Configuring Marlin

This document is based on Marlin 1.1.0 RC8.

Introduction

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 `example_configurations` 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.

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 RepRap machine before, here are some good resources:

- Calibration (http://reprap.org/wiki/Calibration)
- Calibrating Steps-per-unit (http://youtu.be/wAL9d7FgInk)
- Prusa's calculators (http://calculator.josefprusa.cz)
- Triffid Hunter's Calibration Guide (http://reprap.org/wiki/Triffid_Hunter%27s_Calibration_Guide)
- The Essential Calibration Set (http://www.thingiverse.com/thing:5573)
- Calibration of your RepRap (https://sites.google.com/site/repraplogphase/calibration-of-your-reprap)
- XY 20 mm Calibration Box (http://www.thingiverse.com/thing:298812)
- G-Code reference (http://reprap.org/wiki/G-code)
- Marlin3DprinterTool (https://github.com/cabbagecreek/Marlin3DprinterTool)

Before You Begin

The settings in `Configuration.h` you'll need to know include:

- 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`

This document covers the `Configuration.h` file, following 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 010100
```

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 STRING_SPLASH_LINE1 SHORT_BUILD_VERSION // will be shown during bootup in line 1
#define STRING_SPLASH_LINE2 WEBSITE_URL         // will be shown during bootup in line 2
```

- `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, and whenever the board reboots.
- `SHOW_BOOTSCREEN` enables the boot screen for LCD controllers.
- `STRING_SPLASH_LINE1` and `STRING_SPLASH_LINE1` are shown on the boot 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. Serial port 0 will be used by the Arduino bootloader regardless of this setting.

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 here.

Motherboard	Core	Pins	Board name
BOARD_3DRAG	Arduino	pins_3DRAG.h	3DRAG RAMPS v1.4
BOARD_AJ4P	Arduino	pins_A4JP.h	RAMBo "A4JP"
BOARD_AZTEEG_X3	Arduino	pins_AZTEEG_X3.h	Azteeg X3 RAMPS v1.4
BOARD_AZTEEG_X3_PRO	Arduino	pins_AZTEEG_X3_PRO.h	Azteeg X3 PRO RAMPS v1.4
BOARD_BAM_DICE	Arduino	pins_RAMPS_14.h	BAM&DICE
BOARD_BAM_DICE_DUE	Arduino	pins_BAM_DICE_DUE.h	BAM&DICE Due
BOARD_BQ_ZUM_MEGA_3D	Arduino	pins_BQ_ZUM_MEGA_3D.h	bq ZUM Mega 3D
BOARD_CHEAPTRONIC	Arduino	pins_CHEAPTRONIC.h	Cheaptronic v1.0
BOARD_CNCONTROLS_12	Arduino	pins_CNCONTROLS_12.h	Cartesio CN Controls V12
BOARD_ELEFU_3	Arduino	pins_ELEFU_3.h	Elefu RA
BOARD_FELIX2	Arduino	pins_FELIX2.h	FELIXprinters v2.0/3.0
BOARD_K8200	Arduino	pins_K8200.h	K8200 RAMPS v1.3
BOARD_LEAPFROG	Arduino	pins_LEAPFROG.h	Leapfrog Driver board
BOARD_MEGACONTROLLER	Arduino	pins_MEGACONTROLLER.h	Mega Controller
BOARD_MEGATRONICS	Arduino	pins_MEGATRONICS.h	MegaTronics
BOARD_MEGATRONICS_2	Arduino	pins_MEGATRONICS_2.h	MegaTronics v2.0
BOARD_MEGATRONICS_3	Arduino	pins_MEGATRONICS_3.h	MegaTronics v3.0
BOARD_MINIRAMBO	Arduino	pins_MINIRAMBO.h	Mini RAMBo
BOARD_MKS_13	Arduino	pins_MKS_13.h	MKS v1.3
BOARD_MKS_BASE	Arduino	pins_MKS_BASE.h	MKS BASE 1.0
BOARD_RAMBO	Arduino	pins_RAMBO.h	RAMBo

Motherboard	Core	Pins	Board name
BOARD_RAMPS_13_EEB	Arduino	pins_RAMPS_13.h	RAMPS v1.3 (Extruder, Extruder, Bed)
BOARD_RAMPS_13_EEF	Arduino	pins_RAMPS_13.h	RAMPS v1.3 (Extruder, Extruder, Fan)
BOARD_RAMPS_13_EFB	Arduino	pins_RAMPS_13_EFB.h	RAMPS v1.3 (Extruder, Fan, Bed)
BOARD_RAMPS_13_EFF	Arduino	pins_RAMPS_13.h	RAMPS v1.3 (Extruder, Fan, Fan)
BOARD_RAMPS_13_SF	Arduino	pins_RAMPS_13.h	RAMPS v1.3 (Spindle, Controller Fan)
BOARD_RAMPS_14_EEB	Arduino	pins_RAMPS_14.h	RAMPS v1.4 (Extruder, Extruder, Bed)
BOARD_RAMPS_14_EEF	Arduino	pins_RAMPS_14.h	RAMPS v1.4 (Extruder, Extruder, Fan)
BOARD_RAMPS_14_EFB	Arduino	pins_RAMPS_14_EFB.h	RAMPS v1.4 (Extruder, Fan, Bed)
BOARD_RAMPS_14_EFF	Arduino	pins_RAMPS_14.h	RAMPS v1.4 (Extruder, Fan, Fan)
BOARD_RAMPS_14_SF	Arduino	pins_RAMPS_14.h	RAMPS v1.4 (Spindle, Controller Fan)
BOARD_RAMPS_OLD	Arduino	pins_RAMPS_OLD.h	RAMPS v1.0, v1.1, v1.2
BOARD_RIGIDBOARD	Arduino	pins_RIGIDBOARD.h	RIGIDBOARD RAMPS v1.4
BOARD_RIGIDBOARD_V2	Arduino	pins_RIGIDBOARD_V2.h	RIGIDBOARD v2
BOARD_RUMBA	Arduino	pins_RUMBA.h	RUMBA
BOARD_SAINSMART_2IN1	Arduino	pins_SAINSMART_2IN1.h	Sainsmart 2-in-1
BOARD_ULTIMAIN_2	Arduino	pins_ULTIMAIN_2.h	Ultiboard v2.0
BOARD_ULTIMAKER	Arduino	pins_ULTIMAKER.h	Ultimaker
BOARD_ULTIMAKER_OLD	Arduino	pins_ULTIMAKER_OLD.h	Ultimaker "old electronics"
BOARD_BRAINWAVE	Brainwave	pins_BRAINWAVE.h	Brainwave 1.0
BOARD_BRAINWAVE_PRO	Brainwave	pins_BRAINWAVE_PRO.h	Brainwave Pro
BOARD_GEN7_12	Gen7	pins_GEN7_12.h	Gen7 v1.1, v1.2
BOARD_GEN7_13	Gen7	pins_GEN7_13.h	Gen7 v1.3
BOARD_GEN7_14	Gen7	pins_GEN7_14.h	Gen7 v1.4
BOARD_GEN7_CUSTOM	Gen7	pins_GEN7_CUSTOM.h	Gen7 "Alfons3"
BOARD_MINITRONICS	Minitronics	pins_MINITRONICS.h	Minitronics v1.0/1.1
BOARD_AZTEEG_X1	Sanguino	pins_AZTEEG_X1.h	Azteeg X1
BOARD_GEN3_MONOLITHIC	Sanguino	pins_GEN3_MONOLITHIC.h	Gen3 Monolithic Electronics
BOARD_GEN3_PLUS	Sanguino	pins_GEN3_PLUS.h	Gen3+
BOARD_GEN6	Sanguino	pins_GEN6.h	Gen6
BOARD_GEN6_DELUXE	Sanguino	pins_GEN6_DELUXE.h	Gen6 Deluxe
BOARD_MELZI	Sanguino	pins_MELZI.h	Melzi
BOARD_MELZI_MAKR3D	Sanguino	pins_MELZI_MAKR3D.h	Melzi "MaKr3d"
BOARD_OMCA_A	Sanguino	pins_OMCA.h	Open Motion controller
BOARD_SANGUINOLOLU_11	Sanguino	pins_SANGUINOLOLU_11.h	Sanguinololu
BOARD_SANGUINOLOLU_12	Sanguino	pins_SANGUINOLOLU_12.h	Sanguinololu v1.2
BOARD_STB_11	Sanguino	pins_STB_11.h	STB V1.1
BOARD_OMCA	SanguinoA	pins_OMCA_A.h	Open Motion controller "Alpha"
BOARD_SETHI	Sethi 3D	pins_SETHI.h	Sethi 3D_1
BOARD_5DPRINT	Teensy++ 2.0	pins_5DPRINT.h	Makibox 5DPD8
BOARD_PRINTRBOARD	Teensy++ 2.0	pins_PRINTRBOARD.h	Printrboard
BOARD_PRINTRBOARD_REVF	Teensy++ 2.0	pins_PRINTRBOARD_REVF.h	Printrboard RevF
BOARD_SAV_MKI	Teensy++ 2.0	pins_SAV_MKI.h	SAV MkI
BOARD_TEENSY2	Teensy++ 2.0	pins_TEENSY2.h	Teensy++ 2.0
BOARD_TEENSYLU	Teensy++ 2.0	pins_TEENSYLU.h	Teensylu 0.7

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 (https://raw.githubusercontent.com/Lauszus/Sanguino/master/package_lauszus_sanguino_index.json). Then use `Tools > Boards > Boards Manager` to install “Sanguino” from the list. An internet connection is required. (Thanks to Dust’s RepRap Blog (http://dustsreprap.blogspot.my/2015/06/better-way-to-install-sanguino-in.html) 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.

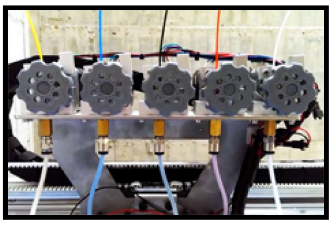
Machine UUID

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

A unique ID for your 3D printer. A suitable unique ID can be generated randomly at uuidgenerator.net (http://www.uuidgenerator.net/version4). Some host programs and slicers may use this identifier to differentiate between specific machines on your network.

Extruder Info

(https://www.youtube.com/watch?v=ocgPAAJouPs)



Extruders

```
#define EXTRUDERS 1
```

This value, from 1 to 4, 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.

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.

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.

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
  //#define HOTEND_OFFSET_Z {0.0, 0.0}
#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.

Mixing Extruder

```
/**
 * "Mixing Extruder"
 *   - Adds a new code, M165, to set the current mix factors.
 *   - Extends the stepping routines to move multiple steppers in proportion to the mix.
 *   - Optional support for Repetier Host M163, M164, and virtual extruder.
 *   - This implementation supports only a single extruder.
 *   - Enable DIRECT_MIXING_IN_G1 for Pia Taubert's reference implementation
 */
//#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 and M164
  //#define DIRECT_MIXING_IN_G1    // Allow ABCDHI mix factors in G1 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.

Hotend Offsets

```
//#define HOTEND_OFFSET_X {0.0, 20.00} // (in mm) for each extruder, offset of the hotend on the X axis
//#define HOTEND_OFFSET_Y {0.0, 5.00}  // (in mm) for each extruder, offset of the hotend on the Y axis
```

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

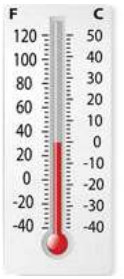
```
#define POWER_SUPPLY 1
```

Use this option to specify the type of power supply you're using. Marlin uses this setting to decide how to switch the power supply on and off. The options are None (0), ATX (1), or X-Box 360 (2). For a non-switchable power supply use 0. A common example of this is the power "brick" (like a big laptop power supply). For a PC power supply (ATX) or LED Constant-Voltage Power Supply select 1. These are the most commonly-used power supplies.

```
//#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 powersupplies. For instance some setups have a separate powersupply for the heaters. In this situation you can save power by leaving the powersupply off until called for. If you don't know what this is leave it.

Thermal Settings



Temperature Sensors

```
#define TEMP_SENSOR_0 5
#define TEMP_SENSOR_1 0
#define TEMP_SENSOR_2 0
#define TEMP_SENSOR_3 0
#define TEMP_SENSOR_BED 3
```

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.
```

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`.

Temperature Ranges

```
#define HEATER_0_MINTEMP 5
#define HEATER_1_MINTEMP 5
#define HEATER_2_MINTEMP 5
#define HEATER_3_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 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.

PID

Marlin uses PID (Proportional, Integral, Derivative) control (Wikipedia (https://en.wikipedia.org/wiki/PID_controller)) 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 (http://reprap.org/wiki/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

```
#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
```

Disable `PIDTEMP` if you want to run your heater in bang-bang mode. Bang_bang is a pure binary mode where the heater is either full on or full off. PID control is PWM and in most cases is superior in it's ability to maintain a stable temperature.

```
#if ENABLED(PIDTEMP)
  //#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
  #define K1 0.95
```

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.

```
  // 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

```
//#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 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.

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
```

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 thermistor, 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.

Delta

```
//#define DELTA
```

For Delta use one of the sample configurations in the `example_configurations/delta` folder as a starting point.

SCARA

```
//#define SCARA
```

For SCARA use the sample configuration in the `example_configurations/SCARA` folder as a starting point.

```
// Enable this option for Toshiba steppers
//#define CONFIG_STEPPERS_TOSHIBA
```

Leave this option disabled for standard NEMA steppers.

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.

Endstop Pullups

```
#define ENDSTOPPULLUPS

#if DISABLED(ENDSTOPPULLUPS)
  // fine endstop settings: Individual pullups. will be ignored if ENDSTOPPULLUPS is defined
  //#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 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.
```

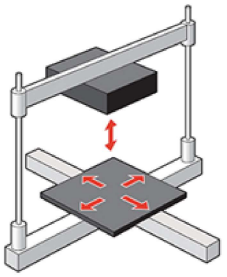
Use `M119` to test if these are set correctly. If an endstop shows up as "TRIGGERED" when not pressed, and "open" when pressed, then it should be inverted here.

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.

Movement



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.

Default Steps per mm

```
/**
 * Default Axis Steps Per Unit (steps/mm)
 * Override with M92
 *                                      X, Y, Z, E0 [, E1[, E2[, E3]]]
 */
#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.

A useful trick is to let the compiler do the calculations for you and just supply the raw values. For example:

```
#define NEMA17_FULL_STEPS 200.0
#define NEMA17_MICROSTEPS 16.0
#define NEMA17_MOTOR_STEPS (NEMA17_FULL_STEPS * NEMA17_MICROSTEPS)
#define PULLEY_PITCH 2.0
#define PULLEY_TEETH 20.0
#define Z_ROD_PITCH 0.8

#define WADE_PULLEY_TEETH 11.0
#define WADE_GEAR_TEETH 45.0
#define HOBBED_BOLT_DIAM 6.0

#define XY_STEPS (NEMA17_MOTOR_STEPS / (PULLEY_PITCH * PULLEY_TEETH))
#define Z_STEPS (NEMA17_MOTOR_STEPS / Z_ROD_PITCH)
#define WADE_GEAR_RATIO (WADE_GEAR_TEETH / WADE_PULLEY_TEETH)
#define HOBBED_BOLD_CIRC (M_PI * HOBBED_BOLT_DIAM)
#define WADE_E_STEPS (NEMA17_MOTOR_STEPS * WADE_GEAR_RATIO / HOBBED_BOLD_CIRC)

#define DEFAULT_AXIS_STEPS_PER_UNIT   { XY_STEPS, XY_STEPS, Z_STEPS, ENG2_E_STEPS }
```

Step Calculator

The Prusa Calculator (http://prusaprinters.org/calculator/) is a great tool to help find the right values for your specific printer configuration.

Default Max Feed Rate

```
/**
 * Default Max Feed Rate (mm/s)
 * Override with M203
 *                                      X, Y, Z, E0 [, E1[, E2[, E3]]]
 */
#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` .

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

Acceleration

Default Max Acceleration

```
/**
 * Default Max Acceleration (change/s) change = mm/s
 * (Maximum start speed for accelerated moves)
 * Override with M201
 *                                      X, Y, Z, E0 [, E1[, E2[, E3]]]
 */
#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.

Default Acceleration

```
/**
 * Default Acceleration (change/s) change = mm/s
 * Override with M204
 *
 *   M204 P    Acceleration
 *   M204 R    Retract Acceleration
 *   M204 T    Travel 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.

> 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.

Jerk

```
/**
 * Default Jerk (mm/s)
 * Override with M205 X Y Z E
 *
 * "Jerk" specifies the minimum speed change that requires acceleration.
 * When changing speed and direction, if the difference is less than the
 * value set here, it may happen instantaneously.
 */
#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.

Z Probe Options

Probe Type

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

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.

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).

Servo Z Probe

```
//#define Z_ENDSTOP_SERVO_NR 0
//#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.

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.

See the Prusa i3 Z-probe Sled Mount (http://www.thingiverse.com/thing:396692) for an example of this kind of probe.

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 (http://reprap.org/wiki/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.

Probe Offsets

```
#define X_PROBE_OFFSET_FROM_EXTRUDER -44  // X offset: -left  [of the nozzle] +right
#define Y_PROBE_OFFSET_FROM_EXTRUDER -8  // Y offset: -front [of the nozzle] +behind
#define Z_PROBE_OFFSET_FROM_EXTRUDER -2.50   // Z offset: -below [the nozzle](for most negative! positive when using tilt probes or the nozzle
obes)
```

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`.

Probing Speed

```
// X and Y axis travel speed (mm/m) between probes
#define XY_PROBE_SPEED 4000
// Speed for the first approach when double-probing (with PROBE_DOUBLE_TOUCH)
#define Z_PROBE_SPEED_FAST HOMING_FEEDRATE_Z
// Speed 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.

Probe Double Touch

```
// Use double touch for probing
//#define PROBE_DOUBLE_TOUCH
```

Some probes may be more accurate with this option, which causes all probes to be done twice — first fast, then slow. The second result is used as the measured Z position.

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).

Probe Pins

```
//#define Z_MIN_PROBE_ENDSTOP
```

If you want to use both probe and end-switch for homing and endstop, enable this. However, This requires extra setups to be done. If you're using Ramps 1.4, the probe pins are located in D32 of the aux4 array that is also used by the lcd panel. You will have to change the pin assignments from your specified board pin file (for example "pins_RAMPS_14.h") located at `#define Z_MIN_PROBE_PIN 32`. I would change this to pin 19 (z max) since it is rarely if ever used. This extra port is actually the Z Probe that is used for your auto bed leveling.

Another way is to change between these pins: `#define Z_MIN_PROBE_PIN 32`, `#define Z_MIN_PIN 18`, and `#define Z_MAX_PIN 19` according to your board. This is not for beginners.

```
#define Z_MIN_PROBE_USES_Z_MIN_ENDSTOP_PIN
```

This uses the same pin for the end-switch and the probe. The advantage is that you don't need to alter any pin-out assignments, however you can only have ONE active at a time.

```
//#define DISABLE_Z_MIN_PROBE_ENDSTOP
```

This typically disables your probe feature. Only applicable to `//#define Z_MIN_PROBE_ENDSTOP` enabled

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.

Probe Clearance

```
#define Z_CLEARANCE_DEPLOY_PROBE   10 // Z Clearance for Deploy/Stow
#define Z_CLEARANCE_BETWEEN_PROBES  5 // Z Clearance between probe points
```

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.

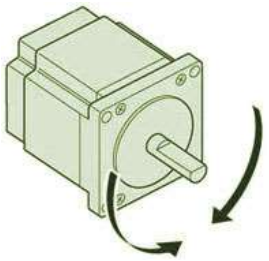
G29 Movement

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.

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 true //disable only inactive extruders and keep 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 true
#define INVERT_Y_DIR false
#define INVERT_Z_DIR true

#define INVERT_E0_DIR false
#define INVERT_E1_DIR false
#define INVERT_E2_DIR false
#define INVERT_E3_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 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.

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.

Software Endstops

```
#define min_software_endstops true
#define max_software_endstops true
```

Set to `true` to enable the option 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`.

Movement Bounds

```
#define X_MIN_POS 0
#define Y_MIN_POS 0
#define Z_MIN_POS 0
#define X_MAX_POS 200
#define Y_MAX_POS 200
#define Z_MAX_POS 170
```

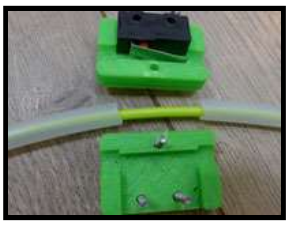
These values specify the physical limits of the machine. Usually the `[XYZ]_MIN_POS` values are set to 0, because endstops are positioned at the bed limits. `[XYZ]_MAX_POS` should be set to the farthest reachable point. By default, these positions are used for homing as well. However, the `MANUAL_[XYZ]_HOME_POS`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

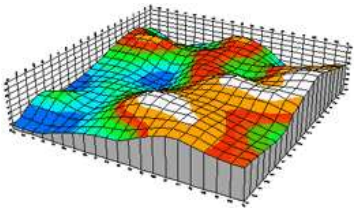
Filament Runout Sensor

```
//#define FILAMENT_RUNOUT_SENSOR
#if ENABLED(FILAMENT_RUNOUT_SENSOR)
  #define FIL_RUNOUT_INVERTING false // set to true to invert the logic of the sensor.
  #define ENDSTOPPULLUP_FIL_RUNOUT // Uncomment to use internal pullup for filament runout pins if the sensor is d
  #define FILAMENT_RUNOUT_SCRIPT "M600"
#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's 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 (/docs/gcode/G29-mbl.html) and `G29` for ABL (/docs/gcode/G29-abl.html).

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 both `AUTO_BED_LEVELING_BILINEAR` and `MESH_BED_LEVELING`, this option adds the `M420 Zn` command to set 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. This reduces computational requirements and resonance from the Z axis above the fade height.

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.

Mesh (Manual) Bed Leveling

```
//#define MESH_BED_LEVELING
```

If your machine lacks a probe, it is still possible to measure and correct for imperfections in the bed. The `MESH_BED_LEVELING` option (MBL) provides a custom `G29` command for measuring the bed height at several points using a piece of paper or feeler gauge.

With MBL enabled:

- `G29 S1` initiates bed probing. A piece of paper or feeler gauge is used to test the nozzle height while manually adjusting the Z position. See `G29` for MBL (/docs/gcode/G29-mbl.html) for the full procedure.
- `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.

> `MESH_BED_LEVELING` is not compatible with Delta and SCARA.

MBL LCD Menu

```
//#define MANUAL_BED_LEVELING
```

Enable to add a "Level Bed" menu item to the LCD that initiates a fully guided leveling procedure. See `G29` for MBL (/docs/gcode/G29-mbl.html) for more details.

Auto Bed Leveling

Auto Bed Leveling (ABL) is a standard feature on many 3D printers. It takes the guess-work out of getting a good first layer and good bed adhesion.

With ABL enabled:

- `G29` automatically probes the bed at various points, measures the bed height, calculates a correction grid or matrix, and turns on leveling compensation.
- The item "Level Bed" is added to the LCD menu.
- `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.

```
//#define AUTO_BED_LEVELING_3POINT
//#define AUTO_BED_LEVELING_LINEAR
//#define AUTO_BED_LEVELING_BILINEAR
```

Enable just one type of Auto Bed Leveling:

- Use `AUTO_BED_LEVELING_3POINT` to probe three points. The flat triangle gives a transform suitable to compensate for a flat but tilted bed.
- Use `AUTO_BED_LEVELING_LINEAR` to probe a grid. A transform is produced by least-squares method to compensate for a flat but tilted bed.
- Use `AUTO_BED_LEVELING_BILINEAR` to probe a grid. The mesh data is used to adjust Z height across the bed using bilinear interpolation.

> For Delta and SCARA, only `AUTO_BED_LEVELING_BILINEAR` is supported.

Linear / Bilinear Options

```
#define LEFT_PROBE_BED_POSITION 15
#define RIGHT_PROBE_BED_POSITION 145
#define FRONT_PROBE_BED_POSITION 20
#define BACK_PROBE_BED_POSITION 150
```

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 ABL_GRID_MAX_POINTS_X 3
#define ABL_GRID_MAX_POINTS_Y ABL_GRID_MAX_POINTS_X
```

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

```
//#define PROBE_Y_FIRST
```

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

3-Point Options

```
#define ABL_PROBE_PT_1_X 15
#define ABL_PROBE_PT_1_Y 180
#define ABL_PROBE_PT_2_X 15
#define ABL_PROBE_PT_2_Y 20
#define ABL_PROBE_PT_3_X 170
#define ABL_PROBE_PT_3_Y 20
```

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

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_MIN_POS + X_MAX_POS) / 2)    // X point for Z homing when homing all axis (G28).
  #define Z_SAFE_HOMING_Y_POINT ((Y_MIN_POS + Y_MAX_POS) / 2)    // Y point for Z homing when homing all axis (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.

Extras 1

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)

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

> Certain EEPROM behaviors may be confusing. For example, when you edit the configurations and re-flash the firmware, you may discover that your new settings don't have any effect! What's going on? They are still overridden by the settings last saved to EEPROM! To apply and preserve your new settings, use `M502` to restore settings to your defaults, then use `M500` to save them to EEPROM. You can always use `M503` to view the current settings in volatile memory (even without `EEPROM_SETTINGS` enabled).

Host Keepalive

```
#define HOST_KEEPALIVE_FEATURE
#define DEFAULT_KEEPALIVE_INTERVAL 2
```

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` (M113.html).

Free Memory Watcher

```
//#define M100_FREE_MEMORY_WATCHER
```

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

Inch Units Support

```
//#define INCH_MODE_SUPPORT
```

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

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.

LCD Language

User Interface Language

```
#define LCD_LANGUAGE en
```

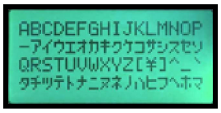
Choose your preferred language for the LCD controller here. Supported languages include:

Code	Language		Code	Language		Code	Language
en	English (Default)		an	Aragonese		bg	Bulgarian
ca	Catalan		cn	Chinese		cz	Czech
de	German		el	Greek		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)
nl	Dutch		pl	Polish		pt	Portuguese
pt-br	Portuguese (Brazilian)		pt-	Portuguese (Brazilian UTF8)		pt_utf8	Portuguese (UTF8)
ru	Russian		tr	Turkish		uk	Ukrainian

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 https://github.com/MarlinFirmware/Marlin/wiki/LCD-Language

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

- `ULTIMAKERCONTROLLER` : The original Ultimaker Controller.
- `ULTIPANEL` : ULTIPANEL (http://www.thingiverse.com/thing:15081) as seen on Thingiverse.
- `PANEL_ONE` : PanelOne from T3P3 (http://reprap.org/wiki/PanelOne) (via RAMPS 1.4 AUX2/AUX3). A variant of `ULTIMAKERCONTROLLER`.
- `REPRAP_DISCOUNT_SMART_CONTROLLER` : RepRapDiscount Smart Controller (http://reprap.org/wiki/RepRapDiscount_Smart_Controller). Usually sold with a white PCB.
- `G3D_PANEL` : Gadgets3D G3D LCD/SD Controller (http://reprap.org/wiki/RAMPS_1.3/1.4_GADGETS3D_Shield_with_Panel). Usually sold with a blue PCB.
- `RIGIDBOT_PANEL` : RigidBot Panel V1.0 (http://www.inventapart.com/).

Graphical LCDs

- `CARTESIO_UI` : Cartesio UI (http://mauk.cc/webshop/cartesio-shop/electronics/user-interface).
- `MAKRPANEL` : MaKr3d Makr-Panel (http://reprap.org/wiki/MaKr3d_MaKrPanel) with graphic controller and SD support.
- `REPRAPWORLD_GRAPHICAL_LCD` : ReprapWorld Graphical LCD (https://reprapworld.com/?products_details&products_id/1218).
- `VIKI2` : Panucatt Devices (http://panucatt.com) Viki 2.0 (http://panucatt.com).
- `miniVIKI` : mini Viki with Graphic LCD (http://panucatt.com).
- `ELB_FULL_GRAPHIC_CONTROLLER` : Adafruit ST7565 Full Graphic Controller (https://github.com/eboston/Adafruit-ST7565-Full-Graphic-Controller/).
- `REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER` : RepRapDiscount Full Graphic Smart Controller (http://reprap.org/wiki/RepRapDiscount_Full_Graphic_Smart_Controller).
- `MINIPANEL` : MakerLab Mini Panel (http://reprap.org/wiki/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.

Keypad

- `REPRAPWORLD_KEYPAD` : RepRapWorld Keypad v1.1 (http://reprapworld.com/?products_details&products_id=202&cPath=1591_1626) 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 (https://github.com/kiyoshigawa/LiquidCrystal_I2C).

- `RA_CONTROL_PANEL` : Elefu RA Board Control Panel (http://www.elefu.com/index.php?route=product/product&product_id=53)
- `LCD_I2C_SAINSMART_YWROBOT` : Sainsmart YWRobot LCM1602 LCD Display (http://henrysbench.capnfatz.com/henrys-bench/arduino-displays/ywrobot-lcm1602-iic-v1-lcd-arduino-tutorial/).
- `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 (https://github.com/lincomatic/LiquidTWI2) v1.2.3 or later.
- `LCD_I2C_VIKI` : Panucatt VIKI LCD with status LEDs, integrated click & L/R/U/D buttons, separate encoder inputs.
- `SAV_3DLCD` : Shift register panels. 2 wire Non-latching LCD SR (https://goo.gl/aJJ4sH). See LCD configuration (http://reprap.org/wiki/SAV_3D_LCD).

I2C Graphical LCDs

These controllers all require the LiquidCrystal_I2C library (https://github.com/kiyoshigawa/LiquidCrystal_I2C).

- `U8GLIB_SSD1306` : SSD1306 OLED full graphics generic display.
- `SAV_3DGLCD` : SAV OLED LCD module support using either SSD1306 or SH1106 based LCD modules.

Extras 2

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.

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 (http://www.doc-diy.net/photo/rc-1_hacked/).

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.

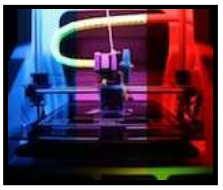
Extras 3

Paste Extruder

```
// Support for the BariCUDA Paste Extruder.
//#define BARICUDA
```

Marlin includes support for the Baricuda Extruder for 3D Printing Sugar and Chocolate (http://www.thingiverse.com/thing:26343) also hosted on GitHub (http://www.github.com/jmil/BariCUDA). The feature adds the codes `M126`, `M127`, `M128`, and `M129` for controlling the pump and valve of the Baricuda.

(http://www.instructables.com/id/3D-Printer-RGB-LED-Feedback/)



Indicator 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.

```
// Support for an RGB LED using 3 separate pins with optional PWM
//#define RGB_LED
#if ENABLED(RGB_LED)
  #define RGB_LED_R_PIN 34
  #define RGB_LED_G_PIN 43
  #define RGB_LED_B_PIN 35
#endif
```

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.

Servos

Number of Servos


```
#define NUM_SERVOS 1 // 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.

Filament Width Sensor

```
//#define FILAMENT_WIDTH_SENSOR
```

Filament sensor

Enable to add support for a filament width sensor such as Filament Width Sensor Prototype Version 3 (http://www.thingiverse.com/thing:454584). 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.

```
#define DEFAULT_NOMINAL_FILAMENT_DIA 3.00
```

The "nominal" filament diameter as written on the filament spool. If you typically use 1.75mm filament, but physically measure the diameter as 1.70mm, you should still use 1.75. Marlin will compensate automatically. The same goes for 3.00mm filament that measures closer to 2.85mm.

```
#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 middle of the filament path (i.e., the nozzle opening).

```
#define MEASURED_UPPER_LIMIT         3.30  //upper limit factor used for sensor reading validation in mm
#define MEASURED_LOWER_LIMIT         1.90  //lower limit factor for sensor reading validation in mm
```

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.

Back to top ⬆