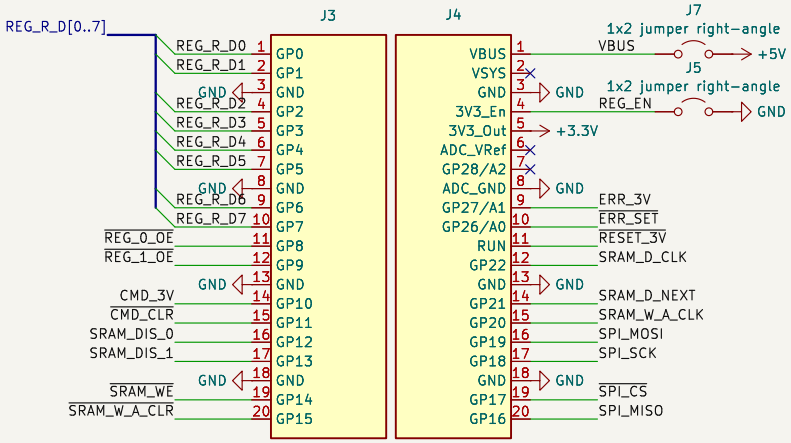
The 3.3V regulator built into the microcontroller is documented as only supplying a max of 300mA to external circuitry.

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.

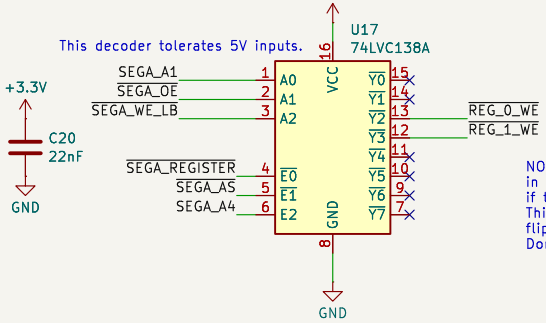
While programming the microcontroller in-place via USB, the jumper should be on the VBUS pins. While operating in the Sega, the jumper should be on the 3V3_EN pins.

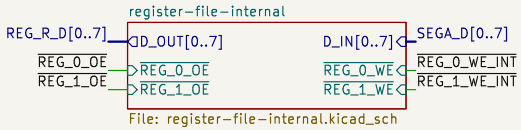


Decoder for registers (W)
Pre-reqs: register & AS & A4

REG_0_WE = write & !read & !A1 (010 = Y2)
REG_1_WE = write & !read & A1 (011 = Y3)

This decoder tolerates 5V inputs.





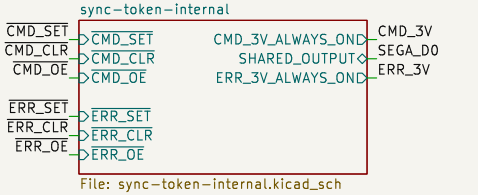
Write 16-bits to even addresses, but only low byte is accepted.

Register addresses
CMD: 0xA13010
ARG: 0xA13012

Command token address
0xA13008

Error token address
0xA1300A

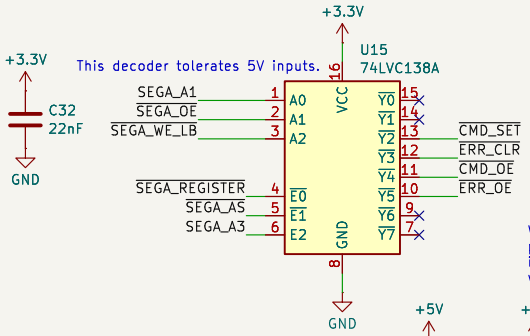
NOTE: Because of limitations in the number of inputs in our address decoders, a conflict is possible if the addresses 0xA13018 - 0xA1301F are written. This range could trigger both a sync token to be flipped and a register to be written at the same time. Don't do it.



Decoder for sync tokens (R/W)
Pre-reqs: register & AS & A3

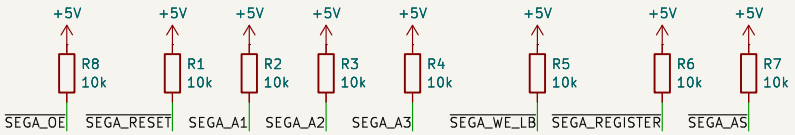
CMD_SET = write & !read & !A1 (010 = Y2)
CMD_OE = !write & read & !A1 (100 = Y4)
ERR_CLR = write & !read & A1 (011 = Y3)
ERR_OE = !write & read & A1 (101 = Y5)

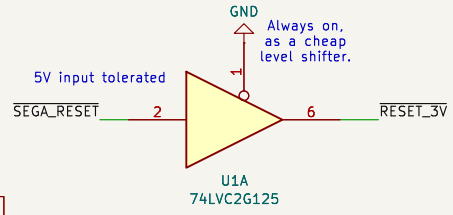
This decoder tolerates 5V inputs.

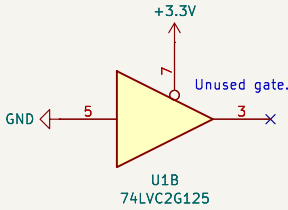


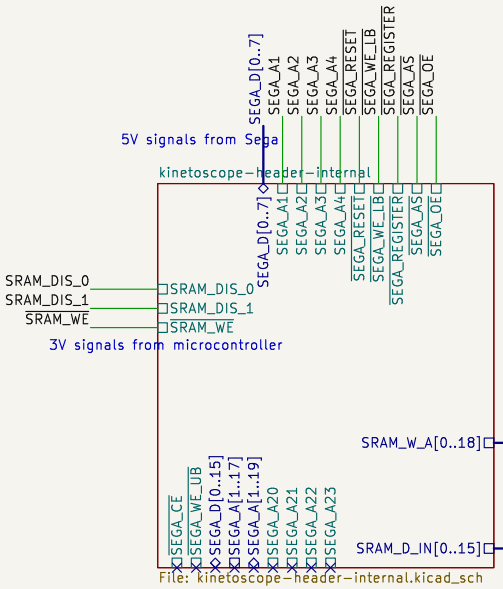
These bridges can be used to snoop on these signals, or cut with a knife to hijack them.

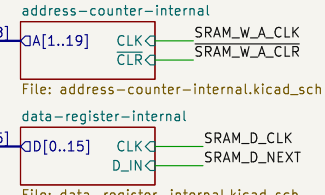
While powering the microcontroller via USB without the Sega (during programming and testing), pull-up resistors on these lines keep inputs from floating. They can be set low for testing with jumper wires in the cartridge pin header.

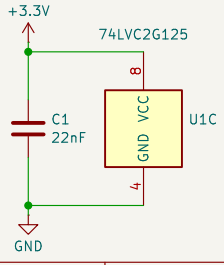




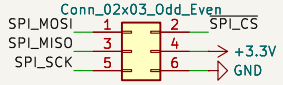








Connector for optional Adafruit Ethernet featherwing (jumper wires or adapter needed)



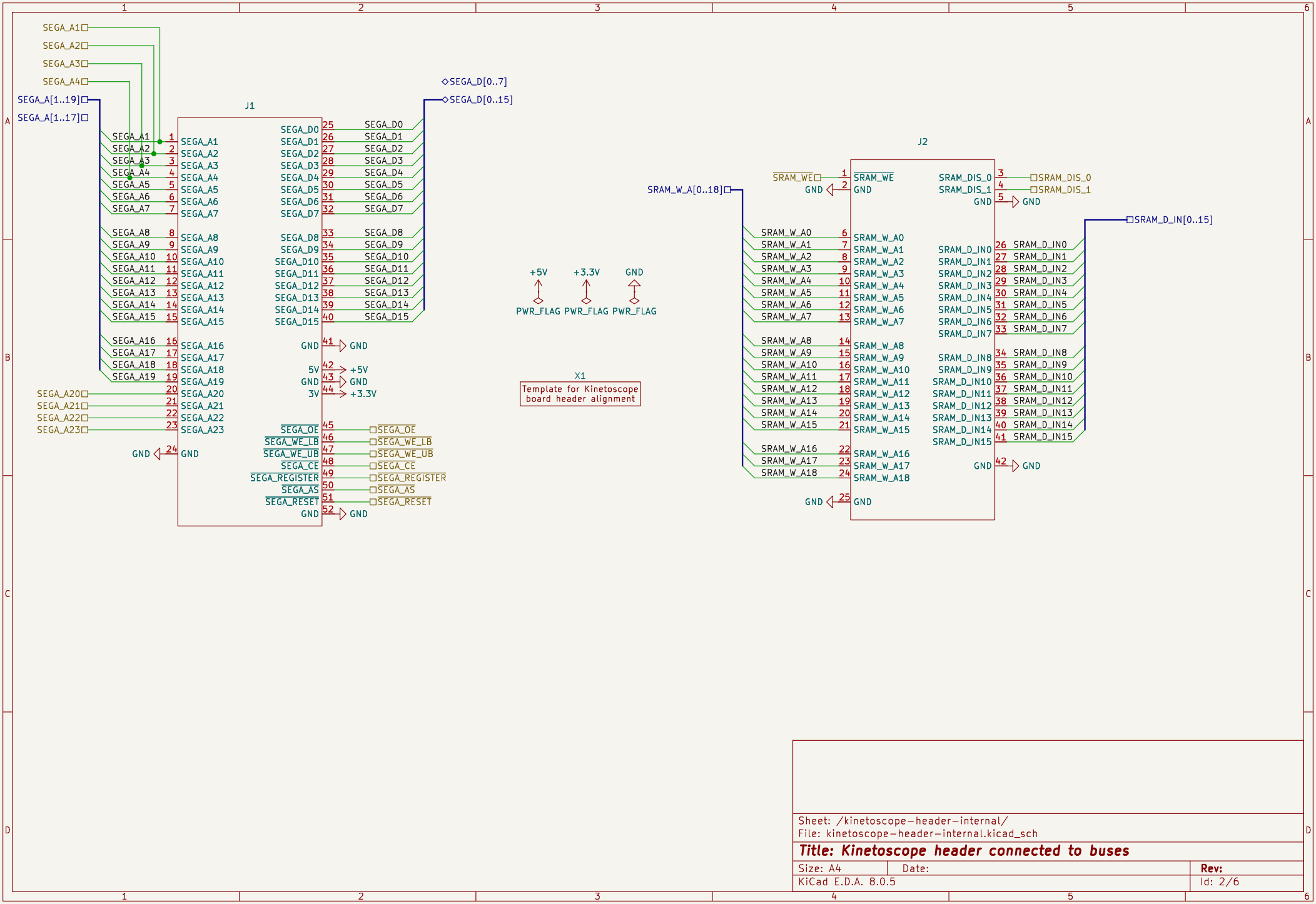
X2

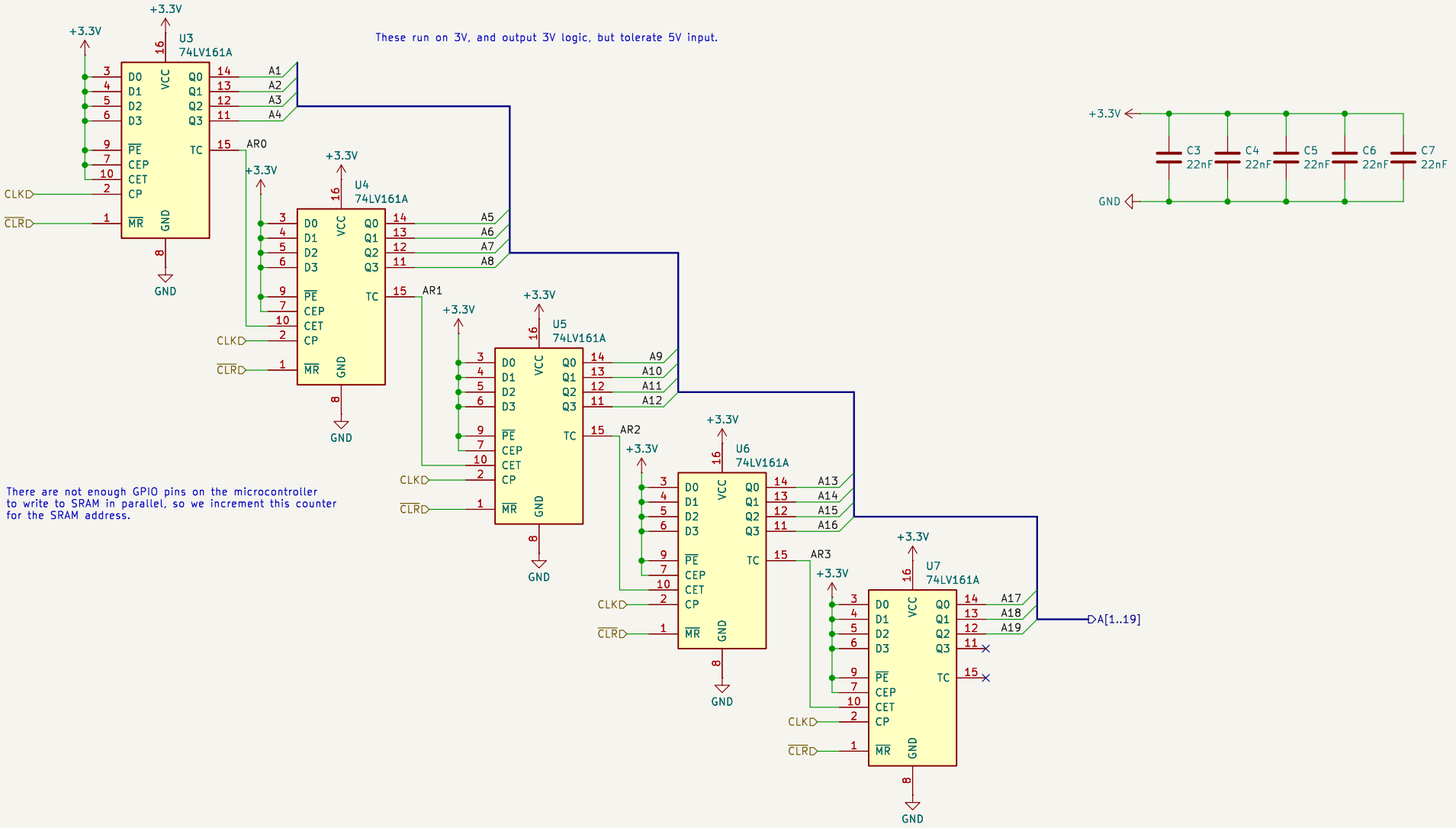
Template for RPiPicoW board header alignment

Sheet: /
File: microcontroller.kicad_sch

Title: Kinetoscope Microcontroller and Register Board

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.5		Id: 1/6


SEGA_A1
SEGA_A2
SEGA_A3
SEGA_A4
SEGA_A[1..19]
SEGA_A[1..17]
J1
SEGA_D[0..7]
SEGA_D[0..15]
SEGA_A20
SEGA_A21
SEGA_A22
SEGA_A23
GND
SEGA_OE
SEGA_WE_LB
SEGA_WE_UB
SEGA_CE
SEGA_REGISTER
SEGA_AS
SEGA_RESET
5V
3V
+5V
+3.3V
PWR_FLAG
X1
Template for Kinetoscope board header alignment
J2
SRAM_W_A[0..18]
SRAM_WE
SRAM_DIS_0
SRAM_DIS_1
SRAM_D_IN[0..15]
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

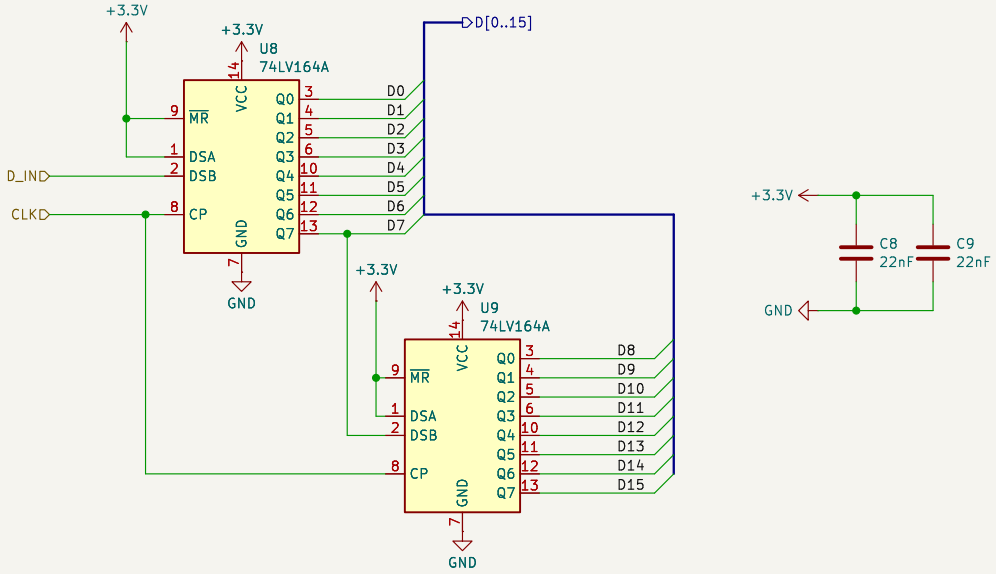


Sheet: /address-counter-internal/
File: address-counter-internal.kicad_sch

Title: Kinetoscope SRAM Address Counter

Size: A4	Date:	Rev:
KiCad E.D.A. 8.0.5		Id: 4/6

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

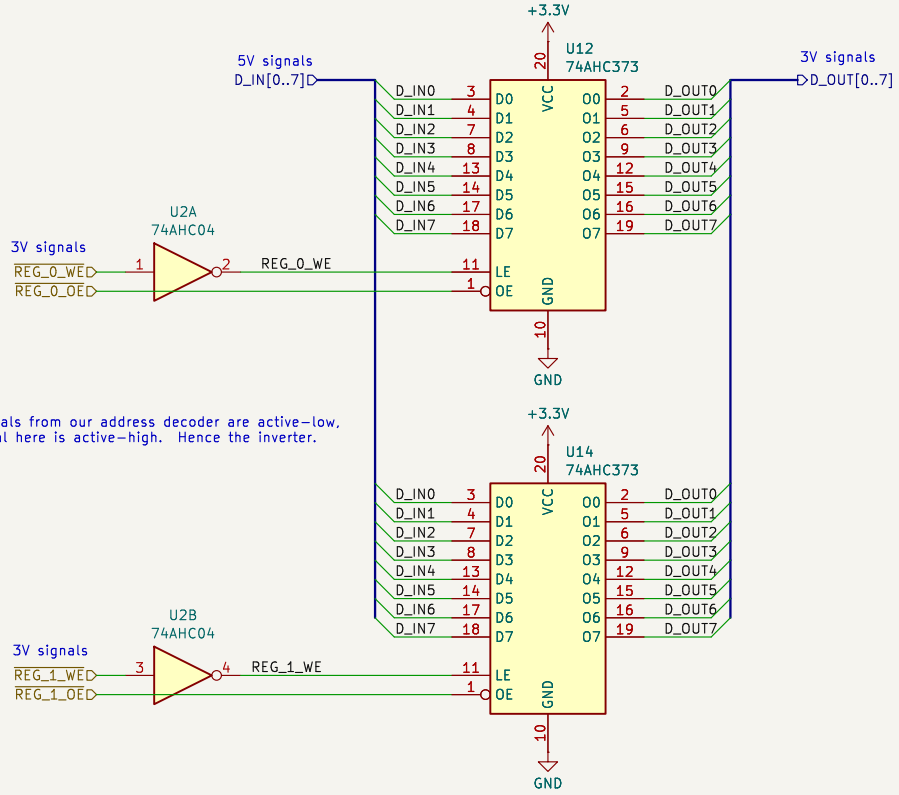


Sheet: /data-register-internal/		
File: data-register-internal.kicad_sch		
Title: Kinetoscope SRAM Data Register		
Size: A4	Date:	**Rev:**
KiCad E.D.A. 8.0.5		Id: 5/6

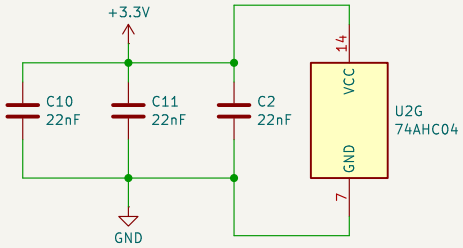
This is a pair of 8-bit registers that can be independently read and written by the two processors (one reads, one writes).

The registers are assumed to 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.

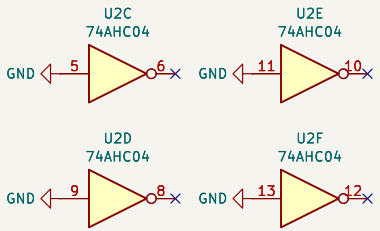
Everything runs on +3.3V, but must be compatible with 5V inputs. If using a slightly different part number, please check input voltages in the data sheet. The data outputs are 3.3V, which the Sega can read.



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



These are unused gates.



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.

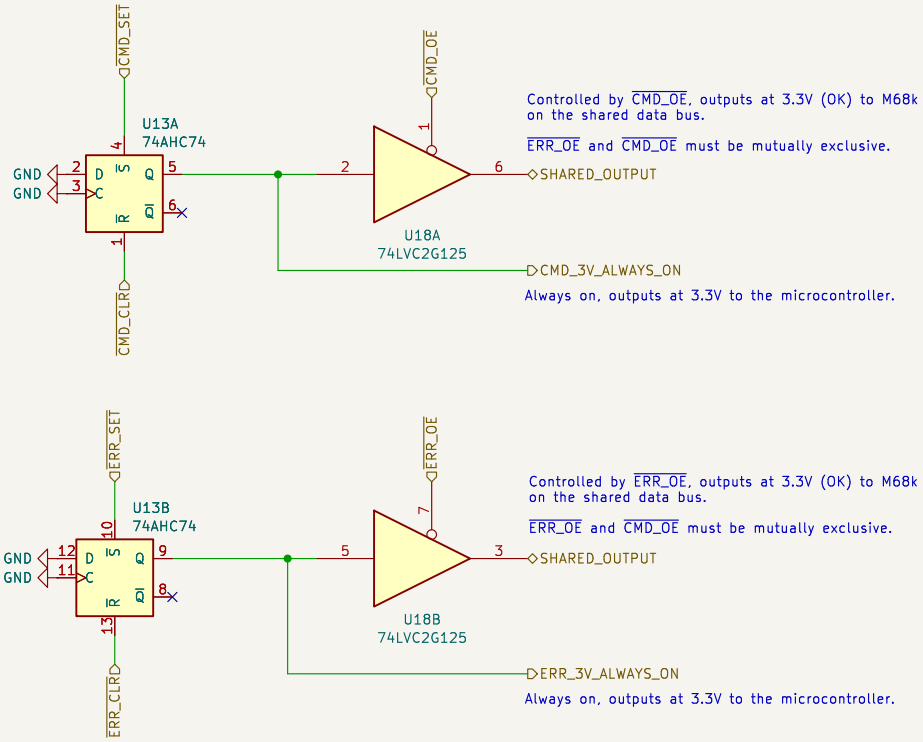
The M68k will be able to set the command token (CMD_SET), indicating to the microcontroller that a command is ready to be executed.

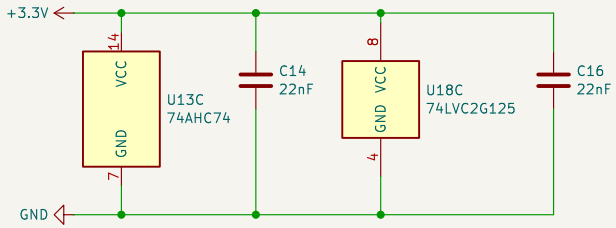
The microcontroller will be able to clear the command token (CMD_CLR), indicating to the M68k that the command has been executed.

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

The microcontroller can also flag an error (ERR_SET) to the M68k, who can clear it (ERR_CLR) after noticing.

The M68k and microcontroller both will be able to read the current state of either token through 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	Date:	Rev:
KiCad E.D.A. 8.0.5		Id: 7/6