Marlin

From RepRapWiki

Marlin is a firmware for RepRap single-processor electronics, supporting RAMPS, RAMBo, Ultimaker, BQ, and several other Arduino-based 3D printers. It supports printing over USB or from SD cards with folders, and uses lookahead trajectory planning. Marlin is licensed under the GNU GPL v3 or later. It is based on Sprinter firmware, licensed under GPL v2 or later. The most active developers of Marlin are currently (January 2016) thinkyhead, AnHardt, ErikZalm, daid, boelle, Wackerbarth, bkubicek, and Wurstnase, with many others contributing patches.

The Marlin Project (https://github.com/MarlinFirmware/Marlin) is hosted on Github.

Current Release: Marlin 1.0.2 (https://github.com/MarlinFirmware/Marlin/tags) :

Works on RAMPS 1.4, Rumba, Ultimaker, Sanguinololu, Generation_6_Electronics, and many other controllers based on AVR 8-bit MCUs. See the file "boards.h" in the source code for a complete list.

Sprinter

Release status: working

Description	a firmware for Arduino
License	GNU GPL v3
Author	EvdZ
Contributors	
Based-on	Sprinter
Categories	Marlin, Firmware, Firm
CAD Models	none
External Link	GitHub (https://github.com/Ma

Contents

Quick Information

This RepRap firmware is a mashup between Sprinter and Grbl with many original parts.

Derived from Sprinter and Grbl by Erik van der Zalm. Sprinter's lead developers are Kliment and caru. Grbl's lead developer is Simen Svale Skogsrud. Sonney Jeon (Chamnit) improved some parts of Grbl. A fork by bkubicek for the Ultimaker was merged, and further development was aided by him. Some features have been added by: Lampmaker, Bradley Feldman, and others...

Features:

- Interrupt based movement with real linear acceleration.
- High steprate.
- Look ahead (Keep the speed high when possible. High cornering speed).
- Interrupt based temperature protection.
- Preliminary support for Matthew Roberts' advance algorithm (http://reprap.org/pipermail/reprap-dev/2011-May/003323.html) .
- Full endstop support.
- SD Card support, including folders and long filenames.
- LCD support, both character-based and graphical. (Ideally 20x4 or 128x64).
- LCD menu system for standalone SD card printing, controlled by a click-encoder.
- EEPROM storage of several settings.
- many small but handy things originating from bkubicek's fork.
- Arc support.
- Temperature oversampling.
- Dynamic Temperature setpointing aka "AutoTemp."
- Endstop trigger reporting to the host software.
- Heater power reporting. Useful for PID monitoring.
- CoreXY and CoreXZ (http://www.corexy.com/theory.html) kinematics.
- Delta kinematics.
- SCARA kinematics.
- Automatic bed leveling and compensation.
- Firmware binary size between around 50kB and 100kB, depending on options.
- Filament Width Sensor support
- Filament Runout Sensor support
- Multiple Extruders (up to 4) supported
-

The default baudrate is 250000. Because this baudrate is directly derived from the usual 16MHz clock of the Arduino MCU, it has less jitter and hence fewer errors than the usual 115200 baud, but 250000 baud is not as well supported by drivers and host-environments.

Differences from and additions to the already good Sprinter firmware

Look-ahead

Marlin has jerk-type look-ahead. Whithout it, it would brake to a stop and re-accelerate at each corner. Lookahead will only decelerate and accelerate to some non-zero velocity, so that the change in vectorial velocity magnitude is less than the `max_xy_jerk`. This is only possible, if some future moves are already processed, hence the name look-ahead. It leads to less over-deposition of material at corners, especially at flat angles.

Arc support

Slic3r can find curves that, although broken into segments, were meant to describe an arc. Marlin is able to print those arcs. The advantage is that the firmware can choose the resolution, and can perform the arc with nearly constant velocity, resulting in a nice finish. Also, less serial communication is needed.

Temperature Oversampling

To reduce noise and make the PID-differential term more useful, 16 ADC conversion results are averaged.

AutoTemp

If your gcode contains a wide spread of extruder velocities, or you realtime change the building speed, the temperature should be changed accordingly. Usually, higher speed requires higher temperature. This can now be performed by the AutoTemp function. Enable AutoTemp mode with `M109 S T F` and disable it with `M109` (without any F value). When active, the maximal extruder stepper rate of all buffered moves is calculated and named "maxerate" [steps/sec]. The desired temperature is then set to t=tempmin+factor*maxerate, constrained between tempmin and tempmax. If the target temperature is set manually or by GCode to a value less then tempmin, it will be kept without change. Ideally, your GCode can be completely free of temperature controls, apart from a `M109 S T F` in the start.gcode, and a `M109 S0` in the end.gcode.

EEPROM

If you have established known working PID constants, acceleration, and max-velocity settings for your own machine, you can set them, then store them in the EEPROM. On each boot-up, Marlin will automatically load these values from EEPROM, independent of what your compiled Configuration.h says.

LCD Menu

If your hardware supports it, you can build a LCD-CardReader+Click+encoder combination. This will allow you to adjust temperatures, accelerations, velocities, and flow rates in realtime (while printing). It also provides the ability to select and print files directly from the SD card, preheat the extruder, disable the stepper motors, and do other interesting things. One working hardware configuration is documented here: http://www.thingiverse.com/thing:12663 . If you have at least a 20x4 or 16x2 display, useful data is shown.

SD card folders

If you have an SD card reader attached to your controller, folders are supported to a depth of 10 levels. Listing files in Pronterface shows "/path/subpath/file.g". You can write to files in subfolders by including the path (in lowercase). (Hidden files, backup files, and non-GCode files are not included in file listings.)

Endstop trigger reporting

If an endstop is hit while moving towards the endstop, the location at which the firmware thinks the endstop was triggered is output to the serial port. This is useful because the user gets a warning message. Tools like QTMarlin can use this to find acceptable combinations of velocity+acceleration.

Coding paradigm

Not relevant from the user perspective, but Marlin is split into thematic chunks, and has tried to partially enforce private variables. This is intended to make interactions between modules clearer, and leads to a higher level of encapsulation. We think this will be useful as a preliminary step for porting to other platforms, such as ARM. Lots of RAM (with enabled LCD ~2200 bytes) was saved by storing character strings in Program Memory. In the serial communication, a #define-based level of abstraction was enforced, so transfer of information is clear (usually beginning with "echo:"), an error "error:", or just normal protocol, necessary for backwards compatibility.

Interrupt based temperature measurements

An interrupt is used to manage ADC conversions, and enforce checking for critical temperatures. This leads to less blocking in the heater management routine.

Delta compatible

TODO...

CoreXY compatible

Can use a CoreXY (http://corexy.com/theory.html) table.

Non-standard M-Codes, different from an old version of Sprinter:

Movement:

- G2 - CW ARC
- G3 - CCW ARC
- G10 - Retract according to settings.
- G11 - Un-retract (recover) according to settings.
- G29 - Automatic bed probing and compensation.
- G30 - Probe Z height at the current location.

General:

- M17 - Enable/Power all stepper motors. Compatibility with ReplicatorG.
- M18 - Disable all stepper motors; same as M84. Compatibility with ReplicatorG.

- M31 - Print the time since last M109 or SD print start to the serial out.
- M42 - Change a single pin's status.
- M80 - Turn on Power Supply.
- M81 - Turn off Power Supply.
- M114 - Output current position to the serial output.
- M119 - Output Endstop status to the serial output.

Movement variables:

- M202 - UNUSED IN MARLIN. Set max acceleration in units/s^2 for travel moves (M202 X1000 Y1000)
- M203 - Set maximum feedrate that your machine can sustain (M203 X200 Y200 Z300 E10000) in mm/sec.
- M204 - Set default acceleration: S for print and retract, R for retract moves, T for travel moves (M204 S3000 T7000) in mm/sec^2.
- M220 - Set the feedrate multiplier in percent. (M220 S95)
- M301 - Set PID parameters P, I, and D. If PID_ADD_EXTRUSION_RATE is enabled, optionally set C (Kc term) and L (Last-Position-Queue length).
- M303 - PID autotune. S = target temperature, E = extruder number or -1 for bed, C = cycles.
- M400 - Finish all buffered moves before processing the next command.

Advance:

- M200 - Set the filament diameter for volumetric extrusion.
- M205 - Advanced settings. S[min feedrate] T[min travel feedrate] B[min segment time] X[max XY jerk] Z[max Z jerk] E[max E jerk].

EEPROM:

- M500 - Store settings in EEPROM.
- M501 - Restore settings from EEPROM.
- M502 - Revert to default "factory settings." (Use M500 if you want to store them to EEPROM.)
- M503 - Print the current settings (as set in memory, not the settings stored in EEPROM.)

Configuring and compilation

What you need to know before starting:

- What kind of printer you are using. If you are using a Cartesian printer (like the Prusa i3), you are going to need to calculate the steps/mm for each axis and for the extruder. To figure this out, you can go to http://prusaprinters.org/calculator/ or Triffid_Hunter's_Calibration_Guide. Write down the values you get for everything.
  - You need to know what kind of drive is being used with your printer. If it's a belt, you need to know what kind. If it's a screw, need to know what kind.
  - The Prusa i3 uses belts for X and Y, and screws for Z.
- A host software like Printrun or Repetier or even Octoprint.

1. Install the arduino software IDE/toolset, version 1.0.5 or 1.0.6 from http://www.arduino.cc/en/Main/Software
2. Download the Marlin firmware from https://github.com/MarlinFirmware/Marlin (see the Download Zip button), or use git to clone it (if you know how to use git).
3. Extract the firmware to a directory of your choice.
4. Start the arduino IDE. Select Tools -> Board -> Arduino Mega 2560 (or whatever your microcontroller is)
5. Select the correct serial port in Tools ->Serial Port, usually there is only one option.
6. Open Marlin.ino in /path/to/Marlin/Marlin
7. Browse to boards.h
    1. This is a list of motherboard types. You'll need to figure out which one you need and write down the word after #define on that line.
        - Example: If you're using a RAMPS 1.4 with a extruder, fan (optional) and heated bed, then you'd need RAMPS_13_EFB.
8. Browse to Configuration.h
    1. Write down the value given in #define BAUDRATE for later.
    2. At the line that says #define MOTHERBOARD, replace whatever follows MOTHERBOARD with what you chose earlier.
    3. Set the #define EXTRUDERS to the number of extruders that you have.
    4. For the TEMP_SENSOR lines, you have to know what kind of thermistor is used by your hot end and heated bed (if you have one). Set the values for these to ones that match the list directly above the lines.
        - If you have an extruder on E0 and a heated bed, you just need to set TEMP_SENSOR_0 and

TEMP_SENSOR_BED.

5. Scroll down to MAX_HEATER_TEMP. If you know that your hotend shouldn't go above a certain value (the Budaschnozzle should not exceed 240C-242C), then change it here. Same for MAX_BED_TEMP.
6. Leave the PIDTEMP stuff alone for now, we'll get back to it later.
7. Further down the page, look for the Mechanical Settings area.
    - For the INVERT_X_DIR and related lines, set it for Mendel if you are using a Mendel-type printer. Otherwise, consult the documentation for your printer design.
        - The Prusa i3 is a Mendel printer.
    - For X_HOME_DIR and the similar commands for Y and Z, look where your endstops are. If a endstop is configured to be at the 0 position for that axis, the setting here needs to be -1. Otherwise, it needs to be 1.
    - The X_MIN_POS, X_MAX_POS, and related entries should correspond to the printable area on your bed. The defaults are common, but if you have a bigger/smaller print area, you will need to change this.
        - Please note that 0,0 should be the cartesian "bottom left" of your print-bed, if your axis homes beyond the bed in any direction, you can use a *negative value* for the X_MIN_POS and Y_MIN_POS to compensate. (If your prints never wind up in the center of the bed, this is the culprit). Also See Configuring Marlin Bed Dimensions.
    - Ignore the auto bed leveling area for now. Hopefully you will never need it.
8. Time for the Feed rates and Steps/mm! You're almost done.
    - For #define DEFAULT_AXIS_STEPS_PER_UNIT, you will need to use the values you calculated for your different printer axes and extruder (see above). The order is {X, Y, Z, E}
    - For DEFAULT_MAX_FEEDRATE, this is the fastest (in mm/s) the printer is allowed to go.
        - For a Prusa i3, setting the Z axis value to 2 is a good idea.
    - DEFAULT_MAX_ACCELERATION is something else that should be tuned based on how good you built the hardware of your printer.
    - You should probably turn DEFAULT_ACCELERATION to a lower value initially (like 200) and adjust the acceleration later once you get everything calibrated and tuned.
9. If you have an LCD or button panel for your printer, uncomment the respective lines for it. Otherwise, save because you are done!

9. Click the Upload button.
    - If all goes well, the firmware has uploaded.

You are now done with your initial configuration of Marlin! It should give you reasonable results, but there's still a lot of setup to do. Head over to Triffid_Hunter's_Calibration_Guide to continue tweaking your printer.

If you ever need to change one of these firmware settings, open the Marlin project with the Arduino IDE, make your changes to Configuration.h, and re-upload (make sure your host software isn't connected when you try to upload).

Bug reports

KNOWN ISSUES: RepG will display: Unknown: marlin x.y.z

For bug reporting please use the Issue tracker on github (https://github.com/MarlinFirmware/Marlin/issues)

Categories: Working developments | Firmware | Firmware development | Marlin

---