

3D PRINTING AND DESIGN REFERENCE DOCUMENT

Document Title:	Document Title
Document No.:	1755550497
Author(s):	jattie
Contributor(s):	

REVISION HISTORY

Revision	Details of Modification(s)	Reason for modification	Date	By
0	Draft release	Document description here	2025/08/18 20:54	jattie

Raspberry Pi Pico MMU

The aim of this project is to build on the work of others that already exist, to learn some and dabble in new things and combine that with 3D printing and 3D design, electronics and controls.

So the idea is to build a MMU for my creality K1 Max and document the journey here. I considered simply upgrading to a K2 or to go the K1 upgrade route, but it's costing a lot, I am not convinced the costs justifies the use of my printers and in doing this project, I have a fun project to work on, tinker with Klipper that I have very little experience of apart from rooting my K1 Max. Also having built the sandtable project using a PI Pico 2040, I thought it might be fun to use the stock I have and repeat the process of fitting the technical lego together.

After some [research](#) and many articles later, I decided to try out the [LH-Stinger Pico MMU](#). It is well documented and so many options that it becomes quite overwhelming, but I believe that it can be integrated with a rooted K1 Max is reasonable fashion, with the caveat that I want to explore using the Raspberry Pi Pico 2040 as a controller instead.

The theory is that you can flash a RPI Pico 2040 with Klipper and once that is achieved you simply define the IO pins for stepper and servo control in Klipper scrips.

The Basic Steps

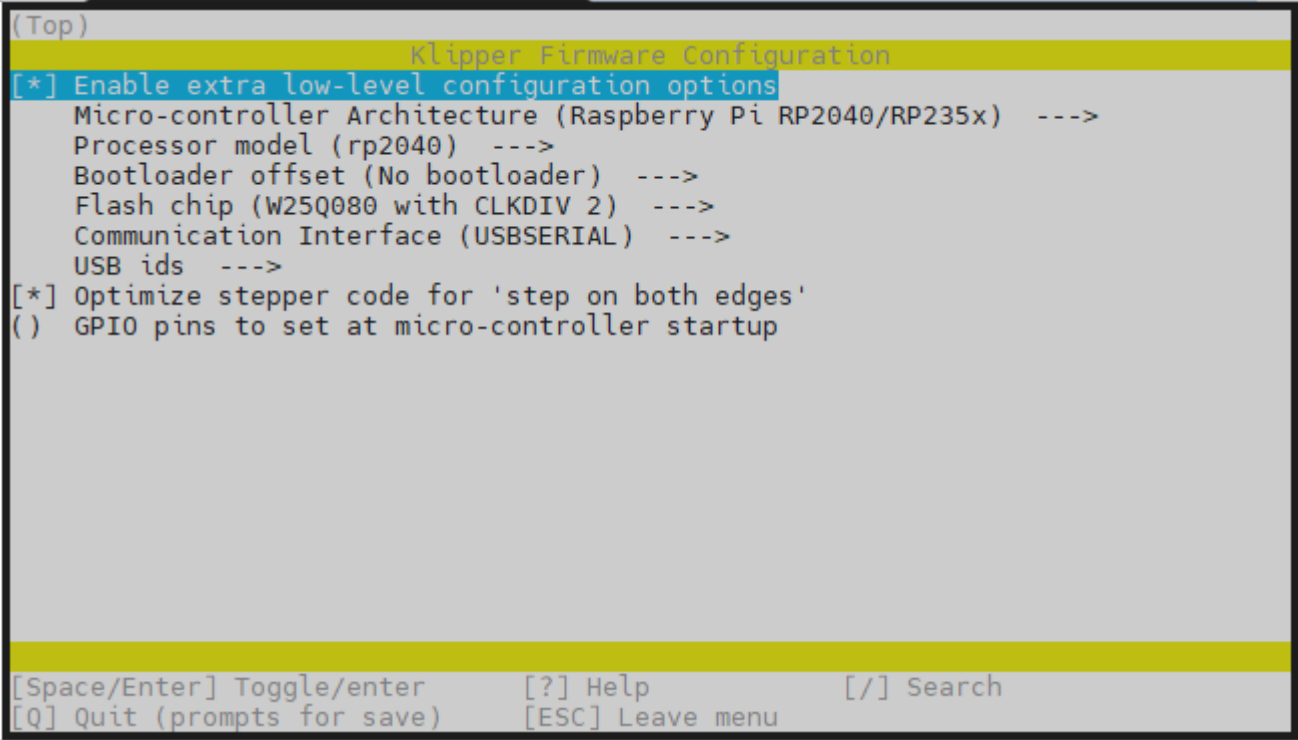
- Install Klipper on a PC
- Create a Configuration
- Build the configuration
- Flash the binary to the Raspberry PI Pico 2040
- Connect the board to the main host application

Install Klipper on a PC

```
git clone https://github.com/Klipper3d/klipper
cd klipper
make menuconfig
```

Create a Configuration

Make Menuconfig brings up this application to easily chose and select options from.



For my test I chose the options as shown above.

The documentation I could find states the following:

Option	Recommended Setting	Why
Enable extra low-level configuration options	☒ Enabled	Required for RP2040-specific tuning
Micro-controller Architecture	Raspberry Pi RP2040/RP235x	Matches your board
Processor model	rp2040	Correct for Raspberry Pi Pico
Bootloader offset	No bootloader	Pico uses drag-and-drop UF2 flashing
Flash chip	W25Q080 with CLKDIV 2	Default for most Pico boards; safe choice
Communication Interface	USB_SERIAL	Needed to connect via USB to your K1 Max
USB IDs	Leave default	Optional unless you need custom device IDs
Optimize stepper code for 'step on both edges'	☒ Enabled	Improves stepper performance (recommended)
GPIO pins to set at micro-controller startup	Leave blank	Only needed for special pin initialization

Build the configuration

```
jattie@Jattie-PC:~/klipper$ make
Building out/autoconf.h
```

```
Compiling out/src/sched.o
make: arm-none-eabi-gcc: No such file or directory
make: *** [Makefile:65: out/src/sched.o] Error 127
```



That error means your system is missing the ARM cross-compiler toolchain—specifically arm-none-eabi-gcc, which is required to compile firmware for ARM-based microcontrollers like the RP2040.

This is resolved by installing the missing tools

```
jattie@Jattie-PC:~/klipper$ sudo apt update
sudo apt install gcc-arm-none-eabi binutils-arm-none-eabi libnewlib-arm-none-eabi
```

Build then runs to completion as expected.

```
jattie@Jattie-PC:~/klipper$ make
Compiling out/src/sched.o
Compiling out/src/command.o
Compiling out/src/basecmd.o
Compiling out/src/debugcmds.o
Compiling out/src/initial_pins.o
Compiling out/src/gpiocmds.o
Compiling out/src/stepper.o
Compiling out/src/endstop.o
Compiling out/src/trsync.o
Compiling out/src/adccmds.o
Compiling out/src/spicmds.o
Compiling out/src/i2ccmds.o
Compiling out/src/pwmcmds.o
Compiling out/src/buttons.o
Compiling out/src/tmcuart.o
Compiling out/src/neopixel.o
Compiling out/src/pulse_counter.o
Compiling out/src/lcd_st7920.o
Compiling out/src/lcd_hd44780.o
Compiling out/src/spi_software.o
Compiling out/src/i2c_software.o
Compiling out/src/thermocouple.o
Compiling out/src/sensor_adxl345.o
Compiling out/src/sensor_lis2dw.o
Compiling out/src/sensor_mpu9250.o
Compiling out/src/sensor_icm20948.o
Compiling out/src/sensor_hx71x.o
Compiling out/src/sensor_ads1220.o
Compiling out/src/sensor_ldc1612.o
Compiling out/src/sensor_angle.o
Compiling out/src/sensor_bulk.o
Compiling out/src/sos_filter.o
Compiling out/src/load_cell_probe.o
Compiling out/src/rp2040/main.o
Compiling out/src/rp2040/watchdog.o
Compiling out/src/rp2040/gpio.o
Compiling out/src/rp2040/adc.o
Compiling out/src/generic/armcm_boot.o
Compiling out/src/generic/armcm_irq.o
Compiling out/src/generic/armcm_reset.o
Compiling out/src/generic/crc16_ccitt.o
Compiling out/src/rp2040/timer.o
Compiling out/src/generic/timer_irq.o
```

```
Compiling out/src/rp2040/bootrom.o
Compiling out/src/rp2040/usbserial.o
Compiling out/src/generic/usb_cdc.o
Compiling out/src/rp2040/chipid.o
Compiling out/src/rp2040/hard_pwm.o
Compiling out/src/rp2040/spi.o
Compiling out/src/rp2040/i2c.o
Building out/compile_time_request.o
Version: v0.13.0-213-gd34d3b05b
Building rp2040 stage2 out/stage2.o
Preprocessing out/src/rp2040/rpxxxx_link.ld
Linking out/klipper.elf
Building out/lib/elf2uf2/elf2uf2
Creating uf2 file out/klipper.uf2
```

Flash the binary to the Raspberry PI Pico 2040

- Plug the Pico into your computer while holding the BOOTSEL button, it will mount as a USB drive.
- Copy klipper.uf2 to the Pico drive, it will reboot automatically.

Connect the board to the main host application

From:
<http://3dfaq.net/> - **3D Printing Wiki**

Permanent link:
http://3dfaq.net/04_projects/03_rpi2040_pico_mmu?rev=1755635610

Last update: **2025/08/19 20:33**

