

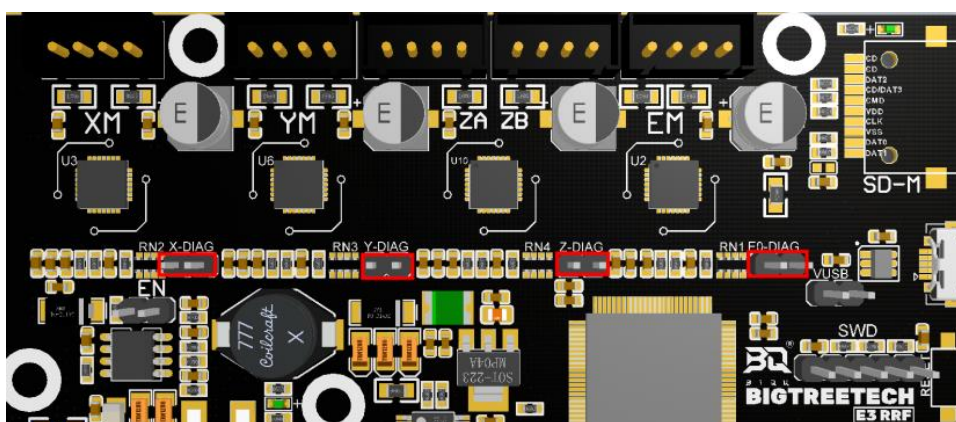
Introduction

This guide is most useful for users who are looking for advanced DIY features from their E3 RRF. If you are a user who has a standard i3 style installation then you can use the pre-compiled firmware.bin file available on our github site.

This guide is arranged to show how each additional feature on the E3 RRF can be used by configuring the hardware and firmware. Each sub-heading covers a separate feature.

TMC2209 Sensorless homing

1. Jumper configuration



Disconnect the limit switch from the motherboard for each axis that uses sensorless homing and close the corresponding DIAG circuit with a jumper. Note that it does not make sense to close the DIAG jumper on the extruder since a stall on the extruder does not indicate a runout.

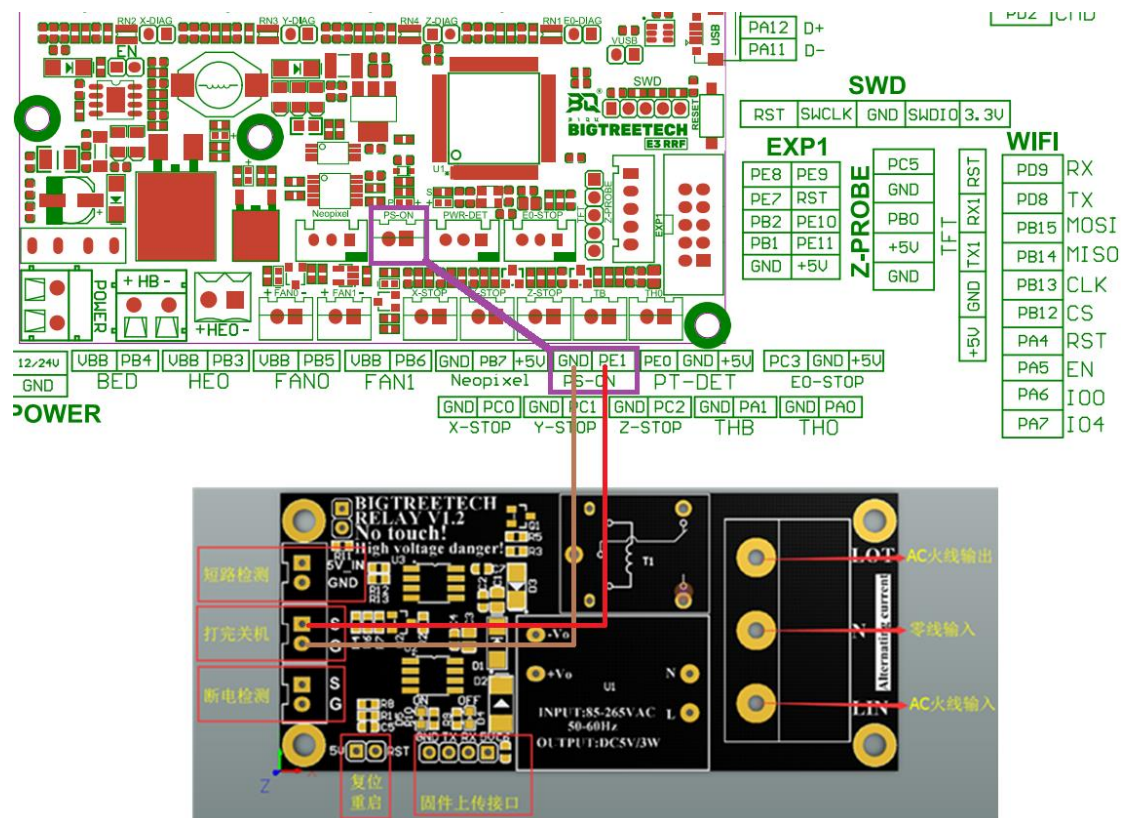
2. Firmware configuration

```
C Configuration.h C Configuration_adv.h X
Marlin > C Configuration_adv.h > SENSORLESS_HOMING
2686 | #define SENSORLESS_HOMING // StallGuard capable drivers only
2687
2688
2689 | #if EITHER(SENSORLESS_HOMING, SENSORLESS_PROBING)
2690 | // TMC2209: 0...255. TMC2130: -64...63
2691 | #define X_STALL_SENSITIVITY 80
2692 | #define X2_STALL_SENSITIVITY X_STALL_SENSITIVITY
2693 | #define Y_STALL_SENSITIVITY 70
2694 | #define Y2_STALL_SENSITIVITY Y_STALL_SENSITIVITY
2695 | #define Z_STALL_SENSITIVITY 60
2696 | // #define Z2_STALL_SENSITIVITY Z_STALL_SENSITIVITY
2697 | // #define Z3_STALL_SENSITIVITY Z_STALL_SENSITIVITY
2698 | // #define Z4_STALL_SENSITIVITY Z_STALL_SENSITIVITY
2699 | // #define SPI_ENDSTOPS // TMC2130 only
2700 | #define IMPROVE_HOMING_RELIABILITY
2701 | #endif
```

```
#define SENSORLESS_HOMING // enable sensorless homing with StallGuard
#define xxx_STALL_SENSITIVITY 80 // For sensorless homing on the TMC2209 the sensitivity needs to be
calibrated for each installation. If you set it too high then the axis will stop before it reaches the correct
endstop point. If you set it too low then the axis will grind up against the endstop point without stopping.
You can use M914 to change the sensitivity on an endstop during runtime. Our test on ender3 shows it
would be more appropriate to set "X=80, Y=70, Z=60" although this can vary with belt tightness and other
factors.
#define IMPROVE_HOMING_RELIABILITY//You can individually set the homing current (X_CURRENT_HOME)
to improve homing reliability. Generally sensorless homing works better when the homing current is set
lower.
```

Automatic power shutdown after print (Relay V1.2 module required)

1. Wiring diagram



Insert the control PIN of the module into the PS-ON port of the motherboard as per the diagram above.

2. Firmware configuration

```

C Configuration.h x C Configuration_adv.h
Marlin > C Configuration.h > PSU_CONTROL
314 #define PSU_CONTROL
315 #define PSU_NAME "Power Supply"
316
317 #if ENABLED(PSU_CONTROL)
318 #define PSU_ACTIVE_STATE HIGH // Set 'LOW' for ATX, 'HIGH' for X-Box
319
320 // #define PSU_DEFAULT_OFF // Keep power off until enabled directly with M80
321 // #define PSU_POWERUP_DELAY 250 // (ms) Delay for the PSU to warm up to full power
322
323 // #define PSU_POWERUP_GCODE "M355 S1" // G-code to run after power-on (e.g., case light on)
324 // #define PSU_POWEROFF_GCODE "M355 S0" // G-code to run before power-off (e.g., case light off)
325
326 // #define AUTO_POWER_CONTROL // Enable automatic control of the PS_ON pin
327 #if ENABLED(AUTO_POWER_CONTROL)
328 #define AUTO_POWER_FANS // Turn on PSU if fans need power
329 #define AUTO_POWER_E_FANS
330 #define AUTO_POWER_CONTROLLERFAN
331 #define AUTO_POWER_CHAMBER_FAN
332 // #define AUTO_POWER_E_TEMP 50 // (°C) Turn on PSU if any extruder is over this temperature
333 // #define AUTO_POWER_CHAMBER_TEMP 30 // (°C) Turn on PSU if the chamber is over this temperature
334 #define POWER_TIMEOUT 30 // (s) Turn off power if the machine is idle for this duration
335 // #define POWER_OFF_DELAY 60 // (s) Delay of poweroff after M81 command. Useful to let fans run for extra time.
336 #endif
337 #endif

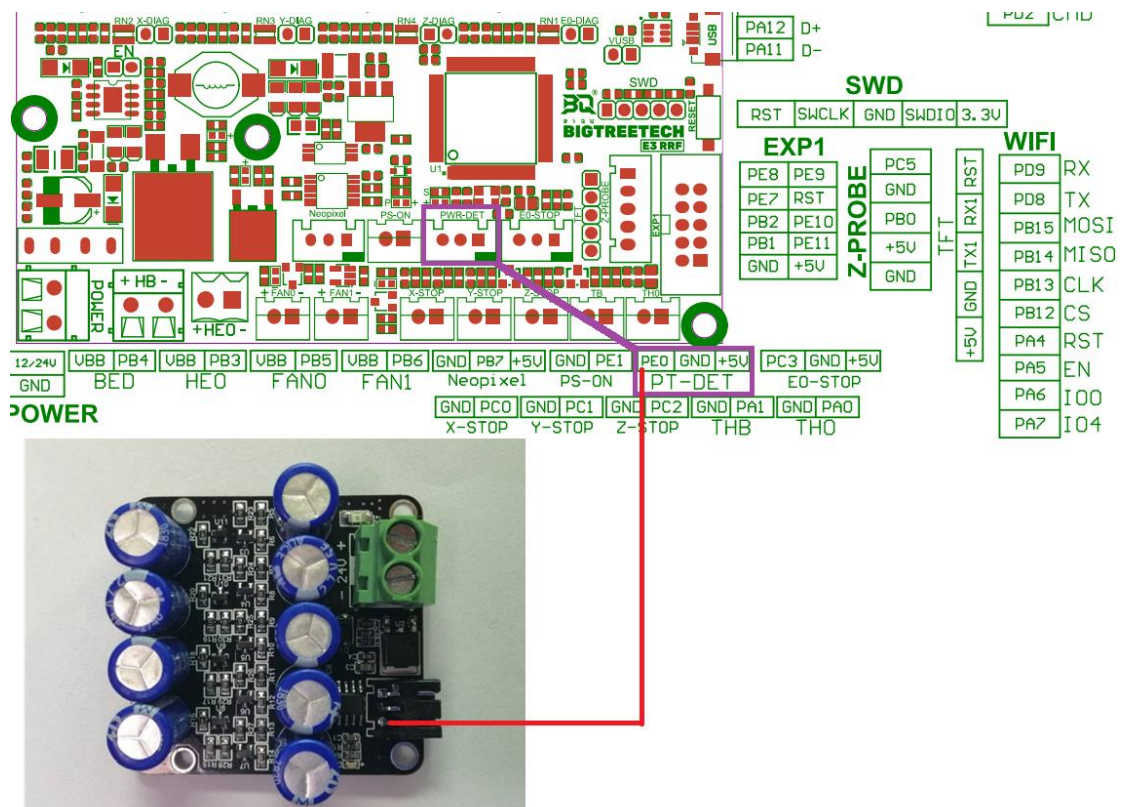
```

#define PSU_CONTROL // Power on/off PSU control with M80/M81

#define PSU_ACTIVE_STATE HIGH // Our Relay V1.2 module is on at a high (logic) level and off at a low level, so the state needs to be set to HIGH, which is different from other shutdown after print modules.

UPS Module (BTT UPS 24V)

1. Wiring diagram

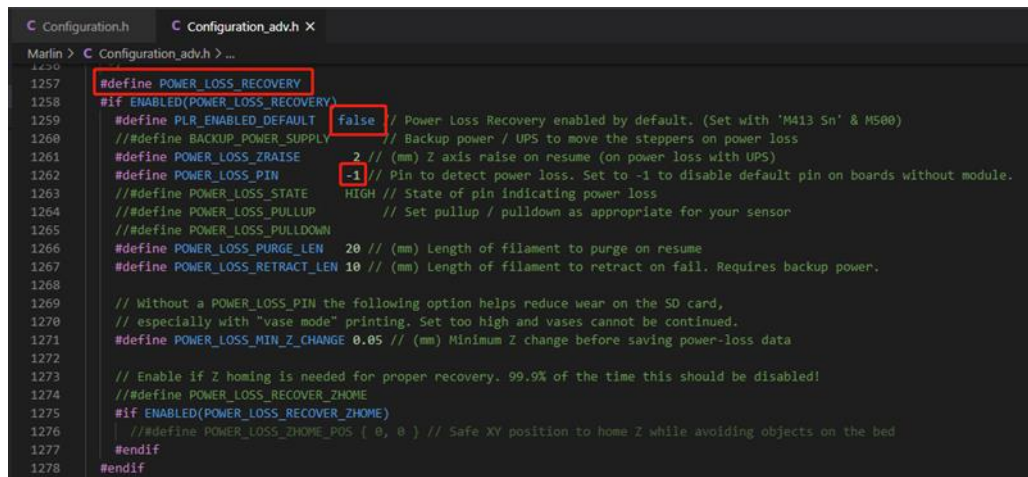


Insert the feedback signal line (PIN) of the module into the PWR-DET port on the motherboard

2. Firmware configuration

Power loss recovery can be achieved with the following methods:

- a. No external module is required. The printing status is saved to the SD card periodically throughout the print. If the power is removed during a print then the print will be resumed from the last known save point on the SD card. The saving frequency may be as low as once per layer depending on your firmware. The resume location may therefore not be exactly from where the print stopped. Additionally constant writes to the SD card can shorten its life meaningfully.

A screenshot of a code editor showing the 'Configuration_adv.h' file in the Marlin firmware. The file is open in a window titled 'Marlin > Configuration_adv.h > ...'. The code is in C++ and shows various preprocessor directives for configuring the printer. Several lines are highlighted with red boxes: line 1257 '#define POWER_LOSS_RECOVERY', line 1258 '#if ENABLED(POWER_LOSS_RECOVERY)', line 1259 '#define PLR_ENABLED_DEFAULT false', line 1261 '#define POWER_LOSS_ZRAISE 2', and line 1262 '#define POWER_LOSS_PIN -1'. The comments for these settings explain their functions, such as enabling power loss recovery, setting the Z-axis raise height, and selecting the pin for power loss detection.

```
1257 #define POWER_LOSS_RECOVERY
1258 #if ENABLED(POWER_LOSS_RECOVERY)
1259   #define PLR_ENABLED_DEFAULT false // Power Loss Recovery enabled by default. (Set with 'M413 S1' & M500)
1260   // #define BACKUP_POWER_SUPPLY // Backup power / UPS to move the steppers on power loss
1261   #define POWER_LOSS_ZRAISE 2 // (mm) Z axis raise on resume (on power loss with UPS)
1262   #define POWER_LOSS_PIN -1 // Pin to detect power loss. Set to -1 to disable default pin on boards without module.
1263   // #define POWER_LOSS_STATE HIGH // State of pin indicating power loss
1264   // #define POWER_LOSS_PULLUP // Set pullup / pulldown as appropriate for your sensor
1265   // #define POWER_LOSS_PULLDOWN
1266   #define POWER_LOSS_PURGE_LEN 20 // (mm) Length of filament to purge on resume
1267   #define POWER_LOSS_RETRACT_LEN 10 // (mm) Length of filament to retract on fail. Requires backup power.
1268
1269   // Without a POWER_LOSS_PIN the following option helps reduce wear on the SD card,
1270   // especially with "vase mode" printing. Set too high and vases cannot be continued.
1271   #define POWER_LOSS_MIN_Z_CHANGE 0.05 // (mm) Minimum Z change before saving power-loss data
1272
1273   // Enable if Z homing is needed for proper recovery. 99.9% of the time this should be disabled!
1274   // #define POWER_LOSS_RECOVER_ZHOME
1275   #if ENABLED(POWER_LOSS_RECOVER_ZHOME)
1276     // #define POWER_LOSS_ZHOME_POS { 0, 0 } // Safe XY position to home Z while avoiding objects on the bed
1277   #endif
1278 #endif
```

#define POWER_LOSS_RECOVERY // enable power-loss recovery feature in the firmware
#define PLR_ENABLED_DEFAULT false // false means disable the feature by default. Can be changed with M413 S1.
#define POWER_LOSS_PIN -1 // Pin to detect power loss. Set to -1 to disable default pin on boards without external detection module.

Power loss recovery is an enabled feature in the firmware.bin file on our Github but as it affects the life of SD card it is set to OFF by default. You can turn it on in "Configuration->Power Outage" on the 12864 screen and save settings in "Configuration->Store Settings", or enable power-loss recovery with "M413 S1" followed by "M500".

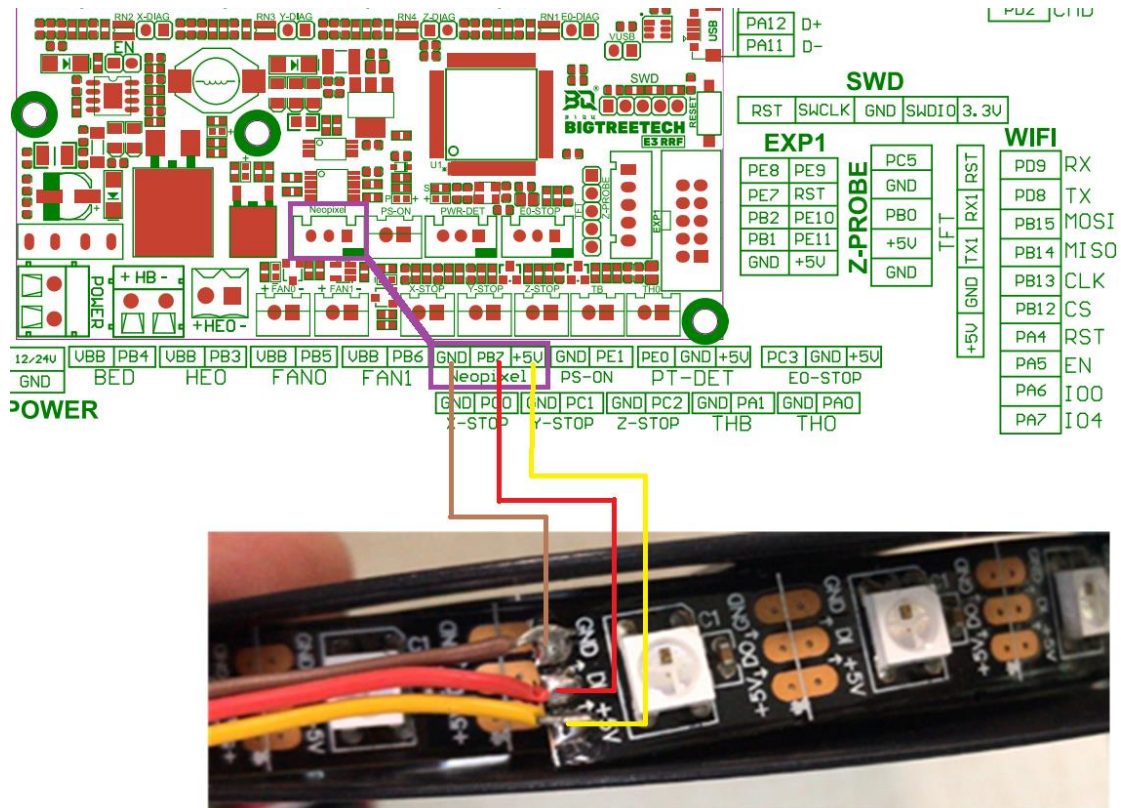
By using UPS modules like BTT UPS 24V, you can keep the motherboard powered for a brief period even after the mains power is removed. The module will then send a signal to the motherboard to indicate that mains power has been lost which will prompt the motherboard to save the printing state and raise the z axis away from the print. This method will only write data to the SD card once the mains power is removed and therefore will not affect the life of the SD card.


```
Configuration.h Configuration_adv.h X
Marlin > C Configuration_adv.h > POWER_LOSS_ZRAISE
1257 #define POWER_LOSS_RECOVERY
1258 #if ENABLED(POWER_LOSS_RECOVERY)
1259   #define PLR_ENABLED_DEFAULT true // Power Loss Recovery enabled by default. (Set with 'M413 Sn' & M500)
1260   #define BACKUP_POWER_SUPPLY // Backup power / UPS to move the steppers on power loss
1261   #define POWER_LOSS_ZRAISE 10 // (mm) Z axis raise on resume (on power loss with UPS)
1262   // #define POWER_LOSS_PIN -1 // Pin to detect power loss. Set to -1 to disable default pin on boards without module.
1263   #define POWER_LOSS_STATE HIGH // State of pin indicating power loss
1264   // #define POWER_LOSS_PULLUP // Set pullup / pulldown as appropriate for your sensor
1265   #define POWER_LOSS_PULLDOWN
1266   #define POWER_LOSS_PURGE_LEN 20 // (mm) Length of filament to purge on resume
1267   #define POWER_LOSS_RETRACT_LEN 10 // (mm) Length of filament to retract on fail. Requires backup power.
1268
1269   // Without a POWER_LOSS_PIN the following option helps reduce wear on the SD card,
1270   // especially with "vase mode" printing. Set too high and vases cannot be continued.
1271   #define POWER_LOSS_MIN_Z_CHANGE 0.05 // (mm) Minimum Z change before saving power-loss data
1272
1273   // Enable if Z homing is needed for proper recovery. 99.9% of the time this should be disabled!
1274   // #define POWER_LOSS_RECOVER_ZHOME
1275   #if ENABLED(POWER_LOSS_RECOVER_ZHOME)
1276     // #define POWER_LOSS_ZHOME_POS { 0, 0 } // Safe XY position to home Z while avoiding objects on the bed
1277   #endif
1278 #endif
```

#define POWER_LOSS_RECOVERY // include power-loss recovery feature in the firmware
#define PLR_ENABLED_DEFAULT true // true means power-loss recovery enabled by default
#define POWER_LOSS_ZRAISE 10 // the nozzle will raise 10mm, which protects against damaged print part
// #define POWER_LOSS_PIN -1 // Don't set the power loss pin here. The correct pin is already set within the pins file for the motherboard.
#define POWER_LOSS_STATE HIGH // This means that marlin will detect a power loss when the input pin goes high. Since the BTT UPS 24V module outputs a high level when the mains power is lost, this is the correct setting.
#define POWER_LOSS_PULLDOWN // This is an essential line since it will ensure that there are no false triggers detected from the BTT UPS 24V module.

RGB strip (WS2812,etc)

1. Wiring diagram



When connecting an RGB light onto the Neopixel port of the motherboard, pay close attention to the wiring of the GND, signal, and +5V lines. Incorrect wiring could result in damage to the motherboard or the peripheral board.

2. Firmware configuration

```
C Configuration.h x C Configuration_adv.h
Marlin > C Configuration.h > NEOPIXEL_LED
2638 #define NEOPIXEL_LED
2639 #if ENABLED(NEOPIXEL_LED)
2640 #define NEOPIXEL_TYPE NEO_GRB // NEO_GRBW / NEO_GRB - four/three channel driver type (defined in Adafruit_NeoPixel.h)
2641 #define NEOPIXEL_PIN 4 // LED driving pin
2642 // #define NEOPIXEL2_TYPE NEOPIXEL2_TYPE
2643 // #define NEOPIXEL2_PIN 5
2644 #define NEOPIXEL_PIXELS 30 // Number of LEDs in the strip. (Longest strip when NEOPIXEL2_SEPARATE is disabled.)
2645 #define NEOPIXEL_IS_SEQUENTIAL // Sequential display for temperature change - LED by LED. Disable to change all LEDs at once.
2646 #define NEOPIXEL_BRIGHTNESS 127 // Initial brightness (0-255)
2647 #define NEOPIXEL_STARTUP_TEST // Cycle through colors at startup
2648
2649 // Support for second Adafruit NeoPixel LED driver controlled with M150 S1 ...
2650 // #define NEOPIXEL2_SEPARATE
2651 #if ENABLED(NEOPIXEL2_SEPARATE)
2652 #define NEOPIXEL2_PIXELS 15 // Number of LEDs in the second strip
2653 #define NEOPIXEL2_BRIGHTNESS 127 // Initial brightness (0-255)
2654 #define NEOPIXEL2_STARTUP_TEST // Cycle through colors at startup
2655 #else
2656 // #define NEOPIXEL2_INSERIES // Default behavior is NeoPixel 2 in parallel
2657 #endif
2658
2659 // Use a single NeoPixel LED for static (background) lighting
2660 // #define NEOPIXEL_BKGD_LED_INDEX 0 // Index of the LED to use
2661 // #define NEOPIXEL_BKGD_COLOR { 255, 255, 255, 0 } // R, G, B, W
2662 // #define NEOPIXEL_BKGD_ALWAYS_ON // Keep the backlight on when other NeoPixels are off
2663 #endif
```

#define NEOPIXEL_LED //enable Neopixel feature

#define NEOPIXEL_TYPE NEO_GRB // set the type of RGB light

```

// #define NEOPIXEL_PIN 4 // Don't set the pin here. It's already set in the motherboard pins file.
#define NEOPIXEL_PIXELS 30 // Set this to the number of LEDs that are connected.
#define NEOPIXEL_STARTUP_TEST // during startup, it will display red, green, and blue in turn for easy testing
#define LED_CONTROL_MENU // Adds a menu to control the colors of LED on the screen

```

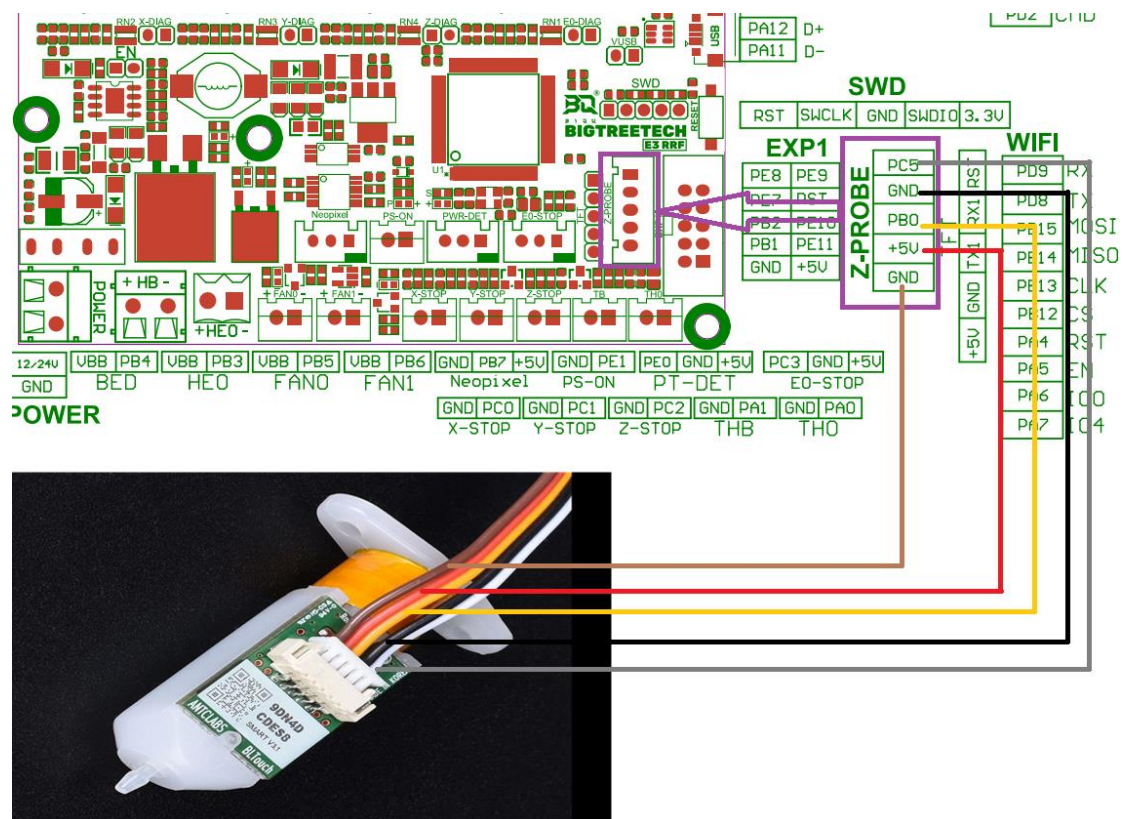
```

C Configuration.h  C Configuration_adv.h X
Marlin > C Configuration_adv.h > LED_CONTROL_MENU
1128  /**
1129   * LED Control Menu
1130   * Add LED Control to the LCD menu
1131   */
1132  #define LED_CONTROL_MENU
1133  #if ENABLED(LED_CONTROL_MENU)
1134    #define LED_COLOR_PRESETS           // Enable the Preset Color menu option
1135    // #define NEO2_COLOR_PRESETS       // Enable a second NeoPixel Preset Color menu option
1136    #if ENABLED(LED_COLOR_PRESETS)
1137      #define LED_USER_PRESET_RED       255 // User defined RED value
1138      #define LED_USER_PRESET_GREEN     128 // User defined GREEN value
1139      #define LED_USER_PRESET_BLUE      0   // User defined BLUE value
1140      #define LED_USER_PRESET_WHITE     255 // User defined WHITE value
1141      #define LED_USER_PRESET_BRIGHTNESS 255 // User defined intensity
1142      // #define LED_USER_PRESET_STARTUP // Have the printer display the user preset color on startup
1143    #endif
1144    #if ENABLED(NEO2_COLOR_PRESETS)
1145      #define NEO2_USER_PRESET_RED       255 // User defined RED value
1146      #define NEO2_USER_PRESET_GREEN     128 // User defined GREEN value
1147      #define NEO2_USER_PRESET_BLUE      0   // User defined BLUE value
1148      #define NEO2_USER_PRESET_WHITE     255 // User defined WHITE value
1149      #define NEO2_USER_PRESET_BRIGHTNESS 255 // User defined intensity
1150      // #define NEO2_USER_PRESET_STARTUP // Have the printer display the user preset color on startup for the second strip
1151    #endif
1152  #endif

```

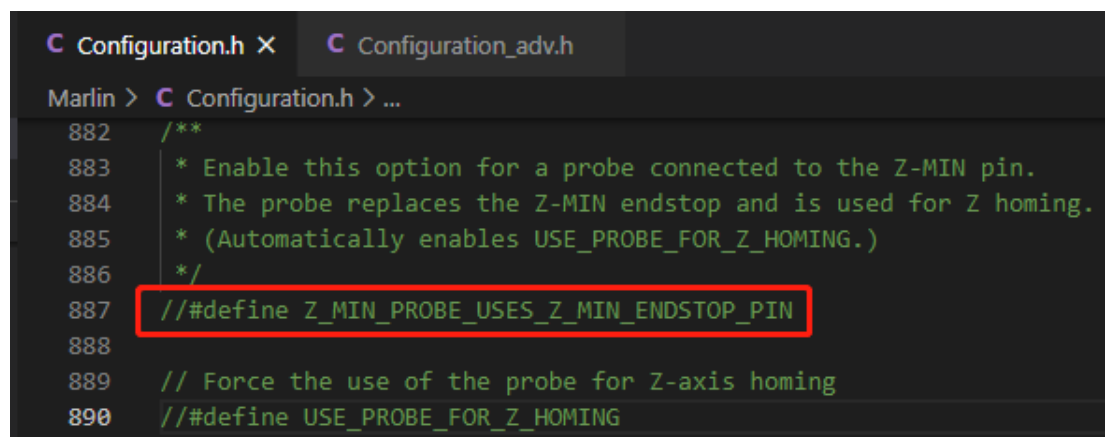
BL Touch

1. Wiring diagram



Connect the 3pin and a 2pin signals of the BL Touch to the 5pin Z-PROBE port of the motherboard according to the wiring shown in the image. If your BL touch does not have the same colour wiring as the one shown in the image then remove the connector and check whether there is a silk screen marking showing the pinout. If there is then make sure that the pinout is wired in the correct order to the z-probe connector.

2. Firmware configuration

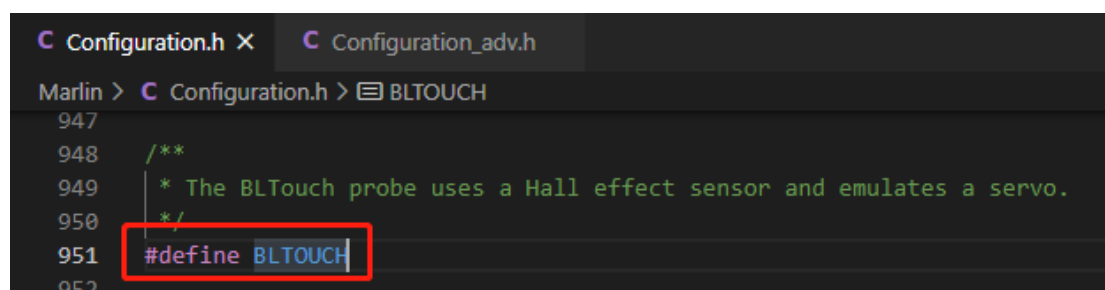


The screenshot shows the Marlin Configuration.h file in a code editor. The file is open to line 882. The code is as follows:

```
882  /**
883   * Enable this option for a probe connected to the Z-MIN pin.
884   * The probe replaces the Z-MIN endstop and is used for Z homing.
885   * (Automatically enables USE_PROBE_FOR_Z_HOMING.)
886   */
887  //#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN
888
889  // Force the use of the probe for Z-axis homing
890  //#define USE_PROBE_FOR_Z_HOMING
```

The line `//#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN` is highlighted with a red box.

`//#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN` Commenting this out will cause the probe to use the dedicated probe input pin. If you want to use the probe as the homing endstop then you can use the “`USE_PROBE_FOR_Z_HOMING`” definition below that statement. This is explained a little later in this document.



The screenshot shows the Marlin Configuration.h file in a code editor. The file is open to line 947. The code is as follows:

```
947
948  /**
949   * The BLTouch probe uses a Hall effect sensor and emulates a servo.
950   */
951  #define BLTOUCH
952
```

The line `#define BLTOUCH` is highlighted with a red box.

In configuration_adv.h make the following changes:

```
#define BLTOUCH // enable BLTOUCH
```

```
#define NOZZLE_TO_PROBE_OFFSET { -40, -10, -1.85 } // You will need to adjust this to the offset between your probe and nozzle.
```

```
#define PROBING_MARGIN 10 // set the probe margin for leveling, for example, the maximum probe area for Ender3 is 0-235 however, probing the full volume will likely hit bed clips or miss the bed slightly and therefore a margin is recommended.
```

PROBING_MARGIN = 10 reduces the probe area for leveling to 10-225.


```
C Configuration.h X C Configuration_adv.h
Marlin > C Configuration.h > ...
1372 */
1373 // #define AUTO_BED_LEVELING_3POINT
1374 // #define AUTO_BED_LEVELING_LINEAR
1375 #define AUTO_BED_LEVELING_BILINEAR
1376 // #define AUTO_BED_LEVELING_UBL
1377 // #define MESH_BED_LEVELING
1378
1379 /**
1380  * Normally G28 leaves leveling disabled on completion. Enable one of
1381  * these options to restore the prior leveling state or to always enable
1382  * leveling immediately after G28.
1383  */
1384 #define RESTORE_LEVELING_AFTER_G28
1385 // #define ENABLE_LEVELING_AFTER_G28
1386
```

#define AUTO_BED_LEVELING_BILINEAR // configure bed leveling

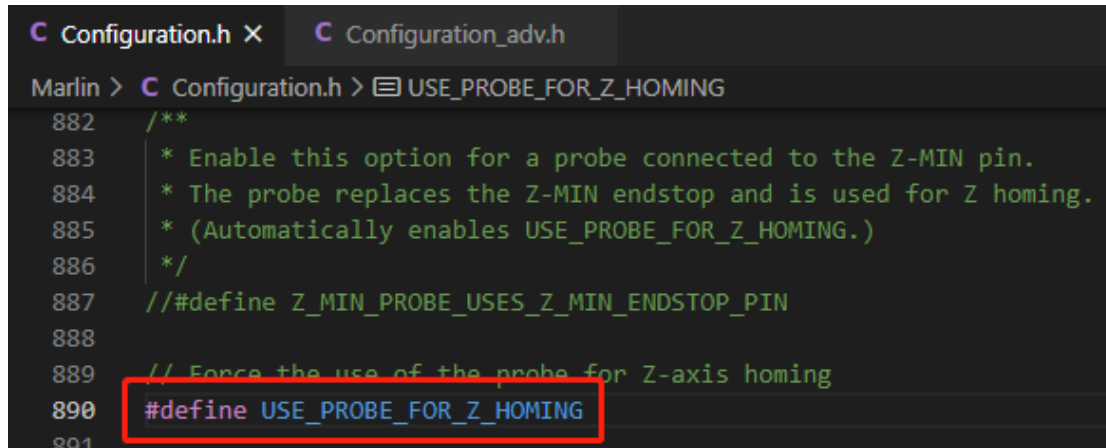
#define RESTORE_LEVELING_AFTER_G28 //restore leveling after G28

```
C Configuration.h X C Configuration_adv.h
Marlin > C Configuration.h > AUTO_BED_LEVELING_BILINEAR
1430 #endif
1431
1432 #endif
1433
1434 #if EITHER(AUTO_BED_LEVELING_LINEAR, AUTO_BED_LEVELING_BILINEAR)
1435
1436 // Set the number of grid points per dimension.
1437 #define GRID_MAX_POINTS_X 5
1438 #define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X
1439
1440 // Probe along the Y axis, advancing X after each column
1441 // #define PROBE_Y_FIRST
1442
1443 #if ENABLED(AUTO_BED_LEVELING_BILINEAR)
1444
1445 // Beyond the probed grid, continue the implied tilt?
1446 // Default is to maintain the height of the nearest edge.
1447 #define EXTRAPOLATE_BEYOND_GRID
1448
1449 //
1450 // Experimental Subdivision of the grid by Catmull-Rom method.
1451 // Synthesizes intermediate points to produce a more detailed mesh.
1452 //
1453 // #define ABL_BILINEAR_SUBDIVISION
1454 #if ENABLED(ABL_BILINEAR_SUBDIVISION)
1455 // Number of subdivisions between probe points
1456 #define BILINEAR_SUBDIVISIONS 3
1457 #endif
1458
1459 #endif
1460
```

#define GRID_MAX_POINTS_X 5 //set the number of grid points, and probe 5 points along the X-axis.
Configure this according to your needs.

```
#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X // probe 5 points along the Y-axis
```

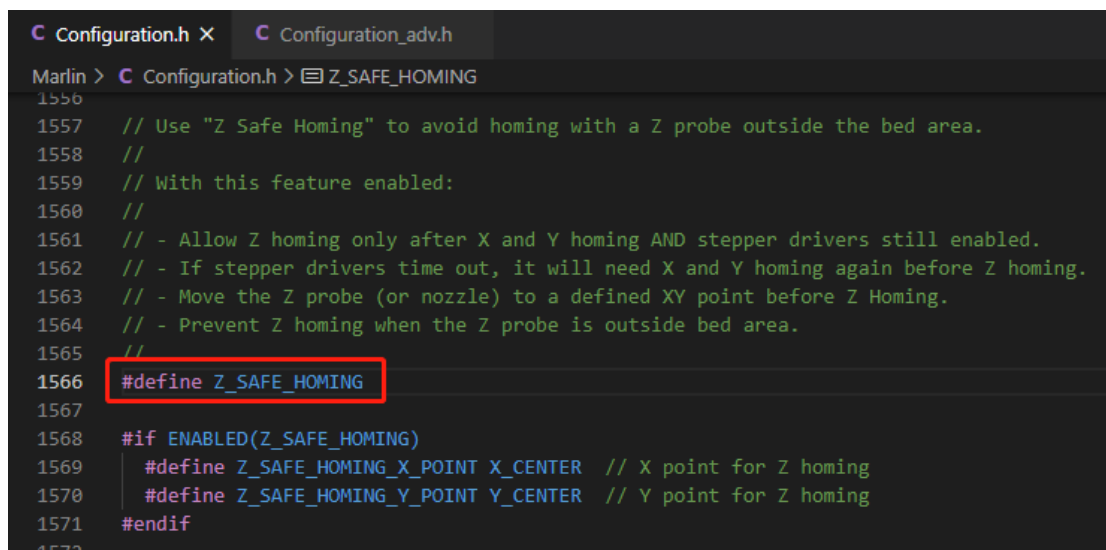
If you use BLTouch instead of Z-axis mechanical switch as Z-axis limit switch, you just modify the firmware setting without changing the wiring of BLTouch.



The screenshot shows the Marlin Configuration.h file in a code editor. The 'Configuration.h' tab is active. The cursor is at line 890, where the line `#define USE_PROBE_FOR_Z_HOMING` is highlighted with a red rectangle. The surrounding code includes comments about enabling this option for a probe connected to the Z-MIN pin and a line defining `Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN`.

```
C Configuration.h X C Configuration_adv.h
Marlin > C Configuration.h > USE_PROBE_FOR_Z_HOMING
882  /**
883   * Enable this option for a probe connected to the Z-MIN pin.
884   * The probe replaces the Z-MIN endstop and is used for Z homing.
885   * (Automatically enables USE_PROBE_FOR_Z_HOMING.)
886   */
887  // #define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN
888
889  // Force the use of the probe for Z-axis homing
890  #define USE_PROBE_FOR_Z_HOMING
891
```

```
#define USE_PROBE_FOR_Z_HOMING // use Z Probe (BLTouch) as Z-axis homing limit switch
```



The screenshot shows the Marlin Configuration.h file in a code editor. The 'Configuration.h' tab is active. The cursor is at line 1566, where the line `#define Z_SAFE_HOMING` is highlighted with a red rectangle. The surrounding code includes comments about using 'Z Safe Homing' to avoid homing with a Z probe outside the bed area, and a conditional block for `ENABLED(Z_SAFE_HOMING)` that defines `Z_SAFE_HOMING_X_POINT` and `Z_SAFE_HOMING_Y_POINT`.

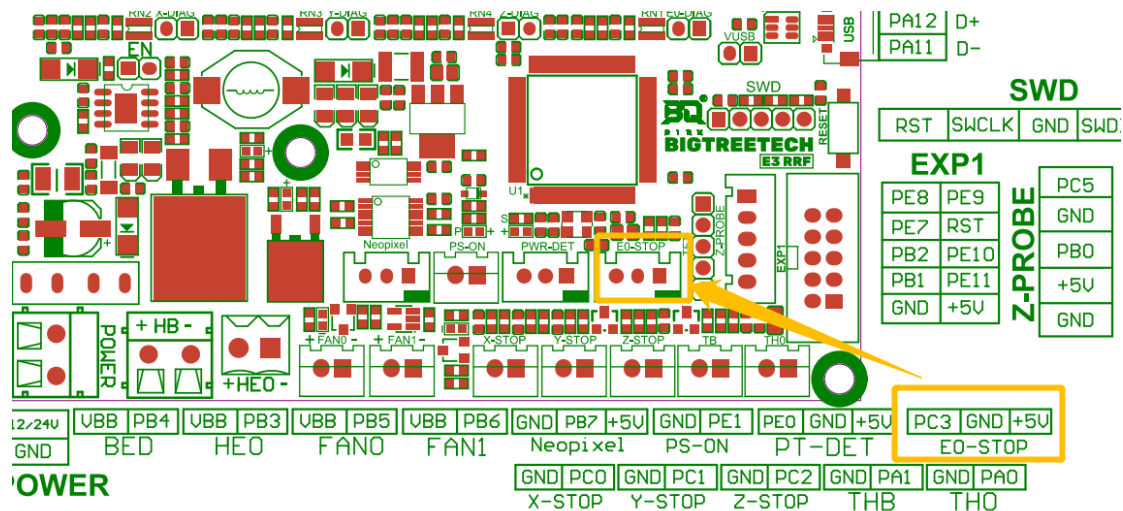
```
C Configuration.h X C Configuration_adv.h
Marlin > C Configuration.h > Z_SAFE_HOMING
1556
1557  // Use "Z Safe Homing" to avoid homing with a Z probe outside the bed area.
1558  //
1559  // With this feature enabled:
1560  //
1561  // - Allow Z homing only after X and Y homing AND stepper drivers still enabled.
1562  // - If stepper drivers time out, it will need X and Y homing again before Z homing.
1563  // - Move the Z probe (or nozzle) to a defined XY point before Z Homing.
1564  // - Prevent Z homing when the Z probe is outside bed area.
1565  //
1566  #define Z_SAFE_HOMING
1567
1568  #if ENABLED(Z_SAFE_HOMING)
1569    #define Z_SAFE_HOMING_X_POINT X_CENTER // X point for Z homing
1570    #define Z_SAFE_HOMING_Y_POINT Y_CENTER // Y point for Z homing
1571  #endif
1572
```

If you decide to use the z-probe for homing then it is recommended that you enable z safe homing.

```
#define Z_SAFE_HOMING // prevents the Z-Probe (BLTouch) is outside bed area by moving to a defined XY point (by default, the middle of the bed) before Z homing.
```

Filament runout detection module

1. Wiring diagram



2. Firmware configuration

This motherboard currently supports two filament runout detection modules:

- The standard filament runout detection module. This is generally nothing more than a mechanical switch which can detect whether filament is present within the path that runs over the switch.

```

C Configuration.h X C Configuration_adv.h
Marlin > C Configuration.h > FILAMENT_RUNOUT_SENSOR
1273 #define FILAMENT_RUNOUT_SENSOR
1274 #if ENABLED(FILAMENT_RUNOUT_SENSOR)
1275 #define FIL_RUNOUT_ENABLED_DEFAULT true // Enable the sensor on startup. Override with M412 followed by M500.
1276 #define NUM_RUNOUT_SENSORS 1 // Number of sensors, up to one per extruder. Define a FIL_RUNOUT_PIN for each.
1277
1278 #define FIL_RUNOUT_STATE LOW // Pin state indicating that filament is NOT present.
1279 #define FIL_RUNOUT_PULLUP // Use internal pullup for filament runout pins.
1280 // #define FIL_RUNOUT_PULLDOWN // Use internal pulldown for filament runout pins.
1281 // #define WATCH_ALL_RUNOUT_SENSORS // Execute runout script on any triggering sensor, not only for the active extruder.
1282 // This is automatically enabled for MIXING_EXTRUDERS.
  
```

#define FILAMENT_RUNOUT_SENSOR // include filament runout detection in the firmware
 #define FIL_RUNOUT_ENABLED_DEFAULT true // the filament runout detection is ON by default. You can disable it in "Configuration->Runout Sensor" and save setting in "Configuration->Store Settings" on the LCD12864, or disable it with "M412 S0" followed by "M500".

#define NUM_RUNOUT_SENSORS 1 // number of filament runout sensors
 #define FIL_RUNOUT_STATE LOW // If the filament runs out, the module produces a low logic level and the state is set to "runout".

- BIGTREETECH Smart Filament Sensor – This sensor continuously sends a pulse train to the motherboard when filament is passing through it. When there is no filament passing through it or when then filament jams it will stop sending a pulse train to the motherboard, thereby allowing the motherboard to detect a runout or jammed condition.

```

C Configuration.h x C Configuration_adv.h
Marlin > C Configuration.h > FILAMENT_MOTION_SENSOR
1273 #define FILAMENT_RUNOUT_SENSOR
1274 #if ENABLED(FILAMENT_RUNOUT_SENSOR)
1275   #define FIL_RUNOUT_ENABLED_DEFAULT true // Enable the sensor on startup. Override with M412 followed by M500.
1276   #define NUM_RUNOUT_SENSORS 1 // Number of sensors, up to one per extruder. Define a FIL_RUNOUT#_PIN for each.
1277
1278   #define FIL_RUNOUT_STATE LOW // Pin state indicating that filament is NOT present.
1279   #define FIL_RUNOUT_PULLUP // Use internal pullup for filament runout pins...
1280
1281   // Override individually if the runout sensors vary...
1282
1283   // #define FIL_RUNOUT2_STATE LOW...
1284   // #define FIL_RUNOUT3_STATE LOW...
1285   // #define FIL_RUNOUT4_STATE LOW...
1286   // #define FIL_RUNOUT5_STATE LOW...
1287   // #define FIL_RUNOUT6_STATE LOW...
1288   // #define FIL_RUNOUT7_STATE LOW...
1289   // #define FIL_RUNOUT8_STATE LOW...
1290
1291   // Commands to execute on filament runout. ...
1292   #define FILAMENT_RUNOUT_SCRIPT "M600"
1293
1294   // After a runout is detected, continue printing this length of filament
1295   // before executing the runout script. Useful for a sensor at the end of
1296   // a feed tube. Requires 4 bytes SRAM per sensor, plus 4 bytes overhead.
1297   #define FILAMENT_RUNOUT_DISTANCE_MM 7
1298
1299   #ifdef FILAMENT_RUNOUT_DISTANCE_MM
1300     // Enable this option to use an encoder disc that toggles the runout pin
1301     // as the filament moves. (Be sure to set FILAMENT_RUNOUT_DISTANCE_MM
1302     // long enough to avoid false positives.)
1303     #define FILAMENT_MOTION_SENSOR
1304   #endif
1305 #endif
1306
1307 #endif
1308
1309 #endif

```

#define FILAMENT_MOTION_SENSOR // set the type of filament runout sensor

#define FILAMENT_RUNOUT_DISTANCE_MM 7 // This allows for 7mm of filament to be extruded without the sensor producing a pulse. You may need to change this in the firmware to suit your setup if you find that you are getting false triggers.

If a runout or blockage is detected, then the firmware can park the nozzle while you correct the issue. Make the following changes to enable this feature.

```

C Configuration.h x C Configuration_adv.h
Marlin > C Configuration.h > NOZZLE_PARK_FEATURE
1696 /**
1697  * Nozzle Park
1698  *
1699  * Park the nozzle at the given XYZ position on idle or G27.
1700  *
1701  * The "P" parameter controls the action applied to the Z axis:
1702  *
1703  * P0 (Default) If Z is below park Z raise the nozzle.
1704  * P1 Raise the nozzle always to Z-park height.
1705  * P2 Raise the nozzle by Z-park amount, limited to Z_MAX_POS.
1706  */
1707 #define NOZZLE_PARK_FEATURE
1708
1709 #if ENABLED(NOZZLE_PARK_FEATURE)
1710   // Specify a park position as { X, Y, Z_raise }
1711   #define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 }
1712   // #define NOZZLE_PARK_X_ONLY // X move only is required to park
1713   // #define NOZZLE_PARK_Y_ONLY // Y move only is required to park
1714   #define NOZZLE_PARK_Z_RAISE_MIN 2 // (mm) Always raise Z by at least this distance
1715   #define NOZZLE_PARK_XY_FEEDRATE 100 // (mm/s) X and Y axes feedrate (also used for delta Z axis)
1716   #define NOZZLE_PARK_Z_FEEDRATE 5 // (mm/s) Z axis feedrate (not used for delta printers)
1717 #endif
1718

```

#define NOZZLE_PARK_FEATURE // nozzle pause feature

#define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 } //specify the XY position and Z-

raise height when the nozzle pauses

`#define ADVANCED_PAUSE_FEATURE` // set the filament retraction length and speed when it pauses, and filament unload length once the nozzle reaches the park point.

```
Configuration.h Configuration_adv.h X
Marlin > Configuration_adv.h > ADVANCED_PAUSE_FEATURE
2180 /**
2181  * Advanced Pause
2182  * Experimental feature for filament change support and for parking the nozzle when paused.
2183  * Adds the GCode M600 for initiating filament change.
2184  * If PARK_HEAD_ON_PAUSE enabled, adds the GCode M125 to pause printing and park the nozzle.
2185  *
2186  * Requires an LCD display.
2187  * Requires NOZZLE_PARK_FEATURE.
2188  * This feature is required for the default FILAMENT_RUNOUT_SCRIPT.
2189  */
2190 #define ADVANCED_PAUSE_FEATURE
2191 #if ENABLED(ADVANCED_PAUSE_FEATURE)
2192   #define PAUSE_PARK_RETRACT_FEEDRATE 60 // (mm/s) Initial retract feedrate.
2193   #define PAUSE_PARK_RETRACT_LENGTH 2 // (mm) Initial retract.
2194   // This short retract is done immediately, before parking the nozzle.
2195   #define FILAMENT_CHANGE_UNLOAD_FEEDRATE 10 // (mm/s) Unload filament feedrate. This can be pretty fast.
2196   #define FILAMENT_CHANGE_UNLOAD_ACCEL 25 // (mm/s^2) Lower acceleration may allow a faster feedrate.
2197   #define FILAMENT_CHANGE_UNLOAD_LENGTH 400 // (mm) The length of filament for a complete unload.
2198   // For Bowden, the full length of the tube and nozzle.
2199   // For direct drive, the full length of the nozzle.
2200   // Set to 0 for manual unloading.
2201   #define FILAMENT_CHANGE_SLOW_LOAD_FEEDRATE 6 // (mm/s) Slow move when starting load.
2202   #define FILAMENT_CHANGE_SLOW_LOAD_LENGTH 0 // (mm) Slow length, to allow time to insert material.
2203   // 0 to disable start loading and skip to fast load only
2204   #define FILAMENT_CHANGE_FAST_LOAD_FEEDRATE 6 // (mm/s) Load filament feedrate. This can be pretty fast.
2205   #define FILAMENT_CHANGE_FAST_LOAD_ACCEL 25 // (mm/s^2) Lower acceleration may allow a faster feedrate.
2206   #define FILAMENT_CHANGE_FAST_LOAD_LENGTH 350 // (mm) Load length of filament, from extruder gear to nozzle.
2207   // For Bowden, the full length of the tube and nozzle.
2208   // For direct drive, the full length of the nozzle.
2209   // #define ADVANCED_PAUSE_CONTINUOUS_PURGE // Purge continuously up to the purge length until interrupted.
2210   #define ADVANCED_PAUSE_PURGE_FEEDRATE 3 // (mm/s) Extrude feedrate (after loading). Should be slower than load feedrate.
2211   #define ADVANCED_PAUSE_PURGE_LENGTH 50 // (mm) Length to extrude after loading.
```

Fan configuration

For the Ender3, by default, the cooling fan for the hotend is always-on and directly connected to the 24V power supply.

The FAN1 port on the motherboard is a PWM controllable fan. By default, the motherboard cooling fan is connected to the FAN1 port and you can turn this fan on to cool the motherboard during prints or when the bed is being heated. Use the following firmware configuration to do so.

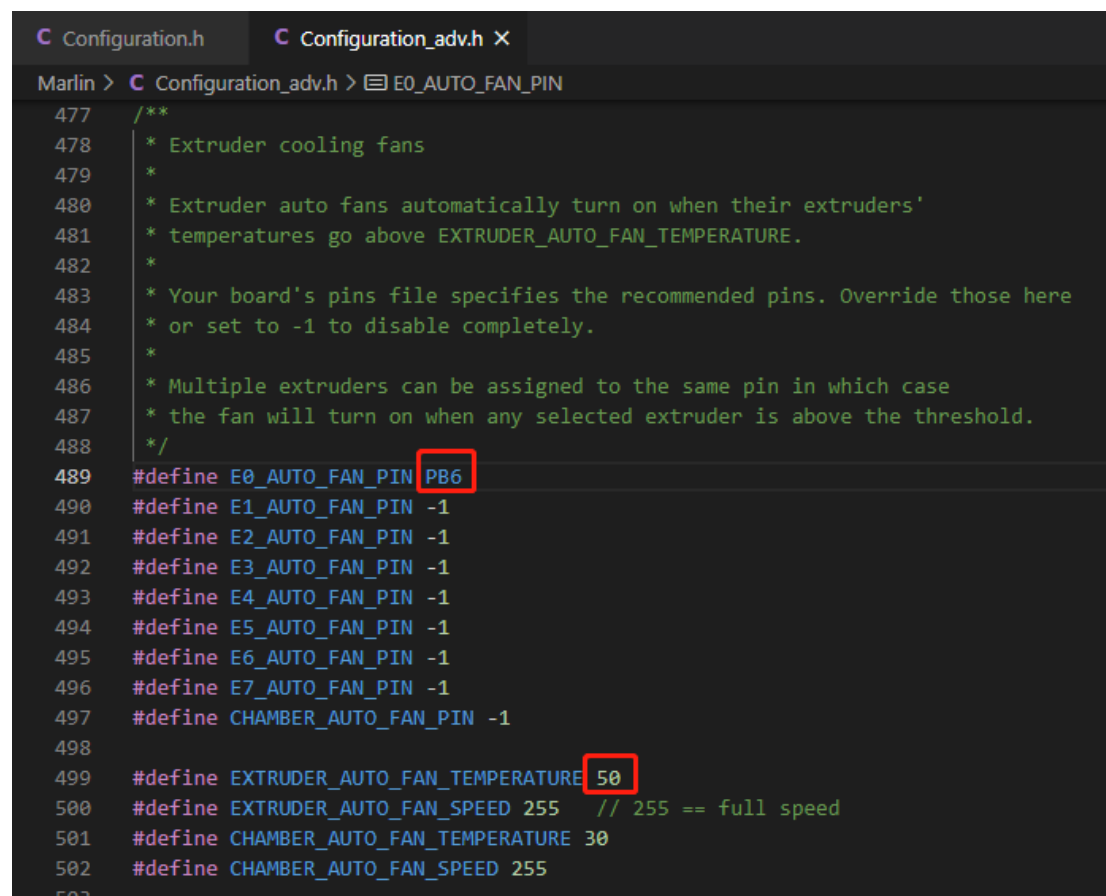
```
Configuration.h Configuration_adv.h X
Marlin > Configuration_adv.h > USE_CONTROLLER_FAN
402
403 /**
404  * Controller Fan
405  * To cool down the stepper drivers and MOSFETs.
406  *
407  * The fan turns on automatically whenever any driver is enabled and turns
408  * off (or reduces to idle speed) shortly after drivers are turned off.
409  */
410 #define USE_CONTROLLER_FAN
411 #if ENABLED(USE_CONTROLLER_FAN)
412   // #define CONTROLLER_FAN_PIN -1 // Set a custom pin for the controller fan
413   // #define CONTROLLER_FAN_USE_Z_ONLY // With this option only the Z axis is considered
414   // #define CONTROLLER_FAN_IGNORE_Z // Ignore Z stepper. Useful when stepper timeout is disabled.
415   #define CONTROLLERFAN_SPEED_MIN 0 // (0-255) Minimum speed. (If set below this value the fan is turned off.)
416   #define CONTROLLERFAN_SPEED_ACTIVE 255 // (0-255) Active speed, used when any motor is enabled
417   #define CONTROLLERFAN_SPEED_IDLE 0 // (0-255) Idle speed, used when motors are disabled
418   #define CONTROLLERFAN_IDLE_TIME 60 // (seconds) Extra time to keep the fan running after disabling motors
419   #define CONTROLLER_FAN_EDITABLE // Enable M710 configurable settings
420   #if ENABLED(CONTROLLER_FAN_EDITABLE)
421     #define CONTROLLER_FAN_MENU // Enable the Controller Fan submenu
422   #endif
423 #endif
424
```

If, instead, you want to control the hotend cooling fan, you can connect the hotend to the FAN1 port and then apply the configuration below. Note that you can also connect the motherboard cooling fan in parallel with the hotend cooling fan using the configuration below since it will cause the motherboard fan to turn on when the hotend is on which is the general condition during a print when the motherboard needs cooling.

First, disable the config that only enables the motherboard cooling fan:

```
#define USE_CONTROLLER_FAN
```

Next, enable config that will allow the hotend cooling fan to be controllable by setting the E0_AUTO_FAN_PIN as shown below.



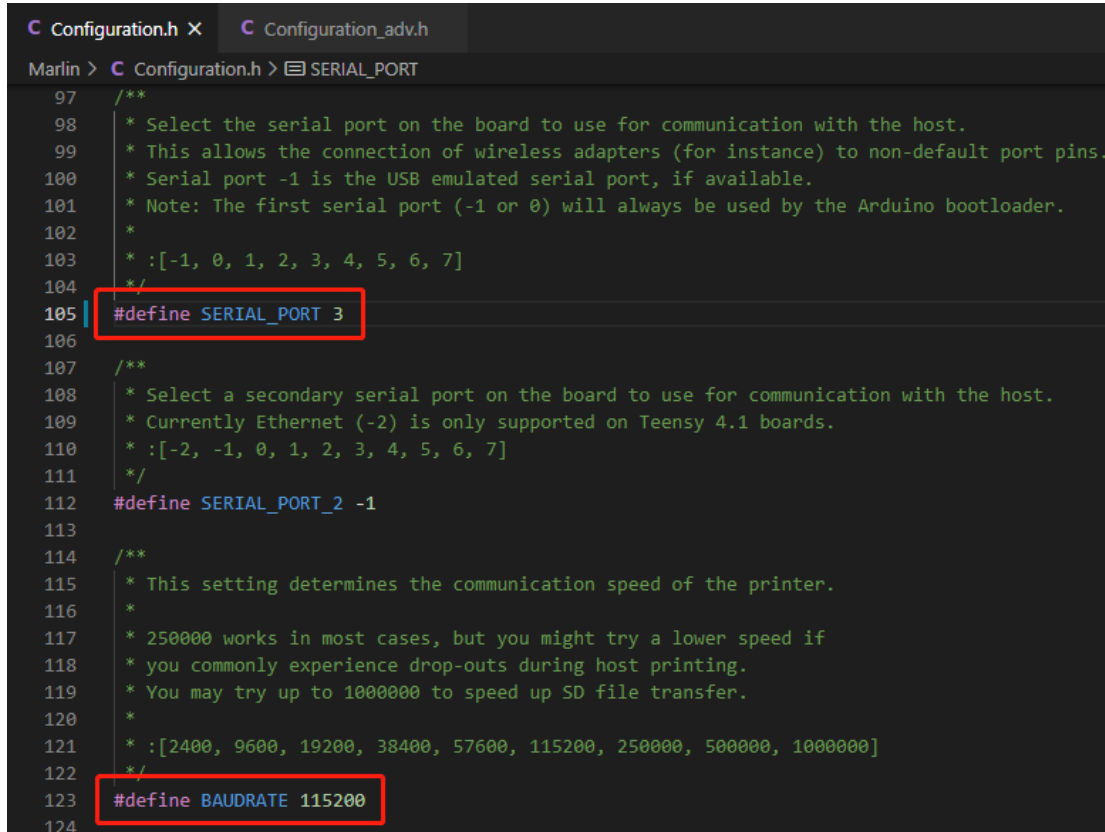
```
Configuration.h Configuration_adv.h X
Marlin > C Configuration_adv.h > E0_AUTO_FAN_PIN
477 /**
478  * Extruder cooling fans
479  *
480  * Extruder auto fans automatically turn on when their extruders'
481  * temperatures go above EXTRUDER_AUTO_FAN_TEMPERATURE.
482  *
483  * Your board's pins file specifies the recommended pins. Override those here
484  * or set to -1 to disable completely.
485  *
486  * Multiple extruders can be assigned to the same pin in which case
487  * the fan will turn on when any selected extruder is above the threshold.
488  */
489 #define E0_AUTO_FAN_PIN PB6
490 #define E1_AUTO_FAN_PIN -1
491 #define E2_AUTO_FAN_PIN -1
492 #define E3_AUTO_FAN_PIN -1
493 #define E4_AUTO_FAN_PIN -1
494 #define E5_AUTO_FAN_PIN -1
495 #define E6_AUTO_FAN_PIN -1
496 #define E7_AUTO_FAN_PIN -1
497 #define CHAMBER_AUTO_FAN_PIN -1
498
499 #define EXTRUDER_AUTO_FAN_TEMPERATURE 50
500 #define EXTRUDER_AUTO_FAN_SPEED 255 // 255 == full speed
501 #define CHAMBER_AUTO_FAN_TEMPERATURE 30
502 #define CHAMBER_AUTO_FAN_SPEED 255
503
```

```
#define E0_AUTO_FAN_PIN PB6 // set PB6(FAN1) to E0 AUTO FAN
```

```
#define EXTRUDER_AUTO_FAN_TEMPERATURE 50 // turn on the cooling fan when the nozzle temperature exceeds 50
```

ESP3D

The onboard ESP8266 module allows Marlin to use the Wi-Fi control panel of the open source ESP3D software without any additional hardware. Just set the correct "SERIAL_PORT" and "BAUDRATE" in Marlin. The serial port for communication between ESP8266 onboard and Marlin is UART3, so SERIAL_PORT is set to 3.



```
197  /**
198  * Select the serial port on the board to use for communication with the host.
199  * This allows the connection of wireless adapters (for instance) to non-default port pins.
200  * Serial port -1 is the USB emulated serial port, if available.
201  * Note: The first serial port (-1 or 0) will always be used by the Arduino bootloader.
202  *
203  * :[-1, 0, 1, 2, 3, 4, 5, 6, 7]
204  */
205  #define SERIAL_PORT 3
206
207  /**
208  * Select a secondary serial port on the board to use for communication with the host.
209  * Currently Ethernet (-2) is only supported on Teensy 4.1 boards.
210  * :[-2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
211  */
212  #define SERIAL_PORT_2 -1
213
214  /**
215  * This setting determines the communication speed of the printer.
216  *
217  * 250000 works in most cases, but you might try a lower speed if
218  * you commonly experience drop-outs during host printing.
219  * You may try up to 1000000 to speed up SD file transfer.
220  *
221  * :[2400, 9600, 19200, 38400, 57600, 115200, 250000, 500000, 1000000]
222  */
223  #define BAUDRATE 115200
224
```

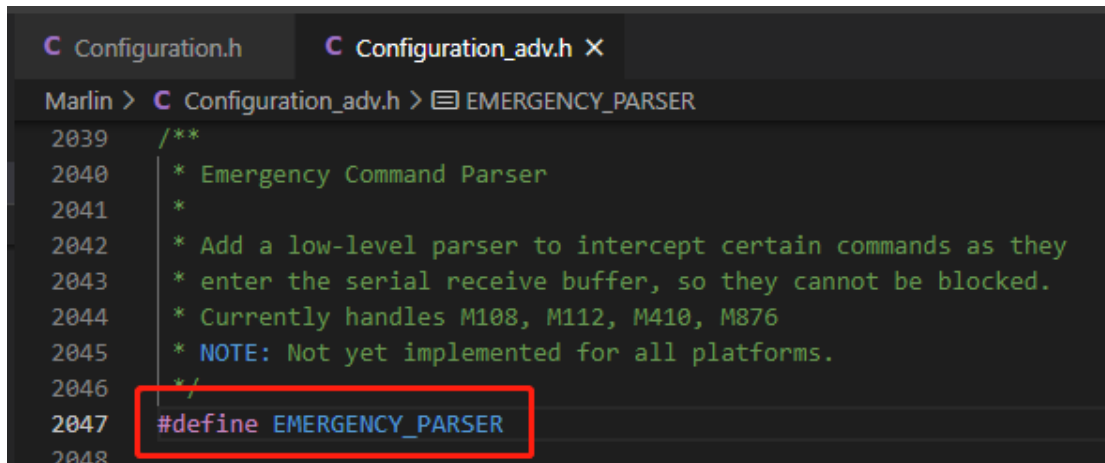
If you would like to update the firmware on the ESP module, you can download the latest ESP3D firmware at <https://github.com/luc-github/ESP3D> and compile your binary file. Rename it "ESP3D.bin", copy it to the root directory in the microSD card, and then reset the motherboard. The bootloader on the motherboard will automatically update ESP3D.bin to the ESP8266, and the file will be renamed "ESP3D.CUR" after the update has completed.

Notes:

If you use the BTT TFT serial screen and connect any of the above-mentioned expansion modules to the motherboard, you need to enable the following within Marlin:

```
#define EMERGENCY_PARSER
```

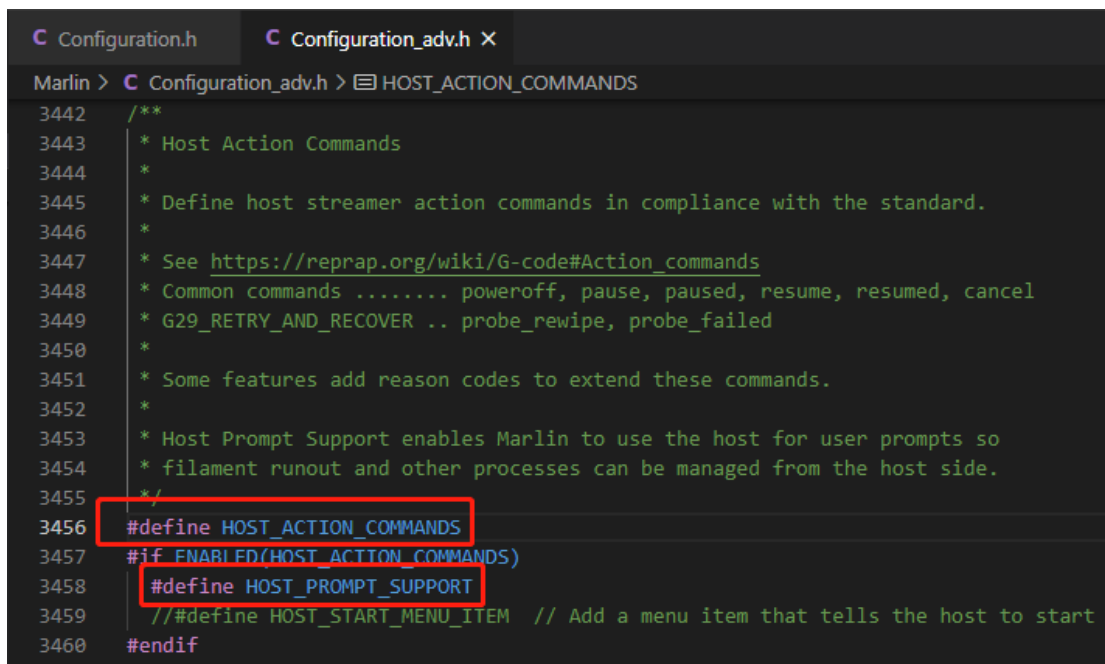
```
#define HOST_ACTION_COMMANDS
```



The screenshot shows the Marlin Configuration_adv.h file in a code editor. The file is open to the EMERGENCY_PARSER section. The code is as follows:

```
2039  /**
2040  * Emergency Command Parser
2041  *
2042  * Add a low-level parser to intercept certain commands as they
2043  * enter the serial receive buffer, so they cannot be blocked.
2044  * Currently handles M108, M112, M410, M876
2045  * NOTE: Not yet implemented for all platforms.
2046  */
2047  #define EMERGENCY_PARSER
2048
```

The line `#define EMERGENCY_PARSER` is highlighted with a red box.



The screenshot shows the Marlin Configuration_adv.h file in a code editor. The file is open to the HOST_ACTION_COMMANDS section. The code is as follows:

```
3442  /**
3443  * Host Action Commands
3444  *
3445  * Define host streamer action commands in compliance with the standard.
3446  *
3447  * See https://reprap.org/wiki/G-code#Action\_commands
3448  * Common commands ..... poweroff, pause, paused, resume, resumed, cancel
3449  * G29_RETRY_AND_RECOVER .. probe_rewipe, probe_failed
3450  *
3451  * Some features add reason codes to extend these commands.
3452  *
3453  * Host Prompt Support enables Marlin to use the host for user prompts so
3454  * filament runout and other processes can be managed from the host side.
3455  */
3456  #define HOST_ACTION_COMMANDS
3457  #if ENABLED(HOST_ACTION_COMMANDS)
3458    #define HOST_PROMPT_SUPPORT
3459    // #define HOST_START_MENU_ITEM // Add a menu item that tells the host to start
3460  #endif
```

The lines `#define HOST_ACTION_COMMANDS` and `#define HOST_PROMPT_SUPPORT` are highlighted with red boxes.

Additionally, you will need to set the following parameters in the config.ini file of the TFT:

```
start_gcode_enabled:1
```

```
end_gcode_enabled:1
```

```
cancel_gcode_enabled:1
```



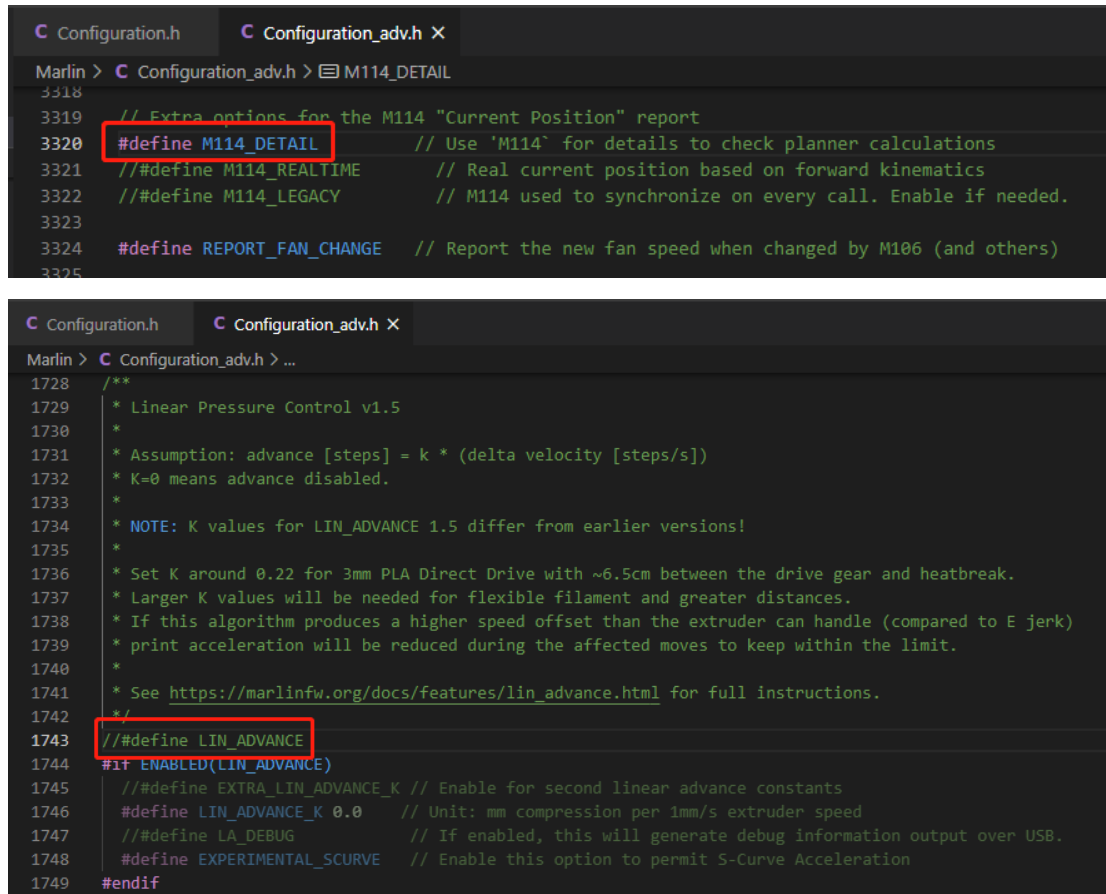
```
config.ini X
C: > Users > Administrator > Desktop > config.ini
606 ##### Default Start Gcode Status
607 # Options: [enable: 1, disable: 0]
608 start_gcode_enabled:1
609
610 ##### Default End Gcode Status
611 # Options: [enable: 1, disable: 0]
612 end_gcode_enabled:1
613
614 ##### Default Cancel Gcode Status
615 # Options: [enable: 1, disable: 0]
616 cancel_gcode_enabled:1
617
618 ##### Start Gcode
619 # This gcode will runs before starting a print if `start_gcode_enabled` is enabled.
620 # Value range: [max 75 characters]
621 start_gcode:G28 XY R20\nM75\n
622
623 ##### End Gcode
624 # This gcode will runs after a print is completed if `end_gcode_enabled` is enabled.
625 # Value range: [max 75 characters]
626 end_gcode:M77\nM104 S0\nM140 S0\nM107\nM18\n
627
628 ##### Cancel Gcode
629 # This gcode will runs when a print is canceled if `cancel_gcode_enabled` is enabled.
630 # Value range: [max 75 characters]
631 cancel_gcode:M77\nM104 S0\nM140 S0\nG28 XY R10\nM107\nM18\n
632
```

Other settings are also required on the TFT to make power loss recovery module compatible with the touch screen.

```
config.ini X
C: > Users > Administrator > Desktop > config.ini
486
487 ##### Default Power Loss Recovery Mode
488 # Enabled by default.
489 # Disable to reduce the loss of SD card or U disk.
490 # Options: [enable: 1, disable: 0]
491 pl_recovery_en:1
492
493 ##### Power Loss Recovery Homing
494 # Home before power loss recovery.
495 # Options: [enable: 1, disable: 0]
496 pl_recovery_home:0
497
498 ##### Power Loss Z Raise
499 # Raise Z axis on resume (on power loss with UPS).
500 # Unit: [height in mm]
501 pl_z_raise:10
502
503 ##### BTT UPS Support
504 # Enable BTT UPS.
505 # Options: [enable: 1, disable: 0]
506 btt_mini_ups:1
507
```

pl_z_raise should be as high as #define POWER_LOSS_ZRAISE set in Marlin

If connecting smart filament sensor to the touch screen, you should enable “#define M114_DETAIL” and disable #define LIN_ADVANCE”. LIN_ADVANCE in Marlin. At the time of writing, linear advance causes an error in the reporting of the extruded distance to the TFT.



```
Configuration.h Configuration_adv.h X
Marlin > Configuration_adv.h > M114_DETAIL
3318
3319 // Extra options for the M114 "Current Position" report
3320 #define M114_DETAIL // Use 'M114' for details to check planner calculations
3321 // #define M114_REALTIME // Real current position based on forward kinematics
3322 // #define M114_LEGACY // M114 used to synchronize on every call. Enable if needed.
3323
3324 #define REPORT_FAN_CHANGE // Report the new fan speed when changed by M106 (and others)
3325

Configuration.h Configuration_adv.h X
Marlin > Configuration_adv.h > ...
1728 /**
1729  * Linear Pressure Control v1.5
1730  *
1731  * Assumption: advance [steps] = k * (delta velocity [steps/s])
1732  * K=0 means advance disabled.
1733  *
1734  * NOTE: K values for LIN_ADVANCE 1.5 differ from earlier versions!
1735  *
1736  * Set K around 0.22 for 3mm PLA Direct Drive with ~6.5cm between the drive gear and heatbreak.
1737  * Larger K values will be needed for flexible filament and greater distances.
1738  * If this algorithm produces a higher speed offset than the extruder can handle (compared to E jerk)
1739  * print acceleration will be reduced during the affected moves to keep within the limit.
1740  *
1741  * See https://marlinfw.org/docs/features/lin\_advance.html for full instructions.
1742  */
1743 ##define LIN_ADVANCE
1744 #if ENABLED(LIN_ADVANCE)
1745   // #define EXTRA_LIN_ADVANCE_K // Enable for second linear advance constants
1746   #define LIN_ADVANCE_K 0.0 // Unit: mm compression per 1mm/s extruder speed
1747   // #define LA_DEBUG // If enabled, this will generate debug information output over USB.
1748   #define EXPERIMENTAL_SCURVE // Enable this option to permit S-Curve Acceleration
1749 #endif
```