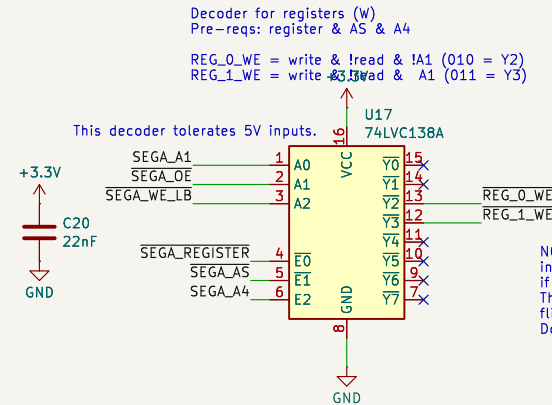
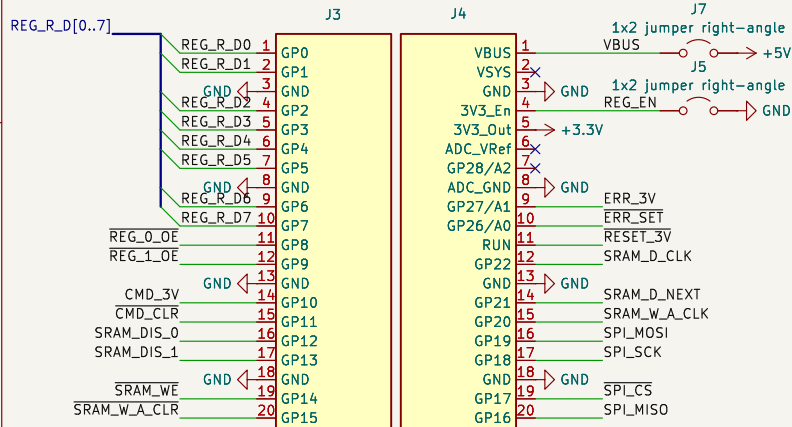


The 3.3V regulator built into the microcontroller is documented as only supplying a max of 300mA to external circuitry.

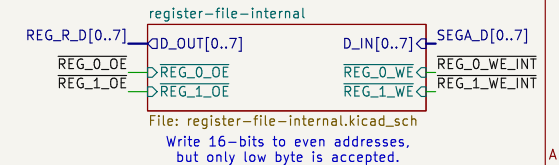
Rather than use that, we use our own regulator in the cartridge board. It can power the microcontroller and all 3V logic chips. Then we power the microcontroller through the 3V3\_OUT and GND pins.

One jumper connects 3V3\_EN connects to GND to disable the board's regulator. Another jumper connects VBUS to the +5V supply to power the 5V chips.

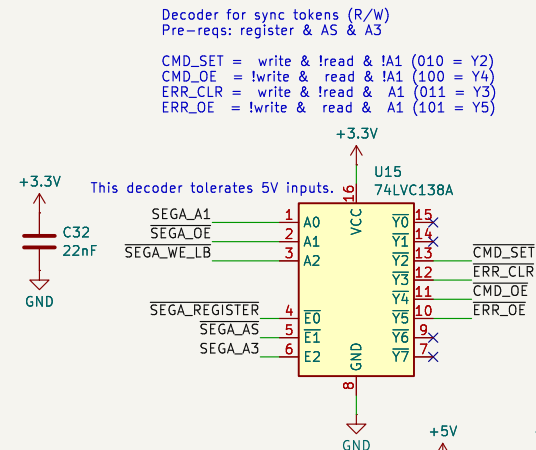
While programming the microcontroller in-place via USB, the jumper should be on the VBUS pins. While operating in the Sega, the jumper should be on the 3V3\_EN pins.



NOTE: Because of limitations in the number of inputs in our address decoders, a conflict is possible if the addresses 0xA13018 - 0xA1301F are written. This range could trigger both a sync token to be flipped and a register to be written at the same time. Don't do it.



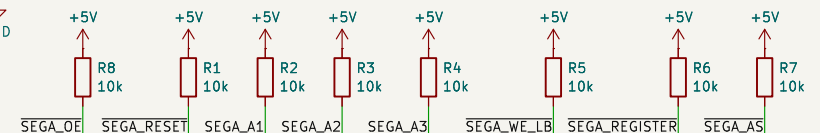
Register addresses  
CMD: 0xA13010  
ARG: 0xA13012  
Command token address  
0xA13008  
Error token address  
0xA1300A



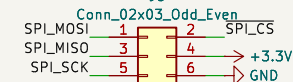
These bridges can be used to snoop on these signals, or cut with a knife to hijack them.



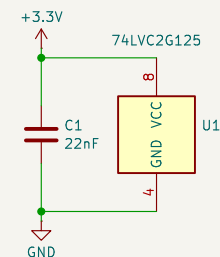
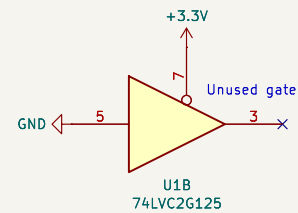
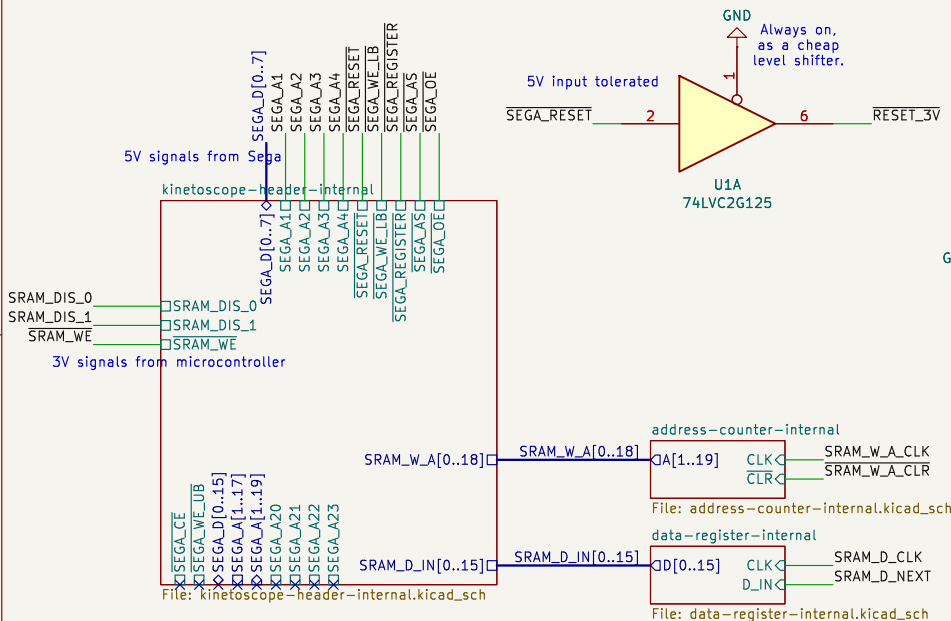
While powering the microcontroller via USB without the Sega (during programming and testing), pull-up resistors on these lines keep inputs from floating. They can be set low for testing with jumper wires in the cartridge pin header.



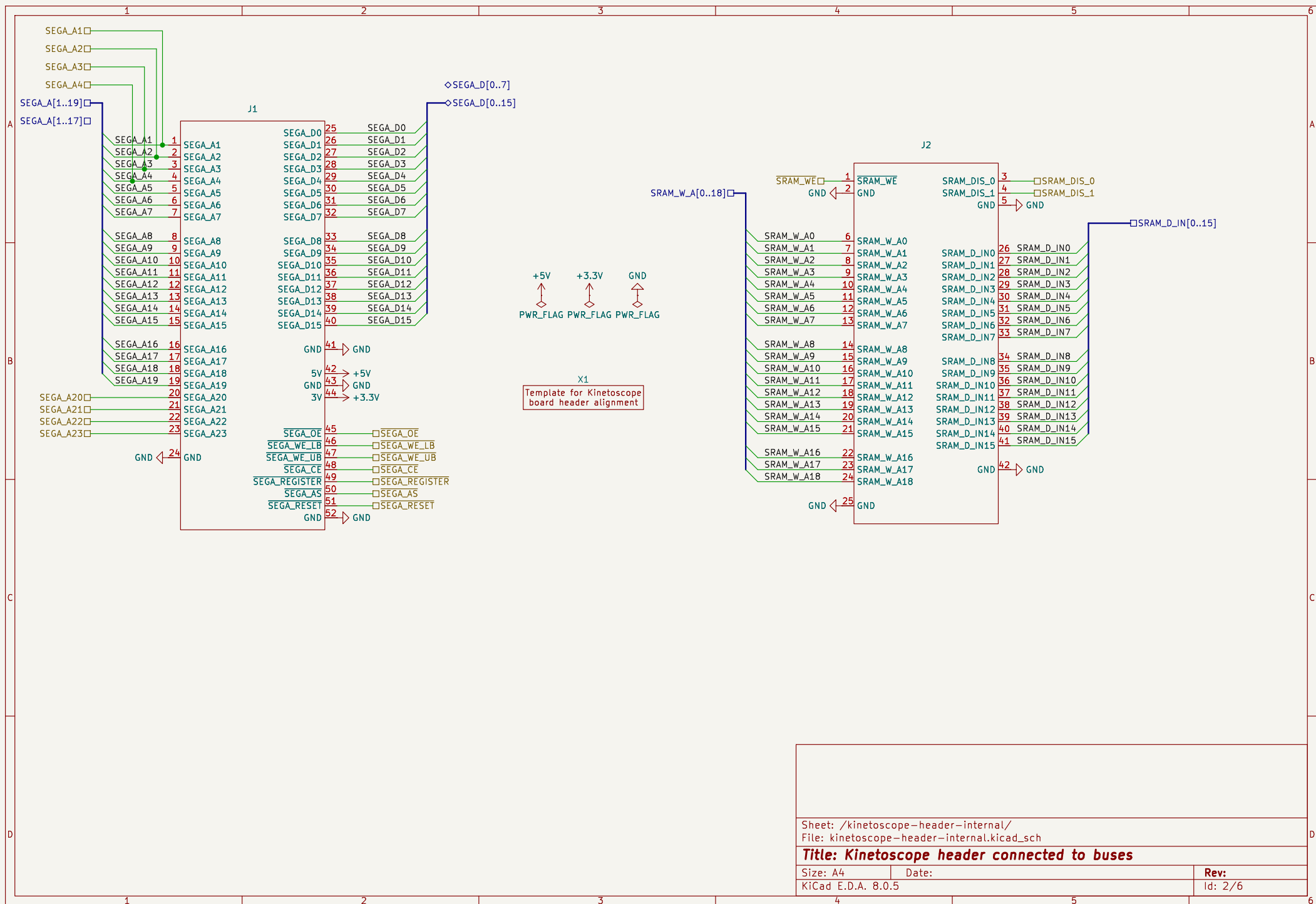
Connector for optional Adafruit Ethernet featherwing (jumper wires or adapter needed)



X2  
Template for RPiPicoW board header alignment



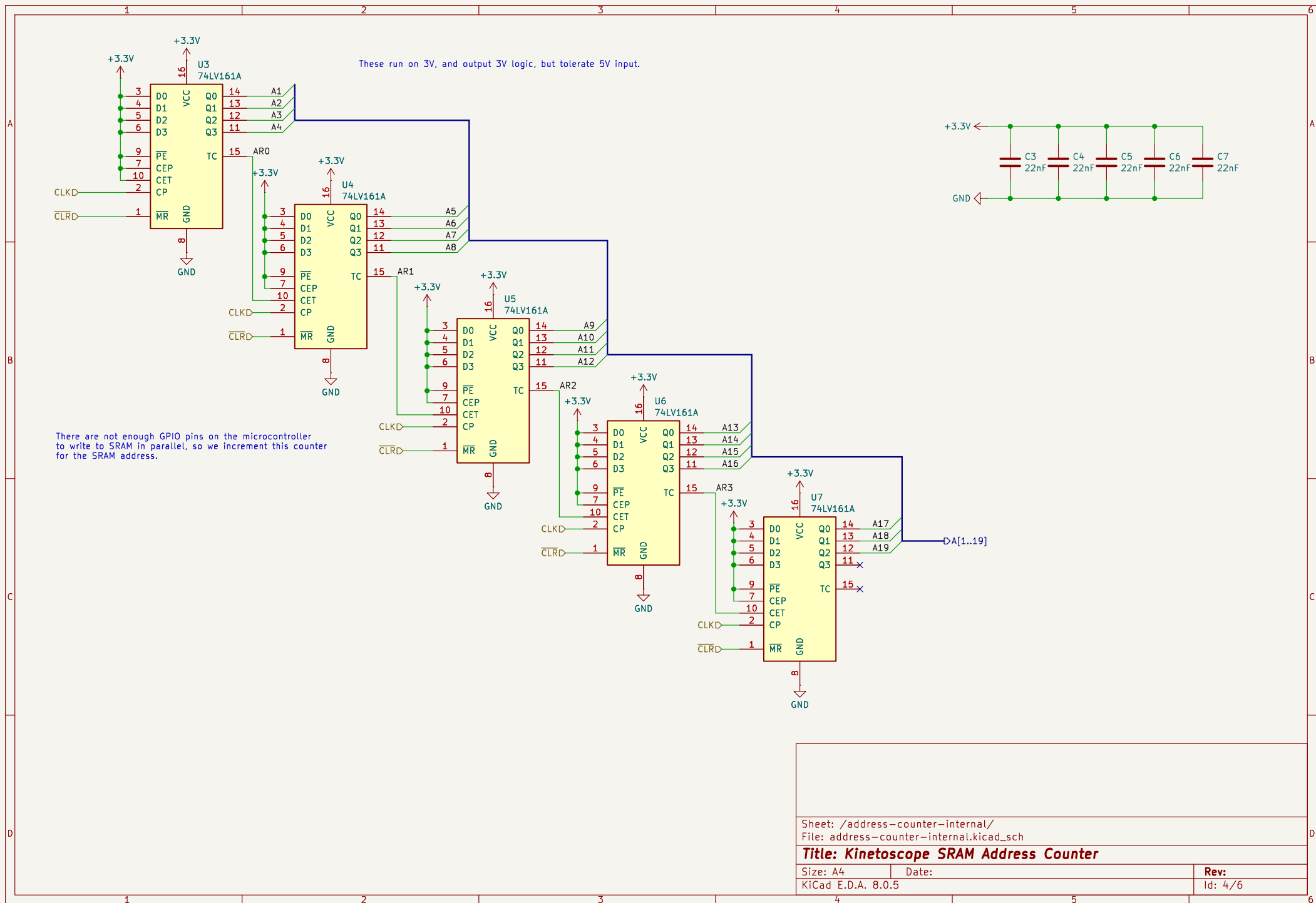
Sheet: /			D
File: microcontroller.kicad_sch			
<b>Title: Kinetoscope Microcontroller and Register Board</b>			
Size: A4	Date:	Rev:	
KiCad E.D.A. 8.0.5		Id: 1/6	



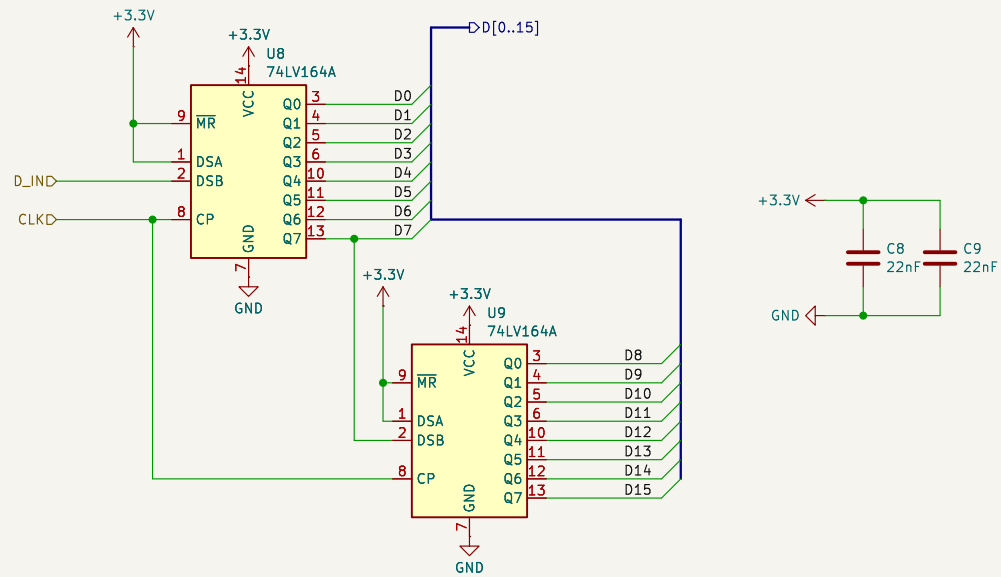
Sheet: /kinetoscope-header-internal/  
File: kinetoscope-header-internal.kicad\_sch

**Title: Kinetoscope header connected to buses**

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.5		Id: 2/6



There are not enough GPIO pins on the microcontroller to write to SRAM in parallel, so we fill this data register serially.

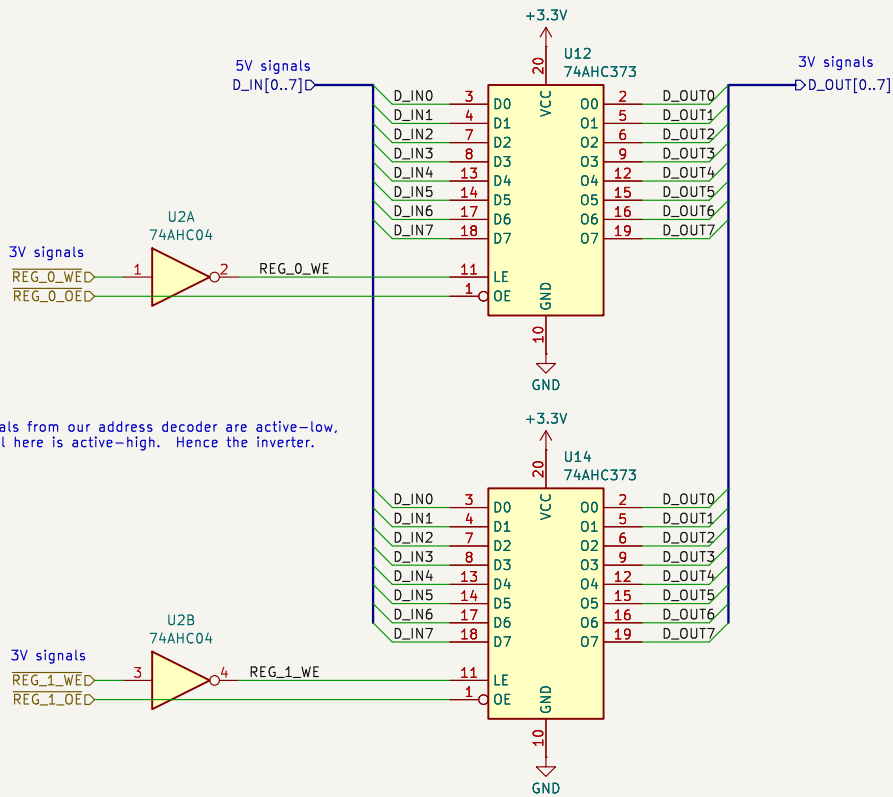


Sheet: /data-register-internal/		
File: data-register-internal.kicad_sch		
<b>Title: Kinetoscope SRAM Data Register</b>		
Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.5	Id: 5/6	

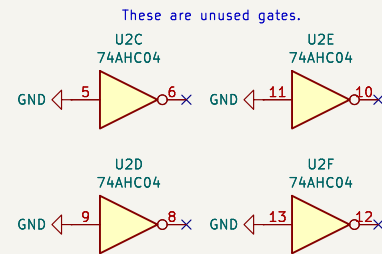
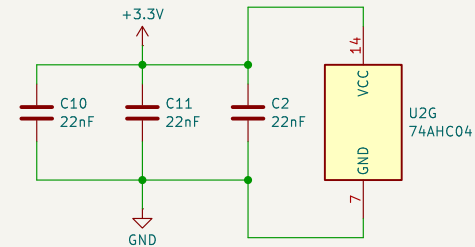
This is a pair of 8-bit registers that can be independently read and written by the two processors (one reads, one writes).

The registers are assumed to be written from the Sega's shared data bus and read by the microcontroller. Each register has its own write-enable and read-enable, but only one register can be read at a time, and only one can be written at a time. It is safe to read from one register while writing to another.

Everything runs on +3.3V, but must be compatible with 5V inputs. If using a slightly different part number, please check input voltages in the data sheet. The data outputs are 3.3V, which the Sega can read.



The write-enable signals from our address decoder are active-low, but the latch signal here is active-high. Hence the inverter.



Sheet: /register-file-internal/  
File: register-file-internal.kicad\_sch

**Title: Kinetoscope Register File connected to buses**

Size: A4

Date:

Rev:

KiCad E.D.A. 8.0.5

Id: 6/6

These are synchronization tokens between the M68k and the microcontroller.

The tokens will be cleared on boot/reset.

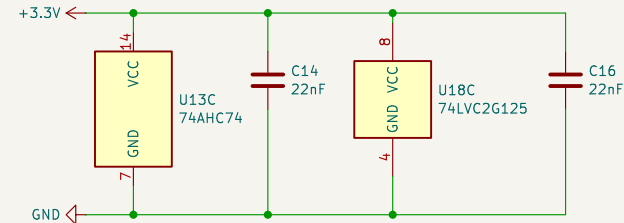
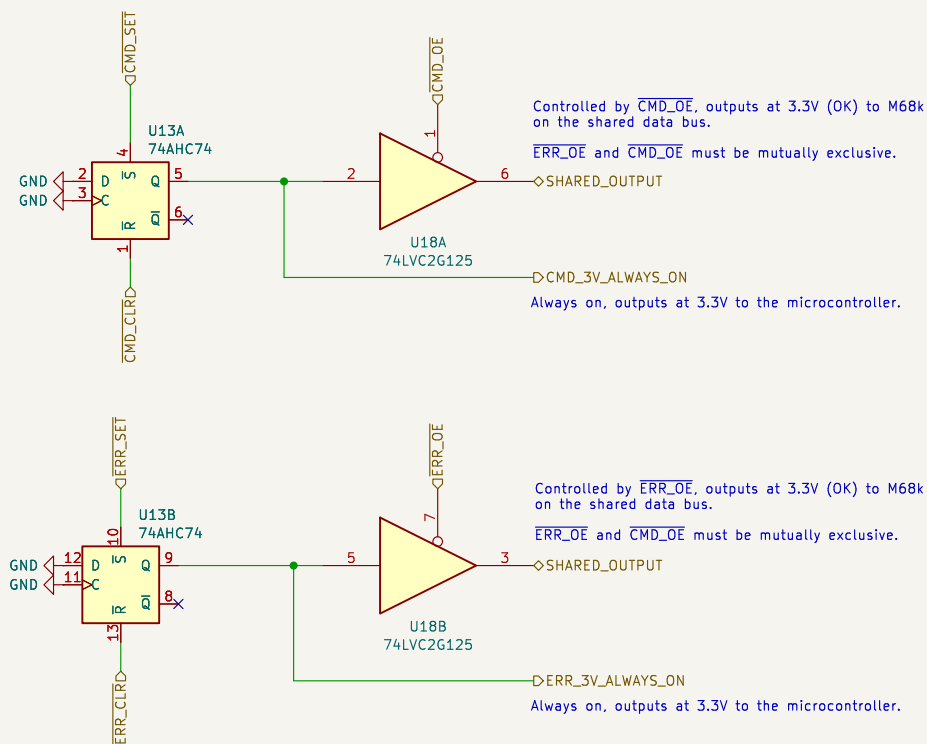
The M68k will be able to set the command token ( $\overline{\text{CMD\_SET}}$ ), indicating to the microcontroller that a command is ready to be executed.

The microcontroller will be able to clear the command token ( $\text{CMD\_CLR}$ ), indicating to the M68k that the command has been executed.

Command codes and arguments are passed through a separate register file.

The microcontroller can also flag an error ( $\overline{\text{ERR\_SET}}$ ) to the M68k, who can clear it ( $\text{ERR\_CLR}$ ) after noticing.

The M68k and microcontroller both will be able to read the current state of either token through  $\text{OUTPUT\_5V}$ , which goes to the shared data bus with 5V logic. 3.3V version of both tokens are output to the microcontroller on dedicated lines.



Sheet: /sync-token-internal/  
File: sync-token-internal.kicad\_sch

**Title: Kinetoscope Sync Token**

Size: A4  
KiCad E.D.A. 8.0.5

Date:

Rev:  
Id: 7/6