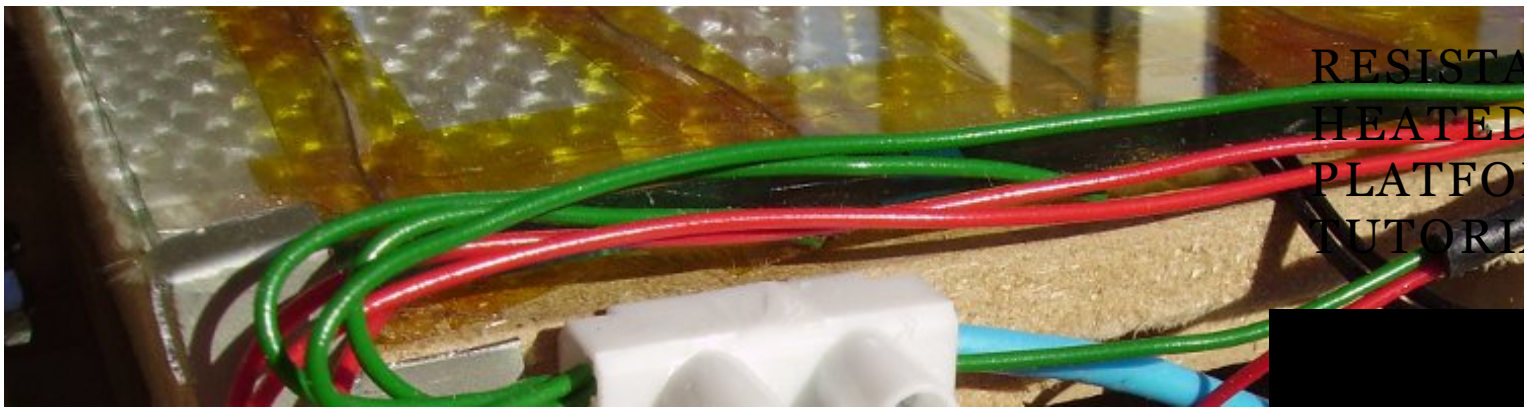
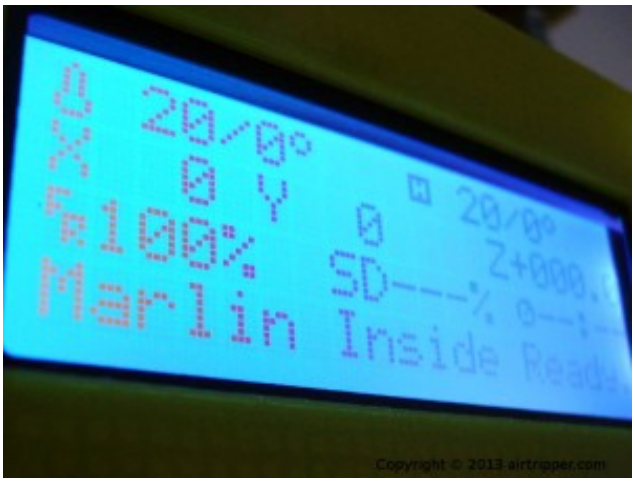
AIRTRIPPER'S 3D PRINTER AND ARDUINO BLOG

3D PRINTER USAGE AND MODIFICATIONS PLUS ARDUINO POWERED ELECTRONIC PROJECTS AND 3D PRINTIN DESIGNS.



Marlin Firmware v1, Basic Configuration Set-up Guide



MARLIN FIRMWARE V1 ON 20×4 LCD PANEL DISPLAY

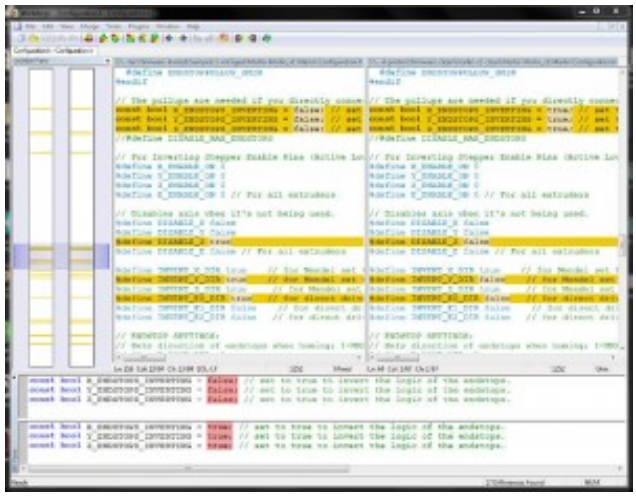
I've just updated the Marlin firmware on my Sumpod 3d printer since I'm always keen to have the latest features and bug fixes. To be honest, I don't mess with the firmware that much, and if it wasn't for the configuration file from my last version, I would struggle to remember what sort of configuration I would need to set in the latest firmware version.

Anyway, while my latest experience with 3d printer firmware is still fresh in the mined, I'll share some notes about what settings you need to know to get a basic Marlin firmware configured enough to get a 3d printer working. The notes will focus on the Marlin firmware v1 and will include setting up a

click encoder and LCD panel. But before going straight into getting the Marlin firmware configured, I'll first quickly introduce you to a handy tool called WinMerge.

WINMERGE

For anybody that's in the business of editing and configuring 3d printer firmware files such as the Marlin firmware, I would suggest downloading a copy of WinMerge. It's free, open source software, and is cross platform, so the same tool will run on Windows and Linux.



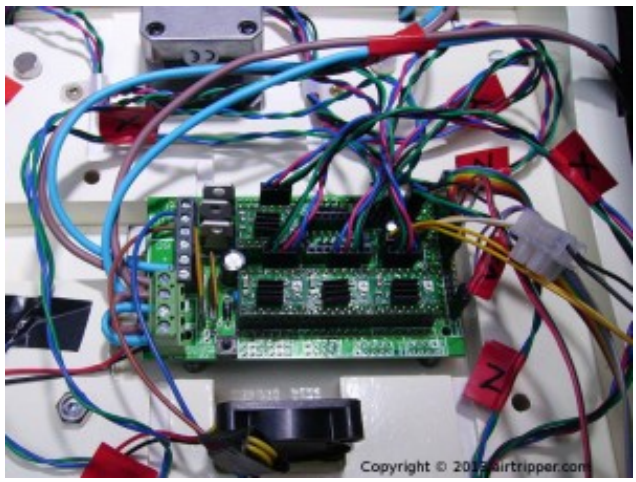
WINMERGE CAN COMPARE BOTH FOLDERS AND FILES – MARLIN FIRMWARE BEING COMPARED

You can use WinMerge to compare a clean version of your Marlin firmware against your edited version that you are using on your 3d printer. This will help to keep track and note all the changes made to the files that you might want to transfer to a newer firmware version.

You can open just two files to compare or you can open two folders to compare. Comparing 3d printer firmware folders will allow you to quickly spot which files that have been edited.

Marlin Firmware Basic Configuration

Some motherboards listed in the Marlin firmware configuration file may not have support for some of the options or features available to configure. The notes will be biased towards the Ramps 1.3 board, but the note will still be valid for any Marlin firmware compatible board.

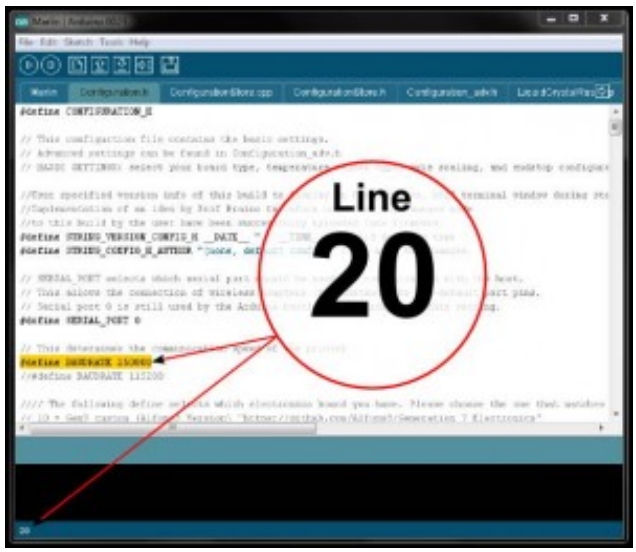


RAMPS 1.3 ON THE UNDERSIDE OF THE SUMPOD 3D PRINTER

This Marlin firmware will be useful to those who wish to configure their own 3d printer firmware and want information that expands on the comments already made in the Marlin firmware. The notes

are a brief guide on what the settings are
and how to use them, leaving it up to the person who is configuring the 3d printer to decide what the final settings should be. I don't guarantee that the information in this Marlin firmware guide is accurate, however, if you spot any mistakes please leave a comment at the end of the post. The Marlin firmware guide is likely to be updated to improve the information where necessary after publishing.

For Marlin Firmware V1 you will need Arduino 0023 IDE to save, compile and upload to the 3d printer motherboard. In the following notes, to edit the firmware, I'll be using line numbers to reference the location of the code in the Marlin firmware configuration file. Although there is no line numbering in the IDE editor window, you will see the line number at the bottom left of the IDE showing the current cursor position. Just move the cursor to any line with the mouse to update the line number.



ARDUINO 0023 IDE – SHOWING LINE NUMBER LOCATION

To get the 3d printer up and running you only need to configure one file in the Marlin firmware and that file is the ***Configuration.h***

MARLIN FIRMWARE CONFIGURATION

So, to get started, open up the Marlin firmware Configuration.h in the Arduino IDE and work down the notes below. Use WinMerge to compare the changes to a clean non-edited version of Configuration.h for final review before uploading to the 3d printer motherboard.

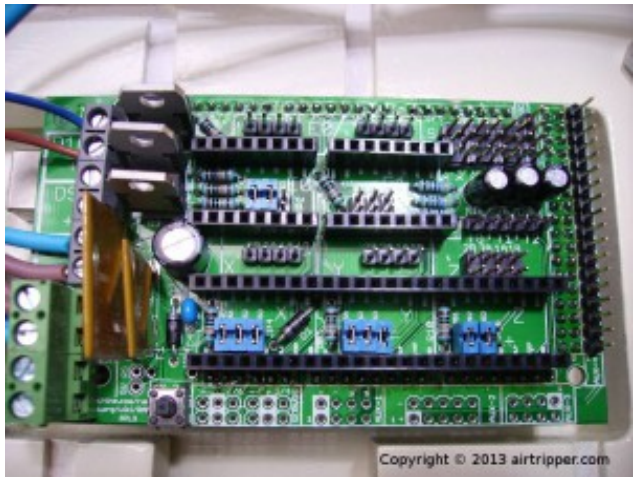
Baud Rate – line 20

#define BAUDRATE 250000
//#define BAUDRATE 115200

Line 20 marks the start of the Marlin firmware configuration journey and this is where the baud rate is set to determine the speed of the communication between the printer interface software and the Marlin firmware. Two common baud rate options are defined with one, preceded with two forward slashes (//), commented out to disable. The default enabled option, 250000 baud rate, works well when used with Printrun/Pronterface and RepSnapper software.

Motherboard – line 49

#ifndef MOTHERBOARD
#define MOTHERBOARD 33
#endif



REPRAP ARDUINO MEGA POLOLU SHIELD, OR RAMPS FOR SHORT

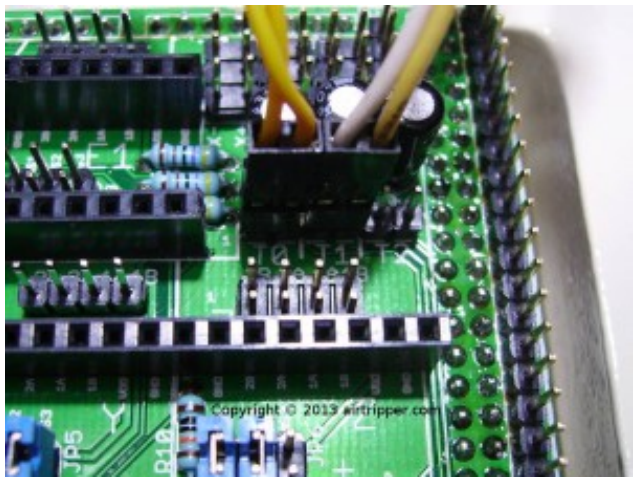
You will see a list of motherboards to choose from preceding the code snippet shown above in the Marlin firmware configuration file. The code snippet above defines the Ramps 1.3 as the motherboard, you can select a board just by changing the number to any board that's on the list. The Marlin firmware will manage circuit board pin assignments to match the motherboard you have selected. Pin assignment details for each motherboard type can be found in the pins.h file of the Marlin firmware.

MARLIN FIRMWARE THERMAL SETTINGS

Now we come to the thermal settings section of the Marlin firmware where things get a bit more complicated, however, we don't need to touch the complicated stuff to get the printer up and running. Basically all we need to do is select a temperature sensor type for each of the sensors installed on the 3d printer. If you did not install the temperature sensors yourself, you may have to investigate what sensors you do have so that you can select the best match from the list. More about thermistors on the RepRap wiki.

Thermistor – line 78

#define TEMP_SENSOR_0 1
#define TEMP_SENSOR_1 0
#define TEMP_SENSOR_2 0
#define TEMP_SENSOR_BED 1



RAMPS 1.3 TEMPERATURE CONNECTORS WITH T0 & T1 CONNECTED TO HOT END & HEATED

If you are looking at the Marlin firmware configuration file you will see a list of temperature sensor types preceding the code snippet shown above. The code snippet above is setting up the 3d printer that features one hot end and a heated

build platform. The temperature sensor selected for both features is 100k thermistor.

BED SENSORS

If you are using the Ramps 1.3 motherboard with the default pin.h file in the Marlin firmware, the motherboard connectors T0 and T1 will be enabled for the hot end sensor and the heated bed sensor. Changing the 1 to a 0 will disable that sensor. Change the number to select the best match for your sensor from the list.

Maximum Temperatures – line 99

#define HEATER_0_MAXTEMP 275
#define HEATER_1_MAXTEMP 275
#define HEATER_2_MAXTEMP 275
#define BED_MAXTEMP 120

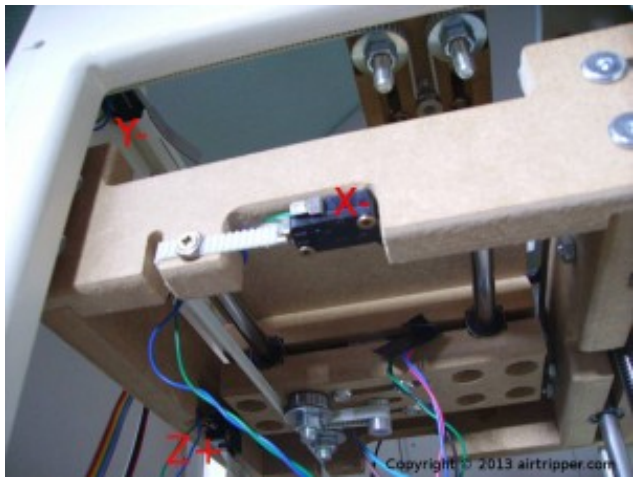
Some hot ends and heated build platforms might have a maximum temperature rating much less than the default settings in the Marlin firmware, reducing the default maximum temperatures will help avoid accidental damage to the 3d printer if set too high in the interface software.

MARLIN FIRMWARE MECHANICAL SETTINGS

The Marlin firmware Mechanical Settings section will be about configuring End Stops, Stepper Motors, Build Platform Printable area and Steps Per Unit.

End Stops & Pull Ups

This section will be about configuring end stops in the Marlin firmware that are the limit switches for each axis on the 3d printer. Issuing a homing command from the interface software will cause the 3d printer to mechanically move each axis towards the end stop until the limit switch is triggered. ENDSTOPPULLUPS will need to be defined where you have limit switches that don't supply a voltage to the signal pin to generate a digital 1. Enabling pull up resistors will ensure that the end stop signal line will read a digital



SUMPOD 3D PRINTER END STOP LOCATIONS

1, and when the signal line is shorted to ground by the limit switch, you get a digital 0.

For more information about end stops please refer to the RepRap wiki for Mechanical Endstop, OptoEndstop 2.1 and Gen7 Endstop 1.3.1.

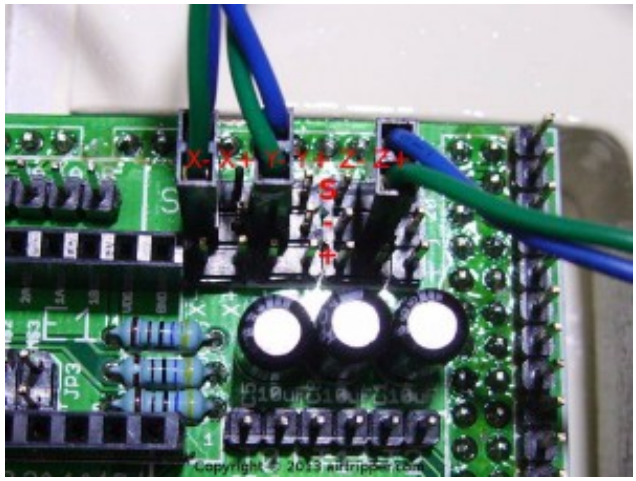
Coarse End Stop Pull Up Resistor – line 194

// corse Endstop Settings
#define ENDSTOPPULLUPS // Comment this out (using // at the start of the line) to disable the endstop pullup resistors

In the Marlin firmware ENDSTOPPULLUPS is defined by default, and ENDSTOPPULLUPS for each end stop connector on the motherboard are enabled individually from line 207. However, commenting out line 194 will only disable ENDSTOPPULLUPS that are also commented out optionally for each connector from line 197. Having this kind of fine tuning makes it easier to configure different types of end stops connected to the 3d printer. You may have mechanical end stop switches for axis X and Y that need pull up resistors enabled and optical end stops that don't need pull up resistors enabled.

Fine End Stop Pull Up Resistor – line 196

#ifndef ENDSTOPPULLUPS
// fine Enstop settings: Individual Pullups. will be ignord if ENDSTOPPULLUPS is defined
#define ENDSTOPPULLUP_XMAX
#define ENDSTOPPULLUP_YMAX
#define ENDSTOPPULLUP_ZMAX
#define ENDSTOPPULLUP_XMIN
#define ENDSTOPPULLUP_YMIN
//#define ENDSTOPPULLUP_ZMIN
#endif

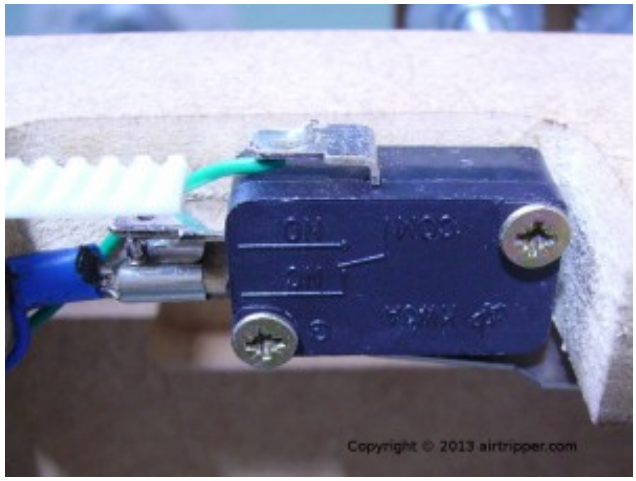


RAMPS 1.3 MECHANICAL END STOP CONNECTIONS X-, Y- & Z+

If ENDSTOPPULLUPS at line 194 in the Configuration.h file of the Marlin firmware is commented out, then the code snippet above, starting at line 196, will execute. This code snippet will allow you to configure individual pull ups for each end stop connector on the motherboard. You would comment out a define statement for an end stop where you don't need a pull up by preceding the line with two forward slashes. You would normally need to enable pull ups when using mechanical end stop type switches that just simply sink the signal pin to ground on the motherboard end stop connector.

Inverting End Stops – line 216

// The pullups are needed if you directly connect a mechanical endswitch between the signal and ground pins.
// set to true to invert the logic of the endstops.
const bool X_ENDSTOPS_INVERTING = false;
const bool Y_ENDSTOPS_INVERTING = false;
const bool Z_ENDSTOPS_INVERTING = false;



3D PRINTER MECHANICAL END STOP WIRED NORMALLY CONNECTED (NC)

For a mechanical end stop that is wired as normally connected (NC), a digital signal of 1 will be read when an axis is in contact with the limit switch. In this case End stop inverting needs to be false. End stop inverting needs to be true if the mechanical end stop is wired as normally open (NO).

For optical end stops, you will need to refer to the suppliers specification or a previous configuration.h file like the one in the Marlin firmware.

You should test the homing command while the axis is positioned at the centre of the travel distance. If you find that the axis won't move when the home command is sent, then you may have the logic incorrectly configured. Be ready to reset the motherboard or turn off the power to avoid axis crash. Before making any test, complete the Marlin firmware configuration as much as possible. If you can reach the end stops easily, you can trigger them early before the axis has completed it's travel for a safe test.

Disable Maximum End Stops – line 219

//#define DISABLE_MAX_ENDSTOPS

The above code snippet found on line 219 of the Marlin firmware Configuration.h file is commented out by default, which allows homing axis to end stops where end stop switches are connected to the X+, Y+ and Z+ connectors on the motherboard.

It is common to have the Z axis end stop switch connected to the Z+ connector in order to home the Z axis away from the hot end. In order to do this, line 219 needs to be commented out. Line 219 may behave differently for boards that don't have X+, Y+ & Z+ end stop connectors.

Disable Axis – line 228

// Disables axis when it's not being used.
#define DISABLE_X false
#define DISABLE_Y false
#define DISABLE_Z true
#define DISABLE_E false // For all extruders

DISABLE Z AXIS = TRUE IN THE MARLIN FIRMWARE CONFIGURATION IF YOU WANT TO OPERATE A Z AXIS HANDLE DURING 3D PRINTING

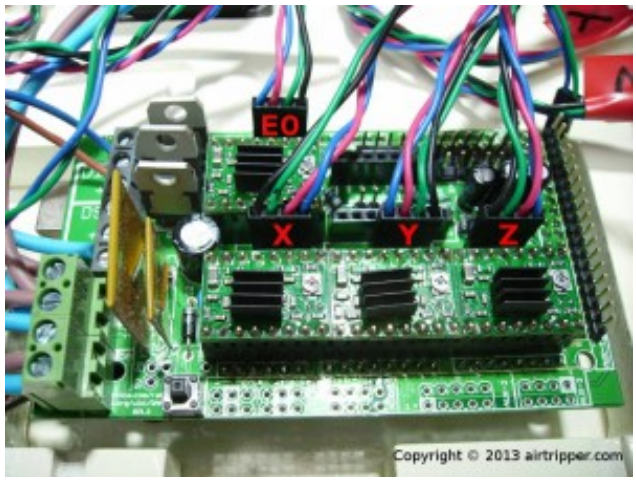
Normally the above code snippet would not be changed in the Marlin firmware and all the settings would be set to false by default. However, if your 3d printer has a Z axis handle fitted like my 3d printer you might want to disable the Z axis so that the stepper motor can be turned by the Z axis handle while the 3d printer is printing. I've often made a Z height adjustment to fine tune the gap between the nozzle and the build bed as the first layer begins to print.

Stepper Motors

We have reached the part in the Marlin firmware configuration file where you configure stepper motor rotation direction, end stop direction, travel limits and steps per unit. As long as the end stops are configured correctly, the following settings should be easy enough to sort out during testing.

Stepper Motor Rotation direction – line 233

#define INVERT_X_DIR true
#define INVERT_Y_DIR true
#define INVERT_Z_DIR true
#define INVERT_E0_DIR true
#define INVERT_E1_DIR false
#define INVERT_E2_DIR false



RAMPS 1.3 STEPPER MOTOR WIRE CONNECTIONS

This is where we decide which direction each axis will go when we control the 3d printer through the interface software. When we send a command to move +10mm on an axis, we expect the axis to

move 10mm in the direction expected. The initial stepper motor direction can be difficult to predict without switching on the printer and performing a test. So I would suggest leaving these settings till last and complete the rest of the Marlin firmware configuration before proceeding with the test.

Once the Marlin firmware Configuration file is configured enough to operate the 3d printer, you can perform a test to check that each axis move in the correct direction. Set each axis midpoint of their full travel distance and then switch on the printer. After connecting to the 3d printer through the interface software such as Printrun/pronterface, test each axis by jogging them 10mm in the positive direction. The stepper motor rotation direction for each axis can be corrected from line 233 in the Marlin firmware configuration by changing the logic.

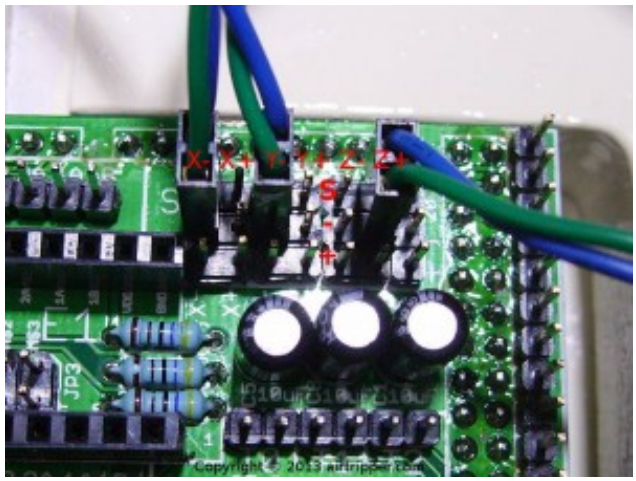
You would need to test the extruder stepper motor direction as well. This can be done without filament loaded and bringing the hot end temperature up to 175 degrees so the Marlin firmware will allow extrusion. Send an extrude command through the 3d printer interface software and observe the direction the filament drive gear pulley rotation. The rotation direction for the extruder can be corrected from line 236 by changing the logic.

When performing axis homing and axis jogging for the first time, it is important to be ready to press the reset button or be ready to turn off the power to avoid axis crashing. Home each axis separately to confirm that the axis is homing towards the end stop. If you can reach the end stops easily, you can trigger them early before the axis has completed it's travel for a safe test.

End Stop Home Direction – line 242

// Sets direction of endstops when homing; 1=MAX, -1=MIN
#define X_HOME_DIR -1
#define Y_HOME_DIR -1
#define Z_HOME_DIR 1

Basically you tell the Marlin firmware which end of the axis the end stop switch is located. It is common for X and Y axis to home the hot end to the Zero location and Z axis end stop home to the maximum positive location. The code snippet above and the image to the right shows that configuration.



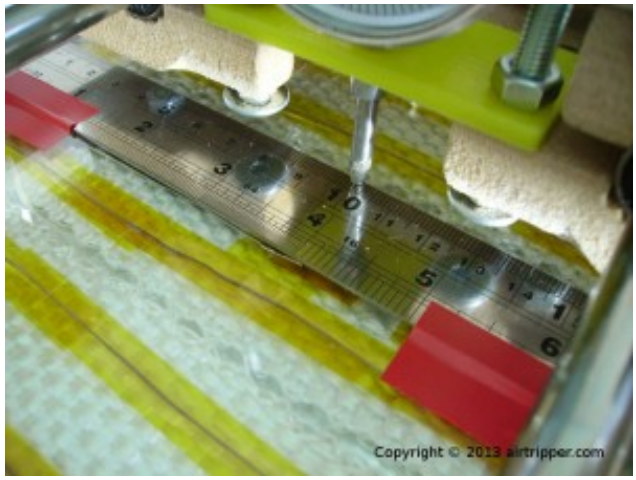
RAMPS 1.3 MECHANICAL END STOP CONNECTIONS X–, Y– & Z+. THESE SHOULD ALSO MATCH END STOP HOME DIRECTION.

Some motherboards like the Ramps 1.3 have a connector for each end of each axis, totalling six connectors. You will need to be sure that the settings above match the end stop connections to the motherboard.

Travel Limits – line 249

#define X_MAX_POS 130
#define X_MIN_POS 0
#define Y_MAX_POS 130
#define Y_MIN_POS 0
#define Z_MAX_POS 107
#define Z_MIN_POS 0

The above defines the printable area of the 3d printer after homing. For the X and Y axis you just measure the travel length of the nozzle from the home position. The maximum travel length will either be restricted by the size of the build platform or by the maximum travel distance of the axis.



MEASURING TRAVEL LIMITS FOR MARLIN FIRMWARE CONFIGURATION. USING A DIAL INDICATOR CAN MAKE IT EASIER TO SEE THE MEASUREMENTS.

When setting up the Z axis for the first time, it is best to set the Z axis travel length a bit shorter than what is measured until a software and hardware test of the 3d printer is completed. This will help to avoid accidentally crashing the build bed into the hot end during initial tests. The Z axis travel length can be fine tuned later after the tests have been satisfied.

The measurement units are in millimeters and are defined from line 249 for the maximum positions in the Marlin firmware configuration file. The minimum positions can be left at the default 0 for this configuration.

Steps Per Unit – line 275

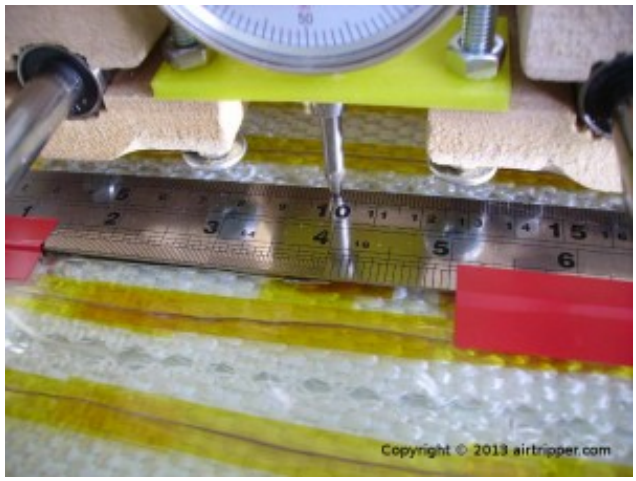
#define DEFAULT_AXIS_STEPS_PER_UNIT {106.76, 106.76, 800, 48.14}

Configuring steps per unit will be one of the last bits of fine tuning you do before you start printing for the first time. Calculating steps per unit accurately will give the 3d printed parts the best start possible. However, if you are just upgrading the Marlin firmware or upgrading from any other firmware, you can get the steps per unit figures from the old configuration file if you still have it.

Steps per unit means the number of steps a stepper motor has to turn to equal 1mm of axis travel. The units at line 275 of the Marlin firmware configuration.h file are in the order of X, Y, Z and E. Getting the steps per unit from another printer of the same design would be very close to what you need, and will help you run some stepper motor tests before you get down to calculating accurate steps per unit for the Marlin firmware configuration file.

The code snippet above shows the steps per unit for my Sumpod 3d printer and is not the default setting in the Marlin firmware configuration.

Steps Per Unit Calculation



MEASURING TRAVEL DISTANCE FOR MARLIN FIRMWARE STEPS PER UNIT CALCULATION

To calculate steps per unit (mm) to put in the Marlin firmware configuration file, you need to find a good way to measure axis travel distance accurately. You may have to temporary remove the hot end so that the filament can be extruded in order to make measurements. Using a dial indicator in place of the hot end and a ruler taped to the bed can provide a good accurate way to measured travelled distance. To get the best accuracy you should sample at least 100mm of axis travel. You command the printer to move the chosen axis 100mm using an interface software such as Pronterface. You then measure the actual distance the the axis travelled. Using the formula below you can calculate the new steps per unit.

Steps Per Unit Formula

NewStepsPerUnit = SampleTravelDist / ActualTravelDist x OldStepsPerUnit

You then repeat the above formula as many times as necessary until the commanded travel distance matches the actual distance travelled, using the NewStepsPerUnit as the OldStepsPerUnit each time.

Marlin Firmware Basic Configuration – The End

At this point, you have done enough configuration in the Marlin firmware and can now start 3d printing. If you are interested in LCD display and click encoders, read on.

The last section in the Marlin firmware configuration file is for additional features, this section allows you to configure some of the optional extras you might have attached to your 3d printer. For the purpose of this guide I'm just going to include notes for the LCD 16×2 and the LCD 20×4 with click encoder control panel. The RAMPS 1.3 Arduino shield and my Marlin firmware configuration will be used for this guide.

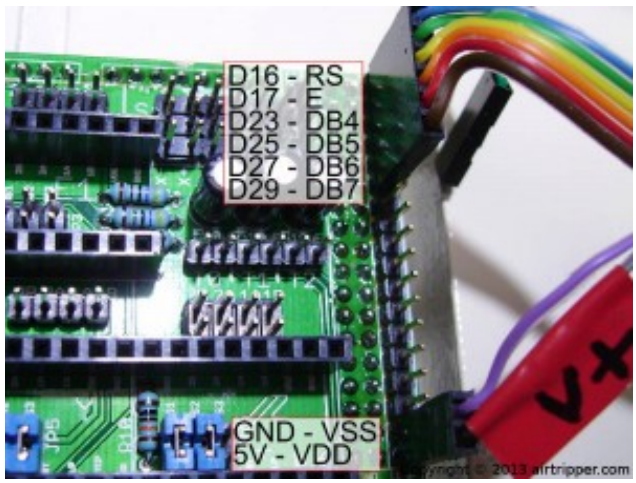
SUMPOD LCD CLICK ENCODER CONTROL PANEL CONFIGURED WITH THE MARLIN FIRMWARE

Enabling an attached 16×2 LCD or click encoder control panel is straight forward in the Marlin firmware. However, the pin assignments for the attachment connectors need to match those in the Marlin firmware pins.h file. You can check if the LCD and click encoder panel features are supported for your motherboard by looking through the the pins.h file of the Marlin firmware. If you are just updating the Marlin firmware you can check for pin assignment changes by comparing your old pins.h configuration file with the new version of that file. Any changes found can be used to update the latest version of the Marlin firmware.

Adding LCD Support – line 303

#define ULTRA_LCD

To enable any type of LCD support in the Marlin firmware, *ULTRA_LCD* needs to be defined. To add support for 16×2 LCD display, line 303 needs to be uncommented like the code snippet above, by removing the preceding forward slashes. By default, the Marlin firmware does not have LCD or click encoder control panel enabled. Enabling line 303 and nothing else will give you support for 16×2 LCD screen that is connected to the motherboard pins as assigned in the pins.h file. Pin assignments are found in the pins.h file of the Marlin firmware



RAMPS 1.3 TO LCD PIN CONNECTIONS – MARLIN FIRMWARE DEFAULT PIN ASSIGNMENT FOR THIS BOARD.

under each supported motherboard type.

Adding Click Encoder Support

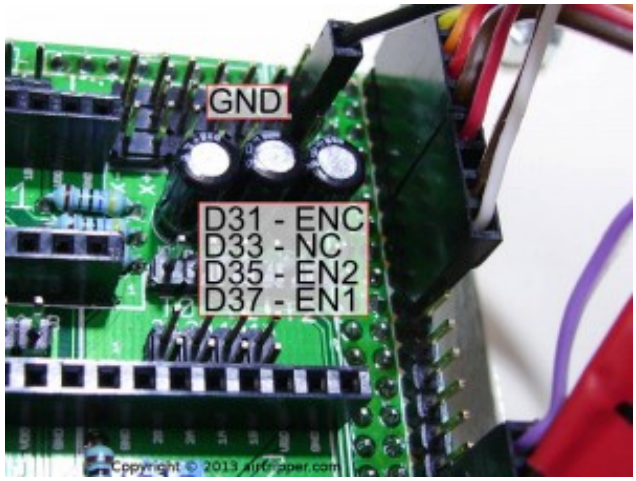
There are two popular types of click encoder control panels that can be enabled for the RAMPS 1.3 board. The first type is the Ultipanel, which can be found on Thingiverse, and the other type is the RepRapDiscount Smart Controller. For this guide I'm just going add notes for the Ultipanel since the other type is supported by RepRapDiscount RepRap wiki for the Marlin firmware.

Enabling ULTIPANEL – line 307

#define ULTIPANEL

Enabling NEWPANEL – line 331

#define NEWPANEL



RAMPS 1.3 CLICK ENCODER PIN CONNECTIONS – MARLIN FIRMWARE DEFAULT PIN ASSIGNMENT FOR THIS BOARD & ULTIPANEL.

Both lines 307 and 331 are not enabled by default. To enable 20×4 LCD display and click encoder, uncomment both lines by removing the forward slashes. This would also enable SD Card support as well, a feature that will be covered in another guide. It will not be necessary to uncomment line 303 if line 307 is enabled by uncommenting. *ULTRA_LCD* will automatically be defined when ULTIPANEL is defined.

Marlin Firmware Guide – The End

Well that concludes the Marlin firmware guide for now, and I would expect to be making updates going forward to correct errors if any are found or just to improve certain aspects of the guide.

If you have any questions or comments about the Marlin firmware guide, please leave them below. However, if you need Marlin firmware support, this is perhaps not the best place to get it.

7 RESPONSES TO MARLIN FIRMWARE V1, BASIC CONFIGURATION SET-UP GUIDE

Kevin *says:*
FEBRUARY 25, 2013 AT 6:33 PM

Wow. This is a fantastically organized and crystal clear post. Thank you so much, it's helped tremendously for me!

REPLY

J Mir *says:*
MARCH 4, 2013 AT 7:43 AM

hi there mark,

Thnks for the guide, one q thou, i need to use a RAMPS 1.4 to print with a TOM hardware.
i just have trouble make the Z axis and/or the gcode with slic3r be the correct one, x,

y, and e, seems to work fine, but Z keeps struck at the Z home position, on the gcode it only move .325 on the firs Z position, i dont get to Z goes all the way down to the hot bed and print where it supposed to.

As the TOM used to work with its old firmware, first it need to be centered the axis, xyz, then run a calibration scrip, generate the gcode and then the z axis came from the top, positioning just right above the hot bed and print.

i think i might have something wrong with aligning to the center/homing/calibrating.

Hope u could help.

Regards

REPLY

Mark Heywood *says:*
MARCH 5, 2013 AT 1:55 AM

It could be that the firmware configuration is not correct for the Z end stop on your printer.
Your printer Z axis may not be moving to the bed to print because according to the firmware your printer Z axis is already at 0 (Z_MIN). If your Z axis should home away from the bed then the firmware needs to be set to home the Z axis to Z_MAX. This is done by the following code at about line 244:
#define Z_HOME_DIR 1
Also, the Z endstop needs to be connected to the +Z connector on the RAMPS 1.4.

If your Z axis does indeed home away from the bed you must make sure the Z_MAX_POS travel measurement between the nozzle and the bed is correct to avoid crashing.

If the above did not help, send me your Marlin configuration.h file to airtripper2010@gmail.com to check if it is consistent with your hardware set up.

Mark

REPLY

J Mir *says:*

MARCH 12, 2013 AT 12:48 AM

Dear Mark

Appreciate the response let me try this things and get back to you, thanks

REPLY

Martin *says:*

MARCH 11, 2013 AT 3:42 PM

Hi Mark,

The guide is very useful. I'm trying to use a Ramps 1.4 with Ultipanel and a dual extruder but it doesn't work neither with a single extruder. It seems that the click/encoder it's wrong configurated because the LCD it's continuously changing between the main interface and the main menu.

I check the pins with the previous firmware (the marlin not v1) and they are the same. What do you think? I don't know what I can do.

Thanks for all.

Martin.

REPLY

Mark Heywood *says:*

MARCH 11, 2013 AT 7:40 PM

Hi Martin,

It looks like you might have the click encoders wires mixed up. Your Marlin firmware thinks you are holding down the click button and that's why the menu on the LCD Screen changes back and forth. You may have a click encoder rotary wire connected where the click button wire should go. The pictures above show the configuration for the ULTIPANEL definition.

There are two click encoder pin configurations, one for ULTIPANEL and one for REPRAP_DISCOUNT_SMART_CONTROLLER. If you purchased the LCD and click encoder as a kit then it might be wired for

REPRAP_DISCOUNT_SMART_CONTROLLER configuration. This means you will have to #define REPRAP_DISCOUNT_SMART_CONTROLLER instead of #define ULTIPANEL in the Marlin configuration.h file.

The wiring looks very similar for both ULTIPANEL and REPRAP_DISCOUNT_SMART_CONTROLLER and setting the wrong definition in firmware could cause the symptoms you described on the LCD Screen.

You could just keep the firmware as it is and just correct the wires on the click encoder.

Mark

REPLY

Martin *says:*

MARCH 12, 2013 AT 10:10 AM

Hi again Mark,

It was that, thanks you so much for your help. I confuse the both LCD's.

Regards.

Martin.

REPLY