Configuration reference

This document is a reference for options available in the Klipper config file.

The descriptions in this document are formatted so that it is possible to cut-and-paste them into a printer config file. See the installation document for information on setting up Klipper and choosing an initial config file.

Micro-controller configuration

Format of micro-controller pin names

Many config options require the name of a micro-controller pin. Klipper uses the hardware names for these pins - for example `PA4`.

Pin names may be preceded by `!` to indicate that a reverse polarity should be used (eg, trigger on low instead of high).

Input pins may be preceded by `^` to indicate that a hardware pull-up resistor should be enabled for the pin. If the micro-controller supports pull-down resistors then an input pin may alternatively be preceded by `~`.

Note, some config sections may "create" additional pins. Where this occurs, the config section defining the pins must be listed in the config file before any sections using those pins.

[mcu]

Configuration of the primary micro-controller.

```
[mcu]
serial:
#   The serial port to connect to the MCU. If unsure (or if it
#   changes) see the "Where's my serial port?" section of the FAQ.
#   This parameter must be provided when using a serial port.
#baud: 250000
#   The baud rate to use. The default is 250000.
#canbus_uuid:
#   If using a device connected to a CAN bus then this sets the unique
#   chip identifier to connect to. This value must be provided when using
#   CAN bus for communication.
#canbus_interface:
#   If using a device connected to a CAN bus then this sets the CAN
#   network interface to use. The default is 'can0'.
#restart_method:
#   This controls the mechanism the host will use to reset the
#   micro-controller. The choices are 'arduino', 'cheetah', 'rpi_usb',
#   and 'command'. The 'arduino' method (toggle DTR) is common on
#   Arduino boards and clones. The 'cheetah' method is a special
#   method needed for some Fysetc Cheetah boards. The 'rpi_usb' method
#   is useful on Raspberry Pi boards with micro-controllers powered
#   over USB - it briefly disables power to all USB ports to
#   accomplish a micro-controller reset. The 'command' method involves
#   sending a Klipper command to the micro-controller so that it can
#   reset itself. The default is 'arduino' if the micro-controller
#   communicates over a serial port, 'command' otherwise.
```

[mcu my_extra_mcu]

Additional micro-controllers (one may define any number of sections with an "mcu" prefix). Additional micro-controllers introduce additional pins that may be configured as heaters, steppers, fans, etc.. For example, if an "[mcu extra_mcu]" section is introduced, then pins such as "extra_mcu:ar9" may then be used elsewhere in the config (where "ar9" is a hardware pin name or alias name on the given mcu).

```
[mcu my_extra_mcu]
# See the "mcu" section for configuration parameters.
```

Common kinematic settings

[printer]

The printer section controls high level printer settings.

```
[printer]
kinematics:
#   The type of printer in use. This option may be one of: cartesian,
#   corexy, corexz, hybrid_corexy, hybrid_corexz, rotary_delta, delta,
#   deltesian, polar, winch, or none. This parameter must be specified.
max_velocity:
#   Maximum velocity (in mm/s) of the toolhead (relative to the
#   print). This parameter must be specified.
max_accel:
#   Maximum acceleration (in mm/s^2) of the toolhead (relative to the
#   print). This parameter must be specified.
#max_accel_to_decel:
#   A pseudo acceleration (in mm/s^2) controlling how fast the
#   toolhead may go from acceleration to deceleration. It is used to
#   reduce the top speed of short zig-zag moves (and thus reduce
#   printer vibration from these moves). The default is half of
#   max_accel.
#square_corner_velocity: 5.0
#   The maximum velocity (in mm/s) that the toolhead may travel a 90
#   degree corner at. A non-zero value can reduce changes in extruder
#   flow rates by enabling instantaneous velocity changes of the
#   toolhead during cornering. This value configures the internal
#   centripetal velocity cornering algorithm; corners with angles
#   larger than 90 degrees will have a higher cornering velocity while
#   corners with angles less than 90 degrees will have a lower
#   cornering velocity. If this is set to zero then the toolhead will
#   decelerate to zero at each corner. The default is 5mm/s.
```

[stepper]

Stepper motor definitions. Different printer types (as specified by the "kinematics" option in the [printer] config section) require different names for the stepper (eg, `stepper_x` vs `stepper_a`). Below are common stepper definitions.

See the rotation distance document for information on calculating the `rotation_distance` parameter. See the Multi-MCU homing document for information on homing using multiple micro-controllers.

```
[stepper_x]
step_pin:
#   Step GPIO pin (triggered high). This parameter must be provided.
dir_pin:
#   Direction GPIO pin (high indicates positive direction). This
#   parameter must be provided.
enable_pin:
#   Enable pin (default is enable high; use ! to indicate enable
#   low). If this parameter is not provided then the stepper motor
#   driver must always be enabled.
rotation_distance:
#   Distance (in mm) that the axis travels with one full rotation of
#   the stepper motor (or final gear if gear_ratio is specified).
#   This parameter must be provided.
microsteps:
#   The number of microsteps the stepper motor driver uses. This
#   parameter must be provided.
#full_steps_per_rotation: 200
#   The number of full steps for one rotation of the stepper motor.
#   Set this to 200 for a 1.8 degree stepper motor or set to 400 for a
#   0.9 degree motor. The default is 200.
#gear_ratio:
#   The gear ratio if the stepper motor is connected to the axis via a
#   gearbox. For example, one may specify "5:1" if a 5 to 1 gearbox is
#   in use. If the axis has multiple gearboxes one may specify a comma
#   separated list of gear ratios (for example, "57:11, 2:1"). If a
#   gear_ratio is specified then rotation_distance specifies the
#   distance the axis travels for one full rotation of the final gear.
#   The default is to not use a gear ratio.
#step_pulse_duration:
#   The minimum time between the step pulse signal edge and the
#   following "unstep" signal edge. This is also used to set the
#   minimum time between a step pulse and a direction change signal.
#   The default is 0.000000100 (100ns) for TMC steppers that are
#   configured in UART or SPI mode, and the default is 0.000002 (which
#   is 2us) for all other steppers.
endstop_pin:
#   Endstop switch detection pin. If this endstop pin is on a
#   different mcu than the stepper motor then it enables "multi-mcu
#   homing". This parameter must be provided for the X, Y, and Z
#   steppers on cartesian style printers.
#position_min: 0
```

```
#   Minimum valid distance (in mm) the user may command the stepper to
#   move to.  The default is 0mm.
position_endstop:
#   Location of the endstop (in mm). This parameter must be provided
#   for the X, Y, and Z steppers on cartesian style printers.
position_max:
#   Maximum valid distance (in mm) the user may command the stepper to
#   move to. This parameter must be provided for the X, Y, and Z
#   steppers on cartesian style printers.
#homing_speed: 5.0
#   Maximum velocity (in mm/s) of the stepper when homing. The default
#   is 5mm/s.
#homing_retract_dist: 5.0
#   Distance to backoff (in mm) before homing a second time during
#   homing. Set this to zero to disable the second home. The default
#   is 5mm.
#homing_retract_speed:
#   Speed to use on the retract move after homing in case this should
#   be different from the homing speed, which is the default for this
#   parameter
#second_homing_speed:
#   Velocity (in mm/s) of the stepper when performing the second home.
#   The default is homing_speed/2.
#homing_positive_dir:
#   If true, homing will cause the stepper to move in a positive
#   direction (away from zero); if false, home towards zero. It is
#   better to use the default than to specify this parameter. The
#   default is true if position_endstop is near position_max and false
#   if near position_min.
```

Cartesian Kinematics

See example-cartesian.cfg for an example cartesian kinematics config file.

Only parameters specific to cartesian printers are described here - see common kinematic settings for available parameters.

```
[printer]
kinematics: cartesian
max_z_velocity:
#   This sets the maximum velocity (in mm/s) of movement along the z
#   axis. This setting can be used to restrict the maximum speed of
#   the z stepper motor. The default is to use max_velocity for
#   max_z_velocity.
max_z_accel:
#   This sets the maximum acceleration (in mm/s^2) of movement along
#   the z axis. It limits the acceleration of the z stepper motor. The
#   default is to use max_accel for max_z_accel.

# The stepper_x section is used to describe the stepper controlling
# the X axis in a cartesian robot.
```

```
[stepper_x]

# The stepper_y section is used to describe the stepper controlling
# the Y axis in a cartesian robot.
[stepper_y]

# The stepper_z section is used to describe the stepper controlling
# the Z axis in a cartesian robot.
[stepper_z]
```

Linear Delta Kinematics

See example-delta.cfg for an example linear delta kinematics config file. See the delta calibrate guide for information on calibration.

Only parameters specific to linear delta printers are described here - see common kinematic settings for available parameters.

```
[printer]
kinematics: delta
max_z_velocity:
#   For delta printers this limits the maximum velocity (in mm/s) of
#   moves with z axis movement. This setting can be used to reduce the
#   maximum speed of up/down moves (which require a higher step rate
#   than other moves on a delta printer). The default is to use
#   max_velocity for max_z_velocity.
#max_z_accel:
#   This sets the maximum acceleration (in mm/s^2) of movement along
#   the z axis. Setting this may be useful if the printer can reach higher
#   acceleration on XY moves than Z moves (eg, when using input shaper).
#   The default is to use max_accel for max_z_accel.
#minimum_z_position: 0
#   The minimum Z position that the user may command the head to move
#   to. The default is 0.
delta_radius:
#   Radius (in mm) of the horizontal circle formed by the three linear
#   axis towers. This parameter may also be calculated as:
#    delta_radius = smooth_rod_offset - effector_offset - carriage_offset
#   This parameter must be provided.
#print_radius:
#   The radius (in mm) of valid toolhead XY coordinates. One may use
#   this setting to customize the range checking of toolhead moves. If
#   a large value is specified here then it may be possible to command
#   the toolhead into a collision with a tower. The default is to use
#   delta_radius for print_radius (which would normally prevent a
#   tower collision).

# The stepper_a section describes the stepper controlling the front
# left tower (at 210 degrees). This section also controls the homing
# parameters (homing_speed, homing_retract_dist) for all towers.
[stepper_a]
```

```
position_endstop:
#   Distance (in mm) between the nozzle and the bed when the nozzle is
#   in the center of the build area and the endstop triggers. This
#   parameter must be provided for stepper_a; for stepper_b and
#   stepper_c this parameter defaults to the value specified for
#   stepper_a.
arm_length:
#   Length (in mm) of the diagonal rod that connects this tower to the
#   print head. This parameter must be provided for stepper_a; for
#   stepper_b and stepper_c this parameter defaults to the value
#   specified for stepper_a.
#angle:
#   This option specifies the angle (in degrees) that the tower is
#   at. The default is 210 for stepper_a, 330 for stepper_b, and 90
#   for stepper_c.

# The stepper_b section describes the stepper controlling the front
# right tower (at 330 degrees).
[stepper_b]

# The stepper_c section describes the stepper controlling the rear
# tower (at 90 degrees).
[stepper_c]

# The delta_calibrate section enables a DELTA_CALIBRATE extended
# g-code command that can calibrate the tower endstop positions and
# angles.
[delta_calibrate]
radius:
#   Radius (in mm) of the area that may be probed. This is the radius
#   of nozzle coordinates to be probed; if using an automatic probe
#   with an XY offset then choose a radius small enough so that the
#   probe always fits over the bed. This parameter must be provided.
#speed: 50
#   The speed (in mm/s) of non-probing moves during the calibration.
#   The default is 50.
#horizontal_move_z: 5
#   The height (in mm) that the head should be commanded to move to
#   just prior to starting a probe operation. The default is 5.
```

Deltesian Kinematics

See example-deltesian.cfg for an example deltesian kinematics config file.

Only parameters specific to deltesian printers are described here - see common kinematic settings for available parameters.

```
[printer]
kinematics: deltesian
max_z_velocity:
#   For deltesian printers, this limits the maximum velocity (in mm/s) of
#   moves with z axis movement. This setting can be used to reduce the
#   maximum speed of up/down moves (which require a higher step rate
#   than other moves on a deltesian printer). The default is to use
#   max_velocity for max_z_velocity.
#max_z_accel:
#   This sets the maximum acceleration (in mm/s^2) of movement along
#   the z axis. Setting this may be useful if the printer can reach higher
#   acceleration on XY moves than Z moves (eg, when using input shaper).
#   The default is to use max_accel for max_z_accel.
#minimum_z_position: 0
#   The minimum Z position that the user may command the head to move
#   to. The default is 0.
#min_angle: 5
#   This represents the minimum angle (in degrees) relative to horizontal
#   that the deltesian arms are allowed to achieve. This parameter is
#   intended to restrict the arms from becomming completely horizontal,
#   which would risk accidental inversion of the XZ axis. The default is
5.
#print_width:
#   The distance (in mm) of valid toolhead X coordinates. One may use
#   this setting to customize the range checking of toolhead moves. If
#   a large value is specified here then it may be possible to command
#   the toolhead into a collision with a tower. This setting usually
#   corresponds to bed width (in mm).
#slow_ratio: 3
#   The ratio used to limit velocity and acceleration on moves near the
#   extremes of the X axis. If vertical distance divided by horizontal
#   distance exceeds the value of slow_ratio, then velocity and
#   acceleration are limited to half their nominal values. If vertical
#   distance divided by horizontal distance exceeds twice the value of
#   the slow_ratio, then velocity and acceleration are limited to one
#   quarter of their nominal values. The default is 3.

# The stepper_left section is used to describe the stepper controlling
# the left tower. This section also controls the homing parameters
# (homing_speed, homing_retract_dist) for all towers.
[stepper_left]
position_endstop:
#   Distance (in mm) between the nozzle and the bed when the nozzle is
#   in the center of the build area and the endstops are triggered. This
#   parameter must be provided for stepper_left; for stepper_right this
#   parameter defaults to the value specified for stepper_left.
arm_length:
#   Length (in mm) of the diagonal rod that connects the tower carriage to
#   the print head. This parameter must be provided for stepper_left; for
#   stepper_right, this parameter defaults to the value specified for
#   stepper_left.
arm_x_length:
#   Horizontal distance between the print head and the tower when the
#   printers is homed. This parameter must be provided for stepper_left;
#   for stepper_right, this parameter defaults to the value specified for
#   stepper_left.

# The stepper_right section is used to desribe the stepper controlling the
# right tower.
[stepper_right]

# The stepper_y section is used to describe the stepper controlling
# the Y axis in a deltesian robot.
[stepper_y]
```

CoreXY Kinematics

See example-corexy.cfg for an example corexy (and h-bot) kinematics file.

Only parameters specific to corexy printers are described here - see common kinematic settings for available parameters.

```
[printer]
kinematics: corexy
max_z_velocity:
#   This sets the maximum velocity (in mm/s) of movement along the z
#   axis. This setting can be used to restrict the maximum speed of
#   the z stepper motor. The default is to use max_velocity for
#   max_z_velocity.
max_z_accel:
#   This sets the maximum acceleration (in mm/s^2) of movement along
#   the z axis. It limits the acceleration of the z stepper motor. The
#   default is to use max_accel for max_z_accel.

# The stepper_x section is used to describe the X axis as well as the
# stepper controlling the X+Y movement.
[stepper_x]

# The stepper_y section is used to describe the Y axis as well as the
# stepper controlling the X-Y movement.
[stepper_y]

# The stepper_z section is used to describe the stepper controlling
# the Z axis.
[stepper_z]
```

CoreXZ Kinematics

See example-corexz.cfg for an example corexz kinematics config file.

Only parameters specific to corexz printers are described here - see common kinematic settings for available parameters.

```
[printer]
kinematics: corexz
max_z_velocity:
#   This sets the maximum velocity (in mm/s) of movement along the z
#   axis. The default is to use max_velocity for max_z_velocity.
max_z_accel:
#   This sets the maximum acceleration (in mm/s^2) of movement along
#   the z axis. The default is to use max_accel for max_z_accel.

# The stepper_x section is used to describe the X axis as well as the
# stepper controlling the X+Z movement.
[stepper_x]

# The stepper_y section is used to describe the stepper controlling
# the Y axis.
[stepper_y]

# The stepper_z section is used to describe the Z axis as well as the
# stepper controlling the X-Z movement.
[stepper_z]
```

Hybrid-CoreXY Kinematics

See example-hybrid-corexy.cfg for an example hybrid corexy kinematics config file.

This kinematic is also known as Markforged kinematic.

Only parameters specific to hybrid corexy printers are described here see common kinematic settings for available parameters.

```
[printer]
kinematics: hybrid_corexy
max_z_velocity:
#   This sets the maximum velocity (in mm/s) of movement along the z
#   axis. The default is to use max_velocity for max_z_velocity.
max_z_accel:
#   This sets the maximum acceleration (in mm/s^2) of movement along
#   the z axis. The default is to use max_accel for max_z_accel.

# The stepper_x section is used to describe the X axis as well as the
# stepper controlling the X-Y movement.
[stepper_x]

# The stepper_y section is used to describe the stepper controlling
# the Y axis.
[stepper_y]

# The stepper_z section is used to describe the stepper controlling
# the Z axis.
[stepper_z]
```

Hybrid-CoreXZ Kinematics

See example-hybrid-corexz.cfg for an example hybrid corexz kinematics config file.

This kinematic is also known as Markforged kinematic.

Only parameters specific to hybrid corexy printers are described here see common kinematic settings for available parameters.

```
[printer]
kinematics: hybrid_corexz
max_z_velocity:
#   This sets the maximum velocity (in mm/s) of movement along the z
#   axis. The default is to use max_velocity for max_z_velocity.
max_z_accel:
#   This sets the maximum acceleration (in mm/s^2) of movement along
#   the z axis. The default is to use max_accel for max_z_accel.

# The stepper_x section is used to describe the X axis as well as the
# stepper controlling the X-Z movement.
[stepper_x]

# The stepper_y section is used to describe the stepper controlling
# the Y axis.
[stepper_y]

# The stepper_z section is used to describe the stepper controlling
# the Z axis.
[stepper_z]
```

Polar Kinematics

See example-polar.cfg for an example polar kinematics config file.

Only parameters specific to polar printers are described here - see common kinematic settings for available parameters.

POLAR KINEMATICS ARE A WORK IN PROGRESS. Moves around the 0, 0 position are known to not work properly.

```
[printer]
kinematics: polar
max_z_velocity:
#   This sets the maximum velocity (in mm/s) of movement along the z
#   axis. This setting can be used to restrict the maximum speed of
#   the z stepper motor. The default is to use max_velocity for
#   max_z_velocity.
max_z_accel:
#   This sets the maximum acceleration (in mm/s^2) of movement along
#   the z axis. It limits the acceleration of the z stepper motor. The
#   default is to use max_accel for max_z_accel.

# The stepper_bed section is used to describe the stepper controlling
# the bed.
```

```
[stepper_bed]
gear_ratio:
#   A gear_ratio must be specified and rotation_distance may not be
#   specified. For example, if the bed has an 80 toothed pulley driven
#   by a stepper with a 16 toothed pulley then one would specify a
#   gear ratio of "80:16". This parameter must be provided.

# The stepper_arm section is used to describe the stepper controlling
# the carriage on the arm.
[stepper_arm]

# The stepper_z section is used to describe the stepper controlling
# the Z axis.
[stepper_z]
```

Rotary delta Kinematics

See example-rotary-delta.cfg for an example rotary delta kinematics config file.

Only parameters specific to rotary delta printers are described here - see common kinematic settings for available parameters.

ROTARY DELTA KINEMATICS ARE A WORK IN PROGRESS. Homing moves may timeout and some boundary checks are not implemented.

```
[printer]
kinematics: rotary_delta
max_z_velocity:
#   For delta printers this limits the maximum velocity (in mm/s) of
#   moves with z axis movement. This setting can be used to reduce the
#   maximum speed of up/down moves (which require a higher step rate
#   than other moves on a delta printer). The default is to use
#   max_velocity for max_z_velocity.
#minimum_z_position: 0
#   The minimum Z position that the user may command the head to move
#   to.  The default is 0.
shoulder_radius:
#   Radius (in mm) of the horizontal circle formed by the three
#   shoulder joints, minus the radius of the circle formed by the
#   effector joints. This parameter may also be calculated as:
#     shoulder_radius = (delta_f - delta_e) / sqrt(12)
#   This parameter must be provided.
shoulder_height:
#   Distance (in mm) of the shoulder joints from the bed, minus the
#   effector toolhead height. This parameter must be provided.

# The stepper_a section describes the stepper controlling the rear
# right arm (at 30 degrees). This section also controls the homing
# parameters (homing_speed, homing_retract_dist) for all arms.
[stepper_a]
gear_ratio:
```

```
#   A gear_ratio must be specified and rotation_distance may not be
#   specified. For example, if the arm has an 80 toothed pulley driven
#   by a pulley with 16 teeth, which is in turn connected to a 60
#   toothed pulley driven by a stepper with a 16 toothed pulley, then
#   one would specify a gear ratio of "80:16, 60:16". This parameter
#   must be provided.
position_endstop:
#   Distance (in mm) between the nozzle and the bed when the nozzle is
#   in the center of the build area and the endstop triggers. This
#   parameter must be provided for stepper_a; for stepper_b and
#   stepper_c this parameter defaults to the value specified for
#   stepper_a.
upper_arm_length:
#   Length (in mm) of the arm connecting the "shoulder joint" to the
#   "elbow joint". This parameter must be provided for stepper_a; for
#   stepper_b and stepper_c this parameter defaults to the value
#   specified for stepper_a.
lower_arm_length:
#   Length (in mm) of the arm connecting the "elbow joint" to the
#   "effector joint". This parameter must be provided for stepper_a;
#   for stepper_b and stepper_c this parameter defaults to the value
#   specified for stepper_a.
#angle:
#   This option specifies the angle (in degrees) that the arm is at.
#   The default is 30 for stepper_a, 150 for stepper_b, and 270 for
#   stepper_c.

# The stepper_b section describes the stepper controlling the rear
# left arm (at 150 degrees).
[stepper_b]

# The stepper_c section describes the stepper controlling the front
# arm (at 270 degrees).
[stepper_c]

# The delta_calibrate section enables a DELTA_CALIBRATE extended
# g-code command that can calibrate the shoulder endstop positions.
[delta_calibrate]
radius:
#   Radius (in mm) of the area that may be probed. This is the radius
#   of nozzle coordinates to be probed; if using an automatic probe
#   with an XY offset then choose a radius small enough so that the
#   probe always fits over the bed. This parameter must be provided.
#speed: 50
#   The speed (in mm/s) of non-probing moves during the calibration.
#   The default is 50.
#horizontal_move_z: 5
#   The height (in mm) that the head should be commanded to move to
#   just prior to starting a probe operation. The default is 5.
```

Cable winch Kinematics

See the example-winch.cfg for an example cable winch kinematics config file.

Only parameters specific to cable winch printers are described here - see common kinematic settings for available parameters.

CABLE WINCH SUPPORT IS EXPERIMENTAL. Homing is not implemented on cable winch kinematics. In order to home the printer, manually send movement commands until the toolhead is at 0, 0, 0 and then issue a `G28` command.

```
[printer]
kinematics: winch

# The stepper_a section describes the stepper connected to the first
# cable winch. A minimum of 3 and a maximum of 26 cable winches may be
# defined (stepper_a to stepper_z) though it is common to define 4.
[stepper_a]
rotation_distance:
#   The rotation_distance is the nominal distance (in mm) the toolhead
#   moves towards the cable winch for each full rotation of the
#   stepper motor. This parameter must be provided.
anchor_x:
anchor_y:
anchor_z:
#   The X, Y, and Z position of the cable winch in cartesian space.
#   These parameters must be provided.
```

None Kinematics

It is possible to define a special "none" kinematics to disable kinematic support in Klipper. This may be useful for controlling devices that are not typical 3d-printers or for debugging purposes.

```
[printer]
kinematics: none
max_velocity: 1
max_accel: 1
#   The max_velocity and max_accel parameters must be defined. The
#   values are not used for "none" kinematics.
```

Common extruder and heated bed support

[extruder]

The extruder section is used to describe the heater parameters for the nozzle hotend along with the stepper controlling the extruder. See the command reference for additional information. See the pressure advance guide for information on tuning pressure advance.

```
[extruder]
step_pin:
dir_pin:
enable_pin:
microsteps:
rotation_distance:
#full_steps_per_rotation:
#gear_ratio:
#   See the "stepper" section for a description of the above
#   parameters. If none of the above parameters are specified then no
#   stepper will be associated with the nozzle hotend (though a
#   SYNC_EXTRUDER_MOTION command may associate one at run-time).
nozzle_diameter:
#   Diameter of the nozzle orifice (in mm). This parameter must be
#   provided.
filament_diameter:
#   The nominal diameter of the raw filament (in mm) as it enters the
#   extruder. This parameter must be provided.
#max_extrude_cross_section:
#   Maximum area (in mm^2) of an extrusion cross section (eg,
#   extrusion width multiplied by layer height). This setting prevents
#   excessive amounts of extrusion during relatively small XY moves.
#   If a move requests an extrusion rate that would exceed this value
#   it will cause an error to be returned. The default is: 4.0 *
#   nozzle_diameter^2
#instantaneous_corner_velocity: 1.000
#   The maximum instantaneous velocity change (in mm/s) of the
#   extruder during the junction of two moves. The default is 1mm/s.
#max_extrude_only_distance: 50.0
#   Maximum length (in mm of raw filament) that a retraction or
#   extrude-only move may have. If a retraction or extrude-only move
#   requests a distance greater than this value it will cause an error
#   to be returned. The default is 50mm.
#max_extrude_only_velocity:
#max_extrude_only_accel:
#   Maximum velocity (in mm/s) and acceleration (in mm/s^2) of the
#   extruder motor for retractions and extrude-only moves. These
#   settings do not have any impact on normal printing moves. If not
#   specified then they are calculated to match the limit an XY
#   printing move with a cross section of 4.0*nozzle_diameter^2 would
#   have.
#pressure_advance: 0.0
#   The amount of raw filament to push into the extruder during
#   extruder acceleration. An equal amount of filament is retracted
#   during deceleration. It is measured in millimeters per
#   millimeter/second. The default is 0, which disables pressure
#   advance.
#pressure_advance_smooth_time: 0.040
#   A time range (in seconds) to use when calculating the average
#   extruder velocity for pressure advance. A larger value results in
#   smoother extruder movements. This parameter may not exceed 200ms.
#   This setting only applies if pressure_advance is non-zero. The
#   default is 0.040 (40 milliseconds).
```

```
#
# The remaining variables describe the extruder heater.
heater_pin:
#   PWM output pin controlling the heater. This parameter must be
#   provided.
#max_power: 1.0
#   The maximum power (expressed as a value from 0.0 to 1.0) that the
#   heater_pin may be set to. The value 1.0 allows the pin to be set
#   fully enabled for extended periods, while a value of 0.5 would
#   allow the pin to be enabled for no more than half the time. This
#   setting may be used to limit the total power output (over extended
#   periods) to the heater. The default is 1.0.
sensor_type:
#   Type of sensor - common thermistors are "EPCOS 100K B57560G104F",
#   "ATC Semitec 104GT-2", "ATC Semitec 104NT-4-R025H42G", "Generic
#   3950","Honeywell 100K 135-104LAG-J01", "NTC 100K MGB18-104F39050L32",
#   "SliceEngineering 450", and "TDK NTCG104LH104JT1". See the
#   "Temperature sensors" section for other sensors. This parameter
#   must be provided.
sensor_pin:
#   Analog input pin connected to the sensor. This parameter must be
#   provided.
#pullup_resistor: 4700
#   The resistance (in ohms) of the pullup attached to the thermistor.
#   This parameter is only valid when the sensor is a thermistor. The
#   default is 4700 ohms.
#smooth_time: 1.0
#   A time value (in seconds) over which temperature measurements will
#   be smoothed to reduce the impact of measurement noise. The default
#   is 1 seconds.
control:
#   Control algorithm (either pid or watermark). This parameter must
#   be provided.
pid_Kp:
pid_Ki:
pid_Kd:
#   The proportional (pid_Kp), integral (pid_Ki), and derivative
#   (pid_Kd) settings for the PID feedback control system. Klipper
#   evaluates the PID settings with the following general formula:
#     heater_pwm = (Kp*error + Ki*integral(error) - Kd*derivative(error))
/ 255
#   Where "error" is "requested_temperature - measured_temperature"
#   and "heater_pwm" is the requested heating rate with 0.0 being full
#   off and 1.0 being full on. Consider using the PID_CALIBRATE
#   command to obtain these parameters. The pid_Kp, pid_Ki, and pid_Kd
#   parameters must be provided for PID heaters.
#max_delta: 2.0
#   On 'watermark' controlled heaters this is the number of degrees in
#   Celsius above the target temperature before disabling the heater
#   as well as the number of degrees below the target before
#   re-enabling the heater. The default is 2 degrees Celsius.
#pwm_cycle_time: 0.100
#   Time in seconds for each software PWM cycle of the heater. It is
#   not recommended to set this unless there is an electrical
#   requirement to switch the heater faster than 10 times a second.
#   The default is 0.100 seconds.
#min_extrude_temp: 170
#   The minimum temperature (in Celsius) at which extruder move
#   commands may be issued. The default is 170 Celsius.
min_temp:
max_temp:
#   The maximum range of valid temperatures (in Celsius) that the
#   heater must remain within. This controls a safety feature
#   implemented in the micro-controller code - should the measured
#   temperature ever fall outside this range then the micro-controller
#   will go into a shutdown state. This check can help detect some
#   heater and sensor hardware failures. Set this range just wide
#   enough so that reasonable temperatures do not result in an error.
#   These parameters must be provided.
```

[heater_bed]

The heater_bed section describes a heated bed. It uses the same heater settings described in the "extruder" section.

```
[heater_bed]
heater_pin:
sensor_type:
sensor_pin:
control:
min_temp:
max_temp:
#   See the "extruder" section for a description of the above parameters.
```

Bed level support

[bed_mesh]

Mesh Bed Leveling. One may define a bed_mesh config section to enable move transformations that offset the z axis based on a mesh generated from probed points. When using a probe to home the z-axis, it is recommended to define a safe_z_home section in printer.cfg to home toward the center of the print area.

See the bed mesh guide and command reference for additional information.

Visual Examples:

```
 rectangular bed, probe_count = 3, 3:
             x---x---x (max_point)
             |
             x---x---x
                     |
 (min_point) x---x---x
 round bed, round_probe_count = 5, bed_radius = r:
                 x (0, r) end
               /
             x---x---x
                       \
 (-r, 0) x---x---x---x---x (r, 0)
           \
             x---x---x
                   /
                 x  (0, -r) start
```

```
[bed_mesh]
#speed: 50
#   The speed (in mm/s) of non-probing moves during the calibration.
#   The default is 50.
#horizontal_move_z: 5
#   The height (in mm) that the head should be commanded to move to
#   just prior to starting a probe operation. The default is 5.
#mesh_radius:
#   Defines the radius of the mesh to probe for round beds. Note that
#   the radius is relative to the coordinate specified by the
#   mesh_origin option. This parameter must be provided for round beds
#   and omitted for rectangular beds.
#mesh_origin:
#   Defines the center X, Y coordinate of the mesh for round beds. This
#   coordinate is relative to the probe's location. It may be useful
#   to adjust the mesh_origin in an effort to maximize the size of the
#   mesh radius. Default is 0, 0. This parameter must be omitted for
#   rectangular beds.
#mesh_min:
#   Defines the minimum X, Y coordinate of the mesh for rectangular
#   beds. This coordinate is relative to the probe's location. This
#   will be the first point probed, nearest to the origin. This
#   parameter must be provided for rectangular beds.
#mesh_max:
#   Defines the maximum X, Y coordinate of the mesh for rectangular
#   beds. Adheres to the same principle as mesh_min, however this will
#   be the furthest point probed from the bed's origin. This parameter
#   must be provided for rectangular beds.
#probe_count: 3, 3
#   For rectangular beds, this is a comma separate pair of integer
#   values X, Y defining the number of points to probe along each
#   axis. A single value is also valid, in which case that value will
#   be applied to both axes. Default is 3, 3.
#round_probe_count: 5
#   For round beds, this integer value defines the maximum number of
#   points to probe along each axis. This value must be an odd number.
#   Default is 5.
#fade_start: 1.0
#   The gcode z position in which to start phasing out z-adjustment
#   when fade is enabled. Default is 1.0.
```

```
#fade_end: 0.0
#   The gcode z position in which phasing out completes. When set to a
#   value below fade_start, fade is disabled. It should be noted that
#   fade may add unwanted scaling along the z-axis of a print. If a
#   user wishes to enable fade, a value of 10.0 is recommended.
#   Default is 0.0, which disables fade.
#fade_target:
#   The z position in which fade should converge. When this value is
#   set to a non-zero value it must be within the range of z-values in
#   the mesh. Users that wish to converge to the z homing position
#   should set this to 0. Default is the average z value of the mesh.
#split_delta_z: .025
#   The amount of Z difference (in mm) along a move that will trigger
#   a split. Default is .025.
#move_check_distance: 5.0
#   The distance (in mm) along a move to check for split_delta_z.
#   This is also the minimum length that a move can be split. Default
#   is 5.0.
#mesh_pps: 2, 2
#   A comma separated pair of integers X, Y defining the number of
#   points per segment to interpolate in the mesh along each axis. A
#   "segment" can be defined as the space between each probed point.
#   The user may enter a single value which will be applied to both
#   axes. Default is 2, 2.
#algorithm: lagrange
#   The interpolation algorithm to use. May be either "lagrange" or
#   "bicubic". This option will not affect 3x3 grids, which are forced
#   to use lagrange sampling. Default is lagrange.
#bicubic_tension: .2
#   When using the bicubic algorithm the tension parameter above may
#   be applied to change the amount of slope interpolated. Larger
#   numbers will increase the amount of slope, which results in more
#   curvature in the mesh. Default is .2.
#relative_reference_index:
#   A point index in the mesh to reference all z values to. Enabling
#   this parameter produces a mesh relative to the probed z position
#   at the provided index.
#faulty_region_1_min:
#faulty_region_1_max:
#   Optional points that define a faulty region.  See docs/Bed_Mesh.md
#   for details on faulty regions.  Up to 99 faulty regions may be added.
#   By default no faulty regions are set.
```

[bed_tilt]

Bed tilt compensation. One may define a bed_tilt config section to enable move transformations that account for a tilted bed. Note that bed_mesh and bed_tilt are incompatible; both cannot be defined.

See the command reference for additional information.

```
[bed_tilt]
#x_adjust: 0
#   The amount to add to each move's Z height for each mm on the X
#   axis. The default is 0.
#y_adjust: 0
#   The amount to add to each move's Z height for each mm on the Y
#   axis. The default is 0.
#z_adjust: 0
#   The amount to add to the Z height when the nozzle is nominally at
#   0, 0. The default is 0.
# The remaining parameters control a BED_TILT_CALIBRATE extended
# g-code command that may be used to calibrate appropriate x and y
# adjustment parameters.
#points:
#   A list of X, Y coordinates (one per line; subsequent lines
#   indented) that should be probed during a BED_TILT_CALIBRATE
#   command. Specify coordinates of the nozzle and be sure the probe
#   is above the bed at the given nozzle coordinates. The default is
#   to not enable the command.
#speed: 50
#   The speed (in mm/s) of non-probing moves during the calibration.
#   The default is 50.
#horizontal_move_z: 5
#   The height (in mm) that the head should be commanded to move to
#   just prior to starting a probe operation. The default is 5.
```

[bed_screws]

Tool to help adjust bed leveling screws. One may define a [bed_screws] config section to enable a BED_SCREWS_ADJUST g-code command.

See the leveling guide and command reference for additional information.

```
[bed_screws]
#screw1:
#   The X, Y coordinate of the first bed leveling screw. This is a
#   position to command the nozzle to that is directly above the bed
#   screw (or as close as possible while still being above the bed).
#   This parameter must be provided.
#screw1_name:
#   An arbitrary name for the given screw. This name is displayed when
#   the helper script runs. The default is to use a name based upon
#   the screw XY location.
#screw1_fine_adjust:
#   An X, Y coordinate to command the nozzle to so that one can fine
#   tune the bed leveling screw. The default is to not perform fine
#   adjustments on the bed screw.
#screw2:
#screw2_name:
#screw2_fine_adjust:
```

```
#...
#   Additional bed leveling screws. At least three screws must be
#   defined.
#horizontal_move_z: 5
#   The height (in mm) that the head should be commanded to move to
#   when moving from one screw location to the next. The default is 5.
#probe_height: 0
#   The height of the probe (in mm) after adjusting for the thermal
#   expansion of bed and nozzle. The default is zero.
#speed: 50
#   The speed (in mm/s) of non-probing moves during the calibration.
#   The default is 50.
#probe_speed: 5
#   The speed (in mm/s) when moving from a horizontal_move_z position
#   to a probe_height position. The default is 5.
```

[screws_tilt_adjust]

Tool to help adjust bed screws tilt using Z probe. One may define a screws_tilt_adjust config section to enable a SCREWS_TILT_CALCULATE g-code command.

See the leveling guide and command reference for additional information.

```
[screws_tilt_adjust]
#screw1:
#   The (X, Y) coordinate of the first bed leveling screw. This is a
#   position to command the nozzle to so that the probe is directly
#   above the bed screw (or as close as possible while still being
#   above the bed). This is the base screw used in calculations. This
#   parameter must be provided.
#screw1_name:
#   An arbitrary name for the given screw. This name is displayed when
#   the helper script runs. The default is to use a name based upon
#   the screw XY location.
#screw2:
#screw2_name:
#...
#   Additional bed leveling screws. At least two screws must be
#   defined.
#speed: 50
#   The speed (in mm/s) of non-probing moves during the calibration.
#   The default is 50.
#horizontal_move_z: 5
#   The height (in mm) that the head should be commanded to move to
#   just prior to starting a probe operation. The default is 5.
#screw_thread: CW-M3
#   The type of screw used for bed level, M3, M4 or M5 and the
#   direction of the knob used to level the bed, clockwise decrease
#   counter-clockwise decrease.
#   Accepted values: CW-M3, CCW-M3, CW-M4, CCW-M4, CW-M5, CCW-M5.
#   Default value is CW-M3, most printers use an M3 screw and
#   turning the knob clockwise decrease distance.
```

[z_tilt]

Multiple Z stepper tilt adjustment. This feature enables independent adjustment of multiple z steppers (see the "stepper_z1" section) to adjust for tilt. If this section is present then a Z_TILT_ADJUST extended G-Code command becomes available.

```
[z_tilt]
#z_positions:
#   A list of X, Y coordinates (one per line; subsequent lines
#   indented) describing the location of each bed "pivot point". The
#   "pivot point" is the point where the bed attaches to the given Z
#   stepper. It is described using nozzle coordinates (the X, Y position
#   of the nozzle if it could move directly above the point). The
#   first entry corresponds to stepper_z, the second to stepper_z1,
#   the third to stepper_z2, etc. This parameter must be provided.
#points:
#   A list of X, Y coordinates (one per line; subsequent lines
#   indented) that should be probed during a Z_TILT_ADJUST command.
#   Specify coordinates of the nozzle and be sure the probe is above
#   the bed at the given nozzle coordinates. This parameter must be
#   provided.
#speed: 50
#   The speed (in mm/s) of non-probing moves during the calibration.
#   The default is 50.
#horizontal_move_z: 5
#   The height (in mm) that the head should be commanded to move to
#   just prior to starting a probe operation. The default is 5.
#retries: 0
#   Number of times to retry if the probed points aren't within
#   tolerance.
#retry_tolerance: 0
#   If retries are enabled then retry if largest and smallest probed
#   points differ more than retry_tolerance. Note the smallest unit of
#   change here would be a single step. However if you are probing
#   more points than steppers then you will likely have a fixed
#   minimum value for the range of probed points which you can learn
#   by observing command output.
```

[quad_gantry_level]

Moving gantry leveling using 4 independently controlled Z motors. Corrects hyperbolic parabola effects (potato chip) on moving gantry which is more flexible. WARNING: Using this on a moving bed may lead to undesirable results. If this section is present then a QUAD_GANTRY_LEVEL extended G-Code command becomes available. This routine assumes the following Z motor configuration:

```
 ----------------
 |Z1          Z2|
 |  ---------   |
 |  |       |   |
 |  |       |   |
 |  x--------   |
 |Z           Z3|
 ----------------
```

Where x is the 0, 0 point on the bed

```
[quad_gantry_level]
#gantry_corners:
#   A newline separated list of X, Y coordinates describing the two
#   opposing corners of the gantry. The first entry corresponds to Z,
#   the second to Z2. This parameter must be provided.
#points:
#   A newline separated list of four X, Y points that should be probed
#   during a QUAD_GANTRY_LEVEL command. Order of the locations is
#   important, and should correspond to Z, Z1, Z2, and Z3 location in
#   order. This parameter must be provided. For maximum accuracy,
#   ensure your probe offsets are configured.
#speed: 50
#   The speed (in mm/s) of non-probing moves during the calibration.
#   The default is 50.
#horizontal_move_z: 5
#   The height (in mm) that the head should be commanded to move to
#   just prior to starting a probe operation. The default is 5.
#max_adjust: 4
#   Safety limit if an adjustment greater than this value is requested
#   quad_gantry_level will abort.
#retries: 0
#   Number of times to retry if the probed points aren't within
#   tolerance.
#retry_tolerance: 0
#   If retries are enabled then retry if largest and smallest probed
#   points differ more than retry_tolerance.
```

[skew_correction]

Printer Skew Correction. It is possible to use software to correct printer skew across 3 planes, xy, xz, yz. This is done by printing a calibration model along a plane and measuring three lengths. Due to the nature of skew correction these lengths are set via gcode. See Skew Correction and Command Reference for details.

```
[skew_correction]
```

[z_thermal_adjust]

Temperature-dependant toolhead Z position adjustment. Compensate for vertical toolhead movement caused by thermal expansion of the printer's frame in real-time using a temperature sensor (typically coupled to a vertical section of frame).

See also: extended g-code commands.

```
[z_thermal_adjust]
#temp_coeff:
#   The temperature coefficient of expansion, in mm/degC. For example, a
#   temp_coeff of 0.01 mm/degC will move the Z axis downwards by 0.01 mm
for
#   every degree Celsius that the temperature sensor increases. Defaults
to
#   0.0 mm/degC, which applies no adjustment.
#smooth_time:
#   Smoothing window applied to the temperature sensor, in seconds. Can
reduce
#   motor noise from excessive small corrections in response to sensor
noise.
#   The default is 2.0 seconds.
#z_adjust_off_above:
#   Disables adjustments above this Z height [mm]. The last computed
correction
#   will remain applied until the toolhead moves below the specified Z
height
#   again. The default is 99999999.0 mm (always on).
#max_z_adjustment:
#   Maximum absolute adjustment that can be applied to the Z axis [mm].
The
#   default is 99999999.0 mm (unlimited).
#sensor_type:
#sensor_pin:
#min_temp:
#max_temp:
#   Temperature sensor configuration.
#   See the "extruder" section for the definition of the above
#   parameters.
#gcode_id:
#   See the "heater_generic" section for the definition of this
#   parameter.
```

Customized homing

[safe_z_home]

Safe Z homing. One may use this mechanism to home the Z axis at a specific X, Y coordinate. This is useful if the toolhead, for example has to move to the center of the bed before Z can be homed.

```
[safe_z_home]
home_xy_position:
```

```
#   A X, Y coordinate (e.g. 100, 100) where the Z homing should be
#   performed. This parameter must be provided.
#speed: 50.0
#   Speed at which the toolhead is moved to the safe Z home
#   coordinate. The default is 50 mm/s
#z_hop:
#   Distance (in mm) to lift the Z axis prior to homing. This is
#   applied to any homing command, even if it doesn't home the Z axis.
#   If the Z axis is already homed and the current Z position is less
#   than z_hop, then this will lift the head to a height of z_hop. If
#   the Z axis is not already homed the head is lifted by z_hop.
#   The default is to not implement Z hop.
#z_hop_speed: 15.0
#   Speed (in mm/s) at which the Z axis is lifted prior to homing. The
#   default is 15 mm/s.
#move_to_previous: False
#   When set to True, the X and Y axes are reset to their previous
#   positions after Z axis homing. The default is False.
```

[homing_override]

Homing override. One may use this mechanism to run a series of g-code commands in place of a G28 found in the normal g-code input. This may be useful on printers that require a specific procedure to home the machine.

```
[homing_override]
gcode:
#   A list of G-Code commands to execute in place of G28 commands
#   found in the normal g-code input. See docs/Command_Templates.md
#   for G-Code format. If a G28 is contained in this list of commands
#   then it will invoke the normal homing procedure for the printer.
#   The commands listed here must home all axes. This parameter must
#   be provided.
#axes: xyz
#   The axes to override. For example, if this is set to "z" then the
#   override script will only be run when the z axis is homed (eg, via
#   a "G28" or "G28 Z0" command). Note, the override script should
#   still home all axes. The default is "xyz" which causes the
#   override script to be run in place of all G28 commands.
#set_position_x:
#set_position_y:
#set_position_z:
#   If specified, the printer will assume the axis is at the specified
#   position prior to running the above g-code commands. Setting this
#   disables homing checks for that axis. This may be useful if the
#   head must move prior to invoking the normal G28 mechanism for an
#   axis. The default is to not force a position for an axis.
```

[endstop_phase]

Stepper phase adjusted endstops. To use this feature, define a config section with an "endstop_phase" prefix followed by the name of the corresponding stepper config section (for example, "[endstop_phase stepper_z]"). This feature can improve the accuracy of endstop switches. Add a bare "[endstop_phase]" declaration to enable the ENDSTOP_PHASE_CALIBRATE command.

See the endstop phases guide and command reference for additional information.

```
[endstop_phase stepper_z]
#endstop_accuracy:
#   Sets the expected accuracy (in mm) of the endstop. This represents
#   the maximum error distance the endstop may trigger (eg, if an
#   endstop may occasionally trigger 100um early or up to 100um late
#   then set this to 0.200 for 200um). The default is
#   4*rotation_distance/full_steps_per_rotation.
#trigger_phase:
#   This specifies the phase of the stepper motor driver to expect
#   when hitting the endstop. It is composed of two numbers separated
#   by a forward slash character - the phase and the total number of
#   phases (eg, "7/64"). Only set this value if one is sure the
#   stepper motor driver is reset every time the mcu is reset. If this
#   is not set, then the stepper phase will be detected on the first
#   home and that phase will be used on all subsequent homes.
#endstop_align_zero: False
#   If true then the position_endstop of the axis will effectively be
#   modified so that the zero position for the axis occurs at a full
#   step on the stepper motor. (If used on the Z axis and the print
#   layer height is a multiple of a full step distance then every
#   layer will occur on a full step.) The default is False.
```

G-Code macros and events

[gcode_macro]

G-Code macros (one may define any number of sections with a "gcode_macro" prefix). See the command template guide for more information.

```
[gcode_macro my_cmd]
#gcode:
#   A list of G-Code commands to execute in place of "my_cmd". See
#   docs/Command_Templates.md for G-Code format. This parameter must
#   be provided.
#variable_<name>:
#   One may specify any number of options with a "variable_" prefix.
#   The given variable name will be assigned the given value (parsed
#   as a Python literal) and will be available during macro expansion.
#   For example, a config with "variable_fan_speed = 75" might have
#   gcode commands containing "M106 S{ fan_speed * 255 }". Variables
#   can be changed at run-time using the SET_GCODE_VARIABLE command
#   (see docs/Command_Templates.md for details). Variable names may
#   not use upper case characters.
```

```
#rename_existing:
#   This option will cause the macro to override an existing G-Code
#   command and provide the previous definition of the command via the
#   name provided here. This can be used to override builtin G-Code
#   commands. Care should be taken when overriding commands as it can
#   cause complex and unexpected results. The default is to not
#   override an existing G-Code command.
#description: G-Code macro
#   This will add a short description used at the HELP command or while
#   using the auto completion feature. Default "G-Code macro"
```

[delayed_gcode]

Execute a gcode on a set delay. See the command template guide and command reference for more information.

```
[delayed_gcode my_delayed_gcode]
gcode:
#   A list of G-Code commands to execute when the delay duration has
#   elapsed. G-Code templates are supported. This parameter must be
#   provided.
#initial_duration: 0.0
#   The duration of the initial delay (in seconds). If set to a
#   non-zero value the delayed_gcode will execute the specified number
#   of seconds after the printer enters the "ready" state. This can be
#   useful for initialization procedures or a repeating delayed_gcode.
#   If set to 0 the delayed_gcode will not execute on startup.
#   Default is 0.
```

[save_variables]

Support saving variables to disk so that they are retained across restarts. See command templates and G-Code reference for further information.

```
[save_variables]
filename:
#   Required - provide a filename that would be used to save the
#   variables to disk e.g. ~/variables.cfg
```

[idle_timeout]

Idle timeout. An idle timeout is automatically enabled - add an explicit idle_timeout config section to change the default settings.

```
[idle_timeout]
#gcode:
```

```
#   A list of G-Code commands to execute on an idle timeout. See
#   docs/Command_Templates.md for G-Code format. The default is to run
#   "TURN_OFF_HEATERS" and "M84".
#timeout: 600
#   Idle time (in seconds) to wait before running the above G-Code
#   commands. The default is 600 seconds.
```

Optional G-Code features

[virtual_sdcard]

A virtual sdcard may be useful if the host machine is not fast enough to run OctoPrint well. It allows the Klipper host software to directly print gcode files stored in a directory on the host using standard sdcard G-Code commands (eg, M24).

```
[virtual_sdcard]
path:
#   The path of the local directory on the host machine to look for
#   g-code files. This is a read-only directory (sdcard file writes
#   are not supported). One may point this to OctoPrint's upload
#   directory (generally ~/.octoprint/uploads/ ). This parameter must
#   be provided.
#on_error_gcode:
#   A list of G-Code commands to execute when an error is reported.
```

[sdcard_loop]

Some printers with stage-clearing features, such as a part ejector or a belt printer, can find use in looping sections of the sdcard file. (For example, to print the same part over and over, or repeat the a section of a part for a chain or other repeated pattern).

See the command reference for supported commands. See the sample-macros.cfg file for a Marlin compatible M808 G-Code macro.

```
[sdcard_loop]
```

[force_move]

Support manually moving stepper motors for diagnostic purposes. Note, using this feature may place the printer in an invalid state - see the command reference for important details.

```
[force_move]
#enable_force_move: False
#   Set to true to enable FORCE_MOVE and SET_KINEMATIC_POSITION
#   extended G-Code commands. The default is false.
```

[pause_resume]

Pause/Resume functionality with support of position capture and restore. See the command reference for more information.

```
[pause_resume]
#recover_velocity: 50.
#   When capture/restore is enabled, the speed at which to return to
#   the captured position (in mm/s). Default is 50.0 mm/s.
```

[firmware_retraction]

Firmware filament retraction. This enables G10 (retract) and G11 (unretract) GCODE commands issued by many slicers. The parameters below provide startup defaults, although the values can be adjusted via the SET_RETRACTION command), allowing per-filament settings and runtime tuning.

```
[firmware_retraction]
#retract_length: 0
#   The length of filament (in mm) to retract when G10 is activated,
#   and to unretract when G11 is activated (but see
#   unretract_extra_length below). The default is 0 mm.
#retract_speed: 20
#   The speed of retraction, in mm/s. The default is 20 mm/s.
#unretract_extra_length: 0
#   The length (in mm) of *additional* filament to add when
#   unretracting.
#unretract_speed: 10
#   The speed of unretraction, in mm/s. The default is 10 mm/s.
```

[gcode_arcs]

Support for gcode arc (G2/G3) commands.

```
[gcode_arcs]
#resolution: 1.0
#   An arc will be split into segments. Each segment's length will
#   equal the resolution in mm set above. Lower values will produce a
#   finer arc, but also more work for your machine. Arcs smaller than
#   the configured value will become straight lines. The default is
#   1mm.
```

[respond]

Enable the "M118" and "RESPOND" extended commands.

```
[respond]
#default_type: echo
#   Sets the default prefix of the "M118" and "RESPOND" output to one
#   of the following:
#       echo: "echo: " (This is the default)
#       command: "// "
#       error: "!! "
#default_prefix: echo:
#   Directly sets the default prefix. If present, this value will
#   override the "default_type".
```

[exclude_object]

Enables support to exclude or cancel individual objects during the printing process.

See the exclude objects guide and command reference for additional information. See the sample-macros.cfg file for a Marlin/RepRapFirmware compatible M486 G-Code macro.

```
[exclude_object]
```

Resonance compensation

[input_shaper]

Enables resonance compensation. Also see the command reference.

```
[input_shaper]
#shaper_freq_x: 0
#   A frequency (in Hz) of the input shaper for X axis. This is
#   usually a resonance frequency of X axis that the input shaper
#   should suppress. For more complex shapers, like 2- and 3-hump EI
#   input shapers, this parameter can be set from different
#   considerations. The default value is 0, which disables input
#   shaping for X axis.
#shaper_freq_y: 0
#   A frequency (in Hz) of the input shaper for Y axis. This is
#   usually a resonance frequency of Y axis that the input shaper
#   should suppress. For more complex shapers, like 2- and 3-hump EI
#   input shapers, this parameter can be set from different
#   considerations. The default value is 0, which disables input
#   shaping for Y axis.
#shaper_type: mzv
#   A type of the input shaper to use for both X and Y axes. Supported
#   shapers are zv, mzv, zvd, ei, 2hump_ei, and 3hump_ei. The default
#   is mzv input shaper.
#shaper_type_x:
#shaper_type_y:
#   If shaper_type is not set, these two parameters can be used to
#   configure different input shapers for X and Y axes. The same
#   values are supported as for shaper_type parameter.
#damping_ratio_x: 0.1
#damping_ratio_y: 0.1
#   Damping ratios of vibrations of X and Y axes used by input shapers
#   to improve vibration suppression. Default value is 0.1 which is a
#   good all-round value for most printers. In most circumstances this
#   parameter requires no tuning and should not be changed.
```

[adxl345]

Support for ADXL345 accelerometers. This support allows one to query accelerometer measurements from the sensor. This enables an ACCELEROMETER_MEASURE command (see G-Codes for more information). The default chip name is "default", but one may specify an explicit name (eg, [adxl345 my_chip_name]).

```
[adxl345]
cs_pin:
#   The SPI enable pin for the sensor. This parameter must be provided.
#spi_speed: 5000000
#   The SPI speed (in hz) to use when communicating with the chip.
#   The default is 5000000.
#spi_bus:
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   See the "common SPI settings" section for a description of the
#   above parameters.
#axes_map: x, y, z
#   The accelerometer axis for each of the printer's X, Y, and Z axes.
#   This may be useful if the accelerometer is mounted in an
#   orientation that does not match the printer orientation. For
#   example, one could set this to "y, x, z" to swap the X and Y axes.
#   It is also possible to negate an axis if the accelerometer
#   direction is reversed (eg, "x, z, -y"). The default is "x, y, z".
#rate: 3200
#   Output data rate for ADXL345. ADXL345 supports the following data
#   rates: 3200, 1600, 800, 400, 200, 100, 50, and 25. Note that it is
#   not recommended to change this rate from the default 3200, and
#   rates below 800 will considerably affect the quality of resonance
#   measurements.
```

[mpu9250]

Support for MPU-9250, MPU-9255, MPU-6515, MPU-6050, and MPU-6500 accelerometers (one may define any number of sections with an "mpu9250" prefix).

```
[mpu9250 my_accelerometer]
#i2c_address:
#   Default is 104 (0x68). If AD0 is high, it would be 0x69 instead.
```

```
#i2c_mcu:
#i2c_bus:
#i2c_speed: 400000
#   See the "common I2C settings" section for a description of the
#   above parameters. The default "i2c_speed" is 400000.
#axes_map: x, y, z
#   See the "adxl345" section for information on this parameter.
```

[resonance_tester]

Support for resonance testing and automatic input shaper calibration. In order to use most of the functionality of this module, additional software dependencies must be installed; refer to Measuring Resonances and the command reference for more information. See the Max smoothing section of the measuring resonances guide for more information on `max_smoothing` parameter and its use.

```
[resonance_tester]
#probe_points:
#   A list of X, Y, Z coordinates of points (one point per line) to test
#   resonances at. At least one point is required. Make sure that all
#   points with some safety margin in XY plane (~a few centimeters)
#   are reachable by the toolhead.
#accel_chip:
#   A name of the accelerometer chip to use for measurements. If
#   adxl345 chip was defined without an explicit name, this parameter
#   can simply reference it as "accel_chip: adxl345", otherwise an
#   explicit name must be supplied as well, e.g. "accel_chip: adxl345
#   my_chip_name". Either this, or the next two parameters must be
#   set.
#accel_chip_x:
#accel_chip_y:
#   Names of the accelerometer chips to use for measurements for each
#   of the axis. Can be useful, for instance, on bed slinger printer,
#   if two separate accelerometers are mounted on the bed (for Y axis)
#   and on the toolhead (for X axis). These parameters have the same
#   format as 'accel_chip' parameter. Only 'accel_chip' or these two
#   parameters must be provided.
#max_smoothing:
#   Maximum input shaper smoothing to allow for each axis during shaper
#   auto-calibration (with 'SHAPER_CALIBRATE' command). By default no
#   maximum smoothing is specified. Refer to Measuring_Resonances guide
#   for more details on using this feature.
#min_freq: 5
#   Minimum frequency to test for resonances. The default is 5 Hz.
#max_freq: 133.33
#   Maximum frequency to test for resonances. The default is 133.33 Hz.
#accel_per_hz: 75
#   This parameter is used to determine which acceleration to use to
#   test a specific frequency: accel = accel_per_hz * freq. Higher the
#   value, the higher is the energy of the oscillations. Can be set to
#   a lower than the default value if the resonances get too strong on
#   the printer. However, lower values make measurements of
#   high-frequency resonances less precise. The default value is 75
#   (mm/sec).
#hz_per_sec: 1
#   Determines the speed of the test. When testing all frequencies in
#   range [min_freq, max_freq], each second the frequency increases by
#   hz_per_sec. Small values make the test slow, and the large values
#   will decrease the precision of the test. The default value is 1.0
#   (Hz/sec == sec^-2).
```

Config file helpers

[board_pins]

Board pin aliases (one may define any number of sections with a "board_pins" prefix). Use this to define aliases for the pins on a micro-controller.

```
[board_pins my_aliases]
mcu: mcu
#   A comma separated list of micro-controllers that may use the
#   aliases. The default is to apply the aliases to the main "mcu".
aliases:
aliases_<name>:
#   A comma separated list of "name=value" aliases to create for the
#   given micro-controller. For example, "EXP1_1=PE6" would create an
#   "EXP1_1" alias for the "PE6" pin. However, if "value" is enclosed
#   in "<>" then "name" is created as a reserved pin (for example,
#   "EXP1_9=<GND>" would reserve "EXP1_9"). Any number of options
#   starting with "aliases_" may be specified.
```

[include]

Include file support. One may include additional config file from the main printer config file. Wildcards may also be used (eg, "configs/*.cfg").

```
[include my_other_config.cfg]
```

[duplicate_pin_override]

This tool allows a single micro-controller pin to be defined multiple times in a config file without normal error checking. This is intended for diagnostic and debugging purposes. This section is not needed where Klipper supports using the same pin multiple times, and using this override may cause confusing and unexpected results.

```
[duplicate_pin_override]
pins:
#   A comma separated list of pins that may be used multiple times in
#   a config file without normal error checks. This parameter must be
#   provided.
```

Bed probing hardware

[probe]

Z height probe. One may define this section to enable Z height probing hardware. When this section is enabled, PROBE and QUERY_PROBE extended g-code commands become available. Also, see the probe calibrate guide. The probe section also creates a virtual "probe:z_virtual_endstop" pin. One may set the stepper_z endstop_pin to this virtual pin on cartesian style printers that use the probe in place of a z endstop. If using "probe:z_virtual_endstop" then do not define a position_endstop in the stepper_z config section.

```
[probe]
pin:
#   Probe detection pin. If the pin is on a different microcontroller
#   than the Z steppers then it enables "multi-mcu homing". This
#   parameter must be provided.
#deactivate_on_each_sample: True
#   This determines if Klipper should execute deactivation gcode
#   between each probe attempt when performing a multiple probe
#   sequence. The default is True.
#x_offset: 0.0
#   The distance (in mm) between the probe and the nozzle along the
#   x-axis. The default is 0.
#y_offset: 0.0
#   The distance (in mm) between the probe and the nozzle along the
#   y-axis. The default is 0.
z_offset:
#   The distance (in mm) between the bed and the nozzle when the probe
#   triggers. This parameter must be provided.
#speed: 5.0
#   Speed (in mm/s) of the Z axis when probing. The default is 5mm/s.
#samples: 1
#   The number of times to probe each point. The probed z-values will
#   be averaged. The default is to probe 1 time.
#sample_retract_dist: 2.0
#   The distance (in mm) to lift the toolhead between each sample (if
#   sampling more than once). The default is 2mm.
#lift_speed:
#   Speed (in mm/s) of the Z axis when lifting the probe between
#   samples. The default is to use the same value as the 'speed'
#   parameter.
#samples_result: average
#   The calculation method when sampling more than once - either
#   "median" or "average". The default is average.
#samples_tolerance: 0.100
#   The maximum Z distance (in mm) that a sample may differ from other
#   samples. If this tolerance is exceeded then either an error is
#   reported or the attempt is restarted (see
#   samples_tolerance_retries). The default is 0.100mm.
#samples_tolerance_retries: 0
#   The number of times to retry if a sample is found that exceeds
#   samples_tolerance. On a retry, all current samples are discarded
#   and the probe attempt is restarted. If a valid set of samples are
#   not obtained in the given number of retries then an error is
#   reported. The default is zero which causes an error to be reported
#   on the first sample that exceeds samples_tolerance.
#activate_gcode:
#   A list of G-Code commands to execute prior to each probe attempt.
#   See docs/Command_Templates.md for G-Code format. This may be
#   useful if the probe needs to be activated in some way. Do not
#   issue any commands here that move the toolhead (eg, G1). The
#   default is to not run any special G-Code commands on activation.
#deactivate_gcode:
#   A list of G-Code commands to execute after each probe attempt
#   completes. See docs/Command_Templates.md for G-Code format. Do not
#   issue any commands here that move the toolhead. The default is to
#   not run any special G-Code commands on deactivation.
```

[bltouch]

BLTouch probe. One may define this section (instead of a probe section) to enable a BLTouch probe. See BL-Touch guide and command reference for further information. A virtual "probe:z_virtual_endstop" pin is also created (see the "probe" section for the details).

```
[bltouch]
sensor_pin:
#   Pin connected to the BLTouch sensor pin. Most BLTouch devices
#   require a pullup on the sensor pin (prefix the pin name with "^").
#   This parameter must be provided.
control_pin:
#   Pin connected to the BLTouch control pin. This parameter must be
#   provided.
#pin_move_time: 0.680
#   The amount of time (in seconds) to wait for the BLTouch pin to
#   move up or down. The default is 0.680 seconds.
#stow_on_each_sample: True
#   This determines if Klipper should command the pin to move up
#   between each probe attempt when performing a multiple probe
#   sequence. Read the directions in docs/BLTouch.md before setting
#   this to False. The default is True.
#probe_with_touch_mode: False
#   If this is set to True then Klipper will probe with the device in
#   "touch_mode". The default is False (probing in "pin_down" mode).
#pin_up_reports_not_triggered: True
#   Set if the BLTouch consistently reports the probe in a "not
#   triggered" state after a successful "pin_up" command. This should
#   be True for all genuine BLTouch devices. Read the directions in
#   docs/BLTouch.md before setting this to False. The default is True.
```

```
#pin_up_touch_mode_reports_triggered: True
#   Set if the BLTouch consistently reports a "triggered" state after
#   the commands "pin_up" followed by "touch_mode". This should be
#   True for all genuine BLTouch devices. Read the directions in
#   docs/BLTouch.md before setting this to False. The default is True.
#set_output_mode:
#   Request a specific sensor pin output mode on the BLTouch V3.0 (and
#   later). This setting should not be used on other types of probes.
#   Set to "5V" to request a sensor pin output of 5 Volts (only use if
#   the controller board needs 5V mode and is 5V tolerant on its input
#   signal line). Set to "OD" to request the sensor pin output use
#   open drain mode. The default is to not request an output mode.
#x_offset:
#y_offset:
#z_offset:
#speed:
#lift_speed:
#samples:
#sample_retract_dist:
#samples_result:
#samples_tolerance:
#samples_tolerance_retries:
#   See the "probe" section for information on these parameters.
```

[smart_effector]

The "Smart Effector" from Duet3d implements a Z probe using a force sensor. One may define this section instead of `[probe]` to enable the Smart Effector specific features. This also enables runtime commands to adjust the parameters of the Smart Effector at run time.

```
[smart_effector]
pin:
#   Pin connected to the Smart Effector Z Probe output pin (pin 5). Note
that
#   pullup resistor on the board is generally not required. However, if
the
#   output pin is connected to the board pin with a pullup resistor, that
#   resistor must be high value (e.g. 10K Ohm or more). Some boards have a
low
#   value pullup resistor on the Z probe input, which will likely result
in an
#   always-triggered probe state. In this case, connect the Smart Effector
to
#   a different pin on the board. This parameter is required.
#control_pin:
#   Pin connected to the Smart Effector control input pin (pin 7). If
provided,
#   Smart Effector sensitivity programming commands become available.
#probe_accel:
#   If set, limits the acceleration of the probing moves (in mm/sec^2).
#   A sudden large acceleration at the beginning of the probing move may
#   cause spurious probe triggering, especially if the hotend is heavy.
#   To prevent that, it may be necessary to reduce the acceleration of
#   the probing moves via this parameter.
#recovery_time: 0.4
#   A delay between the travel moves and the probing moves in seconds. A
fast
#   travel move prior to probing may result in a spurious probe
triggering.
#   This may cause 'Probe triggered prior to movement' errors if no delay
#   is set. Value 0 disables the recovery delay.
#   Default value is 0.4.
#x_offset:
#y_offset:
#   Should be left unset (or set to 0).
z_offset:
#   Trigger height of the probe. Start with -0.1 (mm), and adjust later
using
#   `PROBE_CALIBRATE` command. This parameter must be provided.
#speed:
#   Speed (in mm/s) of the Z axis when probing. It is recommended to start
#   with the probing speed of 20 mm/s and adjust it as necessary to
improve
#   the accuracy and repeatability of the probe triggering.
#samples:
#sample_retract_dist:
#samples_result:
#samples_tolerance:
#samples_tolerance_retries:
#activate_gcode:
#deactivate_gcode:
#deactivate_on_each_sample:
#   See the "probe" section for more information on the parameters above.
```

Additional stepper motors and extruders

[stepper_z1]

Multi-stepper axes. On a cartesian style printer, the stepper controlling a given axis may have additional config blocks defining steppers that should be stepped in concert with the primary stepper. One may define any number of sections with a numeric suffix starting at 1 (for example, "stepper_z1", "stepper_z2", etc.).

```
[stepper_z1]
#step_pin:
#dir_pin:
#enable_pin:
#microsteps:
#rotation_distance:
#   See the "stepper" section for the definition of the above parameters.
#endstop_pin:
#   If an endstop_pin is defined for the additional stepper then the
#   stepper will home until the endstop is triggered. Otherwise, the
#   stepper will home until the endstop on the primary stepper for the
#   axis is triggered.
```

[extruder1]

In a multi-extruder printer add an additional extruder section for each additional extruder. The additional extruder sections should be named "extruder1", "extruder2", "extruder3", and so on. See the "extruder" section for a description of available parameters.

See sample-multi-extruder.cfg for an example configuration.

```
[extruder1]
#step_pin:
#dir_pin:
#...
#   See the "extruder" section for available stepper and heater
#   parameters.
#shared_heater:
#   This option is deprecated and should no longer be specified.
```

[dual_carriage]

Support for cartesian printers with dual carriages on a single axis. The active carriage is set via the SET_DUAL_CARRIAGE extended g-code command. The "SET_DUAL_CARRIAGE CARRIAGE=1" command will activate the carriage defined in this section (CARRIAGE=0 will return activation to the primary carriage). Dual carriage support is typically combined with extra extruders - the SET_DUAL_CARRIAGE command is often called at the same time as the ACTIVATE_EXTRUDER command. Be sure to park the carriages during deactivation.

See sample-idex.cfg for an example configuration.

```
[dual_carriage]
axis:
#   The axis this extra carriage is on (either x or y). This parameter
#   must be provided.
#step_pin:
#dir_pin:
#enable_pin:
#microsteps:
#rotation_distance:
#endstop_pin:
#position_endstop:
#position_min:
#position_max:
#   See the "stepper" section for the definition of the above parameters.
```

[extruder_stepper]

Support for additional steppers synchronized to the movement of an extruder (one may define any number of sections with an "extruder_stepper" prefix).

See the command reference for more information.

```
[extruder_stepper my_extra_stepper]
extruder:
#   The extruder this stepper is synchronized to. If this is set to an
#   empty string then the stepper will not be synchronized to an
#   extruder. This parameter must be provided.
#step_pin:
#dir_pin:
#enable_pin:
#microsteps:
#rotation_distance:
#   See the "stepper" section for the definition of the above
#   parameters.
```

[manual_stepper]

Manual steppers (one may define any number of sections with a "manual_stepper" prefix). These are steppers that are controlled by the MANUAL_STEPPER g-code command. For example: "MANUAL_STEPPER STEPPER=my_stepper MOVE=10 SPEED=5". See G-Codes file for a description of the MANUAL_STEPPER command. The steppers are not connected to the normal printer kinematics.

```
[manual_stepper my_stepper]
#step_pin:
#dir_pin:
#enable_pin:
#microsteps:
#rotation_distance:
#   See the "stepper" section for a description of these parameters.
#velocity:
#   Set the default velocity (in mm/s) for the stepper. This value
#   will be used if a MANUAL_STEPPER command does not specify a SPEED
#   parameter. The default is 5mm/s.
#accel:
#   Set the default acceleration (in mm/s^2) for the stepper. An
#   acceleration of zero will result in no acceleration. This value
#   will be used if a MANUAL_STEPPER command does not specify an ACCEL
#   parameter. The default is zero.
#endstop_pin:
#   Endstop switch detection pin. If specified, then one may perform
#   "homing moves" by adding a STOP_ON_ENDSTOP parameter to
#   MANUAL_STEPPER movement commands.
```

Custom heaters and sensors

[verify_heater]

Heater and temperature sensor verification. Heater verification is automatically enabled for each heater that is configured on the printer. Use verify_heater sections to change the default settings.

```
[verify_heater heater_config_name]
#max_error: 120
#   The maximum "cumulative temperature error" before raising an
#   error. Smaller values result in stricter checking and larger
#   values allow for more time before an error is reported.
#   Specifically, the temperature is inspected once a second and if it
#   is close to the target temperature then an internal "error
#   counter" is reset; otherwise, if the temperature is below the
#   target range then the counter is increased by the amount the
#   reported temperature differs from that range. Should the counter
#   exceed this "max_error" then an error is raised. The default is
#   120.
#check_gain_time:
#   This controls heater verification during initial heating. Smaller
#   values result in stricter checking and larger values allow for
#   more time before an error is reported. Specifically, during
#   initial heating, as long as the heater increases in temperature
#   within this time frame (specified in seconds) then the internal
#   "error counter" is reset. The default is 20 seconds for extruders
#   and 60 seconds for heater_bed.
#hysteresis: 5
#   The maximum temperature difference (in Celsius) to a target
#   temperature that is considered in range of the target. This
#   controls the max_error range check. It is rare to customize this
#   value. The default is 5.
#heating_gain: 2
#   The minimum temperature (in Celsius) that the heater must increase
#   by during the check_gain_time check. It is rare to customize this
#   value. The default is 2.
```

[homing_heaters]

Tool to disable heaters when homing or probing an axis.

```
[homing_heaters]
#steppers:
#   A comma separated list of steppers that should cause heaters to be
#   disabled. The default is to disable heaters for any homing/probing
#   move.
#   Typical example: stepper_z
#heaters:
#   A comma separated list of heaters to disable during homing/probing
#   moves. The default is to disable all heaters.
#   Typical example: extruder, heater_bed
```

[thermistor]

Custom thermistors (one may define any number of sections with a "thermistor" prefix). A custom thermistor may be used in the sensor_type field of a heater config section. (For example, if one defines a "[thermistor my_thermistor]" section then one may use a "sensor_type: my_thermistor" when defining a heater.) Be sure to place the thermistor section in the config file above its first use in a heater section.

```
[thermistor my_thermistor]
#temperature1:
#resistance1:
#temperature2:
#resistance2:
#temperature3:
#resistance3:
#   Three resistance measurements (in Ohms) at the given temperatures
#   (in Celsius). The three measurements will be used to calculate the
#   Steinhart-Hart coefficients for the thermistor. These parameters
#   must be provided when using Steinhart-Hart to define the
#   thermistor.
#beta:
#   Alternatively, one may define temperature1, resistance1, and beta
#   to define the thermistor parameters. This parameter must be
#   provided when using "beta" to define the thermistor.
```

[adc_temperature]

Custom ADC temperature sensors (one may define any number of sections with an "adc_temperature" prefix). This allows one to define a custom temperature sensor that measures a voltage on an Analog to Digital Converter (ADC) pin and uses linear interpolation between a set of configured temperature/voltage (or temperature/resistance) measurements to determine the temperature. The resulting sensor can be used as a sensor_type in a heater section. (For example, if one defines a "[adc_temperature my_sensor]" section then one may use a "sensor_type: my_sensor" when defining a heater.) Be sure to place the sensor section in the config file above its first use in a heater section.

```
[adc_temperature my_sensor]
#temperature1:
#voltage1:
#temperature2:
#voltage2:
#...
#   A set of temperatures (in Celsius) and voltages (in Volts) to use
#   as reference when converting a temperature. A heater section using
#   this sensor may also specify adc_voltage and voltage_offset
#   parameters to define the ADC voltage (see "Common temperature
#   amplifiers" section for details). At least two measurements must
#   be provided.
#temperature1:
#resistance1:
#temperature2:
```

```
#resistance2:
#...
#   Alternatively one may specify a set of temperatures (in Celsius)
#   and resistance (in Ohms) to use as reference when converting a
#   temperature. A heater section using this sensor may also specify a
#   pullup_resistor parameter (see "extruder" section for details). At
#   least two measurements must be provided.
```

[heater_generic]

Generic heaters (one may define any number of sections with a "heater_generic" prefix). These heaters behave similarly to standard heaters (extruders, heated beds). Use the SET_HEATER_TEMPERATURE command (see G-Codes for details) to set the target temperature.

```
[heater_generic my_generic_heater]
#gcode_id:
#   The id to use when reporting the temperature in the M105 command.
#   This parameter must be provided.
#heater_pin:
#max_power:
#sensor_type:
#sensor_pin:
#smooth_time:
#control:
#pid_Kp:
#pid_Ki:
#pid_Kd:
#pwm_cycle_time:
#min_temp:
#max_temp:
#   See the "extruder" section for the definition of the above
#   parameters.
```

[temperature_sensor]

Generic temperature sensors. One can define any number of additional temperature sensors that are reported via the M105 command.

```
[temperature_sensor my_sensor]
#sensor_type:
#sensor_pin:
#min_temp:
#max_temp:
#   See the "extruder" section for the definition of the above
#   parameters.
#gcode_id:
#   See the "heater_generic" section for the definition of this
#   parameter.
```

Temperature sensors

Klipper includes definitions for many types of temperature sensors. These sensors may be used in any config section that requires a temperature sensor (such as an `[extruder]` or `[heater_bed]` section).

Common thermistors

Common thermistors. The following parameters are available in heater sections that use one of these sensors.

```
sensor_type:
#   One of "EPCOS 100K B57560G104F", "ATC Semitec 104GT-2",
#   "ATC Semitec 104NT-4-R025H42G", "Generic 3950",
#   "Honeywell 100K 135-104LAG-J01", "NTC 100K MGB18-104F39050L32",
#   "SliceEngineering 450", or "TDK NTCG104LH104JT1"
sensor_pin:
#   Analog input pin connected to the thermistor. This parameter must
#   be provided.
#pullup_resistor: 4700
#   The resistance (in ohms) of the pullup attached to the thermistor.
#   The default is 4700 ohms.
#inline_resistor: 0
#   The resistance (in ohms) of an extra (not heat varying) resistor
#   that is placed inline with the thermistor. It is rare to set this.
#   The default is 0 ohms.
```

Common temperature amplifiers

Common temperature amplifiers. The following parameters are available in heater sections that use one of these sensors.

```
sensor_type:
#   One of "PT100 INA826", "AD595", "AD597", "AD8494", "AD8495",
#   "AD8496", or "AD8497".
sensor_pin:
#   Analog input pin connected to the sensor. This parameter must be
#   provided.
#adc_voltage: 5.0
#   The ADC comparison voltage (in Volts). The default is 5 volts.
#voltage_offset: 0
#   The ADC voltage offset (in Volts). The default is 0.
```

Directly connected PT1000 sensor

Directly connected PT1000 sensor. The following parameters are available in heater sections that use one of these sensors.

```
sensor_type: PT1000
sensor_pin:
#   Analog input pin connected to the sensor. This parameter must be
#   provided.
#pullup_resistor: 4700
#   The resistance (in ohms) of the pullup attached to the sensor. The
#   default is 4700 ohms.
```

MAXxxxxx temperature sensors

MAXxxxxx serial peripheral interface (SPI) temperature based sensors. The following parameters are available in heater sections that use one of these sensor types.

```
sensor_type:
#   One of "MAX6675", "MAX31855", "MAX31856", or "MAX31865".
sensor_pin:
#   The chip select line for the sensor chip. This parameter must be
#   provided.
#spi_speed: 4000000
#   The SPI speed (in hz) to use when communicating with the chip.
#   The default is 4000000.
#spi_bus:
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   See the "common SPI settings" section for a description of the
#   above parameters.
#tc_type: K
#tc_use_50Hz_filter: False
#tc_averaging_count: 1
#   The above parameters control the sensor parameters of MAX31856
#   chips. The defaults for each parameter are next to the parameter
#   name in the above list.
#rtd_nominal_r: 100
#rtd_reference_r: 430
#rtd_num_of_wires: 2
#rtd_use_50Hz_filter: False
#   The above parameters control the sensor parameters of MAX31865
#   chips. The defaults for each parameter are next to the parameter
#   name in the above list.
```

BMP280/BME280/BME680 temperature sensor

BMP280/BME280/BME680 two wire interface (I2C) environmental sensors. Note that these sensors are not intended for use with extruders and heater beds, but rather for monitoring ambient temperature (C), pressure (hPa), relative humidity and in case of the BME680 gas level. See sample-macros.cfg for a gcode_macro that may be used to report pressure and humidity in addition to temperature.

```
sensor_type: BME280
#i2c_address:
#   Default is 118 (0x76). Some BME280 sensors have an address of 119
#   (0x77).
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   See the "common I2C settings" section for a description of the
#   above parameters.
```

HTU21D sensor

HTU21D family two wire interface (I2C) environmental sensor. Note that this sensor is not intended for use with extruders and heater beds, but rather for monitoring ambient temperature (C) and relative humidity. See sample-macros.cfg for a gcode_macro that may be used to report humidity in addition to temperature.

```
sensor_type:
#   Must be "HTU21D" , "SI7013", "SI7020", "SI7021" or "SHT21"
#i2c_address:
#   Default is 64 (0x40).
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   See the "common I2C settings" section for a description of the
#   above parameters.
#htu21d_hold_master:
#   If the sensor can hold the I2C buf while reading. If True no other
#   bus communication can be performed while reading is in progress.
#   Default is False.
#htu21d_resolution:
#   The resolution of temperature and humidity reading.
#   Valid values are:
#    'TEMP14_HUM12' -> 14bit for Temp and 12bit for humidity
#    'TEMP13_HUM10' -> 13bit for Temp and 10bit for humidity
#    'TEMP12_HUM08' -> 12bit for Temp and 08bit for humidity
#    'TEMP11_HUM11' -> 11bit for Temp and 11bit for humidity
#   Default is: "TEMP11_HUM11"
#htu21d_report_time:
#   Interval in seconds between readings. Default is 30
```

LM75 temperature sensor

LM75/LM75A two wire (I2C) connected temperature sensors. These sensors have a range of -55~125 C, so are usable for e.g. chamber temperature monitoring. They can also function as simple fan/heater controllers.

```
sensor_type: LM75
#i2c_address:
```

```
#   Default is 72 (0x48). Normal range is 72-79 (0x48-0x4F) and the 3
#   low bits of the address are configured via pins on the chip
#   (usually with jumpers or hard wired).
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   See the "common I2C settings" section for a description of the
#   above parameters.
#lm75_report_time:
#   Interval in seconds between readings. Default is 0.8, with minimum
#   0.5.
```

Builtin micro-controller temperature sensor

The atsam, atsamd, and stm32 micro-controllers contain an internal temperature sensor. One can use the "temperature_mcu" sensor to monitor these temperatures.

```
sensor_type: temperature_mcu
#sensor_mcu: mcu
#   The micro-controller to read from. The default is "mcu".
#sensor_temperature1:
#sensor_adc1:
#   Specify the above two parameters (a temperature in Celsius and an
#   ADC value as a float between 0.0 and 1.0) to calibrate the
#   micro-controller temperature. This may improve the reported
#   temperature accuracy on some chips. A typical way to obtain this
#   calibration information is to completely remove power from the
#   printer for a few hours (to ensure it is at the ambient
#   temperature), then power it up and use the QUERY_ADC command to
#   obtain an ADC measurement. Use some other temperature sensor on
#   the printer to find the corresponding ambient temperature. The
#   default is to use the factory calibration data on the
#   micro-controller (if applicable) or the nominal values from the
#   micro-controller specification.
#sensor_temperature2:
#sensor_adc2:
#   If sensor_temperature1/sensor_adc1 is specified then one may also
#   specify sensor_temperature2/sensor_adc2 calibration data. Doing so
#   may provide calibrated "temperature slope" information. The
#   default is to use the factory calibration data on the
#   micro-controller (if applicable) or the nominal values from the
#   micro-controller specification.
```

Host temperature sensor

Temperature from the machine (eg Raspberry Pi) running the host software.

```
sensor_type: temperature_host
#sensor_path:
```

```
#   The path to temperature system file. The default is
#   "/sys/class/thermal/thermal_zone0/temp" which is the temperature
#   system file on a Raspberry Pi computer.
```

DS18B20 temperature sensor

DS18B20 is a 1-wire (w1) digital temperature sensor. Note that this sensor is not intended for use with extruders and heater beds, but rather for monitoring ambient temperature (C). These sensors have range up to 125 C, so are usable for e.g. chamber temperature monitoring. They can also function as simple fan/heater controllers. DS18B20 sensors are only supported on the "host mcu", e.g. the Raspberry Pi. The w1-gpio Linux kernel module must be installed.

```
sensor_type: DS18B20
serial_no:
#   Each 1-wire device has a unique serial number used to identify the
device,
#   usually in the format 28-031674b175ff. This parameter must be
provided.
#   Attached 1-wire devices can be listed using the following Linux
command:
#   ls /sys/bus/w1/devices/
#ds18_report_time:
#   Interval in seconds between readings. Default is 3.0, with a minimum
of 1.0
#sensor_mcu:
#   The micro-controller to read from. Must be the host_mcu
```

Fans

[fan]

Print cooling fan.

```
[fan]
pin:
#   Output pin controlling the fan. This parameter must be provided.
#max_power: 1.0
#   The maximum power (expressed as a value from 0.0 to 1.0) that the
#   pin may be set to. The value 1.0 allows the pin to be set fully
#   enabled for extended periods, while a value of 0.5 would allow the
#   pin to be enabled for no more than half the time. This setting may
#   be used to limit the total power output (over extended periods) to
#   the fan. If this value is less than 1.0 then fan speed requests
#   will be scaled between zero and max_power (for example, if
#   max_power is .9 and a fan speed of 80% is requested then the fan
#   power will be set to 72%). The default is 1.0.
#shutdown_speed: 0
#   The desired fan speed (expressed as a value from 0.0 to 1.0) if
#   the micro-controller software enters an error state. The default
#   is 0.
#cycle_time: 0.010
#   The amount of time (in seconds) for each PWM power cycle to the
#   fan. It is recommended this be 10 milliseconds or greater when
#   using software based PWM. The default is 0.010 seconds.
#hardware_pwm: False
#   Enable this to use hardware PWM instead of software PWM. Most fans
#   do not work well with hardware PWM, so it is not recommended to
#   enable this unless there is an electrical requirement to switch at
#   very high speeds. When using hardware PWM the actual cycle time is
#   constrained by the implementation and may be significantly
#   different than the requested cycle_time. The default is False.
#kick_start_time: 0.100
#   Time (in seconds) to run the fan at full speed when either first
#   enabling or increasing it by more than 50% (helps get the fan
#   spinning). The default is 0.100 seconds.
#off_below: 0.0
#   The minimum input speed which will power the fan (expressed as a
#   value from 0.0 to 1.0). When a speed lower than off_below is
#   requested the fan will instead be turned off. This setting may be
#   used to prevent fan stalls and to ensure kick starts are
#   effective. The default is 0.0.
#
#   This setting should be recalibrated whenever max_power is adjusted.
#   To calibrate this setting, start with off_below set to 0.0 and the
#   fan spinning. Gradually lower the fan speed to determine the lowest
#   input speed which reliably drives the fan without stalls. Set
#   off_below to the duty cycle corresponding to this value (for
#   example, 12% -> 0.12) or slightly higher.
#tachometer_pin:
#   Tachometer input pin for monitoring fan speed. A pullup is generally
#   required. This parameter is optional.
#tachometer_ppr: 2
#   When tachometer_pin is specified, this is the number of pulses per
#   revolution of the tachometer signal. For a BLDC fan this is
#   normally half the number of poles. The default is 2.
#tachometer_poll_interval: 0.0015
#   When tachometer_pin is specified, this is the polling period of the
#   tachometer pin, in seconds. The default is 0.0015, which is fast
#   enough for fans below 10000 RPM at 2 PPR. This must be smaller than
#   30/(tachometer_ppr*rpm), with some margin, where rpm is the
#   maximum speed (in RPM) of the fan.
#enable_pin:
#   Optional pin to enable power to the fan. This can be useful for fans
#   with dedicated PWM inputs. Some of these fans stay on even at 0% PWM
#   input. In such a case, the PWM pin can be used normally, and e.g. a
#   ground-switched FET(standard fan pin) can be used to control power to
#   the fan.
```

[heater_fan]

Heater cooling fans (one may define any number of sections with a "heater_fan" prefix). A "heater fan" is a fan that will be enabled whenever its associated heater is active. By default, a heater_fan has a shutdown_speed equal to max_power.

```
[heater_fan my_nozzle_fan]
#pin:
#max_power:
#shutdown_speed:
#cycle_time:
#hardware_pwm:
#kick_start_time:
#off_below:
#tachometer_pin:
#tachometer_ppr:
#tachometer_poll_interval:
#enable_pin:
#   See the "fan" section for a description of the above parameters.
#heater: extruder
#   Name of the config section defining the heater that this fan is
#   associated with. If a comma separated list of heater names is
#   provided here, then the fan will be enabled when any of the given
#   heaters are enabled. The default is "extruder".
#heater_temp: 50.0
#   A temperature (in Celsius) that the heater must drop below before
#   the fan is disabled. The default is 50 Celsius.
#fan_speed: 1.0
#   The fan speed (expressed as a value from 0.0 to 1.0) that the fan
#   will be set to when its associated heater is enabled. The default
#   is 1.0
```

[controller_fan]

Controller cooling fan (one may define any number of sections with a "controller_fan" prefix). A "controller fan" is a fan that will be enabled whenever its associated heater or its associated stepper driver is active. The fan will stop whenever an idle_timeout is reached to ensure no overheating will occur after deactivating a watched component.

```
[controller_fan my_controller_fan]
#pin:
#max_power:
#shutdown_speed:
#cycle_time:
#hardware_pwm:
#kick_start_time:
#off_below:
#tachometer_pin:
#tachometer_ppr:
#tachometer_poll_interval:
#enable_pin:
```

```
#   See the "fan" section for a description of the above parameters.
#fan_speed: 1.0
#   The fan speed (expressed as a value from 0.0 to 1.0) that the fan
#   will be set to when a heater or stepper driver is active.
#   The default is 1.0
#idle_timeout:
#   The amount of time (in seconds) after a stepper driver or heater
#   was active and the fan should be kept running. The default
#   is 30 seconds.
#idle_speed:
#   The fan speed (expressed as a value from 0.0 to 1.0) that the fan
#   will be set to when a heater or stepper driver was active and
#   before the idle_timeout is reached. The default is fan_speed.
#heater:
#stepper:
#   Name of the config section defining the heater/stepper that this fan
#   is associated with. If a comma separated list of heater/stepper names
#   is provided here, then the fan will be enabled when any of the given
#   heaters/steppers are enabled. The default heater is "extruder", the
#   default stepper is all of them.
```

[temperature_fan]

Temperature-triggered cooling fans (one may define any number of sections with a "temperature_fan" prefix). A "temperature fan" is a fan that will be enabled whenever its associated sensor is above a set temperature. By default, a temperature_fan has a shutdown_speed equal to max_power.

See the command reference for additional information.

```
[temperature_fan my_temp_fan]
#pin:
#max_power:
#shutdown_speed:
#cycle_time:
#hardware_pwm:
#kick_start_time:
#off_below:
#tachometer_pin:
#tachometer_ppr:
#tachometer_poll_interval:
#enable_pin:
#   See the "fan" section for a description of the above parameters.
#sensor_type:
#sensor_pin:
#control:
#max_delta:
#min_temp:
#max_temp:
#   See the "extruder" section for a description of the above parameters.
#pid_Kp:
#pid_Ki:
```

```
#pid_Kd:
#   The proportional (pid_Kp), integral (pid_Ki), and derivative
#   (pid_Kd) settings for the PID feedback control system. Klipper
#   evaluates the PID settings with the following general formula:
#     fan_pwm = max_power - (Kp*e + Ki*integral(e) - Kd*derivative(e)) /
255
#   Where "e" is "target_temperature - measured_temperature" and
#   "fan_pwm" is the requested fan rate with 0.0 being full off and
#   1.0 being full on. The pid_Kp, pid_Ki, and pid_Kd parameters must
#   be provided when the PID control algorithm is enabled.
#pid_deriv_time: 2.0
#   A time value (in seconds) over which temperature measurements will
#   be smoothed when using the PID control algorithm. This may reduce
#   the impact of measurement noise. The default is 2 seconds.
#target_temp: 40.0
#   A temperature (in Celsius) that will be the target temperature.
#   The default is 40 degrees.
#max_speed: 1.0
#   The fan speed (expressed as a value from 0.0 to 1.0) that the fan
#   will be set to when the sensor temperature exceeds the set value.
#   The default is 1.0.
#min_speed: 0.3
#   The minimum fan speed (expressed as a value from 0.0 to 1.0) that
#   the fan will be set to for PID temperature fans.
#   The default is 0.3.
#gcode_id:
#   If set, the temperature will be reported in M105 queries using the
#   given id. The default is to not report the temperature via M105.
```

[fan_generic]

Manually controlled fan (one may define any number of sections with a "fan_generic" prefix). The speed of a manually controlled fan is set with the SET_FAN_SPEED gcode command.

```
[fan_generic extruder_partfan]
#pin:
#max_power:
#shutdown_speed:
#cycle_time:
#hardware_pwm:
#kick_start_time:
#off_below:
#tachometer_pin:
#tachometer_ppr:
#tachometer_poll_interval:
#enable_pin:
#   See the "fan" section for a description of the above parameters.
```

LEDs

[led]

Support for LEDs (and LED strips) controlled via micro-controller PWM pins (one may define any number of sections with an "led" prefix). See the command reference for more information.

```
[led my_led]
#red_pin:
#green_pin:
#blue_pin:
#white_pin:
#   The pin controlling the given LED color. At least one of the above
#   parameters must be provided.
#cycle_time: 0.010
#   The amount of time (in seconds) per PWM cycle. It is recommended
#   this be 10 milliseconds or greater when using software based PWM.
#   The default is 0.010 seconds.
#hardware_pwm: False
#   Enable this to use hardware PWM instead of software PWM. When
#   using hardware PWM the actual cycle time is constrained by the
#   implementation and may be significantly different than the
#   requested cycle_time. The default is False.
#initial_RED: 0.0
#initial_GREEN: 0.0
#initial_BLUE: 0.0
#initial_WHITE: 0.0
#   Sets the initial LED color. Each value should be between 0.0 and
#   1.0. The default for each color is 0.
```

[neopixel]

Neopixel (aka WS2812) LED support (one may define any number of sections with a "neopixel" prefix). See the command reference for more information.

Note that the linux mcu implementation does not currently support directly connected neopixels. The current design using the Linux kernel interface does not allow this scenario because the kernel GPIO interface is not fast enough to provide the required pulse rates.

```
[neopixel my_neopixel]
pin:
#   The pin connected to the neopixel. This parameter must be
#   provided.
#chain_count:
#   The number of Neopixel chips that are "daisy chained" to the
#   provided pin. The default is 1 (which indicates only a single
#   Neopixel is connected to the pin).
#color_order: GRB
#   Set the pixel order required by the LED hardware (using a string
#   containing the letters R, G, B, W with W optional). Alternatively,
#   this may be a comma separated list of pixel orders - one for each
#   LED in the chain. The default is GRB.
```

```
#initial_RED: 0.0
#initial_GREEN: 0.0
#initial_BLUE: 0.0
#initial_WHITE: 0.0
#   See the "led" section for information on these parameters.
```

[dotstar]

Dotstar (aka APA102) LED support (one may define any number of sections with a "dotstar" prefix). See the command reference for more information.

```
[dotstar my_dotstar]
data_pin:
#   The pin connected to the data line of the dotstar. This parameter
#   must be provided.
clock_pin:
#   The pin connected to the clock line of the dotstar. This parameter
#   must be provided.
#chain_count:
#   See the "neopixel" section for information on this parameter.
#initial_RED: 0.0
#initial_GREEN: 0.0
#initial_BLUE: 0.0
#   See the "led" section for information on these parameters.
```

[pca9533]

PCA9533 LED support. The PCA9533 is used on the mightyboard.

```
[pca9533 my_pca9533]
#i2c_address: 98
#   The i2c address that the chip is using on the i2c bus. Use 98 for
#   the PCA9533/1, 99 for the PCA9533/2. The default is 98.
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   See the "common I2C settings" section for a description of the
#   above parameters.
#initial_RED: 0.0
#initial_GREEN: 0.0
#initial_BLUE: 0.0
#initial_WHITE: 0.0
#   See the "led" section for information on these parameters.
```

[pca9632]

PCA9632 LED support. The PCA9632 is used on the FlashForge Dreamer.

```
[pca9632 my_pca9632]
#i2c_address: 98
#   The i2c address that the chip is using on the i2c bus. This may be
#   96, 97, 98, or 99.  The default is 98.
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   See the "common I2C settings" section for a description of the
#   above parameters.
#scl_pin:
#sda_pin:
#   Alternatively, if the pca9632 is not connected to a hardware I2C
#   bus, then one may specify the "clock" (scl_pin) and "data"
#   (sda_pin) pins. The default is to use hardware I2C.
#color_order: RGBW
#   Set the pixel order of the LED (using a string containing the
#   letters R, G, B, W). The default is RGBW.
#initial_RED: 0.0
#initial_GREEN: 0.0
#initial_BLUE: 0.0
#initial_WHITE: 0.0
#   See the "led" section for information on these parameters.
```

Additional servos, buttons, and other pins

[servo]

Servos (one may define any number of sections with a "servo" prefix). The servos may be controlled using the SET_SERVO g-code command. For example: SET_SERVO SERVO=my_servo ANGLE=180

```
[servo my_servo]
pin:
#   PWM output pin controlling the servo. This parameter must be
#   provided.
#maximum_servo_angle: 180
#   The maximum angle (in degrees) that this servo can be set to. The
#   default is 180 degrees.
#minimum_pulse_width: 0.001
#   The minimum pulse width time (in seconds). This should correspond
#   with an angle of 0 degrees. The default is 0.001 seconds.
#maximum_pulse_width: 0.002
#   The maximum pulse width time (in seconds). This should correspond
#   with an angle of maximum_servo_angle. The default is 0.002
#   seconds.
#initial_angle:
#   Initial angle (in degrees) to set the servo to. The default is to
#   not send any signal at startup.
#initial_pulse_width:
#   Initial pulse width time (in seconds) to set the servo to. (This
#   is only valid if initial_angle is not set.) The default is to not
#   send any signal at startup.
```

[gcode_button]

Execute gcode when a button is pressed or released (or when a pin changes state). You can check the state of the button by using `QUERY_BUTTON button=my_gcode_button`.

```
[gcode_button my_gcode_button]
pin:
#   The pin on which the button is connected. This parameter must be
#   provided.
#analog_range:
#   Two comma separated resistances (in Ohms) specifying the minimum
#   and maximum resistance range for the button. If analog_range is
#   provided then the pin must be an analog capable pin. The default
#   is to use digital gpio for the button.
#analog_pullup_resistor:
#   The pullup resistance (in Ohms) when analog_range is specified.
#   The default is 4700 ohms.
#press_gcode:
#   A list of G-Code commands to execute when the button is pressed.
#   G-Code templates are supported. This parameter must be provided.
#release_gcode:
#   A list of G-Code commands to execute when the button is released.
#   G-Code templates are supported. The default is to not run any
#   commands on a button release.
```

[output_pin]

Run-time configurable output pins (one may define any number of sections with an "output_pin" prefix). Pins configured here will be setup as output pins and one may modify them at run-time using "SET_PIN PIN=my_pin VALUE=.1" type extended g-code commands.

```
[output_pin my_pin]
pin:
#   The pin to configure as an output. This parameter must be
#   provided.
#pwm: False
#   Set if the output pin should be capable of pulse-width-modulation.
#   If this is true, the value fields should be between 0 and 1; if it
#   is false the value fields should be either 0 or 1. The default is
#   False.
#static_value:
#   If this is set, then the pin is assigned to this value at startup
#   and the pin can not be changed during runtime. A static pin uses
#   slightly less ram in the micro-controller. The default is to use
#   runtime configuration of pins.
#value:
```

```
#   The value to initially set the pin to during MCU configuration.
#   The default is 0 (for low voltage).
#shutdown_value:
#   The value to set the pin to on an MCU shutdown event. The default
#   is 0 (for low voltage).
#maximum_mcu_duration:
#   The maximum duration a non-shutdown value may be driven by the MCU
#   without an acknowledge from the host.
#   If host can not keep up with an update, the MCU will shutdown
#   and set all pins to their respective shutdown values.
#   Default: 0 (disabled)
#   Usual values are around 5 seconds.
#cycle_time: 0.100
#   The amount of time (in seconds) per PWM cycle. It is recommended
#   this be 10 milliseconds or greater when using software based PWM.
#   The default is 0.100 seconds for pwm pins.
#hardware_pwm: False
#   Enable this to use hardware PWM instead of software PWM. When
#   using hardware PWM the actual cycle time is constrained by the
#   implementation and may be significantly different than the
#   requested cycle_time. The default is False.
#scale:
#   This parameter can be used to alter how the 'value' and
#   'shutdown_value' parameters are interpreted for pwm pins. If
#   provided, then the 'value' parameter should be between 0.0 and
#   'scale'. This may be useful when configuring a PWM pin that
#   controls a stepper voltage reference. The 'scale' can be set to
#   the equivalent stepper amperage if the PWM were fully enabled, and
#   then the 'value' parameter can be specified using the desired
#   amperage for the stepper. The default is to not scale the 'value'
#   parameter.
```

[static_digital_output]

Statically configured digital output pins (one may define any number of sections with a "static_digital_output" prefix). Pins configured here will be setup as a GPIO output during MCU configuration. They can not be changed at run-time.

```
[static_digital_output my_output_pins]
pins:
#   A comma separated list of pins to be set as GPIO output pins. The
#   pin will be set to a high level unless the pin name is prefaced
#   with "!". This parameter must be provided.
```

[multi_pin]

Multiple pin outputs (one may define any number of sections with a "multi_pin" prefix). A multi_pin output creates an internal pin alias that can modify multiple output pins each time the alias pin is set. For example, one could define a "[multi_pin my_fan]" object containing two pins and then set "pin=multi_pin:my_fan" in the "[fan]" section - on each fan change both output pins would be updated. These aliases may not be used with stepper motor pins.

```
[multi_pin my_multi_pin]
pins:
#   A comma separated list of pins associated with this alias. This
#   parameter must be provided.
```

TMC stepper driver configuration

Configuration of Trinamic stepper motor drivers in UART/SPI mode. Additional information is in the TMC Drivers guide and in the command reference.

[tmc2130]

Configure a TMC2130 stepper motor driver via SPI bus. To use this feature, define a config section with a "tmc2130" prefix followed by the name of the corresponding stepper config section (for example, "[tmc2130 stepper_x]").

```
[tmc2130 stepper_x]
cs_pin:
#   The pin corresponding to the TMC2130 chip select line. This pin
#   will be set to low at the start of SPI messages and raised to high
#   after the message completes. This parameter must be provided.
#spi_speed:
#spi_bus:
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   See the "common SPI settings" section for a description of the
#   above parameters.
#chain_position:
#chain_length:
#   These parameters configure an SPI daisy chain. The two parameters
#   define the stepper position in the chain and the total chain length.
#   Position 1 corresponds to the stepper that connects to the MOSI
signal.
#   The default is to not use an SPI daisy chain.
#interpolate: True
#   If true, enable step interpolation (the driver will internally
#   step at a rate of 256 micro-steps). This interpolation does
#   introduce a small systemic positional deviation - see
#   TMC_Drivers.md for details. The default is True.
run_current:
#   The amount of current (in amps RMS) to configure the driver to use
#   during stepper movement. This parameter must be provided.
#hold_current:
#   The amount of current (in amps RMS) to configure the driver to use
#   when the stepper is not moving. Setting a hold_current is not
#   recommended (see TMC_Drivers.md for details). The default is to
#   not reduce the current.
#sense_resistor: 0.110
#   The resistance (in ohms) of the motor sense resistor. The default
#   is 0.110 ohms.
#stealthchop_threshold: 0
#   The velocity (in mm/s) to set the "stealthChop" threshold to. When
#   set, "stealthChop" mode will be enabled if the stepper motor
#   velocity is below this value. The default is 0, which disables
#   "stealthChop" mode.
#driver_MSLUT0: 2863314260
#driver_MSLUT1: 1251300522
#driver_MSLUT2: 608774441
#driver_MSLUT3: 269500962
#driver_MSLUT4: 4227858431
#driver_MSLUT5: 3048961917
#driver_MSLUT6: 1227445590
#driver_MSLUT7: 4211234
#driver_W0: 2
#driver_W1: 1
#driver_W2: 1
#driver_W3: 1
#driver_X1: 128
#driver_X2: 255
#driver_X3: 255
#driver_START_SIN: 0
#driver_START_SIN90: 247
#   These fields control the Microstep Table registers directly. The
optimal
#   wave table is specific to each motor and might vary with current. An
#   optimal configuration will have minimal print artifacts caused by
#   non-linear stepper movement. The values specified above are the
default
#   values used by the driver. The value must be specified as a decimal
integer
#   (hex form is not supported). In order to compute the wave table
fields,
#   see the tmc2130 "Calculation Sheet" from the Trinamic website.
#driver_IHOLDDELAY: 8
#driver_TPOWERDOWN: 0
#driver_TBL: 1
#driver_TOFF: 4
#driver_HEND: 7
#driver_HSTRT: 0
#driver_PWM_AUTOSCALE: True
#driver_PWM_FREQ: 1
#driver_PWM_GRAD: 4
#driver_PWM_AMPL: 128
#driver_SGT: 0
#   Set the given register during the configuration of the TMC2130
#   chip. This may be used to set custom motor parameters. The
#   defaults for each parameter are next to the parameter name in the
#   above list.
#diag0_pin:
```

```
#diag1_pin:
#   The micro-controller pin attached to one of the DIAG lines of the
#   TMC2130 chip. Only a single diag pin should be specified. The pin
#   is "active low" and is thus normally prefaced with "^!". Setting
#   this creates a "tmc2130_stepper_x:virtual_endstop" virtual pin
#   which may be used as the stepper's endstop_pin. Doing this enables
#   "sensorless homing". (Be sure to also set driver_SGT to an
#   appropriate sensitivity value.) The default is to not enable
#   sensorless homing.
```

[tmc2208]

Configure a TMC2208 (or TMC2224) stepper motor driver via single wire UART. To use this feature, define a config section with a "tmc2208" prefix followed by the name of the corresponding stepper config section (for example, "[tmc2208 stepper_x]").

```
[tmc2208 stepper_x]
uart_pin:
#   The pin connected to the TMC2208 PDN_UART line. This parameter
#   must be provided.
#tx_pin:
#   If using separate receive and transmit lines to communicate with
#   the driver then set uart_pin to the receive pin and tx_pin to the
#   transmit pin. The default is to use uart_pin for both reading and
#   writing.
#select_pins:
#   A comma separated list of pins to set prior to accessing the
#   tmc2208 UART. This may be useful for configuring an analog mux for
#   UART communication. The default is to not configure any pins.
#interpolate: True
#   If true, enable step interpolation (the driver will internally
#   step at a rate of 256 micro-steps). This interpolation does
#   introduce a small systemic positional deviation - see
#   TMC_Drivers.md for details. The default is True.
run_current:
#   The amount of current (in amps RMS) to configure the driver to use
#   during stepper movement. This parameter must be provided.
#hold_current:
#   The amount of current (in amps RMS) to configure the driver to use
#   when the stepper is not moving. Setting a hold_current is not
#   recommended (see TMC_Drivers.md for details). The default is to
#   not reduce the current.
#sense_resistor: 0.110
#   The resistance (in ohms) of the motor sense resistor. The default
#   is 0.110 ohms.
#stealthchop_threshold: 0
#   The velocity (in mm/s) to set the "stealthChop" threshold to. When
#   set, "stealthChop" mode will be enabled if the stepper motor
#   velocity is below this value. The default is 0, which disables
#   "stealthChop" mode.
#driver_IHOLDDELAY: 8
```

```
#driver_TPOWERDOWN: 20
#driver_TBL: 2
#driver_TOFF: 3
#driver_HEND: 0
#driver_HSTRT: 5
#driver_PWM_AUTOGRAD: True
#driver_PWM_AUTOSCALE: True
#driver_PWM_LIM: 12
#driver_PWM_REG: 8
#driver_PWM_FREQ: 1
#driver_PWM_GRAD: 14
#driver_PWM_OFS: 36
#   Set the given register during the configuration of the TMC2208
#   chip. This may be used to set custom motor parameters. The
#   defaults for each parameter are next to the parameter name in the
#   above list.
```

[tmc2209]

Configure a TMC2209 stepper motor driver via single wire UART. To use this feature, define a config section with a "tmc2209" prefix followed by the name of the corresponding stepper config section (for example, "[tmc2209 stepper_x]").

```
[tmc2209 stepper_x]
uart_pin:
#tx_pin:
#select_pins:
#interpolate: True
run_current:
#hold_current:
#sense_resistor: 0.110
#stealthchop_threshold: 0
#   See the "tmc2208" section for the definition of these parameters.
#uart_address:
#   The address of the TMC2209 chip for UART messages (an integer
#   between 0 and 3). This is typically used when multiple TMC2209
#   chips are connected to the same UART pin. The default is zero.
#driver_IHOLDDELAY: 8
#driver_TPOWERDOWN: 20
#driver_TBL: 2
#driver_TOFF: 3
#driver_HEND: 0
#driver_HSTRT: 5
#driver_PWM_AUTOGRAD: True
#driver_PWM_AUTOSCALE: True
#driver_PWM_LIM: 12
#driver_PWM_REG: 8
#driver_PWM_FREQ: 1
#driver_PWM_GRAD: 14
#driver_PWM_OFS: 36
#driver_SGTHRS: 0
```

```
#   Set the given register during the configuration of the TMC2209
#   chip. This may be used to set custom motor parameters. The
#   defaults for each parameter are next to the parameter name in the
#   above list.
#diag_pin:
#   The micro-controller pin attached to the DIAG line of the TMC2209
#   chip. The pin is normally prefaced with "^" to enable a pullup.
#   Setting this creates a "tmc2209_stepper_x:virtual_endstop" virtual
#   pin which may be used as the stepper's endstop_pin. Doing this
#   enables "sensorless homing". (Be sure to also set driver_SGTHRS to
#   an appropriate sensitivity value.) The default is to not enable
#   sensorless homing.
```

[tmc2660]

Configure a TMC2660 stepper motor driver via SPI bus. To use this feature, define a config section with a tmc2660 prefix followed by the name of the corresponding stepper config section (for example, "[tmc2660 stepper_x]").

```
[tmc2660 stepper_x]
cs_pin:
#   The pin corresponding to the TMC2660 chip select line. This pin
#   will be set to low at the start of SPI messages and set to high
#   after the message transfer completes. This parameter must be
#   provided.
#spi_speed: 4000000
#   SPI bus frequency used to communicate with the TMC2660 stepper
#   driver. The default is 4000000.
#spi_bus:
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   See the "common SPI settings" section for a description of the
#   above parameters.
#interpolate: True
#   If true, enable step interpolation (the driver will internally
#   step at a rate of 256 micro-steps). This only works if microsteps
#   is set to 16. Interpolation does introduce a small systemic
#   positional deviation - see TMC_Drivers.md for details. The default
#   is True.
run_current:
#   The amount of current (in amps RMS) used by the driver during
#   stepper movement. This parameter must be provided.
#sense_resistor:
#   The resistance (in ohms) of the motor sense resistor. This
#   parameter must be provided.
#idle_current_percent: 100
#   The percentage of the run_current the stepper driver will be
#   lowered to when the idle timeout expires (you need to set up the
#   timeout using a [idle_timeout] config section). The current will
#   be raised again once the stepper has to move again. Make sure to
#   set this to a high enough value such that the steppers do not lose
#   their position. There is also small delay until the current is
#   raised again, so take this into account when commanding fast moves
#   while the stepper is idling. The default is 100 (no reduction).
#driver_TBL: 2
#driver_RNDTF: 0
#driver_HDEC: 0
#driver_CHM: 0
#driver_HEND: 3
#driver_HSTRT: 3
#driver_TOFF: 4
#driver_SEIMIN: 0
#driver_SEDN: 0
#driver_SEMAX: 0
#driver_SEUP: 0
#driver_SEMIN: 0
#driver_SFILT: 0
#driver_SGT: 0
#driver_SLPH: 0
#driver_SLPL: 0
#driver_DISS2G: 0
#driver_TS2G: 3
#   Set the given parameter during the configuration of the TMC2660
#   chip. This may be used to set custom driver parameters. The
#   defaults for each parameter are next to the parameter name in the
#   list above. See the TMC2660 datasheet about what each parameter
#   does and what the restrictions on parameter combinations are. Be
#   especially aware of the CHOPCONF register, where setting CHM to
#   either zero or one will lead to layout changes (the first bit of
#   HDEC) is interpreted as the MSB of HSTRT in this case).
```

[tmc5160]

Configure a TMC5160 stepper motor driver via SPI bus. To use this feature, define a config section with a "tmc5160" prefix followed by the name of the corresponding stepper config section (for example, "[tmc5160 stepper_x]").

```
[tmc5160 stepper_x]
cs_pin:
#   The pin corresponding to the TMC5160 chip select line. This pin
#   will be set to low at the start of SPI messages and raised to high
#   after the message completes. This parameter must be provided.
#spi_speed:
#spi_bus:
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   See the "common SPI settings" section for a description of the
#   above parameters.
#chain_position:
#chain_length:
```

```
#   These parameters configure an SPI daisy chain. The two parameters
#   define the stepper position in the chain and the total chain length.
#   Position 1 corresponds to the stepper that connects to the MOSI
signal.
#   The default is to not use an SPI daisy chain.
#interpolate: True
#   If true, enable step interpolation (the driver will internally
#   step at a rate of 256 micro-steps). The default is True.
run_current:
#   The amount of current (in amps RMS) to configure the driver to use
#   during stepper movement. This parameter must be provided.
#hold_current:
#   The amount of current (in amps RMS) to configure the driver to use
#   when the stepper is not moving. Setting a hold_current is not
#   recommended (see TMC_Drivers.md for details). The default is to
#   not reduce the current.
#sense_resistor: 0.075
#   The resistance (in ohms) of the motor sense resistor. The default
#   is 0.075 ohms.
#stealthchop_threshold: 0
#   The velocity (in mm/s) to set the "stealthChop" threshold to. When
#   set, "stealthChop" mode will be enabled if the stepper motor
#   velocity is below this value. The default is 0, which disables
#   "stealthChop" mode.
#driver_MSLUT0: 2863314260
#driver_MSLUT1: 1251300522
#driver_MSLUT2: 608774441
#driver_MSLUT3: 269500962
#driver_MSLUT4: 4227858431
#driver_MSLUT5: 3048961917
#driver_MSLUT6: 1227445590
#driver_MSLUT7: 4211234
#driver_W0: 2
#driver_W1: 1
#driver_W2: 1
#driver_W3: 1
#driver_X1: 128
#driver_X2: 255
#driver_X3: 255
#driver_START_SIN: 0
#driver_START_SIN90: 247
#   These fields control the Microstep Table registers directly. The
optimal
#   wave table is specific to each motor and might vary with current. An
#   optimal configuration will have minimal print artifacts caused by
#   non-linear stepper movement. The values specified above are the
default
#   values used by the driver. The value must be specified as a decimal
integer
#   (hex form is not supported). In order to compute the wave table
fields,
#   see the tmc2130 "Calculation Sheet" from the Trinamic website.
#driver_IHOLDDELAY: 6
#driver_TPOWERDOWN: 10
```

```
#driver_TBL: 2
#driver_TOFF: 3
#driver_HEND: 2
#driver_HSTRT: 5
#driver_FD3: 0
#driver_TPFD: 4
#driver_CHM: 0
#driver_VHIGHFS: 0
#driver_VHIGHCHM: 0
#driver_DISS2G: 0
#driver_DISS2VS: 0
#driver_PWM_AUTOSCALE: True
#driver_PWM_AUTOGRAD: True
#driver_PWM_FREQ: 0
#driver_FREEWHEEL: 0
#driver_PWM_GRAD: 0
#driver_PWM_OFS: 30
#driver_PWM_REG: 4
#driver_PWM_LIM: 12
#driver_SGT: 0
#driver_SEMIN: 0
#driver_SEUP: 0
#driver_SEMAX: 0
#driver_SEDN: 0
#driver_SEIMIN: 0
#driver_SFILT: 0
#   Set the given register during the configuration of the TMC5160
#   chip. This may be used to set custom motor parameters. The
#   defaults for each parameter are next to the parameter name in the
#   above list.
#diag0_pin:
#diag1_pin:
#   The micro-controller pin attached to one of the DIAG lines of the
#   TMC5160 chip. Only a single diag pin should be specified. The pin
#   is "active low" and is thus normally prefaced with "^!". Setting
#   this creates a "tmc5160_stepper_x:virtual_endstop" virtual pin
#   which may be used as the stepper's endstop_pin. Doing this enables
#   "sensorless homing". (Be sure to also set driver_SGT to an
#   appropriate sensitivity value.) The default is to not enable
#   sensorless homing.
```

Run-time stepper motor current configuration

[ad5206]

Statically configured AD5206 digipots connected via SPI bus (one may define any number of sections with an "ad5206" prefix).

```
[ad5206 my_digipot]
enable_pin:
#   The pin corresponding to the AD5206 chip select line. This pin
#   will be set to low at the start of SPI messages and raised to high
#   after the message completes. This parameter must be provided.
#spi_speed:
#spi_bus:
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   See the "common SPI settings" section for a description of the
#   above parameters.
#channel_1:
#channel_2:
#channel_3:
#channel_4:
#channel_5:
#channel_6:
#   The value to statically set the given AD5206 channel to. This is
#   typically set to a number between 0.0 and 1.0 with 1.0 being the
#   highest resistance and 0.0 being the lowest resistance. However,
#   the range may be changed with the 'scale' parameter (see below).
#   If a channel is not specified then it is left unconfigured.
#scale:
#   This parameter can be used to alter how the 'channel_x' parameters
#   are interpreted. If provided, then the 'channel_x' parameters
#   should be between 0.0 and 'scale'. This may be useful when the
#   AD5206 is used to set stepper voltage references. The 'scale' can
#   be set to the equivalent stepper amperage if the AD5206 were at
#   its highest resistance, and then the 'channel_x' parameters can be
#   specified using the desired amperage value for the stepper. The
#   default is to not scale the 'channel_x' parameters.
```

[mcp4451]

Statically configured MCP4451 digipot connected via I2C bus (one may define any number of sections with an "mcp4451" prefix).

```
[mcp4451 my_digipot]
i2c_address:
#   The i2c address that the chip is using on the i2c bus. This
#   parameter must be provided.
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   See the "common I2C settings" section for a description of the
#   above parameters.
#wiper_0:
#wiper_1:
#wiper_2:
#wiper_3:
#   The value to statically set the given MCP4451 "wiper" to. This is
#   typically set to a number between 0.0 and 1.0 with 1.0 being the
#   highest resistance and 0.0 being the lowest resistance. However,
#   the range may be changed with the 'scale' parameter (see below).
#   If a wiper is not specified then it is left unconfigured.
#scale:
#   This parameter can be used to alter how the 'wiper_x' parameters
#   are interpreted. If provided, then the 'wiper_x' parameters should
#   be between 0.0 and 'scale'. This may be useful when the MCP4451 is
#   used to set stepper voltage references. The 'scale' can be set to
#   the equivalent stepper amperage if the MCP4451 were at its highest
#   resistance, and then the 'wiper_x' parameters can be specified
#   using the desired amperage value for the stepper. The default is
#   to not scale the 'wiper_x' parameters.
```

[mcp4728]

Statically configured MCP4728 digital-to-analog converter connected via I2C bus (one may define any number of sections with an "mcp4728" prefix).

```
[mcp4728 my_dac]
#i2c_address: 96
#   The i2c address that the chip is using on the i2c bus. The default
#   is 96.
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   See the "common I2C settings" section for a description of the
#   above parameters.
#channel_a:
#channel_b:
#channel_c:
#channel_d:
#   The value to statically set the given MCP4728 channel to. This is
#   typically set to a number between 0.0 and 1.0 with 1.0 being the
#   highest voltage (2.048V) and 0.0 being the lowest voltage.
#   However, the range may be changed with the 'scale' parameter (see
#   below). If a channel is not specified then it is left
#   unconfigured.
#scale:
#   This parameter can be used to alter how the 'channel_x' parameters
#   are interpreted. If provided, then the 'channel_x' parameters
#   should be between 0.0 and 'scale'. This may be useful when the
#   MCP4728 is used to set stepper voltage references. The 'scale' can
#   be set to the equivalent stepper amperage if the MCP4728 were at
#   its highest voltage (2.048V), and then the 'channel_x' parameters
#   can be specified using the desired amperage value for the
#   stepper. The default is to not scale the 'channel_x' parameters.
```

[mcp4018]

Statically configured MCP4018 digipot connected via two gpio "bit banging" pins (one may define any number of sections with an "mcp4018" prefix).

```
[mcp4018 my_digipot]
scl_pin:
#   The SCL "clock" pin. This parameter must be provided.
sda_pin:
#   The SDA "data" pin. This parameter must be provided.
wiper:
#   The value to statically set the given MCP4018 "wiper" to. This is
#   typically set to a number between 0.0 and 1.0 with 1.0 being the
#   highest resistance and 0.0 being the lowest resistance. However,
#   the range may be changed with the 'scale' parameter (see below).
#   This parameter must be provided.
#scale:
#   This parameter can be used to alter how the 'wiper' parameter is
#   interpreted. If provided, then the 'wiper' parameter should be
#   between 0.0 and 'scale'. This may be useful when the MCP4018 is
#   used to set stepper voltage references. The 'scale' can be set to
#   the equivalent stepper amperage if the MCP4018 is at its highest
#   resistance, and then the 'wiper' parameter can be specified using
#   the desired amperage value for the stepper. The default is to not
#   scale the 'wiper' parameter.
```

Display support

[display]

Support for a display attached to the micro-controller.

```
[display]
lcd_type:
#   The type of LCD chip in use. This may be "hd44780", "hd44780_spi",
#   "st7920", "emulated_st7920", "uc1701", "ssd1306", or "sh1106".
#   See the display sections below for information on each type and
#   additional parameters they provide. This parameter must be
#   provided.
#display_group:
#   The name of the display_data group to show on the display. This
#   controls the content of the screen (see the "display_data" section
#   for more information). The default is _default_20x4 for hd44780
#   displays and _default_16x4 for other displays.
#menu_timeout:
#   Timeout for menu. Being inactive this amount of seconds will
#   trigger menu exit or return to root menu when having autorun
#   enabled. The default is 0 seconds (disabled)
#menu_root:
#  Name of the main menu section to show when clicking the encoder
#   on the home screen. The defaults is __main, and this shows the
#   the default menus as defined in klippy/extras/display/menu.cfg
#menu_reverse_navigation:
#   When enabled it will reverse up and down directions for list
#   navigation. The default is False. This parameter is optional.
```

```
#encoder_pins:
#   The pins connected to encoder. 2 pins must be provided when using
#   encoder. This parameter must be provided when using menu.
#encoder_steps_per_detent:
#   How many steps the encoder emits per detent ("click"). If the
#   encoder takes two detents to move between entries or moves two
#   entries from one detent, try changing this. Allowed values are 2
#   (half-stepping) or 4 (full-stepping). The default is 4.
#click_pin:
#   The pin connected to 'enter' button or encoder 'click'. This
#   parameter must be provided when using menu. The presence of an
#   'analog_range_click_pin' config parameter turns this parameter
#   from digital to analog.
#back_pin:
#   The pin connected to 'back' button. This parameter is optional,
#   menu can be used without it. The presence of an
#   'analog_range_back_pin' config parameter turns this parameter from
#   digital to analog.
#up_pin:
#   The pin connected to 'up' button. This parameter must be provided
#   when using menu without encoder. The presence of an
#   'analog_range_up_pin' config parameter turns this parameter from
#   digital to analog.
#down_pin:
#   The pin connected to 'down' button. This parameter must be
#   provided when using menu without encoder. The presence of an
#   'analog_range_down_pin' config parameter turns this parameter from
#   digital to analog.
#kill_pin:
#   The pin connected to 'kill' button. This button will call
#   emergency stop. The presence of an 'analog_range_kill_pin' config
#   parameter turns this parameter from digital to analog.
#analog_pullup_resistor: 4700
#   The resistance (in ohms) of the pullup attached to the analog
#   button. The default is 4700 ohms.
#analog_range_click_pin:
#   The resistance range for a 'enter' button. Range minimum and
#   maximum comma-separated values must be provided when using analog
#   button.
#analog_range_back_pin:
#   The resistance range for a 'back' button. Range minimum and
#   maximum comma-separated values must be provided when using analog
#   button.
#analog_range_up_pin:
#   The resistance range for a 'up' button. Range minimum and maximum
#   comma-separated values must be provided when using analog button.
#analog_range_down_pin:
#   The resistance range for a 'down' button. Range minimum and
#   maximum comma-separated values must be provided when using analog
#   button.
#analog_range_kill_pin:
#   The resistance range for a 'kill' button. Range minimum and
#   maximum comma-separated values must be provided when using analog
#   button.
```

hd44780 display

Information on configuring hd44780 displays (which is used in "RepRapDiscount 2004 Smart Controller" type displays).

```
[display]
lcd_type: hd44780
#   Set to "hd44780" for hd44780 displays.
rs_pin:
e_pin:
d4_pin:
d5_pin:
d6_pin:
d7_pin:
#   The pins connected to an hd44780 type lcd. These parameters must
#   be provided.
#hd44780_protocol_init: True
#   Perform 8-bit/4-bit protocol initialization on an hd44780 display.
#   This is necessary on real hd44780 devices. However, one may need
#   to disable this on some "clone" devices. The default is True.
#line_length:
#   Set the number of characters per line for an hd44780 type lcd.
#   Possible values are 20 (default) and 16. The number of lines is
#   fixed to 4.
...
```

hd44780_spi display

Information on configuring an hd44780_spi display - a 20x04 display controlled via a hardware "shift register" (which is used in mightyboard based printers).

```
[display]
lcd_type: hd44780_spi
#   Set to "hd44780_spi" for hd44780_spi displays.
latch_pin:
spi_software_sclk_pin:
spi_software_mosi_pin:
spi_software_miso_pin:
#   The pins connected to the shift register controlling the display.
#   The spi_software_miso_pin needs to be set to an unused pin of the
#   printer mainboard as the shift register does not have a MISO pin,
#   but the software spi implementation requires this pin to be
#   configured.
#hd44780_protocol_init: True
#   Perform 8-bit/4-bit protocol initialization on an hd44780 display.
#   This is necessary on real hd44780 devices. However, one may need
#   to disable this on some "clone" devices. The default is True.
```

```
#line_length:
#   Set the number of characters per line for an hd44780 type lcd.
#   Possible values are 20 (default) and 16. The number of lines is
#   fixed to 4.
...
```

st7920 display

Information on configuring st7920 displays (which is used in "RepRapDiscount 12864 Full Graphic Smart Controller" type displays).

```
[display]
lcd_type: st7920
#   Set to "st7920" for st7920 displays.
cs_pin:
sclk_pin:
sid_pin:
#   The pins connected to an st7920 type lcd. These parameters must be
#   provided.
...
```

emulated_st7920 display

Information on configuring an emulated st7920 display - found in some "2.4 inch touchscreen devices" and similar.

```
[display]
lcd_type: emulated_st7920
#   Set to "emulated_st7920" for emulated_st7920 displays.
en_pin:
spi_software_sclk_pin:
spi_software_mosi_pin:
spi_software_miso_pin:
#   The pins connected to an emulated_st7920 type lcd. The en_pin
#   corresponds to the cs_pin of the st7920 type lcd,
#   spi_software_sclk_pin corresponds to sclk_pin and
#   spi_software_mosi_pin corresponds to sid_pin. The
#   spi_software_miso_pin needs to be set to an unused pin of the
#   printer mainboard as the st7920 as no MISO pin but the software
#   spi implementation requires this pin to be configured.
...
```

uc1701 display

Information on configuring uc1701 displays (which is used in "MKS Mini 12864" type displays).

```
[display]
lcd_type: uc1701
#   Set to "uc1701" for uc1701 displays.
cs_pin:
a0_pin:
#   The pins connected to a uc1701 type lcd. These parameters must be
#   provided.
#rst_pin:
#   The pin connected to the "rst" pin on the lcd. If it is not
#   specified then the hardware must have a pull-up on the
#   corresponding lcd line.
#contrast:
#   The contrast to set. The value may range from 0 to 63 and the
#   default is 40.
...
```

ssd1306 and sh1106 displays

Information on configuring ssd1306 and sh1106 displays.

```
[display]
lcd_type:
#   Set to either "ssd1306" or "sh1106" for the given display type.
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   Optional parameters available for displays connected via an i2c
#   bus. See the "common I2C settings" section for a description of
#   the above parameters.
#cs_pin:
#dc_pin:
#spi_speed:
#spi_bus:
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   The pins connected to the lcd when in "4-wire" spi mode. See the
#   "common SPI settings" section for a description of the parameters
#   that start with "spi_". The default is to use i2c mode for the
#   display.
#reset_pin:
#   A reset pin may be specified on the display. If it is not
#   specified then the hardware must have a pull-up on the
#   corresponding lcd line.
#contrast:
#   The contrast to set. The value may range from 0 to 256 and the
#   default is 239.
#vcomh: 0
#   Set the Vcomh value on the display. This value is associated with
#   a "smearing" effect on some OLED displays. The value may range
#   from 0 to 63. Default is 0.
#invert: False
#   TRUE inverts the pixels on certain OLED displays.  The default is
#   False.
#x_offset: 0
#   Set the horizontal offset value on SH1106 displays. The default is
#   0.
...
```

[display_data]

Support for displaying custom data on an lcd screen. One may create any number of display groups and any number of data items under those groups. The display will show all the data items for a given group if the display_group option in the [display] section is set to the given group name.

A default set of display groups are automatically created. One can replace or extend these display_data items by overriding the defaults in the main printer.cfg config file.

```
[display_data my_group_name my_data_name]
position:
#   Comma separated row and column of the display position that should
#   be used to display the information. This parameter must be
#   provided.
text:
#   The text to show at the given position. This field is evaluated
#   using command templates (see docs/Command_Templates.md). This
#   parameter must be provided.
```

[display_template]

Display data text "macros" (one may define any number of sections with a display_template prefix). See the command templates document for information on template evaluation.

This feature allows one to reduce repetitive definitions in display_data sections. One may use the builtin `render()` function in display_data sections to evaluate a template. For example, if one were to define `[display_template my_template]` then one could use `{ render('my_template') }` in a display_data section.

This feature can also be used for continuous LED updates using the SET_LED_TEMPLATE command.

```
[display_template my_template_name]
#param_<name>:
#   One may specify any number of options with a "param_" prefix. The
#   given name will be assigned the given value (parsed as a Python
#   literal) and will be available during macro expansion. If the
#   parameter is passed in the call to render() then that value will
#   be used during macro expansion. For example, a config with
#   "param_speed = 75" might have a caller with
#   "render('my_template_name', param_speed=80)". Parameter names may
#   not use upper case characters.
text:
#   The text to return when the this template is rendered. This field
#   is evaluated using command templates (see
#   docs/Command_Templates.md). This parameter must be provided.
```

[display_glyph]

Display a custom glyph on displays that support it. The given name will be assigned the given display data which can then be referenced in the display templates by their name surrounded by two "tilde" symbols i.e. `~my_display_glyph~`

See sample-glyphs.cfg for some examples.

```
[display_glyph my_display_glyph]
#data:
#   The display data, stored as 16 lines consisting of 16 bits (1 per
#   pixel) where '.' is a blank pixel and '*' is an on pixel (e.g.,
#   "****************" to display a solid horizontal line).
#   Alternatively, one can use '0' for a blank pixel and '1' for an on
#   pixel. Put each display line into a separate config line. The
#   glyph must consist of exactly 16 lines with 16 bits each. This
#   parameter is optional.
#hd44780_data:
#   Glyph to use on 20x4 hd44780 displays. The glyph must consist of
#   exactly 8 lines with 5 bits each. This parameter is optional.
#hd44780_slot:
#   The hd44780 hardware index (0..7) to store the glyph at. If
#   multiple distinct images use the same slot then make sure to only
#   use one of those images in any given screen. This parameter is
#   required if hd44780_data is specified.
```

[display my_extra_display]

If a primary [display] section has been defined in printer.cfg as shown above it is possible to define multiple auxiliary displays. Note that auxiliary displays do not currently support menu functionality, thus they do not support the "menu" options or button configuration.

```
[display my_extra_display]
# See the "display" section for available parameters.
```

[menu]

Customizable lcd display menus.

A default set of menus are automatically created. One can replace or extend the menu by overriding the defaults in the main printer.cfg config file.

See the command template document for information on menu attributes available during template rendering.

```
# Common parameters available for all menu config sections.
#[menu __some_list __some_name]
#type: disabled
#   Permanently disabled menu element, only required attribute is 'type'.
#   Allows you to easily disable/hide existing menu items.

#[menu some_name]
#type:
#   One of command, input, list, text:
#       command - basic menu element with various script triggers
#       input   - same like 'command' but has value changing capabilities.
#                 Press will start/stop edit mode.
#       list    - it allows for menu items to be grouped together in a
#                 scrollable list.  Add to the list by creating menu
#                 configurations using "some_list" as a prefix - for
#                 example: [menu some_list some_item_in_the_list]
#       vsdlist - same as 'list' but will append files from virtual sdcard
#                 (will be removed in the future)
#name:
#   Name of menu item - evaluated as a template.
#enable:
#   Template that evaluates to True or False.
#index:
#   Position where an item needs to be inserted in list. By default
#   the item is added at the end.

#[menu some_list]
#type: list
#name:
#enable:
#   See above for a description of these parameters.

#[menu some_list some_command]
#type: command
#name:
#enable:
#   See above for a description of these parameters.
#gcode:
#   Script to run on button click or long click. Evaluated as a
#   template.

#[menu some_list some_input]
#type: input
#name:
#enable:
#   See above for a description of these parameters.
```

```
#input:
#   Initial value to use when editing - evaluated as a template.
#   Result must be float.
#input_min:
#   Minimum value of range - evaluated as a template. Default -99999.
#input_max:
#   Maximum value of range - evaluated as a template. Default 99999.
#input_step:
#   Editing step - Must be a positive integer or float value. It has
#   internal fast rate step. When "(input_max - input_min) /
#   input_step > 100" then fast rate step is 10 * input_step else fast
#   rate step is same input_step.
#realtime:
#   This attribute accepts static boolean value. When enabled then
#   gcode script is run after each value change. The default is False.
#gcode:
#   Script to run on button click, long click or value change.
#   Evaluated as a template. The button click will trigger the edit
#   mode start or end.
```

Filament sensors

[filament_switch_sensor]

Filament Switch Sensor. Support for filament insert and runout detection using a switch sensor, such as an endstop switch.

See the command reference for more information.

```
[filament_switch_sensor my_sensor]
#pause_on_runout: True
#   When set to True, a PAUSE will execute immediately after a runout
#   is detected. Note that if pause_on_runout is False and the
#   runout_gcode is omitted then runout detection is disabled. Default
#   is True.
#runout_gcode:
#   A list of G-Code commands to execute after a filament runout is
#   detected. See docs/Command_Templates.md for G-Code format. If
#   pause_on_runout is set to True this G-Code will run after the
#   PAUSE is complete. The default is not to run any G-Code commands.
#insert_gcode:
#   A list of G-Code commands to execute after a filament insert is
#   detected. See docs/Command_Templates.md for G-Code format. The
#   default is not to run any G-Code commands, which disables insert
#   detection.
#event_delay: 3.0
#   The minimum amount of time in seconds to delay between events.
#   Events triggered during this time period will be silently
#   ignored. The default is 3 seconds.
#pause_delay: 0.5
#   The amount of time to delay, in seconds, between the pause command
#   dispatch and execution of the runout_gcode. It may be useful to
#   increase this delay if OctoPrint exhibits strange pause behavior.
#   Default is 0.5 seconds.
#switch_pin:
#   The pin on which the switch is connected. This parameter must be
#   provided.
```

[filament_motion_sensor]

Filament Motion Sensor. Support for filament insert and runout detection using an encoder that toggles the output pin during filament movement through the sensor.

See the command reference for more information.

```
[filament_motion_sensor my_sensor]
detection_length: 7.0
#   The minimum length of filament pulled through the sensor to trigger
#   a state change on the switch_pin
#   Default is 7 mm.
extruder:
#   The name of the extruder section this sensor is associated with.
#   This parameter must be provided.
switch_pin:
#pause_on_runout:
#runout_gcode:
#insert_gcode:
#event_delay:
#pause_delay:
#   See the "filament_switch_sensor" section for a description of the
#   above parameters.
```

[tsl1401cl_filament_width_sensor]

TSLl401CL Based Filament Width Sensor. See the guide for more information.

```
[tsl1401cl_filament_width_sensor]
#pin:
#default_nominal_filament_diameter: 1.75 # (mm)
#   Maximum allowed filament diameter difference as mm.
#max_difference: 0.2
#   The distance from sensor to the melting chamber as mm.
#measurement_delay: 100
```

[hall_filament_width_sensor]

Hall filament width sensor (see Hall Filament Width Sensor).

```
[hall_filament_width_sensor]
adc1:
adc2:
#   Analog input pins connected to the sensor. These parameters must
#   be provided.
#cal_dia1: 1.50
#cal_dia2: 2.00
#   The calibration values (in mm) for the sensors. The default is
#   1.50 for cal_dia1 and 2.00 for cal_dia2.
#raw_dia1: 9500
#raw_dia2: 10500
#   The raw calibration values for the sensors. The default is 9500
#   for raw_dia1 and 10500 for raw_dia2.
#default_nominal_filament_diameter: 1.75
#   The nominal filament diameter. This parameter must be provided.
#max_difference: 0.200
#   Maximum allowed filament diameter difference in millimeters (mm).
#   If difference between nominal filament diameter and sensor output
#   is more than +- max_difference, extrusion multiplier is set back
#   to %100. The default is 0.200.
#measurement_delay: 70
#   The distance from sensor to the melting chamber/hot-end in
#   millimeters (mm). The filament between the sensor and the hot-end
#   will be treated as the default_nominal_filament_diameter. Host
#   module works with FIFO logic. It keeps each sensor value and
#   position in an array and POP them back in correct position. This
#   parameter must be provided.
#enable: False
#   Sensor enabled or disabled after power on. The default is to
#   disable.
#measurement_interval: 10
#   The approximate distance (in mm) between sensor readings. The
#   default is 10mm.
#logging: False
#   Out diameter to terminal and klipper.log can be turn on|of by
#   command.
#min_diameter: 1.0
#   Minimal diameter for trigger virtual filament_switch_sensor.
#use_current_dia_while_delay: False
#   Use the current diameter instead of the nominal diameter while
#   the measurement delay has not run through.
#pause_on_runout:
#runout_gcode:
#insert_gcode:
#event_delay:
#pause_delay:
#   See the "filament_switch_sensor" section for a description of the
#   above parameters.
```

Board specific hardware support

[sx1509]

Configure an SX1509 I2C to GPIO expander. Due to the delay incurred by I2C communication you should NOT use SX1509 pins as stepper enable, step or dir pins or any other pin that requires fast bit-banging. They are best used as static or gcode controlled digital outputs or hardware-pwm pins for e.g. fans. One may define any number of sections with an "sx1509" prefix. Each expander provides a set of 16 pins (sx1509_my_sx1509:PIN_0 to sx1509_my_sx1509:PIN_15) which can be used in the printer configuration.

See the generic-duet2-duex.cfg file for an example.

```
[sx1509 my_sx1509]
i2c_address:
#   I2C address used by this expander. Depending on the hardware
#   jumpers this is one out of the following addresses: 62 63 112
#   113. This parameter must be provided.
#i2c_mcu:
#i2c_bus:
#i2c_speed:
#   See the "common I2C settings" section for a description of the
#   above parameters.
#i2c_bus:
#   If the I2C implementation of your micro-controller supports
#   multiple I2C busses, you may specify the bus name here. The
#   default is to use the default micro-controller i2c bus.
```

[samd_sercom]

SAMD SERCOM configuration to specify which pins to use on a given SERCOM. One may define any number of sections with a "samd_sercom" prefix. Each SERCOM must be configured prior to using it as SPI or I2C peripheral. Place this config section above any other section that makes use of SPI or I2C buses.

```
[samd_sercom my_sercom]
sercom:
#   The name of the sercom bus to configure in the micro-controller.
#   Available names are "sercom0", "sercom1", etc.. This parameter
#   must be provided.
tx_pin:
#   MOSI pin for SPI communication, or SDA (data) pin for I2C
#   communication. The pin must have a valid pinmux configuration
#   for the given SERCOM peripheral. This parameter must be provided.
#rx_pin:
#   MISO pin for SPI communication. This pin is not used for I2C
#   communication (I2C uses tx_pin for both sending and receiving).
#   The pin must have a valid pinmux configuration for the given
#   SERCOM peripheral. This parameter is optional.
clk_pin:
#   CLK pin for SPI communication, or SCL (clock) pin for I2C
#   communication. The pin must have a valid pinmux configuration
#   for the given SERCOM peripheral. This parameter must be provided.
```

[adc_scaled]

Duet2 Maestro analog scaling by vref and vssa readings. Defining an adc_scaled section enables virtual adc pins (such as "my_name:PB0") that are automatically adjusted by the board's vref and vssa monitoring pins. Be sure to define this config section above any config sections that use one these virtual pins.

See the generic-duet2-maestro.cfg file for an example.

```
[adc_scaled my_name]
vref_pin:
#   The ADC pin to use for VREF monitoring. This parameter must be
#   provided.
vssa_pin:
#   The ADC pin to use for VSSA monitoring. This parameter must be
#   provided.
#smooth_time: 2.0
#   A time value (in seconds) over which the vref and vssa
#   measurements will be smoothed to reduce the impact of measurement
#   noise. The default is 2 seconds.
```

[replicape]

Replicape support - see the beaglebone guide and the generic-replicape.cfg file for an example.

```
# The "replicape" config section adds "replicape:stepper_x_enable"
# virtual stepper enable pins (for steppers X, Y, Z, E, and H) and
# "replicape:power_x" PWM output pins (for hotbed, e, h, fan0, fan1,
# fan2, and fan3) that may then be used elsewhere in the config file.
[replicape]
revision:
#   The replicape hardware revision. Currently only revision "B3" is
#   supported. This parameter must be provided.
#enable_pin: !gpio0_20
#   The replicape global enable pin. The default is !gpio0_20 (aka
#   P9_41).
host_mcu:
#   The name of the mcu config section that communicates with the
#   Klipper "linux process" mcu instance. This parameter must be
#   provided.
#standstill_power_down: False
#   This parameter controls the CFG6_ENN line on all stepper
#   motors. True sets the enable lines to "open". The default is
#   False.
#stepper_x_microstep_mode:
#stepper_y_microstep_mode:
#stepper_z_microstep_mode:
#stepper_e_microstep_mode:
#stepper_h_microstep_mode:
#   This parameter controls the CFG1 and CFG2 pins of the given
#   stepper motor driver. Available options are: disable, 1, 2,
#   spread2, 4, 16, spread4, spread16, stealth4, and stealth16. The
#   default is disable.
#stepper_x_current:
#stepper_y_current:
#stepper_z_current:
#stepper_e_current:
#stepper_h_current:
#   The configured maximum current (in Amps) of the stepper motor
#   driver. This parameter must be provided if the stepper is not in a
#   disable mode.
#stepper_x_chopper_off_time_high:
#stepper_y_chopper_off_time_high:
#stepper_z_chopper_off_time_high:
#stepper_e_chopper_off_time_high:
#stepper_h_chopper_off_time_high:
#   This parameter controls the CFG0 pin of the stepper motor driver
#   (True sets CFG0 high, False sets it low). The default is False.
#stepper_x_chopper_hysteresis_high:
#stepper_y_chopper_hysteresis_high:
#stepper_z_chopper_hysteresis_high:
#stepper_e_chopper_hysteresis_high:
#stepper_h_chopper_hysteresis_high:
#   This parameter controls the CFG4 pin of the stepper motor driver
#   (True sets CFG4 high, False sets it low). The default is False.
#stepper_x_chopper_blank_time_high:
#stepper_y_chopper_blank_time_high:
#stepper_z_chopper_blank_time_high:
#stepper_e_chopper_blank_time_high:
#stepper_h_chopper_blank_time_high:
#   This parameter controls the CFG5 pin of the stepper motor driver
#   (True sets CFG5 high, False sets it low). The default is True.
```

Other Custom Modules

[palette2]

Palette 2 multimaterial support - provides a tighter integration supporting Palette 2 devices in connected mode.

This modules also requires `[virtual_sdcard]` and `[pause_resume]` for full functionality.

If you use this module, do not use the Palette 2 plugin for Octoprint as they will conflict, and 1 will fail to initialize properly likely aborting your print.

If you use Octoprint and stream gcode over the serial port instead of printing from virtual_sd, then remo **M1** and **M0** from *Pausing commands* in *Settings > Serial Connection > Firmware & protocol* will prevent the need to start print on the Palette 2 and unpausing in Octoprint for your print to begin.

```
[palette2]
serial:
```

```
#   The serial port to connect to the Palette 2.
#baud: 115200
#   The baud rate to use. The default is 115200.
#feedrate_splice: 0.8
#   The feedrate to use when splicing, default is 0.8
#feedrate_normal: 1.0
#   The feedrate to use after splicing, default is 1.0
#auto_load_speed: 2
#   Extrude feedrate when autoloading, default is 2 (mm/s)
#auto_cancel_variation: 0.1
#   Auto cancel print when ping varation is above this threshold
```

[angle]

Magnetic hall angle sensor support for reading stepper motor angle shaft measurements using a1333, as5047d, or tle5012b SPI chips. The measurements are available via the API Server and motion analysis tool. See the G-Code reference for available commands.

```
[angle my_angle_sensor]
sensor_type:
#   The type of the magnetic hall sensor chip. Available choices are
#   "a1333", "as5047d", and "tle5012b". This parameter must be
#   specified.
#sample_period: 0.000400
#   The query period (in seconds) to use during measurements. The
#   default is 0.000400 (which is 2500 samples per second).
#stepper:
#   The name of the stepper that the angle sensor is attached to (eg,
#   "stepper_x"). Setting this value enables an angle calibration
#   tool. To use this feature, the Python "numpy" package must be
#   installed. The default is to not enable angle calibration for the
#   angle sensor.
cs_pin:
#   The SPI enable pin for the sensor. This parameter must be provided.
#spi_speed:
#spi_bus:
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   See the "common SPI settings" section for a description of the
#   above parameters.
```

Common bus parameters

Common SPI settings

The following parameters are generally available for devices using an SPI bus.

```
#spi_speed:
#   The SPI speed (in hz) to use when communicating with the device.
#   The default depends on the type of device.
#spi_bus:
#   If the micro-controller supports multiple SPI busses then one may
#   specify the micro-controller bus name here. The default depends on
#   the type of micro-controller.
#spi_software_sclk_pin:
#spi_software_mosi_pin:
#spi_software_miso_pin:
#   Specify the above parameters to use "software based SPI". This
#   mode does not require micro-controller hardware support (typically
#   any general purpose pins may be used). The default is to not use
#   "software spi".
```

Common I2C settings

The following parameters are generally available for devices using an I2C bus.

Note that Klipper's current micro-controller support for i2c is generally not tolerant to line noise. Unexpected errors on the i2c wires may result in Klipper raising a run-time error. Klipper's support for error recovery varies between each micro-controller type. It is generally recommended to only use i2c devices that are on the same printed circuit board as the micro-controller.

Most Klipper micro-controller implementations only support an `i2c_speed` of 100000. The Klipper "linux" micro-controller supports a 400000 speed, but it must be set in the operating system and the `i2c_speed` parameter is otherwise ignored. The Klipper "rp2040" micro-controller supports a rate of 400000 via the `i2c_speed` parameter. All other Klipper micro-controllers use a 100000 rate and ignore the `i2c_speed` parameter.

```
#i2c_address:
#   The i2c address of the device. This must specified as a decimal
#   number (not in hex). The default depends on the type of device.
#i2c_mcu:
#   The name of the micro-controller that the chip is connected to.
#   The default is "mcu".
#i2c_bus:
#   If the micro-controller supports multiple I2C busses then one may
#   specify the micro-controller bus name here. The default depends on
#   the type of micro-controller.
#i2c_speed:
#   The I2C speed (in Hz) to use when communicating with the device.
#   The Klipper implementation on most micro-controllers is hard-coded
#   to 100000 and changing this value has no effect. The default is
#   100000.
```