

Konrad Beckmann

Sheet: /

File: picocart64\_v2.kicad\_sch

Title: PicoCart64

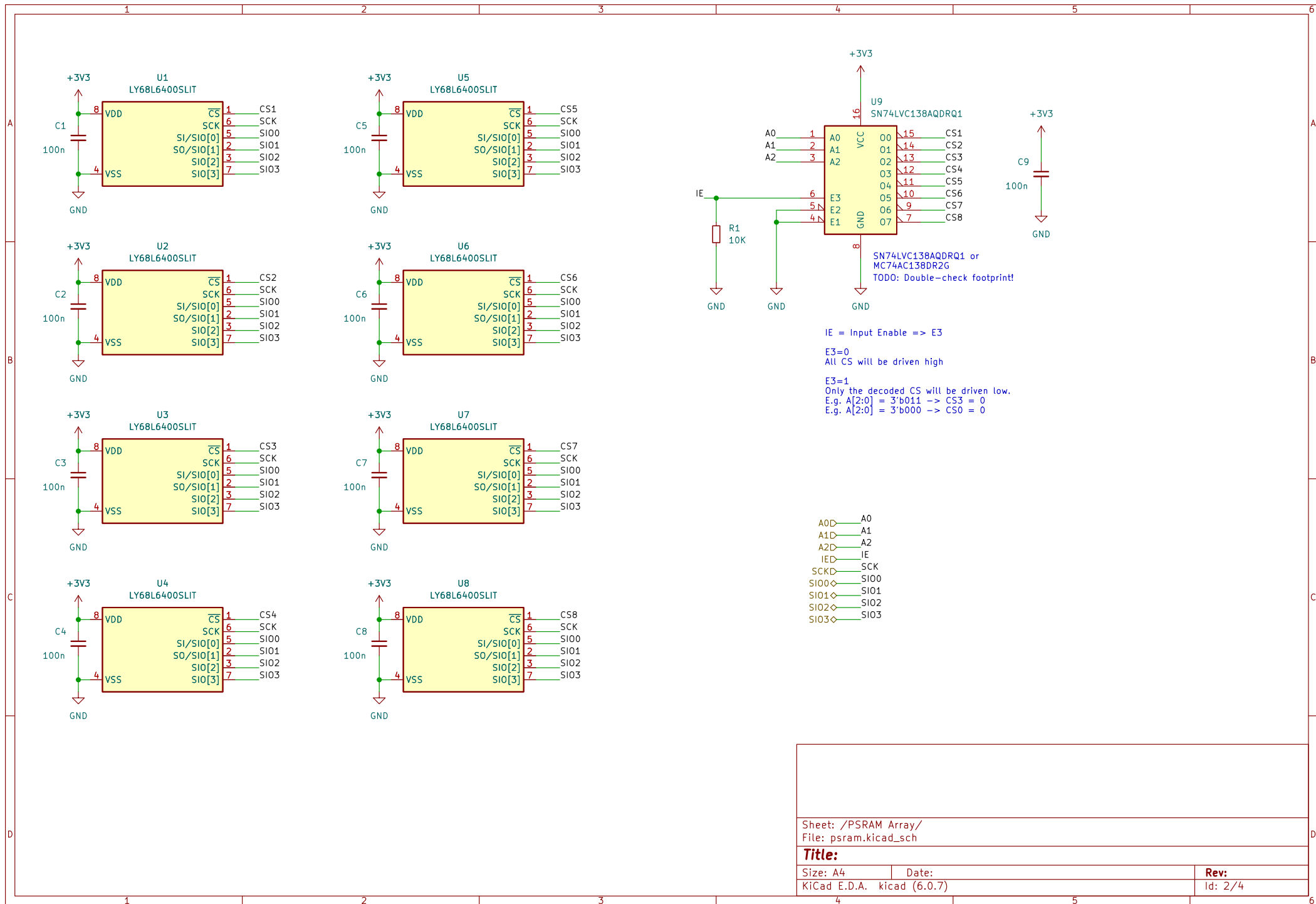
Size: A4

Date: 2022-07-02

KiCad E.D.A. kicad (6.0.7)

Rev: V2.0

Id: 1/4



Sheet: /PSRAM Array/  
File: psram.kicad\_sch

**Title:**

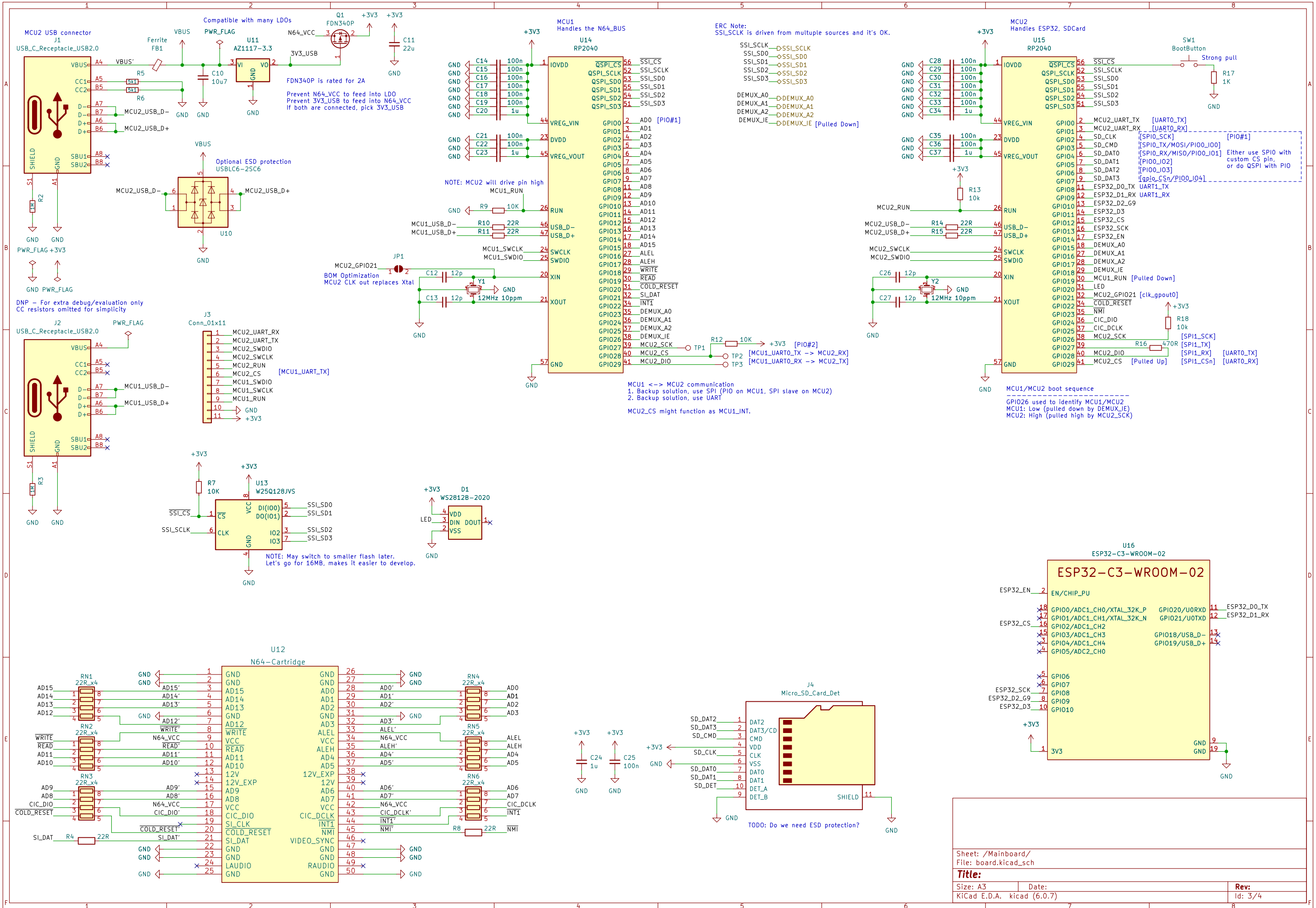
Size: A4

Date:

Rev:

KiCad E.D.A. kicad (6.0.7)

Id: 2/4



MCU1, MCU2 and all PSRAMs are connected to SSI. Both can read/write.  
MCU1 and MCU2 are connected to a 3-bit, 8 output demux + IE pin.  
Demux is connected to each PSRAM's CS pin.

MCU2 holds MCU1 in reset, fills PSRAM with data.  
MCU1 handles N64 bus requests, reads data directly from PSRAM.

MCU1 handles bus request, read/store in internal SRAM.  
Stores to external flash directly, and/or to SDCard via MCU2.

MCU1 handles this on the core that doesn't handle the N64\_PI bus.  
Stores to external flash directly, and/or to SDCard via MCU2.

MCU2 handles this on a dedicated core.

MCU2\_CS should be configured as input on MCU2, and Open Drain output on MCU1.  
MCU1 acts as an SPI Master with PIO.  
Combining miso/mosi to one pin, we save 1 pin.

- \* Small Flash on shared SSI bus

Bootloader firmware for both MCU1 and MCU2 is stored on this

MCU2 is "main" and will boot first (MCU1\_RUN has external pulldown). After boot, MCU2 realizes it's MCU2 by checking MCU1\_RUN pin (will be low). MCU2 may download application firmware from SDCard into RAM and execute. MCU2 disables output on SSLCS and SSLCLK. MCU2 drives MCU1\_RUN high, letting MCU1 boot.

After loading bootloader from flash, detects it's MCU1.  
! The GPIO chosen for MCU2's [MCU1\_RUN GPIO] must be connected to an  
idle-high signal on MCU1, e.g. MCU2's MCU2\_CS? !  
MCU1 optionally requests application from MCU2 over DIO-SPI.

Firmware can be updated from MCU2 with a recovery image.

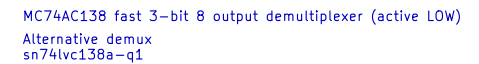
MCU2 can update firmware by pulling GP9 low, toggling EN, then perform update over UART.  
 WiFi module enables loading ROMs from network to SDCard.  
 WiFi module enables saving/loading SRAM/EEPROM saves to network.  
 Potential multiplayer stuff.  
 MCU1 can access it via MCU2.  
 MCU2 uses PIO QSPI controller.  
 QSPI slave interface can be implemented and assigned on any GPIO.

```
MCU1:
- Core 0:
---- N64_BUS [PIO #1]
---- N64_INT control
---- N64_CR interrupt
---- PSRAM synchronous access
- Core 1:
---- Asynchronous tasks
---- MCU2 SPI [PIO #2]
---- N64_S_DAT communication
```

```

- Core 0:
---- PSRAM
---- SDCard [PIO #1]
---- ESP32 QSPI [PIO #2]
---- LED control
---- N64_NMI control
---- SPI slave (MCU1 is master)
- Core 1:
---- CIC
---- N64_CR interrupt

```

PSRAM alternative to ly68l6400slit:  
APS6404L-3SQR-SN

RP2040 internal SRAM = 264 kB

Sheet: /Overview/  
File: overview.kicad\_sch

**Title:**

Size: A4	Date:
KiCad E.D.A. kicad (6.0.7)	

Rev:  
Id: 4/4