--- PSRAM Architecture ---
Pico1, Pico2 and all PSRAMs are connected to SSI. Both can read/write.
Pico1 and Pico2 are connected to a 3-bit, 8 output demux + OE pin.
Demux is connected to each PSRAM's CS pin.

--- N64 ROM ---
Pico2 holds Pico1 in reset, fills PSRAM with data.
Pico1 handles N64 bus requests, reads data directly from PSRAM.

--- N64 SRAM ---
Pico1 handles bus request, read/store in internal SRAM.
Stores to external flash directly, and/or to SDCard via Pico2.

--- N64 Serial Interface (EEPROM/RTC) ---
Pico1 handles this on the core that doesn't handle the N64_PI bus.
Stores to external flash directly, and/or to SDCard via Pico2.

--- Pico1 -> Pico2 communication ---
Pico2_CS should be configured as Open Drain [with external pullup?].
if SSI communication works, use this + Pico2_CS.
otherwise, Pico1 acts as an SPI Master with PIO with single miso/mosi pin, saving 1 pin.

--- Firmware delivery ---
* Small Flash on shared SSI bus
Bootloader firmware for both Pico1 and Pico2 is stored on this

* Pico2:
Pico2 is "main" and will boot first (Pico1_RUN has external pulldown).
After boot, Pico2 realizes it's Pico2 by checking Pico1_RUN pin (will be low)
Pico2 may download application firmware from SDCard into RAM and executes.
Pico2 Disables SSI interface, disabling output on SSI_CS and all the other pins.
Pico2 drives Pico1_RUN high, letting Pico1 boot.

* Pico1:
After loading bootloader from flash, detects it's Pico1
  ! The GPIO chosen for Pico2's [Pico1_RUN GPIO] must be connected to a idle-high signal on Pico1,
    e.g. Pico2's Pico2_CS? !
Pico1 optionally requests application from Pico2 over DIO-SPI.

* WiFi module:
Firmware can be updated from Pico2 with a recovery image.

--- WiFi module role ---
Pico2 can update firmware by aserting GP0, EN, then perform update over UART.
WiFi module can load ROMs from network to SDCard
WiFi module can save SRAM/EEPROM saves to network
Potential multiplayer stuff.
Pico1 can access it via Pico2.
Pico2 uses PIO QSPI controller. Overkill? Can figure out later.
Might use SSI bus instead with solder jumpers. Drawback - Pico1 can't serve N64_PI requests during access.

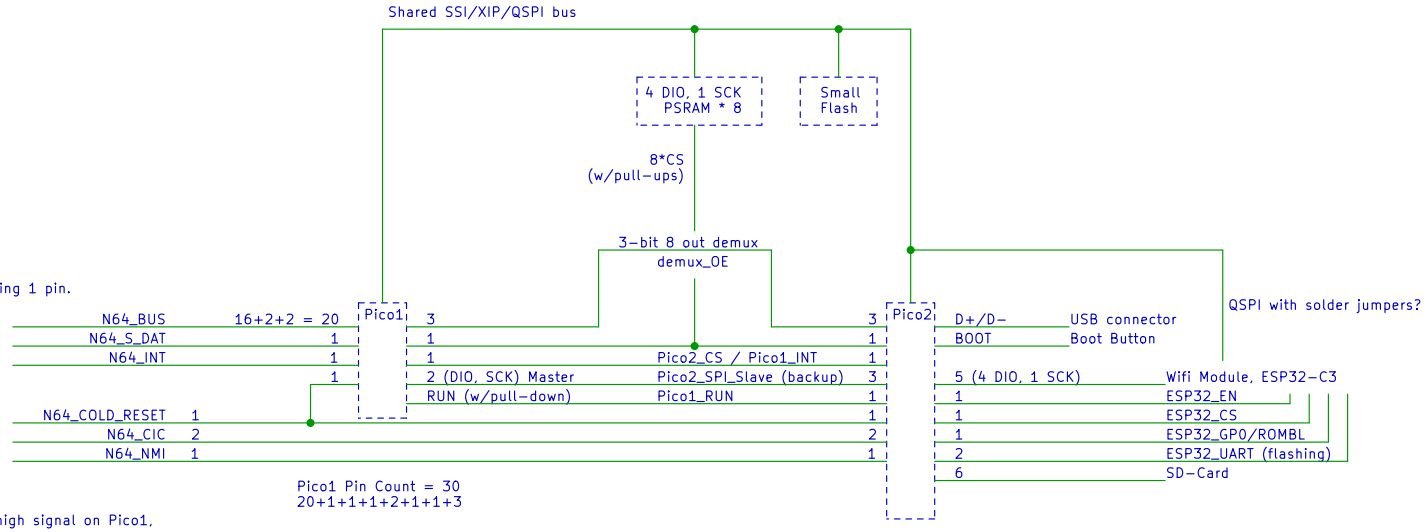
--------- Open questions

USB Debugging - Doable on Pico2.
Use a separate core for CIC, so interrupts don't interfere.

WiFi access while game is running:
Should be ok via the DIO-SPI bus to Pico2!

Debug/Print via WiFi
Doable via the DIO-SPI bus to Pico2!

Real debugging using INT?
Doable! Pin must be connected to Pico1 (for fastresponse)



Pico1 Pin Count = 30
20+1+1+1+2+1+1+3

Pico2 Pin Count = 29
3+1+1+3+1+1+2+1+6+2+1+1+1+5

MC74AC138 fast 3-bit 8 output demultiplexer (active LOW)

Alternative demux
sn74lvc138a-q1

PSRAM alternative to ly68l6400slit:
APS6404L-3SQR-SN

Pico SRAM = 264 kB

PSRAM = 4 I/O + CLK
CE = 1 per chip

CE demux 3 bit -> 8 CE

Sheet: /Concept5/
File: Concept5.kicad_sch

Title:

Size: A4	Date:	Rev:
KiCad E.D.A. kicad (6.0.6)		Id: 6/6