MARLIN 3D PRINTER FRIMWARE CONFIGURATION

Configuring Marlin

Marlin is a huge C++ program composed of many files, but here we'll only be talking about the two files that contain all of Marlin's compile-time configuration options:

- `Configuration.h` contains the core settings for the hardware, language and controller selection, and settings for the most common features and components.
- `Configuration_adv.h` serves up more detailed customization options, add-ons, experimental features, and other esoterica.

These two files contain all of Marlin's build-time configuration options. Simply edit or replace these files before building and uploading Marlin to the board. A variety of pre-built configurations are included in the `config/examples` folder to get you started.

To use configurations from an earlier version of Marlin, try dropping them into the newer Marlin and building. As part of the build process, the `SanityCheck.h` will print helpful error messages explaining what needs to be changed.

Also a tool like Winmerge is usefull to compare the old file to the new one and you can copy over the settings.

Compiler Directives

Marlin is configured using C++ compiler directives. This allows Marlin to leverage the C++ preprocessor and include only the code and data needed for the enabled options. This results in the smallest possible binary. A build of Marlin can range from 50K to over 230K in size.

Settings can be enabled, disabled, and assigned values using C preprocessor syntax like so:

```
#define THIS_IS_ENABLED    // this switch is enabled

//#define THIS_IS_DISABLED // this switch is disabled

#define OPTION_VALUE 22    // this setting is "22"
```

Sources of Documentation

The most authoritative source on configuration details will always be **the configuration files themselves**. They provide good descriptions of each option, and are themselves the source for most of the information presented here.

If you've never configured and calibrated a 3D Printer before, here are some good resources:

- Calibration

- Calibrating Steps-per-unit
- Prusa's calculators
- Triffid Hunter's Calibration Guide
- The Essential Calibration Set
- Calibration of your RepRap
- XY 20 mm Calibration Box
- G-code reference
- Marlin3DprinterTool

Before You Begin

To get your core `Configuration.h` settings right you'll need to know the following things about your printer:

- Printer style, such as Cartesian, Delta, CoreXY, or SCARA
- Driver board, such as RAMPS, RUMBA, Teensy, etc.
- Number of extruders
- Steps-per-mm for XYZ axes and extruders (can be tuned later)
- Endstop positions
- Thermistors and/or thermocouples
- Probes and probing settings
- LCD controller brand and model
- Add-ons and custom components

`Configuration.h`

The core and default settings of Marlin live in the `Configuration.h` file. Most of these settings are fixed. Once you compile Marlin, that's it. To change them you need to re-compile. However, several items in `Configuration.h` only provide defaults -factory settings- that can be changed via the user interface, stored on EEPROM and reloaded or restored to initial values.

Settings that can be changed and saved to EEPROM are marked with . Options marked with can be changed from the LCD controller.

Settings saved in EEPROM persist across reboots and still remain after flashing new firmware, so always send `M502`, `M500` (or "Reset EEPROM" from the LCD) after flashing.

This section follows the order of settings as they appear. The order isn't always logical, so "Search In Page" may be helpful. We've tried to keep descriptions brief and to the point. For more detailed information on various topics, please read the main articles and follow the links provided in the option descriptions.

Configuration versioning

```
#define CONFIGURATION_H_VERSION 020000
```

Marlin now checks for a configuration version and won't compile without this setting. If you want to upgrade from an earlier version of Marlin, add this line to your old configuration file. During compilation, Marlin will throw errors explaining what needs to be changed.

Firmware Info

```
#define STRING_CONFIG_H_AUTHOR "(none, default config)"



#define SHOW_BOOTSCREEN

#define SHOW_CUSTOM_BOOTSCREEN

#define CUSTOM_STATUS_SCREEN_IMAGE
```

- `STRING_CONFIG_H_AUTHOR` is shown in the Marlin startup message, and is meant to identify the author (and optional variant) of the firmware. Use this setting as a way to uniquely identify all your custom configurations. The startup message is printed when connecting to host software, when the board reboots and M115.
- `SHOW_BOOTSCREEN` enables the boot screen for LCD controllers.
- `SHOW_CUSTOM_BOOTSCREEN` shows the bitmap in Marlin/_Bootscreen.h on startup.
- `CUSTOM_STATUS_SCREEN_IMAGE` shows the bitmap in Marlin/_Statusscreen.h on the status screen.

Hardware Info

Serial Port

```
#define SERIAL_PORT 0
```

The index of the on-board serial port that will be used for primary host communication. Change this if, for example, you need to connect a wireless adapter to non-default port pins.

The first serial port (-1 or 0) will always be used by the Arduino bootloader regardless of this setting.

```
#define SERIAL_PORT_2 -1
```

Enable this if your board has a secondary serial port.

Serial port -1 is the USB emulated serial port, if available.

Baud Rate

```
#define BAUDRATE 115200
```

The serial communication speed of the printer should be as fast as it can manage without generating errors. In most cases 115200 gives a good balance between speed and stability. Start with 250000 and only go lower if “line number” and “checksum” errors start to appear. Note that some boards (e.g., a temperamental Sanguinololu clone based on the ATMEGA1284P) may not be able to handle a baudrate over 57600. Allowed values: 2400, 9600, 19200, 38400, 57600, 115200, 250000.

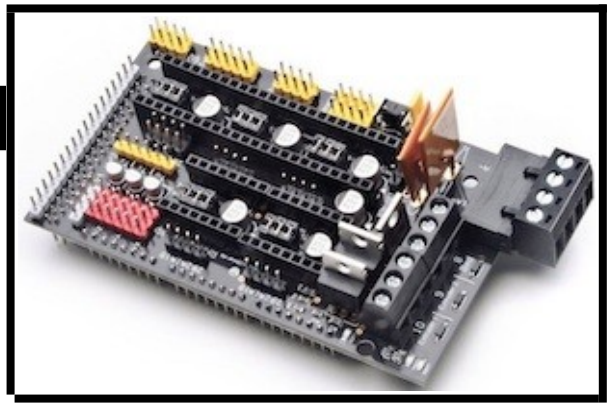
Bluetooth

```
#define BLUETOOTH
```

Enable the Bluetooth serial interface. For boards based on the AT90USB.

Motherboard

```
#define MOTHERBOARD BOARD_RAMPS_14_EFB
```



The most important setting is Marlin is the motherboard. The firmware needs to know what board it will be running on so it can assign the right functions to all pins and take advantage of the full capabilities of the board. Setting this incorrectly will lead to unpredictable results.

Using `boards.h` as a reference, replace `BOARD_RAMPS_14_EFB` with your board’s ID.
The `boards.h` file has the most up-to-date listing of supported boards, so check it first if you don’t see yours listed there.

The Sanguino board requires adding “Sanguino” support to Arduino IDE. Open `Preferences` and locate the `Additional Boards Manager URLs` field. Copy and paste this source URL. Then use `Tools` > `Boards` > `Boards Manager` to install “Sanguino” from the list. An internet connection is required. (Thanks to “Dust’s RepRap Blog” for the tip.)

Custom Machine Name

```
//#define CUSTOM_MACHINE_NAME "3D Printer"
```

This is the name of your printer as displayed on the LCD and by `M115`. For example, if you set this to “My Delta” the LCD will display “My Delta ready” when the printer starts up.

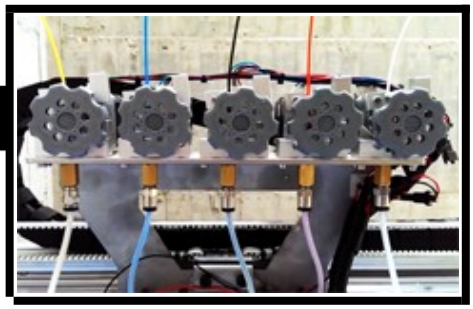
Machine UUID

```
//#define MACHINE_UUID "00000000-0000-0000-0000-000000000000"
```

A unique ID for your 3D printer. A suitable unique ID can be generated randomly at uuidtools.com. Some host programs and slicers may use this identifier to differentiate between specific machines on your network.

Extruder Info

Extruders



```
#define EXTRUDERS 1
```

This value, from 0 to 6, defines how many extruders (or E steppers) the printer has. By default Marlin will assume separate nozzles all moving together on a single carriage. If you have a single nozzle, a switching extruder, a mixing extruder, or dual X carriages, specify that below.

This value should be set to the total number of E stepper motors on the machine, even if there's only a single nozzle.

Filament Diameter

```
#define DEFAULT_NOMINAL_FILAMENT_DIA 3.00
```

This is the "nominal" filament diameter as written on the filament spool (1.75, 2.85, 3.0). If you typically use 1.75mm filament, but physically measure the diameter as 1.70mm, you should still use 1.75 if that's what you have set in your slicer.

This value is used by Marlin to compensate for Filament Width when printing in volumetric mode (See `M200`), and by the Unified Bed Leveling command `G26` when printing a test grid.

You can override this value with `M404 W`.

Single Nozzle

```
#define SINGLENOZZLE
```

Enable `SINGLENOZZLE` if you have an E3D Cyclops or any other "multi-extruder" system that shares a single nozzle. In a single-nozzle setup, only one filament drive is engaged at a time, and each needs to retract before the next filament can be loaded and begin purging and extruding.

###Průša MK2 Single Nozzle Multi-Material Multiplexer

```
//#define MK2_MULTIPLEXER
```

Enabling `MK2_MULTIPLEXER` allows one stepper driver on a control board to drive two to eight stepper motors, one at a time.

```
//#define E_MUX0_PIN 40  // Always Required
```

```
//#define E_MUX1_PIN 42  // Needed for 3 to 8 inputs

//#define E_MUX2_PIN 44  // Needed for 5 to 8 inputs
```

Override the default DIO selector pins.

Prusa MMU2

```
#define PRUSA_MMU2
```

Enable support for the Prusa Multi-material unit 2. This requires a free serial port on your printer board. To use the MMU2 you also have to

- enable NOZZLE_PARK_FEATURE
- set EXTRUDERS = 5

All details are configured in [Configuration_adv.h]

Switching Extruder

```
//#define SWITCHING_EXTRUDER

#if ENABLED(SWITCHING_EXTRUDER)

  #define SWITCHING_EXTRUDER_SERVO_NR 0

  #define SWITCHING_EXTRUDER_SERVO_ANGLES { 0, 90 } // Angles for E0, E1[, E2, E3]

  #if EXTRUDERS > 3

    #define SWITCHING_EXTRUDER_E23_SERVO_NR 1

  #endif

#endif
```

A Switching Extruder is a dual extruder that uses a single stepper motor to drive two filaments, but only one at a time. The servo is used to switch the side of the extruder that will drive the filament. The E motor also reverses direction for the second filament. Set the servo sub-settings above according to your particular extruder's setup instructions.

Switching Nozzle

```
//#define SWITCHING_NOZZLE

#if ENABLED(SWITCHING_NOZZLE)

  #define SWITCHING_NOZZLE_SERVO_NR 0
```

```
  //#define SWITCHING_NOZZLE_E1_SERVO_NR 1          // If two servos are used, the index of the second

  #define SWITCHING_NOZZLE_SERVO_ANGLES { 0, 90 }   // Angles for E0, E1 (single servo) or lowered/raised (dual servo)

#endif
```

A Switching Nozzle is a carriage with 2 nozzles. A servo is used to move one of the nozzles up and down. The servo either lowers the active nozzle or raises the inactive one. Set the servo sub-settings above according to your particular extruder's setup instructions.

Parking extruder (with solenoid)

```
//#define PARKING_EXTRUDER
```

Two separate X-carriages with extruders that connect to a moving part via a solenoid docking mechanism. Requires SOL1_PIN and SOL2_PIN.

Parking extruder (with magnets)

```
//#define MAGNETIC_PARKING_EXTRUDER
```

Two separate X-carriages with extruders that connect to a moving part via a magnetic docking mechanism using movements and no solenoid

```
#if EITHER(PARKING_EXTRUDER, MAGNETIC_PARKING_EXTRUDER)



  #define PARKING_EXTRUDER_PARKING_X { -78, 184 }     // X positions for parking the extruders

  #define PARKING_EXTRUDER_GRAB_DISTANCE 1            // (mm) Distance to move beyond the parking point to grab the extruder

  //#define MANUAL_SOLENOID_CONTROL                   // Manual control of docking solenoids with M380 S / M381



  #if ENABLED(PARKING_EXTRUDER)



    #define PARKING_EXTRUDER_SOLENOIDS_INVERT           // If enabled, the solenoid is NOT magnetized with applied voltage
```

```
    #define PARKING_EXTRUDER_SOLENOIDS_PINS_ACTIVE LOW  // LOW or HIGH pin
signal energizes the coil

    #define PARKING_EXTRUDER_SOLENOIDS_DELAY 250        // (ms) Delay for
magnetic field. No delay if 0 or not defined.

    //#define MANUAL_SOLENOID_CONTROL                   // Manual control of
docking solenoids with M380 S / M381



  #elif ENABLED(MAGNETIC_PARKING_EXTRUDER)



    #define MPE_FAST_SPEED      9000      // (mm/m) Speed for travel before
last distance point

    #define MPE_SLOW_SPEED      4500      // (mm/m) Speed for last distance
travel to park and couple

    #define MPE_TRAVEL_DISTANCE   10      // (mm) Last distance point

    #define MPE_COMPENSATION       0      // Offset Compensation -1 , 0 , 1
(multiplier) only for coupling



  #endif



#endif
```

Adjust the relavant settings to your specifications for use with either `PARKING_EXTRUDER` or `MAGNETIC_PARKING_EXTRUDER`.

Switching Toolhead

```
//#define SWITCHING_TOOLHEAD
```

Support for swappable and dockable toolheads, such as the E3D Tool Changer. Toolheads are locked with a servo.

Magnetic Switching Toolhead

```
//#define MAGNETIC_SWITCHING_TOOLHEAD
```

Support swappable and dockable toolheads with a magnetic docking mechanism using movement and no servo.

Electromagnetic Switching Toolhead

```
//#define ELECTROMAGNETIC_SWITCHING_TOOLHEAD
```

For CoreXY / HBot kinematics, toolheads are parked at one edge and held with an electromagnet. Supports more than 2 Toolheads. See https://youtu.be/JolbsAKTKf4

```
#if ANY(SWITCHING_TOOLHEAD, MAGNETIC_SWITCHING_TOOLHEAD,
ELECTROMAGNETIC_SWITCHING_TOOLHEAD)

  #define SWITCHING_TOOLHEAD_Y_POS          235         // (mm) Y position of
the toolhead dock

  #define SWITCHING_TOOLHEAD_Y_SECURITY      10         // (mm) Security
distance Y axis

  #define SWITCHING_TOOLHEAD_Y_CLEAR         60         // (mm) Minimum
distance from dock for unobstructed X axis

  #define SWITCHING_TOOLHEAD_X_POS          { 215, 0 }  // (mm) X positions for
parking the extruders

  #if ENABLED(SWITCHING_TOOLHEAD)

    #define SWITCHING_TOOLHEAD_SERVO_NR       2         // Index of the servo
connector

    #define SWITCHING_TOOLHEAD_SERVO_ANGLES { 0, 180 }  // (degrees) Angles for
Lock, Unlock

  #elif ENABLED(MAGNETIC_SWITCHING_TOOLHEAD)

    #define SWITCHING_TOOLHEAD_Y_RELEASE      5         // (mm) Security
distance Y axis

    #define SWITCHING_TOOLHEAD_X_SECURITY   { 90, 150 } // (mm) Security
distance X axis (T0,T1)

    //#define PRIME_BEFORE_REMOVE                       // Prime the nozzle
before release from the dock

    #if ENABLED(PRIME_BEFORE_REMOVE)

      #define SWITCHING_TOOLHEAD_PRIME_MM          20  // (mm)   Extruder
prime length
```

```
      #define SWITCHING_TOOLHEAD_RETRACT_MM         10  // (mm)   Retract after
priming length

      #define SWITCHING_TOOLHEAD_PRIME_FEEDRATE    300  // (mm/m) Extruder
prime feedrate

      #define SWITCHING_TOOLHEAD_RETRACT_FEEDRATE 2400  // (mm/m) Extruder
retract feedrate

    #endif

  #elif ENABLED(ELECTROMAGNETIC_SWITCHING_TOOLHEAD)

    #define SWITCHING_TOOLHEAD_Z_HOP          2         // (mm) Z raise for
switching

  #endif

#endif
```

Adjust the relavant settings to your specifications for use with `SWITCHING_TOOLHEAD`, `PARKING_EXTRUDER` or `MAGNETIC_PARKING_EXTRUDER`.

Mixing Extruder

```
//#define MIXING_EXTRUDER

#if ENABLED(MIXING_EXTRUDER)

  #define MIXING_STEPPERS 2        // Number of steppers in your mixing
extruder

  #define MIXING_VIRTUAL_TOOLS 16  // Use the Virtual Tool method with [`M163`]
(/docs/gcode/M163.html) and [`M164`](/docs/gcode/M164.html)

  //#define DIRECT_MIXING_IN_G1    // Allow ABCDHI mix factors in [`G1`](/docs/
gcode/G000-G001.html) movement commands

#endif
```

A Mixing Extruder uses two or more stepper motors to drive multiple filaments into a mixing chamber, with the mixed filaments extruded from a single nozzle. This option adds the ability to set a mixture, to save mixtures, and to recall mixtures using the `T` command. The extruder still uses a single E axis, while the current mixture is used to determine the proportion of each filament to use. An “experimental” `G1` direct mixing option is included.

`MIXING_EXTRUDER` enables `M163` - set mix factor, `M164` - save mix, and `M165` - set mix.

Hotend Offsets

```
//#define HOTEND_OFFSET_X { 0.0, 20.00 } // (mm) relative X-offset for each nozzle

//#define HOTEND_OFFSET_Y { 0.0, 5.00 }  // (mm) relative Y-offset for each nozzle

//#define HOTEND_OFFSET_Z { 0.0, 0.00 }  // (mm) relative Z-offset for each nozzle
```

Hotend offsets are needed if your extruder has more than one nozzle. These values specify the offset from the first nozzle to each nozzle. So the first element is always set to 0.0. The next element corresponds to the next nozzle, and so on. Add more offsets if you have 3 or more nozzles.

Power Supply

```
//#define PSU_CONTROL

//#define PSU_NAME "Power Supply"



#if ENABLED(PSU_CONTROL)

  #define PSU_ACTIVE_HIGH false // Set 'false' for ATX (1), 'true' for X-Box (2)



  //#define PS_DEFAULT_OFF      // Keep power off until enabled directly with M80



  //#define AUTO_POWER_CONTROL  // Enable automatic control of the PS_ON pin

  #if ENABLED(AUTO_POWER_CONTROL)

    #define AUTO_POWER_FANS           // Turn on PSU if fans need power

    #define AUTO_POWER_E_FANS

    #define AUTO_POWER_CONTROLLERFAN

    #define AUTO_POWER_CHAMBER_FAN

    //#define AUTO_POWER_E_TEMP        50 // (°C) Turn on PSU over this temperature
```

```
    //#define AUTO_POWER_CHAMBER_TEMP  30 // (°C) Turn on PSU over this temperature

    #define POWER_TIMEOUT 30

  #endif

#endif
```

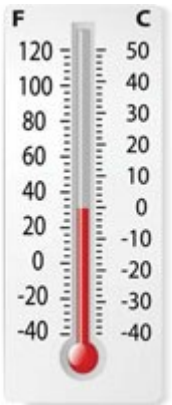
This option allows the controller board to switch the power supply 12v on and off with `M80` and `M81`. Requires `PS_ON_PIN`.

```
//#define PS_DEFAULT_OFF
```

Enable this if you don't want the power supply to switch on when you turn on the printer. This is for printers that have dual power supplies. For instance some setups have a separate power supply for the heaters. In this situation you can save power by leaving the power supply off until needed. If you don't know what this is leave it.

Thermal Settings

Temperature Sensors




```
#define TEMP_SENSOR_0 1

#define TEMP_SENSOR_1 0

#define TEMP_SENSOR_2 0

#define TEMP_SENSOR_3 0

#define TEMP_SENSOR_4 0

#define TEMP_SENSOR_5 0

#define TEMP_SENSOR_BED 0

#define TEMP_SENSOR_CHAMBER 0
```

Temperature sensors are vital components in a 3D printer. Fast and accurate sensors ensure that the temperature will be well controlled, to keep plastic flowing smoothly and to prevent mishaps. Use these settings to specify the hotend and bed temperature sensors. Every 3D printer will have a hotend thermistor, and most will have a bed thermistor.

The listing above these options in `Configuration.h` contains all the thermistors and thermocouples that Marlin knows and supports. Try to match your brand and model with one of the sensors in the list. If no match is found, use a profile for a similar sensor of the same brand, or try "1" – the generic profile. Each profile is calibrated for a particular temperature sensor so it's important to be as precise as possible.

It is crucial to obtain accurate temperature measurements. As a last resort, use 100k thermistor for `TEMP_SENSOR` and `TEMP_SENSOR_BED` but be highly skeptical of the temperature accuracy.

```
// Dummy thermistor constant temperature readings, for use with 998 and 999

#define DUMMY_THERMISTOR_998_VALUE 25

#define DUMMY_THERMISTOR_999_VALUE 100
```

Marlin provides two dummy sensors for testing purposes. Set their constant temperature readings here.

```
//#define TEMP_SENSOR_1_AS_REDUNDANT

#define MAX_REDUNDANT_TEMP_SENSOR_DIFF 10
```

Enable this option to use sensor 1 as a redundant sensor for sensor 0. This is an advanced way to protect against temp sensor failure. If the temperature difference between sensors exceeds `MAX_REDUNDANT_TEMP_SENSOR_DIFF` Marlin will abort the print and disable the heater.

Temperature Stability

```
#define TEMP_RESIDENCY_TIME 10  // (seconds)

#define TEMP_HYSTERESIS 3       // (degC) range of +/- temperatures considered
"close" to the target one

#define TEMP_WINDOW 1           // (degC) Window around target to start the
residency timer x degC early.
```

Extruders must maintain a stable temperature for `TEMP_RESIDENCY_TIME` before `M109` will return success and start the print. Tune what “stable” means using `TEMP_HYSTERESIS` and `TEMP_WINDOW`.

```
#define TEMP_BED_RESIDENCY_TIME 10  // (seconds)

#define TEMP_BED_HYSTERESIS 3       // (degC) range of +/- temperatures
considered "close" to the target one

#define TEMP_BED_WINDOW 1           // (degC) Window around target to start the
residency timer x degC early.
```

The bed must maintain a stable temperature for `TEMP_BED_RESIDENCY_TIME` before `M109` will return success and start the print. Tune what “stable” means using `TEMP_BED_HYSTERESIS` and `TEMP_BED_WINDOW`.

```
#define TEMP_CHAMBER_HYSTERESIS  3  // (°C) Temperature proximity considered
"close enough" to the target
```

Set how far from target the chamber can be and still be considered ok.

Temperature Ranges

```
#define HEATER_0_MINTEMP 5

#define HEATER_1_MINTEMP 5

#define HEATER_2_MINTEMP 5

#define HEATER_3_MINTEMP 5

#define HEATER_4_MINTEMP 5

#define BED_MINTEMP 5
```

These parameters help prevent the printer from overheating and catching fire. Temperature sensors report abnormally low values when they fail or become disconnected. Set these to the lowest value (in degrees C) that the machine is likely to experience. Indoor temperatures range from 10C-40C, but a value of 0 might be appropriate for an unheated workshop.

If any sensor goes below the minimum temperature set here, Marlin will **shut down the printer** with a “MINTEMP” error.

`Err: MINTEMP`: This error means your thermistor has disconnected or become an open circuit. (Or the machine is just very cold.)

```
#define HEATER_0_MAXTEMP 285

#define HEATER_1_MAXTEMP 275

#define HEATER_2_MAXTEMP 275

#define HEATER_3_MAXTEMP 275

#define HEATER_4_MAXTEMP 275

#define BED_MAXTEMP 130
```

Maximum temperature for each temperature sensor. If Marlin reads a temperature above these values, it will immediately shut down for safety reasons. For the E3D V6 hotend, many use 285 as a maximum value.

`Err: MAXTEMP`: This error usually means that the temperature sensor wires are shorted together. It may also indicate an issue with the heater MOSFET or relay that is causing it to stay on. Remember that cold surfaces near hot surfaces can lead to **condensation**, which is NOT GOOD for electronics. Use blower fans to keep air moving and use a to check your local dew point.

PID

Marlin uses PID (Proportional, Integral, Derivative) control (Wikipedia) to stabilize the dynamic heating system for the hotends and bed. When PID values are set correctly, heaters reach their target temperatures faster, maintain temperature better, and experience less wear over time.

Most vitally, correct PID settings will prevent excessive overshoot, which is a safety hazard. During PID calibration, use the highest target temperature you intend to use (where overshoots are more critical).

See the PID Tuning topic on the RepRap wiki for detailed instructions on `M303` auto-tuning. The PID settings should be tuned whenever changing a hotend, temperature sensor, heating element, board, power supply voltage (12v/24v), or anything else related to the high-voltage circuitry.

Hotend PID Options

```
#define PIDTEMP

#define BANG_MAX 255     // limits current to nozzle while in bang-bang mode; 255=full current

#define PID_MAX BANG_MAX // limits current to nozzle while PID is active (see PID_FUNCTIONAL_RANGE below); 255=full current

#define K1 0.95
```

Disable `PIDTEMP` to run extruders in bang-bang mode. Bang-bang is a pure binary mode - the heater is either fully-on or fully-off for a long period. PID control uses higher frequency PWM and (in most cases) is superior for maintaining a stable temperature.

```
#if ENABLED(PIDTEMP)

  //#define PID_EDIT_MENU

  //#define PID_AUTOTUNE_MENU

  //#define PID_DEBUG

  //#define PID_OPENLOOP 1

  //#define SLOW_PWM_HEATERS

  //#define PID_PARAMS_PER_HOTEND

  #define PID_FUNCTIONAL_RANGE 10
```

Enable `PID_AUTOTUNE_MENU` to add an option on the LCD to run an Autotune cycle and automatically apply the result. Enable `PID_PARAMS_PER_HOTEND` if you have more than one extruder and they are different models.

PID Values

```
// Ultimaker
#define  DEFAULT_Kp 22.2
#define  DEFAULT_Ki 1.08
#define  DEFAULT_Kd 114

// MakerGear
//#define  DEFAULT_Kp 7.0
//#define  DEFAULT_Ki 0.1
//#define  DEFAULT_Kd 12

// Mendel Parts V9 on 12V
//#define  DEFAULT_Kp 63.0
//#define  DEFAULT_Ki 2.25
//#define  DEFAULT_Kd 440
```

Sample PID values are included for reference, but they won't apply to most setups. The PID values you get from `M303` may be very different, but will be better for your specific machine.

`M301` can be used to set Hotend PID and is also accessible through the LCD. `M304` can be used to set bed PID. `M303` should be used to tune PID values before using any new hotend components.

Bed PID Options

```
//#define PIDTEMPBED
```

Enable `PIDTEMPBED` to use PID for the bed heater (at the same PWM frequency as the extruders). With the default PID_dT the PWM frequency is 7.689Hz, fine for driving a square wave into a resistive load without significant impact on FET heating. This also works fine on a Fotek SSR-10DA Solid State Relay into a 250W heater. If your configuration is significantly

different than this and you don't understand the issues involved, you probably shouldn't use bed PID until it's verified that your hardware works. Use `M303 E-1` to tune the bed PID for this option.

```
//#define BED_LIMIT_SWITCHING
```

Enable `BED_LIMIT_SWITCHING`

```
#define MAX_BED_POWER 255
```

The max power delivered to the bed. All forms of bed control obey this (PID, bang-bang, bang-bang with hysteresis). Setting this to anything other than 255 enables a form of PWM. As with `PIDTEMPBED`, don't enable this unless your bed hardware is ok with PWM.

Bed PID Values

```
#if ENABLED(PIDTEMPBED)



  //#define PID_BED_DEBUG // Sends debug data to the serial port.



  //120V 250W silicone heater into 4mm borosilicate (MendelMax 1.5+)

  //from FOPDT model - kp=.39 Tp=405 Tdead=66, Tc set to 79.2, aggressive factor of .15 (vs .1, 1, 10)

  #define  DEFAULT_bedKp 10.00

  #define  DEFAULT_bedKi .023

  #define  DEFAULT_bedKd 305.4



  //120V 250W silicone heater into 4mm borosilicate (MendelMax 1.5+)

  //from pidautotune

  //#define  DEFAULT_bedKp 97.1

  //#define  DEFAULT_bedKi 1.41

  //#define  DEFAULT_bedKd 1675.16



  // FIND YOUR OWN: "M303 E-1 C8 S90" to run autotune on the bed at 90 degreesC for 8 cycles.
```

```
#endif // PIDTEMPBED
```

Sample Bed PID values are included for reference, but use the result from `M303 E-1` for your specific machine.

Safety

Prevent Cold Extrusion

```
#define PREVENT_COLD_EXTRUSION

#define EXTRUDE_MINTEMP 170
```

So-called “cold extrusion” can damage a machine in several ways, but it usually just results in gouged filament and a jammed extruder. With this option, the extruder motor won’t move if the hotend is below the specified temperature. Override this setting with `M302` if needed.

Prevent Lengthy Extrude

```
#define PREVENT_LENGTHY_EXTRUDE

#define EXTRUDE_MAXLENGTH 200
```

A lengthy extrusion may not damage your machine, but it can be an awful waste of filament. This feature is meant to prevent a typo or glitch in a `G1` command from extruding some enormous amount of filament. For Bowden setups, the max length should be set greater than or equal to the load/eject length.

Thermal Protection

```
#define THERMAL_PROTECTION_HOTENDS // Enable thermal protection for all
extruders

#define THERMAL_PROTECTION_BED     // Enable thermal protection for the heated
bed

#define THERMAL_PROTECTION_CHAMBER // Enable thermal protection for the heated
chamber
```

Thermal protection is one of the most vital safety features in Marlin, allowing the firmware to catch a bad situation and shut down heaters before it goes too far. Consider what happens when a thermistor comes loose during printing. The firmware sees a low temperature reading so it keeps the heat on. As long as the temperature reading is low, the hotend will continue to heat up indefinitely, leading to smoke, oozing, a ruined print, and possibly even fire.

Marlin offers two levels of thermal protection:

1. Check that the temperature is actually increasing when a heater is on. If the temperature fails to rise enough within a certain time period (by default, 2 degrees in 20 seconds), the machine

will shut down with a “`Heating failed`” error. This will detect a disconnected, loose, or misconfigured thermistor, or a disconnected heater.

2. Monitor thermal stability. If the measured temperature drifts too far from the target temperature for too long, the machine will shut down with a “`Thermal runaway`” error. This error may indicate poor contact between thermistor and hot end, poor PID tuning, or a cold environment.

More thermal protection options are located in `Configuration_adv.h`. In most setups these can be left unchanged, but should be tuned as needed to prevent false positives.

Information

For false thermal runaways not caused by a loose temperature sensor, try increasing `WATCH_TEMP_PERIOD` or decreasing `WATCH_TEMP_INCREASE`. Heating may be slowed in a cold environment, if a fan is blowing on the heat block, or if the heater has high resistance.

Kinematics

Marlin supports four kinematic motion systems: Cartesian, Core (H-Bot), Delta, and SCARA. Cartesian is the simplest, applying each stepper directly to an axis. CoreXY uses a special belt arrangement to do XY motion, requiring a little extra maths. Delta robots convert the motion of three vertical carriages into XYZ motion in an “effector” attached to the carriages by six arms. SCARA robots move an arm in the XY plane using two angular joints.

CoreXY

```
//#define COREXY

//#define COREXZ

//#define COREYZ

//#define COREYX

//#define COREZX

//#define COREZY
```

Enable the option that applies to the specific Core setup. Both normal and reversed options are included for completeness.

Endstops

In open loop systems, endstops are an inexpensive way to establish the actual position of the carriage on all axes. In the procedure known as “homing,” each axis is moved towards one end until the endstop switch is triggered, at which point the

machine knows that the axis is at the endstop (home) position. From this point on, the machine “knows” its position by keeping track of how far the steppers have been moved. If the machine gets out of step for any reason, re-homing may be required.

Endstop Plugs

```
#define USE_XMIN_PLUG

#define USE_YMIN_PLUG

#define USE_ZMIN_PLUG

//#define USE_XMAX_PLUG

//#define USE_YMAX_PLUG

//#define USE_ZMAX_PLUG
```

Specify all the endstop connectors that are connected to any endstop or probe. Most printers will use all three min plugs. On delta machines, all the max plugs should be used. Probes can share the Z min plug, or can use one or more of the extra connectors. Don’t enable plugs used for non-endstop and non-probe purposes here.

`SENSORLESS_HOMING` will still need endstop connectors declared.

Endstop Pullups

```
#define ENDSTOPPULLUPS



#if DISABLED(ENDSTOPPULLUPS)

  // Disable ENDSTOPPULLUPS to set pullups individually

  //#define ENDSTOPPULLUP_XMAX

  //#define ENDSTOPPULLUP_YMAX

  //#define ENDSTOPPULLUP_ZMAX

  //#define ENDSTOPPULLUP_XMIN

  //#define ENDSTOPPULLUP_YMIN

  //#define ENDSTOPPULLUP_ZMIN

  //#define ENDSTOPPULLUP_ZMIN_PROBE

#endif
```

By default all endstops have pullup resistors enabled. This is best for NC switches, preventing the values from “floating.” If only some endstops should have pullup resistors, you can disable `ENDSTOPPULLUPS` and enable pullups individually.

Endstop Pulldowns

```
//#define ENDSTOPPULLDOWNS



#if DISABLED(ENDSTOPPULLDOWNS)

  // Disable ENDSTOPPULLDOWNS to set pulldowns individually

  //#define ENDSTOPPULLDOWN_XMAX

  //#define ENDSTOPPULLDOWN_YMAX

  //#define ENDSTOPPULLDOWN_ZMAX

  //#define ENDSTOPPULLDOWN_XMIN

  //#define ENDSTOPPULLDOWN_YMIN

  //#define ENDSTOPPULLDOWN_ZMIN

  //#define ENDSTOPPULLDOWN_ZMIN_PROBE

#endif
```

By default all endstops have pulldown resistors disabled.

Endstop Inverting

```
// Mechanical endstop with COM to ground and NC to Signal uses "false" here
(most common setup).

#define X_MIN_ENDSTOP_INVERTING false // set to true to invert the logic of the
endstop.

#define Y_MIN_ENDSTOP_INVERTING false // set to true to invert the logic of the
endstop.

#define Z_MIN_ENDSTOP_INVERTING false // set to true to invert the logic of the
endstop.

#define X_MAX_ENDSTOP_INVERTING false // set to true to invert the logic of the
endstop.
```

```
#define Y_MAX_ENDSTOP_INVERTING false // set to true to invert the logic of the endstop.

#define Z_MAX_ENDSTOP_INVERTING false // set to true to invert the logic of the endstop.

#define Z_MIN_PROBE_ENDSTOP_INVERTING false // set to true to invert the logic of the endstop.
```

Use `M119` to test if these are set correctly. If an endstop shows up as “TRIGGERED” when not pressed, and “open” when pressed, then it should be inverted here.

Stepper Drivers

```
//#define X_DRIVER_TYPE  A4988

//#define Y_DRIVER_TYPE  A4988

//#define Z_DRIVER_TYPE  A4988

//#define X2_DRIVER_TYPE A4988

//#define Y2_DRIVER_TYPE A4988

//#define Z2_DRIVER_TYPE A4988

//#define Z3_DRIVER_TYPE A4988

//#define E0_DRIVER_TYPE A4988

//#define E1_DRIVER_TYPE A4988

//#define E2_DRIVER_TYPE A4988

//#define E3_DRIVER_TYPE A4988

//#define E4_DRIVER_TYPE A4988

//#define E5_DRIVER_TYPE A4988
```

These settings allow Marlin to tune stepper driver timing and enable advanced options for stepper drivers that support them. You may also override timing options in Configuration_adv.h.

Endstop Interrupts

```
//#define ENDSTOP_INTERRUPTS_FEATURE
```

Enable this feature if all enabled endstop pins are interrupt-capable. This will remove the need to poll the interrupt pins, saving many CPU cycles.

Endstop Noise Threshold

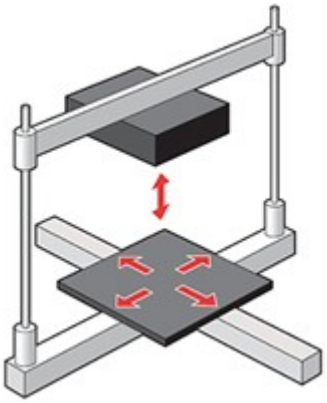
```
//#define ENDSTOP_NOISE_FEATURE
```

Enable if your probe or endstops falsely trigger due to noise.

Movement

Distinct E Factors

```
//#define DISTINCT_E_FACTORS
```



Enable `DISTINCT_E_FACTORS` if your extruders are not all mechanically identical. With this setting you can optionally specify different steps-per-mm, max feedrate, and max acceleration for each extruder.

Default Steps per mm

```
#define DEFAULT_AXIS_STEPS_PER_UNIT   { 80, 80, 4000, 500 }
```

These are the most crucial settings for your printer, as they determine how accurately the steppers will position the axes. Here we're telling the firmware how many individual steps produce a single millimeter (or degree on SCARA) of movement. These depend on various factors, including belt pitch, number of teeth on the pulley, thread pitch on leadscrews, micro-stepping settings, and extruder style.

Override with `M92`.

Step Calculator

The Prusa Calculator is a great tool to help find the right values for your specific printer configuration.

Default Max Feed Rate

```
#define DEFAULT_MAX_FEEDRATE { 500, 500, 2.25, 45 }
```

In any move, the velocities (in mm/sec) in the X, Y, Z, and E directions will be limited to the corresponding `DEFAULT_MAX_FEEDRATE`.

Override with `M203`.

Setting these too high will cause the corresponding stepper motor to lose steps, especially on high speed movements.

Acceleration

Default Max Acceleration

```
#define DEFAULT_MAX_ACCELERATION      { 3000, 3000, 100, 10000 }
```

When the velocity of any axis changes, its acceleration (or deceleration) in mm/s/s is limited by the current max acceleration setting. Also see the jerk settings below, which specify the largest instant speed change that can occur between segments.

A value of 3000 means that an axis may accelerate from 0 to 3000mm/m (50mm/s) within a one second movement.

Jerk sets the floor for accelerated moves. If the change in top speed for a given axis between segments is less than the jerk value for the axis, an instantaneous change in speed may be allowed. Limits placed on other axes also apply. Basically, lower jerk values result in more accelerated moves, which may be near-instantaneous in some cases, depending on the final acceleration determined by the planner.

Override with `M201`.

Default Acceleration

```
#define DEFAULT_ACCELERATION          3000    // X, Y, Z and E acceleration for printing moves

#define DEFAULT_RETRACT_ACCELERATION  3000    // E acceleration for retracts

#define DEFAULT_TRAVEL_ACCELERATION   3000    // X, Y, Z acceleration for travel (non printing) moves
```

The planner uses the default accelerations set here (or by `M204`) as the starting values for movement acceleration, and then constrains them further, if needed. There are separate default acceleration values for printing moves, retraction moves, and travel moves.

- Printing moves include E plus at least one of the XYZ axes.
- Retraction moves include only the E axis.
- Travel moves include only the XYZ axes.

In print/travel moves, `DEFAULT_ACCELERATION` and `DEFAULT_TRAVEL_ACCELERATION` apply to the XYZ axes. In retraction moves, `DEFAULT_RETRACT_ACCELERATION` applies only to the E-axis. During movement planning, Marlin constrains the default accelerations to the maximum acceleration of all axes involved in the move.

Override with `M204`.

Don’t set these too high. Larger acceleration values can lead to excessive vibration, noisy steppers, or even skipped steps. Lower acceleration produces smoother motion, eliminates vibration, and helps reduce wear on mechanical parts.

Junction Deviation

```
//#define JUNCTION_DEVIATION

#if ENABLED(JUNCTION_DEVIATION)

  #define JUNCTION_DEVIATION_MM 0.02  // (mm) Distance from real junction edge
```

```
#endif
```

Use Junction Deviation instead of traditional Jerk Limiting. Jerk settings is overriden with Junction Deviation.

Jerk

```
#define DEFAULT_XJERK                20.0
#define DEFAULT_YJERK                20.0
#define DEFAULT_ZJERK                 0.4
#define DEFAULT_EJERK                 5.0
```

Jerk works in conjunction with acceleration (see above). Jerk is the maximum change in velocity (in mm/sec) that can occur instantaneously. It can also be thought of as the minimum change in velocity that will be done as an accelerated (not instantaneous) move.

Both acceleration and jerk affect your print quality. If jerk is too low, the extruder will linger too long on small segments and corners, possibly leaving blobs. If the jerk is set too high, direction changes will apply too much torque and you may see “ringing” artifacts or dropped steps.

Override with `M205`.

S-Curve Acceleration

```
//#define S_CURVE_ACCELERATION
```

This option eliminates vibration during printing by fitting a Bézier curve to move acceleration, producing much smoother direction changes.

Z Probe Options

Probe Pins

```
#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN
```

Use this option in all cases when the probe is connected to the Z MIN endstop plug. This option is used for `DELTA` robots, which always home to MAX, and may be used in other setups.

You can use this option to configure a machine with no Z endstops. In that case the probe will be used to home Z and you will need to enable `Z_SAFE_HOMING` to ensure that the probe is positioned over the bed when homing the Z axis - done after X and Y.

```
//#define Z_MIN_PROBE_PIN 32
```

Use this option if you’ve connected the probe to a pin other than the Z MIN endstop pin. With this option enabled, by default Marlin will use the `Z_MIN_PROBE_PIN` specified in your board’s pins file

(usually the X or Z MAX endstop pin since these are the most likely to be unused). If you need to use a different pin, define your custom pin number for `Z_MIN_PROBE_PIN` in `Configuration.h`.

Probe Type

Marlin supports any kind of probe that can be made to work like a switch. Specific types of probes have different needs.

Manual Probe (no probe)

```
//#define PROBE_MANUALLY

//#define MANUAL_PROBE_START_Z 0.2
```

Even if you have no bed probe you can still use any of the core `AUTO_BED_LEVELING_*` options below by selecting this option. With `PROBE_MANUALLY` the `G29` command only moves the nozzle to the next probe point where it pauses. You adjust the Z height with a piece of paper or feeler gauge, then send `G29` again to continue to the next point. You can also enable `LCD_BED_LEVELING` to add a “Level Bed” Menu item to the LCD for a fully interactive leveling process. `MANUAL_PROBE_START_Z` sets the z-height the printer initially moves to at each mesh point during manual probing. With this disabled, the printer will move to Z0 for the first probe point. Then each consecutive probe point uses the Z position of the probe point preceding it.

Fix Mounted Probe

```
//#define FIX_MOUNTED_PROBE
```

This option is for any probe that’s fixed in place, with no need to be deployed or stowed. Specify this type for an inductive probe or when using the nozzle itself as the probe.

Servo Z Probe

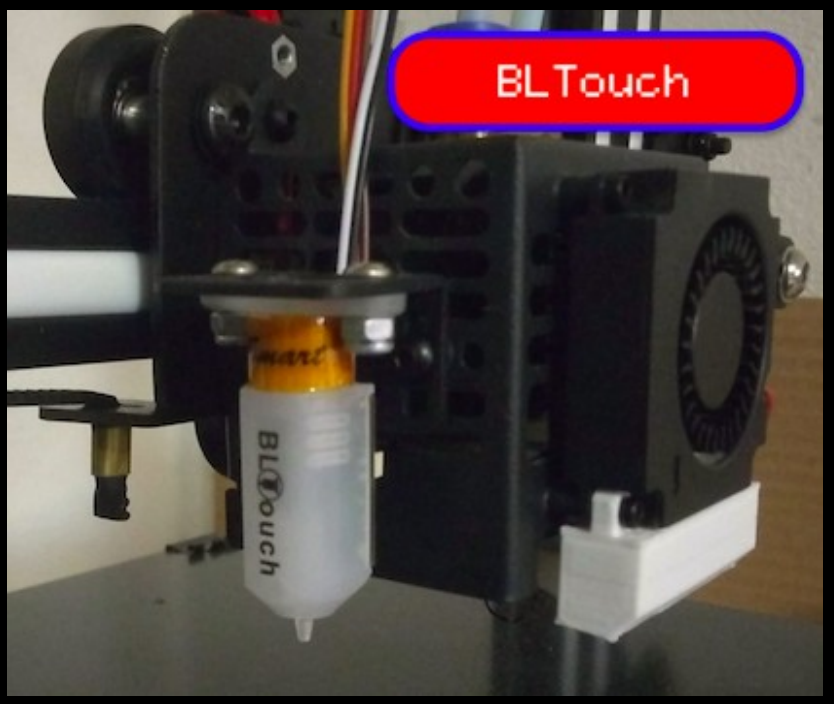
```
//#define Z_PROBE_SERVO_NR 0       // Defaults to SERVO 0 connector.

//#define Z_SERVO_ANGLES { 70, 0 } // Z Servo Deploy and Stow angles
```

To indicate a Servo Z Probe (e.g., an endstop switch mounted

on a rotating arm) just specify the servo index. Use the `M280` command to find the best `Z_SERVO_ANGLES` values.

BLTouch

```
//#define BLTOUCH
```



The ANTCLABS BLTouch probe uses custom circuitry and a magnet to raise and lower a metal pin which acts as a touch probe. The BLTouch uses the servo connector and is controlled using specific servo angles. With this option enabled the other required settings are automatically configured (so there's no need to enter servo angles, for example).

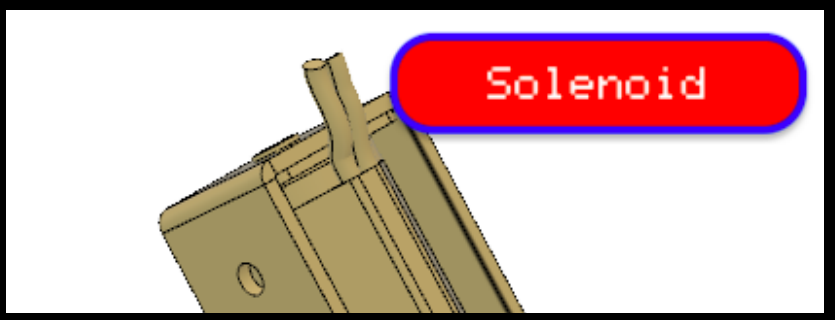
TOUCH MI PROBE

```
//#define TOUCH_MI_PROBE

#if ENABLED(TOUCH_MI_PROBE)

  #define TOUCH_MI_RETRACT_Z 0.5                  // Height at which the probe retracts

  //#define TOUCH_MI_DEPLOY_XPOS (X_MAX_BED + 2)  // For a magnet on the right side of the bed

  //#define TOUCH_MI_MANUAL_DEPLOY                // For manual deploy (LCD menu)

#endif
```

Touch-MI Probe by hotends.fr is deployed and activated by moving the X-axis to a magnet at the edge of the bed. By default, the magnet is assumed to be on the left and activated by a home. If the magnet is on the right, enable and set TOUCH_MI_DEPLOY_XPOS to the deploy position. Also option requires: `BABYSTEPPING`, `BABYSTEP_ZPROBE_OFFSET`, `Z_SAFE_HOMING`, and a minimum `Z_HOMING_HEIGHT` of 10.

Solenoid Probe

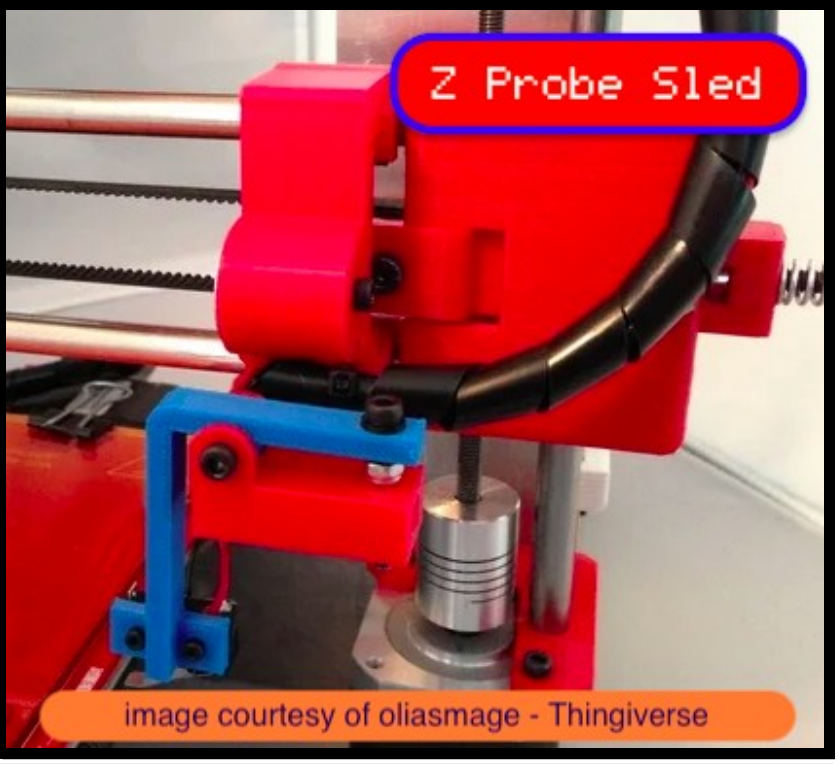
```
//#define SOLENOID_PROBE
```



A probe that is deployed and stowed with a solenoid pin (Defined as `SOL1_PIN`.)

Z Probe Sled

```
//#define Z_PROBE_SLED

//#define
SLED_DOCKING_OFFSET 5
```



This type of probe is mounted on a detachable “sled” that sits at the far end of the X axis. Before probing, the X carriage moves to the far end and picks up the sled. When probing is completed, it drops the sled off.
The `SLED_DOCKING_OFFSET` specifies the extra distance the X axis must travel to pickup the sled. 0 should be fine but it may be pushed further if needed.

Rack-and-pinion probe

```
//#define RACK_AND_PINION_PROBE

#if ENABLED(RACK_AND_PINION_PROBE)

  #define Z_PROBE_DEPLOY_X  X_MIN_POS

  #define Z_PROBE_RETRACT_X X_MAX_POS

#endif
```

A probe deployed by moving the x-axis, such as the Wilson II’s rack-and-pinion probe designed by Marty Rice.

Allen Key

```
//#define Z_PROBE_ALLEN_KEY
```

A retractable z-probe for deltas that uses an Allen key as the probe. See “Kossel automatic bed leveling probe” at the RepRap wiki. It deploys by leveraging against the z-axis belt, and retracts by pushing the probe down.

More information will be included in an upcoming Delta configuration page.

Probe Offsets

These offsets specify the distance from the tip of the nozzle to the probe — or more precisely, to the point at which the probe triggers. The X and Y offsets are specified as integers. The Z offset should be specified as exactly as possible using a decimal value. The Z offset can be overridden with `M851 Z` or the LCD controller. The `M851` offset is saved to EEPROM with `M500`.

Distance from edge

```
#define X_PROBE_OFFSET_FROM_EXTRUDER 10  // X offset: -left  +right  [of the nozzle]

#define Y_PROBE_OFFSET_FROM_EXTRUDER 10  // Y offset: -front +behind [the nozzle]

#define Z_PROBE_OFFSET_FROM_EXTRUDER 0   // Z offset: -below +above  [the nozzle]



#define MIN_PROBE_EDGE 10
```

Certain types of probe need to stay away from the edge

Probing Speed

```
// X and Y axis travel speed (mm/m) between probes

#define XY_PROBE_SPEED 8000



// Feedrate (mm/m) for the first approach when double-probing (MULTIPLE_PROBING == 2)

#define Z_PROBE_SPEED_FAST HOMING_FEEDRATE_Z



// Feedrate (mm/m) for the "accurate" probe of each point

#define Z_PROBE_SPEED_SLOW (Z_PROBE_SPEED_FAST / 2)
```

Probing should be done quickly, but the Z speed should be tuned for best repeatability. Depending on the probe, a slower Z probing speed may be needed for repeatable results.

Multiple Probes

```
//#define MULTIPLE_PROBING 2

//#define EXTRA_PROBING    1
```

Probing mutiple times yields better results. Set to 2 for a fast/slow probe - the second probe result will be used. Set to 3 or more for slow probes - the average result will be used.

Probe Clearance

```
#define Z_CLEARANCE_DEPLOY_PROBE   10 // Z Clearance for Deploy/Stow

#define Z_CLEARANCE_BETWEEN_PROBES  5 // Z Clearance between probe points

#define Z_CLEARANCE_MULTI_PROBE     5 // Z Clearance between multiple probes

//#define Z_AFTER_PROBING           5 // Z position after probing is done



#define Z_PROBE_LOW_POINT          -2 // Farthest distance below the trigger-
point to go before stopping
```

Z probes require clearance when deploying, stowing, and moving between probe points to avoid hitting the bed and other hardware. Servo-mounted probes require extra space for the arm to rotate. Inductive probes need space to keep from triggering early.

Use these settings to specify the distance (mm) to raise the probe (or lower the bed). The values set here apply over and above any (negative) probe Z Offset set with `Z_PROBE_OFFSET_FROM_EXTRUDER`, `M851`, or the LCD. Only integer values >= 1 are valid for these settings.

- Example: `M851 Z-5` with a CLEARANCE of 4 => 9mm from bed to nozzle.
- But: `M851 Z+1` with a CLEARANCE of 2 => 2mm from bed to nozzle.

G29 Movement

Make sure you have enough clearance for the probe to move between points!

```
#define Z_PROBE_OFFSET_RANGE_MIN -20

#define Z_PROBE_OFFSET_RANGE_MAX 20
```

For `M851` and LCD menus give a range for adjusting the Z probe offset.

Probe Testing

```
#define Z_MIN_PROBE_REPEATABILITY_TEST
```

This enables you to test the reliability of your probe. Issue a `M48` command to start testing. It will give you a standard deviation for the probe. Tip: 0.02 mm is normally acceptable for bed leveling to work.

```
// Before deploy/stow pause for user confirmation

//#define PAUSE_BEFORE_DEPLOY_STOW

#if ENABLED(PAUSE_BEFORE_DEPLOY_STOW)

  //#define PAUSE_PROBE_DEPLOY_WHEN_TRIGGERED // For Manual Deploy Allenkey Probe

#endif
```

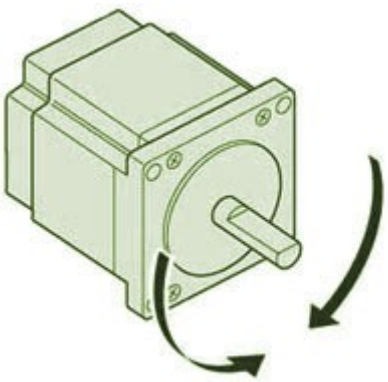
Before deploy/stow pause for user confirmation

Probe with heaters off

```
//#define PROBING_HEATERS_OFF       // Turn heaters off when probing

#if ENABLED(PROBING_HEATERS_OFF)

  //#define WAIT_FOR_BED_HEATER     // Wait for bed to heat back up between probes (to improve accuracy)

#endif

//#define PROBING_FANS_OFF          // Turn fans off when probing

//#define PROBING_STEPPERS_OFF      // Turn steppers off (unless needed to hold position) when probing

//#define DELAY_BEFORE_PROBING 200  // (ms) To prevent vibrations from triggering piezo sensors
```

Heatinging the bed and extruder for probing will produce results that more accurately correspond with your bed if you typically print with the bed heated. Enable `PROBING_HEATERS_OFF` if you are experiencing electrical noise. A delay can also be added to allow noise and vibration to settle.

Stepper Drivers



Motor Enable

```
#define X_ENABLE_ON 0

#define Y_ENABLE_ON 0

#define Z_ENABLE_ON 0
```

```
#define E_ENABLE_ON 0 // For all extruders
```

These options set the pin states used for stepper enable. The most common setting is 0 (`LOW`) for Active Low. For Active High use 1 or `HIGH`.

Motor Disable

```
#define DISABLE_X false

#define DISABLE_Y false

#define DISABLE_Z false
```

Use these options to disable steppers when not being issued a movement. This was implemented as a hack to run steppers at higher-than-normal current in an effort to produce more torque at the cost of increased heat for drivers and steppers.

Disabling the steppers between moves gives the motors and drivers a chance to cool off. It sounds good in theory, but in practice it has drawbacks. Disabled steppers can't hold the carriage stable. This results in poor accuracy and carries a strong probability of axial drift (i.e., lost steps).

Most 3D printers use an "open loop" control system, meaning the software can't ascertain the actual carriage position at a given time. It simply sends commands and assumes they have been obeyed. In practice with a well-calibrated machine this is not an issue and using open loop is a major cost saving with excellent quality.

We don't recommend this hack. There are much better ways to address the problem of stepper/driver overheating. Some examples: stepper/driver heatsink, active cooling, dual motors on the axis, reduce microstepping, check belt for over tension, check components for smooth motion, etc.

```
//#define DISABLE_REDUCED_ACCURACY_WARNING
```

Enable this option to suppress the warning given in cases when reduced accuracy is likely to occur.

```
#define DISABLE_E false                                        // For all
extruders

#define DISABLE_INACTIVE_EXTRUDER       false   // Keep only the active
extruder enabled
```

The E disable option works like `DISABLE_[XYZ]` but pertains to one or more extruders. The default setting keeps the active extruder enabled, disabling all inactive extruders. This is reasonable for situations where a "wipe tower" or other means is used to ensure that the nozzle is primed and not oozing between uses.

Motor Direction

```
#define INVERT_X_DIR false

#define INVERT_Y_DIR true

#define INVERT_Z_DIR false



#define INVERT_E0_DIR false

#define INVERT_E1_DIR false

#define INVERT_E2_DIR false

#define INVERT_E3_DIR false

#define INVERT_E4_DIR false
```

These settings reverse the motor direction for each axis. Be careful when first setting these. Axes moving the wrong direction can cause damage. Get these right without belts attached first, if possible. Before testing, move the carriage and bed to the middle. Test each axis for proper movemnt using the host or LCD “Move Axis” menu. If an axis is inverted, either flip the plug around or change its invert setting.

Homing and Bounds

Z Homing Height

```
//#define NO_MOTION_BEFORE_HOMING       // Inhibit movement
until all axes have been homed

//#define UNKNOWN_Z_NO_RAISE            // Don't raise Z (lower
the bed) if Z is "unknown."


                                                              //For
beds that fall when Z is powered off.

//#define Z_HOMING_HEIGHT 4
```

This value raises Z to the specified height above the bed before homing X or Y. This is useful to prevent the head crashing into bed mountings such as screws, bulldog clips, etc. This also works with auto bed leveling enabled and will be triggered only when the Z axis height is less than the defined value, otherwise the Z axis will not
move. `NO_MOTION_BEFORE_HOMING` and `UNKNOWN_Z_NO_RAISE`

Homing Direction

```
#define X_HOME_DIR -1

#define Y_HOME_DIR -1

#define Z_HOME_DIR -1
```

Homing direction for each axis: -1 = min, 1 = max. Most cartesian and core machines have three min endstops. Deltas have three max endstops. For other configurations set these values appropriately.

Movement Bounds

```
#define X_BED_SIZE 200

#define Y_BED_SIZE 200
```

With Marlin you can directly specify the bed size. This allows Marlin to do extra logic related to the bed size when it differs from the movement limits below. If the XY carriage is able to move outside of the bed, you can specify a wider range below.

```
#define X_MIN_POS 0

#define Y_MIN_POS 0

#define Z_MIN_POS 0

#define X_MAX_POS X_BED_SIZE

#define Y_MAX_POS Y_BED_SIZE

#define Z_MAX_POS 170
```

These values specify the physical limits of the machine. Usually the `[XYZ]_MIN_POS` values are set to 0, because endstops are positioned at the bed limits. `[XYZ]_MAX_POS` should be set to the farthest reachable point. By default, these are used as your homing positions as well. However, the `MANUAL_[XYZ]_HOME_POS` options can be used to override these, if needed.

Home Offset

Although home positions are fixed, `M206` can be used to apply offsets to the home position if needed.

Software Endstops

```
#define MIN_SOFTWARE_ENDSTOPS

#if ENABLED(MIN_SOFTWARE_ENDSTOPS)
```

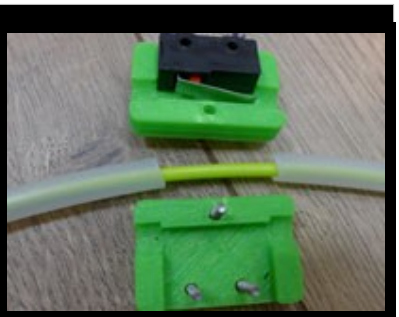
```
  #define MIN_SOFTWARE_ENDSTOP_X
  #define MIN_SOFTWARE_ENDSTOP_Y
  #define MIN_SOFTWARE_ENDSTOP_Z
#endif

#define MAX_SOFTWARE_ENDSTOPS
#if ENABLED(MAX_SOFTWARE_ENDSTOPS)
  #define MAX_SOFTWARE_ENDSTOP_X
  #define MAX_SOFTWARE_ENDSTOP_Y
  #define MAX_SOFTWARE_ENDSTOP_Z
#endif
```

Enable these options to constrain movement to the physical boundaries of the machine (as set by `[XYZ]_(MIN|MAX)_POS`). For example, `G1 Z-100` can be min constrained to `G1 Z0`. It is recommended to enable these options as a safety feature. If software endstops need to be disabled, use `M211 S0`.

```
#if EITHER(MIN_SOFTWARE_ENDSTOPS, MAX_SOFTWARE_ENDSTOPS)
  //#define SOFT_ENDSTOPS_MENU_ITEM
#endif
```

Enable/Disable software endstops from the LCD

Filament Runout Sensor



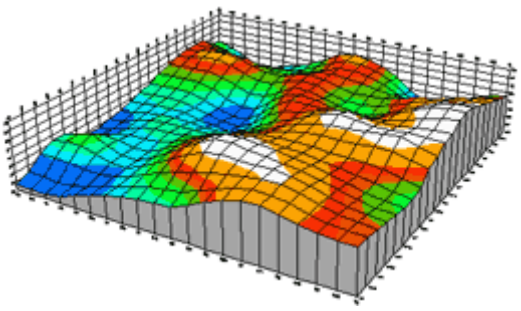
```
//#define FILAMENT_RUNOUT_SENSOR

#if ENABLED(FILAMENT_RUNOUT_SENSOR)

  #define NUM_RUNOUT_SENSORS   1     // Number of
sensors, up to one per extruder. Define a
FIL_RUNOUT#_PIN for each.

  #define FIL_RUNOUT_INVERTING false // Set to true to
invert the logic of the sensor.

  #define FIL_RUNOUT_PULLUP          // Use internal pullup for filament runout
pins.
```

```
  //#define FIL_RUNOUT_PULLDOWN      // Use internal pulldown for filament
runout pins.


  // Set one or more commands to execute on filament runout.
  // (After 'M412 H' Marlin will ask the host to handle the process.)
  #define FILAMENT_RUNOUT_SCRIPT "M600"


  // After a runout is detected, continue printing this length of filament
  // before executing the runout script. Useful for a sensor at the end of
  // a feed tube. Requires 4 bytes SRAM per sensor, plus 4 bytes overhead.
  //#define FILAMENT_RUNOUT_DISTANCE_MM 25


  #ifdef FILAMENT_RUNOUT_DISTANCE_MM
    // Enable this option to use an encoder disc that toggles the runout pin
    // as the filament moves. (Be sure to set FILAMENT_RUNOUT_DISTANCE_MM
    // large enough to avoid false positives.)
    //#define FILAMENT_MOTION_SENSOR
  #endif
#endif
```

With this feature, a mechanical or opto endstop switch is used to check for the presence of filament in the feeder (usually the switch is closed when filament is present). If the filament runs out, Marlin will run the specified GCode script (by default `M600`). RAMPS-based boards use `SERVO3_PIN`. For other boards you may need to define `FIL_RUNOUT_PIN`.

Bed Leveling

There are many cases where it is useful to measure variances in bed height. Even if the bed on a 3D printer is perfectly flat and level, there may still be imperfections in the mechanics. For example, a machine may have a very flat bed, but a corner of the XY gantry is a half-mm high. The ends of the Z axis may not be perfectly level. The



bed may move slightly in the Z plane as it moves in the X and/or Y plane. On a Delta there may be a lingering bowl-shape to its XY trajectory.

Bed Compensation or "— Bed Leveling" allows the machine —with a bed probe or user assistance— to take accurate measurements of the "bed height" at various points in the XY plane. With this data the machine can then adjust movement to align better to the tilt or "height" variances in the bed. (I'm scare-quoting "height" here because variances may come from other than the bed.)

For more details on these features, see `G29` for MBL and `G29` for ABL.

We recommend that you try and get your printer the best it can be before using bedlevel, after all bedlevel only compensates for "bad" hardware, it does not correct it.

Bed Leveling Style

Bed Leveling is a standard feature on many 3D printers. It takes the guess-work out of getting a good first layer and good bed adhesion. All forms of bed leveling add `G29` Bed Probing, `M420` enable/disable, and can save their results to EEPROM with `M500`. Bravo!

With Bed Leveling enabled:

- `G28` disables bed leveling, but leaves previous leveling data intact.
- `G29` automatically or manually probes the bed at various points, measures the bed height, calculates a correction grid or matrix, and turns on leveling compensation. Specific behavior depends on configuration and type of bed leveling.
- `M500` saves the bed leveling data to EEPROM. Use `M501` to load it, `M502` to clear it, and `M503` to report it.
- `M420 S<bool>` can be used to enable/disable bed leveling. For example, `M420 S1` must be used after `M501` to enable the loaded mesh or matrix, and to re-enable leveling after `G28`, which disables leveling compensation.
- A "Level Bed" menu item can be added to the LCD with the `LCD_BED_LEVELING` option.

```
//#define AUTO_BED_LEVELING_3POINT

//#define AUTO_BED_LEVELING_LINEAR

//#define AUTO_BED_LEVELING_BILINEAR

//#define AUTO_BED_LEVELING_UBL

//#define MESH_BED_LEVELING
```

Enable just one type of Bed Leveling.

- `AUTO_BED_LEVELING_3POINT` probes three points in a triangle. The flat plane gives a transform matrix suitable to compensate for a flat but tilted bed.
- `AUTO_BED_LEVELING_LINEAR` probes the bed in a grid. A transform matrix is produced by least-squares method to compensate for a flat but tilted bed.

- `AUTO_BED_LEVELING_BILINEAR` probes the bed in a grid, with optional Catmull-Rom subdivision. The mesh data is used to adjust Z height across the bed using bilinear interpolation. Good for delta, large, or uneven beds.
- `AUTO_BED_LEVELING_UBL` (recommended) combines the features of 3-point, linear, bilinear, and mesh leveling. As with bilinear leveling, the mesh data generated by UBL is used to adjust Z height across the bed using bilinear interpolation. An LCD controller is currently required.
- `MESH_BED_LEVELING` provides a custom `G29` command to measure the bed height at several grid points using a piece of paper or feeler gauge. See `G29` for MBL for the full procedure. This type of leveling is only compatible with `PROBE_MANUALLY`.

Only `AUTO_BED_LEVELING_BILINEAR` and `AUTO_BED_LEVELING_UBL` support `DELTA`.
Only `AUTO_BED_LEVELING_BILINEAR` currently supports `SCARA`.
`MESH_BED_LEVELING` is incompatible with Delta and SCARA.

Restore after G28

```
//#define RESTORE_LEVELING_AFTER_G28
```

Normally `G28` causes leveling to be disabled, so you have to re-enable it with `M420 S1` or `G29`. If you enable this option then `G28` will make sure to turn leveling back on if it was enabled beforehand.

Debug Leveling

```
//#define DEBUG_LEVELING_FEATURE
```

Use this option to enable extra debugging of homing and leveling. You can then use `M111 S32` before issuing `G28` and `G29 V4` to get a detailed log of the process for diagnosis. This option is useful to figure out the cause of unexpected behaviors, or when reporting issues to the project.

Leveling Fade Height

```
#define ENABLE_LEVELING_FADE_HEIGHT
```

Available with `MESH_BED_LEVELING`, `AUTO_BED_LEVELING_BILINEAR`, and `AUTO_BED_LEVELING_UBL`.

This option adds the `Z` parameter to `M420` which sets a fade distance over which leveling will be gradually reduced. Above the given Z height, leveling compensation will no longer be applied.

This feature exists to prevent irregularities in the bed from propagating through the model's entire height. Fading out leveling also reduces computational requirements and resonance from the Z axis above the fade height. For a well-aligned machine, this feature can improve print results.

Example: To have leveling fade out over the first 10mm of layer printing use `M420 Z10`. If each layer is 0.2mm high, leveling compensation will be reduced by 1/50th (2%) after each layer. Above 10mm the machine will move without compensation.

G26 Mesh Validation Pattern

```
/**

 * Enable the G26 Mesh Validation Pattern tool.

 */

#define G26_MESH_VALIDATION   // Enable G26 mesh validation

#if ENABLED(G26_MESH_VALIDATION)

  #define MESH_TEST_NOZZLE_SIZE     0.4   // (mm) Diameter of primary nozzle.

  #define MESH_TEST_LAYER_HEIGHT    0.2   // (mm) Default layer height for the
G26 Mesh Validation Tool.

  #define MESH_TEST_HOTEND_TEMP   205   // (°C) Default nozzle temperature for
the G26 Mesh Validation Tool.

  #define MESH_TEST_BED_TEMP       60   // (°C) Default bed temperature for the
G26 Mesh Validation Tool.

  #define G26_XY_FEEDRATE         20    // (mm/s) Feedrate for XY Moves for the
G26 Mesh Validation Tool.



#endif
```

When using any of the mesh-based leveling systems (1.1.7) you can activate `G26_MESH_VALIDATION` to print test patterns and fine-tune the mesh. See `G26` Mesh Validation for full details. The `G26` command accepts parameters for nozzle size, layer height, etc. The sub-options above specify the default values that will be applied for omitted parameters.

Linear / Bilinear Options

```
#define GRID_MAX_POINTS_X 3

#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X
```

These options specify the default number of points to probe in each dimension during `G29`.

```
  //#define LEFT_PROBE_BED_POSITION MIN_PROBE_EDGE

  //#define RIGHT_PROBE_BED_POSITION (X_BED_SIZE - (MIN_PROBE_EDGE))

  //#define FRONT_PROBE_BED_POSITION MIN_PROBE_EDGE

  //#define BACK_PROBE_BED_POSITION (Y_BED_SIZE - (MIN_PROBE_EDGE))
```

These settings specify the boundaries for probing with `G29`. This will most likely be a sub-section of the bed because probes are not usually able to reach every point that the nozzle can. Take account of the probe's XY offsets when setting these boundaries.

```
//#define PROBE_Y_FIRST
```

Enable this option if probing should proceed in the Y dimension first instead of X first.

Bilinear Options

```
//#define EXTRAPOLATE_BEYOND_GRID
```

Usually the probed grid doesn't extend all the way to the edges of the bed. So, outside the bounds of the probed grid, Z adjustment can take one of two approaches. Either the Z height can continue to raise/lower by the established tilt of the nearest grid box (best when most of the bed was probed), or it can follow the contour of the nearest edge (the default). Enable this option for extrapolation.

```
//#define ABL_BILINEAR_SUBDIVISION

#if ENABLED(ABL_BILINEAR_SUBDIVISION)

  // Number of subdivisions between probe points

  #define BILINEAR_SUBDIVISIONS 3

#endif
```

If you have SRAM to spare, this option will multiply the resolution of the bilinear grid using the Catmull-Rom subdivision method. This option only applies to bilinear leveling. If the default value of 3 is too expensive, try 2 or 1. (In Marlin 1.1.1, the default grid will be stored in PROGMEM, as UBL now does.)

Unified Bed Leveling Options

Probe Points

```
#define UBL_MESH_INSET 1          // Mesh inset margin on print area

#define GRID_MAX_POINTS_X 10      // Don't use more than 15 points per axis,
implementation limited.

#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X



  #define UBL_MESH_EDIT_MOVES_Z     // Sophisticated users prefer no movement
of nozzle
```

```
  #define UBL_SAVE_ACTIVE_ON_M500   // Save the currently active mesh in the current slot on M500



  //#define UBL_Z_RAISE_WHEN_OFF_MESH 2.5 // When the nozzle is off the mesh, this value is used

                                          // as the Z-Height correction value.
```

These options specify the inset, grid, and 3-point triangle to use for UBL. Note that probe XY offsets and movement limits may constrain the probeable area of the bed.

Mesh Bed Leveling Options

```
#define MESH_INSET 10          // Mesh inset margin on print area

#define GRID_MAX_POINTS_X 3    // Don't use more than 7 points per axis, implementation limited.

#define GRID_MAX_POINTS_Y GRID_MAX_POINTS_X



//#define MESH_G28_REST_ORIGIN // After homing all axes ('G28' or 'G28 XYZ') rest Z at Z_MIN_POS
```

These options specify the number of points that will always be probed in each dimension during `G29`. The mesh inset is used to automatically calculate the probe boundaries. These can be set explicitly in `Configuration_adv.h`. `MESH_G28_REST_ORIGIN` moves the nozzle to rest at `Z_MIN_POS` when mesh probing is done. If Z is offset (e.g., due to `home_offset` or some other cause) this is intended to move Z to a good starting point, usually Z=0.

3-Point Options

```
#if EITHER(AUTO_BED_LEVELING_3POINT, AUTO_BED_LEVELING_UBL)

  //#define PROBE_PT_1_X 15

  //#define PROBE_PT_1_Y 180

  //#define PROBE_PT_2_X 15

  //#define PROBE_PT_2_Y 20

  //#define PROBE_PT_3_X 170

  //#define PROBE_PT_3_Y 20
```

```
#endif
```

These options specify the three points that will be probed during `G29`.

LCD Bed Leveling

```
//#define LCD_BED_LEVELING

#if ENABLED(LCD_BED_LEVELING)

  #define MESH_EDIT_Z_STEP  0.025 // (mm) Step size while manually probing Z axis.

  #define LCD_PROBE_Z_RANGE 4     // (mm) Z Range centered on Z_MIN_POS for LCD Z adjustment

  //#define MESH_EDIT_MENU        // Add a menu to edit mesh points

#endif
```

`LCD_BED_LEVELING` adds a “Level Bed” menu to the LCD that starts a step-by-step guided leveling procedure that requires no probe. For Mesh Bed Leveling see `G29` for MBL, and for `PROBE_MANUALLY` see `G29` for ABL.

Available with `MESH_BED_LEVELING` and `PROBE_MANUALLY` (all forms of Auto Bed Leveling). See the `Configuration.h` file for sub-options.

Corner Leveling

```
//#define LEVEL_BED_CORNERS

#if ENABLED(LEVEL_BED_CORNERS)

  #define LEVEL_CORNERS_INSET 30    // (mm) An inset for corner leveling

  #define LEVEL_CORNERS_Z_HOP  4.0  // (mm) Move nozzle up before moving between corners

  #define LEVEL_CORNERS_HEIGHT 0.0  // (mm) Z height of nozzle at leveling points

  //#define LEVEL_CENTER_TOO        // Move to the center after the last corner

#endif
```

Add a menu item to move between bed corners for manual bed adjustment.

Z Probe End Script

```
//#define Z_PROBE_END_SCRIPT "G1 Z10 F12000\nG1 X15 Y330\nG1 Z0.5\nG1 Z10"
```

A custom script to do at the very end of `G29`. If multiple commands are needed, divide them with `\n` (the newline character).

Homing Options

Bed Center at 0,0

```
//#define BED_CENTER_AT_0_0
```

Enable this option if the bed center is at X0 Y0. This setting affects the way automatic home positions (those not set with `MANUAL_[XYZ]_POS`) are calculated. This should always be enabled with `DELTA`.

Manual Home Position

```
//#define MANUAL_X_HOME_POS 0

//#define MANUAL_Y_HOME_POS 0

//#define MANUAL_Z_HOME_POS 0 // Distance from nozzle to printbed after homing
```

These settings are used to override the home position. Leave them undefined for automatic settings. For `DELTA` Z home must be set to the top-most position.

Z Safe Homing

```
#define Z_SAFE_HOMING



#if ENABLED(Z_SAFE_HOMING)

  #define Z_SAFE_HOMING_X_POINT ((X_BED_SIZE) / 2)    // X point for Z homing when homing all axes (G28).

  #define Z_SAFE_HOMING_Y_POINT ((Y_BED_SIZE) / 2)    // Y point for Z homing when homing all axes (G28).

#endif
```

Z Safe Homing prevents Z from homing when the probe (or nozzle) is outside bed area by moving to a defined XY point (by default, the middle of the bed) before Z Homing when homing all axes with `G28`. As a side-effect, X and Y homing are required before Z homing. If stepper drivers time out, X and Y homing will be required again.

Enable this option if a probe (not an endstop) is being used for Z homing. Z Safe Homing isn't needed if a Z endstop is used for homing, but it may also be enabled just to have XY always move to some custom position after homing.

Homing Speed

```
// Homing speeds (mm/m)

#define HOMING_FEEDRATE_XY (50*60)

#define HOMING_FEEDRATE_Z  (4*60)
```

Homing speed for use in auto home and auto bed leveling. These values may be set to the fastest speeds your machine can achieve. Homing and probing speeds are constrained by the current max feedrate and max acceleration settings.

Setting these values too high may result in reduced accuracy and/or skipped steps. Reducing acceleration may help to achieve higher top speeds.

```
#define VALIDATE_HOMING_ENDSTOPS
```

Validate that endstops are triggered on homing moves.

Bed Skew Compensation

```
//#define SKEW_CORRECTION


#if ENABLED(SKEW_CORRECTION)

  // Input all length measurements here:

  #define XY_DIAG_AC 282.8427124746

  #define XY_DIAG_BD 282.8427124746

  #define XY_SIDE_AD 200


  // Or, set the default skew factors directly here

  // to override the above measurements:

  #define XY_SKEW_FACTOR 0.0


  //#define SKEW_CORRECTION_FOR_Z
```

```
  #if ENABLED(SKEW_CORRECTION_FOR_Z)
    #define XZ_DIAG_AC 282.8427124746
    #define XZ_DIAG_BD 282.8427124746
    #define YZ_DIAG_AC 282.8427124746
    #define YZ_DIAG_BD 282.8427124746
    #define YZ_SIDE_AD 200
    #define XZ_SKEW_FACTOR 0.0
    #define YZ_SKEW_FACTOR 0.0
  #endif


  // Enable this option for M852 to set skew at runtime
  //#define SKEW_CORRECTION_GCODE
#endif
```

Correct for misalignment in the XYZ axes. See `configuration.h` for a thorough explanation.

Additonal Features

EEPROM

```
//#define EEPROM_SETTINGS
```

Commands like `M92` only change the settings in volatile memory, and these settings are lost when the machine is powered off. With this option enabled, Marlin uses the built-in EEPROM to preserve settings across reboots. Settings saved to EEPROM (with `M500`) are loaded automatically whenever the machine restarts (and in most setups, when connecting to a host), overriding the defaults set in the configuration files. This option is highly recommended, as it makes configurations easier to manage.

The EEPROM-related commands are:

- `M500`: Save all current settings to EEPROM.
- `M501`: Load all settings last saved to EEPROM.
- `M502`: Reset all settings to their default values (as set by `Configuration.h`)
- `M503`: Print the current settings (in RAM, not EEPROM)

EEPROM Options

```
//#define DISABLE_M503    // Saves ~2700 bytes of PROGMEM. Disable for release!

#define EEPROM_CHITCHAT   // Give feedback on EEPROM commands. Disable to save PROGMEM.
```

These EEPROM options should be left as they are, but for 128K and smaller boards they may be used to recover some program memory. Vendors are strongly discouraged from using `DISABLE_M503`.

Settings that can be changed and saved to EEPROM are marked with . Options marked with can be changed from the LCD controller.
When you change saveable settings in the configuration files and re-flash, the new values don't take effect right away. They are still overridden by the saved values in EEPROM. To get your new default settings into the EEPROM, use `M502` followed by `M500`.

Host Keepalive

```
#define HOST_KEEPALIVE_FEATURE        // Disable this if your host doesn't like keepalive messages

#define DEFAULT_KEEPALIVE_INTERVAL 2  // Number of seconds between "busy" messages. Set with M113.

#define BUSY_WHILE_HEATING            // Some hosts require "busy" messages even during heating
```

When Host Keepalive is enabled Marlin will send a busy status message to the host every couple of seconds when it can't accept commands. Disable if your host doesn't like keepalive messages. Use `DEFAULT_KEEPALIVE_INTERVAL` for the default number of seconds between "busy" messages. Override with `M113`. Marlin 1.1.5 and up include the `BUSY_WHILE_HEATING` option for hosts that treat host keepalive as a strict busy protocol.

Free Memory Watcher

```
//#define M100_FREE_MEMORY_WATCHER
```

Uncomment to add the `M100` Free Memory Watcher for debugging purposes.

Inch Units

```
//#define INCH_MODE_SUPPORT
```

This option adds support for the `G20` and `G21` commands, allowing G-code to specify units in inches.

Temperature Units

```
//#define TEMPERATURE_UNITS_SUPPORT
```

This option adds support for `M149 C`, `M149 K`, and `M149 F` to set temperature units to Celsius, Kelvin, or Fahrenheit. Without this option all temperatures must be specified in Celsius units.

LCD Material Presets

```
#define PREHEAT_1_TEMP_HOTEND 180

#define PREHEAT_1_TEMP_BED     70

#define PREHEAT_1_FAN_SPEED     0 // Value from 0 to 255



#define PREHEAT_2_TEMP_HOTEND 240

#define PREHEAT_2_TEMP_BED    110

#define PREHEAT_2_FAN_SPEED     0 // Value from 0 to 255
```

These are the default values for the `Prepare` > `Preheat` LCD menu options. These values can be overridden using the `M145` command or the `Control` > `Temperature` > `Preheat Material X conf` submenus.

Nozzle Park

```
//#define NOZZLE_PARK_FEATURE

#if ENABLED(NOZZLE_PARK_FEATURE)

  // Specify a park position as { X, Y, Z_raise }

  #define NOZZLE_PARK_POINT { (X_MIN_POS + 10), (Y_MAX_POS - 10), 20 }

  #define NOZZLE_PARK_XY_FEEDRATE 100   // (mm/s) X and Y axes feedrate (also
used for delta Z axis)

  #define NOZZLE_PARK_Z_FEEDRATE 5      // (mm/s) Z axis feedrate (not used for
delta printers)

#endif
```

Park the nozzle at the given XYZ position on idle or `G27`.

The “P” parameter controls the action applied to the Z axis:

- `P0` - (Default) If Z is below park Z raise the nozzle.
- `P1` - Raise the nozzle always to Z-park height.
- `P2` - Raise the nozzle by Z-park amount, limited to `Z_MAX_POS`.

Nozzle Clean

```
//#define NOZZLE_CLEAN_FEATURE

#if ENABLED(NOZZLE_CLEAN_FEATURE)

  // Default number of pattern repetitions

  #define NOZZLE_CLEAN_STROKES  12



  // Default number of triangles

  #define NOZZLE_CLEAN_TRIANGLES  3



  // Specify positions as { X, Y, Z }

  #define NOZZLE_CLEAN_START_POINT { 30, 30, (Z_MIN_POS + 1)}

  #define NOZZLE_CLEAN_END_POINT   { 100, 60, (Z_MIN_POS + 1) }



  // Circular pattern radius

  #define NOZZLE_CLEAN_CIRCLE_RADIUS 6.5

  // Circular pattern circle fragments number

  #define NOZZLE_CLEAN_CIRCLE_FN 10

  // Middle point of circle

  #define NOZZLE_CLEAN_CIRCLE_MIDDLE NOZZLE_CLEAN_START_POINT



  // Move the nozzle to the initial position after cleaning

  #define NOZZLE_CLEAN_GOBACK
```

```
  // Enable for a purge/clean station that's always at the gantry height (thus no Z move)

  //#define NOZZLE_CLEAN_NO_Z

#endif
```

Adds the `G12` command to perform a nozzle cleaning process. See `Configuration.h` for additional configuration options.

Print Job Timer

```
#define PRINTJOB_TIMER_AUTOSTART
```

Automatically start and stop the print job timer when `M104`/`M109`/`M190` commands are received. Also adds the following commands to control the timer:

- `M75` - Start the print job timer.
- `M76` - Pause the print job timer.
- `M77` - Stop the print job timer.

Print Counter

```
//#define PRINTCOUNTER
```

When enabled Marlin will keep track of some print statistics such as:

- Total print jobs
- Total successful print jobs
- Total failed print jobs
- Total time printing

This information can be viewed by the `M78` command.

LCD Language

User Interface Language

```
#define LCD_LANGUAGE en
```

Choose your preferred language for the LCD controller here. Supported languages include:

Code	Language		Code	Language		Code	Language
en	English (Default)		an	Aragonese		bg	Bulgarian
ca	Catalan		cn	Chinese		cz	Czech
da	Danish		de	German		el	Greek

Code	Language	Code	Language	Code	Language
el-gr	Greek (Greece)	es	Spanish	eu	Basque-Euskera
fi	Finnish	fr	French	gl	Galician
hr	Croatian	it	Italian	kana	Japanese
kana_utf8	Japanese (UTF8)	ko_KR	Korean (South Korea)	nl	Dutch
pl	Polish	pt	Portuguese	pt-br	Portuguese (Brazilian)
pt-	Portuguese (Brazilian UTF8)	pt_utf8	Portuguese (UTF8)	ru	Russian
sk_utf8	Slovak (UTF8)	tr	Turkish	uk	Ukrainian
vi	Vietnamese	zh_CN	Chinese (Simplified)	zh_TW	Chinese (Traditional)

See `language.h` for the latest list of supported languages and their international language codes.

HD44780 Character Set

```
#define DISPLAY_CHARSET_HD44780 JAPANESE
```

This option applies only to character-based displays. Character-based displays (based on the Hitachi HD44780) provide an ASCII character set plus one of the following language extensions:

- `JAPANESE` … the most common
- `WESTERN` …. with more accented characters
- `CYRILLIC` … for the Russian language

To determine the language extension installed on your controller:

- Compile and upload with `LCD_LANGUAGE` set to 'test'
- Click the controller to view the LCD menu
- The LCD will display Japanese, Western, or Cyrillic text

See LCD Language System for in-depth info on how the Marlin display system currently works.

LCD Type

```
//#define ULTRA_LCD // Character based

//#define DOGLCD    // Full graphics display
```

The base LCD Type is either character-based or graphical. Marlin will automatically set the correct one for your specific display, specified below. Unless your display is unsupported by Marlin, you can leave these options disabled.

SD Card

```
//#define SDSUPPORT // Enable SD Card Support in Hardware Console
```

Enable to use SD printing, whether as part of an LCD controller or as a standalone SDCard slot.

The `SDSUPPORT` option must be enabled or SD printing will not be supported. It is no longer enabled automatically for LCD controllers with built-in SDCard slot.

SPI Speed

```
//#define SPI_SPEED SPI_HALF_SPEED

//#define SPI_SPEED SPI_QUARTER_SPEED

//#define SPI_SPEED SPI_EIGHTH_SPEED
```

Uncomment ONE of these options to use a slower SPI transfer speed. This is usually required if you're getting volume init errors.

Enable CRC

```
//#define SD_CHECK_AND_RETRY
```

Use CRC checks and retries on the SD communication.

Encoder

Encoder Resolution

```
//#define ENCODER_PULSES_PER_STEP 1
```

This option overrides the default number of encoder pulses needed to produce one step. Should be increased for high-resolution encoders.

```
//#define ENCODER_STEPS_PER_MENU_ITEM 5
```

Use this option to override the number of step signals required to move between next/prev menu items.

Encoder Direction

Test your encoder's behavior first with both of the following options disabled.

- Reversed Value Edit and Menu Nav? Enable `REVERSE_ENCODER_DIRECTION`.
- Reversed Menu Navigation only? Enable `REVERSE_MENU_DIRECTION`.
- Reversed Value Editing only? Enable BOTH options.

```
//#define REVERSE_ENCODER_DIRECTION
```

This option reverses the encoder direction everywhere. Set if CLOCKWISE causes values to DECREASE.

```
//#define REVERSE_MENU_DIRECTION
```

This option reverses the encoder direction for navigating LCD menus. If CLOCKWISE normally moves DOWN this makes it go UP. If CLOCKWISE normally moves UP this makes it go DOWN.

```
//#define INDIVIDUAL_AXIS_HOMING_MENU
```

Add individual axis homing items (Home X, Home Y, and Home Z) to the LCD menu.

Speaker

```
//#define SPEAKER
```

By default Marlin assumes you have a buzzer with a fixed frequency. If you have a speaker that can produce tones, enable it here.

```
//#define LCD_FEEDBACK_FREQUENCY_DURATION_MS 100

//#define LCD_FEEDBACK_FREQUENCY_HZ 1000
```

The duration and frequency for the UI feedback sound. Set these to 0 to disable audio feedback in the LCD menus. Test audio output with the G-code `M300 S<frequency Hz> P<duration ms>`

LCD Controller

Marlin includes support for several controllers. The two most popular controllers supported by Marlin are:

- `REPRAP_DISCOUNT_SMART_CONTROLLER` A 20 x 4 character-based LCD controller with click-wheel.
- `REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER` A monochrome 128 x 64 pixel-based LCD controller with click-wheel. Able to display simple bitmap graphics and up to 5 lines of text.

Most other LCD controllers are variants of these. Enable just one of the following options for your specific controller:

Character LCDs

Option	Description
`ULTIMAKERCONTROLLER`	The original Ultimaker Controller.

Option	Description
ULTIPANEL	ULTIPANEL as seen on Thingiverse.
PANEL_ONE	PanelOne from T3P3 (via RAMPS 1.4 AUX2/AUX3). A variant of ULTIMAKERCONTROLLER.
REPRAP_DISCOUNT_SMART_CONTROLLER	RepRapDiscount Smart Controller. Usually sold with a white PCB.
G3D_PANEL	Gadgets3D G3D LCD/SD Controller. Usually sold with a blue PCB.
RIGIDBOT_PANEL	RigidBot Panel V1.0.
ANET_KEYPAD_LCD	Anet Keypad LCD for the Anet A3

Graphical LCDs

Option	Description
CARTESIO_UI	Cartesio UI.
MAKRPANEL	MaKr3d Makr-Panel with graphic controller and SD support.
REPRAPWORLD_GRAPHICAL_LCD	ReprapWorld Graphical LCD.
VIKI2	Panucatt Devices Viki 2.0.
miniVIKI	mini Viki with Graphic LCD.
ELB_FULL_GRAPHIC_CONTROLLER	Adafruit ST7565 Full Graphic Controller.
REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER	RepRapDiscount Full Graphic Smart Controller.
MINIPANEL	MakerLab Mini Panel with graphic controller and SD support.
BQ_LCD_SMART_CONTROLLER	BQ LCD Smart Controller shipped with the BQ Hephestos 2 and Witbox 2.
ANET_FULL_GRAPHICS_LCD	Anet Full Graphics LCD for the Anet A3

Keypads

Option	Description
REPRAPWORLD_KEYPAD	RepRapWorld Keypad v1.1 Use REPRAPWORLD_KEYPAD_MOVE_STEP to set how much the robot should move on each keypress (e.g., 10mm per click).

I2C Character LCDs

These controllers all require the LiquidCrystal_I2C library.

Option	Description
RA_CONTROL_PANEL	Elefu RA Board Control Panel
LCD_I2C_SAINSMART_YWROBOT	Sainsmart YWRobot LCM1602 LCD Display.
LCM1602	Generic LCM1602 LCD adapter
LCD_I2C_PANELOLU2	PANELOLU2 LCD with status LEDs, separate encoder and click inputs. The click input can either be directly connected to a pin (if BTN_ENC is defined) or read through I2C (with BTN_ENC undefined). Requires LiquidTWI2 library v1.2.3 or later.
LCD_I2C_VIKI	Panucatt VIKI LCD with status LEDs, integrated click & L/R/U/D buttons, separate encoder inputs.

Option	Description
SAV_3DLCD	Shift register panels. 2 wire Non-latching LCD SR. See LCD configuration.

I2C Graphical LCDs

These controllers all require the LiquidCrystal_I2C library.

Option	Description
U8GLIB_SSD1306	SSD1306 OLED full graphics generic display.
SAV_3DGLCD	SAV OLED LCD module support using either SSD1306 or SH1106 based LCD modules.
OLED_PANEL_TINYBOY2	TinyBoy2 128x64 OLED / Encoder Panel

Fan PWM

```
//#define FAST_PWM_FAN
```

Increase the FAN PWM frequency. Removes the PWM noise but increases heating in the FET/Arduino.

```
//#define FAN_SOFT_PWM
```

Use software PWM to drive the fan, as with the heaters. This uses a very low frequency which is not as annoying as with the hardware PWM. On the other hand, if this frequency is too low, you should also increment SOFT_PWM_SCALE.

```
#define SOFT_PWM_SCALE 0
```

Incrementing this by 1 will double the software PWM frequency, affecting heaters (and the fan if FAN_SOFT_PWM is enabled). However, control resolution will be halved for each increment; at zero value, there are 128 effective control positions.

```
//#define SOFT_PWM_DITHER
```

If SOFT_PWM_SCALE is set to a value higher than 0, dithering can be used to mitigate the associated resolution loss. If enabled, some of the PWM cycles are stretched so on average the desired duty cycle is attained.

Temperature Status LEDs

```
//#define TEMP_STAT_LEDS
```

Temperature status LEDs that display the hotend and bed temperature. If all hotend and bed temperature setpoint are < 54C then the BLUE led is on. Otherwise the RED led is on. There is 1C hysteresis.

Photo Pin

```
//#define PHOTOGRAPH_PIN     23
```

`M240` triggers a camera by emulating a Canon RC-1 Remote Data as described on this site.

SkeinForge Arc Fix

```
//#define SF_ARC_FIX
```

Files sliced with SkeinForge contain the wrong arc GCodes when using “Arc Point” as fillet procedure. This option works around that bug, but otherwise should be left off.

Extra Features

Fast PWM Fan

```
//#define FAST_PWM_FAN
```

`FAST_PWM_FAN` increases the FAN PWM frequency. The frequency and scaling can be adjusted in the `configuration_adv.h` file.

Fan Software PWM

```
//#define FAN_SOFT_PWM

#define SOFT_PWM_SCALE 0

//#define SOFT_PWM_DITHER
```

Use software PWM to drive the fan. This uses a very low frequency which is not as annoying as with the hardware PWM. Increase `SOFT_PWM_SCALE` if the frequency is too low. If experiencing resolution loss when `SOFT_PWM_SCALE` is set to a value greater than 0, `SOFT_PWM_DITHER` can be used to mitigate it. If enabled.

Temperature Status LEDs

```
//#define TEMP_STAT_LEDS
```

Adds a simple temperature status indicators using LEDs.

SkeinForge ARC G-code correction

```
//#define SF_ARC_FIX
```

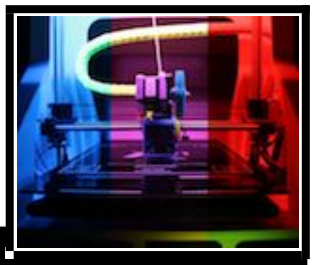
Correct the wrong arc g-codes sent by SkeinForge when using Arc Point as fillet procedure

Paste Extruder

```
// Support for the BariCUDA Paste Extruder.

//#define BARICUDA
```

Marlin includes support for the Baricuda Extruder for 3D Printing Sugar and Chocolate also hosted on GitHub. The feature adds the codes `M126`, `M127`, `M128`, and `M129` for controlling the pump and valve of the Baricuda.

RGB Color LEDs



Marlin currently supplies two options for RGB-addressable color indicators. In both cases the color is set using `M150 Rr Ug Bb` to specify RGB components from 0 to 255.

```
//define BlinkM/CyzRgb Support

//#define BLINKM
```

The BLINKM board supplies the backlighting for some LCD controllers. Its color is set using I2C messages.

```
//define PCA9632 PWM LED driver Support

//#define PCA9632
```

The Philips PCA9632 is a common PWM LED driver, controlled (like BlinkM) using I2C.

```
//#define RGB_LED

//#define RGBW_LED

#if ENABLED(RGB_LED) || ENABLED(RGBW_LED)

  #define RGB_LED_R_PIN 34

  #define RGB_LED_G_PIN 43

  #define RGB_LED_B_PIN 35

  #define RGB_LED_W_PIN -1

#endif
```

Enable support for an RGB(W) LED connected to 5V digital pins, or an RGB(W) Strip connected to MOSFETs controlled by digital pins. An inexpensive RGB LED can be used simply by assigning digital pins for each component. If the pins are able to do hardware PWM then a wide range of colors will be available. With simple digital pins only 7 colors are possible.

Adds the `M150` command to set the LED (or LED strip) color. If pins are PWM capable (e.g., 4, 5, 6, 11) then a range of luminance values can be set from 0 to 255.

LED Strips require a MOFSET Chip between PWM lines and LEDs, as the Arduino cannot handle the current the LEDs will require. Failure to follow this precaution can destroy your Arduino!

Adafruit Neopixel LED Driver

```
//#define NEOPIXEL_LED

#if ENABLED(NEOPIXEL_LED)

  #define NEOPIXEL_TYPE   NEO_GRBW // NEO_GRBW / NEO_GRB - four/three channel driver type (defined in Adafruit_NeoPixel.h)

  #define NEOPIXEL_PIN     4       // LED driving pin

  //#define NEOPIXEL2_TYPE NEOPIXEL_TYPE

  //#define NEOPIXEL2_PIN    5

  #define NEOPIXEL_PIXELS 30       // Number of LEDs in the strip, larger of 2 strips if 2 neopixel strips are used

  #define NEOPIXEL_IS_SEQUENTIAL   // Sequential display for temperature change - LED by LED. Disable to change all LEDs at once.

  #define NEOPIXEL_BRIGHTNESS 127  // Initial brightness (0-255)

  //#define NEOPIXEL_STARTUP_TEST  // Cycle through colors at startup

  //#define NEOPIXEL_BKGD_LED_INDEX  0              // Index of the LED to use

  //#define NEOPIXEL_BKGD_COLOR { 255, 255, 255, 0 } // R, G, B, W

#endif
```

NEOPIXELS

Printer Event LEDs

```
#if ENABLED(BLINKM) || ENABLED(RGB_LED) || ENABLED(RGBW_LED) || ENABLED(PCA9632)

  #define PRINTER_EVENT_LEDS

#endif
```

This option causes the printer to give status feedback on the installed color LED, BLINKM, or PCA9632:

- Gradually change from blue to violet as the heated bed gets to target temp.

- Gradually change from violet to red as the hotend gets to temperature.
- Change to white to illuminate work surface.
- Change to green once print has finished.
- Turn off after the print has finished and the user has pushed a button.

Servos

Number of Servos


```
//#define NUM_SERVOS 3 // Servo index starts with 0 for M280 command
```

The total number of servos to enable for use. One common application for a servo is a Z bed probe consisting of an endstop switch mounted on a rotating arm. To use one of the servo connectors for this type of probe, set `Z_ENDSTOP_SERVO_NR` in the probe options above.

Servo Deactivation

```
#define SERVO_DELAY 300
```

Delay (in microseconds) before the next move will start, to give the servo time to reach its target angle. 300ms is a good value but you can try less delay. Specify a large enough delay so the servo has enough time to complete a full motion before deactivation.

```
//#define DEACTIVATE_SERVOS_AFTER_MOVE
```

With this option servos are powered only during movement, then turned off to prevent jitter. We recommend enabling this option to keep electrical noise from active servos from interfering with other components. The high amperage generated by extruder motor wiring during movement can also induce movement in active servos. Leave this option enabled to avoid all such servo-related troubles.

Configuration_adv.h

Temperature Options

Custom Thermistor 1000 Parameters

```
#if TEMP_SENSOR_0 == 1000

  #define HOTEND0_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor

  #define HOTEND0_RESISTANCE_25C_OHMS  100000  // Resistance at 25C

  #define HOTEND0_BETA                 3950    // Beta value
```

```
#endif


#if TEMP_SENSOR_1 == 1000
  #define HOTEND1_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND1_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND1_BETA                 3950    // Beta value
#endif


#if TEMP_SENSOR_2 == 1000
  #define HOTEND2_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND2_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND2_BETA                 3950    // Beta value
#endif


#if TEMP_SENSOR_3 == 1000
  #define HOTEND3_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND3_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND3_BETA                 3950    // Beta value
#endif


#if TEMP_SENSOR_4 == 1000
  #define HOTEND4_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor
  #define HOTEND4_RESISTANCE_25C_OHMS  100000  // Resistance at 25C
  #define HOTEND4_BETA                 3950    // Beta value
#endif
```

```
#if TEMP_SENSOR_5 == 1000

  #define HOTEND5_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor

  #define HOTEND5_RESISTANCE_25C_OHMS  100000  // Resistance at 25C

  #define HOTEND5_BETA                 3950    // Beta value

#endif



#if TEMP_SENSOR_BED == 1000

  #define BED_PULLUP_RESISTOR_OHMS     4700    // Pullup resistor

  #define BED_RESISTANCE_25C_OHMS      100000  // Resistance at 25C

  #define BED_BETA                     3950    // Beta value

#endif



#if TEMP_SENSOR_CHAMBER == 1000

  #define CHAMBER_PULLUP_RESISTOR_OHMS 4700    // Pullup resistor

  #define CHAMBER_RESISTANCE_25C_OHMS  100000  // Resistance at 25C

  #define CHAMBER_BETA                 3950    // Beta value

#endif
```

Marlin 2.0 allows for custom temperature sensors.

```
//

// Hephestos 2 24V heated bed upgrade kit.

// https://store.bq.com/en/heated-bed-kit-hephestos2

//

//#define HEPHESTOS2_HEATED_BED_KIT

#if ENABLED(HEPHESTOS2_HEATED_BED_KIT)

  #undef TEMP_SENSOR_BED

  #define TEMP_SENSOR_BED 70
```

```
  #define HEATER_BED_INVERTING true

#endif
```

Enables the use of Hephestos 2 24V heated bed.

Heated Chamber

```
#if TEMP_SENSOR_CHAMBER

  #define CHAMBER_MINTEMP             5

  #define CHAMBER_MAXTEMP            60

  #define TEMP_CHAMBER_HYSTERESIS     1   // (°C) Temperature proximity
considered "close enough" to the target

  //#define CHAMBER_LIMIT_SWITCHING

  //#define HEATER_CHAMBER_PIN       44   // Chamber heater on/off pin

  //#define HEATER_CHAMBER_INVERTING false

#endif
```

A heated chamber can greatly improve print quality. Check the pins file of your board for `TEMP_CHAMBER_PIN`. The spare extruder and hotend temperature pins can be used for `HEATER_CHAMBER_PIN` and `TEMP_CHAMBER_PIN`.

Bang-Bang Bed Heating

```
#if DISABLED(PIDTEMPBED)

  #define BED_CHECK_INTERVAL 5000 // ms between checks in bang-bang control

  #if ENABLED(BED_LIMIT_SWITCHING)

    #define BED_HYSTERESIS 2 // Only disable heating if T>target+BED_HYSTERESIS
and enable heating if T>target-BED_HYSTERESIS

  #endif

#endif
```

These sub-options can be used when the bed isn't using PID heating. A "bang-bang" heating method will be used instead, simply checking against current temperature at regular intervals.

Thermal Protection Settings

Hotend Thermal Protection

```
#if ENABLED(THERMAL_PROTECTION_HOTENDS)

  #define THERMAL_PROTECTION_PERIOD 40        // Seconds

  #define THERMAL_PROTECTION_HYSTERESIS 4     // Degrees Celsius

  #define WATCH_TEMP_PERIOD 20                // Seconds

  #define WATCH_TEMP_INCREASE 2               // Degrees Celsius

#endif
```

Hot end thermal protection can be tuned with these sub-options.

The first two options deal with continuous thermal protection during an entire print job.

The second set of options applies to changes in target temperature. Whenever an `M104` or `M109` increases the target temperature the firmware will wait for the `WATCH_TEMP_PERIOD` to expire, and if the temperature hasn't increased by `WATCH_TEMP_INCREASE` degrees, the machine is halted, requiring a hard reset. This test restarts with any `M104`/`M109`, but only if the current temperature is far enough below the target for a reliable test.

If you get false positives for "Heating failed" increase `WATCH_TEMP_PERIOD` and/or decrease `WATCH_TEMP_INCREASE`. (`WATCH_TEMP_INCREASE` should not be set below 2.)

Bed Thermal Protection

```
#if ENABLED(THERMAL_PROTECTION_BED)

  #define THERMAL_PROTECTION_BED_PERIOD 20    // Seconds

  #define THERMAL_PROTECTION_BED_HYSTERESIS 2 // Degrees Celsius

  #define WATCH_BED_TEMP_PERIOD 60            // Seconds

  #define WATCH_BED_TEMP_INCREASE 2           // Degrees Celsius

#endif
```

Heated bed thermal protection can be tuned with these sub-options.

The first two options deal with continuous thermal protection during an entire print job.

The second set of options applies to changes in target temperature. Whenever an `M140` or `M190` increases the target temperature the firmware will wait for the `WATCH_BED_TEMP_PERIOD` to expire, and if the temperature hasn't increased by `WATCH_BED_TEMP_INCREASE` degrees, the machine is halted, requiring a hard reset. This test

restarts with any `M140`/`M190`, but only if the current temperature is far enough below the target for a reliable test.

If you get too many “Heating failed” errors, increase `WATCH_BED_TEMP_PERIOD` and/or decrease `WATCH_BED_TEMP_INCREASE`. (`WATCH_BED_TEMP_INCREASE` should not be set below 2.)

Heated Chamber Thermal Protection

```
#if ENABLED(THERMAL_PROTECTION_CHAMBER)

  #define THERMAL_PROTECTION_CHAMBER_PERIOD 20    // Seconds

  #define THERMAL_PROTECTION_CHAMBER_HYSTERESIS 2 // Degrees Celsius

  #define WATCH_CHAMBER_TEMP_PERIOD 60            // Seconds

  #define WATCH_CHAMBER_TEMP_INCREASE 2           // Degrees Celsius

#endif
```

Similar to the description for the Bed Thermal Protection above.
Use `M141`|(/docs/gcode/M141.html) to set target chamber temperature and `M191` to set and wait target chamber temperature.

PID Extrusion Scaling

```
#if ENABLED(PIDTEMP)

  // this adds an experimental additional term to the heating power,
proportional to the extrusion speed.

  // if Kc is chosen well, the additional required power due to increased
melting should be compensated.

  //#define PID_EXTRUSION_SCALING

  #if ENABLED(PID_EXTRUSION_SCALING)

    #define DEFAULT_Kc (100) //heating power=Kc*(e_speed)

    #define LPQ_MAX_LEN 50

  #endif

#endif
```

This option further improves hotend temperature control by accounting for the extra heat energy consumed by cold filament entering the hotend melt chamber. If material enters the hotend more quickly, then more heat will need to be added to maintain energy balance. This option adds a scaling factor that must be tuned for your setup and material.

Extrusion scaling keeps a circular buffer of forward E movements done at each temperature measurement which acts to delay the applied factor and allow for heat dissipation. The size of this queue during printing is set by `M301 L`, limited by `LPQ_MAX_LEN`.

Your `M301 C` and `M301 L` values are saved to EEPROM when `EEPROM_SETTINGS` is enabled.

Automatic Temperature

```
#define AUTOTEMP

#if ENABLED(AUTOTEMP)

  #define AUTOTEMP_OLDWEIGHT 0.98

#endif
```

With Automatic Temperature the hotend target temperature is calculated by all the buffered lines of gcode. The maximum buffered steps/sec of the extruder motor is called "`se`". Start autotemp mode with `M109 F<factor> S<mintemp> B<maxtemp>`, giving a range of temperatures. The target temperature is set to `mintemp + factor * se[steps/sec]` and is limited by `mintemp` and `maxtemp`. Turn this off by executing `M109` without `F`. If the temperature is set to a value below `mintemp` (e.g., by `M104`) autotemp will not be applied.

Example: Try `M109 S215 B260 F1` in your `start.gcode` to set a minimum temperature of 215 when idle, which will boost up to 260 as extrusion increases in speed.

Temperature Report ADC

```
//#define SHOW_TEMP_ADC_VALUES
```

Enable this option to have `M105` and automatic temperature reports include raw ADC values from the temperature sensors.

High Temperature Thermistors

```
//#define MAX_CONSECUTIVE_LOW_TEMPERATURE_ERROR_ALLOWED 0
```

High temperature thermistors may give aberrant readings. If this is an issue, use this option to set the maximum number of consecutive low temperature errors that can occur before Min Temp Error is triggered. If you require a value over 10, this could indicate a problem.

```
//#define MILLISECONDS_PREHEAT_TIME 0
```

High Temperature Thermistors tend to give poor readings at ambient and lower temperatures. Until they reach a sufficient temperature, these sensors usually return the lowest raw value, and this will cause a Min Temp Error.

To solve this issue, this option sets the number of milliseconds a hotend will preheat before Marlin starts to check the temperature. Set a delay sufficient to reach a temperature your sensor

can reliably read. Lower values are better and safer. If you require a value over 30000, this could indicate a problem.

AD595 / AD8495

```
#define TEMP_SENSOR_AD595_OFFSET  0.0

#define TEMP_SENSOR_AD595_GAIN    1.0

#define TEMP_SENSOR_AD8495_OFFSET 0.0

#define TEMP_SENSOR_AD8495_GAIN   1.0
```

These defines help to calibrate the AD595 sensor in case you get wrong temperature measurements. The final reading is derived from `measuredTemp * TEMP_SENSOR_AD595_GAIN + TEMP_SENSOR_AD595_OFFSET`.

Extruder Runout Prevention

```
//#define EXTRUDER_RUNOUT_PREVENT

#if ENABLED(EXTRUDER_RUNOUT_PREVENT)

  #define EXTRUDER_RUNOUT_MINTEMP 190

  #define EXTRUDER_RUNOUT_SECONDS 30

  #define EXTRUDER_RUNOUT_SPEED 1500  // mm/m

  #define EXTRUDER_RUNOUT_EXTRUDE 5   // mm

#endif
```

When the machine is idle and the temperature over a given value, Marlin can extrude a short length of filament every couple of seconds.

Cooling Fans

Cooling fans are needed on 3D printers to keep components cool and prevent failure.

Controller Fan

```
//#define USE_CONTROLLER_FAN

#if ENABLED(USE_CONTROLLER_FAN)

  //#define CONTROLLER_FAN_PIN -1           // Set a custom pin for the
controller fan
```

```
  #define CONTROLLERFAN_SECS 60             // Duration in seconds for the fan
to run after all motors are disabled

  #define CONTROLLERFAN_SPEED 255           // 255 == full speed

  //#define CONTROLLERFAN_SPEED_Z_ONLY 127  // Reduce noise on machines that
keep Z enabled

#endif
```

A controller fan is useful to cool down the stepper drivers and MOSFETs. When stepper drivers reach a certain temperature they'll turn off, either stuttering or stopping. With this option enabled the fan will turn on automatically whenever any steppers are enabled and turn off after a set period when all steppers are turned off.

PWM Fans Kickstart

```
//#define FAN_KICKSTART_TIME 100
```

When PWM fans are set to low speed, they may need a higher-energy kickstart first to get moving. Once up to speed the fan can drop back to the set speed. This option specifies the kickstart duration in milliseconds. **This option doesn't work with the software PWM fan on Sanguinololu.**

PWM Fans Minimum and maximum Speeds

```
//#define FAN_MIN_PWM 50

//#define FAN_MAX_PWM 128
```

This option can be defined to set the minimum and maximum PWM speeds (1-255) required to keep the PWM fans moving. Fan speeds set by `M106` will be scaled to the reduced range above this minimum.

```
#if ENABLED(FAST_PWM_FAN)

  //#define FAST_PWM_FAN_FREQUENCY 31400

  //#define USE_OCR2A_AS_TOP

#endif
```

The default frequency for `FAST_PWM_FAN` is F = F_CPU/(22551). See `configuration_adv.h` for further information.

Extruder Auto-Cooling Fans

```
#define E0_AUTO_FAN_PIN -1
```

```
#define E1_AUTO_FAN_PIN -1

#define E2_AUTO_FAN_PIN -1

#define E3_AUTO_FAN_PIN -1

#define E4_AUTO_FAN_PIN -1

#define EXTRUDER_AUTO_FAN_TEMPERATURE 50

#define EXTRUDER_AUTO_FAN_SPEED 255   // 255 == full speed

#define CHAMBER_AUTO_FAN_TEMPERATURE 30

#define CHAMBER_AUTO_FAN_SPEED 255
```

Extruder auto fans turn on whenever their extruder temperatures go above `EXTRUDER_AUTO_FAN_TEMPERATURE`. Your board's pins file already specifies the recommended pins. Override those here or set to -1 to disable the fans completely.

Multiple extruders can be assigned to the same pin in which case the fan will turn on when any selected extruder is above the threshold.

Part-Cooling Fan Multiplexer

```
#define FANMUX0_PIN -1

#define FANMUX1_PIN -1

#define FANMUX2_PIN -1
```

This feature allows you to digitally multiplex the fan output. The multiplexer is automatically switched at tool-change. To enable, just assign one or more `FANMUX[012]_PIN` values for up to 2, 4, or 8 multiplexed fans.

Case Light

```
//#define CASE_LIGHT_ENABLE

#if ENABLED(CASE_LIGHT_ENABLE)

  //#define CASE_LIGHT_PIN 4                  // Override the default pin if
needed

  #define INVERT_CASE_LIGHT false             // Set true if Case Light is ON
when pin is LOW

  #define CASE_LIGHT_DEFAULT_ON true          // Set default power-up state on
```

```
  #define CASE_LIGHT_DEFAULT_BRIGHTNESS 105   // Set default power-up
brightness (0-255, requires PWM pin)

  //#define CASE_LIGHT_MENU                   // Add Case Light options to the
LCD menu

  //#define CASE_LIGHT_NO_BRIGHTNESS          // Disable brightness control.
Enable for non-PWM lighting.

  //#define CASE_LIGHT_USE_NEOPIXEL           // Use Neopixel LED as case
light, requires NEOPIXEL_LED.

  #if ENABLED(CASE_LIGHT_USE_NEOPIXEL)

    #define CASE_LIGHT_NEOPIXEL_COLOR { 255, 255, 255, 255 } // { Red, Green,
Blue, White }

  #endif

#endif
```

Enable this option for a firmware-controlled digital or PWM case light. Use `M355` to turn on/off and control the brightness.

Endstops Always On

```
//#define ENDSTOPS_ALWAYS_ON_DEFAULT
```

Enable this option to keep the endstops on (by default) even when not homing. Override at any time with `M120`, `M121`.

Z Late Enable

```
//#define Z_LATE_ENABLE
```

With this option is active, the Z steppers will only turn on at the last moment before they move. This option may be needed if your Z driver tends to overheat. Not compatible with Core kinematics.

External Closed Loop Controller

```
//#define EXTERNAL_CLOSED_LOOP_CONTROLLER

#if ENABLED(EXTERNAL_CLOSED_LOOP_CONTROLLER)

  //#define CLOSED_LOOP_ENABLE_PIN        -1
```

```
  //#define CLOSED_LOOP_MOVE_COMPLETE_PIN -1

#endif
```

Employ an external closed loop controller that can be activated or deactivated by the main controller. Using a single wire for the control signal and another for the return “move complete” signal to signify whether or not the move was able to be made successfully.

Benefits

Dual Steppers / Dual Endstops

```
//#define X_DUAL_STEPPER_DRIVERS

#if ENABLED(X_DUAL_STEPPER_DRIVERS)

  #define INVERT_X2_VS_X_DIR true   // Set 'true' if X motors should rotate in opposite directions

  //#define X_DUAL_ENDSTOPS

  #if ENABLED(X_DUAL_ENDSTOPS)

    #define X2_USE_ENDSTOP _XMAX_

    #define X_DUAL_ENDSTOPS_ADJUSTMENT  0

  #endif

#endif



//#define Y_DUAL_STEPPER_DRIVERS

#if ENABLED(Y_DUAL_STEPPER_DRIVERS)

  #define INVERT_Y2_VS_Y_DIR true   // Set 'true' if Y motors should rotate in opposite directions

  //#define Y_DUAL_ENDSTOPS

  #if ENABLED(Y_DUAL_ENDSTOPS)

    #define Y2_USE_ENDSTOP _YMAX_

    #define Y_DUAL_ENDSTOPS_ADJUSTMENT  0

  #endif

#endif
```

```
//#define Z_DUAL_STEPPER_DRIVERS
#if ENABLED(Z_DUAL_STEPPER_DRIVERS)
  //#define Z_DUAL_ENDSTOPS
  #if ENABLED(Z_DUAL_ENDSTOPS)
    #define Z2_USE_ENDSTOP _XMAX_
    #define Z_DUAL_ENDSTOPS_ADJUSTMENT  0
  #endif
#endif

//#define Z_TRIPLE_STEPPER_DRIVERS
#if ENABLED(Z_TRIPLE_STEPPER_DRIVERS)
  //#define Z_TRIPLE_ENDSTOPS
  #if ENABLED(Z_TRIPLE_ENDSTOPS)
    #define Z2_USE_ENDSTOP _XMAX_
    #define Z3_USE_ENDSTOP _YMAX_
    #define Z_TRIPLE_ENDSTOPS_ADJUSTMENT2  0
    #define Z_TRIPLE_ENDSTOPS_ADJUSTMENT3  0
  #endif
#endif
```

These options allow you to use extra E drivers to drive a second motor for X, Y, and/or Z axes.

Set `X_DUAL_STEPPER_DRIVERS` to use a second X motor. If the X motors need to spin in opposite directions set `INVERT_X2_VS_X_DIR` to `true`. If the second motor has its own endstop set `X_DUAL_ENDSTOPS`. (This can adjust for “racking.”) Use `X2_USE_ENDSTOP` to set the endstop plug that should be used for the second endstop. Extra endstops will appear in the output of ‘M119’.

If the two X axes aren’t perfectly aligned, use `X_DUAL_ENDSTOP_ADJUSTMENT` to adjust for the difference. This offset is applied to the X2 motor after homing with `G28`. The dual endstop offsets can be set at runtime with `M666 X[offset] Y[offset] Z[offset]`.

Requires enabling the corresponding stepper driver ie `X2_DRIVER_TYPE` in `configuration.h`. Do `NOT` enable `E2_DRIVER_TYPE` - this may produce undesireabe results that can harm your machine.

Dual X Carriage

```
//#define DUAL_X_CARRIAGE

#if ENABLED(DUAL_X_CARRIAGE)

  #define X1_MIN_POS X_MIN_POS

  #define X1_MAX_POS X_BED_SIZE

  #define X2_MIN_POS    80

  #define X2_MAX_POS   353

  #define X2_HOME_DIR    1

  #define X2_HOME_POS X2_MAX_POS

  #define DEFAULT_DUAL_X_CARRIAGE_MODE DXC_AUTO_PARK_MODE

  #define DEFAULT_DUPLICATION_X_OFFSET 100

#endif
```

Enable this option if you have Dual X-Carriages that move independently. The Dual X-Carriage design allows the inactive extruder to be parked, which keeps ooze from contaminating the print, reduces the weight of each carriage, and enables faster printing speeds. With this option simply connect the X2 stepper to the first unused E plug.

In a Dual X-Carriage setup the first x-carriage (`T0`) homes to the minimum endstop, while the second x-carriage (`T1`) homes to the maximum endstop.

With Dual X-Carriage the `HOTEND_OFFSET_X` setting for `T1` overrides `X2_HOME_POS`. Use `M218 T1 X[homepos]` to set a custom X2 home position, and `M218 T1 X0` to use `X2_HOME_POS`. This offset can be saved to EEPROM with `M500`.

In your slicer, be sure to set the second extruder X-offset to 0.

Dual X-Carriage has three different movement modes, set with `M605 S[mode]`:

- Mode 0: Full Control Mode. (`M605 S1`) Slicers that fully support dual x-carriages can use this mode for optimal travel results.
- Mode 1: Auto-park Mode. (`M605 S1`) The firmware automatically parks/unparks the carriages on tool-change. No slicer support is required. (`M605 S1`)
- Mode 2: Duplication Mode. (`[M605](/docs/gcode/M605.html) S2 X[offs] R[temp]`) The firmware will transparently make the second x-carriage and extruder copy all actions of the first x-carriage. This allows the printer to print 2 arbitrary items at once. (The 2nd

extruder's X and temp offsets are set using: [M605](/docs/gcode/M605.html) S2 X[offs] R[offs].)

Solenoid

```
//#define EXT_SOLENOID
```

Adds control for any solenoid attached to an extruder. Activate the solenoid on the active extruder with M380. Disable all with M381.

Requires defining the corresponding pin ie SOL0_PIN, SOL1_PIN, etc.

Homing

```
#define X_HOME_BUMP_MM 5

#define Y_HOME_BUMP_MM 5

#define Z_HOME_BUMP_MM 2

#define HOMING_BUMP_DIVISOR { 2, 2, 4 }

//#define QUICK_HOME

//#define HOME_Y_BEFORE_X

#define AXIS_RELATIVE_MODES {false, false, false, false}

//#define DUAL_NOZZLE_DUPLICATION_MODE
```

After an endstop is triggered during homing, the printerhead backs off by the set HOME_BUMP_MM distance then homes again at a slower speed. The slower homing speed for each axis is set by HOMING_BUMP_DIVISOR.

BLTouch

```
#if ENABLED(BLTOUCH)

//#define BLTOUCH_DELAY 500

//#define BLTOUCH_FORCE_SW_MODE

//#define BLTOUCH_SET_5V_MODE

//#define BLTOUCH_FORCE_MODE_SET

//#define BLTOUCH_HS_MODE

//#define BLTOUCH_LCD_VOLTAGE_MENU
```

```
#endif
```

The default BLTouch settings can be overriden with these options. See `configuration_adv.h` for more information.

Z Steppers Auto-Alignment

```
//#define Z_STEPPER_AUTO_ALIGN

#if ENABLED(Z_STEPPER_AUTO_ALIGN)

  #define Z_STEPPER_ALIGN_X { 10, 150, 290 }

  #define Z_STEPPER_ALIGN_Y { 290, 10, 290 }

  #define Z_STEPPER_ALIGN_ITERATIONS 3

  #define RESTORE_LEVELING_AFTER_G34

  #define G34_MAX_GRADE  5

  #define Z_STEPPER_ALIGN_AMP 1.0

  #define Z_STEPPER_ALIGN_ACC 0.02

#endif
```

Add the `G34` command to align multiple Z steppers using a bed probe.

TODO Options

```
#define AXIS_RELATIVE_MODES { false, false, false, false }

//#define MULTI_NOZZLE_DUPLICATION



#define INVERT_X_STEP_PIN false

#define INVERT_Y_STEP_PIN false

#define INVERT_Z_STEP_PIN false

#define INVERT_E_STEP_PIN false



#define DEFAULT_STEPPER_DEACTIVE_TIME 120

#define DISABLE_INACTIVE_X true
```

```
#define DISABLE_INACTIVE_Y true

#define DISABLE_INACTIVE_Z true

#define DISABLE_INACTIVE_E true



#define DEFAULT_MINIMUMFEEDRATE       0.0

#define DEFAULT_MINTRAVELFEEDRATE     0.0



//#define HOME_AFTER_DEACTIVATE



#if ENABLED(ULTIPANEL)

  #define MANUAL_FEEDRATE {50*60, 50*60, 4*60, 60} // Feedrates for manual moves along X, Y, Z, E from panel

  #define ULTIPANEL_FEEDMULTIPLY  // Comment to disable setting feedrate multiplier via encoder

#endif



// minimum time in microseconds that a movement needs to take if the buffer is emptied.

#define DEFAULT_MINSEGMENTTIME        20000



// If defined the movements slow down when the look ahead buffer is only half full

#define SLOWDOWN



//#define XY_FREQUENCY_LIMIT  15



#define MINIMUM_PLANNER_SPEED 0.05
```

```
#define MICROSTEP_MODES {16,16,16,16,16} // [1,2,4,8,16]


//#define PWM_MOTOR_CURRENT { 1300, 1300, 1250 }
//#define DIGIPOT_MOTOR_CURRENT { 135,135,135,135,135 }
//#define DAC_MOTOR_CURRENT_DEFAULT { 70, 80, 90, 80 }


//#define DIGIPOT_I2C
#if ENABLED(DIGIPOT_I2C) && !defined(DIGIPOT_I2C_ADDRESS_A)
  #define DIGIPOT_I2C_ADDRESS_A 0x2C
  #define DIGIPOT_I2C_ADDRESS_B 0x2D
#endif


//#define DIGIPOT_MCP4018
#define DIGIPOT_I2C_NUM_CHANNELS 8
#define DIGIPOT_I2C_MOTOR_CURRENTS { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 }


#define ENCODER_RATE_MULTIPLIER
#define ENCODER_10X_STEPS_PER_SEC 75
#define ENCODER_100X_STEPS_PER_SEC 160
//#define CHDK 4
#define CHDK_DELAY 50


//#define LCD_INFO_MENU


//#define STATUS_MESSAGE_SCROLLING


//#define LCD_DECIMAL_SMALL_XY
```

```
//#define LCD_TIMEOUT_TO_STATUS 15000
```

SD Card Extras

The options listed below help to fix, improve, and optimize SD Card performance.

SD Detect Inverted

```
  #define SD_DETECT_INVERTED
```

Some RAMPS and other boards don't detect when an SD card is inserted. You can work around this by connecting a push button or single throw switch to the pin defined as `SD_DETECT_PIN` in your board's pins definitions. This setting should be disabled unless you are using a push button, pulling the pin to ground. Note: This option is forced off for most LCD controllers (all `ULTIPANEL` except `ELB_FULL_GRAPHIC_CONTROLLER`).

SD Finished Stepper Release

```
#define SD_FINISHED_STEPPERRELEASE true          // Disable steppers when SD Print is finished

#define SD_FINISHED_RELEASECOMMAND "[`M84`](/docs/gcode/M018.html) X Y Z E" // You might want to keep the z enabled so your bed stays in place.
```

SD Menu Autostart

```
//#define MENU_ADDAUTOSTART  // Add an option in the menu to run all auto#.g files
```

SD Card Sorting

Recent First

```
#define SDCARD_RATHERRECENTFIRST
```

Reverse SD sort to show "more recent" files first, according to the card's FAT. Since the FAT gets out of order with usage, `SDCARD_SORT_ALPHA` is recommended.

Alpha Sort

```
//#define SDCARD_SORT_ALPHA
```

```
// SD Card Sorting options

#if ENABLED(SDCARD_SORT_ALPHA)

  #define SDSORT_LIMIT       40     // Maximum number of sorted items (10-256). Costs 27 bytes each.

  #define FOLDER_SORTING     -1     // -1=above  0=none  1=below

  #define SDSORT_GCODE       false  // Allow turning sorting on/off with LCD and M34 g-code.

  #define SDSORT_USES_RAM    false  // Pre-allocate a static array for faster pre-sorting.

  #define SDSORT_USES_STACK  false  // Prefer the stack for pre-sorting to give back some SRAM. (Negated by next 2 options.)

  #define SDSORT_CACHE_NAMES false  // Keep sorted items in RAM longer for speedy performance. Most expensive option.

  #define SDSORT_DYNAMIC_RAM false  // Use dynamic allocation (within SD menus). Least expensive option. Set SDSORT_LIMIT before use!

  #define SDSORT_CACHE_VFATS 2      // Maximum number of 13-byte VFAT entries to use for sorting.

                                    // Note: Only affects SCROLL_LONG_FILENAMES with SDSORT_CACHE_NAMES but not SDSORT_DYNAMIC_RAM.

#endif
```

With this option enabled, items on SD cards will be sorted by name for easier navigation.

By default…

- Use the slowest -but safest- method for sorting.
- Folders are sorted to the top.
- The sort key is statically allocated.
- No added G-code (`M34`) support.
- 40 item sorting limit. (Items after the first 40 are unsorted.)

SD sorting uses static allocation (as set by `SDSORT_LIMIT`), allowing the compiler to calculate the worst-case usage and throw an error if the SRAM limit is exceeded.

- `SDSORT_USES_RAM` provides faster sorting via a static directory buffer.
- `SDSORT_USES_STACK` does the same, but uses a local stack-based buffer.
- `SDSORT_CACHE_NAMES` will retain the sorted file listing in RAM. (Expensive!)

- `SDSORT_DYNAMIC_RAM` only uses RAM when the SD menu is visible. (Use with caution!)

Progress Bar (character LCD)

```
//#define LCD_PROGRESS_BAR

#if ENABLED(LCD_PROGRESS_BAR)

  #define PROGRESS_BAR_BAR_TIME 2000  // Amount of time (ms) to show the progress bar

  #define PROGRESS_BAR_MSG_TIME 3000  // Amount of time (ms) to show the status message

  #define PROGRESS_MSG_EXPIRE      0  // Amount of time (ms) to retain the status message (0=forever)

  //#define PROGRESS_MSG_ONCE         // Show messages for MSG_TIME then hide them

  //#define LCD_PROGRESS_BAR_TEST     // Add a menu item to test the progress bar.

#endif
```

Show a progress bar on HD44780 LCDs for SD printing. Sub-options determine how long to show the progress bar and status message, how long to retain the status message, and whether to include a progress bar test in the Debug menu.

Set Print Progress

```
//#define LCD_SET_PROGRESS_MANUALLY
```

Add an `M73` G-code to set the current percentage.

Long Filename Host Support

```
//#define LONG_FILENAME_HOST_SUPPORT
```

Allow hosts to request long names for files and folders with `M33` `[path]`.

Scroll Long Filenames

```
//#define SCROLL_LONG_FILENAMES
```

Enable this option to scroll long filenames in the SD card menu.

Abort on Endstop Hit

```
//#define SD_ABORT_ON_ENDSTOP_HIT
```

Add an option for the firmware to abort SD printing if any endstop is triggered. Turn on with `M540 S1` (or from the LCD menu) and make sure endstops are enabled (`M120`) during SD printing.

Reprint Last File

```
//#define SD_REPRINT_LAST_SELECTED_FILE
```

This option makes it easier to print the same SD Card file again. Whenever an SD print completes the LCD Menu will open with the same file selected. From there you can click to start a new print, or you can navigate elsewhere.

Graphical Display Extras

```
#if ENABLED(DOGLCD)

  #define XYZ_HOLLOW_FRAME      // Enable to save many cycles by drawing a hollow frame on the Info Screen

  #define MENU_HOLLOW_FRAME     // Enable to save many cycles by drawing a hollow frame on Menu Screens

  //#define USE_BIG_EDIT_FONT   // A bigger font is available for edit items. Costs 3120 bytes of PROGMEM.

                                // Western only. Not available for Cyrillic, Kana, Turkish, Greek, or Chinese.

  //#define USE_SMALL_INFOFONT  // A smaller font may be used on the Info Screen. Costs 2300 bytes of PROGMEM.

                                // Western only. Not available for Cyrillic, Kana, Turkish, Greek, or Chinese.

  //#define DOGM_SPI_DELAY_US 5 // Enable this option and reduce the value to optimize screen updates.

                                // The normal delay is 10µs. Use the lowest value that still gives a reliable display.

#endif
```

Use the optimizations here to improve printing performance, which can be adversely affected by graphical display drawing, especially when doing several short moves, and when printing on DELTA and SCARA machines.

Some of these options may result in the display lagging behind controller events, as there is a trade-off between reliable printing performance versus fast display updates.

Watchdog

```
#define USE_WATCHDOG
```

The hardware watchdog should reset the microcontroller, disabling all outputs, in case the firmware gets stuck and doesn't do temperature regulation.

Watchdog Manual Reset

```
#if ENABLED(USE_WATCHDOG)

  //#define WATCHDOG_RESET_MANUAL

#endif
```

If you have a watchdog reboot in an ATmega2560 the device can hang forever, as a watchdog reset will leave the watchdog on. The `WATCHDOG_RESET_MANUAL` option works around this by eschewing the hardware reset. However, **this feature is unsafe** because it only works if interrupts are disabled, and the code could hang in an interrupt routine with interrupts disabled.

Babystepping

```
//#define BABYSTEPPING

#if ENABLED(BABYSTEPPING)

  //#define BABYSTEP_XY             // Also enable X/Y Babystepping. Not supported on DELTA!

  #define BABYSTEP_INVERT_Z false    // Change if Z babysteps should go the other way

  #define BABYSTEP_MULTIPLICATOR 1   // Babysteps are very small. Increase for faster motion.

  //#define BABYSTEP_ZPROBE_OFFSET   // Enable to combine M851 and Babystepping

  //#define DOUBLECLICK_FOR_Z_BABYSTEPPING // Double-click on the Status Screen for Z Babystepping.

  #define DOUBLECLICK_MAX_INTERVAL 1250 // Maximum interval between clicks, in milliseconds.
```

```
                                        // Note: Extra time may be added to
mitigate controller latency.

  //#define BABYSTEP_ZPROBE_GFX_OVERLAY // Enable graphical overlay on Z-offset
editor

  //#define BABYSTEP_ZPROBE_GFX_REVERSE // Reverses the direction of the CW/CCW
indicators

#endif
```

Babystepping enables `M290` and LCD menu items to move the axes by tiny increments without changing the current position values. This feature is used primarily to adjust the Z axis in the first layer of a print in real-time. Warning: Does not respect endstops!

Linear Advance

```
//#define LIN_ADVANCE



#if ENABLED(LIN_ADVANCE)

  #define LIN_ADVANCE_K 75

  #define LIN_ADVANCE_E_D_RATIO 0 // The calculated ratio (or 0) according to
the formula W * H / ((D / 2) ^ 2 * PI)

                                  // Example: 0.4 * 0.2 / ((1.75 / 2) ^ 2 *
PI) = 0.033260135

#endif
```

This feature allows Marlin to use linear pressure control for print extrusion, to eliminate ooze, improve corners, etc. See `Configuration_adv.h` and the Linear Advance page for more complete documentation.

Delta / Scara Limits

```
#if ENABLED(DELTA) && !defined(DELTA_PROBEABLE_RADIUS)

  #define DELTA_PROBEABLE_RADIUS DELTA_PRINTABLE_RADIUS

#elif IS_SCARA && !defined(SCARA_PRINTABLE_RADIUS)

  #define SCARA_PRINTABLE_RADIUS (SCARA_LINKAGE_1 + SCARA_LINKAGE_2)

#endif
```

Custom Mesh Bounds

```
#if ENABLED(MESH_BED_LEVELING) || ENABLED(AUTO_BED_LEVELING_UBL)

  // Override the mesh area if the automatic (max) area is too large

  //#define MESH_MIN_X MESH_INSET

  //#define MESH_MIN_Y MESH_INSET

  //#define MESH_MAX_X X_BED_SIZE - (MESH_INSET)

  //#define MESH_MAX_Y Y_BED_SIZE - (MESH_INSET)

#endif
```

Enhanced G-code

G2/G3 Arc Support

```
#define ARC_SUPPORT               // Disable this feature to save ~3226 bytes

#if ENABLED(ARC_SUPPORT)

  #define MM_PER_ARC_SEGMENT  1   // Length of each arc segment

  #define N_ARC_CORRECTION   25   // Number of intertpolated segments between
corrections

  //#define ARC_P_CIRCLES         // Enable the 'P' parameter to specify
complete circles

  //#define CNC_WORKSPACE_PLANES  // Allow G2/G3 to operate in XY, ZX, or YZ
planes

#endif
```

`G2/G3` Arc Support

G5 Bezier Curve

```
//#define BEZIER_CURVE_SUPPORT
```

Support for `G5` with XYZE destination and IJPQ offsets. Requires ~2666 bytes.

G38.2/G38.3 Probe Target

```
//#define G38_PROBE_TARGET
```

```
#if ENABLED(G38_PROBE_TARGET)

  #define G38_MINIMUM_MOVE 0.0275 // (mm) Minimum distance that will produce a move

#endif
```

Add commands `G38.2` and `G38.3` to probe towards target. Enable `PROBE_DOUBLE_TOUCH` if you want `G38` to double touch.

Minimum Steps Per Segment

```
#define MIN_STEPS_PER_SEGMENT 6
```

Moves (or segments) with fewer steps than this will be joined with the next move.

Minimum Stepper Pulse

```
#define MINIMUM_STEPPER_PULSE 0 // (µs) The smallest stepper pulse allowed
```

The minimum pulse width (in µs) for stepping a stepper. Set this if you find stepping unreliable, or if using a very fast CPU.

Parallel Heaters

```
//#define HEATERS_PARALLEL
```

Control heater 0 and heater 1 in parallel.

Buffer / Hosts

Block Buffer

```
#if ENABLED(SDSUPPORT)

  #define BLOCK_BUFFER_SIZE 16 // SD,LCD,Buttons take more memory, block buffer needs to be smaller

#else

  #define BLOCK_BUFFER_SIZE 16 // maximize block buffer

#endif
```

The number of linear motions that can be in the plan at any give time.
The `BLOCK_BUFFER_SIZE` must be a power of 2, (8, 16, 32, etc.) because shifts and ors are used to do the ring-buffering.

Serial Command Buffer

```
#define MAX_CMD_SIZE 96

#define BUFSIZE 4
```

The ASCII buffer for serial input. Individual command line length is set by `MAX_CMD_SIZE`, and should be long enough to hold a complete G-code line. Set the number of lines with `BUFSIZE`.

Transmit to Host Buffer

```
#define TX_BUFFER_SIZE 0
```

Transmission to Host buffer size. To save 386 bytes of PROGMEM (and `TX_BUFFER_SIZE`+3 bytes of SRAM) set to 0. To buffer a simple “ok” you need 4 bytes. An `ADVANCED_OK` (`M105`) needs 32 bytes. For debug-echo: 128 bytes for the optimal speed. Other output doesn’t need to be that speedy.

Host Receive Buffer

```
//#define RX_BUFFER_SIZE 1024

#if RX_BUFFER_SIZE >= 1024

  //#define SERIAL_XON_XOFF

#endif
```

Host Receive buffer size. Without XON/XOFF flow control (see `SERIAL_XON_XOFF` below) 32 bytes should be enough. To use flow control, set this buffer size to at least 1024 bytes.

SD Transfer Stats

```
#if ENABLED(SDSUPPORT)

  //#define SERIAL_STATS_MAX_RX_QUEUED

  //#define SERIAL_STATS_DROPPED_RX

#endif
```

Emergency Parser

```
//#define EMERGENCY_PARSER
```

Enable an emergency-command parser to intercept certain commands as they enter the serial receive buffer, so they cannot be blocked. Currently handles `M108`, `M112`, and `M410`. Does not work on boards using AT90USB (USBCON) processors!

No Timeouts

```
//#define NO_TIMEOUTS 1000 // (ms)
```

Bad serial connections can miss a received command by sending an “ok”, and some hosts will abort after 30 seconds. Some hosts start sending commands while receiving a ‘wait’. This “wait” is only sent when the buffer is empty. 1 second is a good value here.
The `HOST_KEEPALIVE` feature provides another way to keep the host alive.

Advanced OK

```
//#define ADVANCED_OK
```

Include extra information about the buffer in “ok” messages. Some hosts will have this feature soon. This could make the `NO_TIMEOUTS` unnecessary.

Firmware Retraction

```
//#define FWRETRACT  // ONLY PARTIALLY TESTED

#if ENABLED(FWRETRACT)

  #define MIN_AUTORETRACT 0.1             // When auto-retract is on, convert E moves of this length and over

  #define MAX_AUTORETRACT 10.0            // Upper limit for auto-retract conversion

  #define RETRACT_LENGTH 3                // Default retract length (positive mm)

  #define RETRACT_LENGTH_SWAP 13          // Default swap retract length (positive mm), for extruder change

  #define RETRACT_FEEDRATE 45             // Default feedrate for retracting (mm/s)

  #define RETRACT_ZLIFT 0                 // Default retract Z-lift

  #define RETRACT_RECOVER_LENGTH 0        // Default additional recover length (mm, added to retract length when recovering)

  #define RETRACT_RECOVER_LENGTH_SWAP 0   // Default additional swap recover length (mm, added to retract length when recovering from extruder change)
```

```
  #define RETRACT_RECOVER_FEEDRATE 8      // Default feedrate for recovering
from retraction (mm/s)

  #define RETRACT_RECOVER_FEEDRATE_SWAP 8 // Default feedrate for recovering
from swap retraction (mm/s)

#endif
```

This option adds `G10`/`G11` commands for automatic firmware-based retract/recover.
Use `M207` and `M208` to set the parameters, and `M209` to enable/disable. With auto-retract enabled, all `G1 E` moves within the set range will be converted to firmware-based retract/recover moves.

Be sure to turn off auto-retract during filament change! All `M207`/`M208`/`M209` settings are saved to EEPROM.

Extra Fan Speed

```
//#define EXTRA_FAN_SPEED
```

Add a secondary fan speed for each print-cooling fan. `M106`

- `M106 P[fan] T3-255` sets a secondary speed for [fan].
- `M106 P[fan] T2` uses the set secondary speed.
- `M106 P[fan] T1` restores the previous fan speed

Advanced Pause

```
//#define ADVANCED_PAUSE_FEATURE

#if ENABLED(ADVANCED_PAUSE_FEATURE)

  #define PAUSE_PARK_X_POS 3

  #define PAUSE_PARK_Y_POS 3

  #define PAUSE_PARK_Z_ADD 10

  #define PAUSE_PARK_XY_FEEDRATE 100

  #define PAUSE_PARK_Z_FEEDRATE 5

  #define PAUSE_PARK_RETRACT_FEEDRATE 60

  #define PAUSE_PARK_RETRACT_LENGTH 2

  #define FILAMENT_CHANGE_UNLOAD_FEEDRATE 10

  #define FILAMENT_CHANGE_UNLOAD_LENGTH 100
```

```
  #define FILAMENT_CHANGE_LOAD_FEEDRATE 6

  #define FILAMENT_CHANGE_LOAD_LENGTH 0

  #define ADVANCED_PAUSE_EXTRUDE_FEEDRATE 3

  #define ADVANCED_PAUSE_EXTRUDE_LENGTH 50

  #define PAUSE_PARK_NOZZLE_TIMEOUT 45        // Turn off nozzle if user
doesn't change filament within this time limit in seconds

  #define FILAMENT_CHANGE_NUMBER_OF_ALERT_BEEPS 5 // Number of alert beeps
before printer goes quiet

  #define PAUSE_PARK_NO_STEPPER_TIMEOUT       // Enable to have stepper motors
hold position during filament change

                                               // even if it takes longer than
DEFAULT_STEPPER_DEACTIVE_TIME.

  //#define PARK_HEAD_ON_PAUSE                // Go to filament change position
on pause, return to print position on resume

  //#define HOME_BEFORE_FILAMENT_CHANGE       // Ensure homing has been
completed prior to parking for filament change

#endif
```

Experimental feature for filament change support and parking the nozzle when paused. Adds the `M600` command to perform a filament change. With `PARK_HEAD_ON_PAUSE` enabled also adds the `M115` command to pause printing and park the nozzle. Requires an LCD display. Note that `M600` is required for the default `FILAMENT_RUNOUT_SCRIPT`.

Stepper Drivers

Trinamic TMC26X

```
//#define HAVE_TMCDRIVER
```

Enable this section if you have TMC26X motor drivers. You'll need to import the TMC26XStepper library into the Arduino IDE. See the `Configuration_adv.h` file for the full set of sub-options.

Trinamic TMC2130

```
//#define HAVE_TMC2130
```

Enable this option for SilentStepStick Trinamic TMC2130 SPI-configurable stepper drivers. You'll also need the TMC2130Stepper Arduino library. See the `Configuration_adv.h` file for the full set of sub-options.

To use TMC2130 stepper drivers in SPI mode connect your SPI2130 pins to the hardware SPI interface on your board and define the required CS pins in your `pins_MYBOARD.h` file. (e.g., RAMPS 1.4 uses AUX3 pins `X_CS_PIN 53`, `Y_CS_PIN 49`, etc.).

L6470 Drivers

```
//#define HAVE_L6470DRIVER
```

Enable this section if you have L6470 motor drivers. You need to import the L6470 library into the Arduino IDE for this. See the `Configuration_adv.h` file for the full set of sub-options.

Experimental i2c Bus

```
//#define EXPERIMENTAL_I2CBUS

#define I2C_SLAVE_ADDRESS  0 // Set a value from 8 to 127 to act as a slave
```

This feature can be used to talk to slave devices on the i2c bus, passing data back to the host. With additional work the `TWIBus` class can be used to build a full protocol and add remote control features to Marlin, distributing load over two or more boards.

```
; Example #1
; This macro send the string "Marlin" to the slave device with address 0x63
(99)
; It uses multiple [`M260`](/docs/gcode/M260.html) commands with one B[base 10]
arg
[`M260`](/docs/gcode/M260.html) A99  ; Target slave address
M260 B77  ; M
M260 B97  ; a
M260 B114 ; r
M260 B108 ; l
M260 B105 ; i
M260 B110 ; n
M260 S1   ; Send the current buffer

; Example #2
; Request 6 bytes from slave device with address 0x63 (99)
[`M261`](/docs/gcode/M261.html) A99 B5

; Example #3
; Example serial output of a M261 request
echo:i2c-reply: from:99 bytes:5 data:hello
```

Spindle / Laser

```
//#define SPINDLE_LASER_ENABLE

#if ENABLED(SPINDLE_LASER_ENABLE)



  #define SPINDLE_LASER_ENABLE_INVERT   false  // set to "true" if the on/off
function is reversed

  #define SPINDLE_LASER_PWM             true   // set to true if your
controller supports setting the speed/power

  #define SPINDLE_LASER_PWM_INVERT      true   // set to "true" if the
speed/power goes up when you want it to go slower

  #define SPINDLE_LASER_POWERUP_DELAY   5000   // delay in milliseconds to
allow the spindle/laser to come up to speed/power

  #define SPINDLE_LASER_POWERDOWN_DELAY 5000   // delay in milliseconds to
allow the spindle to stop

  #define SPINDLE_DIR_CHANGE            true   // set to true if your spindle
controller supports changing spindle direction

  #define SPINDLE_INVERT_DIR            false

  #define SPINDLE_STOP_ON_DIR_CHANGE    true   // set to true if Marlin should
stop the spindle before changing rotation direction



  #define SPEED_POWER_SLOPE    118.4

  #define SPEED_POWER_INTERCEPT  0

  #define SPEED_POWER_MIN     5000

  #define SPEED_POWER_MAX    30000    // SuperPID router controller 0 - 30,000
RPM



  //#define SPEED_POWER_SLOPE      0.3922

  //#define SPEED_POWER_INTERCEPT  0

  //#define SPEED_POWER_MIN       10
```

```
  //#define SPEED_POWER_MAX      100      // 0-100%

#endif
```

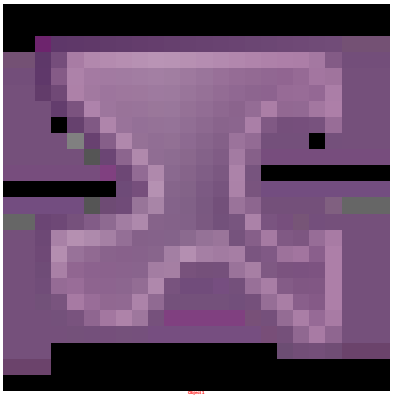
Enable for Spindle and Laser control. Adds the `M3`, `M4`, and `M5` commands to turn the spindle/laser on and off, and to set spindle speed, spindle direction, and laser power.

SuperPid is a router/spindle speed controller used in the CNC milling community. Marlin can be used to turn the spindle on and off. It can also be used to set the spindle speed from 5,000 to 30,000 RPM.

You'll need to select a pin for the ON/OFF function and optionally choose a 0-5V hardware PWM pin for the speed control and a pin for the rotation direction.

See the Laser and Spindle page for more details.

Filament Width Sensor



```
//#define FILAMENT_WIDTH_SENSOR
```

Enable to add support for a filament width sensor such as Filament Width Sensor Prototype Version 3. With a filament sensor installed, Marlin can adjust the flow rate according to the measured filament width. Adjust the sub-options below according to your setup.

Only a single extruder is supported at this time.

```
#define FILAMENT_SENSOR_EXTRUDER_NUM 0
```

Only one extruder can have a filament sensor. Specify here which extruder has it.

```
#define MEASUREMENT_DELAY_CM        14
```

Distance from the filament width sensor to the melt chamber.

```
#define MEASURED_UPPER_LIMIT         3.30 // (mm) Upper limit used to validate
sensor reading
```

```
#define MEASURED_LOWER_LIMIT         1.90 // (mm) Lower limit used to validate sensor reading
```

The range of your filament width. Set these according to your filament preferences. The sample values here apply to 3mm. For 1.75mm you'll use a range more like 1.60 to 1.90.

```
#define MAX_MEASUREMENT_DELAY       20
```

This defines the size of the buffer to allocate for use with `MEASUREMENT_DELAY_CM`. The value must be greater than or equal to `MEASUREMENT_DELAY_CM`. Keep this setting low to reduce RAM usage.

```
#define FILAMENT_LCD_DISPLAY
```

Periodically display a message on the LCD showing the measured filament diameter.

CNC Coordinate Systems

```
//#define CNC_COORDINATE_SYSTEMS
```

Enables `G53` and `G54`-`G59.3` commands to select coordinate systems, plus `G92.1` to reset the current workspace to native machine space. Workspaces set with this feature are also saved to EEPROM.

Pins Debugging

```
//#define PINS_DEBUGGING
```

Enable this option to add the `M43` Debug Pins G-code. This command can be used to list pins, display their status, to watch pins for changes, observe endstops, toggle LEDs, test Z servo probe, toggle pins, etc.

Temperature Auto-Report

```
#define AUTO_REPORT_TEMPERATURES
```

It is recommended to enable this feature (along with `EXTENDED_CAPABILITIES_REPORT`) to install the `M155` Auto-Report Temperature command. `M115` tells Marlin to send the current temperature to the host at regular intervals, instead of requiring the host software to send `M105` repeatedly. This saves a space in the command buffer and reduces overhead.

Extended Capabilities Report

```
#define EXTENDED_CAPABILITIES_REPORT
```

This option adds a list of capabilities to the output of `M115`, allowing savvy host software to take advantage of add-ons like `AUTO_REPORT_TEMPERATURES`.

Volumetric Mode Default

```
//#define VOLUMETRIC_DEFAULT_ON
```

Activate this option to make volumetric extrusion the default method The last values loaded or set by `M404 W` and `M200 D` will be used as the Nominal and Actual filament diameters. With this option, `M200 D0` must be used to disable volumetric mode when running length-based G-code.

No Workspace Offsets

```
//#define NO_WORKSPACE_OFFSETS
```

Enable this option for a leaner build of Marlin that removes all workspace offsets. This simplifies all coordinate transformations, leveling, etc., and may allow for slightly faster printing. With this option, `M206` and `M428` are disabled, and `G92` reverts to its old behavior, as it is in Marlin 1.0.

Proportional Font Ratio

```
#define PROPORTIONAL_FONT_RATIO 1.0
```

Some hosts use a proportional font in their output console. This makes it hard to read output from Marlin that relies on fixed-width for alignment. This option tells Marlin how many spaces are required to fill up a typical character space in the host font. For clients that use a fixed-width font (like OctoPrint), leave this set to 1.0. Otherwise, adjust according to your host.

Faster G-code Parser

```
#define FASTER_GCODE_PARSER
```

This option uses a 28 byte SRAM buffer and an alternative method to get parameter values so the G-code parser can run a little faster. If possible, always leave this option enabled.

Even More Options…

```
/**
 * User-defined menu items that execute custom GCode
 */
//#define CUSTOM_USER_MENUS
#if ENABLED(CUSTOM_USER_MENUS)
```

```
  #define USER_SCRIPT_DONE "M117 User Script Done"

  #define USER_SCRIPT_AUDIBLE_FEEDBACK

  //#define USER_SCRIPT_RETURN  // Return to status screen after a script



  #define USER_DESC_1 "Home & UBL Info"

  #define USER_GCODE_1 "G28\nG29 W"



  #define USER_DESC_2 "Preheat for PLA"

  #define USER_GCODE_2 "M140 S" STRINGIFY(PREHEAT_1_TEMP_BED) "\nM104 S"
STRINGIFY(PREHEAT_1_TEMP_HOTEND)



  #define USER_DESC_3 "Preheat for ABS"

  #define USER_GCODE_3 "M140 S" STRINGIFY(PREHEAT_2_TEMP_BED) "\nM104 S"
STRINGIFY(PREHEAT_2_TEMP_HOTEND)



  #define USER_DESC_4 "Heat Bed/Home/Level"

  #define USER_GCODE_4 "M140 S" STRINGIFY(PREHEAT_2_TEMP_BED) "\nG28\nG29"



  #define USER_DESC_5 "Home & Info"

  #define USER_GCODE_5 "G28\nM503"

#endif



//#define ACTION_ON_KILL "poweroff"



//#define I2C_POSITION_ENCODERS

#if ENABLED(I2C_POSITION_ENCODERS)
```

```
  #define I2CPE_ENCODER_CNT        1                      // The number of
encoders installed; max of 5

                                                            // encoders
supported currently.



  #define I2CPE_ENC_1_ADDR         I2CPE_PRESET_ADDR_X    // I2C address of
the encoder. 30-200.

  #define I2CPE_ENC_1_AXIS         X_AXIS                 // Axis the encoder
module is installed on.  <X|Y|Z|E>_AXIS.

  #define I2CPE_ENC_1_TYPE         I2CPE_ENC_TYPE_LINEAR  // Type of encoder:
I2CPE_ENC_TYPE_LINEAR -or-

                                                            //
I2CPE_ENC_TYPE_ROTARY.

  #define I2CPE_ENC_1_TICKS_UNIT   2048                   // 1024 for
magnetic strips with 2mm poles; 2048 for

  //#define I2CPE_ENC_1_TICKS_REV     (16 * 200)          // Only needed for
rotary encoders; number of stepper

  //#define I2CPE_ENC_1_INVERT                           // Invert the
direction of axis travel.

  #define I2CPE_ENC_1_EC_METHOD    I2CPE_ECM_NONE         // Type of error
error correction.

  #define I2CPE_ENC_1_EC_THRESH    0.10                   // Threshold size
for error (in mm) above which the



  #define I2CPE_ENC_2_ADDR         I2CPE_PRESET_ADDR_Y    // Same as above,
but for encoder 2.

  #define I2CPE_ENC_2_AXIS         Y_AXIS

  #define I2CPE_ENC_2_TYPE         I2CPE_ENC_TYPE_LINEAR

  #define I2CPE_ENC_2_TICKS_UNIT   2048

  //#define I2CPE_ENC_2_TICKS_REV   (16 * 200)

  //#define I2CPE_ENC_2_INVERT
```

```
  #define I2CPE_ENC_2_EC_METHOD     I2CPE_ECM_NONE

  #define I2CPE_ENC_2_EC_THRESH     0.10


  #define I2CPE_ENC_3_ADDR          I2CPE_PRESET_ADDR_Z     // Encoder 3.  Add
additional configuration options

  #define I2CPE_ENC_3_AXIS          Z_AXIS                  // as above, or use
defaults below.


  #define I2CPE_ENC_4_ADDR          I2CPE_PRESET_ADDR_E     // Encoder 4.

  #define I2CPE_ENC_4_AXIS          E_AXIS


  #define I2CPE_ENC_5_ADDR          34                      // Encoder 5.

  #define I2CPE_ENC_5_AXIS          E_AXIS


  #define I2CPE_DEF_TYPE            I2CPE_ENC_TYPE_LINEAR

  #define I2CPE_DEF_ENC_TICKS_UNIT  2048

  #define I2CPE_DEF_TICKS_REV       (16 * 200)

  #define I2CPE_DEF_EC_METHOD       I2CPE_ECM_NONE

  #define I2CPE_DEF_EC_THRESH       0.1


  //#define I2CPE_ERR_THRESH_ABORT  100.0


  #define I2CPE_TIME_TRUSTED        10000

  #define I2CPE_MIN_UPD_TIME_MS     100

  #define I2CPE_ERR_ROLLING_AVERAGE


#endif
```

```
//#define MAX7219_DEBUG

#if ENABLED(MAX7219_DEBUG)

  #define MAX7219_CLK_PIN   64  // 77 on Re-ARM

  // Configuration of the 3 pins to control the display

  #define MAX7219_DIN_PIN   57  // 78 on Re-ARM

  #define MAX7219_LOAD_PIN  44  // 79 on Re-ARM



  #define MAX7219_DEBUG_PRINTER_ALIVE    // Blink corner LED of 8x8 matrix to
show that the firmware is functioning

  #define MAX7219_DEBUG_STEPPER_HEAD  3  // Show the stepper queue head
position on this and the next LED matrix row

  #define MAX7219_DEBUG_STEPPER_TAIL  5  // Show the stepper queue tail
position on this and the next LED matrix row



  #define MAX7219_DEBUG_STEPPER_QUEUE 0  // Show the current stepper queue
depth on this and the next LED matrix row

#endif
```

Prusa MMU2 advanced settings

Serial connection

A serial connection is required for communication between the printer board and the MMU2. The configuration differs between 8- and 32-bit boards.

8-bit AVR boards

On a board with a ATmega2560/1280 microcontroller you have three potential serial ports to use for the MMU2: serial 1 (pins 18/19), serial 2 (pins 16/17), serial 3 (pins 14/15). Define the port your MMU2 is connected to

```
  #define INTERNAL_SERIAL_PORT 2
```

This activates an additional serial connection in Marlin named internalSerial. So the second define in the example configuration can just remain as it is.

```
#define MMU2_SERIAL internalSerial
```

32-bit boards

When using a 32-bit board you just have to define the name of the serial port which will be used for communication with the MMU2.

```
#define MMU2_SERIAL Serial1
```

MMU2 Reset

The MMU2 provides two options how the printer board can trigger a reset: software and hardware reset. By default software reset is enabled. Hardware reset requires a digital output pin wired to the reset pin on the MMU2. To activate hardware reset you define the pin to use on the printer board

```
#define MMU2_RST_PIN 23
```

12V mode

If your MMU2 is powered from 12 V you can activate a special mode on the MMU2.

```
// Enable if the MMU2 has 12V stepper motors (MMU2 Firmware 1.0.2 and up)

#define MMU2_MODE_12V
```

This should reduce the noise of the MMU2 but has no effect on the general operation.

Filament runout handling

Here you define the gcode script which will be executed when the so-called FINDA sensor on the MMU2 detects a filament runout.

```
// G-code to execute when MMU2 F.I.N.D.A. probe detects filament runout

#define MMU2_FILAMENT_RUNOUT_SCRIPT "M600"
```

The default is `M600` which requires ADVANCED_PAUSE_FEATURE.

LCD Menu

```
// Add MMU2 controls to the LCD menu

#define MMU2_MENUS
```

Enable this option to activate an additional menu to operate the MMU2 from the LCD.

Filament load/unload settings

Load to nozzle

The MMU2 LCD menu allows you to load filament to the nozzle. The MMU2 will transport the filament all the way to the extruder gears. The required extruder steps to load it into the hotend have to be defined in Marlin.

```
// This is for Prusa MK3-style extruders. Customize for your hardware.
#define MMU2_LOAD_TO_NOZZLE_SEQUENCE \
  {  7.2,  562 }, \
  { 14.4,  871 }, \
  { 36.0, 1393 }, \
  { 14.4,  871 }, \
  { 50.0,  198 }
```

The values are relative E distances and feed rates in mm/m. The defaults are based on the nozzle to extruder gear distance of a Prusa MK3 extruder, so if required you have to modifiy those to your extruder/hotend setup accordingly.

Unload filament

To unload filament using the LCD menu a generic ramming sequence will be exectued before the MMU2 will retract the filament. The steps to do so are defined using

```
#define MMU2_RAMMING_SEQUENCE \
  {   1.0, 1000 }, \
  {   1.0, 1500 }, \
  {   2.0, 2000 }, \
  {   1.5, 3000 }, \
  {   2.5, 4000 }, \
  { -15.0, 5000 }, \
  { -14.0, 1200 }, \
  {  -6.0,  600 }, \
```

```
      {  10.0,  700 }, \
      { -10.0,  400 }, \
      { -50.0, 2000 }
```

The values are relative E distances and feed rates in mm/m. The default values are based on a E3D V6 hotend and the nozzle to extruder gear distance of a Prusa MK3 extruder, so if required you have to modifiy those to your extruder/hotend setup accordingly.

Eject filament

Eject filament will do a simple retraction of the filament out of the hotend without ramming. The feedrate to do so is defined using

```
    #define MMU2_FILAMENTCHANGE_EJECT_FEED 80.0
```

Debug

```
  #define MMU2_DEBUG  // Write debug info to serial output
```

Enable this option to get debug output related to the printer to MMU2 communication. This will consume some PROGMEM.