Полезные настройки для принтера PLUS 4

Опробованные и протестированные мной изменения для более точной печати, вывода графиков input shaping, проверка натяжения ремней, калибровка стола и т.д.

ВНИМАНИЕ!

Данная статья не является призывом к действию и не несет обязательный характер!

Все изменения Вы делаете под собственную ответственность!

Перед внесением изменений создайте архив конфигурационных файлов printer.cfg и gcode_macro.cfg

Предлагаю для удобства внесения изменений создать отдельный файл, чтобы вносить в него изменения используя **include**.

Подготовка:

Итак, создаем файл в корне, где лежат все наши файлы конфигурации и назовем его к примеру **mod.cfg**

image.png

После того как мы создали новый файл mod.cfg, нам необходимо его подключить в файле **printer.cfg**, открываем его и добавляем в самом конце перед SAVE_CONFIG

> ### Подключения файла с макросами
>
> [include mod.cfg]
>
> #*# <---------------------- SAVE_CONFIG ---------------------->

Подготовка на этом заканчивается и давайте приступим к добавлению наших новых полезных изменений.

1. Охлаждение материнской платы

В QIDI Plus 4 по умолчанию установлено значение controller_fan для активации вентилятора охлаждения материнской платы, это означает, что вентилятор не запустится, пока мы не начнем печать.

Чтобы улучшить это, мы собираемся внести некоторые изменения в настройки по умолчанию.

Сначала мы отключим настройки вентилятора охлаждения материнской платы, которые мы можем найти в разделе [controller_fan board_fan] в файле **printer.cfg**, данный блок мы закомментируем, добавив символ # в начале каждой строки:

> ### Закомментируем этот блок в файле printer.cfg
>
> # [controller_fan board_fan]
>
> # pin:U_1:PC4
>
> # max_power:1.0
>
> # shutdown_speed:1.0
>
> # cycle_time:0.01
>
> # fan_speed: 1.0
>
> # heater:chamber
>
> # stepper:stepper_x,stepper_y

Далее мы вставляем в созданный нами файл **mod.cfg** первый наш блок:

> ### Настройка и отображение температуры MB
>
> [temperature_fan board_fan]
>
> pin:U_1:PC4
>
> max_power: 1.0

```
shutdown_speed: 1.0

cycle_time: 0.01

off_below: 0

sensor_type: temperature_host

control: pid

pid_deriv_time: 2.0

pid_Kp: 5

pid_Ki: 2

pid_Kd: 5

target_temp: 45

min_speed: 0.3

max_speed: 1.0

min_temp: 0

max_temp: 100
```

После этого у нас в интерфейсе появится удобное управление температурой материнской платы и ее отображение. Конечно желательно установить кулер большего размера, так как наш маленький 40мм не справляется. Но и так уже гораздо лучше.

image.png

2. Улучшение показаний температуры камеры

В QIDI Plus 4 датчик температуры расположен не в самом лучшем месте, которое, вероятно, не является наиболее оптимальным. Найти его можно в задней правой части, под рамой шасси:

image.png

Наиболее оптимальным всегда является то, чтобы температура была в точке, ближайшей к зоне печати, поэтому размещение датчика в углу, рядом с металлом корпуса, на возвышении и на стороне, противоположной той, куда дует сам нагреватель, не поможет

обеспечить надежные показатели температуры.

Чтобы улучшить эти показания и лучше управлять нагревателем, мы воспользуемся датчиком температуры, установленным в голове принтера (GD32), чтобы комбинировать показания температуры и таким образом, повысить точность.

Благодаря гибкости и мощности Klipper мы собираемся создать "виртуальный" датчик температуры, в котором мы будем комбинировать показания обоих датчиков:

Сначала мы прокомментируем блок [heater_generic chamber] в файле **printer.cfg**:

```
# [heater_generic chamber]
  # heater_pin:U_1:PC8
  # max_power:0.4
  # sensor_type:NTC 100K MGB18-104F39050L32
  # sensor_pin:U_1:PA1
  # control = pid
  # pid_Kp=63.418
  # pid_Ki=1.342
  # pid_Kd=749.125
  # min_temp:-100
  # max_temp:70
  # target_min_temp:45
  # target_max_temp:65
  # heat_with_heater_bed:True
  # heat_with_heater_bed_tem_add:25
```

Теперь мы снова определим датчик температуры камеры под другим названием `[temperature_sensor chamber_probe]` и изменим `[heater_generic chamber]` так, чтобы он использовал виртуальный датчик, используя комбинированные показания обоих датчиков (датчика головы (GD32) и камеры). По моим тестам лучший результат достигается при

соотношении 3 к 1, где влияние датчика GD32 составляет 75%, а влияние датчика камеры - 25%.

Добавим наш новый блок в файл **mod.cfg**:

> ### Показания температуры сенсоров головы и датчика нагрева камеры
>
> [temperature_sensor chamber_probe]
>
> sensor_type:NTC 100K MGB18-104F39050L32
>
> sensor_pin:U_1:PA1
>
> [heater_generic chamber]
>
> heater_pin:U_1:PC8
>
> max_power:0.5 # Original 0.7 or 0.4 depending on firmware version
>
> control = pid
>
> pid_Kp=63.418
>
> pid_Ki=1.342
>
> pid_Kd=749.125
>
> min_temp:-100
>
> max_temp:80
>
> sensor_type: temperature_combined
>
> sensor_list: temperature_sensor GD32, temperature_sensor GD32, temperature_sensor GD32, temperature_sensor chamber_probe
>
> combination_method: mean
>
> maximum_deviation: 70
>
> target_min_temp:0
>
> target_max_temp:65

```
heat_with_heater_bed:True

heat_with_heater_bed_tem_add:25
```

Почему это действительно работает?

- Сенсор на голове (GD32) хорошо отражает **температуру воздуха рядом с моделью**
- А сенсор в корпусе (chamber_probe) — **общий фон** камеры

Сочетая их, ты получаешь **среднюю, но смещённую в пользу реального места печати температуру**, и это оптимально для контроля нагрева камеры.

Метод	Надёжность	Точность около модели	Простота	Рекомендуется
Один сенсор (в корпусе)	⭐⭐	❌ **низкая**	✅	**Нет**
Один сенсор (на голове)	⭐⭐	✅ **хорошая**	✅	**Лучше**
Комбинированный подход	✅	⭐⭐ **отличная**	✅	✅ **Лучший**

Если хочешь максимально точное управление температурой камеры, особенно при печати PETG, ABS, ASA, PC и т.д., — комбинированный метод определённо лучше.

Пример других весов для вычисления показаний в камере:

- 50% / 50% → просто GD32, chamber_probe
- 80% / 20% → GD32, GD32, GD32, GD32, chamber_probe
- 90% / 10% → GD32 ×9, chamber_probe ×1

Просто подбираешь нужную комбинацию вручную. У меня стоит 75% на 25%. И теперь при печати я более точно могу контролировать показания температуры в камере.

3. Предотвращение перегрева при печати моделей в полную высоту 280мм

Принимая во внимание расположение системы нагревателя камеры, когда у нас есть модель, которая достигает или превышает 268 мм, сама конструкция стола перекрывает выход горячего воздуха, что иногда приводит к тому, что термозащита Klipper может остановить текущую печать и вылететь с ошибкой. Поскольку это «проблема» дизайна, которую трудно решить, не расширив выходное отверстие в корпусе нагревателя либо

заменить сам корпус нагревателя на кастомную модель например такую Qidi Plus 4 chamber heater, мы можем использовать расширенные функции, доступные в макросах Klipper, чтобы минимизировать эту проблему. Для этого мы воспользуемся макросом SET_PRINT_STATS_INFO, который мы добавим в **mod.cfg**, чтобы снизить температуру нагревателя, чтобы печать завершилась без ошибок.

Добавляем следующий блок в наш файл **mod.cfg**:

```
[gcode_macro SET_PRINT_STATS_INFO]
rename_existing: SET_PRINT_STATS_INFO_BASE
gcode:
    {% set curlayer = params.CURRENT_LAYER|default(1)|int %}
    {% set total_layer = params.TOTAL_LAYER|default(none) %}
    {% set layer_height = params.LAYER_HEIGHT|default(none) %}
    {% if printer.toolhead.position.z >= 268 %}
      M141 S15
    {% endif %}
    {% set cmd = "SET_PRINT_STATS_INFO_BASE CURRENT_LAYER=%d" %
curlayer %}
    {% if total_layer is not none %}
      {% set cmd = cmd + " TOTAL_LAYER=%s" % total_layer %}
    {% endif %}
    {% if layer_height is not none %}
      {% set cmd = cmd + " LAYER_HEIGHT=%s" % layer_height %}
    {% endif %}

    { cmd }
```

Для меня такой вариант сейчас самый простой.

4. Изменение линии очистки при начале печати на адаптивную (KAMP)

Мне не очень нравится как очищается сопло при начале печати, и иногда оно очень сильно прилипает. И если мы следим за началом печати можно вовремя избежать повреждение пластины. Поэтому я решил воспользоваться уже готовым решением, макросом LINE_PURGE.

Для начала необходимо раскомментировать строку в файле **KAMP_Settings.cfg**, который найдем в том же месте, что и наш **printer.cfg**, в нем мы убираем символ # перед строкой:

> [include ./KAMP/Line_Purge.cfg] # Раскомментировал строку

Далее необходимо поправить стартовый g-code в слайсере:

image.png

Удалил из стартового g-code принтера следующий блок, который расположен между строками `G1 E3 F1800` и `G1 Z1 F600`:

> G1 X{(min(print_bed_max[0] - 12, first_layer_print_min[0] + 80))} E{85 * 0.5 * initial_layer_print_height * nozzle_diameter[0]} F3000
>
> G1 Y{max((min(print_bed_max[1] - 3, first_layer_print_min[1] + 80) - 85), 0) + 2} E{2 * 0.5 * initial_layer_print_height * nozzle_diameter[0]} F3000
>
> G1 X{max((min(print_bed_max[0] - 12, first_layer_print_min[0] + 80) - 85), 0)} E{85 * 0.5 * initial_layer_print_height * nozzle_diameter[0]} F3000
>
> G1 Y{max((min(print_bed_max[1] - 3, first_layer_print_min[1] + 80) - 85), 0) + 85} E{83 * 0.5 * initial_layer_print_height * nozzle_diameter[0]} F3000
>
> G1 X{max((min(print_bed_max[0] - 12, first_layer_print_min[0] + 80) - 85), 0) + 2} E{2 * 0.5 * initial_layer_print_height * nozzle_diameter[0]} F3000
>
> G1 Y{max((min(print_bed_max[1] - 3, first_layer_print_min[1] + 80) - 85), 0) + 3} E{82 * 0.5 * initial_layer_print_height * nozzle_diameter[0]} F3000

> G1 X{max((min(print_bed_max[0] - 12, first_layer_print_min[0] + 80) - 85), 0) + 3} Z0
>
> G1 X{max((min(print_bed_max[0] - 12, first_layer_print_min[0] + 80) - 85), 0) + 6}

И вставил команду `LINE_PURGE`.

Итоговый g-code теперь выглядит вот так:

> PRINT_START BED=[bed_temperature_initial_layer_single] HOTEND=[nozzle_temperature_initial_layer] CHAMBER=[chamber_temperatures]
>
> SET_PRINT_STATS_INFO TOTAL_LAYER=[total_layer_count]
>
> M83
>
> M140 S[bed_temperature_initial_layer_single]
>
> M104 S[nozzle_temperature_initial_layer]
>
> M141 S[chamber_temperatures]
>
> G4 P3000
>
> T[initial_tool]
>
> G0 X{max((min(print_bed_max[0] - 12, first_layer_print_min[0] + 80) - 85), 0)} Y{max((min(print_bed_max[1] - 3, first_layer_print_min[1] + 80) - 85), 0)} Z5 F6000
>
> G0 Z[initial_layer_print_height] F600
>
> G1 E3 F1800
>
> LINE_PURGE
>
> G1 Z1 F600
>
> SET_PRINT_STATS_INFO CURRENT_LAYER=1

После этих изменений линия очистки сопла выглядит вот так:

image.png

5. Улучшения конфигурации шаговых двигателей

Этот пункт не обязателен но по моим тестам это дает более лучшие результаты на высоких скоростях и более точно позиционирование, чем есть сейчас в стандартной конфигурации.

Что ты получаем в итоге?

- **Четче движение** по X/Y, меньше люфтов и артефактов.
- **Стабильное позиционирование** на Z, особенно при небольших слоях.
- **Улучшенное качество печати** при ускорении и высоких скоростях.

ВАЖНО!

Если мы увеличим значения run_current, рекомендую обязательно улучшить охлаждение электроники выполнить пункт 1 и желательно установить радиаторы на шаговые двигатели X/Y, но и на ось Z тоже можно установить — лучше будут охлаждаться.

Добавляем в наш файл **mod.cfg** следующий код:

```
### Настройка шагов двигателей

[tmc2209 extruder]

interpolate: False ### Original True


[stepper_x]

microsteps:64 ### Original 32


[stepper_y]
```

```
microsteps: 64 ### Original 32


[stepper_z]

microsteps: 16 ### Original 128


[stepper_z1]

microsteps: 16 ### Original 128


[tmc2240 stepper_y]

run_current: 1.15 ### Original 1.07

interpolate: False ### Original True


[tmc2240 stepper_x]

run_current: 1.15 ### Original 1.07

interpolate: False ### Original True


[tmc2209 stepper_z]

interpolate: False ### Original True


[tmc2209 stepper_z1]

interpolate: False ### Original True
```

Что такое **interpolate: False**

- Отключает **интерполяцию до 256 микрошагов**, которую драйверы TMC делают автоматически, даже если ты задаешь меньшее число микрошагов.

Зачем отключать?

- Интерполяция может сглаживать движение, но **иногда она мешает точности**, особенно при высоком разрешении модели.
- Отключение дает **более предсказуемое** поведение моторов — полезно для точной калибровки и измерений.

Риски:

- Возможен **более шумный** звук работы двигателей.
- Не всегда заметна разница, но для прецизионной настройки — важно.

В сухом остатке до изменений у меня была погрешность около 0.1 а после по всем сторонам на тестовом кубике 20мм х 20мм я добился по нулям:

image.png

6. Улучшения автоматического выравнивания стола и снятия карты

QIDI Plus 4 оснащен несколькими системами выравнивания, которые значительно упрощают, упрощают и автоматизируют процесс выравнивания стола. Мы постараемся улучшить некоторые из них.

`Если вы не выполнили пункт 5, то предложенные дальше изменения будут менее заметны. Но можно добавить в файл **mod.cfg** изменения только для шаговых двигателей оси Z, так как они у нас главные в данном пункте.`

`Опять повторюсь, если вы **применили** пункт 5 то ничего вносить не надо, у вас все будет работать.`

`Тем же кто пропустил пункт 5 необходимо добавить следующий код в файл **mod.cfg**:`

> [stepper_z]
>
> microsteps: 16 ### Original 128
>
> [stepper_z1]
>
> microsteps: 16 ### Original 128

> [tmc2209 stepper_z]
>
> interpolate: False ### Original True
>
> [tmc2209 stepper_z1]
>
> interpolate: False ### Original True

Итак, приступим, первое, что мы сделаем, это изменим **smart_effector** - Точность срабатывания датчика:

> # Настройки Smart Effector
>
> [smart_effector]
>
> speed:2.5
>
> samples: 1
>
> sample_retract_dist: 10
>
> samples_tolerance: 0.013

Эти параметры особенно полезны при использовании multi-probing и дают более точную компенсацию Z-offset.

Далее мы изменяем **bed_mesh** - улучшаем сетку выравнивания стола:

> # Настройки конфигурации сетки стола
>
> [bed_mesh]
>
> horizontal_move_z:10
>
> probe_count:10,10
>
> bicubic_tension:0.3

Эта сетка дает очень точную компенсацию искривлений стола — особенно полезно для больших моделей или тонких первых слоев.

Изменяем **z_tilt** - Автовыравнивание по двум винтам Z:

> # Улучшения Z-Tilt
>
> [z_tilt]
>
> horizontal_move_z: 10
>
> retries: 5
>
> retry_tolerance: 0.013

Z-tilt нужен, чтобы компенсировать несинхронность левых и правых винтов Z. Особенно важно при печати больших моделей.

7. Улучшаем выравнивание стола в ручном режиме с помощью датчиков

Несмотря на то, что QIDI Plus 4 оснащен помощником по выравниванию стола вручную, этот процесс выполняется полностью вручную с использованием листа бумаги или фирменного листа от QIDI. Но у нас современный принтер, давайте воспользуемся тем, что у нас есть датчик выравнивания и автоматизируем, забыв уже про листок.

Добавляем следующий код в файл **mod.cfg**:

> # Ручное выравнивание стола с помощью датчика.
>
> [screws_tilt_adjust]
>
> screw1:250,250
>
> screw1_name: Central
>
> screw2:0,20
>
> screw2_name: Front left

```
screw3: 260,20
screw3_name: Front right
screw4: 260,280
screw4_name: Back right
screw5: 0,280
screw5_name: Back left
screw_thread: CW-M4


# Добавляем макрос
[gcode_macro SCREWS_TILT_CALCULATE]
rename_existing: _SCREWS_TILT_CALCULATE_BASE
gcode:
  { action_respond_info("starting screw rotation calculation...") }
  M141 S0
  M4031
  G28
  _SCREWS_TILT_CALCULATE_BASE
```

После изменения нам доступен макрос:

`Чтобы отрегулировать стол с помощью барашков, сначала слегка ослабьте контргайки. Затем вращайте пальцами только барашки, которые расположены выше контргаек.`

`Отправьте все оси в HOME перед использованием макроса`

image.png

После того как макрос отработал мы получаем следующее окно, где нам показано на сколько необходимо закрутить или открутить регулировочный барашек стола:

image.png

И можно получить очень хороший результат по итогу:

image.png

8. Вывод информации Input Shaper

Для получения графиков Input Shaper будем использовать Klippain и его модуль shaketune

Подключаемся к принтеру по SSH.

В Linux / macOS / Windows открываем терминал (cmd), или используем любой другой доступный вам клиент SSH.
Для подключения вводим команду:

> ssh mks@192.168.2.70

image.png

192.168.2.70 **заменить на IP адрес вашего принтера.**
Вводим пароль: **makerbase** - помним, что вводимый пароль не отображается! Не забудь нажать Enter после ввода.

Клонируем и устанавливаем репозиторий ShakeTune, адаптированный для работы на QIDI вводим следующую команду в терминале подключенном к нашему принтеру

> cd /home/mks && wget -O - https://raw.githubusercontent.com/stew675/klippain-shaketune-for-qidi-plus4/main/install.sh | bash

image.png

После установки идем в наш интерфейс принтера и добавляем следующий код в файл **printer.cfg**:

> [shaketune]
>
> timeout: 1200

```
#    The maximum time in seconds to let Shake&Tune process the CSV files and
generate the graphs.

# result_folder: ~/printer_data/config/ShakeTune_results

#    The folder where the results will be stored. It will be created if it doesn't
exist.

# number_of_results_to_keep: 3

#    The number of results to keep in the result_folder. The oldest results will

#    be automatically deleted after each runs.

# keep_raw_csv: False

#    If True, the raw CSV files will be kept in the result_folder alongside the

#    PNG graphs. If False, they will be deleted and only the graphs will be kept.

# show_macros_in_webui: True

#    Mainsail and Fluidd doesn't create buttons for "system" macros that are not
in the

#    printer.cfg file. If you want to see the macros in the webui, set this to True.
```

После сохранения и перезапуска Klipper нам будут доступны новые макросы:

image.png

Первым запускаем **COMPARE_BELTS_RESPONSES**, но сначала необходимо отрегулировать натяжение ремней как описано в QIDI Wiki. Мое дополнение к натяжению ремней, желательно процедуру перемещения головы принтера сделать 2 раза на ослабленных винтах соответственно, уменьшаем вероятность западания натяжителя. По завершении теста мы получим график, который доступен из веб-интерфейс внутри папки /ShakeTune_results/belts:

image.png

Смотрим готовый результат и либо продолжаем бороться с натяжением ремней, либо у нас все хорошо как на моем графике ниже:

image.png

Далее запускаем второй макрос AXES_SHAPER_CALIBRATION по завершении мы найдем результат в папке / ShakeTune_results / input_shaper и получим изображение, сгенерированное по осям X/Y:

image.png

Получаем два графика по осям X/Y:

image.png

image.png

Настройка Input Shaper для QIDI Plus 4

Мы не будем углубляться в подробности оптимизации и настройки **Input Shaper**, так как подробное руководство по анализу результатов вы найдёте в руководстве по Input Shaper.

Для всех вопросов, связанных с макросами **Klippain ShakeTune**, также доступна дополнительная информация по этой ссылке.

Несколько ключевых замечаний

- **Тип кинематики:**
  QIDI Plus 4 — это **CoreXY**-принтер, и для него особенно важно, чтобы **натяжение ремней было правильным**.
- **Алгоритмы Input Shaper:**
  Из-за особенностей кинематики CoreXY, рекомендуется использовать **одинаковый тип Input Shaper для осей X и Y**, так как они работают совместно.
  Наиболее подходящие алгоритмы для QIDI Plus 4:
  - `ZV`
  - `MZV`
- **Важность подготовки:**
  Перед запуском измерений для настройки Input Shaper обязательно убедитесь, что:
  - Ремни правильно натянуты
  - Направляющие и винты смазаны
  - Все механические соединения надёжно закреплены

Это был последний пункт, спасибо за внимание!

Источники и полезные ссылки

1. Klippain и модуль ShakeTune - инструмент для калибровки Input Shaper и анализа натяжения ремней
2. QIDI Plus 4 chamber heater shell replacement - модель кастомного корпуса нагревателя камеры
3. QIDI Wiki - Регулировка натяжения ремней - официальная документация по настройке ремней
4. Руководство по Input Shaper - подробное руководство по настройке и анализу результатов Input Shaper
5. Документация Klippain ShakeTune для QIDI Plus 4 - дополнительная информация по использованию макросов ShakeTune
6. QIDI Plus 4 Technical Guide - техническое руководство по улучшениям и настройкам принтера

Статью подготовил: @bender_say

Revision #3
Created 2026-10-08 10:28:36 UTC by Volvik88
Updated 2026-10-09 14:18:24 UTC by Volvik88