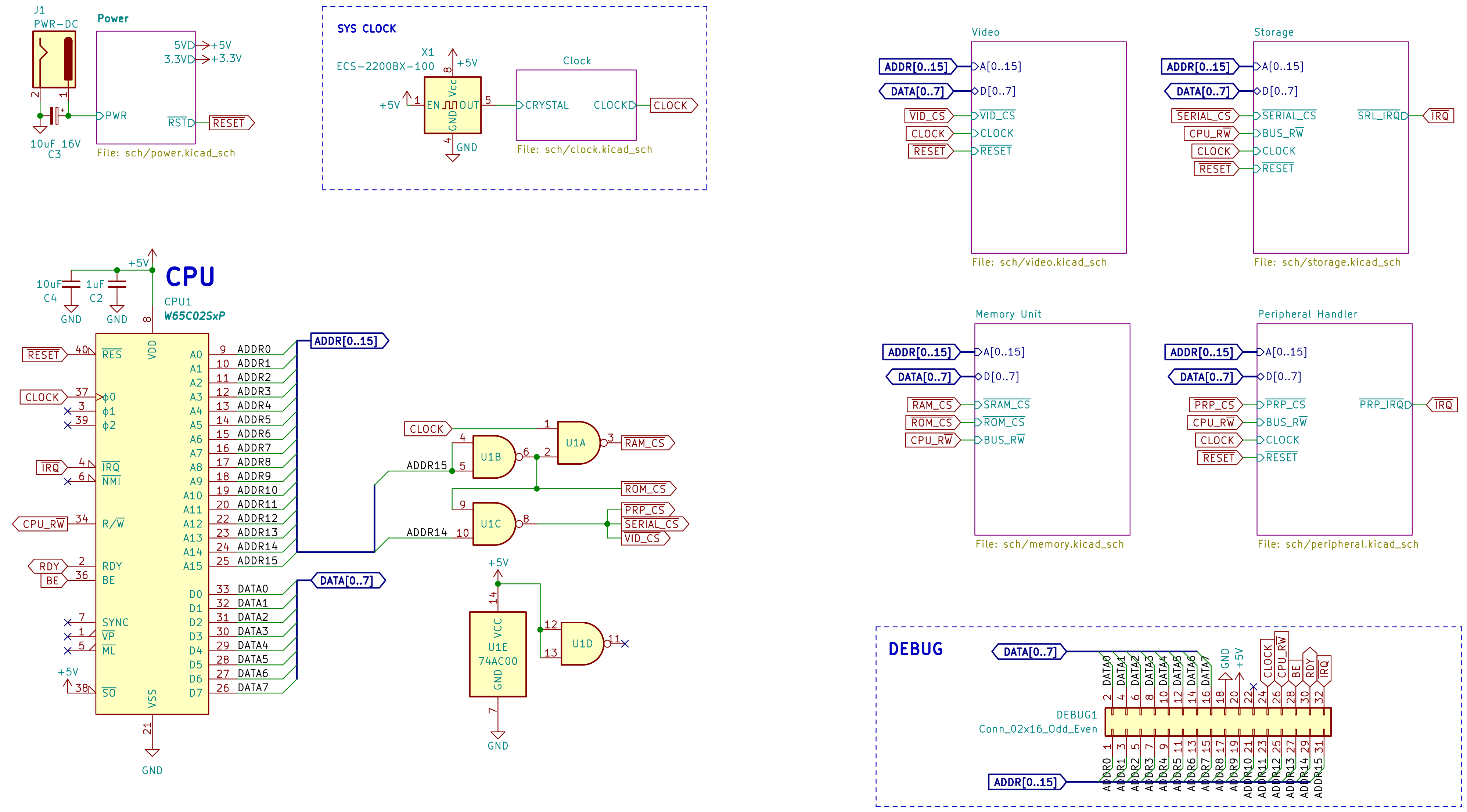
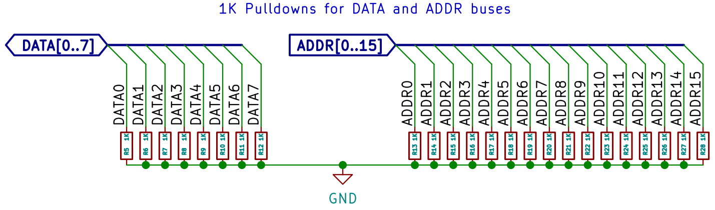




MEMORY MAP

RAM: 0x0000 – 0x3FFF
VIA: 0x6000 – 0x600F
SER: 0x5000 – 0x500F
VID: 0x4800 – 0x480F
ROM: 0x8000 – 0xFFFF

Homebrew 8-Bit Computer
Open Source Hardware Project
https://github.com/zrthxn/8puter
zrthxn

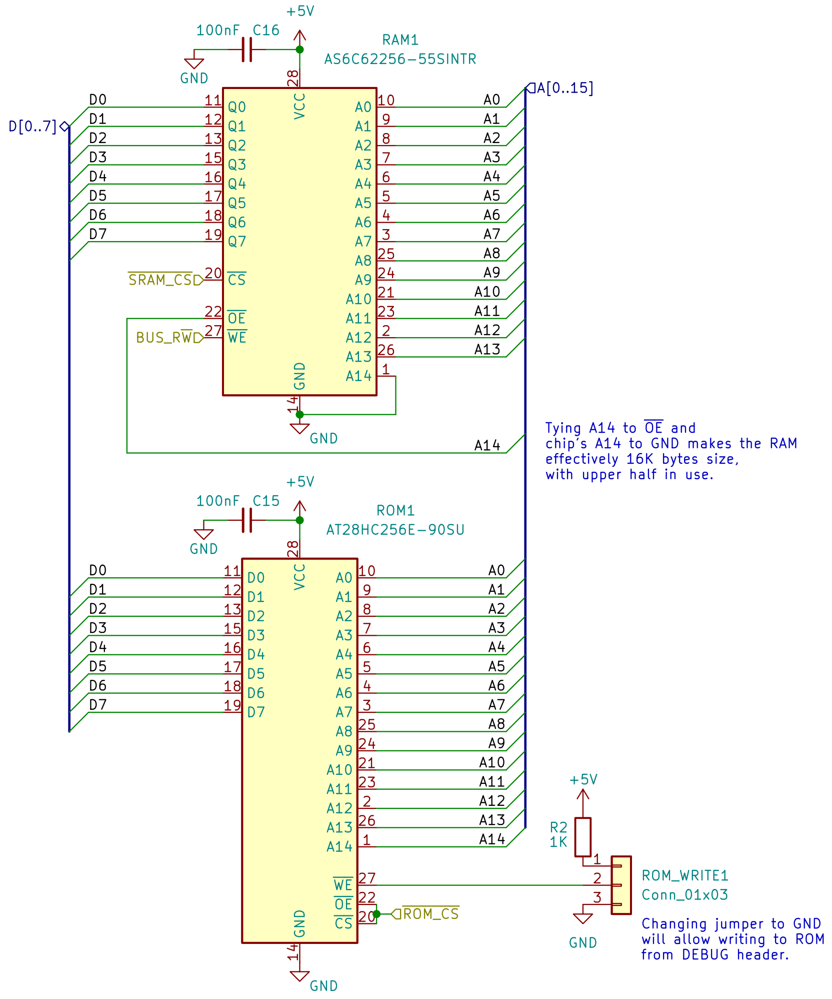
Sheet: /
File: 8puter.kicad_sch

Title: 8puter

Size: A4	Date:	**Rev: 1.0**
KiCad E.D.A. eeschema 6.0.4-1.fc35		Id: 1/7

System Memory

RAM: 0x0000 – 0x3FFF
ROM: 0x8000 – 0xFFFF



Tying A14 to OE and chip's A14 to GND makes the RAM effectively 16K bytes size, with upper half in use.

Changing jumper to GND will allow writing to ROM from DEBUG header.

Homebrew 8-Bit Computer
Open Source Hardware Project
https://github.com/zrthxn/8puter
zrthxn

Sheet: /Memory Unit/
File: memory.kicad_sch

Title: 8puter

Size: A4	Date:	**Rev: 1.0**
KiCad E.D.A. eeschema 6.0.4-1.fc35		Id: 2/7

Peripheral Interface

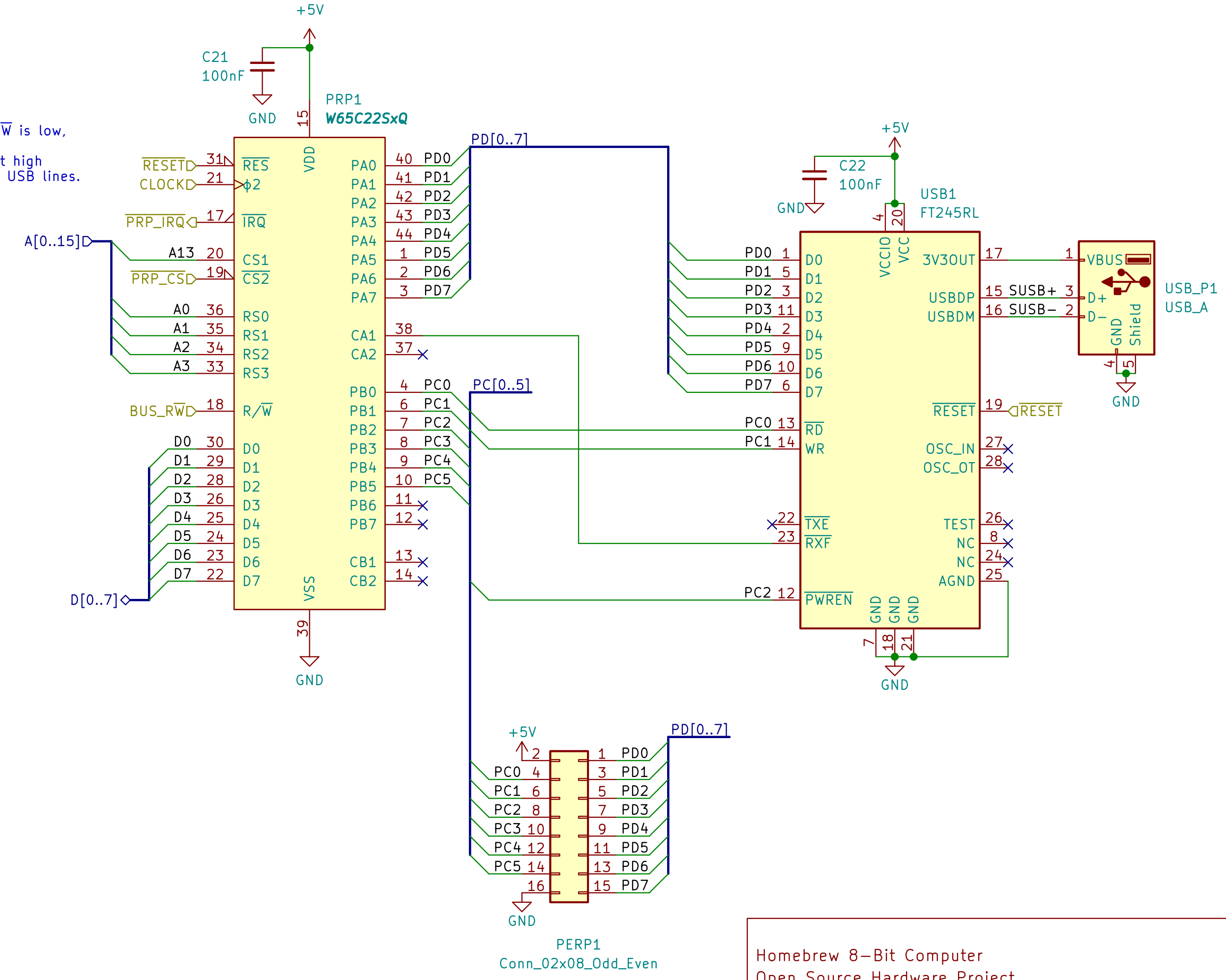
0x6000 – 0x600F

READ FROM USB

Controller's RD (PC0) is set low.
USB data is shifting onto PORT A.
Controller pulls RXF low, PORT A is latched
onto DATA bus and IRQ is set.

WRITE TO USB

When the VIA's both CS are active and BUS_RW is low,
it latches DATA bus onto PORT A.
On the next clock edge, controller's WR is set high
and it starts transmitting from PORT A to the USB lines.
Then WR is set low again.



Homebrew 8-Bit Computer
Open Source Hardware Project
https://github.com/zrthxn/8puter
zrthxn

Sheet: /Peripheral Handler/
File: peripheral.kicad_sch

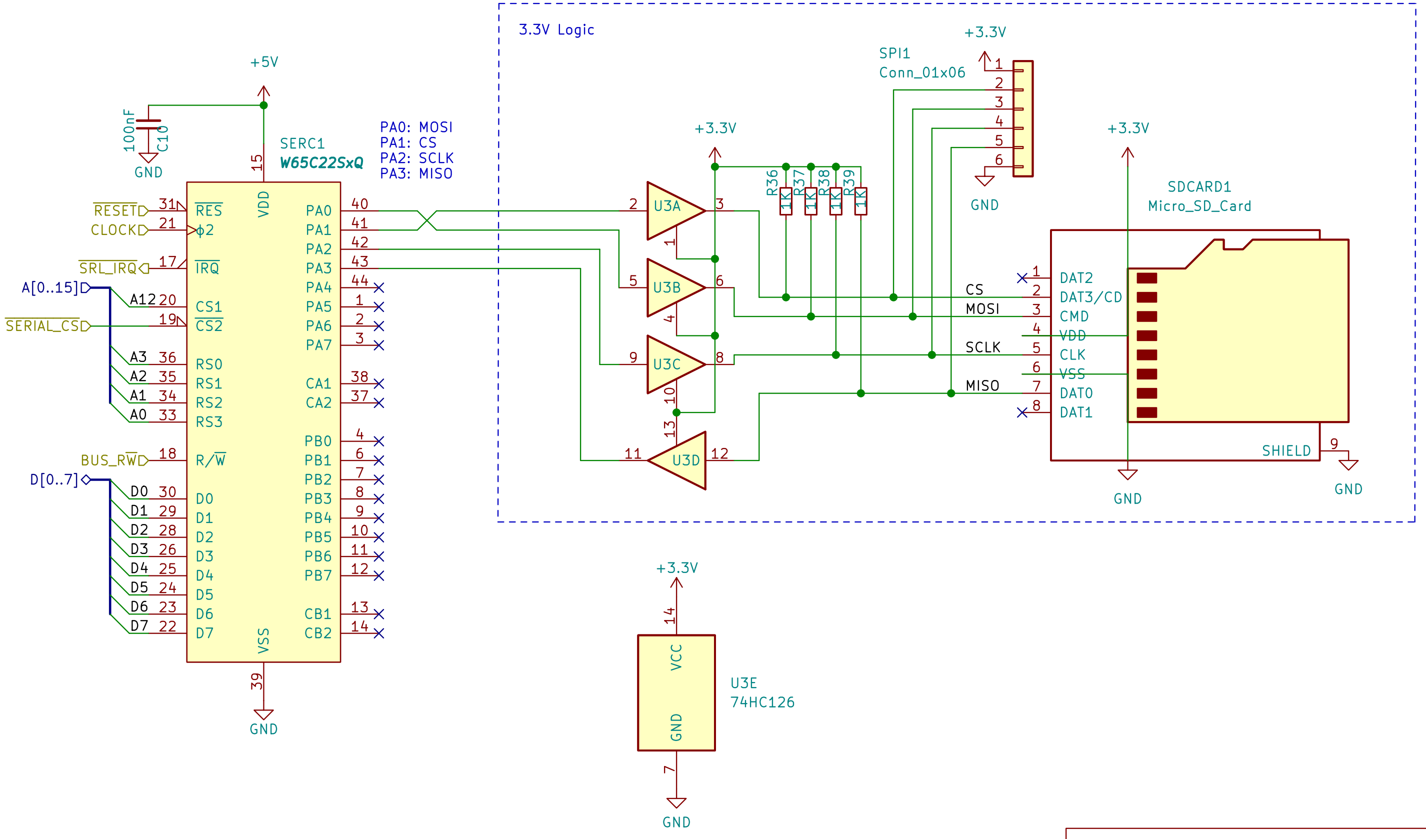
Title: 8puter

Size: A4	Date:	**Rev: 1.0**
KiCad E.D.A. eeschema 6.0.4-1.fc35		Id: 3/7

SPI SD Card Interface

0x5000 – 0x500F

Serial SPI Interface will
be used to interact primarily
with SD cards. There is also a
SPI Header to allow connecting other
devices but only ONE at a time.



Homebrew 8-Bit Computer
Open Source Hardware Project
https://github.com/zrthxn/8puter
zrthxn

Sheet: /Storage/
File: storage.kicad_sch

Title: 8puter

Size: A4	Date:	**Rev: 1.0**
KiCad E.D.A. eeschema 6.0.4-1.fc35		Id: 4/7

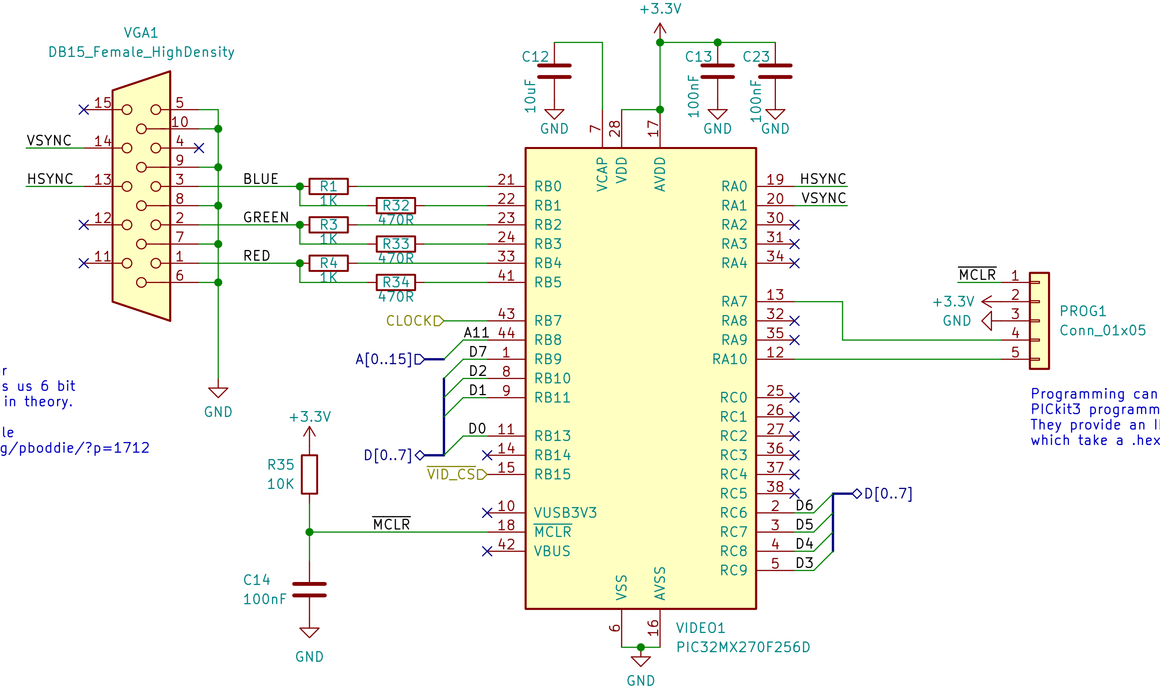
Video Controller

0x4800 – 0x480F

Video will be generated using a microcontroller which is fast enough to generate the VGA timing signals.

CPU will send a single byte to the MCU which can be be a char code or index of glyph, and the MCU just generates the video signal.

This setup is limited in generating graphics but it avoids having to keep a large framebuffer in system memory.



Two bits are used for each color, this gives us 6 bit color and 32 colors in theory.

Ideas from this article
https://blogs.fsfe.org/pboddie/?p=1712

Programming can be done with PICkit3 programmer made by Microchip. They provide an IDE called MPLAB X which take a .hex file and flashes it into memory.

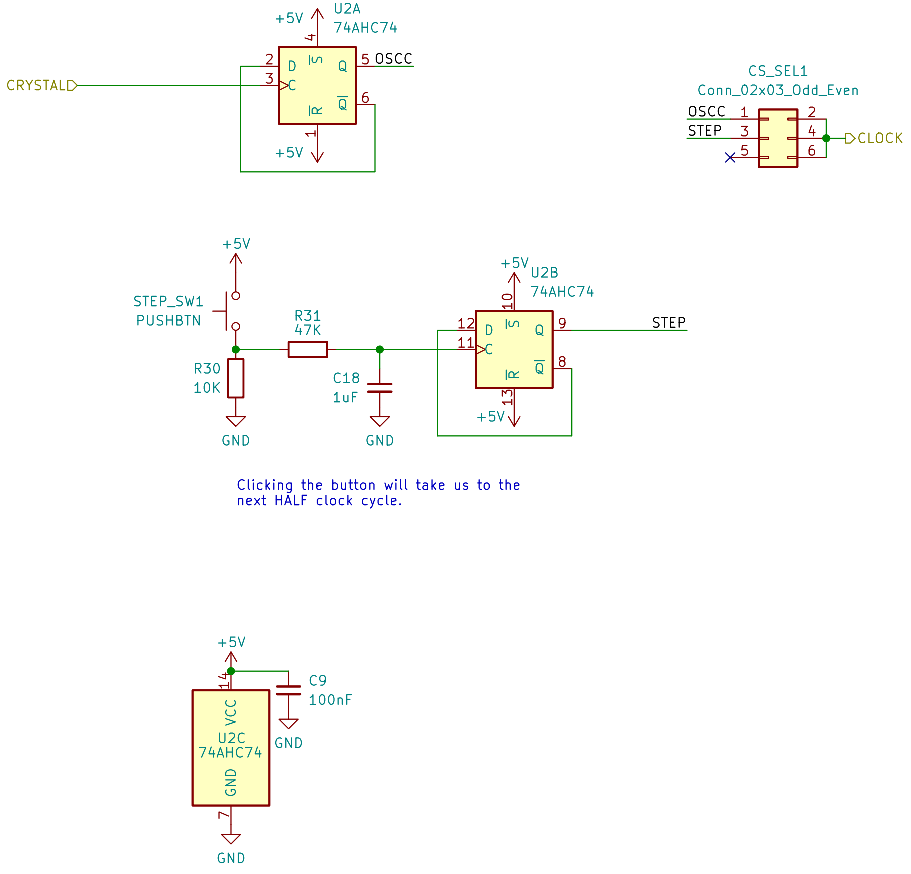
Homebrew 8-Bit Computer
Open Source Hardware Project
https://github.com/zrthxn/8puter
zrthxn

Sheet: /Video/
File: video.kicad_sch

Title: 8puter

Size: A4 Date: **Rev: 1.0**
KiCad E.D.A. eeschema 6.0.4-1.fc35 Id: 5/7

Clock Source Select



Clicking the button will take us to the next HALF clock cycle.

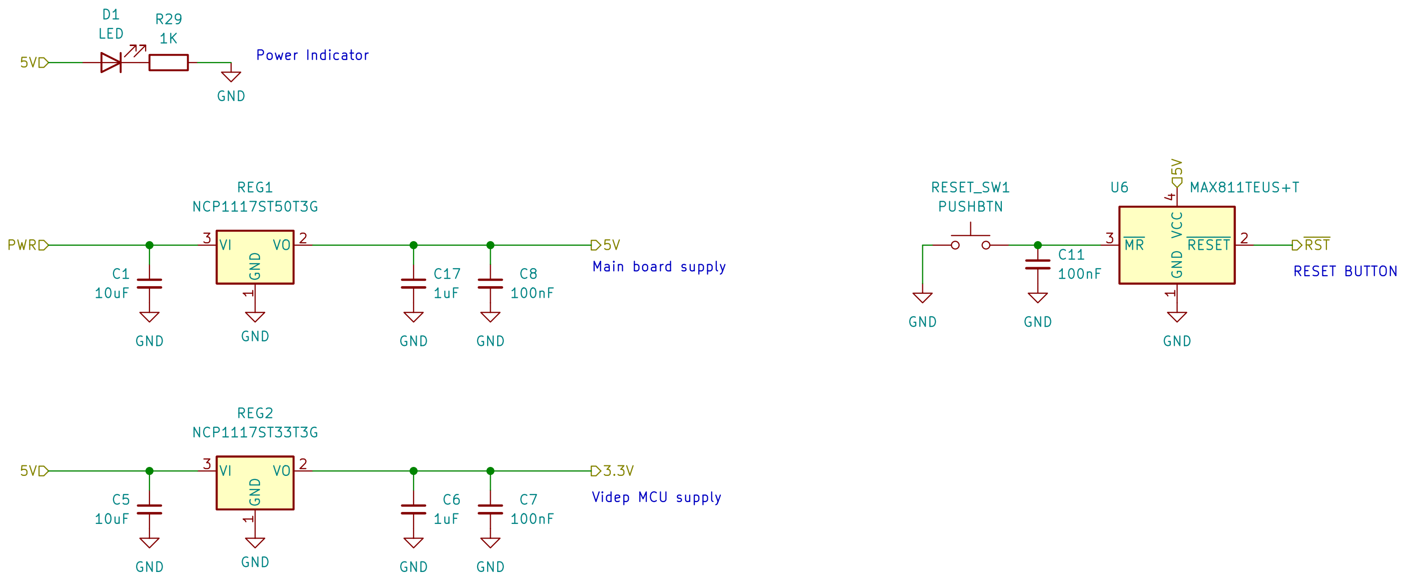
Homebrew 8-Bit Computer
Open Source Hardware Project
https://github.com/zrthxn/8puter
zrthxn

Sheet: /Clock/
File: clock.kicad_sch

Title: 8puter

Size: A4	Date:	**Rev: 1.0**
KiCad E.D.A. eeschema 6.0.4-1.fc35		Id: 6/7

Power Delivery



Homebrew 8-Bit Computer
Open Source Hardware Project
https://github.com/zrthxn/8puter
zrthxn

Sheet: /Power/
File: power.kicad_sch

Title: 8puter

Size: A4 Date: **Rev: 1.0**
KiCad E.D.A. eeschema 6.0.4-1.fc35 Id: 7/7