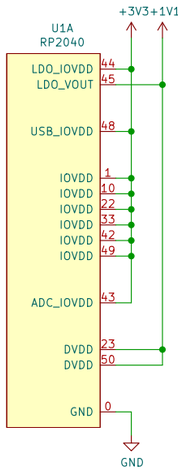
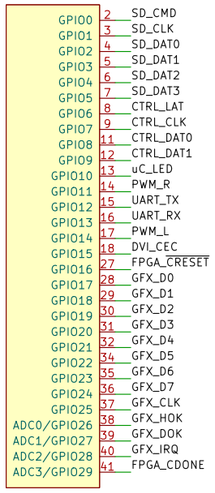
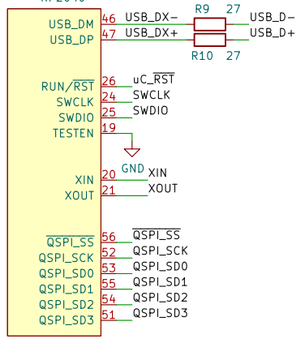
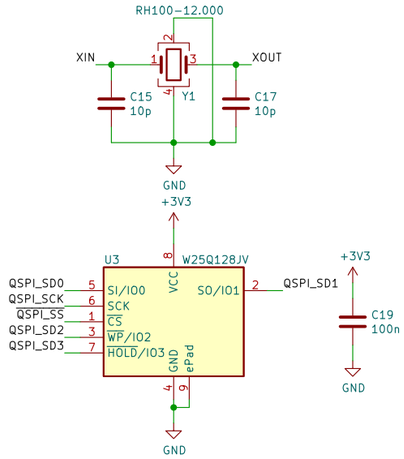
Microcontroller + IO

The microcontroller is responsible for programming the FPGA, generating audio signals, reading controller ports, accessing SD card, running game logic, streaming texture assets from flash to VRAM, and providing a USB programming interface for the whole system via the UF2 bootloader in RP2040's bootrom



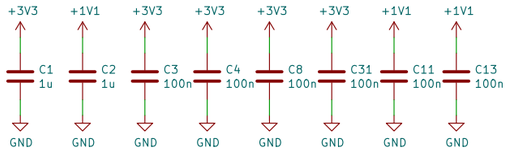




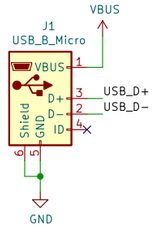


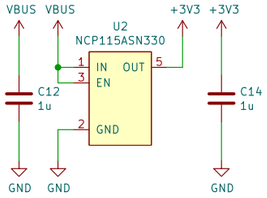
GFX is an 8 bit parallel bus I made up, with a free−running clock, flow control handshake and an interrupt line

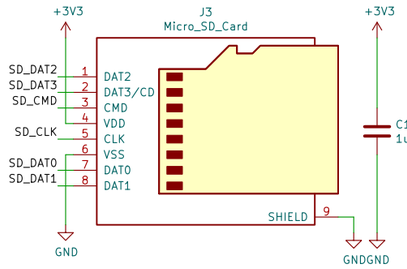
For GFX CLK we want the option of a GPCK generator (although PIO is probably fine). For all others, just use PIO. Need parallel bus to be on consecutive pins for PIO use.



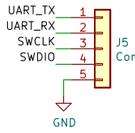
Power chain is: USB VBUS −> 3V3 (via discrete LDO) −> FPGA/MCU core (via RP2040's LDO)





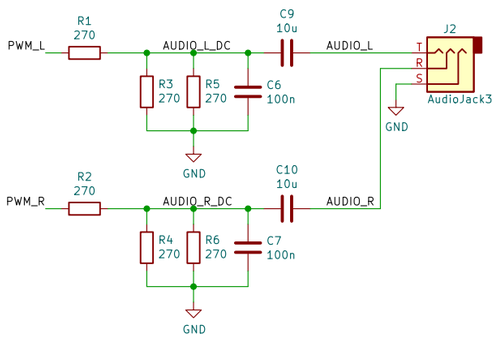


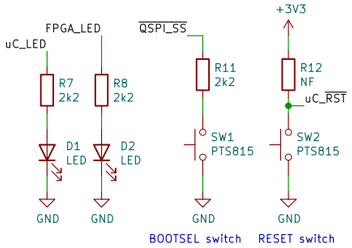
Optional debug header



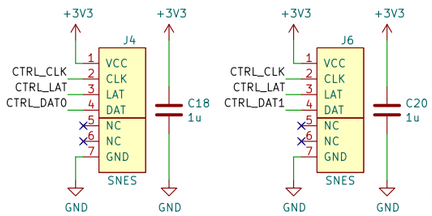
Relying on RP2040 pad pulls for correct SD operation

−3dB at ~17.7 kHz
If I was a real engineer there would be a separate audio supply and a buffer





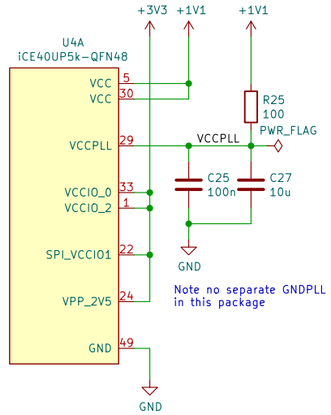
BOOTSEL switch RESET switch

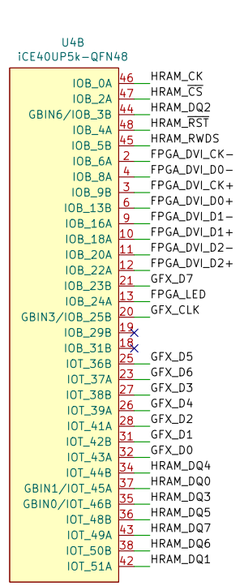


The SNES supplies 5V to its controllers, but the original controllers' CD4021B shift registers are rated for 3−>15 V operation. Hopefully modern replacements off Amazon will also tolerate 3V3 VCC.

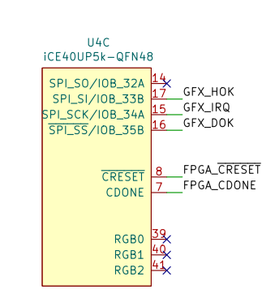
FPGA + Video

The FPGA contains 3D graphics hardware, a HyperRAM interface for VRAM, a bus endpoint for the 8−bit GFX bus from the MCU, and a DVI video−out circuit.





Note GFX_CLK must be on a GBIN



CRESET is initially pulled down by default MCU pulldown state (note the iCE40 does *not* apply any internal pull to CRESET at power−on).

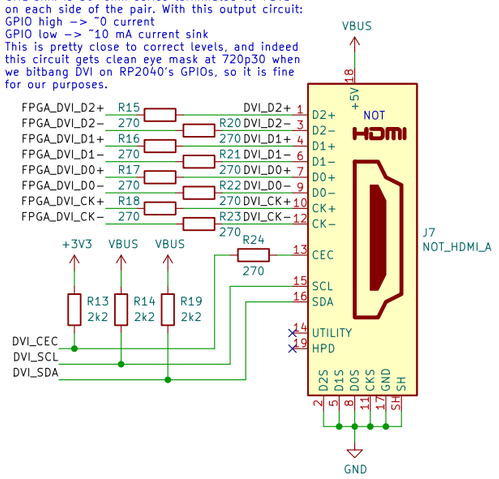
MCU boots, ensures SPI_SS is asserted, and deasserts CRESET, to put FPGA into slave configuration mode. MCU then decompresses FPGA bitstream from flash and streams to FPGA.

Once FPGA is configured, MCU reconfigures the GPIOs it used for FPGA configuration (yay PIO), to connect to FPGA gateware's GFX bus.

We are using RP2040's core LDO to provide the UP5k's core supply, and bumping RP2040's core supply up to 1.2V in software before configuring the FPGA should work out fine (he says). This *breaks* UP5k's power sequencing rules, but have seen schematics of boards which have 3V3 and 1V2 come up simultaneously (e.g. iCESugar) so we should get away with it for the first prototype.

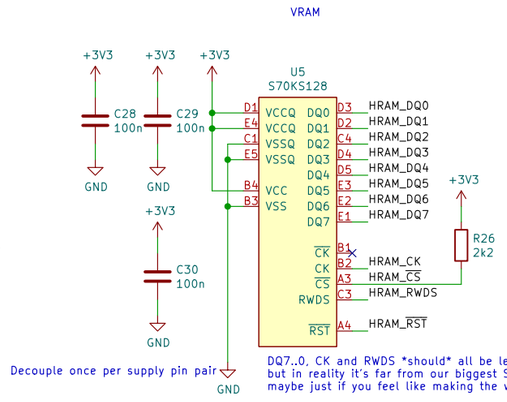
Note the *nominal* max on RP2040 LDO is 100 mA, but like much of RP2040 it is a little overengineered, and on first test batch we have seen no out−of−regulation LDOs at a 200 mA test current. In fact the tester is not able to draw enough current to make the LDO fail, which is a pain in the ass

CML sink is 50−ohm series terminated to +3V3 on each side of the pair. With this output circuit:
GPIO high −> ~0 current
GPIO low −> ~10 mA current sink
This is pretty close to correct levels, and indeed this circuit gets clean eye mask at 720p30 when we bitbang DVI on RP2040's GPIOs, so it is fine for our purposes.



Note the DDC is unconnected on this board, we just provide the mandatory pullups (you aren't likely to do anything above the basic DVI modes on the UP5k anyway, so no need for EDID)

CEC is connected to the microcontroller :)



Decouple once per supply pin pair

DQ7..0, CK and RWDS *should* all be length matched, but in reality it's far from our biggest SI problem, so maybe just if you feel like making the wiggles?

Luke Wren

Sheet: /
File: picostation.sch

Title: PicoStation 3D

Size: A3	Date: 2020−10−24	**Rev: A**
KiCad E.D.A.	kicad 5.99.0−unknown−8eca23aabe115ubuntu20.04.1	Id: 1/1