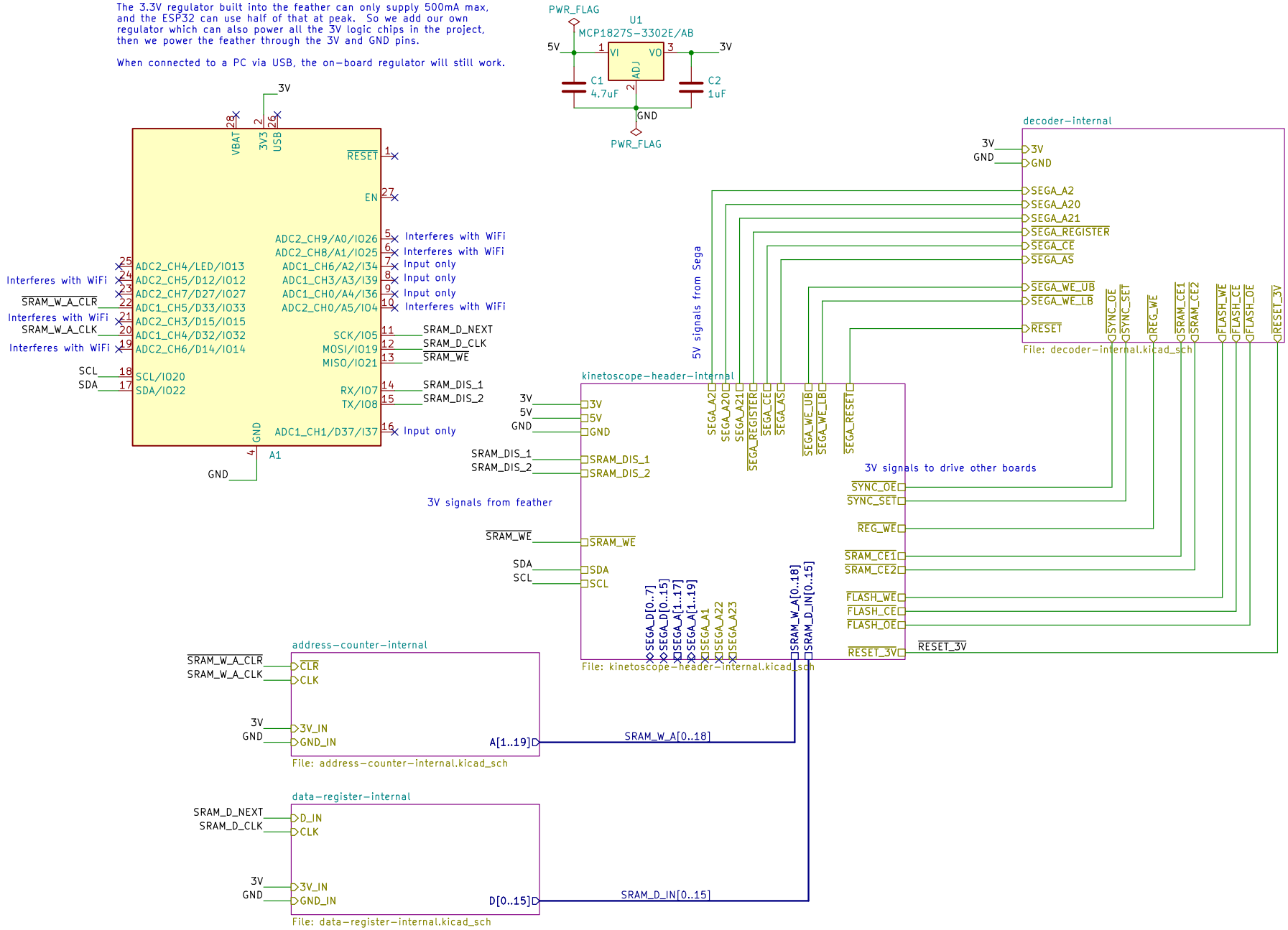
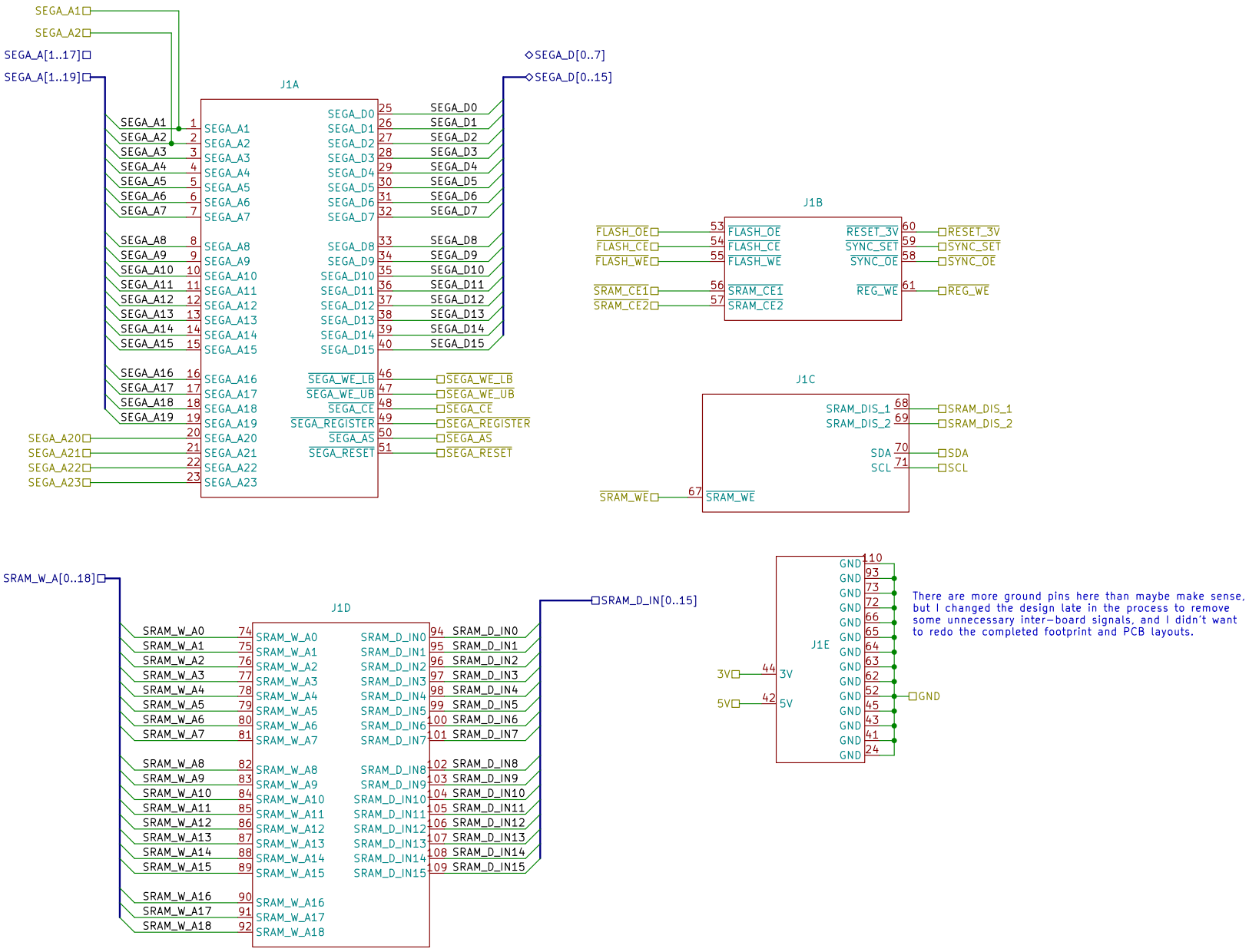
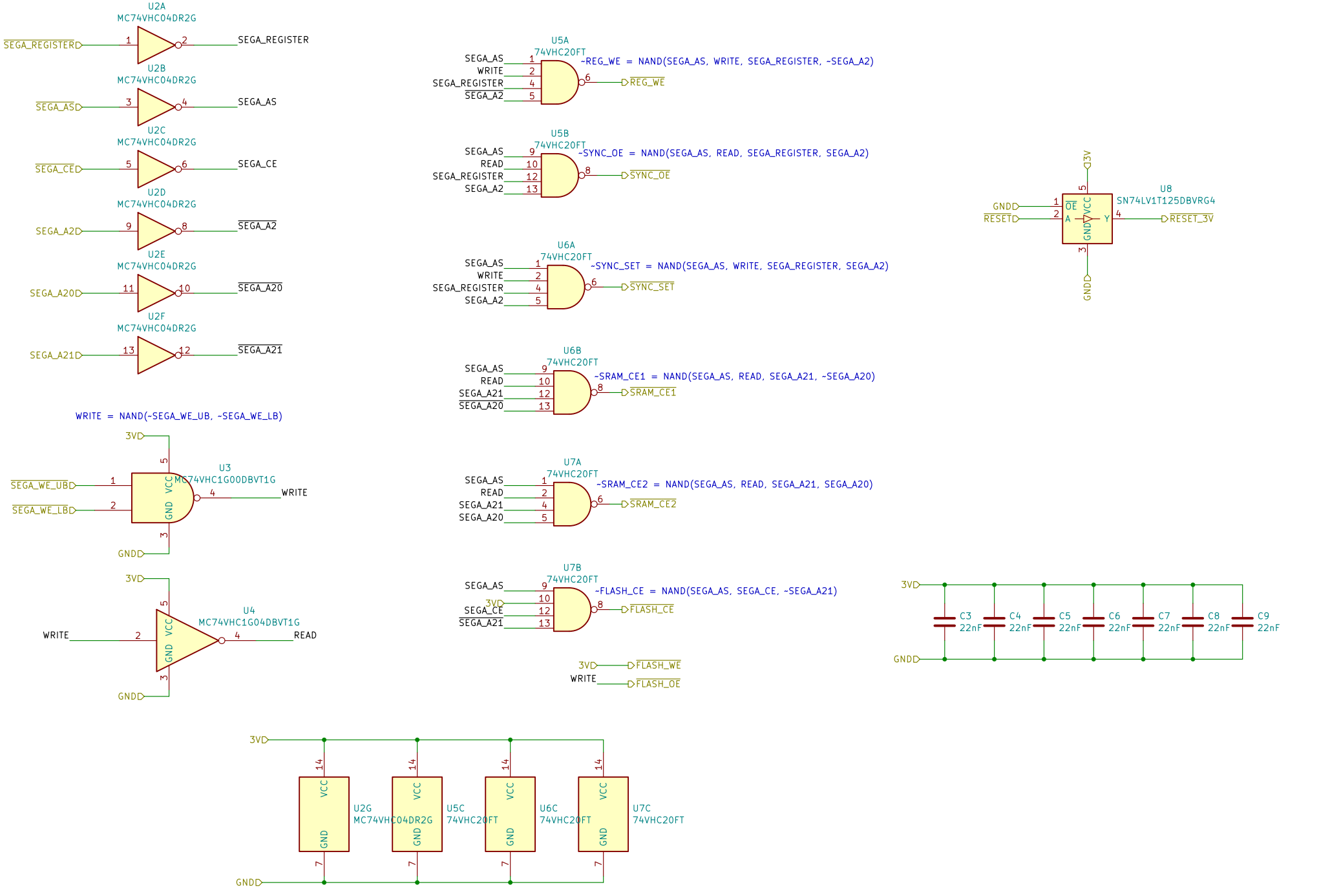
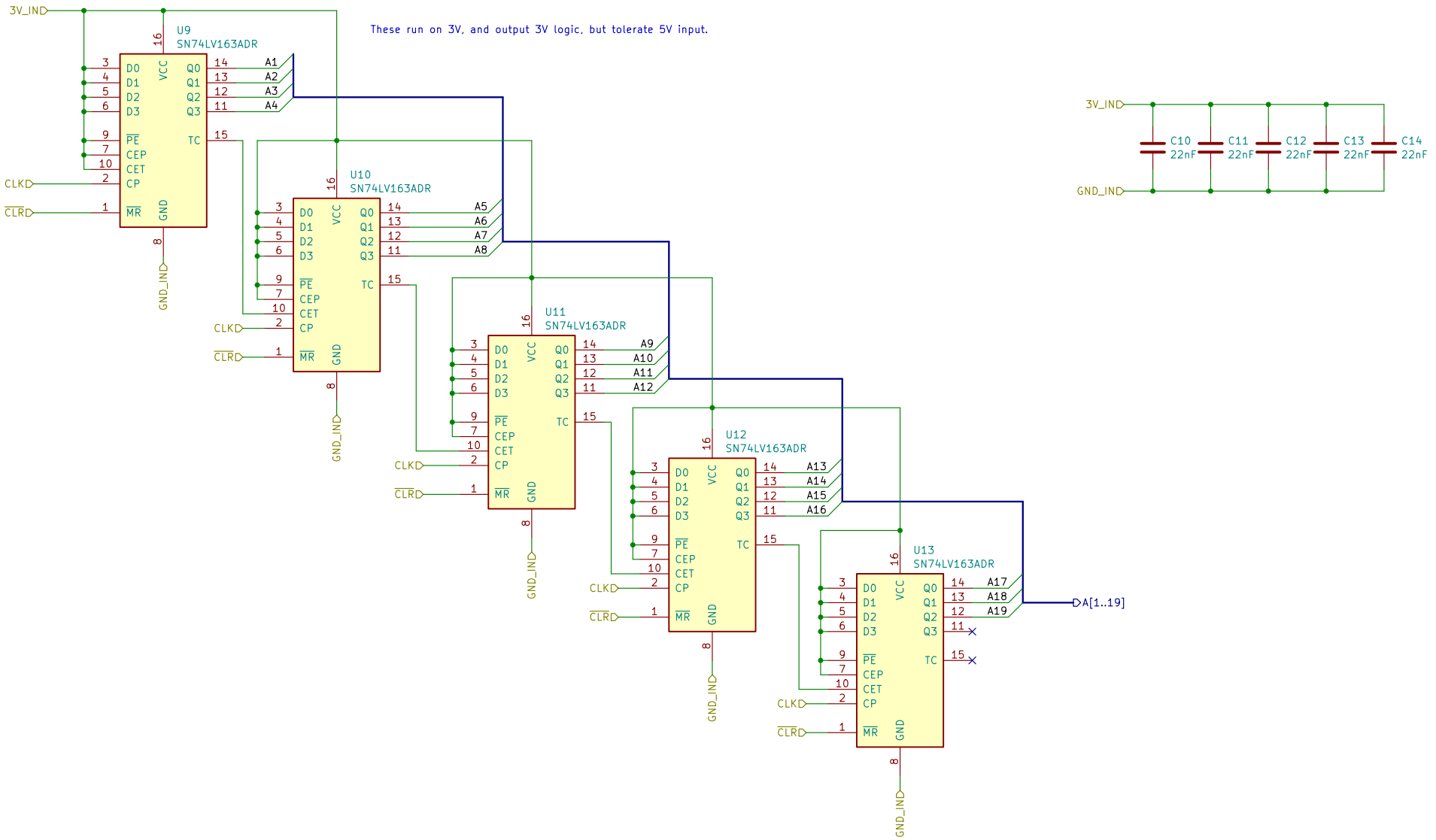
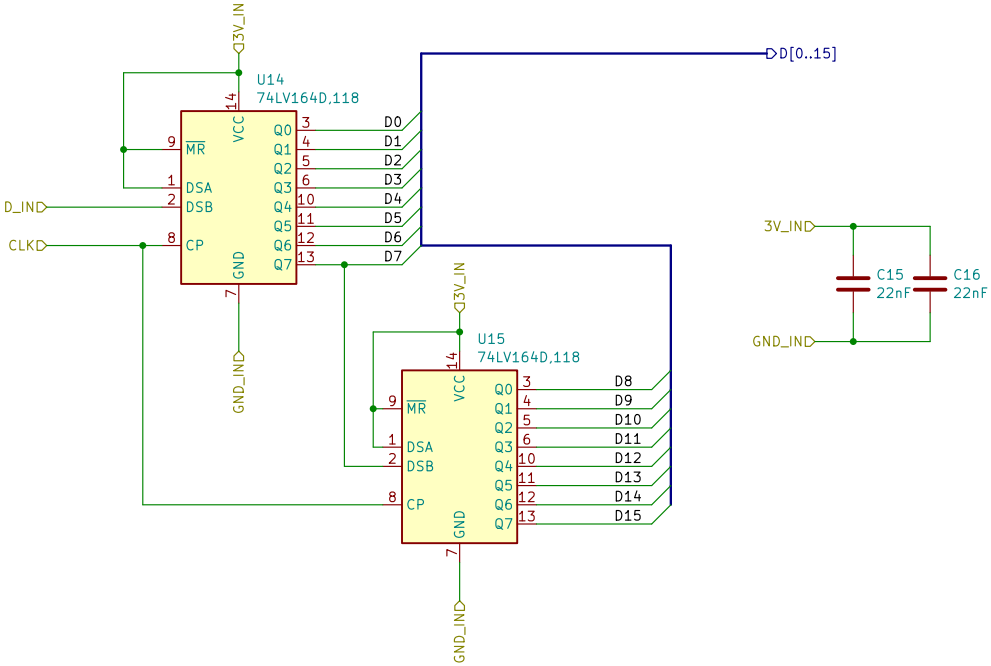

The 3.3V regulator built into the feather can only supply 500mA max, and the ESP32 can use half of that at peak. So we add our own regulator which can also power all the 3V logic chips in the project, then we power the feather through the 3V and GND pins.
When connected to a PC via USB, the on-board regulator will still work.
PWR_FLAG
U1
MCP1827S-3302E/AB
5V
VI
VO
ADJ
3V
C1
4.7uF
C2
1uF
GND
PWR_FLAG
3V
VBAT
3V3
USB
RESET
EN
ADC2_CH9/A0/IO26
ADC2_CH8/A1/IO25
ADC1_CH6/A2/I34
ADC1_CH3/A3/I39
ADC1_CH0/A4/I36
ADC2_CH0/A5/IO4
Interferes with WiFi
Input only
ADC2_CH4/LED/IO13
ADC2_CH5/D12/IO12
ADC2_CH7/D27/IO27
ADC1_CH5/D33/IO33
ADC2_CH3/D15/IO15
ADC1_CH4/D32/IO32
ADC2_CH6/D14/IO14
SRAM_W_A_CLR
SRAM_W_A_CLK
SCK/IO5
MOSI/IO19
MISO/IO21
SRAM_D_NEXT
SRAM_D_CLK
SRAM_WE
SCL/IO20
SDA/IO22
RX/IO7
TX/IO8
SRAM_DIS_1
SRAM_DIS_2
ADC1_CH1/D37/I37
A1
GND
decoder-internal
SEGA_A2
SEGA_A20
SEGA_A21
SEGA_REGISTER
SEGA_CE
SEGA_AS
SEGA_WE_UB
SEGA_WE_LB
RESET
SYNC_OE
SYNC_SET
REG_WE
SRAM_CE1
SRAM_CE2
FLASH_WE
FLASH_CE
FLASH_OE
RESET_3V
File: decoder-internal.kicad_sch
5V signals from Sega
kinetoscope-header-internal
3V signals from feather
3V signals to drive other boards
SEGA_RESET
SEGA_D[0..7]
SEGA_D[0..15]
SEGA_A[1..17]
SEGA_A[1..19]
SEGA_A1
SEGA_A22
SEGA_A23
SRAM_W_A[0..18]
SRAM_D_IN[0..15]
File: kinetoscope-header-internal.kicad_sch
address-counter-internal
3V_IN
GND_IN
A[1..19]
SRAM_W_A[0..18]
File: address-counter-internal.kicad_sch
data-register-internal
D_IN
D[0..15]
SRAM_D_IN[0..15]
File: data-register-internal.kicad_sch


SEGA_A1
SEGA_A2
SEGA_A[1..17]
SEGA_A[1..19]
SEGA_D[0..7]
SEGA_D[0..15]
J1A
J1B
J1C
J1D
J1E
SRAM_W_A[0..18]
SRAM_D_IN[0..15]
There are more ground pins here than maybe make sense, but I changed the design late in the process to remove some unnecessary inter-board signals, and I didn't want to redo the completed footprint and PCB layouts.


U2A
MC74VHC04DR2G
SEGA_REGISTER
SEGA_REGISTER
U2B
MC74VHC04DR2G
SEGA_AS
SEGA_AS
U2C
MC74VHC04DR2G
SEGA_CE
SEGA_CE
U2D
MC74VHC04DR2G
SEGA_A2
SEGA_A2
U2E
MC74VHC04DR2G
SEGA_A20
SEGA_A20
U2F
MC74VHC04DR2G
SEGA_A21
SEGA_A21
U5A
74VHC20FT
SEGA_AS
WRITE
SEGA_REGISTER
SEGA_A2
~REG_WE = NAND(SEGA_AS, WRITE, SEGA_REGISTER, ~SEGA_A2)
REG_WE
U5B
74VHC20FT
SEGA_AS
READ
SEGA_REGISTER
SEGA_A2
~SYNC_OE = NAND(SEGA_AS, READ, SEGA_REGISTER, SEGA_A2)
SYNC_OE
U6A
74VHC20FT
SEGA_AS
WRITE
SEGA_REGISTER
SEGA_A2
~SYNC_SET = NAND(SEGA_AS, WRITE, SEGA_REGISTER, SEGA_A2)
SYNC_SET
U6B
74VHC20FT
SEGA_AS
READ
SEGA_A21
SEGA_A20
~SRAM_CE1 = NAND(SEGA_AS, READ, SEGA_A21, ~SEGA_A20)
SRAM_CE1
U7A
74VHC20FT
SEGA_AS
READ
SEGA_A21
SEGA_A20
~SRAM_CE2 = NAND(SEGA_AS, READ, SEGA_A21, SEGA_A20)
SRAM_CE2
U7B
74VHC20FT
SEGA_AS
3V
SEGA_CE
SEGA_A21
~FLASH_CE = NAND(SEGA_AS, SEGA_CE, ~SEGA_A21)
FLASH_CE
3V
FLASH_WE
WRITE
FLASH_OE
U8
SN74LV1T125DBVRG4
GNDD
RESET
3V
RESET_3V
WRITE = NAND(~SEGA_WE_UB, ~SEGA_WE_LB)
U3
MC74VHC1G00DBVT1G
SEGA_WE_UB
SEGA_WE_LB
WRITE
3V
GNDD
U4
MC74VHC1G04DBVT1G
WRITE
READ
3V
GNDD
C3 22nF
C4 22nF
C5 22nF
C6 22nF
C7 22nF
C8 22nF
C9 22nF
3V
GNDD
U2G
MC74VHC04DR2G
U5C
74VHC20FT
U6C
74VHC20FT
U7C
74VHC20FT
3V
GNDD


These run on 3V, and output 3V logic, but tolerate 5V input.
3V_IN
GND_IN
U9
SN74LV163ADR
U10
SN74LV163ADR
U11
SN74LV163ADR
U12
SN74LV163ADR
U13
SN74LV163ADR
D0
D1
D2
D3
VCC
Q0
Q1
Q2
Q3
PE
CEP
CET
CP
TC
MR
GND
CLK
CLR
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A[1..19]
C10
22nF
C11
22nF
C12
22nF
C13
22nF
C14
22nF

There are not enough GPIO pins on the ESP32 V2 Feather
to write to SRAM in parallel, so we fill this register serially.