Frequently Asked Questions

How can I donate to the project?

Thank you for your support. See the Sponsors page for information.

How do I calculate the rotation_distance config parameter?

See the rotation distance document.

Where's my serial port?

The general way to find a USB serial port is to run `ls /dev/serial/by-id/*` from an ssh terminal on the host machine. It will likely produce output similar to the following:

```
/dev/serial/by-id/usb-1a86_USB2.0-Serial-if00-port0
```

The name found in the above command is stable and it is possible to use it in the config file and while flashing the micro-controller code. For example, a flash command might look similar to:

```
sudo service klipper stop
make flash FLASH_DEVICE=/dev/serial/by-id/usb-1a86_USB2.0-Serial-if00-
port0
sudo service klipper start
```

and the updated config might look like:

```
[mcu]
serial: /dev/serial/by-id/usb-1a86_USB2.0-Serial-if00-port0
```

Be sure to copy-and-paste the name from the "ls" command that you ran above as the name will be different for each printer.

If you are using multiple micro-controllers and they do not have unique ids (common on boards with a CH340 USB chip) then follow the directions above using the command `ls /dev/serial/by-path/*` instead.

When the micro-controller restarts the device changes to /dev/ttyUSB1

Follow the directions in the "Where's my serial port?" section to prevent this from occurring.

The "make flash" command doesn't work

The code attempts to flash the device using the most common method for each platform. Unfortunately, there is a lot of variance in flashing methods, so the "make flash" command may not work on all boards.

If you're having an intermittent failure or you do have a standard setup, then double check that Klipper isn't running when flashing (sudo service klipper stop), make sure OctoPrint isn't trying to connect directly to the device (open the Connection tab in the web page and click Disconnect if the Serial Port is set to the device), and make sure FLASH_DEVICE is set correctly for your board (see the question above).

However, if "make flash" just doesn't work for your board, then you will need to manually flash. See if there is a config file in the config directory with specific instructions for flashing the device. Also, check the board manufacturer's documentation to see if it describes how to flash the device. Finally, it may be possible to manually flash the device using tools such as "avrdude" or "bossac" - see the bootloader document for additional information.

How do I change the serial baud rate?

The recommended baud rate for Klipper is 250000. This baud rate works well on all micro-controller boards that Klipper supports. If you've found an online guide recommending a different baud rate, then ignore that part of the guide and continue with the default value of 250000.

If you want to change the baud rate anyway, then the new rate will need to be configured in the micro-controller (during **make menuconfig**) and that updated code will need to be compiled and flashed to the micro-controller. The Klipper printer.cfg file will also need to be updated to match that baud rate (see the config reference for details). For example:

```
[mcu]
baud: 250000
```

The baud rate shown on the OctoPrint web page has no impact on the internal Klipper micro-controller baud rate. Always set the OctoPrint baud rate to 250000 when using Klipper.

The Klipper micro-controller baud rate is not related to the baud rate of the micro-controller's bootloader. See the bootloader document for additional information on bootloaders.

Can I run Klipper on something other than a Raspberry Pi 3?

The recommended hardware is a Raspberry Pi 2, Raspberry Pi 3, or Raspberry Pi 4.

Klipper will run on a Raspberry Pi 1 and on the Raspberry Pi Zero, but these boards don't have enough processing power to run OctoPrint well. It is common for print stalls to occur on these slower machines when printing directly from OctoPrint. (The printer may move faster than OctoPrint can send movement commands.) If you wish to run on one one of these slower boards anyway, consider using the "virtual_sdcard" feature when printing (see config reference for details).

For running on the Beaglebone, see the Beaglebone specific installation instructions.

Klipper has been run on other machines. The Klipper host software only requires Python running on a Linux (or similar) computer. However, if you wish to run it on a different machine you will need Linux admin

knowledge to install the system prerequisites for that particular machine. See the install-octopi.sh script for further information on the necessary Linux admin steps.

If you are looking to run the Klipper host software on a low-end chip, then be aware that, at a minimum, a machine with "double precision floating point" hardware is required.

If you are looking to run the Klipper host software on a shared general-purpose desktop or server class machine, then note that Klipper has some real-time scheduling requirements. If, during a print, the host computer also performs an intensive general-purpose computing task (such as defragmenting a hard drive, 3d rendering, heavy swapping, etc.), then it may cause Klipper to report print errors.

Note: If you are not using an OctoPi image, be aware that several Linux distributions enable a "ModemManager" (or similar) package that can disrupt serial communication. (Which can cause Klipper to report seemingly random "Lost communication with MCU" errors.) If you install Klipper on one of these distributions you may need to disable that package.

Can I run multiple instances of Klipper on the same host machine?

It is possible to run multiple instances of the Klipper host software, but doing so requires Linux admin knowledge. The Klipper installation scripts ultimately cause the following Unix command to be run:

```
~/klippy-env/bin/python ~/klipper/klippy/klippy.py ~/printer.cfg -l
/tmp/klippy.log
```

One can run multiple instances of the above command as long as each instance has its own printer config file, its own log file, and its own pseudo-tty. For example:

```
~/klippy-env/bin/python ~/klipper/klippy/klippy.py ~/printer2.cfg -l
/tmp/klippy2.log -I /tmp/printer2
```

If you choose to do this, you will need to implement the necessary start, stop, and installation scripts (if any). The install-octopi.sh script and the klipper-start.sh script may be useful as examples.

Do I have to use OctoPrint?

The Klipper software is not dependent on OctoPrint. It is possible to use alternative software to send commands to Klipper, but doing so requires Linux admin knowledge.

Klipper creates a "virtual serial port" via the "/tmp/printer" file, and it emulates a classic 3d-printer serial interface via that file. In general, alternative software may work with Klipper as long as it can be configured to use "/tmp/printer" for the printer serial port.

Why can't I move the stepper before homing the printer?

The code does this to reduce the chance of accidentally commanding the head into the bed or a wall. Once the printer is homed the software attempts to verify each move is within the position_min/max defined in

the config file. If the motors are disabled (via an M84 or M18 command) then the motors will need to be homed again prior to movement.

If you want to move the head after canceling a print via OctoPrint, consider changing the OctoPrint cancel sequence to do that for you. It's configured in OctoPrint via a web browser under: Settings->GCODE Scripts

If you want to move the head after a print finishes, consider adding the desired movement to the "custom g-code" section of your slicer.

If the printer requires some additional movement as part of the homing process itself (or fundamentally does not have a homing process) then consider using a safe_z_home or homing_override section in the config file. If you need to move a stepper for diagnostic or debugging purposes then consider adding a force_move section to the config file. See config reference for further details on these options.

Why is the Z position_endstop set to 0.5 in the default configs?

For cartesian style printers the Z position_endstop specifies how far the nozzle is from the bed when the endstop triggers. If possible, it is recommended to use a Z-max endstop and home away from the bed (as this reduces the potential for bed collisions). However, if one must home towards the bed then it is recommended to position the endstop so it triggers when the nozzle is still a small distance away from the bed. This way, when homing the axis, it will stop before the nozzle touches the bed. See the bed level document for more information.

I converted my config from Marlin and the X/Y axes work fine, but I just get a screeching noise when homing the Z axis

Short answer: First, make sure you have verified the stepper configuration as described in the config check document. If the problem persists, try reducing the max_z_velocity setting in the printer config.

Long answer: In practice Marlin can typically only step at a rate of around 10000 steps per second. If it is requested to move at a speed that would require a higher step rate then Marlin will generally just step as fast as it can. Klipper is able to achieve much higher step rates, but the stepper motor may not have sufficient torque to move at a higher speed. So, for a Z axis with a high gearing ratio or high microsteps setting the actual obtainable max_z_velocity may be smaller than what is configured in Marlin.

My TMC motor driver turns off in the middle of a print

If using the TMC2208 (or TMC2224) driver in "standalone mode" then make sure to use the latest version of Klipper. A workaround for a TMC2208 "stealthchop" driver problem was added to Klipper in mid-March of 2020.

I keep getting random "Lost communication with MCU" errors

This is commonly caused by hardware errors on the USB connection between the host machine and the micro-controller. Things to look for:

- Use a good quality USB cable between the host machine and micro-controller. Make sure the plugs are secure.
- If using a Raspberry Pi, use a good quality power supply for the Raspberry Pi and use a good quality USB cable to connect that power supply to the Pi. If you get "under voltage" warnings from

OctoPrint, this is related to the power supply and it must be fixed.
- Make sure the printer's power supply is not being overloaded. (Power fluctuations to the micro-controller's USB chip may result in resets of that chip.)
- Verify stepper, heater, and other printer wires are not crimped or frayed. (Printer movement may place stress on a faulty wire causing it to lose contact, briefly short, or generate excessive noise.)
- There have been reports of high USB noise when both the printer's power supply and the host's 5V power supply are mixed. (If you find that the micro-controller powers on when either the printer's power supply is on or the USB cable is plugged in, then it indicates the 5V power supplies are being mixed.) It may help to configure the micro-controller to use power from only one source. (Alternatively, if the micro-controller board can not configure its power source, one may modify a USB cable so that it does not carry 5V power between the host and micro-controller.)

My Raspberry Pi keeps rebooting during prints

This is most likely do to voltage fluctuations. Follow the same troubleshooting steps for a "Lost communication with MCU" error.

When I set `restart_method=command` my AVR device just hangs on a restart

Some old versions of the AVR bootloader have a known bug in watchdog event handling. This typically manifests when the printer.cfg file has restart_method set to "command". When the bug occurs, the AVR device will be unresponsive until power is removed and reapplied to the device (the power or status LEDs may also blink repeatedly until the power is removed).

The workaround is to use a restart_method other than "command" or to flash an updated bootloader to the AVR device. Flashing a new bootloader is a one time step that typically requires an external programmer - see Bootloaders for further details.

Will the heaters be left on if the Raspberry Pi crashes?

The software has been designed to prevent that. Once the host enables a heater, the host software needs to confirm that enablement every 5 seconds. If the micro-controller does not receive a confirmation every 5 seconds it goes into a "shutdown" state which is designed to turn off all heaters and stepper motors.

See the "config_digital_out" command in the MCU commands document for further details.

In addition, the micro-controller software is configured with a minimum and maximum temperature range for each heater at startup (see the min_temp and max_temp parameters in the config reference for details). If the micro-controller detects that the temperature is outside of that range then it will also enter a "shutdown" state.

Separately, the host software also implements code to check that heaters and temperature sensors are functioning correctly. See the config reference for further details.

How do I convert a Marlin pin number to a Klipper pin name?

Short answer: A mapping is available in the sample-aliases.cfg file. Use that file as a guide to finding the actual micro-controller pin names. (It is also possible to copy the relevant board_pins config section into

your config file and use the aliases in your config, but it is preferable to translate and use the actual micro-controller pin names.) Note that the sample-aliases.cfg file uses pin names that start with the prefix "ar" instead of "D" (eg, Arduino pin `D23` is Klipper alias `ar23`) and the prefix "analog" instead of "A" (eg, Arduino pin `A14` is Klipper alias `analog14`).

Long answer: Klipper uses the standard pin names defined by the micro-controller. On the Atmega chips these hardware pins have names like `PA4`, `PC7`, or `PD2`.

Long ago, the Arduino project decided to avoid using the standard hardware names in favor of their own pin names based on incrementing numbers - these Arduino names generally look like `D23` or `A14`. This was an unfortunate choice that has lead to a great deal of confusion. In particular the Arduino pin numbers frequently don't translate to the same hardware names. For example, `D21` is `PD0` on one common Arduino board, but is `PC7` on another common Arduino board.

To avoid this confusion, the core Klipper code uses the standard pin names defined by the micro-controller.

Do I have to wire my device to a specific type of micro-controller pin?

It depends on the type of device and type of pin:

ADC pins (or Analog pins): For thermistors and similar "analog" sensors, the device must be wired to an "analog" or "ADC" capable pin on the micro-controller. If you configure Klipper to use a pin that is not analog capable, Klipper will report a "Not a valid ADC pin" error.

PWM pins (or Timer pins): Klipper does not use hardware PWM by default for any device. So, in general, one may wire heaters, fans, and similar devices to any general purpose IO pin. However, fans and output_pin devices may be optionally configured to use `hardware_pwm: True`, in which case the micro-controller must support hardware PWM on the pin (otherwise, Klipper will report a "Not a valid PWM pin" error).

IRQ pins (or Interrupt pins): Klipper does not use hardware interrupts on IO pins, so it is never necessary to wire a device to one of these micro-controller pins.

SPI pins: When using hardware SPI it is necessary to wire the pins to the micro-controller's SPI capable pins. However, most devices can be configured to use "software SPI", in which case any general purpose IO pins may be used.

I2C pins: When using I2C it is necessary to wire the pins to the micro-controller's I2C capable pins.

Other devices may be wired to any general purpose IO pin. For example, steppers, heaters, fans, Z probes, servos, LEDs, common hd44780/st7920 LCD displays, the Trinamic UART control line may be wired to any general purpose IO pin.

How do I cancel an M109/M190 "wait for temperature" request?

Navigate to the OctoPrint terminal tab and issue an M112 command in the terminal box. The M112 command will cause Klipper to enter into a "shutdown" state, and it will cause OctoPrint to disconnect from Klipper. Navigate to the OctoPrint connection area and click on "Connect" to cause OctoPrint to reconnect. Navigate back to the terminal tab and issue a FIRMWARE_RESTART command to clear the Klipper error

state. After completing this sequence, the previous heating request will be canceled and a new print may be started.

Can I find out whether the printer has lost steps?

In a way, yes. Home the printer, issue a `GET_POSITION` command, run your print, home again and issue another `GET_POSITION`. Then compare the values in the `mcu:` line.

This might be helpful to tune settings like stepper motor currents, accelerations and speeds without needing to actually print something and waste filament: just run some high-speed moves in between the `GET_POSITION` commands.

Note that endstop switches themselves tend to trigger at slightly different positions, so a difference of a couple of microsteps is likely the result of endstop inaccuracies. A stepper motor itself can only lose steps in increments of 4 full steps. (So, if one is using 16 microsteps, then a lost step on the stepper would result in the "mcu:" step counter being off by a multiple of 64 microsteps.)

Why does Klipper report errors? I lost my print!

Short answer: We want to know if our printers detect a problem so that the underlying issue can be fixed and we can obtain great quality prints. We definitely do not want our printers to silently produce low quality prints.

Long answer: Klipper has been engineered to automatically workaround many transient problems. For example, it automatically detects communication errors and will retransmit; it schedules actions in advance and buffers commands at multiple layers to enable precise timing even with intermittent interference. However, should the software detect an error that it can not recover from, if it is commanded to take an invalid action, or if it detects it is hopelessly unable to perform its commanded task, then Klipper will report an error. In these situations there is a high risk of producing a low-quality print (or worse). It is hoped that alerting the user will empower them to fix the underlying issue and improve the overall quality of their prints.

There are some related questions: Why doesn't Klipper pause the print instead? Report a warning instead? Check for errors before the print? Ignore errors in user typed commands? etc? Currently Klipper reads commands using the G-Code protocol, and unfortunately the G-Code command protocol is not flexible enough to make these alternatives practical today. There is developer interest in improving the user experience during abnormal events, but it is expected that will require notable infrastructure work (including a shift away from G-Code).

How do I upgrade to the latest software?

The first step to upgrading the software is to review the latest config changes document. On occasion, changes are made to the software that require users to update their settings as part of a software upgrade. It is a good idea to review this document prior to upgrading.

When ready to upgrade, the general method is to ssh into the Raspberry Pi and run:

```
cd ~/klipper
git pull
~/klipper/scripts/install-octopi.sh
```

Then one can recompile and flash the micro-controller code. For example:

```
make menuconfig
make clean
make

sudo service klipper stop
make flash FLASH_DEVICE=/dev/ttyACM0
sudo service klipper start
```

However, it's often the case that only the host software changes. In this case, one can update and restart just the host software with:

```
cd ~/klipper
git pull
sudo service klipper restart
```

If after using this shortcut the software warns about needing to reflash the micro-controller or some other unusual error occurs, then follow the full upgrade steps outlined above.

If any errors persist then double check the config changes document, as you may need to modify the printer configuration.

Note that the RESTART and FIRMWARE_RESTART g-code commands do not load new software - the above "sudo service klipper restart" and "make flash" commands are needed for a software change to take effect.

How do I uninstall Klipper?

On the firmware end, nothing special needs to happen. Just follow the flashing directions for the new firmware.

On the raspberry pi end, an uninstall script is available in scripts/klipper-uninstall.sh. For example:

```
sudo ~/klipper/scripts/klipper-uninstall.sh
rm -rf ~/klippy-env ~/klipper
```