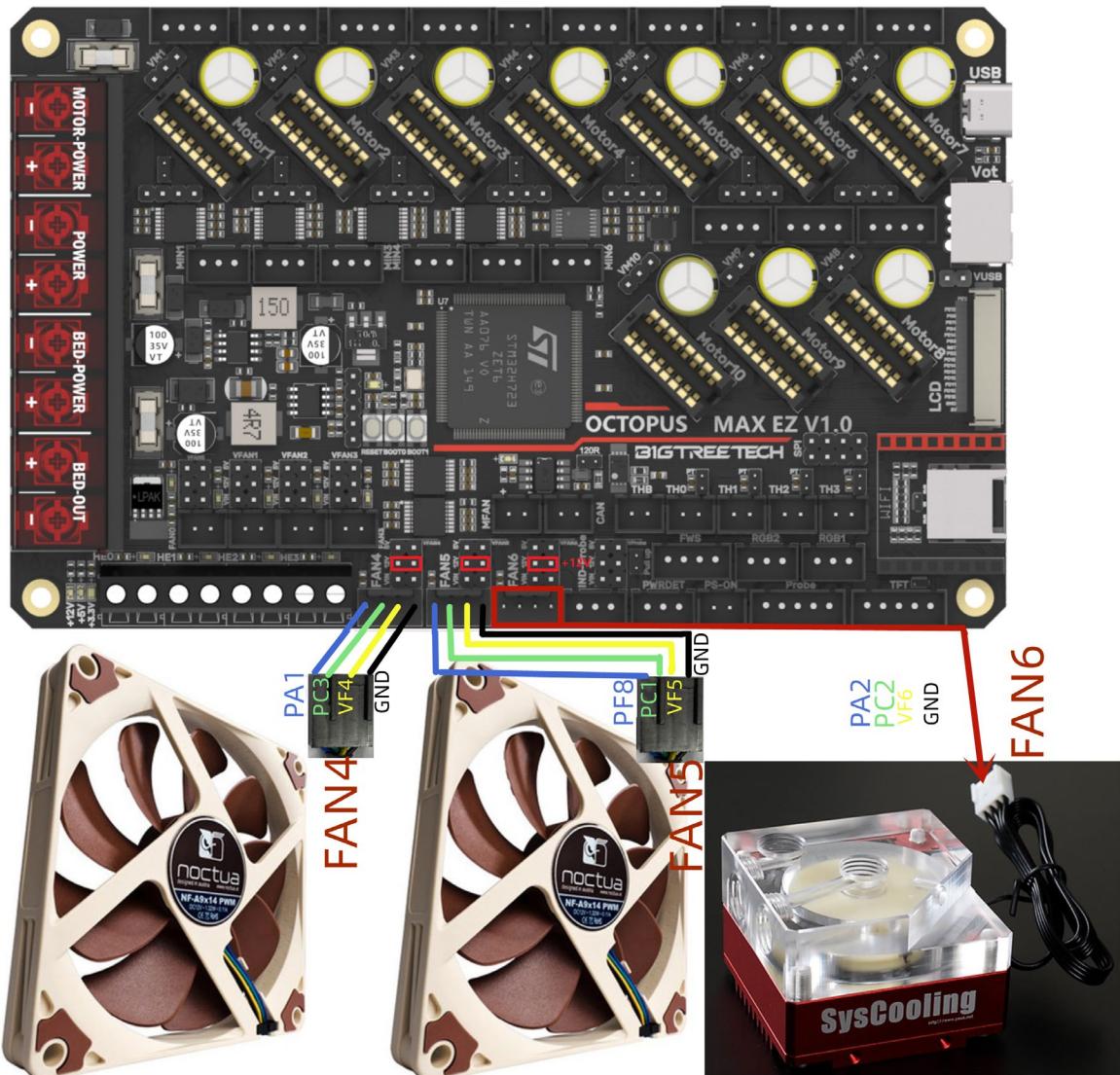


As shown in the figure below, 24V as an example, normally closed (PNP type), need for shorting through a jumper.



Wiring of 4 pins CNC Fan and Water Cooling System

(12V as an example:)



Marlin

Install Compiling Environment

<https://github.com/bigtreeetech/Document/blob/master/How%20to%20install%20VScode%2BPlatformio.md>

https://marlinfw.org/docs/basics/install_platformio_vscode.html

Refer to the link above for tutorial on installing VSCode and PlatformIO plugin.

Download Marlin Firmware

1. Download the newest bugfix version of Marlin from the official website:
<https://github.com/MarlinFirmware/Marlin/tree/bugfix-2.0.x>
2. Download pre-configured firmware from our GitHub page:
<https://github.com/bigtreeetech/BIGTREETECH-OCTOPUS-Max-EZ>

Configure Firmware

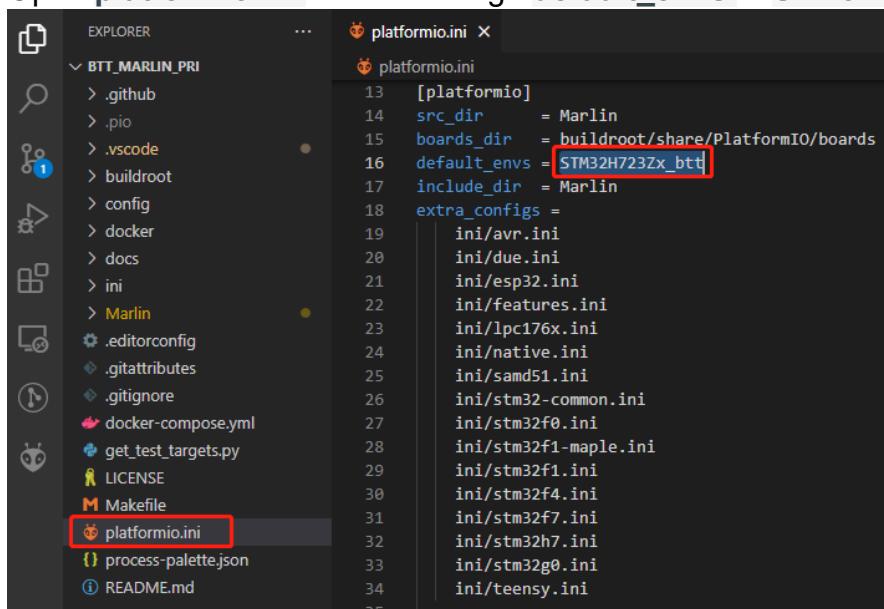
Open Marlin Project

You can open Marlin in VS Code in one of several ways:

- Drag the downloaded Marlin Firmware folder onto the VScode application icon;
- Use the **Open...** command in the VSCode **File** menu;
- Open the PIO Home tab and click the **Open Project** button.

Compiling Environment

Open **platformio.ini** file and change **default_envs** to **STM32H723Zx_btt**.

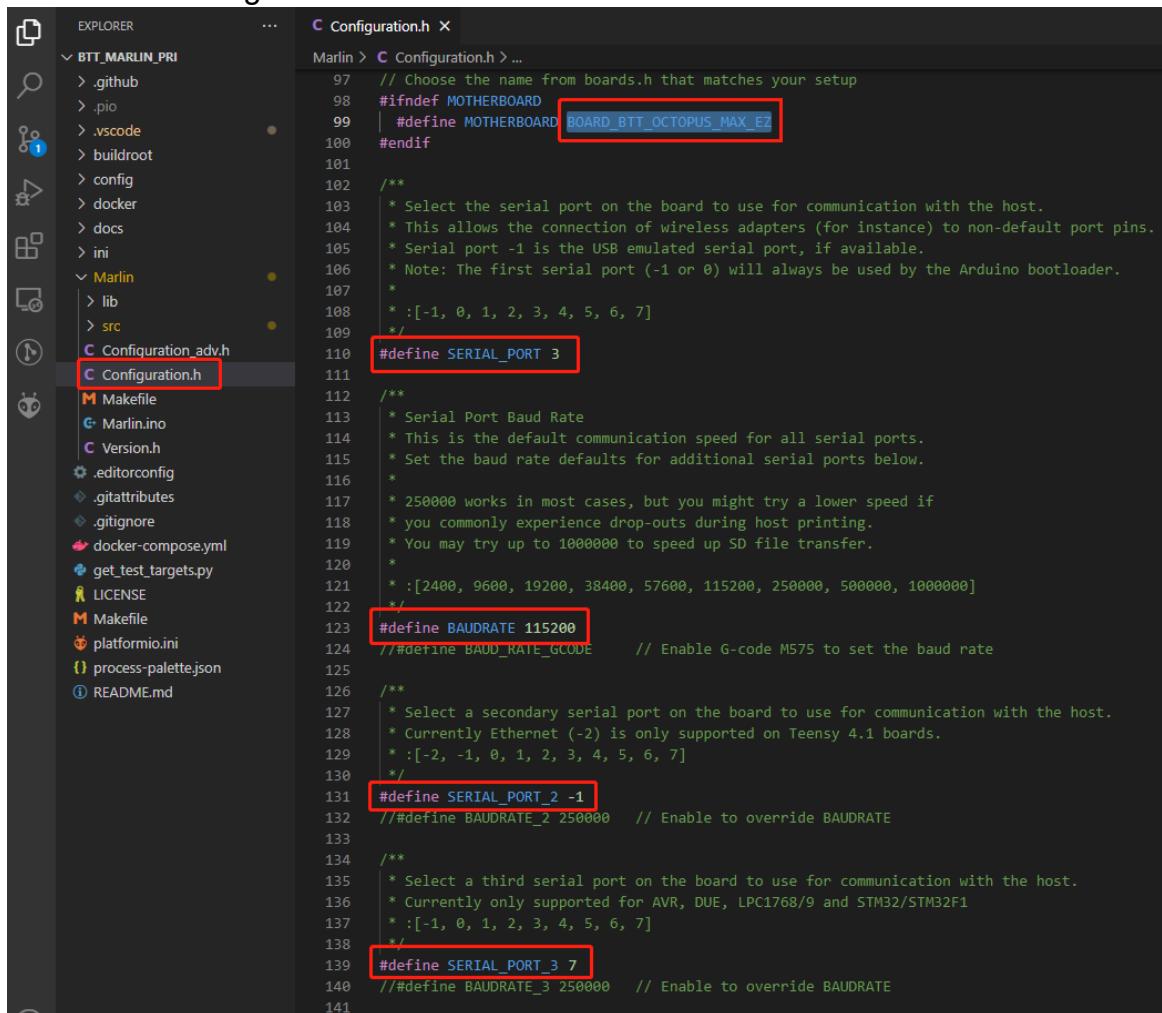


Configure Motherboard and Serial Port

Set **MOTHERBOARD** to **BOARD_BTT_OCTOPUS_MAX_EZ**

```
#define MOTHERBOARD BOARD_BTT_OCTOPUS_MAX_EZ
#define SERIAL_PORT 3    (enable TFT serial port)
#define BAUDRATE 115200  (set baudrate to the same as the communication
device)
#define SERIAL_PORT_2 -1 (enable USB serial port)
#define SERIAL_PORT_3 7  (enable WIFI serial port)
```

The above settings can be enabled as needed.

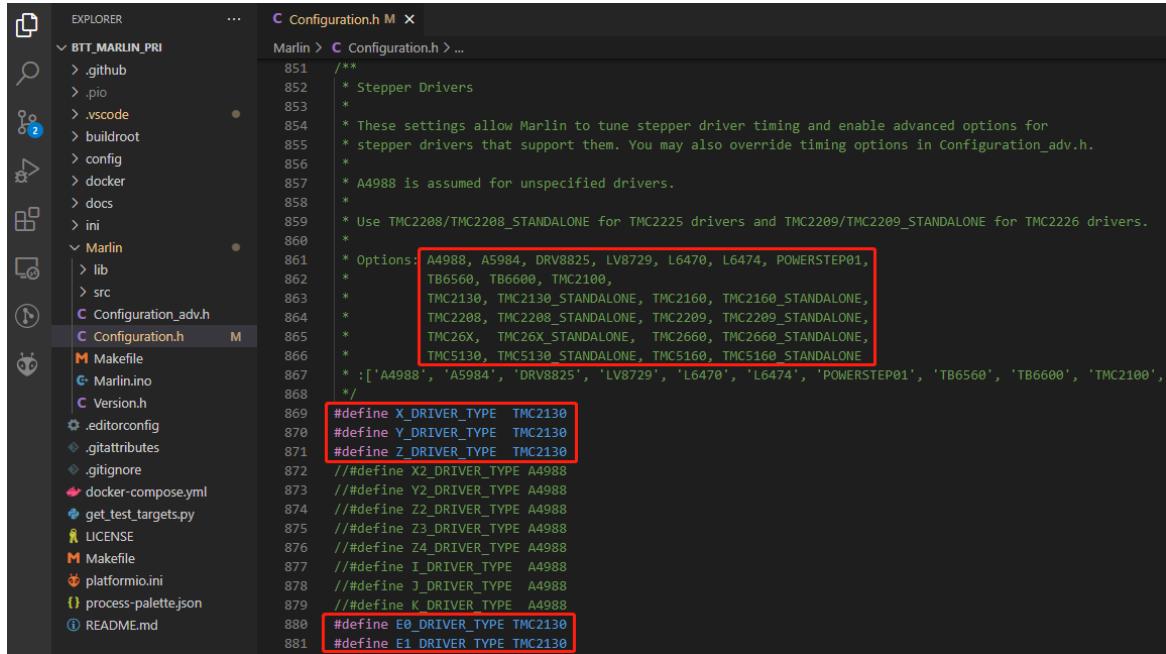


The screenshot shows the PlatformIO IDE interface with the 'Configuration.h' file open in the code editor. The left sidebar displays the project structure, including 'BTT.MARLIN_PRI' and 'Marlin' subfolders containing files like 'Configuration_adv.h', 'Configuration.h', 'Makefile', and 'Marlin.ino'. The main code editor window shows the 'Configuration.h' file with several define statements highlighted by red boxes:

- #define MOTHERBOARD BOARD_BTT_OCTOPUS_MAX_EZ
- #define SERIAL_PORT 3
- #define BAUDRATE 115200
- #define SERIAL_PORT_2 -1
- #define SERIAL_PORT_3 7

The code also includes comments explaining the purpose of each setting, such as selecting the serial port and baud rate.

Configure Stepper Driver



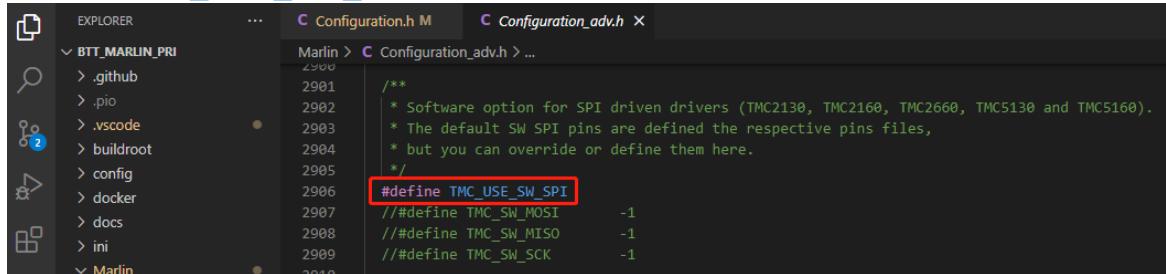
```

EXPLORER    C Configuration.h M ...
BTT_MARLIN_PRI   Marlin > C Configuration.h > ...
>.github      851  /**
>.pio          852  * Stepper Drivers
>.vscode      853  *
>buildroot    854  * These settings allow Marlin to tune stepper driver timing and enable advanced options for
>config       855  * stepper drivers that support them. You may also override timing options in Configuration_adv.h.
>docker       856  *
>docs         857  * A4988 is assumed for unspecified drivers.
>ini          858  *
859  * Use TMC2208/TMC2208_STANDALONE for TMC2225 drivers and TMC2209/TMC2209_STANDALONE for TMC2226 drivers.
860  *
861  * Options: A4988, A5984, DRV8825, LV8729, L6470, L6474, POWERSTEP01,
862  *             TB6560, TB6600, TMC2100,
863  *             TMC2130, TMC2130_STANDALONE, TMC2160, TMC2160_STANDALONE,
864  *             TMC2208, TMC2208_STANDALONE, TMC2209, TMC2209_STANDALONE,
865  *             TMC26X, TMC26X_STANDALONE, TMC2660, TMC2660_STANDALONE,
866  *             TMC5130, TMC5130_STANDALONE, TMC5160, TMC5160_STANDALONE
867  *             :['A4988', 'A5984', 'DRV8825', 'LV8729', 'L6470', 'L6474', 'POWERSTEP01', 'TB6560', 'TB6600', 'TMC2100',
868  *             ]
869  */
870  #define X_DRIVER_TYPE TMC2130
871  #define Y_DRIVER_TYPE TMC2130
872  #define Z_DRIVER_TYPE TMC2130
873  // #define X2_DRIVER_TYPE A4988
874  // #define Y2_DRIVER_TYPE A4988
875  // #define Z2_DRIVER_TYPE A4988
876  // #define Z3_DRIVER_TYPE A4988
877  // #define Z4_DRIVER_TYPE A4988
878  // #define I_DRIVER_TYPE A4988
879  // #define K_DRIVER_TYPE A4988
880  // #define E0_DRIVER_TYPE TMC2130
881  // #define E1_DRIVER_TYPE TMC2130

```

When using SPI mode, you need to enable `TMC_USE_SW_SPI` in `Configuration_adv.h`

`#define TMC_USE_SW_SPI`



```

EXPLORER    C Configuration.h M C Configuration_adv.h X ...
BTT_MARLIN_PRI   Marlin > C Configuration_adv.h > ...
>.github      2901  /**
>.pio          2902  * Software option for SPI driven drivers (TMC2130, TMC2160, TMC2660, TMC5130 and TMC5160).
>.vscode      2903  * The default SW SPI pins are defined the respective pins files,
>buildroot    2904  * but you can override or define them here.
>config       2905  */
2906  #define TMC_USE_SW_SPI
2907  // #define TMC_SW_MOSI      -1
2908  // #define TMC_SW_MISO      -1
2909  // #define TMC_SW_SCK       -1
2910

```

Sensorless Homing

```

EXPLORER          Configuration.h M   Configuration_adv.h M X
BTT_MARLIN_PRI
  .github
  .pio
  .vscode
  buildroot
  config
  docker
  docs
  ini
  Marlin
    lib
    src
  Configuration_adv.h M
  Configuration.h M
  Makefile
  Marlin.ino
  Version.h
  .editorconfig
  .gitattributes
  .gitignore
  docker-compose.yml
  get_test_targets.py
  LICENSE
  Makefile
  platformio.ini
  process-palette.json
  README.md

Marlin > Configuration_adv.h > ...
3047 /**
3048 * Use StallGuard to home / probe X, Y, Z.
3049 *
3050 * TMC2130, TMC2160, TMC2209, TMC2660, TMC5130, and TMC5160 only
3051 * Connect the stepper driver's DIAG1 pin to the X/Y endstop pin.
3052 * X, Y, and Z homing will always be done in spreadCycle mode.
3053 *
3054 * X/Y/Z_STALL_SENSITIVITY is the default stall threshold.
3055 * Use M914 X Y Z to set the stall threshold at runtime:
3056 *
3057 * Sensitivity TMC2209 Others
3058 * HIGHEST      255     -64 (Too sensitive => False positive)
3059 * LOWEST       0       63 (Too insensitive => No trigger)
3060 *
3061 * It is recommended to set HOMING_BUMP_MM to { 0, 0, 0 }.
3062 *
3063 * SPI_ENDSTOPS *** Beta feature! *** TMC2130/TMC5160 Only ***
3064 * Poll the driver through SPI to determine load when homing.
3065 * Removes the need for a wire from DIAG1 to an endstop pin.
3066 *
3067 * IMPROVE_HOMING_RELIABILITY tunes acceleration and jerk when
3068 * homing and adds a guard period for endstop triggering.
3069 *
3070 * Comment *_STALL_SENSITIVITY to disable sensorless homing for that axis.
3071 */
3072 #define SENSORLESS_HOMING // StallGuard capable drivers only
3073
3074 #if EITHER(SENSORLESS_HOMING, SENSORLESS_PROBING)
3075 // TMC2209: 0...255. TMC2130: -64...63
3076 #define X_STALL_SENSITIVITY 8
3077 #define X2_STALL_SENSITIVITY X_STALL_SENSITIVITY
3078 #define Y_STALL_SENSITIVITY 8
3079 #define Y2_STALL_SENSITIVITY Y_STALL_SENSITIVITY
3080 // #define Z_STALL_SENSITIVITY 8
3081 // #define Z2_STALL_SENSITIVITY Z_STALL_SENSITIVITY
3082 // #define Z3_STALL_SENSITIVITY Z_STALL_SENSITIVITY
3083 // #define Z4_STALL_SENSITIVITY Z_STALL_SENSITIVITY
3084 // #define I_STALL_SENSITIVITY 8
3085 // #define J_STALL_SENSITIVITY 8
3086 // #define K_STALL_SENSITIVITY 8
3087 // #define SPI_ENDSTOPS // TMC2130 only
3088 #define IMPROVE_HOMING_RELIABILITY
3089#endif

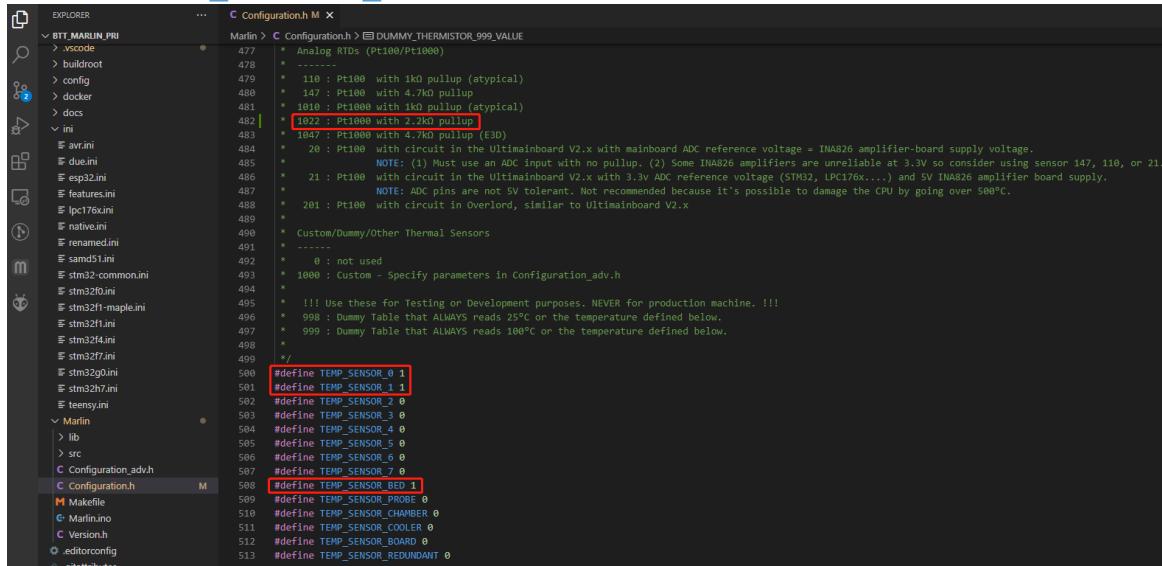
```

#define SENSORLESS_HOMING // enable sensorless homing
#define xx_STALL_SENSITIVITY 8 // sensitivity setting, TMC2209 range from 0 to 255, higher number results in more sensitive trigger threshold, sensitivity too high will cause endpoint to trigger before gantry actually moves to the end, lower number results in less sensitive trigger threshold, too low of sensitivity will cause endpoint to not trigger and gantrying continue. Other drivers range from 63 to -64, lower numbers result in a more sensitive trigger threshold.
#define IMPROVE_HOMING_RELIABILITY // can be used to set independent motor current for homing moves(xx_CURRENT_HOME) to improve homing reliability.

100K NTC or PT1000

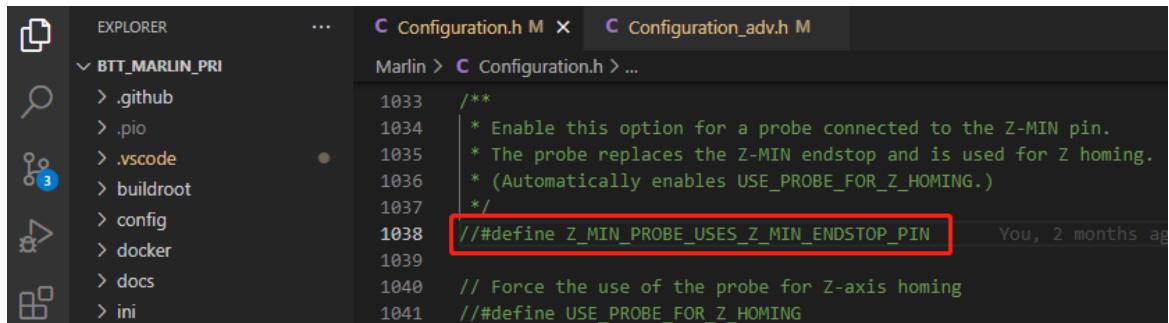
Use jumpers to set the thermistor pull-up resistor to 4.7K (with 100K NTC) or 2.2K (with PT1000). In Marlin firmware, 1 represents 100K NTC + 4.7K pullup, 1022 represents PT1000 + 2.2K pullup. **Note:** Accuracy will be much lower than MAX31865 with this method.

```
#define TEMP_SENSOR_0 1
#define TEMP_SENSOR_1 1
#define TEMP_SENSOR_BED 1
```



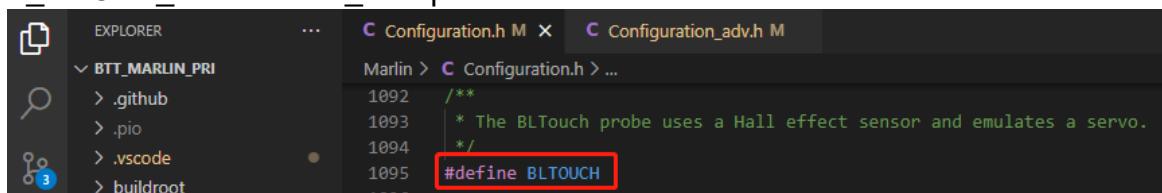
```
EXPLORER          C Configuration.h M × C Configuration_adv.h M
BTT_MARLIN_PRI   Marlin > C Configuration.h > ...
> .vscode
> buildroot
> config
> docker
> docs
> ini
> avr.ini
> due.ini
> esp32ini
> features.ini
> Ipc176xini
> native.ini
> renamedini
> sand51ini
> stm32-commonini
> stm32f0ini
> stm32f1ini
> stm32f1-mapleini
> stm32f1ini
> stm32f4ini
> teensyini
Marlin
> lib
> src
C Configuration.h M
C Configuration_adv.h M
Makefile
Marlin.ino
Version.h
.editorconfig
...
477  #define TEMP_SENSOR_0 1
478  #define TEMP_SENSOR_BED 1
479  // -----
480  // 110 : Pt100 with 1kΩ pullup (atypical)
481  // 147 : Pt100 with 4.7kΩ pullup
482  // 1610 : Pt1000 with 1kΩ pullup (atypical)
483  // 1622 : Pt1000 with 2.2kΩ pullup
484  // 1647 : Pt1000 with 4.7kΩ pullup (E3D)
485  // 20 : Pt100 with circuit in the Ultimainboard V2.x with mainboard ADC reference voltage = INA826 amplifier-board supply voltage.
486  // NOTE: (1) Must use an ADC input with no pullup. (2) Some INA826 amplifiers are unreliable at 3.3V so consider using sensor 147, 110, or 21.
487  // 21 : Pt100 with circuit in the Ultimainboard V2.x with 3.3V ADC reference voltage (STM32, LPC176x,...) and 5V INA826 amplifier board supply.
488  // NOTE: ADC pins are not 5V tolerant. Not recommended because it's possible to damage the CPU by going over 500°C.
489  // 201 : Pt100 with circuit in Overlord, similar to Ultimainboard V2.x
490  // Custom/Dummy/Other Thermal Sensors
491  // -----
492  // 0 : not used
493  // 1000 : Custom - Specify parameters in Configuration_adv.h
494  // !!! Use these for Testing or Development purposes. NEVER for production machine. !!!
495  // 998 : Dummy Table that ALWAYS reads 25°C or the temperature defined below.
496  // 999 : Dummy Table that ALWAYS reads 100°C or the temperature defined below.
497  //
498  //
499  // -----
500  // #define TEMP_SENSOR_0 1
501  // #define TEMP_SENSOR_1 1
502  // #define TEMP_SENSOR_2 0
503  // #define TEMP_SENSOR_3 0
504  // #define TEMP_SENSOR_4 0
505  // #define TEMP_SENSOR_5 0
506  // #define TEMP_SENSOR_6 0
507  // #define TEMP_SENSOR_7 0
508  // #define TEMP_SENSOR_BED 1
509  // #define TEMP_SENSOR_PROBE 0
510  // #define TEMP_SENSOR_CHAMBER 0
511  // #define TEMP_SENSOR_COOLER 0
512  // #define TEMP_SENSOR_BOARD 0
513  // #define TEMP_SENSOR_REDUNDANT 0
```

BLTouch



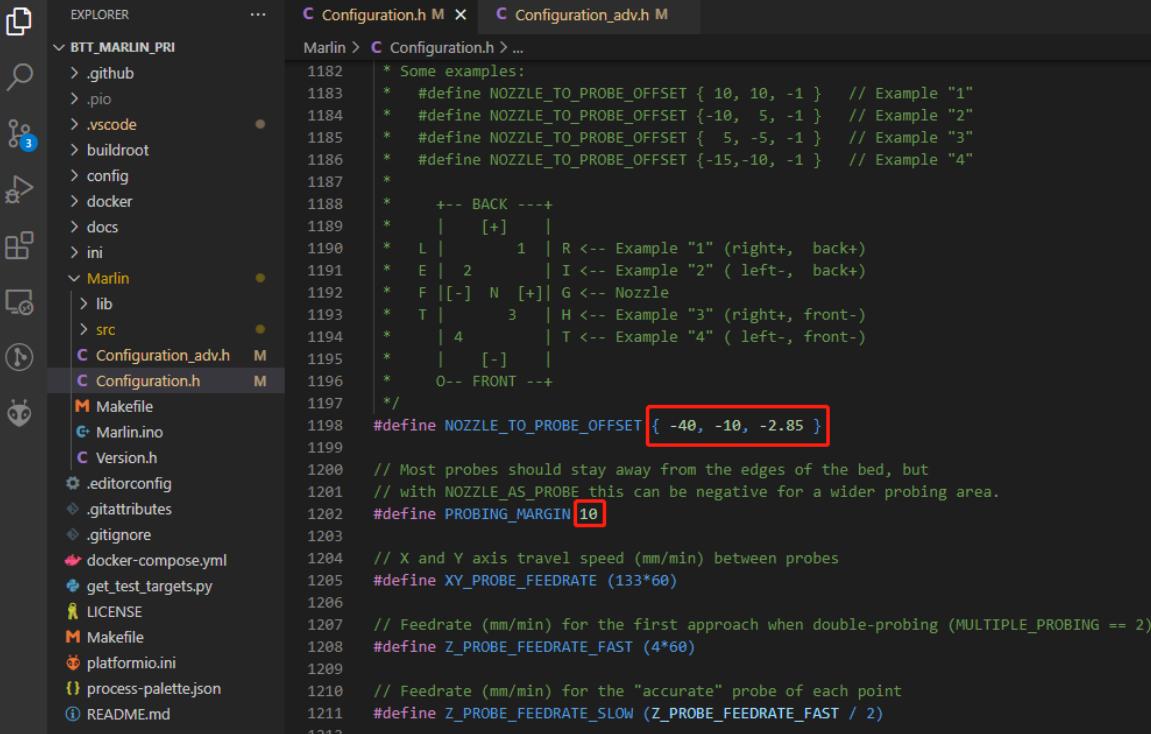
```
EXPLORER          C Configuration.h M × C Configuration_adv.h M
BTT_MARLIN_PRI   Marlin > C Configuration.h > ...
> .github
> .pio
> .vscode
> buildroot
> config
> docker
> docs
> ini
1033  /**
1034  * Enable this option for a probe connected to the Z-MIN pin.
1035  * The probe replaces the Z-MIN endstop and is used for Z homing.
1036  * (Automatically enables USE_PROBE_FOR_Z_HOMING.)
1037  */
1038  //##define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN
1039
1040  // Force the use of the probe for Z-axis homing
1041  //##define USE_PROBE_FOR_Z_HOMING
```

```
//##define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN // Do not remap the
Z_PROBE_PIN to the Z_MIN port.
```



```
EXPLORER          C Configuration.h M × C Configuration_adv.h M
BTT_MARLIN_PRI   Marlin > C Configuration.h > ...
> .github
> .pio
> .vscode
> buildroot
1092  /**
1093  * The BLTouch probe uses a Hall effect sensor and emulates a servo.
1094  */
1095  #define BLTOUCH
1096
```

```
#define BLTOUCH // Enable BLTouch
```

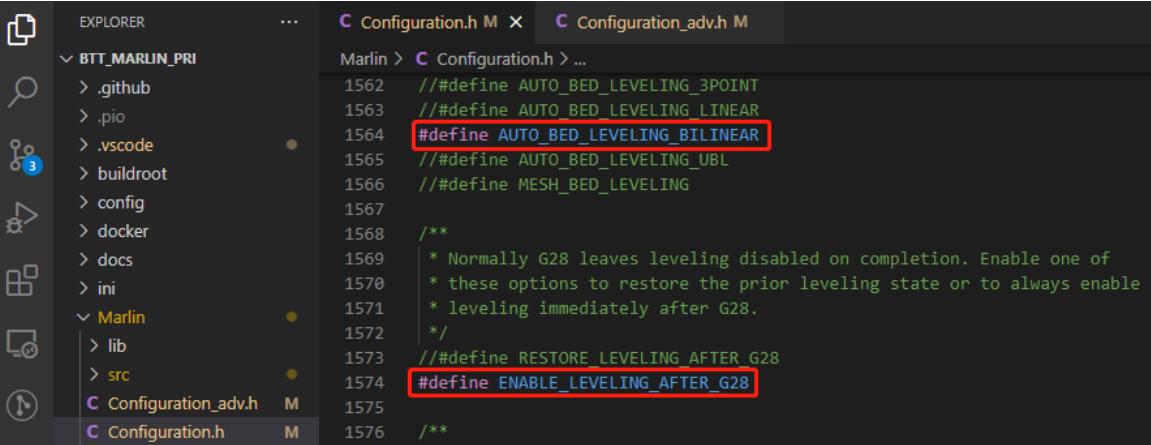


```

EXPLORER ... C Configuration.h M × C Configuration_adv.h M
Marlin > C Configuration.h ...
1182 * Some examples:
1183 * #define NOZZLE_TO_PROBE_OFFSET { 10, 10, -1 } // Example "1"
1184 * #define NOZZLE_TO_PROBE_OFFSET { -10, 5, -1 } // Example "2"
1185 * #define NOZZLE_TO_PROBE_OFFSET { 5, -5, -1 } // Example "3"
1186 * #define NOZZLE_TO_PROBE_OFFSET { -15,-10, -1 } // Example "4"
1187 *
1188 *     +-- BACK ---+
1189 *     | [+] |
1190 *     L | 1 | R <-- Example "1" (right+, back+)
1191 *     E | 2 | I <-- Example "2" (left-, back+)
1192 *     F [-] N [+] G <-- Nozzle
1193 *     T | 3 | H <-- Example "3" (right+, front-)
1194 *     | 4 | T <-- Example "4" (left-, front-)
1195 *     | [-] |
1196 *     O-- FRONT --+
1197 */
1198 #define NOZZLE_TO_PROBE_OFFSET { -40, -10, -2.85 }
1199
1200 // Most probes should stay away from the edges of the bed, but
1201 // with NOZZLE_AS_PROBE this can be negative for a wider probing area.
1202 #define PROBING_MARGIN 10
1203
1204 // X and Y axis travel speed (mm/min) between probes
1205 #define XY_PROBE_FEEDRATE (133*60)
1206
1207 // Feedrate (mm/min) for the first approach when double-probing (MULTIPLE_PROBING == 2)
1208 #define Z_PROBE_FEEDRATE_FAST (4*60)
1209
1210 // Feedrate (mm/min) for the "accurate" probe of each point
1211 #define Z_PROBE_FEEDRATE_SLOW (Z_PROBE_FEEDRATE_FAST / 2)
1212

```

#define NOZZLE_TO_PROBE_OFFSET { -40, -10, -2.85 } // set BLTouch probe offset
#define PROBING_MARGIN 10 // set distance between probe area and print area perimeter



```

EXPLORER ... C Configuration.h M × C Configuration_adv.h M
Marlin > C Configuration.h ...
1562 // #define AUTO_BED_LEVELLING_3POINT
1563 // #define AUTO_BED_LEVELLING_LINEAR
1564 #define AUTO_BED_LEVELLING_BILINEAR
1565 // #define AUTO_BED_LEVELLING_UBL
1566 // #define MESH_BED_LEVELLING
1567
1568 /**
1569 * Normally G28 leaves leveling disabled on completion. Enable one of
1570 * these options to restore the prior leveling state or to always enable
1571 * leveling immediately after G28.
1572 */
1573 // #define RESTORE_LEVELLING_AFTER_G28
1574 #define ENABLE_LEVELLING_AFTER_G28
1575
1576 /**

```

#define AUTO_BED_LEVELLING_BILINEAR // set probe pattern
#define RESTORE_LEVELLING_AFTER_G28 // apply leveling after G28 homing command

```

EXPLORER          C Configuration.h M X  C Configuration_adv.h M
Marlin > C Configuration.h > ...
1628 #if EITHER(AUTO_BED_LEVELING_LINEAR, AUTO_BED_LEVELING_BILINEAR)
1629
1630 // Set the number of grid points per dimension.
1631 #define GRID_MAX_POINTS_X 5
1632 #define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X
1633
1634 // Probe along the Y axis, advancing X after each column
1635 //##define PROBE_Y_FIRST
1636
1637 #if ENABLED(AUTO_BED_LEVELING_BILINEAR)
1638
1639 // Beyond the probed grid, continue the implied tilt?
1640 // Default is to maintain the height of the nearest edge.
1641 //##define EXTRAPOLATE_BEYOND_GRID
1642

```

`#define GRID_MAX_POINTS_X 5` // set number of probe points for X axis,
usually 5 point is sufficient

`#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X` // set the number of
probe points for Y axis to the same as X axis.

If BLTouch also functions as your Z homing sensor, no wiring change is needed,
just set it in the firmware.

```

EXPLORER          C Configuration.h M X  C Configuration_adv.h M
Marlin > C Configuration.h > ...
1033 /**
1034 * Enable this option for a probe connected to the Z-MIN pin.
1035 * The probe replaces the Z-MIN endstop and is used for Z homing.
1036 * (Automatically enables USE_PROBE_FOR_Z_HOMING.)
1037 */
1038 //##define Z_MIN_PROBEUSES_Z_MIN_ENDSTOP_PIN
1039
1040 // Force the use of the probe for Z-axis homing
1041 #define USE_PROBE_FOR_Z_HOMING
1042

```

`#define USE_PROBE_FOR_Z_HOMING` // use Z Probe(BLTouch) for Z homing

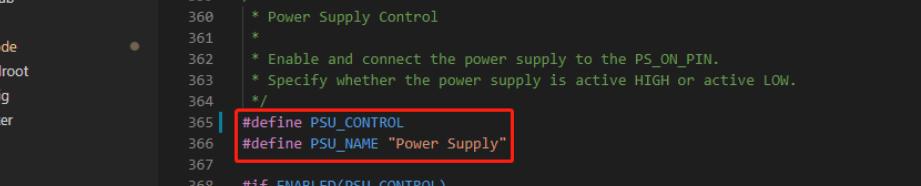
```

EXPLORER          C Configuration.h M X  C Configuration_adv.h M
Marlin > C Configuration.h > ...
1758 /**
1759 * Use "Z Safe Homing" to avoid homing with a Z probe outside the bed area.
1760 *
1761 * - Moves the Z probe (or nozzle) to a defined XY point before Z homing.
1762 * - Allows Z homing only when XY positions are known and trusted.
1763 * - If stepper drivers sleep, XY homing may be required again before Z homing.
1764 */
1765 #define Z_SAFE_HOMING
1766
1767 #if ENABLED(Z_SAFE_HOMING)
1768 #define Z_SAFE_HOMING_X_POINT X_CENTER // X point for Z homing
1769 #define Z_SAFE_HOMING_Y_POINT Y_CENTER // Y point for Z homing
1770#endif

```

`#define Z_SAFE_HOMING` // home Z at the center of print bed to prevent probing
outside of the print bed.

Auto Power Off(Relay V1.2)



The screenshot shows the VS Code interface with the 'EXPLORER' view on the left and the 'CODE' view on the right. The 'CODE' view displays the 'Configuration.h' file under the 'Marlin' folder. A red box highlights the line '#define PSU_ACTIVE_STATE HIGH'. The file contains code related to power supply control, including definitions for PSU_CONTROL and PSU_NAME, and a conditional block for PSU_ACTIVE_STATE.

```
359 /**
360 * Power Supply Control
361 *
362 * Enable and connect the power supply to the PS_ON_PIN.
363 * Specify whether the power supply is active HIGH or active LOW.
364 */
365 #define PSU_CONTROL
366 #define PSU_NAME "Power Supply"
367
368 #if ENABLED(PSU_CONTROL)
369     //##define MKS_PWC
370     //##define PS_OFF_CONFIRM
371     //##define PS_OFF_SOUND
372     #define PSU_ACTIVE_STATE HIGH
373
374     //##define PSU_DEFAULT_OFF
375     //##define PSU_POWERUP_DELAY 250
376
377     //##define POWER_OFF_TIMER
378     //##define POWER_WAIT_FOR_COOLDOWN
```

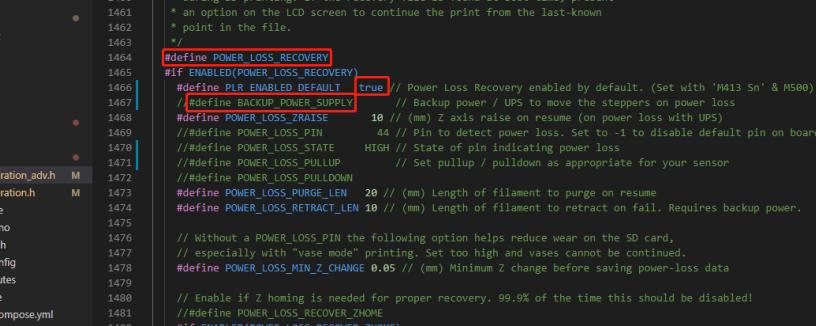
```
#define PSU_CONTROL // enable PSU control to turn on and off using M80 and  
M81
```

```
#define PSU_ACTIVE_STATE HIGH // set turn on level, Relay V1.2 is turned on  
with high level and turned off with low level, so this setting needs to be HIGH.
```

Power Loss Recovery

There are two methods for power loss recovery

1. No extra module needed, the motherboard will write current print status to the SD card after every layer is printed, which shortens the life of the SD card severely.



The screenshot shows the Marlin configuration editor interface. The left sidebar lists project components like BTM_MARLIN_PRJ, Configuration.h, Configuration_adv.h, and Configuration_adv.mk. The main area displays the Configuration.h file with several code snippets highlighted in red:

```
* Store the current state to the SD Card at the start of each layer
* during SD printing. If the recovery file is found at boot time, present
* an option on the LCD screen to continue the print from the last-known
* point in the file.

#define POWER_LOSS_RECOVERY
#if ENABLED(POWER_LOSS_RECOVERY)
#define PLR_ENABLED_DEFAULT true // Power Loss Recovery enabled by default. (Set with 'M413 Sn' & M500)
#define BACKUP_POWER_SUPPLY // Backup power / UPS to move the steppers on power loss
#define POWER_LOSS_RAISE 10 // (mm) Z axis raise on resume (on power loss with UPS)
#define POWER_LOSS_PIN 44 // Pin to detect power loss. Set to -1 to disable default pin on boards without module.
#define POWER_LOSS_STATE HIGH // State of pin indicating power loss
#define POWER_LOSS_PULLUP // Set pullup / pulldown as appropriate for your sensor
#define POWER_LOSS_PULLDOWN

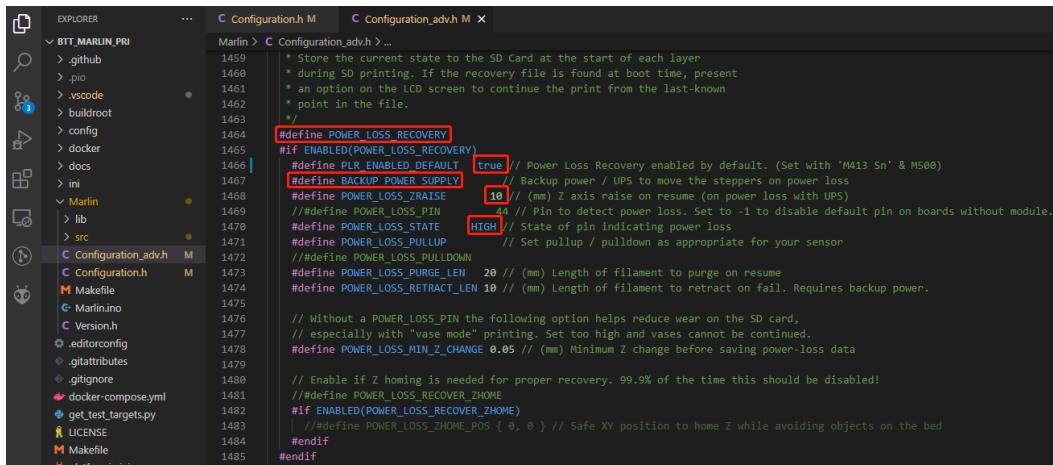
#define POWER_LOSS_PURGE_LEN 20 // (mm) Length of filament to purge on resume
#define POWER_LOSS_RETRACT_LEN 10 // (mm) Length of filament to retract on fail. Requires backup power.

// Without a POWER_LOSS_PIN the following option helps reduce wear on the SD card,
// especially with "vase mode" printing. Set too high and vases cannot be continued.
#define POWER_LOSS_MIN_Z_CHANGE 0.05 // (mm) Minimum Z change before saving power loss data

// Enable if Z homing is needed for proper recovery. 99.9% of the time this should be disabled!
//#define POWER_LOSS_RECOVER_ZHOME
#if ENABLED(POWER_LOSS_RECOVER_ZHOME)
#define POWER_LOSS_ZHOME_POS { 0, 0, 0 } // Safe XY position to home Z while avoiding objects on the bed
#endif
#endif
```

```
#define POWER_LOSS_RECOVERY // enable power loss recovery
#define PLR_ENABLED_DEFAULT true // true default to power loss
recovery enabled
```

- External UPS 24V V1.0 module, when power is cut, the module will provide power to the board and signal the board to save current print status to SD card. This method has virtually no effect on the life of the SD card.



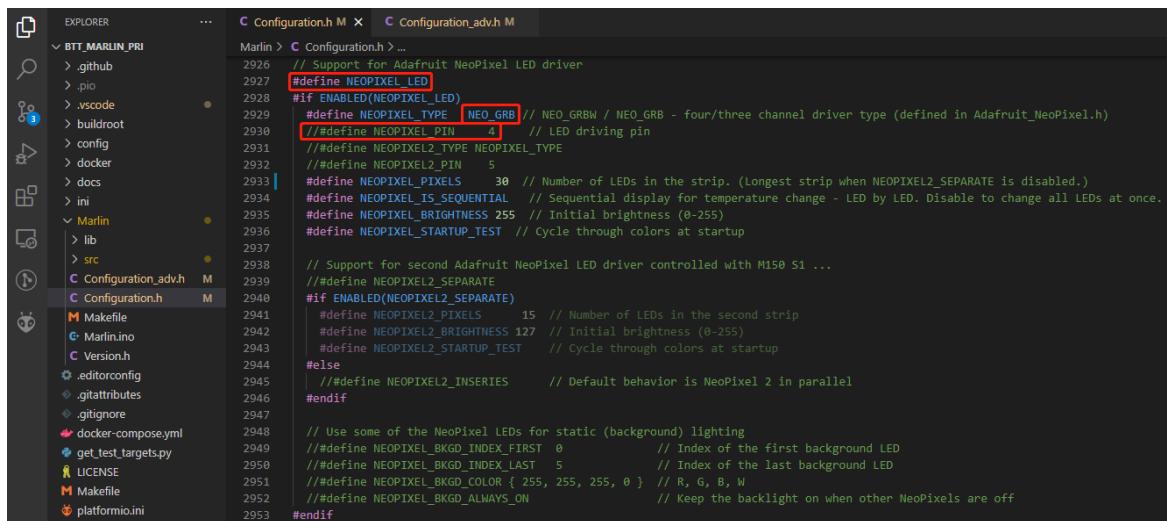
```

    #define POWER LOSS RECOVERY // enable power loss recovery
    #define PLR_ENABLED_DEFAULT true // true default to power loss recovery
    enabled
    #define POWER LOSS ZRAISE 10 // raise the print head by 10mm after
    power loss to prevent the nozzle from touching the printed part

    #define POWER LOSS STATE HIGH // set signal level, UPS 24V V1.0
    returns low level when not triggered and HIGH level when power is cut, thus
    this setting needs to be HIGH

```

RGB

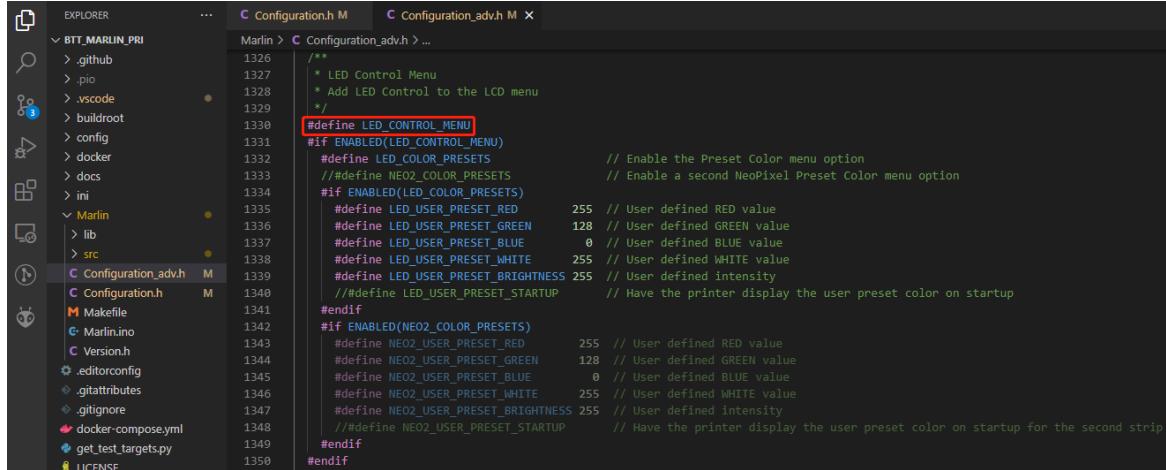


```

    #define NEOPIXEL LED // enable Neopixel
    #define NEOPIXEL_TYPE NEO_GRB // set Neopixel type
    //##define NEOPIXEL_PIN 4 // disable PIN setting, use the correct signal pin in
    the pin file of the motherboard
    #define NEOPIXEL_PIXELS 30 // number of LEDs
    #define NEOPIXEL_STARTUP_TEST // the light will show red green and blue
    sequentially to self-test

```

If you are using displays like LCD2004, 12864, mini12864, etc., you can also control RGB from your display directly.



```

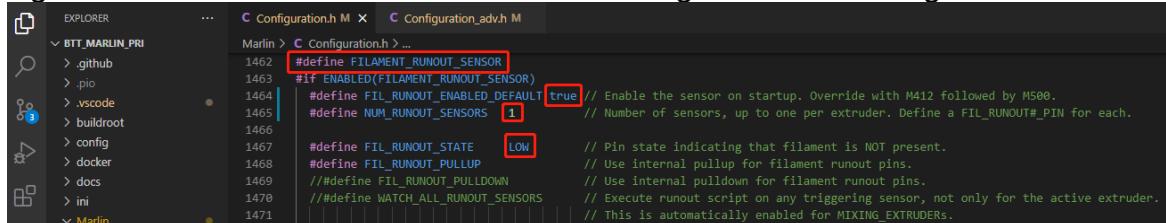
EXPLORER      C Configuration.h M  C Configuration_adv.h M ...
BTT_MARLIN_PRI
> .github
> .pio
> .vscode
> buildroot
> config
> docker
> docs
> ini
> Marlin
> lib
> src
C Configuration_adv.h M 1326  /**
C Configuration.h M 1327  * LED Control Menu
C Configuration_adv.h M 1328  * Add LED Control to the LCD menu
C Configuration.h M 1329  */
C Configuration_adv.h M 1330  #define LED_CONTROL_MENU
C Configuration.h M 1331  #if ENABLED(LED_CONTROL_MENU)
C Configuration_adv.h M 1332  #define LED_COLOR_PRESETS          // Enable the Preset Color menu option
C Configuration.h M 1333  // #define NEO2_COLOR_PRESETS          // Enable a second Neopixel Preset Color menu option
C Configuration_adv.h M 1334  #if ENABLED(LED_COLOR_PRESETS)
C Configuration.h M 1335  #define LED_USER_PRESET_RED        255 // User defined RED value
C Configuration_adv.h M 1336  #define LED_USER_PRESET_GREEN       128 // User defined GREEN value
C Configuration.h M 1337  #define LED_USER_PRESET_BLUE         0 // User defined BLUE value
C Configuration_adv.h M 1338  #define LED_USER_PRESET_WHITE       255 // User defined WHITE value
C Configuration.h M 1339  #define LED_USER_PRESET_BRIGHTNESS  255 // User defined intensity
C Configuration_adv.h M 1340  // #define LED_USER_PRESET_STARTUP   // Have the printer display the user preset color on startup
C Configuration.h M 1341  #endif
C Configuration_adv.h M 1342  #if ENABLED(NEO2_COLOR_PRESETS)
C Configuration.h M 1343  #define NEO2_USER_PRESET_RED        255 // User defined RED value
C Configuration_adv.h M 1344  #define NEO2_USER_PRESET_GREEN       128 // User defined GREEN value
C Configuration.h M 1345  #define NEO2_USER_PRESET_BLUE         0 // User defined BLUE value
C Configuration_adv.h M 1346  #define NEO2_USER_PRESET_WHITE       255 // User defined WHITE value
C Configuration.h M 1347  #define NEO2_USER_PRESET_BRIGHTNESS  255 // User defined intensity
C Configuration_adv.h M 1348  // #define NEO2_USER_PRESET_STARTUP   // Have the printer display the user preset color on startup for the second strip
C Configuration.h M 1349  #endiff
C Configuration_adv.h M 1350  #endiff

```

`#define LED_CONTROL_MENU // add led control to your menu.`

Filament Sensor

Standard filament run out sensors are usually comprised of a microswitch which signals the mainboard of filament status with High or Low level signal.



```

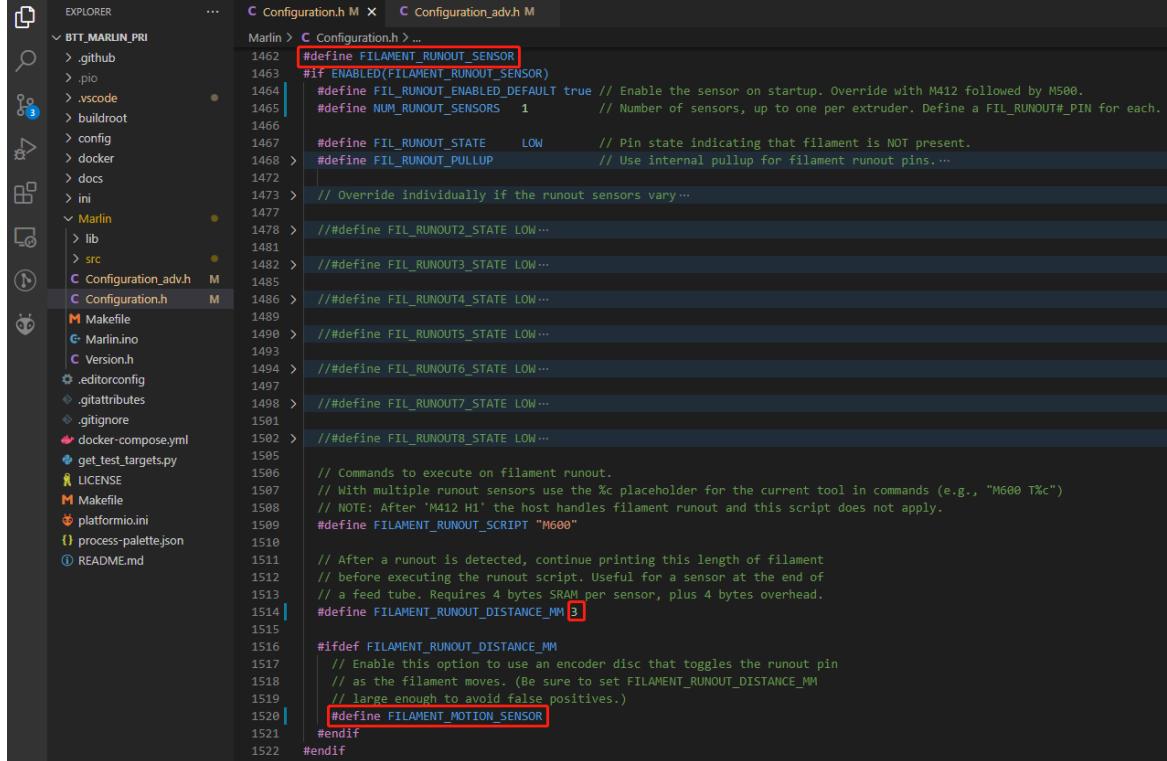
EXPLORER      C Configuration.h M X  C Configuration_adv.h M ...
BTT_MARLIN_PRI
> .github
> .pio
> .vscode
> buildroot
> config
> docker
> docs
> ini
> Marlin
Marlin > C Configuration.h ...
1462  #define FILAMENT_RUNOUT_SENSOR
1463  #if ENABLED(FILAMENT_RUNOUT_SENSOR)
1464  #define FIL_RUNOUT_ENABLED_DEFAULT true // Enable the sensor on startup. Override with M412 followed by M500.
1465  #define NUM_RUNOUT_SENSORS 1 // Number of sensors, up to one per extruder. Define a FIL_RUNOUT#_PIN for each.
1466
1467  #define FIL_RUNOUT_STATE LOW // Pin state indicating that filament is NOT present.
1468  #define FIL_RUNOUT_PULLUP // Use internal pullup for filament runout pins.
1469  // #define FIL_RUNOUT_PULLDOWN // Use internal pulldown for filament runout pins.
1470  // #define WATCH_ALL_RUNOUT_SENSORS // Execute runout script on any triggering sensor, not only for the active extruder.
1471  // This is automatically enabled for MIXING_EXTRUDERS.

```

`#define FILAMENT_RUNOUT_SENSOR // enable filament run out sensor`
`#define FIL_RUNOUT_ENABLED_DEFAULT true // true default to filament run out sensor enabled`
`#define NUM_RUNOUT_SENSORS 1 // number of filament run out sensor`
`#define FIL_RUNOUT_STATE LOW // voltage level of the filament runout sensor trigger signal. Set according to the actual situation of the module. If the module sends a low level when the filament is abnormal, set it to LOW.`

Smart Filament Sensor (SFS V1.0)

The smart filament sensor works by continuously sending signal to the mainboard to communicate filament status.

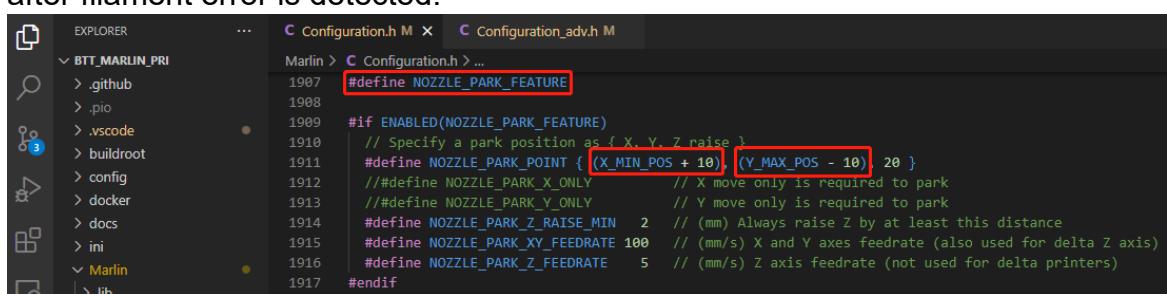


```

#define FILAMENT_MOTION_SENSOR // set encoder type
#define FILAMENT_RUNOUT_DISTANCE_MM 7 // set sensitivity, SFS V1.0
nominal setting should be 7mm, which means if no signal of filament movement is detected after 7mm of filament travel command, filament error will be triggered.

```

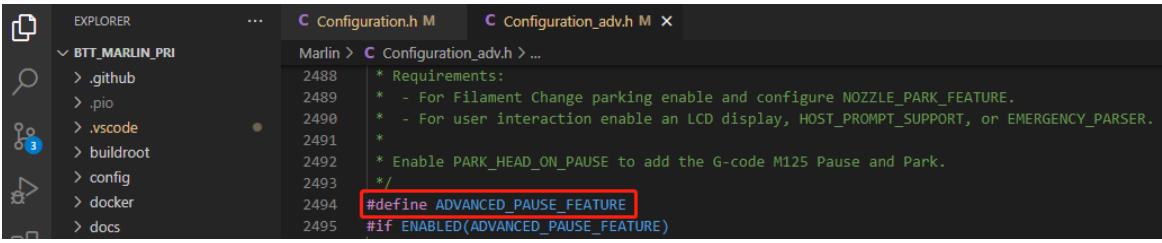
The settings below also need to be set to instruct the printer to park the nozzle after filament error is detected.



```

#define NOZZLE_PARK_FEATURE // park nozzle
#define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 }
// set the X, Y and Z offset coordinate of the nozzle

```



```

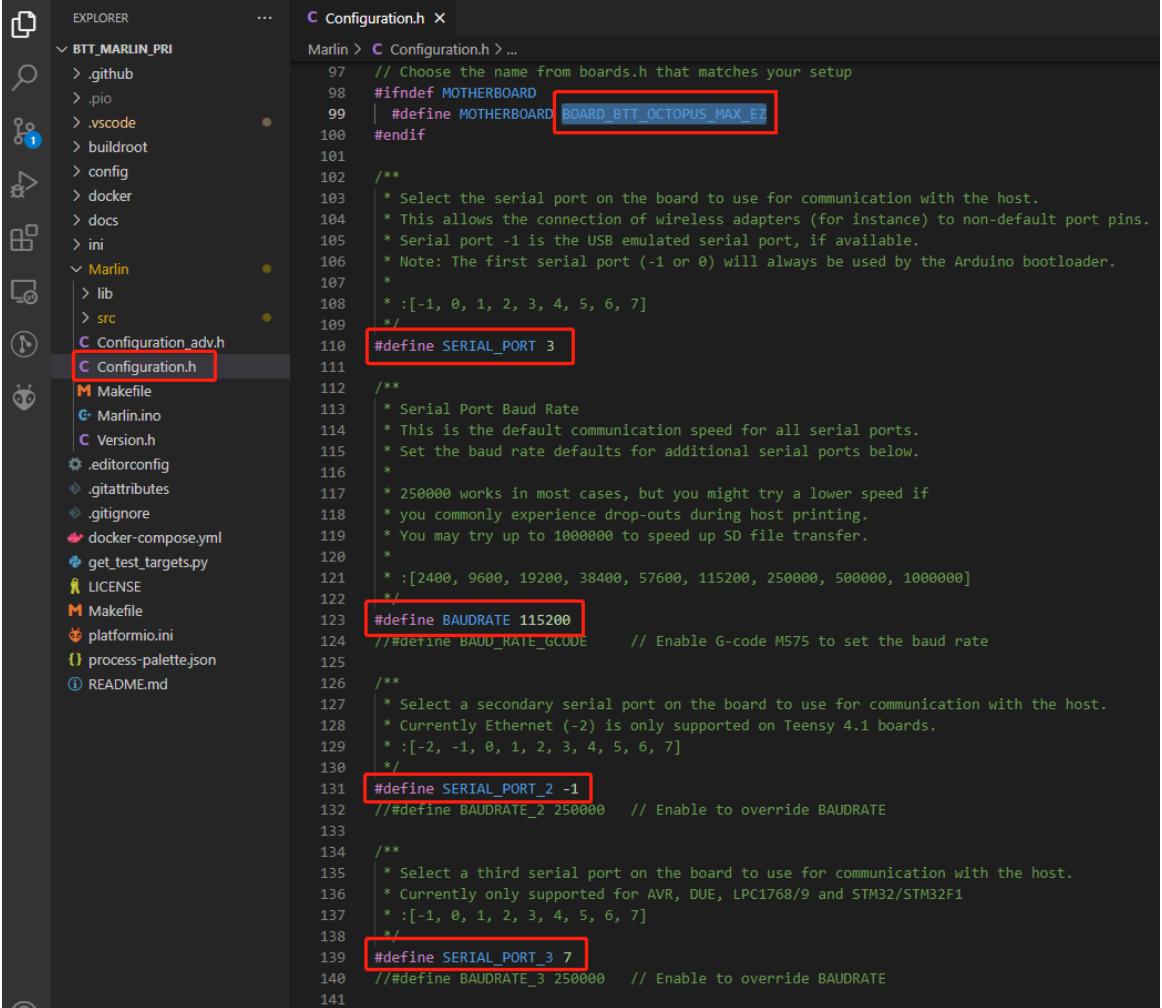
EXPLORER      C Configuration.h M C Configuration_adv.h M
BTT_MARLIN_PRI Marlin > C Configuration_adv.h > ...
2488 * Requirements:
2489 * - For Filament Change parking enable and configure NOZZLE_PARK_FEATURE.
2490 * - For user interaction enable an LCD display, HOST_PROMPT_SUPPORT, or EMERGENCY_PARSER.
2491 *
2492 * Enable PARK_HEAD_ON_PAUSE to add the G-code M125 Pause and Park.
2493 */
2494 #define ADVANCED_PAUSE_FEATURE
2495 #if ENABLED(ADVANCED_PAUSE_FEATURE)

```

`#define ADVANCED_PAUSE_FEATURE` // retraction setting of nozzle park movement and filament purge distance after the print is resumed.

ESP3D

In Marlin, simply set the correct "SERIAL_PORT" and "BAUDRATE". UART3 is used for ESP8266 communication, so set SERIAL_PORT to 3.



```

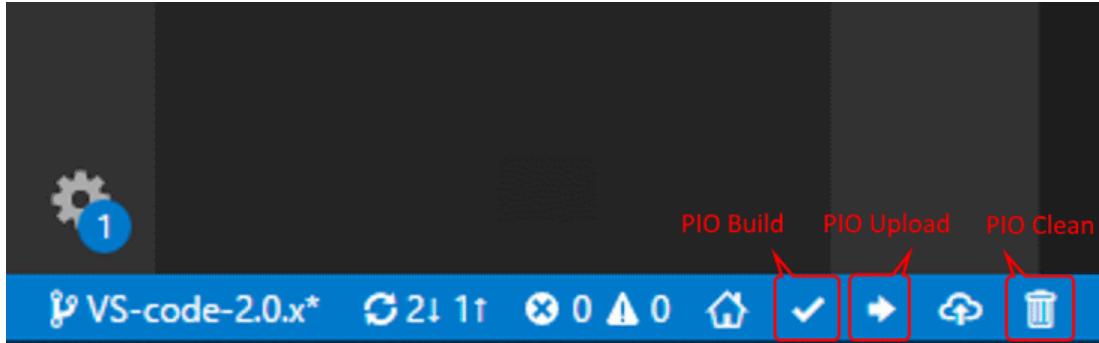
EXPLORER      C Configuration.h X
BTT_MARLIN_PRI Marlin > C Configuration.h > ...
97 // Choose the name from boards.h that matches your setup
98 #ifndef MOTHERBOARD
99 #define MOTHERBOARD BOARD_BTT_OCTOPUS_MAX_EZ
100#endif
101 /**
102 * Select the serial port on the board to use for communication with the host.
103 * This allows the connection of wireless adapters (for instance) to non-default port pins.
104 * Serial port -1 is the USB emulated serial port, if available.
105 * Note: The first serial port (-1 or 0) will always be used by the Arduino bootloader.
106 *
107 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
108 */
109#define SERIAL_PORT 3
110 /**
111 * Serial Port Baud Rate
112 * This is the default communication speed for all serial ports.
113 * Set the baud rate defaults for additional serial ports below.
114 *
115 * 250000 works in most cases, but you might try a lower speed if
116 * you commonly experience drop-outs during host printing.
117 * You may try up to 1000000 to speed up SD file transfer.
118 *
119 * :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
120 */
121#define BAUDRATE 115200
122 //##define BAUD_RATE_GCODE // Enable G-code M575 to set the baud rate
123 /**
124 * Select a secondary serial port on the board to use for communication with the host.
125 * Currently Ethernet (-2) is only supported on Teensy 4.1 boards.
126 * :[-2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
127 */
128#define SERIAL_PORT_2 -1
129 //##define BAUDRATE_2 250000 // Enable to override BAUDRATE
130 /**
131 * Select a third serial port on the board to use for communication with the host.
132 * Currently only supported for AVR, DUE, LPC1768/9 and STM32/STM32F1
133 * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
134 */
135#define SERIAL_PORT_3 7
136 //##define BAUDRATE_3 250000 // Enable to override BAUDRATE
137
138
139
140
141

```

The newest ESP3D firmware can be found at <https://github.com/luc-github/ESP3D>, compile your own binary file and rename it to "esp3d.bin", copy it to the root directory of the SD card, insert into the motherboard and press the reset button. The bootloader will update the firmware to ESP8266 automatically. If updated successfully, the file will be renamed to "ESP3D.CUR".

Compile Firmware

1. Click "√" to compile firmware.



2. Copy the compiled "firmware.bin" to SD card and insert to motherboard to update firmware.

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL
> ▾ TERMINAL
⌚ Compiling .pio\build\STM32H723Zx_btt\FrameworkArduino\wiring_time.c.o
Archiving .pio\build\STM32H723Zx_btt\libFrameworkArduino.a
Linking .pio\build\STM32H723Zx_btt\firmware.elf
Checking size .pio\build\STM32H723Zx_btt\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [ ] 3.0% (used 17472 bytes from 577536 bytes)
Flash: [====] 37.4% (used 195880 bytes from 524288 bytes)
Building .pio\build\STM32H723Zx_btt\firmware.bin
===== [SUCCESS] Took 88.43 seconds =====
Environment Status Duration
STM32H723Zx_btt SUCCESS 00:01:28.432
===== 1 succeeded in 00:01:28.432 =====
```

A screenshot of the VS Code terminal window. It shows the compilation process for a project named 'STM32H723Zx_btt'. The terminal output includes memory usage details for RAM and Flash, followed by the command 'Building .pio\build\STM32H723Zx_btt\firmware.bin'. The entire line of text is highlighted with a red box. At the bottom of the terminal, it shows the environment, status, and duration of the build: 'STM32H723Zx_btt SUCCESS 00:01:28.432'. The entire line of text at the bottom is also highlighted with a red box.

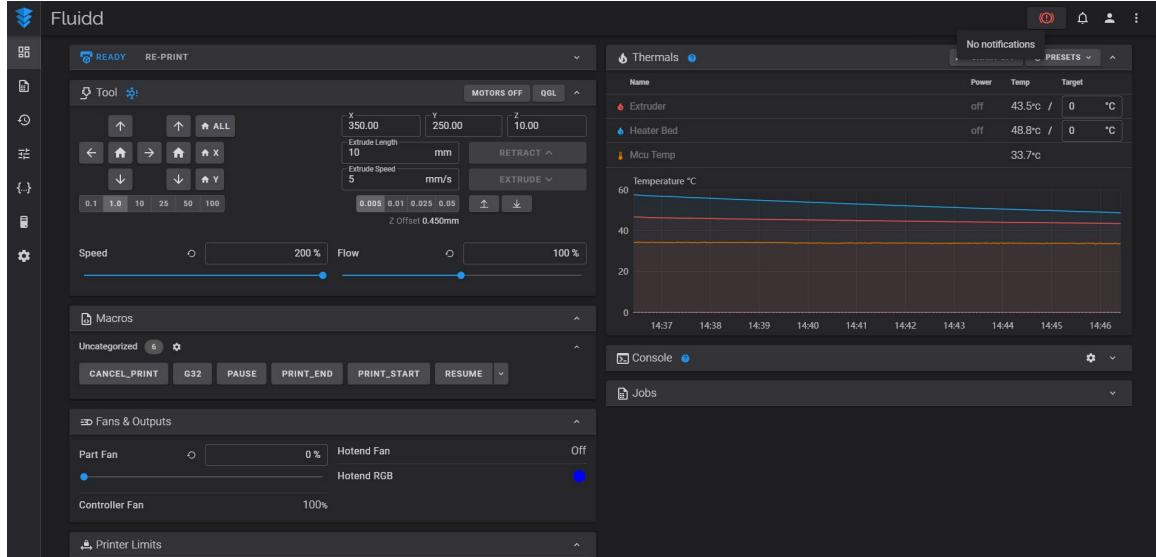
Klipper

Preparation

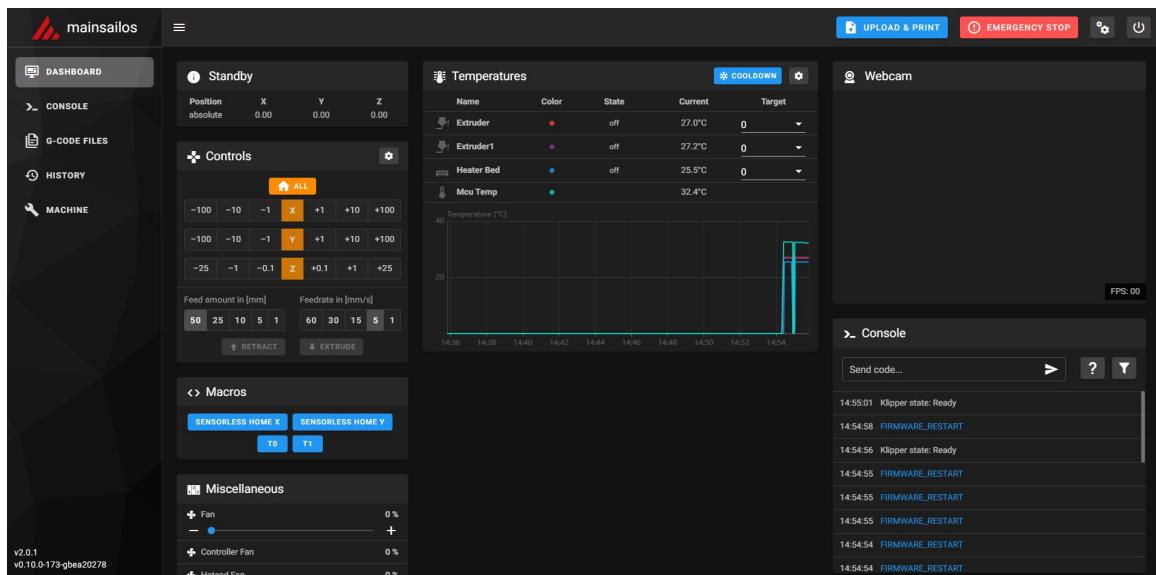
Download OS Image

Download your preferred OS image with build-in WebUI, popular choices are Fluiidd, Mainsail, etc.

Fluiidd: <https://github.com/fluiidd-core/FluiiddPi/releases>



Mainsail: <https://github.com/mainsail-crew/MainsailOS/releases>



Or refer to [Klipper official installation guide](#) using Octoprint.

Download and Install Raspberry Pi Imager

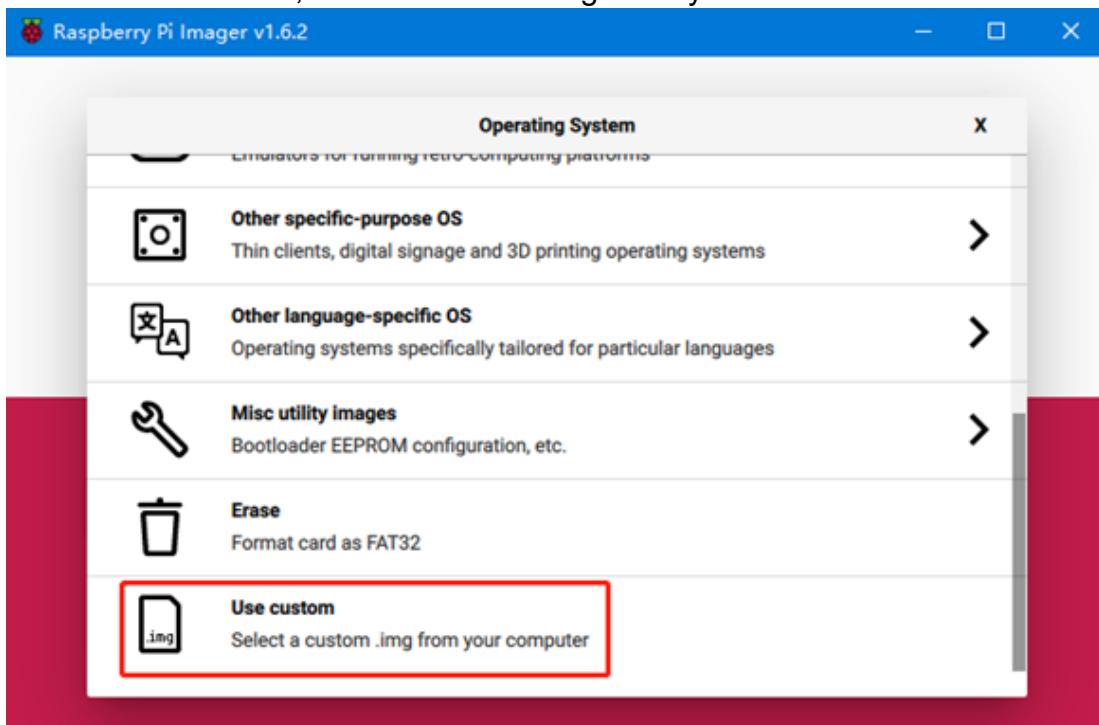
Install the official Raspberry Pi Imager <https://www.raspberrypi.com/software/>

Write Image

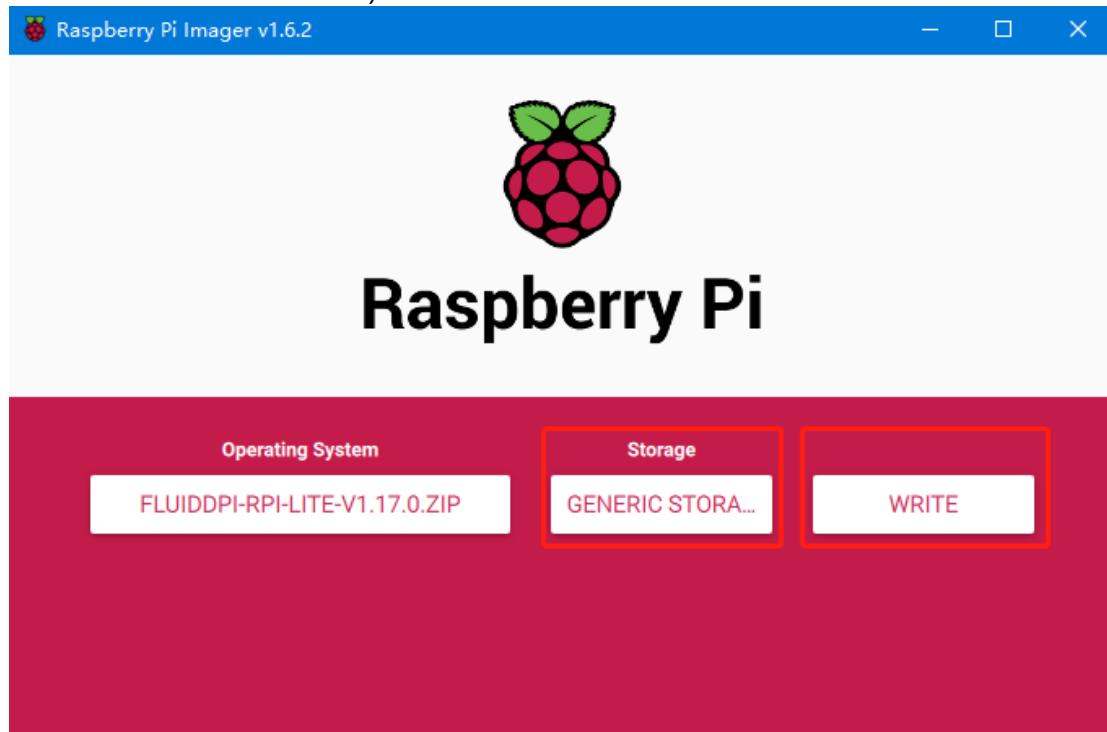
1. Insert microSD into your computer via a card reader.
2. Choose OS.



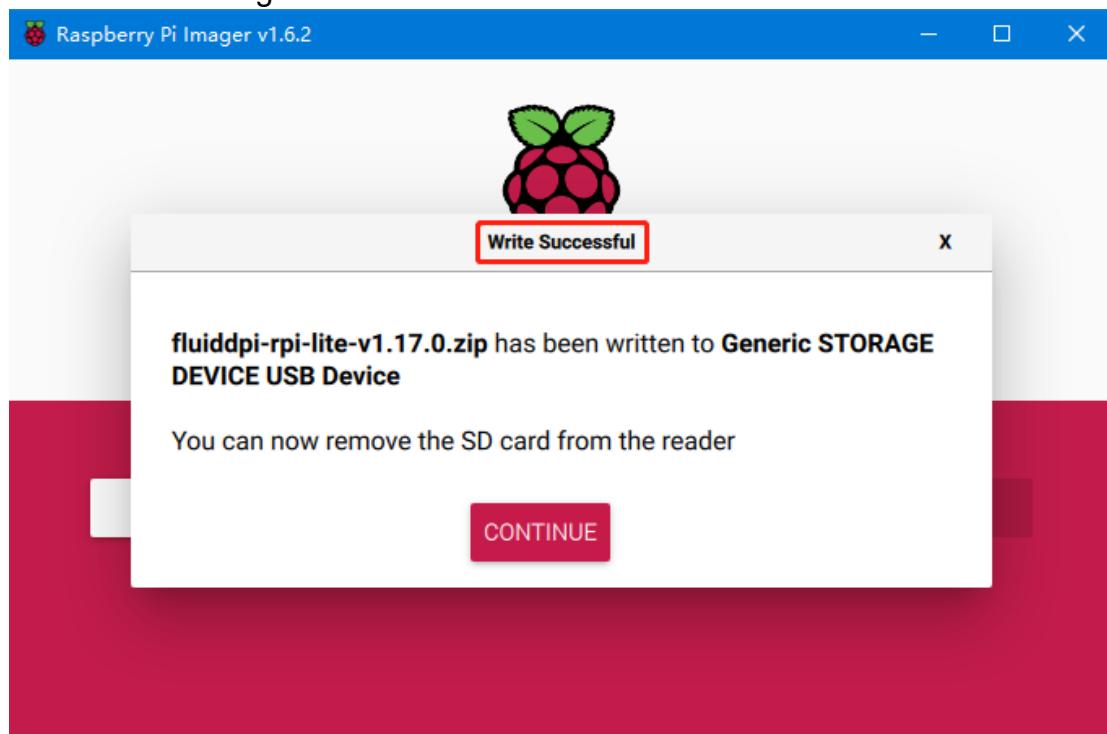
3. Select "Use custom", then select the image that you downloaded.



4. Select the microSD card and click "WRITE" (WRITE the image will format the microSD card. Be careful not to select the wrong storage device, otherwise the data will be formatted).



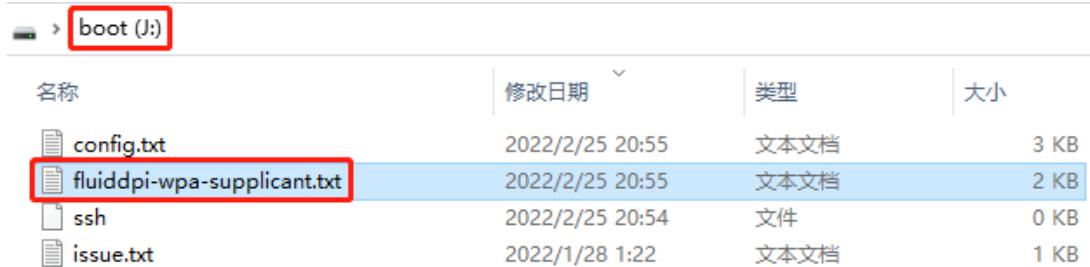
5. Wait for the writing to finish.



WiFi Setting

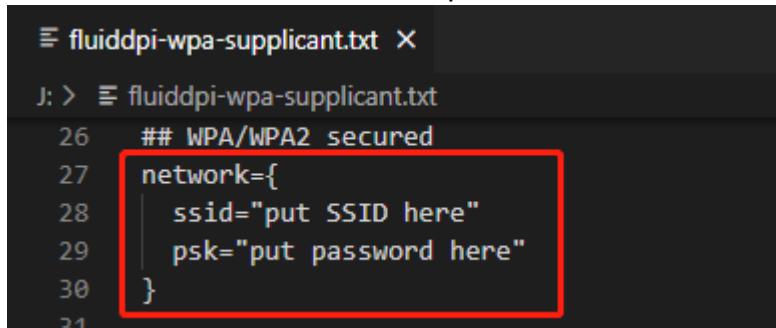
Note: this step can be skipped if you are using a network cable connection.

1. Reinsert the card reader.
2. Find "fluiddpi-wpa-supplicant.txt" or "mainsail-wpa-supplicant.txt" in the SD card root directory, open it with VSCode (do not open it with Windows Notepad)



名称	修改日期	类型	大小
config.txt	2022/2/25 20:55	文本文档	3 KB
fluiddpi-wpa-supplicant.txt	2022/2/25 20:55	文本文档	2 KB
ssh	2022/2/25 20:54	文件	0 KB
issue.txt	2022/1/28 1:22	文本文档	1 KB

3. Remove the '#' character at the beginning of the four lines in the red box, then set the correct WIFI name and password and save.



```

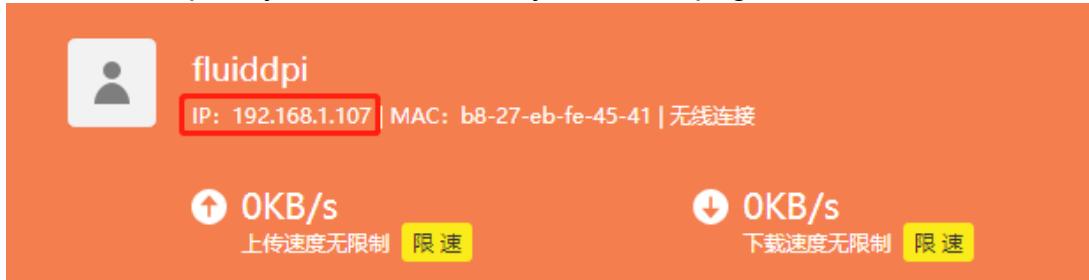
fluiddpi-wpa-supplicant.txt X
J: > fluiddpi-wpa-supplicant.txt
26 ## WPA/WPA2 secured
27 network={
28   ssid="put SSID here"
29   psk="put password here"
30 }
31

```

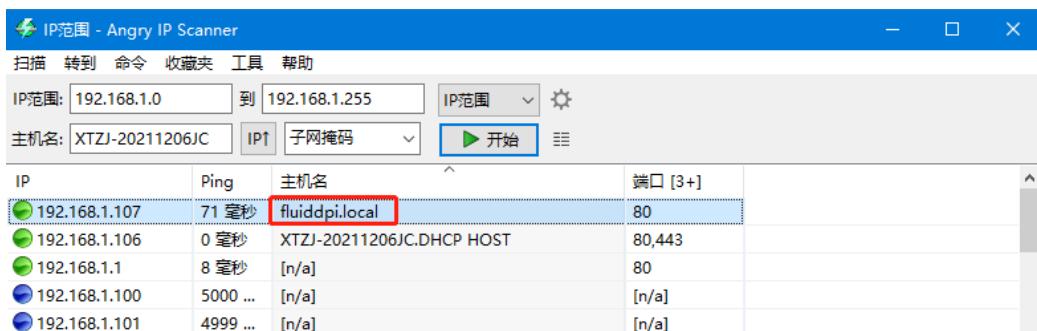
SSH Connect to Raspberry Pi

1. Install the SSH application Mobaxterm:
<https://mobaxterm.mobatek.net/download-home-edition.html>
2. Insert SD card to Raspberry Pi, wait for system to load after power on, approx. 1-2min.
3. The Raspberry Pi will automatically be assigned an IP address after being successfully connected to the network.

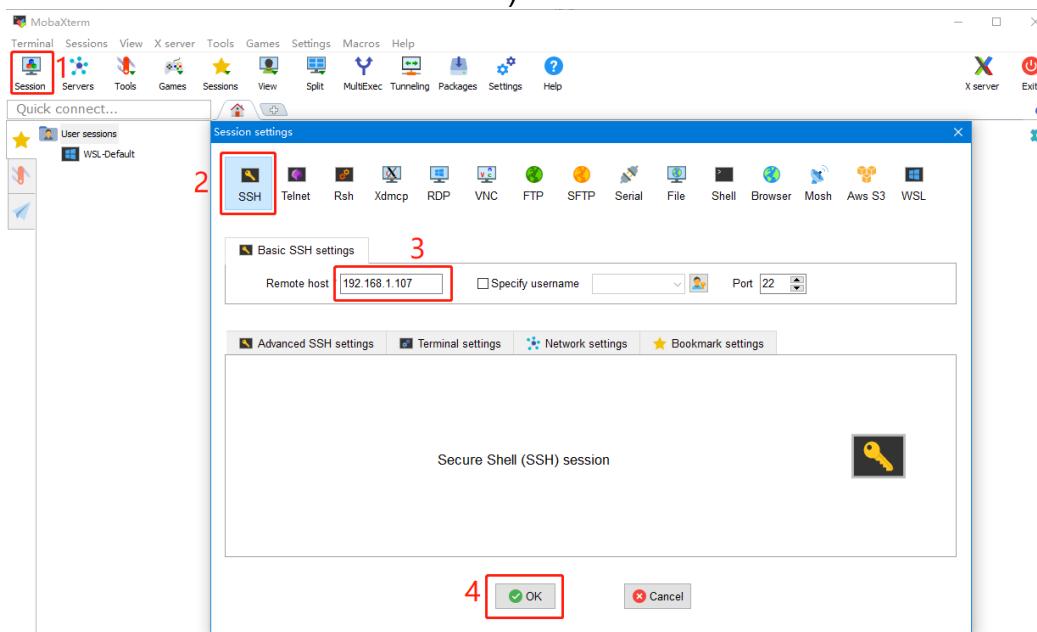
- Find the Raspberry Pi IP address on your router page.



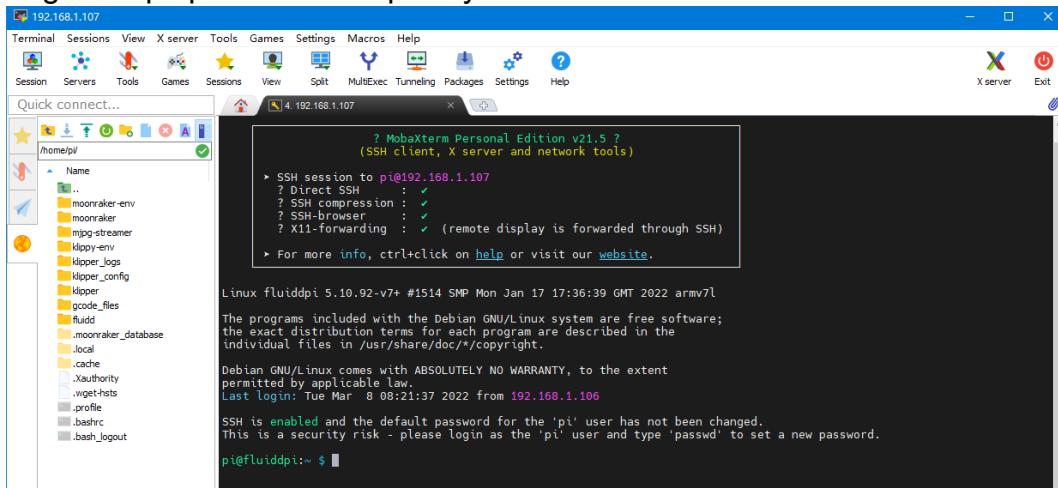
- Or use the <https://angryip.org/> tool, scan all IP addresses in the current network organize by names, and find the IP named Fludd or Mailsail, as shown below.



- Open MobaXterm and click "Session", and click "SSH", inset the Raspberry Pi IP into Remote host and click "OK" (Note: your computer and the Raspberry Pi needs to be in the same network).



7. Login as: pi password: raspberry



Compile Firmware

- After SSH successfully connected to the Raspberry Pi, enter in terminal:

```
cd ~/klipper/
```

```
make menuconfig
```

Compile with the configuration shown below (if the options below are not available, please update your Klipper source code to the newest version).

*** [*] Enable extra low-level configuration options**

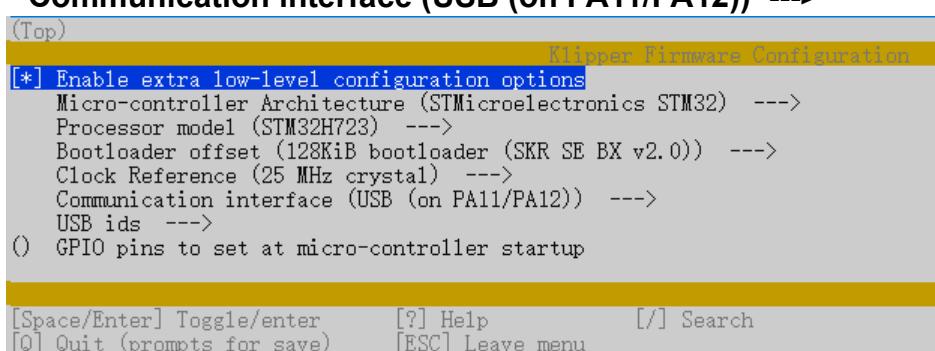
*** Micro-controller Architecture (STMicroelectronics STM32) --->**

*** Processor model (STM32H723) --->**

*** Bootloader offset (128KiB bootloader (SKR SE BX v2.0)) --->**

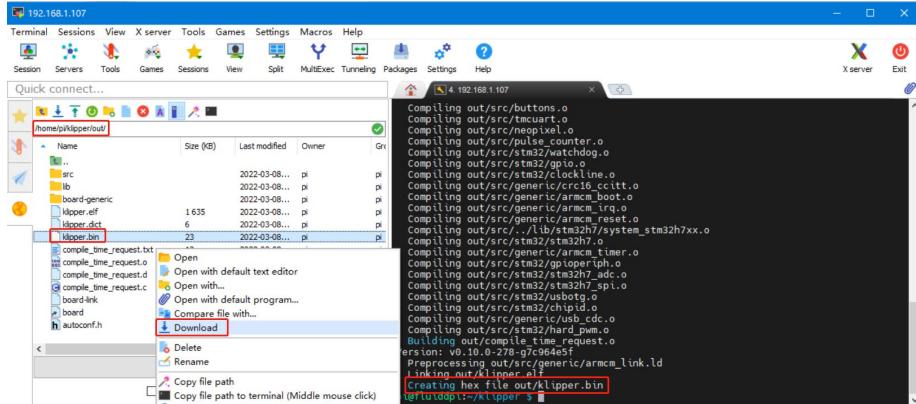
*** Clock Reference (25 MHz crystal) --->**

*** Communication interface (USB (on PA11/PA12)) --->**



- Press **q** to exit, and **Yes** when asked to save the configuration.
- Run **make** to compile firmware, "klipper.bin" file will be generated in **home/pi/klipper/out** folder when **make** is finished, download it onto your

computer using the SSH application.



4. Rename klipper.bin to "firmware.bin", copy to SD card to update firmware.
5. Enter: `ls /dev/serial/by-id/` in command line to check motherboard ID to confirm whether firmware is updated successfully, as shown below.

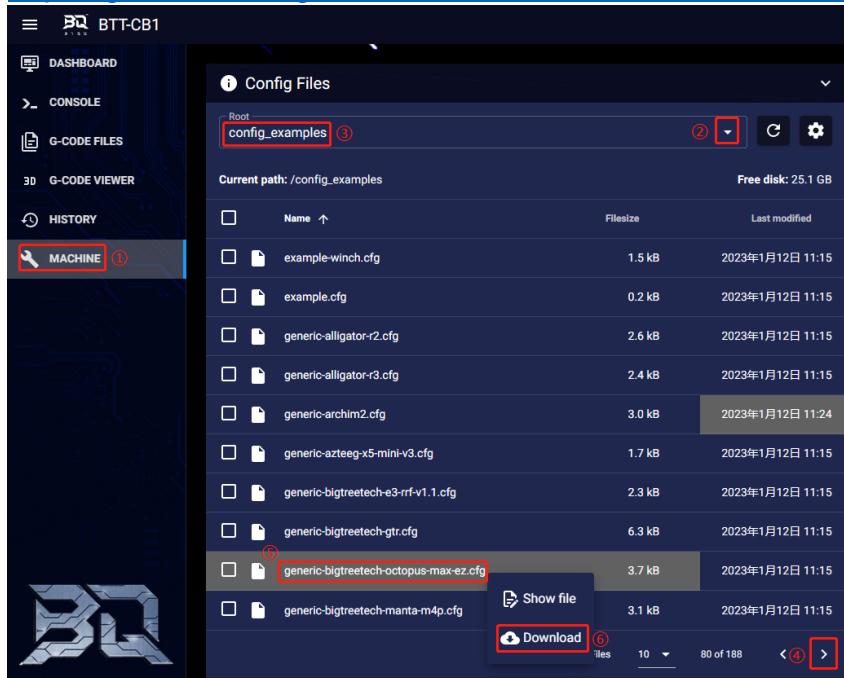
```
pi@fluiddp1:~/klipper $ ls /dev/serial/by-id/
usb-Klipper_stm32h723xx_41003D001751303232383230-if00
pi@fluiddp1:~/klipper $
```

copy and save this ID, it is needed when modifying klipper config.

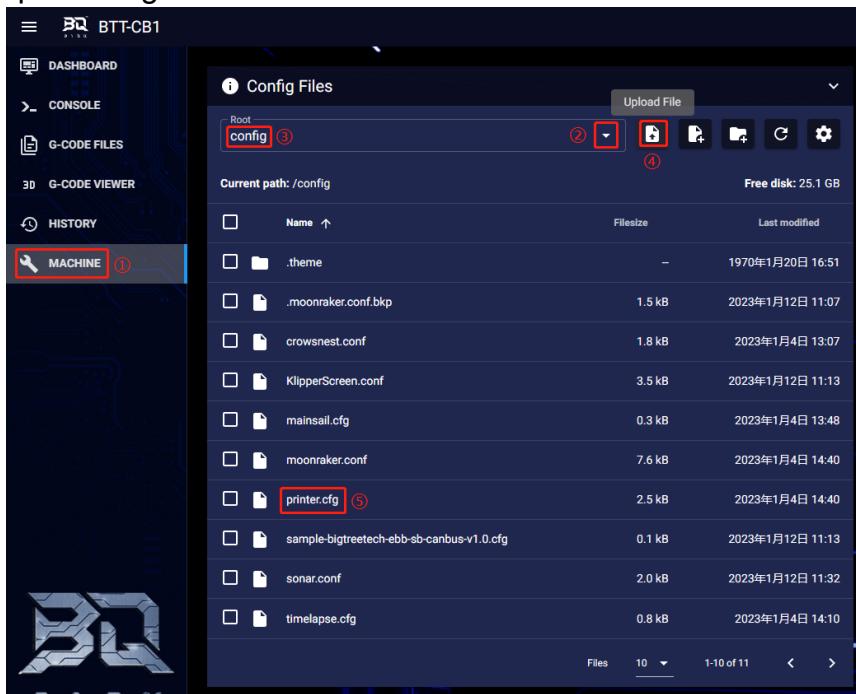
Configure Klipper

1. Enter your Raspberry Pi IP address into your browser to open the webUI, find the reference config for motherboard in the directory shown below, if there is no such config available, update your Klipper source code to the newest version or download from GitHub:

<https://github.com/bigtreeTech/BIGTREETECH-OCTOPUS-Max-EZ>



2. Upload your finished config file into Configuration Files, and rename it to "printer.cfg".



3. Insert the correct motherboard ID.

```
X printer.cfg
81
82 [mcu]
83 serial: /dev/serial/by-id/usb-Klipper_stm32h723xx_41003D001751303232383230-if00
84
```

4. Refer to <https://www.klipper3d.org/Overview.html> for detailed configuration guide according to your machine type.

Firmware Updates

Updating via microSD

1. Ensure the microSD card is formatted as FAT32.
2. Rename the compiled firmware or the firmware downloaded from GitHub to "firmware.bin" (**note:** make sure the computer system's extension settings are clear, as some users hide the extension, and "firmware.bin" actually displays as "firmware").
3. Copy "firmware.bin" to the root directory of the microSD card.
4. Insert the microSD card into the motherboard's slot, power on the motherboard, and the bootloader will automatically update the firmware.
5. The status LED will blink during update.
6. When it stops and the file is renamed "FIRMWARE.CUR", the update is complete.

Updating Klipper via DFU

1. Run `ls /dev/serial/by-id/` to get the board ID. If Klipper is running, it will return a klipper ID.

2. With the ID, enter: `cd ~klipper`

```
make flash FLASH_DEVICE= /dev/serial/by-id/usb-Klipper_stm32h712xx_41003D001751303232383230-if00
```

to flash the firmware (**note:** replace `/dev/serial/by-id/xxx` with the actual ID found in the previous step).

```
bigu@Hurakan:~/klipper$ make flash FLASH_DEVICE=/dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000D50415833323520-if00
Building hid-flash
/bin/sh: 1: pkg-config: not found
hid-flash requires libusb-1.0, please install with:
  sudo apt-get install libusb-1.0
  Flashing out/klipper.bin to /dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000D50415833323520-if00
Entering bootloader on /dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000D50415833323520-if00
Device reconnect on /sys/devices/platform/soc/5200000.usb/usb1/1-1/1-1.1/1-1.1:1.0
sudo dfu-util -p 1-1.1 -R -a 0 -s 0x8002000:leave -D out/klipper.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 = 0x08002000, size = 25264
Download [=====] 100% 25264 bytes
Download done.
File downloaded successfully
dfu-util: Error during download get_status

Failed to flash to /dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000D50415833323520-if00: 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=/dev/serial/by-id/usb-Klipper_stm32g0b1xx_190028000D50415833323520-if00
```

Ignore the dfu-util errors after successful flash.

Precautions

- All plugging and unplugging operations should be performed with the power off.
- When using a fan, make sure the voltage selection matches the fan's working voltage to prevent damage to the fan.

If you need further resources for this product, you can find them at [GitHub](<https://github.com/bigtreetech/>). If you cannot find what you need, you may contact our after-sales support(service005@biqu3d.com).

If you encounter any other problems during use or have suggestions or feedback, please contact us. Thank you for choosing BIGTREETECH products.