G-Codes

This document describes the commands that Klipper supports. These are commands that one may enter into the OctoPrint terminal tab.

G-Code commands

Klipper supports the following standard G-Code commands:

- Move (G0 or G1): `G1 [X<pos>] [Y<pos>] [Z<pos>] [E<pos>] [F<speed>]`
- Dwell: `G4 P<milliseconds>`
- Move to origin: ☑ [Y] [Z]
- Turn off motors: `M18` or `M84`
- Wait for current moves to finish: `M400`
- Use absolute/relative distances for extrusion: `M82`, `M83`
- Use absolute/relative coordinates: `G90`, `G91`
- Set position: `G92 [X<pos>] [Y<pos>] [Z<pos>] [E<pos>]`
- Set speed factor override percentage: `M220 S<percent>`
- Set extrude factor override percentage: `M221 S<percent>`
- Set acceleration: `M204 S<value>` OR `M204 P<value> T<value>`
  - Note: If S is not specified and both P and T are specified, then the acceleration is set to the minimum of P and T. If only one of P or T is specified, the command has no effect.
- Get extruder temperature: `M105`
- Set extruder temperature: `M104 [T<index>] [S<temperature>]`
- Set extruder temperature and wait: `M109 [T<index>] S<temperature>`
  - Note: M109 always waits for temperature to settle at requested value
- Set bed temperature: `M140 [S<temperature>]`
- Set bed temperature and wait: `M190 S<temperature>`
  - Note: M190 always waits for temperature to settle at requested value
- Set fan speed: `M106 S<value>`
- Turn fan off: `M107`
- Emergency stop: `M112`
- Get current position: `M114`
- Get firmware version: `M115`

For further details on the above commands see the RepRap G-Code documentation.

Klipper's goal is to support the G-Code commands produced by common 3rd party software (eg, OctoPrint, Printrun, Slic3r, Cura, etc.) in their standard configurations. It is not a goal to support every possible G-Code command. Instead, Klipper prefers human readable "extended G-Code commands". Similarly, the G-Code terminal output is only intended to be human readable - see the API Server document if controlling Klipper from external software.

If one requires a less common G-Code command then it may be possible to implement it with a custom gcode_macro config section. For example, one might use this to implement: `G12`, `G29`, `G30`, `G31`, `M42`, `M80`, `M81`, `T1`, etc.

Additional Commands

Klipper uses "extended" G-Code commands for general configuration and status. These extended commands all follow a similar format - they start with a command name and may be followed by one or more parameters. For example: `SET_SERVO SERVO=myservo ANGLE=5.3`. In this document, the commands and parameters are shown in uppercase, however they are not case sensitive. (So, "SET_SERVO" and "set_servo" both run the same command.)

This section is organized by Klipper module name, which generally follows the section names specified in the printer configuration file. Note that some modules are automatically loaded.

[adxl345]

The following commands are available when an adxl345 config section is enabled.

ACCELEROMETER_MEASURE

`ACCELEROMETER_MEASURE [CHIP=<config_name>] [NAME=<value>]`: Starts accelerometer measurements at the requested number of samples per second. If CHIP is not specified it defaults to "adxl345". The command works in a start-stop mode: when executed for the first time, it starts the measurements, next execution stops them. The results of measurements are written to a file named `/tmp/adxl345-<chip>-<name>.csv` where `<chip>` is the name of the accelerometer chip (`my_chip_name` from `[adxl345 my_chip_name]`) and `<name>` is the optional NAME parameter. If NAME is not specified it defaults to the current time in "YYYYMMDD_HHMMSS" format. If the accelerometer does not have a name in its config section (simply `[adxl345]`) then `<chip>` part of the name is not generated.

ACCELEROMETER_QUERY

`ACCELEROMETER_QUERY [CHIP=<config_name>] [RATE=<value>]`: queries accelerometer for the current value. If CHIP is not specified it defaults to "adxl345". If RATE is not specified, the default value is used. This command is useful to test the connection to the ADXL345 accelerometer: one of the returned values should be a free-fall acceleration (+/- some noise of the chip).

ACCELEROMETER_DEBUG_READ

`ACCELEROMETER_DEBUG_READ [CHIP=<config_name>] REG=<register>`: queries ADXL345 register "register" (e.g. 44 or 0x2C). Can be useful for debugging purposes.

ACCELEROMETER_DEBUG_WRITE

`ACCELEROMETER_DEBUG_WRITE [CHIP=<config_name>] REG=<register> VAL=<value>`: Writes raw "value" into a register "register". Both "value" and "register" can be a decimal or a hexadecimal integer. Use with care, and refer to ADXL345 data sheet for the reference.

[angle]

The following commands are available when an angle config section is enabled.

ANGLE_CALIBRATE

`ANGLE_CALIBRATE CHIP=<chip_name>`: Perform angle calibration on the given sensor (there must be an `[angle chip_name]` config section that has specified a `stepper` parameter). IMPORTANT - this tool will command the stepper motor to move without checking the normal kinematic boundary limits. Ideally the motor should be disconnected from any printer carriage before performing calibration. If the stepper can not be disconnected from the printer, make sure the carriage is near the center of its rail before starting calibration. (The stepper motor may move forwards or backwards two full rotations during this test.) After completing this test use the `SAVE_CONFIG` command to save the calibration data to the config file. In order to use this tool the Python "numpy" package must be installed (see the measuring resonance document for more information).

ANGLE_DEBUG_READ

`ANGLE_DEBUG_READ CHIP=<config_name> REG=<register>`: Queries sensor register "register" (e.g. 44 or 0x2C). Can be useful for debugging purposes. This is only available for tle5012b chips.

ANGLE_DEBUG_WRITE

`ANGLE_DEBUG_WRITE CHIP=<config_name> REG=<register> VAL=<value>`: Writes raw "value" into register "register". Both "value" and "register" can be a decimal or a hexadecimal integer. Use with care, and refer to sensor data sheet for the reference. This is only available for tle5012b chips.

[bed_mesh]

The following commands are available when the bed_mesh config section is enabled (also see the bed mesh guide).

BED_MESH_CALIBRATE

`BED_MESH_CALIBRATE [METHOD=manual] [<probe_parameter>=<value>] [<mesh_parameter>=<value>]`: This command probes the bed using generated points specified by the parameters in the config. After probing, a mesh is generated and z-movement is adjusted according to the mesh. See the PROBE command for details on the optional probe parameters. If METHOD=manual is specified then the manual probing tool is activated - see the MANUAL_PROBE command above for details on the additional commands available while this tool is active.

BED_MESH_OUTPUT

`BED_MESH_OUTPUT PGP=[<0:1>]`: This command outputs the current probed z values and current mesh values to the terminal. If PGP=1 is specified the X, Y coordinates generated by bed_mesh, along with their associated indices, will be output to the terminal.

BED_MESH_MAP

`BED_MESH_MAP`: Like to BED_MESH_OUTPUT, this command prints the current state of the mesh to the terminal. Instead of printing the values in a human readable format, the state is serialized in json format. This allows octoprint plugins to easily capture the data and generate height maps approximating the bed's surface.

BED_MESH_CLEAR

`BED_MESH_CLEAR`: This command clears the mesh and removes all z adjustment. It is recommended to put this in your end-gcode.

BED_MESH_PROFILE

`BED_MESH_PROFILE LOAD=<name> SAVE=<name> REMOVE=<name>`: This command provides profile management for mesh state. LOAD will restore the mesh state from the profile matching the supplied name. SAVE will save the current mesh state to a profile matching the supplied name. Remove will delete the profile matching the supplied name from persistent memory. Note that after SAVE or REMOVE operations have been run the SAVE_CONFIG gcode must be run to make the changes to persistent memory permanent.

BED_MESH_OFFSET

`BED_MESH_OFFSET [X=<value>] [Y=<value>]`: Applies X and/or Y offsets to the mesh lookup. This is useful for printers with independent extruders, as an offset is necessary to produce correct Z adjustment after a tool change.

[bed_screws]

The following commands are available when the bed_screws config section is enabled (also see the manual level guide).

BED_SCREWS_ADJUST

`BED_SCREWS_ADJUST`: This command will invoke the bed screws adjustment tool. It will command the nozzle to different locations (as defined in the config file) and allow one to make adjustments to the bed screws so that the bed is a constant distance from the nozzle.

[bed_tilt]

The following commands are available when the bed_tilt config section is enabled.

BED_TILT_CALIBRATE

`BED_TILT_CALIBRATE [METHOD=manual] [<probe_parameter>=<value>]`: This command will probe the points specified in the config and then recommend updated x and y tilt adjustments. See the PROBE command for details on the optional probe parameters. If METHOD=manual is specified then the manual probing tool is activated - see the MANUAL_PROBE command above for details on the additional commands available while this tool is active.

[bltouch]

The following command is available when a bltouch config section is enabled (also see the BL-Touch guide).

BLTOUCH_DEBUG

`BLTOUCH_DEBUG COMMAND=<command>`: This sends a command to the BLTouch. It may be useful for debugging. Available commands are: `pin_down`, `touch_mode`, `pin_up`, `self_test`, `reset`. A BL-Touch V3.0 or V3.1 may also support `set_5V_output_mode`, `set_OD_output_mode`, `output_mode_store` commands.

BLTOUCH_STORE

`BLTOUCH_STORE MODE=<output_mode>`: This stores an output mode in the EEPROM of a BLTouch V3.1 Available output_modes are: `5V`, `OD`

[configfile]

The configfile module is automatically loaded.

SAVE_CONFIG

`SAVE_CONFIG`: This command will overwrite the main printer config file and restart the host software. This command is used in conjunction with other calibration commands to store the results of calibration tests.

[delayed_gcode]

The following command is enabled if a delayed_gcode config section has been enabled (also see the template guide).

UPDATE_DELAYED_GCODE

`UPDATE_DELAYED_GCODE [ID=<name>] [DURATION=<seconds>]`: Updates the delay duration for the identified [delayed_gcode] and starts the timer for gcode execution. A value of 0 will cancel a pending delayed gcode from executing.

[delta_calibrate]

The following commands are available when the delta_calibrate config section is enabled (also see the delta calibrate guide).

DELTA_CALIBRATE

`DELTA_CALIBRATE [METHOD=manual] [<probe_parameter>=<value>]`: This command will probe seven points on the bed and recommend updated endstop positions, tower angles, and radius. See the PROBE command for details on the optional probe parameters. If METHOD=manual is specified then the manual probing tool is activated - see the MANUAL_PROBE command above for details on the additional commands available while this tool is active.

DELTA_ANALYZE

`DELTA_ANALYZE`: This command is used during enhanced delta calibration. See Delta Calibrate for details.

[display]

The following command is available when a display config section is enabled.

SET_DISPLAY_GROUP

`SET_DISPLAY_GROUP [DISPLAY=<display>] GROUP=<group>`: Set the active display group of an lcd display. This allows to define multiple display data groups in the config, e.g. `[display_data <group> <elementname>]` and switch between them using this extended gcode command. If DISPLAY is not specified it defaults to "display" (the primary display).

[display_status]

The display_status module is automatically loaded if a display config section is enabled. It provides the following standard G-Code commands:

- Display Message: `M117 <message>`
- Set build percentage: `M73 P<percent>`

Also provided is the following extended G-Code command:

- `SET_DISPLAY_TEXT MSG=<message>`: Performs the equivalent of M117, setting the supplied `MSG` as the current display message. If `MSG` is omitted the display will be cleared.

[dual_carriage]

The following command is available when the dual_carriage config section is enabled.

SET_DUAL_CARRIAGE

`SET_DUAL_CARRIAGE CARRIAGE=[0|1]`: This command will set the active carriage. It is typically invoked from the activate_gcode and deactivate_gcode fields in a multiple extruder configuration.

[endstop_phase]

The following commands are available when an endstop_phase config section is enabled (also see the endstop phase guide).

ENDSTOP_PHASE_CALIBRATE

`ENDSTOP_PHASE_CALIBRATE [STEPPER=<config_name>]`: If no STEPPER parameter is provided then this command will reports statistics on endstop stepper phases during past homing operations. When a STEPPER parameter is provided it arranges for the given endstop phase setting to be written to the config file (in conjunction with the SAVE_CONFIG command).

[exclude_object]

The following commands are available when an exclude_object config section is enabled (also see the exclude object guide):

`EXCLUDE_OBJECT`

`EXCLUDE_OBJECT [NAME=object_name] [CURRENT=1] [RESET=1]`: With no parameters, this will return a list of all currently excluded objects.

When the NAME parameter is given, the named object will be excluded from printing.

When the CURRENT parameter is given, the current object will be excluded from printing.

When the RESET parameter is given, the list of excluded objects will be cleared. Additionally including NAME will only reset the named object. This **can** cause print failures, if layers were already skipped.

EXCLUDE_OBJECT_DEFINE

`EXCLUDE_OBJECT_DEFINE [NAME=object_name [CENTER=X,Y] [POLYGON=[[x,y],...]] [RESET=1] [JSON=1]`: Provides a summary of an object in the file.

With no parameters provided, this will list the defined objects known to Klipper. Returns a list of strings, unless the JSON parameter is given, when it will return object details in json format.

When the NAME parameter is included, this defines an object to be excluded.

- NAME: This parameter is required. It is the identifier used by other commands in this module.
- CENTER: An X,Y coordinate for the object.
- POLYGON: An array of X,Y coordinates that provide an outline for the object.

When the RESET parameter is provided, all defined objects will be cleared, and the [exclude_object] module will be reset.

EXCLUDE_OBJECT_START

`EXCLUDE_OBJECT_START NAME=object_name`: This command takes a NAME parameter and denotes the start of the gcode for an object on the current layer.

EXCLUDE_OBJECT_END

`EXCLUDE_OBJECT_END [NAME=object_name]`: Denotes the end of the object's gcode for the layer. It is paired with EXCLUDE_OBJECT_START. A NAME parameter is optional, and will only warn when the provided name does not match the current object.

[extruder]

The following commands are available if an extruder config section is enabled:

ACTIVATE_EXTRUDER

`ACTIVATE_EXTRUDER EXTRUDER=<config_name>`: In a printer with multiple extruder config sections, this command changes the active hotend.

SET_PRESSURE_ADVANCE

`SET_PRESSURE_ADVANCE [EXTRUDER=<config_name>] [ADVANCE=<pressure_advance>] [SMOOTH_TIME=<pressure_advance_smooth_time>]`: Set pressure advance parameters of an extruder stepper (as defined in an extruder or extruder_stepper config section). If EXTRUDER is not specified, it defaults to the stepper defined in the active hotend.

SET_EXTRUDER_ROTATION_DISTANCE

`SET_EXTRUDER_ROTATION_DISTANCE EXTRUDER=<config_name> [DISTANCE=<distance>]`: Set a new value for the provided extruder stepper's "rotation distance" (as defined in an extruder or extruder_stepper config section). If the rotation distance is a negative number then the stepper motion will be inverted (relative to the stepper direction specified in the config file). Changed settings are not retained on Klipper reset. Use with caution as small changes can result in excessive pressure between extruder and hotend. Do proper calibration with filament before use. If 'DISTANCE' value is not provided then this command will return the current rotation distance.

SYNC_EXTRUDER_MOTION

`SYNC_EXTRUDER_MOTION EXTRUDER=<name> MOTION_QUEUE=<name>`: This command will cause the stepper specified by EXTRUDER (as defined in an extruder or extruder_stepper config section) to become synchronized to the movement of an extruder specified by MOTION_QUEUE (as defined in an extruder config section). If MOTION_QUEUE is an empty string then the stepper will be desynchronized from all extruder movement.

SET_EXTRUDER_STEP_DISTANCE

This command is deprecated and will be removed in the near future.

SYNC_STEPPER_TO_EXTRUDER

This command is deprecated and will be removed in the near future.

[fan_generic]

The following command is available when a fan_generic config section is enabled.

SET_FAN_SPEED

`SET_FAN_SPEED FAN=config_name SPEED=<speed>` This command sets the speed of a fan. "speed" must be between 0.0 and 1.0.

[filament_switch_sensor]

The following command is available when a filament_switch_sensor or filament_motion_sensor config section is enabled.

QUERY_FILAMENT_SENSOR

`QUERY_FILAMENT_SENSOR SENSOR=<sensor_name>`: Queries the current status of the filament sensor. The data displayed on the terminal will depend on the sensor type defined in the configuration.

SET_FILAMENT_SENSOR

`SET_FILAMENT_SENSOR SENSOR=<sensor_name> ENABLE=[0|1]`: Sets the filament sensor on/off. If ENABLE is set to 0, the filament sensor will be disabled, if set to 1 it is enabled.

[firmware_retraction]

The following standard G-Code commands are available when the firmware_retraction config section is enabled. These commands allow you to utilize the firmware retraction feature available in many slicers, to reduce stringing during non-extrusion moves from one part of the print to another. Appropriately configuring pressure advance reduces the length of retraction required.

- `G10`: Retracts the extruder using the currently configured parameters.
- `G11`: Unretracts the extruder using the currently configured parameters.

The following additional commands are also available.

SET_RETRACTION

`SET_RETRACTION [RETRACT_LENGTH=<mm>] [RETRACT_SPEED=<mm/s>] [UNRETRACT_EXTRA_LENGTH=<mm>] [UNRETRACT_SPEED=<mm/s>]`: Adjust the parameters used by firmware retraction. RETRACT_LENGTH determines the length of filament to retract and unretract. The speed of retraction is adjusted via RETRACT_SPEED, and is typically set relatively high. The speed of unretraction is adjusted via UNRETRACT_SPEED, and is not particularly critical, although often lower than RETRACT_SPEED. In some cases it is useful to add a small amount of additional length on unretraction, and this is set via UNRETRACT_EXTRA_LENGTH. SET_RETRACTION is commonly set as part of slicer per-filament configuration, as different filaments require different parameter settings.

GET_RETRACTION

`GET_RETRACTION`: Queries the current parameters used by firmware retraction and displays them on the terminal.

[force_move]

The force_move module is automatically loaded, however some commands require setting `enable_force_move` in the printer config.

STEPPER_BUZZ

`STEPPER_BUZZ STEPPER=<config_name>`: Move the given stepper forward one mm and then backward one mm, repeated 10 times. This is a diagnostic tool to help verify stepper connectivity.

FORCE_MOVE

`FORCE_MOVE STEPPER=<config_name> DISTANCE=<value> VELOCITY=<value> [ACCEL=<value>]`: This command will forcibly move the given stepper the given distance (in mm) at the given constant velocity (in mm/s). If ACCEL is specified and is greater than zero, then the given acceleration (in mm/s^2) will be used; otherwise no acceleration is performed. No boundary checks are performed; no kinematic updates are made; other parallel steppers on an axis will not be moved. Use caution as an incorrect command could cause damage! Using this command will almost certainly place the low-level kinematics in an incorrect state; issue a G28 afterwards to reset the kinematics. This command is intended for low-level diagnostics and debugging.

SET_KINEMATIC_POSITION

`SET_KINEMATIC_POSITION [X=<value>] [Y=<value>] [Z=<value>]`: Force the low-level kinematic code to believe the toolhead is at the given cartesian position. This is a diagnostic and debugging command; use SET_GCODE_OFFSET and/or G92 for regular axis transformations. If an axis is not specified then it will default to the position that the head was last commanded to. Setting an incorrect or invalid position may lead to internal software errors. This command may invalidate future boundary checks; issue a G28 afterwards to reset the kinematics.

[gcode]

The gcode module is automatically loaded.

RESTART

`RESTART`: This will cause the host software to reload its config and perform an internal reset. This command will not clear error state from the micro-controller (see FIRMWARE_RESTART) nor will it load new software (see the FAQ).

FIRMWARE_RESTART

`FIRMWARE_RESTART`: This is similar to a RESTART command, but it also clears any error state from the micro-controller.

STATUS

`STATUS`: Report the Klipper host software status.

HELP

`HELP`: Report the list of available extended G-Code commands.

[gcode_arcs]

The following standard G-Code commands are available if a gcode_arcs config section is enabled:

- Arc Move Clockwise (G2), Arc Move Counter-clockwise (G3): `G2|G3 [X<pos>] [Y<pos>] [Z<pos>] [E<pos>] [F<speed>] I<value> J<value>|I<value> K<value>|J<value> K<value>`
- Arc Plane Select: G17 (XY plane), G18 (XZ plane), G19 (YZ plane)

[gcode_macro]

The following command is available when a gcode_macro config section is enabled (also see the command templates guide).

SET_GCODE_VARIABLE

`SET_GCODE_VARIABLE MACRO=<macro_name> VARIABLE=<name> VALUE=<value>`: This command allows one to change the value of a gcode_macro variable at run-time. The provided VALUE is parsed as a

Python literal.

[gcode_move]

The gcode_move module is automatically loaded.

GET_POSITION

`GET_POSITION`: Return information on the current location of the toolhead. See the developer documentation of GET_POSITION output for more information.

SET_GCODE_OFFSET

`SET_GCODE_OFFSET [X=<pos>|X_ADJUST=<adjust>] [Y=<pos>|Y_ADJUST=<adjust>] [Z=<pos>|Z_ADJUST=<adjust>] [MOVE=1 [MOVE_SPEED=<speed>]]`: Set a positional offset to apply to future G-Code commands. This is commonly used to virtually change the Z bed offset or to set nozzle XY offsets when switching extruders. For example, if "SET_GCODE_OFFSET Z=0.2" is sent, then future G-Code moves will have 0.2mm added to their Z height. If the X_ADJUST style parameters are used, then the adjustment will be added to any existing offset (eg, "SET_GCODE_OFFSET Z=-0.2" followed by "SET_GCODE_OFFSET Z_ADJUST=0.3" would result in a total Z offset of 0.1). If "MOVE=1" is specified then a toolhead move will be issued to apply the given offset (otherwise the offset will take effect on the next absolute G-Code move that specifies the given axis). If "MOVE_SPEED" is specified then the toolhead move will be performed with the given speed (in mm/s); otherwise the toolhead move will use the last specified G-Code speed.

SAVE_GCODE_STATE

`SAVE_GCODE_STATE [NAME=<state_name>]`: Save the current g-code coordinate parsing state. Saving and restoring the g-code state is useful in scripts and macros. This command saves the current g-code absolute coordinate mode (G90/G91), absolute extrude mode (M82/M83), origin (G92), offset (SET_GCODE_OFFSET), speed override (M220), extruder override (M221), move speed, current XYZ position, and relative extruder "E" position. If NAME is provided it allows one to name the saved state to the given string. If NAME is not provided it defaults to "default".

RESTORE_GCODE_STATE

`RESTORE_GCODE_STATE [NAME=<state_name>] [MOVE=1 [MOVE_SPEED=<speed>]]`: Restore a state previously saved via SAVE_GCODE_STATE. If "MOVE=1" is specified then a toolhead move will be issued to move back to the previous XYZ position. If "MOVE_SPEED" is specified then the toolhead move will be performed with the given speed (in mm/s); otherwise the toolhead move will use the restored g-code speed.

[hall_filament_width_sensor]

The following commands are available when the tsl1401cl filament width sensor config section or hall filament width sensor config section is enabled (also see TSLI401CL Filament Width Sensor and Hall Filament Width Sensor):

QUERY_FILAMENT_WIDTH

`QUERY_FILAMENT_WIDTH`: Return the current measured filament width.

RESET_FILAMENT_WIDTH_SENSOR

`RESET_FILAMENT_WIDTH_SENSOR`: Clear all sensor readings. Helpful after filament change.

DISABLE_FILAMENT_WIDTH_SENSOR

`DISABLE_FILAMENT_WIDTH_SENSOR`: Turn off the filament width sensor and stop using it for flow control.

ENABLE_FILAMENT_WIDTH_SENSOR

`ENABLE_FILAMENT_WIDTH_SENSOR`: Turn on the filament width sensor and start using it for flow control.

QUERY_RAW_FILAMENT_WIDTH

`QUERY_RAW_FILAMENT_WIDTH`: Return the current ADC channel readings and RAW sensor value for calibration points.

ENABLE_FILAMENT_WIDTH_LOG

`ENABLE_FILAMENT_WIDTH_LOG`: Turn on diameter logging.

DISABLE_FILAMENT_WIDTH_LOG

`DISABLE_FILAMENT_WIDTH_LOG`: Turn off diameter logging.

[heaters]

The heaters module is automatically loaded if a heater is defined in the config file.

TURN_OFF_HEATERS

`TURN_OFF_HEATERS`: Turn off all heaters.

TEMPERATURE_WAIT

`TEMPERATURE_WAIT SENSOR=<config_name> [MINIMUM=<target>] [MAXIMUM=<target>]`: Wait until the given temperature sensor is at or above the supplied MINIMUM and/or at or below the supplied MAXIMUM.

SET_HEATER_TEMPERATURE

`SET_HEATER_TEMPERATURE HEATER=<heater_name> [TARGET=<target_temperature>]`: Sets the target temperature for a heater. If a target temperature is not supplied, the target is 0.

[idle_timeout]

The idle_timeout module is automatically loaded.

SET_IDLE_TIMEOUT

`SET_IDLE_TIMEOUT [TIMEOUT=<timeout>]`: Allows the user to set the idle timeout (in seconds).

[input_shaper]

The following command is enabled if an input_shaper config section has been enabled (also see the resonance compensation guide).

SET_INPUT_SHAPER

`SET_INPUT_SHAPER [SHAPER_FREQ_X=<shaper_freq_x>] [SHAPER_FREQ_Y=<shaper_freq_y>] [DAMPING_RATIO_X=<damping_ratio_x>] [DAMPING_RATIO_Y=<damping_ratio_y>] [SHAPER_TYPE=<shaper>] [SHAPER_TYPE_X=<shaper_type_x>] [SHAPER_TYPE_Y=<shaper_type_y>]`: Modify input shaper parameters. Note that SHAPER_TYPE parameter resets input shaper for both X and Y axes even if different shaper types have been configured in [input_shaper] section. SHAPER_TYPE cannot be used together with either of SHAPER_TYPE_X and SHAPER_TYPE_Y parameters. See config reference for more details on each of these parameters.

[manual_probe]

The manual_probe module is automatically loaded.

MANUAL_PROBE

`MANUAL_PROBE [SPEED=<speed>]`: Run a helper script useful for measuring the height of the nozzle at a given location. If SPEED is specified, it sets the speed of TESTZ commands (the default is 5mm/s). During a manual probe, the following additional commands are available:

- `ACCEPT`: This command accepts the current Z position and concludes the manual probing tool.
- `ABORT`: This command terminates the manual probing tool.
- `TESTZ Z=<value>`: This command moves the nozzle up or down by the amount specified in "value". For example, `TESTZ Z=-.1` would move the nozzle down .1mm while `TESTZ Z=.1` would move the nozzle up .1mm. The value may also be `+`, `-`, `++`, or `--` to move the nozzle up or down an amount relative to previous attempts.

Z_ENDSTOP_CALIBRATE

`Z_ENDSTOP_CALIBRATE [SPEED=<speed>]`: Run a helper script useful for calibrating a Z position_endstop config setting. See the MANUAL_PROBE command for details on the parameters and the additional commands available while the tool is active.

Z_OFFSET_APPLY_ENDSTOP

`Z_OFFSET_APPLY_ENDSTOP`: Take the current Z Gcode offset (aka, babystepping), and subtract it from the stepper_z endstop_position. This acts to take a frequently used babystepping value, and "make it permanent". Requires a `SAVE_CONFIG` to take effect.

[manual_stepper]

The following command is available when a manual_stepper config section is enabled.

MANUAL_STEPPER

`MANUAL_STEPPER STEPPER=config_name [ENABLE=[0|1]] [SET_POSITION=<pos>] [SPEED=<speed>] [ACCEL=<accel>] [MOVE=<pos> [STOP_ON_ENDSTOP=[1|2|-1|-2]] [SYNC=0]`: This command will alter the state of the stepper. Use the ENABLE parameter to enable/disable the stepper. Use the SET_POSITION parameter to force the stepper to think it is at the given position. Use the MOVE parameter to request a movement to the given position. If SPEED and/or ACCEL is specified then the given values will be used instead of the defaults specified in the config file. If an ACCEL of zero is specified then no acceleration will be performed. If STOP_ON_ENDSTOP=1 is specified then the move will end early should the endstop report as triggered (use STOP_ON_ENDSTOP=2 to complete the move without error even if the endstop does not trigger, use -1 or -2 to stop when the endstop reports not triggered). Normally future G-Code commands will be scheduled to run after the stepper move completes, however if a manual stepper move uses SYNC=0 then future G-Code movement commands may run in parallel with the stepper movement.

[mcp4018]

The following command is available when a mcp4018 config section is enabled.

SET_DIGIPOT

`SET_DIGIPOT DIGIPOT=config_name WIPER=<value>`: This command will change the current value of the digipot. This value should typically be between 0.0 and 1.0, unless a 'scale' is defined in the config. When 'scale' is defined, then this value should be between 0.0 and 'scale'.

[led]

The following command is available when any of the led config sections are enabled.

SET_LED

`SET_LED LED=<config_name> RED=<value> GREEN=<value> BLUE=<value> WHITE=<value> [INDEX=<index>] [TRANSMIT=0] [SYNC=1]`: This sets the LED output. Each color `<value>` must be between 0.0 and 1.0. The WHITE option is only valid on RGBW LEDs. If the LED supports multiple chips in a daisy-chain then one may specify INDEX to alter the color of just the given chip (1 for the first chip, 2 for the second, etc.). If INDEX is not provided then all LEDs in the daisy-chain will be set to the provided color. If TRANSMIT=0 is specified then the color change will only be made on the next SET_LED command that does not specify TRANSMIT=0; this may be useful in combination with the INDEX parameter to batch multiple updates in a daisy-chain. By default, the SET_LED command will sync it's changes with other ongoing gcode commands. This can lead to undesirable behavior if LEDs are being set while the printer is not printing as it will reset the idle timeout. If careful timing is not needed, the optional SYNC=0 parameter can be specified to apply the changes without resetting the idle timeout.

SET_LED_TEMPLATE

`SET_LED_TEMPLATE LED=<led_name> TEMPLATE=<template_name> [<param_x>=<literal>] [INDEX=<index>]`: Assign a display_template to a given LED. For example, if one defined a

`[display_template my_led_template]` config section then one could assign `TEMPLATE=my_led_template` here. The display_template should produce a comma separated string containing four floating point numbers corresponding to red, green, blue, and white color settings. The template will be continuously evaluated and the LED will be automatically set to the resulting colors. One may set display_template parameters to use during template evaluation (parameters will be parsed as Python literals). If INDEX is not specified then all chips in the LED's daisy-chain will be set to the template, otherwise only the chip with the given index will be updated. If TEMPLATE is an empty string then this command will clear any previous template assigned to the LED (one can then use `SET_LED` commands to manage the LED's color settings).

[output_pin]

The following command is available when an output_pin config section is enabled.

SET_PIN

`SET_PIN PIN=config_name VALUE=<value> CYCLE_TIME=<cycle_time>`: Note - hardware PWM does not currently support the CYCLE_TIME parameter and will use the cycle time defined in the config.

[palette2]

The following commands are available when the palette2 config section is enabled.

Palette prints work by embedding special OCodes (Omega Codes) in the GCode file:

- `01`...`032`: These codes are read from the GCode stream and processed by this module and passed to the Palette 2 device.

The following additional commands are also available.

PALETTE_CONNECT

`PALETTE_CONNECT`: This command initializes the connection with the Palette 2.

PALETTE_DISCONNECT

`PALETTE_DISCONNECT`: This command disconnects from the Palette 2.

PALETTE_CLEAR

`PALETTE_CLEAR`: This command instructs the Palette 2 to clear all of the input and output paths of filament.

PALETTE_CUT

`PALETTE_CUT`: This command instructs the Palette 2 to cut the filament currently loaded in the splice core.

PALETTE_SMART_LOAD

`PALETTE_SMART_LOAD`: This command start the smart load sequence on the Palette 2. Filament is loaded automatically by extruding it the distance calibrated on the device for the printer, and instructs the Palette 2 once the loading has been completed. This command is the same as pressing **Smart Load** directly on the Palette 2 screen after the filament load is complete.

[pid_calibrate]

The pid_calibrate module is automatically loaded if a heater is defined in the config file.

PID_CALIBRATE

`PID_CALIBRATE HEATER=<config_name> TARGET=<temperature> [WRITE_FILE=1]`: Perform a PID calibration test. The specified heater will be enabled until the specified target temperature is reached, and then the heater will be turned off and on for several cycles. If the WRITE_FILE parameter is enabled, then the file /tmp/heattest.txt will be created with a log of all temperature samples taken during the test.

[pause_resume]

The following commands are available when the pause_resume config section is enabled:

PAUSE

`PAUSE`: Pauses the current print. The current position is captured for restoration upon resume.

RESUME

`RESUME [VELOCITY=<value>]`: Resumes the print from a pause, first restoring the previously captured position. The VELOCITY parameter determines the speed at which the tool should return to the original captured position.

CLEAR_PAUSE

`CLEAR_PAUSE`: Clears the current paused state without resuming the print. This is useful if one decides to cancel a print after a PAUSE. It is recommended to add this to your start gcode to make sure the paused state is fresh for each print.

CANCEL_PRINT

`CANCEL_PRINT`: Cancels the current print.

[print_stats]

The print_stats module is automatically loaded.

SET_PRINT_STATS_INFO

`SET_PRINT_STATS_INFO [TOTAL_LAYER=<total_layer_count>] [CURRENT_LAYER=<current_layer>]`: Pass slicer info like layer act and total to Klipper. Add `SET_PRINT_STATS_INFO [TOTAL_LAYER=<total_layer_count>]` to your slicer start gcode section and

`SET_PRINT_STATS_INFO [CURRENT_LAYER= <current_layer>]` at the layer change gcode section to pass layer information from your slicer to Klipper.

[probe]

The following commands are available when a probe config section or bltouch config section is enabled (also see the probe calibrate guide).

PROBE

`PROBE [PROBE_SPEED=<mm/s>] [LIFT_SPEED=<mm/s>] [SAMPLES=<count>] [SAMPLE_RETRACT_DIST=<mm>] [SAMPLES_TOLERANCE=<mm>] [SAMPLES_TOLERANCE_RETRIES=<count>] [SAMPLES_RESULT=median|average]`: Move the nozzle downwards until the probe triggers. If any of the optional parameters are provided they override their equivalent setting in the probe config section.

QUERY_PROBE

`QUERY_PROBE`: Report the current status of the probe ("triggered" or "open").

PROBE_ACCURACY

`PROBE_ACCURACY [PROBE_SPEED=<mm/s>] [SAMPLES=<count>] [SAMPLE_RETRACT_DIST=<mm>]`: Calculate the maximum, minimum, average, median, and standard deviation of multiple probe samples. By default, 10 SAMPLES are taken. Otherwise the optional parameters default to their equivalent setting in the probe config section.

PROBE_CALIBRATE

`PROBE_CALIBRATE [SPEED=<speed>] [<probe_parameter>=<value>]`: Run a helper script useful for calibrating the probe's z_offset. See the PROBE command for details on the optional probe parameters. See the MANUAL_PROBE command for details on the SPEED parameter and the additional commands available while the tool is active. Please note, the PROBE_CALIBRATE command uses the speed variable to move in XY direction as well as Z.

Z_OFFSET_APPLY_PROBE

`Z_OFFSET_APPLY_PROBE`: Take the current Z Gcode offset (aka, babystepping), and subtract if from the probe's z_offset. This acts to take a frequently used babystepping value, and "make it permanent". Requires a `SAVE_CONFIG` to take effect.

[query_adc]

The query_adc module is automatically loaded.

QUERY_ADC

`QUERY_ADC [NAME=<config_name>] [PULLUP=<value>]`: Report the last analog value received for a configured analog pin. If NAME is not provided, the list of available adc names are reported. If PULLUP is

provided (as a value in Ohms), the raw analog value along with the equivalent resistance given that pullup is reported.

[query_endstops]

The query_endstops module is automatically loaded. The following standard G-Code commands are currently available, but using them is not recommended:

- Get Endstop Status: `M119` (Use QUERY_ENDSTOPS instead.)

QUERY_ENDSTOPS

`QUERY_ENDSTOPS`: Probe the axis endstops and report if they are "triggered" or in an "open" state. This command is typically used to verify that an endstop is working correctly.

[resonance_tester]

The following commands are available when a resonance_tester config section is enabled (also see the measuring resonances guide).

MEASURE_AXES_NOISE

`MEASURE_AXES_NOISE`: Measures and outputs the noise for all axes of all enabled accelerometer chips.

TEST_RESONANCES

`TEST_RESONANCES AXIS=<axis> OUTPUT=<resonances,raw_data> [NAME=<name>] [FREQ_START=<min_freq>] [FREQ_END=<max_freq>] [HZ_PER_SEC=<hz_per_sec>] [CHIPS=<adxl345_chip_name>] [POINT=x,y,z] [INPUT_SHAPING=[<0:1>]]`: Runs the resonance test in all configured probe points for the requested "axis" and measures the acceleration using the accelerometer chips configured for the respective axis. "axis" can either be X or Y, or specify an arbitrary direction as `AXIS=dx,dy`, where dx and dy are floating point numbers defining a direction vector (e.g. `AXIS=X`, `AXIS=Y`, or `AXIS=1,-1` to define a diagonal direction). Note that `AXIS=dx,dy` and `AXIS=-dx,-dy` is equivalent. `adxl345_chip_name` can be one or more configured adxl345 chip,delimited with comma, for example `CHIPS="adxl345, adxl345 rpi"`. Note that `adxl345` can be omitted from named adxl345 chips. If POINT is specified it will override the point(s) configured in `[resonance_tester]`. If `INPUT_SHAPING=0` or not set(default), disables input shaping for the resonance testing, because it is not valid to run the resonance testing with the input shaper enabled. `OUTPUT` parameter is a comma-separated list of which outputs will be written. If `raw_data` is requested, then the raw accelerometer data is written into a file or a series of files `/tmp/raw_data_<axis>_[<chip_name>_][<point>_]<name>.csv` with (`<point>_` part of the name generated only if more than 1 probe point is configured or POINT is specified). If `resonances` is specified, the frequency response is calculated (across all probe points) and written into `/tmp/resonances_<axis>_<name>.csv` file. If unset, OUTPUT defaults to `resonances`, and NAME defaults to the current time in "YYYYMMDD_HHMMSS" format.

SHAPER_CALIBRATE

`SHAPER_CALIBRATE [AXIS=<axis>] [NAME=<name>] [FREQ_START=<min_freq>] [FREQ_END=<max_freq>] [HZ_PER_SEC=<hz_per_sec>] [MAX_SMOOTHING=<max_smoothing>]`: Similarly to

`TEST_RESONANCES`, runs the resonance test as configured, and tries to find the optimal parameters for the input shaper for the requested axis (or both X and Y axes if `AXIS` parameter is unset). If `MAX_SMOOTHING` is unset, its value is taken from `[resonance_tester]` section, with the default being unset. See the Max smoothing of the measuring resonances guide for more information on the use of this feature. The results of the tuning are printed to the console, and the frequency responses and the different input shapers values are written to a CSV file(s) `/tmp/calibration_data_<axis>_<name>.csv`. Unless specified, NAME defaults to the current time in "YYYYMMDD_HHMMSS" format. Note that the suggested input shaper parameters can be persisted in the config by issuing `SAVE_CONFIG` command.

[respond]

The following standard G-Code commands are available when the respond config section is enabled:

- `M118 <message>`: echo the message prepended with the configured default prefix (or `echo:` if no prefix is configured).

The following additional commands are also available.

RESPOND

- `RESPOND MSG="<message>"`: echo the message prepended with the configured default prefix (or `echo:` if no prefix is configured).
- `RESPOND TYPE=echo MSG="<message>"`: echo the message prepended with `echo:` .
- `RESPOND TYPE=echo_no_space MSG="<message>"`: echo the message prepended with `echo:` without a space between prefix and message, helpful for compatibility with some octoprint plugins that expect very specific formatting.
- `RESPOND TYPE=command MSG="<message>"`: echo the message prepended with `//` . OctoPrint can be configured to respond to these messages (e.g. `RESPOND TYPE=command MSG=action:pause`).
- `RESPOND TYPE=error MSG="<message>"`: echo the message prepended with `!!` .
- `RESPOND PREFIX=<prefix> MSG="<message>"`: echo the message prepended with `<prefix>`. (The `PREFIX` parameter will take priority over the `TYPE` parameter)

[save_variables]

The following command is enabled if a save_variables config section has been enabled.

SAVE_VARIABLE

`SAVE_VARIABLE VARIABLE=<name> VALUE=<value>`: Saves the variable to disk so that it can be used across restarts. All stored variables are loaded into the `printer.save_variables.variables` dict at startup and can be used in gcode macros. The provided VALUE is parsed as a Python literal.

[screws_tilt_adjust]

The following commands are available when the screws_tilt_adjust config section is enabled (also see the manual level guide).

SCREWS_TILT_CALCULATE

`SCREWS_TILT_CALCULATE [DIRECTION=CW|CCW] [MAX_DEVIATION=<value>] [<probe_parameter>=<value>]`: This command will invoke the bed screws adjustment tool. It will command the nozzle to different locations (as defined in the config file) probing the z height and calculate the number of knob turns to adjust the bed level. If DIRECTION is specified, the knob turns will all be in the same direction, clockwise (CW) or counterclockwise (CCW). See the PROBE command for details on the optional probe parameters. IMPORTANT: You MUST always do a G28 before using this command. If MAX_DEVIATION is specified, the command will raise a gcode error if any difference in the screw height relative to the base screw height is greater than the value provided.

[sdcard_loop]

When the sdcard_loop config section is enabled, the following extended commands are available.

SDCARD_LOOP_BEGIN

`SDCARD_LOOP_BEGIN COUNT=<count>`: Begin a looped section in the SD print. A count of 0 indicates that the section should be looped indefinitely.

SDCARD_LOOP_END

`SDCARD_LOOP_END`: End a looped section in the SD print.

SDCARD_LOOP_DESIST

`SDCARD_LOOP_DESIST`: Complete existing loops without further iterations.

[servo]

The following commands are available when a servo config section is enabled.

SET_SERVO

`SET_SERVO SERVO=config_name [ANGLE=<degrees> | WIDTH=<seconds>]`: Set the servo position to the given angle (in degrees) or pulse width (in seconds). Use `WIDTH=0` to disable the servo output.

[skew_correction]

The following commands are available when the skew_correction config section is enabled (also see the Skew Correction guide).

SET_SKEW

`SET_SKEW [XY=<ac_length,bd_length,ad_length>] [XZ=<ac,bd,ad>] [YZ=<ac,bd,ad>] [CLEAR=<0|1>]`: Configures the [skew_correction] module with measurements (in mm) taken from a calibration print. One may enter measurements for any combination of planes, planes not entered will retain their current value. If `CLEAR=1` is entered then all skew correction will be disabled.

GET_CURRENT_SKEW

`GET_CURRENT_SKEW`: Reports the current printer skew for each plane in both radians and degrees. The skew is calculated based on parameters provided via the `SET_SKEW` gcode.

CALC_MEASURED_SKEW

`CALC_MEASURED_SKEW [AC=<ac_length>] [BD=<bd_length>] [AD=<ad_length>]`: Calculates and reports the skew (in radians and degrees) based on a measured print. This can be useful for determining the printer's current skew after correction has been applied. It may also be useful before correction is applied to determine if skew correction is necessary. See Skew Correction for details on skew calibration objects and measurements.

SKEW_PROFILE

`SKEW_PROFILE [LOAD=<name>] [SAVE=<name>] [REMOVE=<name>]`: Profile management for skew_correction. LOAD will restore skew state from the profile matching the supplied name. SAVE will save the current skew state to a profile matching the supplied name. Remove will delete the profile matching the supplied name from persistent memory. Note that after SAVE or REMOVE operations have been run the SAVE_CONFIG gcode must be run to make the changes to persistent memory permanent.

[smart_effector]

Several commands are available when a smart_effector config section is enabled. Be sure to check the official documentation for the Smart Effector on the Duet3D Wiki before changing the Smart Effector parameters. Also check the probe calibration guide.

SET_SMART_EFFECTOR

`SET_SMART_EFFECTOR [SENSITIVITY=<sensitivity>] [ACCEL=<accel>] [RECOVERY_TIME=<time>]`: Set the Smart Effector parameters. When `SENSITIVITY` is specified, the respective value is written to the SmartEffector EEPROM (requires `control_pin` to be provided). Acceptable `<sensitivity>` values are 0..255, the default is 50. Lower values require less nozzle contact force to trigger (but there is a higher risk of false triggering due to vibrations during probing), and higher values reduce false triggering (but require larger contact force to trigger). Since the sensitivity is written to EEPROM, it is preserved after the shutdown, and so it does not need to be configured on every printer startup. `ACCEL` and `RECOVERY_TIME` allow to override the corresponding parameters at run-time, see the config section of Smart Effector for more info on those parameters.

RESET_SMART_EFFECTOR

`RESET_SMART_EFFECTOR`: Resets Smart Effector sensitivity to its factory settings. Requires `control_pin` to be provided in the config section.

[stepper_enable]

The stepper_enable module is automatically loaded.

SET_STEPPER_ENABLE

`SET_STEPPER_ENABLE STEPPER=<config_name> ENABLE=[0|1]`: Enable or disable only the given stepper. This is a diagnostic and debugging tool and must be used with care. Disabling an axis motor does not reset the homing information. Manually moving a disabled stepper may cause the machine to operate the motor outside of safe limits. This can lead to damage to axis components, hot ends, and print surface.

[temperature_fan]

The following command is available when a temperature_fan config section is enabled.

SET_TEMPERATURE_FAN_TARGET

`SET_TEMPERATURE_FAN_TARGET temperature_fan=<temperature_fan_name> [target=<target_temperature>] [min_speed=<min_speed>] [max_speed=<max_speed>]`: Sets the target temperature for a temperature_fan. If a target is not supplied, it is set to the specified temperature in the config file. If speeds are not supplied, no change is applied.

[tmcXXXX]

The following commands are available when any of the tmcXXXX config sections are enabled.

DUMP_TMC

`DUMP_TMC STEPPER=<name>`: This command will read the TMC driver registers and report their values.

INIT_TMC

`INIT_TMC STEPPER=<name>`: This command will initialize the TMC registers. Needed to re-enable the driver if power to the chip is turned off then back on.

SET_TMC_CURRENT

`SET_TMC_CURRENT STEPPER=<name> CURRENT=<amps> HOLDCURRENT=<amps>`: This will adjust the run and hold currents of the TMC driver. (HOLDCURRENT is not applicable to tmc2660 drivers.)

SET_TMC_FIELD

`SET_TMC_FIELD STEPPER=<name> FIELD=<field> VALUE=<value>`: This will alter the value of the specified register field of the TMC driver. This command is intended for low-level diagnostics and debugging only because changing the fields during run-time can lead to undesired and potentially dangerous behavior of your printer. Permanent changes should be made using the printer configuration file instead. No sanity checks are performed for the given values.

[toolhead]

The toolhead module is automatically loaded.

SET_VELOCITY_LIMIT

`SET_VELOCITY_LIMIT [VELOCITY=<value>] [ACCEL=<value>] [ACCEL_TO_DECEL=<value>] [SQUARE_CORNER_VELOCITY=<value>]`: Modify the printer's velocity limits.

[tuning_tower]

The tuning_tower module is automatically loaded.

TUNING_TOWER

`TUNING_TOWER COMMAND=<command> PARAMETER=<name> START=<value> [SKIP=<value>] [FACTOR=<value> [BAND=<value>]] | [STEP_DELTA=<value> STEP_HEIGHT=<value>]`: A tool for tuning a parameter on each Z height during a print. The tool will run the given `COMMAND` with the given `PARAMETER` assigned to a value that varies with `Z` according to a formula. Use `FACTOR` if you will use a ruler or calipers to measure the Z height of the optimum value, or `STEP_DELTA` and `STEP_HEIGHT` if the tuning tower model has bands of discrete values as is common with temperature towers. If `SKIP=<value>` is specified, the tuning process doesn't begin until Z height `<value>` is reached, and below that the value will be set to `START`; in this case, the `z_height` used in the formulas below is actually `max(z - skip, 0)`. There are three possible combinations of options:

- `FACTOR`: The value changes at a rate of `factor` per millimeter. The formula used is: `value = start + factor * z_height`. You can plug the optimum Z height directly into the formula to determine the optimum parameter value.
- `FACTOR` and `BAND`: The value changes at an average rate of `factor` per millimeter, but in discrete bands where the adjustment will only be made every `BAND` millimeters of Z height. The formula used is: `value = start + factor * ((floor(z_height / band) + .5) * band)`.
- `STEP_DELTA` and `STEP_HEIGHT`: The value changes by `STEP_DELTA` every `STEP_HEIGHT` millimeters. The formula used is: `value = start + step_delta * floor(z_height / step_height)`. You can simply count bands or read tuning tower labels to determine the optimum value.

[virtual_sdcard]

Klipper supports the following standard G-Code commands if the virtual_sdcard config section is enabled:

- List SD card: `M20`
- Initialize SD card: `M21`
- Select SD file: `M23 <filename>`
- Start/resume SD print: `M24`
- Pause SD print: `M25`
- Set SD position: `M26 S<offset>`
- Report SD print status: `M27`

In addition, the following extended commands are available when the "virtual_sdcard" config section is enabled.

SDCARD_PRINT_FILE

`SDCARD_PRINT_FILE FILENAME=<filename>`: Load a file and start SD print.

SDCARD_RESET_FILE

`SDCARD_RESET_FILE`: Unload file and clear SD state.

[z_thermal_adjust]

The following commands are available when the z_thermal_adjust config section is enabled.

SET_Z_THERMAL_ADJUST

`SET_Z_THERMAL_ADJUST [ENABLE=<0:1>] [TEMP_COEFF=<value>] [REF_TEMP=<value>]`: Enable or disable the Z thermal adjustment with `ENABLE`. Disabling does not remove any adjustment already applied, but will freeze the current adjustment value - this prevents potentially unsafe downward Z movement. Re-enabling can potentially cause upward tool movement as the adjustment is updated and applied. `TEMP_COEFF` allows run-time tuning of the adjustment temperature coefficient (i.e. the `TEMP_COEFF` config parameter). `TEMP_COEFF` values are not saved to the config. `REF_TEMP` manually overrides the reference temperature typically set during homing (for use in e.g. non-standard homing routines) - will be reset automatically upon homing.

[z_tilt]

The following commands are available when the z_tilt config section is enabled.

Z_TILT_ADJUST

`Z_TILT_ADJUST [<probe_parameter>=<value>]`: This command will probe the points specified in the config and then make independent adjustments to each Z stepper to compensate for tilt. See the PROBE command for details on the optional probe parameters.