# Мгновенное прерывание печати для любого принтера на Klipper

Это для всех принтеров на klipper, а не только для Qidi, так что сами смотрите куда определить.

Данная функция меняет поведение при вызове события _handle_cancel_request на мгновенное прерывание любого действия (некоторые функциональные вызовы все равно будут ожидать своего конца, например, как probe) и отменяет очередь, после чего вызывает макрос CANCEL_PRINT.

Внимание!!! Эта статья не для новичков. Делайте все изменения на свой страх и риск. Я не несу ответственности за сделанные вами изменения и последствия от них. Для некоторых моделей принтеров могут понадобиться свои дополнительные изменения, о которых я не могу знать.

Внимание!!! Мгновенное прерывание работает даже при выполнении хоуминга головы и стола. Поэтому ваш макрос CANCEL_PRINT должен учитывать это и быть безопасным для вызова в любой момент. А именно при вызове CANCEL_PRINT в первую очередь он должен не смотря не на что сначала опустить стол (убрать сопло подальше от стола) на какую-либо дистанцию (хотя бы 5мм, так как в момент прерывания ось может быть еще не определена, либо определена неправильно), потом захоумить голову (опять же оси в момент прерывания могут быть не определены или определены неправильно) и только после этого идут уже любые другие движения. В противном случае вы рискуете получить повреждение принтера.

Также я рекомендую и в самом G28 при вызове X Y сделать проверку на неопределённую ось Z и опускание стола в случае этого.

Сразу скажу, что на моем одном принтере это работает только при вызове через fluidd, а на экране нет. Это происходит из-за того, что fluidd оправляет _handle_cancel_request, а экран просто вызов CANCEL_PRINT.

Для установки, нам нужен принтер с доступом в klipper по ssh.

Будут меняться 3 файла:

klipper/klippy/extras/virtual_sdcard.py

klipper/klippy/extras/pause_resume.py

klipper/klippy/gcode.py

Поэтому перед их изменением сделайте резервную копию этих файлов!

Если у вас в стоке сами методы отличаются от тех что я привел как сток, то смотрите на разницу и добавляйте ее сами в свои методы.

Открывает для редактирования файл klipper/klippy/extras/pause_resume.py

Ищем в нем метод _handle_cancel_request.

В стоке он выглядит так:

```
    def _handle_cancel_request(self, web_request):

        self.gcode.run_script("CANCEL_PRINT")
```

Меняем его на вот такой:

```
    def _handle_cancel_request(self, web_request):

        if self.is_sd_active() or self.sd_paused:

            self.v_sd.do_cancel()
```

```
        self.gcode.run_script("CANCEL_PRINT")
```

Сохраняем и закрываем.

Открываем для редактирования файл klipper/klippy/extras/virtual_sdcard.py

Ищем метод do_cancel.

В стоке он выглядит вот так:

```
    def do_cancel(self):

        if self.current_file is not None:

            self.do_pause()

            self.current_file.close()

            self.current_file = None

            self.print_stats.note_cancel()

        self.file_position = self.file_size = 0
```

Меняем его на вот такой:

```
def do_cancel(self):

        try:

            heaters = self.printer.lookup_object('heaters')

            if hasattr(heaters, 'abort_waits'):

                heaters.abort_waits()
```

```
        except Exception:
            logging.exception("virtual_sdcard do_cancel abort_waits")
        try:
            gcode = self.printer.lookup_object('gcode')
            if hasattr(gcode, 'abort_waits'):
                gcode.abort_waits()
        except Exception:
            logging.exception("virtual_sdcard do_cancel gcode abort_waits")
        if self.current_file is not None:
            self.do_pause()
            self.current_file.close()
            self.current_file = None
            self.lines = 0
            self.print_stats.note_cancel()
        self.file_position = self.file_size = 0
        self.print_stats.reset()
        self.printer.send_event("virtual_sdcard:reset_file")
```

Сохраняем и закрываем.

Открываем для редактирования файл klipper/klippy/gcode.py

Ищем метод __init__ в классе GCodeDispatch.

В стоке он выглядит так:

```
    def __init__(self, printer):

        self.printer = printer

        self.is_fileinput = not not printer.get_start_args().get("debuginput")

        printer.register_event_handler("klippy:ready", self._handle_ready)

        printer.register_event_handler("klippy:shutdown", self._handle_shutdown)

        printer.register_event_handler("klippy:disconnect",

                                       self._handle_disconnect)

        # Command handling

        self.is_printer_ready = False

        self.mutex = printer.get_reactor().mutex()

        self.output_callbacks = []

        self.base_gcode_handlers = self.gcode_handlers = {}

        self.ready_gcode_handlers = {}

        self.mux_commands = {}

        self.gcode_help = {}

        self.status_commands = {}

        # Register commands needed before config file is loaded

        handlers = ['M110', 'M112', 'M115',
```

```
                'RESTART', 'FIRMWARE_RESTART', 'ECHO', 'STATUS', 'HELP']
        for cmd in handlers:
            func = getattr(self, 'cmd_' + cmd)
            desc = getattr(self, 'cmd_' + cmd + '_help', None)
            self.register_command(cmd, func, True, desc)
```

Меняем его на вот такой:

```
    def __init__(self, printer):
        self.printer = printer
        self.is_fileinput = not not printer.get_start_args().get("debuginput")
        printer.register_event_handler("klippy:ready", self._handle_ready)
        printer.register_event_handler("klippy:shutdown", self._handle_shutdown)
        printer.register_event_handler("klippy:disconnect",
                                       self._handle_disconnect)
        # Command handling
        self.is_printer_ready = False
        self.abort_epoch = 0
        self.mutex = printer.get_reactor().mutex()
        self.output_callbacks = []
        self.base_gcode_handlers = self.gcode_handlers = {}
```

```
        self.ready_gcode_handlers = {}
        self.mux_commands = {}
        self.gcode_help = {}
        self.status_commands = {}
        # Register commands needed before config file is loaded
        handlers = ['M110', 'M112', 'M115',
                    'RESTART', 'FIRMWARE_RESTART', 'ECHO', 'STATUS', 'HELP']
        for cmd in handlers:
            func = getattr(self, 'cmd_' + cmd)
            desc = getattr(self, 'cmd_' + cmd + '_help', None)
            self.register_command(cmd, func, True, desc)
        # Gcode macro interupt
        self.break_flag=False
```

Ищем метод _process_commands в классе GCodeDispatch.

В стоке он выглядит так:

```
    def _process_commands(self, commands, need_ack=True):
        for line in commands:
            # Ignore comments and leading/trailing spaces
            line = origline = line.strip()
            cpos = line.find(';')
```

```
if cpos >= 0:
    line = line[:cpos]
# Break line into parts and determine command
parts = self.args_r.split(line.upper())
if ''.join(parts[:2]) == 'N':
    # Skip line number at start of command
    cmd = ''.join(parts[3:5]).strip()
else:
    cmd = ''.join(parts[:3]).strip()
# Build gcode "params" dictionary
params = { parts[i]: parts[i+1].strip()
           for i in range(1, len(parts), 2) }
gcmd = GCodeCommand(self, cmd, origline, params, need_ack)
# Invoke handler for command
handler = self.gcode_handlers.get(cmd, self.cmd_default)
try:
    handler(gcmd)
except self.error as e:
    self._respond_error(str(e))
    self.printer.send_event("gcode:command_error")
```

```
                if not need_ack:
                    raise
            except:
                msg = 'Internal error on command:"%s"' % (cmd,)
                logging.exception(msg)
                self.printer.invoke_shutdown(msg)
                self._respond_error(msg)
                if not need_ack:
                    raise
            gcmd.ack()
```

Меняем его на вот такой и после него сразу добавляем метод abort_waits:

```
def _process_commands(self, commands, need_ack=True):
        local_epoch = self.abort_epoch
        for line in commands:
            if self.abort_epoch != local_epoch:
                break
            # Ignore comments and leading/trailing spaces
            line = origline = line.strip()
            cpos = line.find(';')
```

```
if cpos >= 0:
    line = line[:cpos]
# Break line into parts and determine command
parts = self.args_r.split(line.upper())
if ''.join(parts[:2]) == 'N':
    # Skip line number at start of command
    cmd = ''.join(parts[3:5]).strip()
else:
    cmd = ''.join(parts[:3]).strip()
# Build gcode "params" dictionary
params = { parts[i]: parts[i+1].strip()
           for i in range(1, len(parts), 2) }
gcmd = GCodeCommand(self, cmd, origline, params, need_ack)
# Invoke handler for command
handler = self.gcode_handlers.get(cmd, self.cmd_default)
try:
    handler(gcmd)
except self.error as e:
    self._respond_error(str(e))
    self.printer.send_event("gcode:command_error")
```

```
                if not need_ack:

                    raise

            except:

                msg = 'Internal error on command:"%s"' % (cmd,)

                logging.exception(msg)

                self.printer.invoke_shutdown(msg)

                self._respond_error(msg)

                if not need_ack:

                    raise

            gcmd.ack()

    def abort_waits(self):

        self.abort_epoch += 1
```

Сохраняем и закрываем.

После чего перезапускаем принтер по питанию, либо в ssh вызываем команду:

```
sudo -S systemctl restart klipper
```

Во fluidd нажатие кнопки «ПЕРЕЗАПУСТИТЬ СЕРВИС KLIPPER» может быть недостаточно для применения изменений.

Пример безопасного CANCEL_PRINT:

```
[gcode_macro CANCEL_PRINT]

rename_existing: BASE_CANCEL_PRINT

gcode:

    M104 S0

    M140 S0



    M106 P2 S0

    M106 P0 S0

    M106 P3 S0



    M400

   # Тут главное, чтобы опускание стола и G28 X Y выполнились до других движений

    G91

    G0 Z5 F100  # опускаем стол на 5мм

    G90



    G28 X Y # хоумим голову
```

```
    # дальше идут остальные движения, действия и вызов BASE_CANCEL_PRINT
```

Автор: @cerega66 (другие мои статьи)

---

Revision #3
Created 2026-10-20 11:29:31 UTC by Volvik88
Updated 2026-10-20 11:37:26 UTC by Volvik88