gcode generator

Release V0.01

Martijn Schouten

Dec 01, 2021

CONTENTS:

The gcode generator consists of two classes. The *generator* class is used to make it easier to generate gcode for basic geometric moves. The *calibration_pattern* class is used to generate the actual calibration patterns. To use the module to generate a gcode file called "example.g" with a calibration pattern for tools 1,2,3,4 and 5 and with tool 2 as reference use :

```
from calibration_pattern import calibartion_pattern

pattern = calibartion_pattern()

tool_list = [1,2,3,4,5]
reference_tool = 2
lines = pattern.full_interlocked_print(tool_list,reference_tool,"example.g")
```

By default the code assumes you are using a Diabase H-series 3D printer. To configure it for another printer or change slicer settings of the gen instance in pattern should be changed. For example to make the code suitable for a dual material printer add the following code before running the `calibration_pattern.full_interlocked_print()` function:

```
pattern.gen.tools = [0,1]
pattern.gen.standby_temperatures = [175,175]
pattern.gen.printing_temperatures = [200,200]
pattern.gen.extrusion_multiplier = [1.1,1.1]
pattern.gen.retraction_distance =[5,5]
pattern.gen.x_offsets = [0,0]
pattern.gen.y_offsets = [0,0]
```

CHAPTER

ONE

GENERATOR CLASS

class generator.**generator**

Bases: object

This class can be used to make simple gcode patterns

bed_temp = 60

Temperature in degrees celsius to which the bed will be heated

enable_retraction = True

Wether or not retraction should be enabled

extrude(*amount*)

Extrude a specifc amount of filament

Parameters **amount** – The amount of filament to extrude

Returns The gcode to generate the extrusion

Return type string

extrusion_for_length(*length*)

Convert a desired line length, to the extrusion volume needed

Parameters **volume** – The desired line length

Returns The extrusion volume needed

Return type float

extrusion_multiplier = [1.1, 1.1, 1.1, 1.1, 1.1]

Extrusion multiplier used for the tools in the tools list. Should be of the same size as tools.

extrusion_volume_to_length(*volume*)

Convert a desired volume to be extruded out of the nozzle, to the length of filament that needs to be extruded

Parameters **volume** – The desired volume to be extruded out of the nozzle

Returns The length of filament that needs to be extruded

Return type float

filament_diameter = 1.75

Diameter of the used filament in millimeter

find_tools(*tool_list*)

Find the tool indexes belonging to a tool number. Usefull to generate the tool_index parameter of *generator.tool_change()*

Parameters **tool_list** – A list of tool numbers.

Returns A List of tool indices

Return type list

`insert_pause = True`
Insert a pause between probing the bed and actually printing, during which a piece of paper can be placed on the bed.

`layer_height = 0.2`
Used layer height in millimeter

`line`(*x*, *y*)
Generate the gcode for a line from current position with a specific length in x an y

Parameters

- **x** – Distance to move in the x direction
- **y** – Distance to move in the y direction

Returns The gcode to generate the line

Return type string

`move`(*x*, *y*)
Generate the gcode for a move from current position with a specific length in x an y

Parameters

- **x** – Distance to move in the x direction in millimeter
- **y** – Distance to move in the y direction in millimeter

Returns The gcode to generate the move

Return type string

`move_to`(*x*, *y*)
Generate the gcode for a move from the current position to a specific coordinate

Parameters

- **x** – The x value of the coordinate to move towards in millimeter
- **y** – The y value of the coordinate to move towards in millimeter

Returns The gcode to generate the move

Return type string

`nozzle_diameters = [0.4, 0.4, 0.4, 0.4, 0.4]`
Diameter of the nozzle of the tools in the tools list in millimeter. Should be of the same size as tools.

`print_speed = 30`
Used printing speed in mm/s

`printing_temperatures = [200, 200, 200, 200, 200]`
Printing temperature used for the tools in the tools list in degrees Celsius. Should be of the same size as tools.

`quarter_turn`(*x*, *y*, *clockwise*)
Generate the gcode for a quarter turn from the currrent position to a new position at a specified distance in x and y

Parameters

- **x** – Distance to move in the x direction in millimeter
- **y** – Distance to move in the y direction in millimeter

- **clockwise** – If the turn should be clockwise or counter clockwise

Returns The gcode to generate the quarter_turn

Return type string

reretract()
Generate the gcode for a reverse retraction in case retraction is enabled

Returns The gcode to generate the reverse retraction

Return type string

retract()
Generate the gcode for a retraction in case retraction is enabled

Returns The gcode to generate the retraction

Return type string

retraction_distance = [5, 5, 5, 5, 5]
Retraction distance used for the tools in the tools list. Should be of the same size as tools.

retraction_speed = 80
Retraction speed in mm/s used by all tools

rotate(*x_cor*, *y_cor*)
Rotate a coordinate around the center of the print

Parameters

- **x_cor** – The x value of the coordinate to rotate around the center
- **y_cor** – The y value of the coordinate to rotate around the center

Returns The rotated coordinate

Return type list

rotate_around_origin(*x_cor*, *y_cor*)
Rotate a coordinate around the origin (0,0)

Parameters

- **x_cor** – The x value of the coordinate to rotate around (0,0)
- **y_cor** – The y value of the coordinate to rotate around (0,0)

Returns The rotated coordinate

Return type list

rotation = 0.2617993877991494
Rotation of the structure in radians

square(*x*, *y*, *clockwise*)
Generate the gcode for square with a specific size and the bottom left corner at the current position

Parameters

- **x** – The size of the square in the x direction
- **y** – The size of the square in the y direction
- **clockwise** – Boolean indicating if the square should be printed clockwise or counter clockwise

Returns The gcode to generate the square

Return type string

standby_temperatures = [175, 175, 175, 175, 175]
Standby temperature used for the tools in the tools list in degrees Celsius. Should be of the same size as tools.

starting_code(*tool_list_indexes*)
Generate the gcode to start a print. This includes things as homing, heating up the bed and heating up tools

Parameters **tool_list_indexes** – List of tool indices that should be heated. The indices will be used to select a tool in *generator.tools*

Returns The starting gcode

Return type string

stop_code()
Generate the gcode to stop a print. This includes things as moving up the bed and turning off heaters

Returns The stopping gcode

Return type string

tool_change(*tool_index*)
Generate the gcode to perform a tool change

Parameters **tool_list_indexes** – Tool indices of the tool that should be selected. The indice will be used to select a tool in *generator.tools*

Returns The gcode for the tool change

Return type string

tools = [1, 2, 3, 4, 5]
List containing the tool numbers of the tools of the 3D printer

u_turn(*x*, *y*, *clockwise*)
Generate the gcode for a u turn from the currrent position to a new position at a specified distance in x and y

Parameters

- **x** – Distance to move in the x direction in millimeter
- **y** – Distance to move in the y direction in millimeter
- **clockwise** – Boolean indicating if the turn should be clockwise or counter clockwise

Returns The gcode to generate the u turn

Return type string

x_center = 0
Location where the center of the printed structure will be (in mm)

x_offsets = [0, 0, 0, 0, 0]
List with additional offsets in the x direction (mm) given to all x moves of the tools in the tools list. Should be of the same size as tools.

y_center = 0
Location where the center of the printed structure will be (in mm)

y_offsets = [0, 0, 0, 0, 0]
List with additional offsets in the y direction (mm) given to all y moves of the tools in the tools list. Should be of the same size as tools.

`z_hop = 0.5`
: Wether or not a z-hop should be performed during a retraction

`z_offset = 0.15`
: Additional offset in the z direction given to all z moves in millimeter. Allows compensating for printers improperly calibrated in the z direction

CHAPTER

TWO

CALIBRATION_PATTERN CLASS

class calibration_pattern.**calibration_pattern**

Bases: object

differential_interlocked_reference_pattern(*x_start*, *y_start*, *direction*)

Generate the gcode to print one side (the reference side) of two interlocked patterns, one going up and one going down

Parameters

- **x_start** – The x location of the bottom left corner of the pattern
- **y_start** – The y location of the bottom left corner of the pattern
- **direction** – The direction the pattern should be printed in. Options are: '+y','+x'

Returns The gcode to generate the pattern

Return type string

differential_interlocked_signal_pattern(*x_start*, *y_start*, *direction*)

Generate the gcode to print one side (the signal side) of two interlocked patterns, one going up and one going down

Parameters

- **x_start** – The x location of the bottom left corner of the pattern
- **y_start** – The y location of the bottom left corner of the pattern
- **direction** – The direction the pattern should be printed in. Options are: '+y','+x'

Returns The gcode to generate the pattern

Return type string

effective_length()

Calculates actual length of a structure, taking into account that the

Returns The actual length of a structure

Return type float

effective_length_interlocked()

Calculates actual length of a structure, taking into account that the

Returns The actual length of a structure

Return type float

full_interlocked_print(*tool_list*, *reference_tool*, *save_file_name*)
Generate the gcode to print a complete interlocked calibration pattern, that can be scanned and analysed to find the xy offsets.

Parameters

- **tool** – List of tool number of the tools. Each tool will be used for 4 calibration patterns. One in both the positive and negative x and y directions.
- **save_file_name** – Name of the file the gcode will be written to

Returns The gcode to generate the print

Return type string

gen = <generator.generator object>
An instance of the generator class to generate the gcode

interlocked_period = 4
How many milliemeters it takes before the structure repeats itself

interlocked_pitch = 0.75
The pitch of the lines in the center of the structure in millimeters

interlocked_reference_pattern(*x_start*, *y_start*, *direction*)
Generate the gcode to print a single interlocked reference pattern

Parameters

- **x_start** – The x location of the bottom left corner of the pattern
- **y_start** – The y location of the bottom left corner of the pattern
- **direction** – The direction the pattern should be printed in. Options are: '+y', '-y','+x', '-x'

Returns The gcode to generate the pattern

Return type string

interlocked_signal_pattern(*x_start*, *y_start*, *direction*)

length = 70
Total length of all the meanders

meander_print(*tool*, *save_file_name*)
Generate the gcode to print a simple meandering structure, that fir example can be used to better understand conduction in 3D printed conductors.

Parameters

- **tool** – tool number of the tool that will be used for the print
- **save_file_name** – Name of the file the gcode will be written to

Returns The gcode to generate the print

Return type string

pitch = 1
Milimeter spacing between the lines of the test pattern

repetitions()
Calculates the number of repetions/periods of the repetitie pattern.

Returns The number of repetions/periods of the repetitie pattern.

Return type int

`repetitions_interlocked`()
Calculates the number of repetions/periods of the interlocked repetitive pattern.

Returns The number of repetions/periods of the interlocked repetitive pattern.

Return type int

`sigref_only = 2`
Space where this only a sig or a ref pattern

`single_pattern`(*x_start*, *y_start*, *direction*)
Generate the gcode to print a single simple reference pattern

Parameters

- **x_start** – The x location of the bottom left corner of the pattern
- **y_start** – The y location of the bottom left corner of the pattern
- **direction** – The direction the pattern should be printed in. Options are: '+y', '-y','+x', '-x'

Returns The gcode to generate the pattern

Return type string

`spacing = 3`
Spacing between two patterns of different nozzles

`spacing_to_square = 5`
Spacing between the patterns and the square

`square_lines = 3`
Number of lines of the square around the structure

`square_pattern`(*x_start*, *y_start*, *x*, *y*, *clockwise*, *n*)
Generate the gcode to print a square at a specific location with a specific size and a specified thickness

Parameters

- **x_start** – The x location of the bottom left corner of the square
- **y_start** – The y location of the bottom left corner of the square
- **x** – The size of the square in the x direction
- **y** – The size of the square in the y direction
- **clockwise** – Boolean indicating if the square should be printed clockwise or counter clockwise
- **n** – The thickness of the lines of the square in number of times the nozzle diameter

Returns The gcode to generate the square

Return type string

`total_height`()
Calculates the total height of the patterns in both directions together

Returns the total height of the patterns in both directions together

Return type float

`total_height_interlocked`()
Calculates the total height of the patterns in both directions together in case of an interlocked print

Returns the total height of the patterns in both directions together

Return type float

`total_one_dir_width()`
Calculates the total width of all the patterns in one direction

Returns the total width of all the patterns in one direction

Return type float

`total_width()`
Calculates the total width of the patterns in both directions together

Returns the total width of the patterns in both directions together

Return type float

`total_width_interlocked()`
Calculates the total width of the patterns in both directions together in case of an interlocked print

Returns the total width of the patterns in both directions together

Return type float

`width = 8`
Length of lines in the test pattern

CHAPTER

THREE

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

c

g