TMC drivers

This document provides information on using Trinamic stepper motor drivers in SPI/UART mode on Klipper.

Klipper can also use Trinamic drivers in their "standalone mode". However, when the drivers are in this mode, no special Klipper configuration is needed and the advanced Klipper features discussed in this document are not available.

In addition to this document, be sure to review the TMC driver config reference.

Tuning motor current

A higher driver current increases positional accuracy and torque. However, a higher current also increases the heat produced by the stepper motor and the stepper motor driver. If the stepper motor driver gets too hot it will disable itself and Klipper will report an error. If the stepper motor gets too hot, it loses torque and positional accuracy. (If it gets very hot it may also melt plastic parts attached to it or near it.)

As a general tuning tip, prefer higher current values as long as the stepper motor does not get too hot and the stepper motor driver does not report warnings or errors. In general, it is okay for the stepper motor to feel warm, but it should not become so hot that it is painful to touch.

Prefer to not specify a hold_current

If one configures a `hold_current` then the TMC driver can reduce current to the stepper motor when it detects that the stepper is not moving. However, changing motor current may itself introduce motor movement. This may occur due to "detent forces" within the stepper motor (the permanent magnet in the rotor pulls towards the iron teeth in the stator) or due to external forces on the axis carriage.

Most stepper motors will not obtain a significant benefit to reducing current during normal prints, because few printing moves will leave a stepper motor idle for sufficiently long to activate the `hold_current` feature. And, it is unlikely that one would want to introduce subtle print artifacts to the few printing moves that do leave a stepper idle sufficiently long.

If one wishes to reduce current to motors during print start routines, then consider issuing SET_TMC_CURRENT commands in a START_PRINT macro to adjust the current before and after normal printing moves.

Some printers with dedicated Z motors that are idle during normal printing moves (no bed_mesh, no bed_tilt, no Z skew_correction, no "vase mode" prints, etc.) may find that Z motors do run cooler with a `hold_current`. If implementing this then be sure to take into account this type of uncommanded Z axis movement during bed leveling, bed probing, probe calibration, and similar. The `driver_TPOWERDOWN` and `driver_IHOLDDELAY` should also be calibrated accordingly. If unsure, prefer to not specify a `hold_current`.

Setting "spreadCycle" vs "stealthChop" Mode

By default, Klipper places the TMC drivers in "spreadCycle" mode. If the driver supports "stealthChop" then it can be enabled by adding `stealthchop_threshold: 999999` to the TMC config section.

In general, spreadCycle mode provides greater torque and greater positional accuracy than stealthChop mode. However, stealthChop mode may produce significantly lower audible noise on some printers.

Tests comparing modes have shown an increased "positional lag" of around 75% of a full-step during constant velocity moves when using stealthChop mode (for example, on a printer with 40mm rotation_distance and 200 steps_per_rotation, position deviation of constant speed moves increased by ~0.150mm). However, this "delay in obtaining the requested position" may not manifest as a significant print defect and one may prefer the quieter behavior of stealthChop mode.

It is recommended to always use "spreadCycle" mode (by not specifying `stealthchop_threshold`) or to always use "stealthChop" mode (by setting `stealthchop_threshold` to 999999). Unfortunately, the drivers often produce poor and confusing results if the mode changes while the motor is at a non-zero velocity.

TMC interpolate setting introduces small position deviation

The TMC driver `interpolate` setting may reduce the audible noise of printer movement at the cost of introducing a small systemic positional error. This systemic positional error results from the driver's delay in executing "steps" that Klipper sends it. During constant velocity moves, this delay results in a positional error of nearly half a configured microstep (more precisely, the error is half a microstep distance minus a 512th of a full step distance). For example, on an axis with a 40mm rotation_distance, 200 steps_per_rotation, and 16 microsteps, the systemic error introduced during constant velocity moves is ~0.006mm.

For best positional accuracy consider using spreadCycle mode and disable interpolation (set `interpolate: False` in the TMC driver config). When configured this way, one may increase the `microstep` setting to reduce audible noise during stepper movement. Typically, a microstep setting of `64` or `128` will have similar audible noise as interpolation, and do so without introducing a systemic positional error.

If using stealthChop mode then the positional inaccuracy from interpolation is small relative to the positional inaccuracy introduced from stealthChop mode. Therefore tuning interpolation is not considered useful when in stealthChop mode, and one can leave interpolation in its default state.

Sensorless Homing

Sensorless homing allows to home an axis without the need for a physical limit switch. Instead, the carriage on the axis is moved into the mechanical limit making the stepper motor lose steps. The stepper driver senses the lost steps and indicates this to the controlling MCU (Klipper) by toggling a pin. This information can be used by Klipper as end stop for the axis.

This guide covers the setup of sensorless homing for the X axis of your (cartesian) printer. However, it works the same with all other axes (that require an end stop). You should configure and tune it for one axis at a time.

Limitations

Be sure that your mechanical components are able to handle the load of the carriage bumping into the limit of the axis repeatedly. Especially leadscrews might generate a lot of force. Homing a Z axis by bumping the nozzle into the printing surface might not be a good idea. For best results, verify that the axis carriage will make a firm contact with the axis limit.

Further, sensorless homing might not be accurate enough for your printer. While homing X and Y axes on a cartesian machine can work well, homing the Z axis is generally not accurate enough and may result in an inconsistent first layer height. Homing a delta printer sensorless is not advisable due to missing accuracy.

Further, the stall detection of the stepper driver is dependent on the mechanical load on the motor, the motor current and the motor temperature (coil resistance).

Sensorless homing works best at medium motor speeds. For very slow speeds (less than 10 RPM) the motor does not generate significant back EMF and the TMC cannot reliably detect motor stalls. Further, at very high speeds, the back EMF of the motor approaches the supply voltage of the motor, so the TMC cannot detect stalls anymore. It is advised to have a look in the datasheet of your specific TMCs. There you can also find more details on limitations of this setup.

Prerequisites

A few prerequisites are needed to use sensorless homing:

1. A stallGuard capable TMC stepper driver (tmc2130, tmc2209, tmc2660, or tmc5160).
2. SPI / UART interface of the TMC driver wired to micro-controller (stand-alone mode does not work).
3. The appropriate "DIAG" or "SG_TST" pin of TMC driver connected to the micro-controller.
4. The steps in the config checks document must be run to confirm the stepper motors are configured and working properly.

Tuning

The procedure described here has six major steps:

1. Choose a homing speed.
2. Configure the `printer.cfg` file to enable sensorless homing.
3. Find the stallguard setting with highest sensitivity that successfully homes.
4. Find the stallguard setting with lowest sensitivity that successfully homes with a single touch.
5. Update the `printer.cfg` with the desired stallguard setting.
6. Create or update `printer.cfg` macros to home consistently.

Choose homing speed

The homing speed is an important choice when performing sensorless homing. It's desirable to use a slow homing speed so that the carriage does not exert excessive force on the frame when making contact with the end of the rail. However, the TMC drivers can't reliably detect a stall at very slow speeds.

A good starting point for the homing speed is for the stepper motor to make a full rotation every two seconds. For many axes this will be the `rotation_distance` divided by two. For example:

```
[stepper_x]
rotation_distance: 40
homing_speed: 20
...
```

Configure printer.cfg for sensorless homing

The `homing_retract_dist` setting must be set to zero in the `stepper_x` config section to disable the second homing move. The second homing attempt does not add value when using sensorless homing, it will not work reliably, and it will confuse the tuning process.

Be sure that a `hold_current` setting is not specified in the TMC driver section of the config. (If a hold_current is set then after contact is made, the motor stops while the carriage is pressed against the end of the rail, and reducing the current while in that position may cause the carriage to move - that results in poor performance and will confuse the tuning process.)

It is necessary to configure the sensorless homing pins and to configure initial "stallguard" settings. A tmc2209 example configuration for an X axis might look like:

```
[tmc2209 stepper_x]
diag_pin: ^PA1      # Set to MCU pin connected to TMC DIAG pin
driver_SGTHRS: 255  # 255 is most sensitive value, 0 is least sensitive
...

[stepper_x]
endstop_pin: tmc2209_stepper_x:virtual_endstop
homing_retract_dist: 0
...
```

An example tmc2130 or tmc5160 config might look like:

```
[tmc2130 stepper_x]
diag1_pin: ^!PA1 # Pin connected to TMC DIAG1 pin (or use diag0_pin /
DIAG0 pin)
driver_SGT: -64  # -64 is most sensitive value, 63 is least sensitive
...

[stepper_x]
endstop_pin: tmc2130_stepper_x:virtual_endstop
homing_retract_dist: 0
...
```

An example tmc2660 config might look like:

```
[tmc2660 stepper_x]
driver_SGT: -64     # -64 is most sensitive value, 63 is least sensitive
...

[stepper_x]
endstop_pin: ^PA1   # Pin connected to TMC SG_TST pin
```

```
homing_retract_dist: 0
...
```

The examples above only show settings specific to sensorless homing. See the config reference for all the available options.

Find highest sensitivity that successfully homes

Place the carriage near the center of the rail. Use the SET_TMC_FIELD command to set the highest sensitivity. For tmc2209:

```
SET_TMC_FIELD STEPPER=stepper_x FIELD=SGTHRS VALUE=255
```

For tmc2130, tmc5160, and tmc2660:

```
SET_TMC_FIELD STEPPER=stepper_x FIELD=sgt VALUE=-64
```

Then issue a `G28 X0` command and verify the axis does not move at all or quickly stops moving. If the axis does not stop, then issue an `M112` to halt the printer - something is not correct with the diag/sg_tst pin wiring or configuration and it must be corrected before continuing.

Next, continually decrease the sensitivity of the `VALUE` setting and run the `SET_TMC_FIELD` `G28 X0` commands again to find the highest sensitivity that results in the carriage successfully moving all the way to the endstop and halting. (For tmc2209 drivers this will be decreasing SGTHRS, for other drivers it will be increasing sgt.) Be sure to start each attempt with the carriage near the center of the rail (if needed issue `M84` and then manually move the carriage to the center). It should be possible to find the highest sensitivity that homes reliably (settings with higher sensitivity result in small or no movement). Note the found value as *maximum_sensitivity*. (If the minimum possible sensitivity (SGTHRS=0 or sgt=63) is obtained without any carriage movement then something is not correct with the diag/sg_tst pin wiring or configuration and it must be corrected before continuing.)

When searching for maximum_sensitivity, it may be convenient to jump to different VALUE settings (so as to bisect the VALUE parameter). If doing this then be prepared to issue an `M112` command to halt the printer, as a setting with a very low sensitivity may cause the axis to repeatedly "bang" into the end of the rail.

Be sure to wait a couple of seconds between each homing attempt. After the TMC driver detects a stall it may take a little time for it to clear its internal indicator and be capable of detecting another stall.

During these tuning tests, if a `G28 X0` command does not move all the way to the axis limit, then be careful with issuing any regular movement commands (eg, `G1`). Klipper will not have a correct understanding of the carriage position and a move command may cause undesirable and confusing results.

Find lowest sensitivity that homes with one touch

When homing with the found *maximum_sensitivity* value, the axis should move to the end of the rail and stop with a "single touch" - that is, there should not be a "clicking" or "banging" sound. (If there is a banging or clicking sound at maximum_sensitivity then the homing_speed may be too low, the driver current may be too low, or sensorless homing may not be a good choice for the axis.)

The next step is to again continually move the carriage to a position near the center of the rail, decrease the sensitivity, and run the `SET_TMC_FIELD` `G28 X0` commands - the goal is now to find the lowest sensitivity that still results in the carriage successfully homing with a "single touch". That is, it does not "bang" or "click" when contacting the end of the rail. Note the found value as *minimum_sensitivity*.

Update printer.cfg with sensitivity value

After finding *maximum_sensitivity* and *minimum_sensitivity*, use a calculator to obtain the recommend sensitivity as *minimum_sensitivity + (maximum_sensitivity - minimum_sensitivity)/3*. The recommended sensitivity should be in the range between the minimum and maximum, but slightly closer to the minimum. Round the final value to the nearest integer value.

For tmc2209 set this in the config as `driver_SGTHRS`, for other TMC drivers set this in the config as `driver_SGT`.

If the range between *maximum_sensitivity* and *minimum_sensitivity* is small (eg, less than 5) then it may result in unstable homing. A faster homing speed may increase the range and make the operation more stable.

Note that if any change is made to driver current, homing speed, or a notable change is made to the printer hardware, then it will be necessary to run the tuning process again.

Using Macros when Homing

After sensorless homing completes the carriage will be pressed against the end of the rail and the stepper will exert a force on the frame until the carriage is moved away. It is a good idea to create a macro to home the axis and immediately move the carriage away from the end of the rail.

It is a good idea for the macro to pause at least 2 seconds prior to starting sensorless homing (or otherwise ensure that there has been no movement on the stepper for 2 seconds). Without a delay it is possible for the driver's internal stall flag to still be set from a previous move.

It can also be useful to have that macro set the driver current before homing and set a new current after the carriage has moved away.

An example macro might look something like:

```
[gcode_macro SENSORLESS_HOME_X]
gcode:
    {% set HOME_CUR = 0.700 %}
    {% set driver_config = printer.configfile.settings['tmc2209
stepper_x'] %}
    {% set RUN_CUR = driver_config.run_current %}
    # Set current for sensorless homing
    SET_TMC_CURRENT STEPPER=stepper_x CURRENT={HOME_CUR}
    # Pause to ensure driver stall flag is clear
    G4 P2000
    # Home
    G28 X0
    # Move away
    G90
    G1 X5 F1200
    # Set current during print
    SET_TMC_CURRENT STEPPER=stepper_x CURRENT={RUN_CUR}
```

The resulting macro can be called from a homing_override config section or from a START_PRINT macro.

Note that if the driver current during homing is changed, then the tuning process should be run again.

Tips for sensorless homing on CoreXY

It is possible to use sensorless homing on the X and Y carriages of a CoreXY printer. Klipper uses the `[stepper_x]` stepper to detect stalls when homing the X carriage and uses the `[stepper_y]` stepper to detect stalls when homing the Y carriage.

Use the tuning guide described above to find the appropriate "stall sensitivity" for each carriage, but be aware of the following restrictions:

1. When using sensorless homing on CoreXY, make sure there is no `hold_current` configured for either stepper.
2. While tuning, make sure both the X and Y carriages are near the center of their rails before each home attempt.
3. After tuning is complete, when homing both X and Y, use macros to ensure that one axis is homed first, then move that carriage away from the axis limit, pause for at least 2 seconds, and then start the homing of the other carriage. The move away from the axis avoids homing one axis while the other is pressed against the axis limit (which may skew the stall detection). The pause is necessary to ensure the driver's stall flag is cleared prior to homing again.

An example CoreXY homing macro might look like:

```
[gcode_macro HOME]
gcode:
    G90
    # Home Z
    G28 Z0
    G1 Z10 F1200
    # Home Y
    G28 Y0
    G1 Y5 F1200
    # Home X
    G4 P2000
    G28 X0
    G1 X5 F1200
```

Querying and diagnosing driver settings

The `DUMP_TMC command is a useful tool when configuring and diagnosing the drivers. It will report all fields configured by Klipper as well as all fields that can be queried from the driver.

All of the reported fields are defined in the Trinamic datasheet for each driver. These datasheets can be found on the Trinamic website. Obtain and review the Trinamic datasheet for the driver to interpret the results of DUMP_TMC.

Configuring driver_XXX settings

Klipper supports configuring many low-level driver fields using `driver_XXX` settings. The TMC driver config reference has the full list of fields available for each type of driver.

In addition, almost all fields can be modified at run-time using the SET_TMC_FIELD command.

Each of these fields is defined in the Trinamic datasheet for each driver. These datasheets can be found on the Trinamic website.

Note that the Trinamic datasheets sometime use wording that can confuse a high-level setting (such as "hysteresis end") with a low-level field value (eg, "HEND"). In Klipper, `driver_XXX` and SET_TMC_FIELD always set the low-level field value that is actually written to the driver. So, for example, if the Trinamic datasheet states that a value of 3 must be written to the HEND field to obtain a "hysteresis end" of 0, then set `driver_HEND=3` to obtain the high-level value of 0.

Common Questions

Can I use stealthChop mode on an extruder with pressure advance?

Many people successfully use "stealthChop" mode with Klipper's pressure advance. Klipper implements smooth pressure advance which does not introduce any instantaneous velocity changes.

However, "stealthChop" mode may produce lower motor torque and/or produce higher motor heat. It may or may not be an adequate mode for your particular printer.

I keep getting "Unable to read tmc uart 'stepper_x' register IFCNT" errors?

This occurs when Klipper is unable to communicate with a tmc2208 or tmc2209 driver.

Make sure that the motor power is enabled, as the stepper motor driver generally needs motor power before it can communicate with the micro-controller.

If this error occurs after flashing Klipper for the first time, then the stepper driver may have been previously programmed in a state that is not compatible with Klipper. To reset the state, remove all power from the printer for several seconds (physically unplug both USB and power plugs).

Otherwise, this error is typically the result of incorrect UART pin wiring or an incorrect Klipper configuration of the UART pin settings.

I keep getting "Unable to write tmc spi 'stepper_x' register ..." errors?

This occurs when Klipper is unable to communicate with a tmc2130 or tmc5160 driver.

Make sure that the motor power is enabled, as the stepper motor driver generally needs motor power before it can communicate with the micro-controller.

Otherwise, this error is typically the result of incorrect SPI wiring, an incorrect Klipper configuration of the SPI settings, or an incomplete configuration of devices on an SPI bus.

Note that if the driver is on a shared SPI bus with multiple devices then be sure to fully configure every device on that shared SPI bus in Klipper. If a device on a shared SPI bus is not configured, then it may incorrectly respond to commands not intended for it and corrupt the communication to the intended device. If there is a device on a shared SPI bus that can not be configured in Klipper, then use a static_digital_output config section to set the CS pin of the unused device high (so that it will not attempt to use the SPI bus). The board's schematic is often a useful reference for finding which devices are on an SPI bus and their associated pins.

Why did I get a "TMC reports error: ..." error?

This type of error indicates the TMC driver detected a problem and has disabled itself. That is, the driver stopped holding its position and ignored movement commands. If Klipper detects that an active driver has disabled itself, it will transition the printer into a "shutdown" state.

It's also possible that a **TMC reports error** shutdown occurs due to SPI errors that prevent communication with the driver (on tmc2130, tmc5160, or tmc2660). If this occurs, it's common for the reported driver status to show `00000000` or `ffffffff` - for example: `TMC reports error: DRV_STATUS: ffffffff ...` OR `TMC reports error: READRSP@RDSEL2: 00000000 ...`. Such a failure may be due to an SPI wiring problem or may be due to a self-reset or failure of the TMC driver.

Some common errors and tips for diagnosing them:

TMC reports error: `... ot=1(OvertempError!)`

This indicates the motor driver disabled itself because it became too hot. Typical solutions are to decrease the stepper motor current, increase cooling on the stepper motor driver, and/or increase cooling on the stepper motor.

TMC reports error: `... ShortToGND` OR `LowSideShort`

This indicates the driver has disabled itself because it detected very high current passing through the driver. This may indicate a loose or shorted wire to the stepper motor or within the stepper motor itself.

This error may also occur if using stealthChop mode and the TMC driver is not able to accurately predict the mechanical load of the motor. (If the driver makes a poor prediction then it may send too much current through the motor and trigger its own over-current detection.) To test this, disable stealthChop mode and check if the errors continue to occur.

TMC reports error: `... reset=1(Reset)` OR `CS_ACTUAL=0(Reset?)` OR `SE=0(Reset?)`

This indicates that the driver has reset itself mid-print. This may be due to voltage or wiring issues.

TMC reports error: `... uv_cp=1(Undervoltage!)`

This indicates the driver has detected a low-voltage event and has disabled itself. This may be due to wiring or power supply issues.

How do I tune spreadCycle/coolStep/etc. mode on my drivers?

The Trinamic website has guides on configuring the drivers. These guides are often technical, low-level, and may require specialized hardware. Regardless, they are the best source of information.