High Bandwidth Low Latency Communication with SPI Devices Controlled by PIC32

Bridging High-Speed Peripherals with External SRAM Buffers

Arturo Salinas and Alexander Xhemo

September 15, 2025

Outline

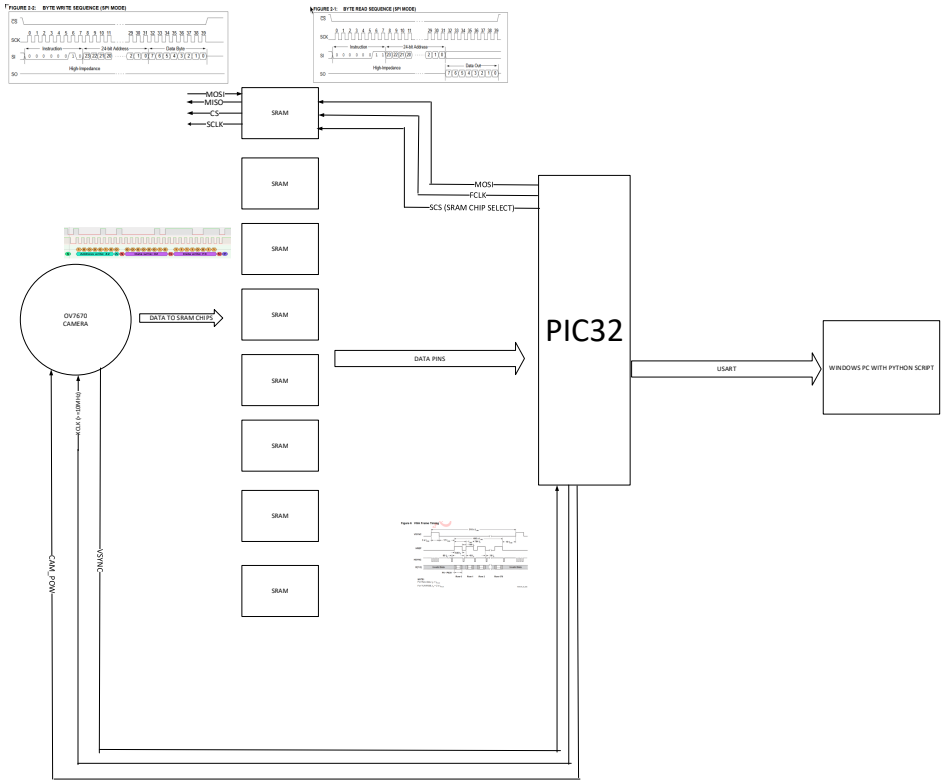
Project Overview

- **Problem:** OV7670 camera outputs 8-bit parallel pixel data at **10–48 MHz**, while PIC32 runs at **40 MHz** and cannot sample/store 8 pins every few cycles without overrun.
- **Idea:** Use **8× 23LC1024 SPI SRAM** devices as a high-speed **parallel buffer** to absorb camera bursts, then **read out serially** via PIC32 at MCU pace.
- **Pipeline:** Target (camera) → SRAM (parallel write) → PIC32 (SPI read) → PC (UART → Python viewer).
- **Outcome:** Aligned frames captured; verified Y (luma) ordering and data integrity; color conversion needs further tuning.

Project Objectives

1. Capture a full CIF frame from OV7670 without MCU-cycle loss.
2. Buffer high-rate parallel data directly into SRAM with minimal CPU intervention.
3. Read buffered data back over SPI and stream to PC.
4. Demonstrate low-latency control and high-bandwidth ingestion.

System Diagram

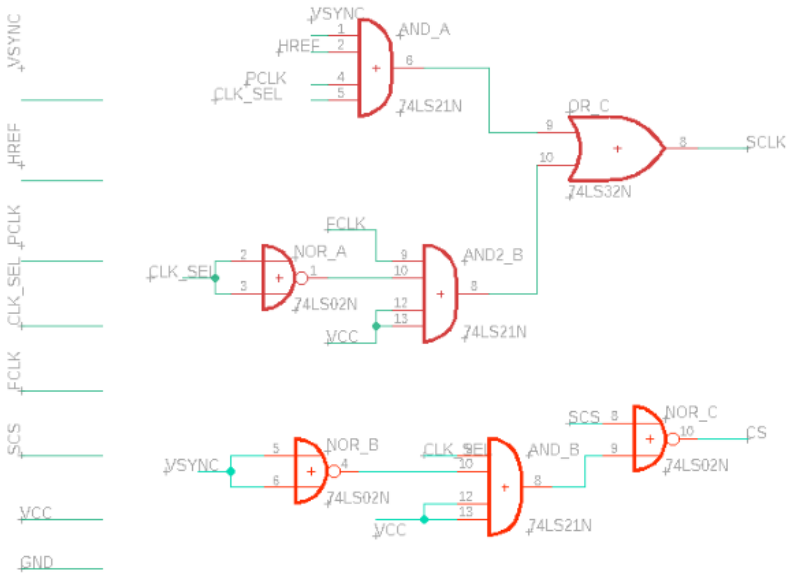


Hardware Components Used

- **PIC32** MCU at 40 MHz (SPI, UART, Output Compare for XCLK, GPIO).
- **OV7670 Camera**: 8-bit parallel data, SCCB configuration.
- **8 × 23LC1024 SPI SRAM**: parallelized as byte-wide buffer.
- **Glue Logic**: CS gating, clock selection (FCLK vs. PCLK), diode OR for MOSI lines, pull-downs.
- **Test Equipment**: Function generator (10 MHz XCLK), oscilloscope/logic analyzer.

Software Features

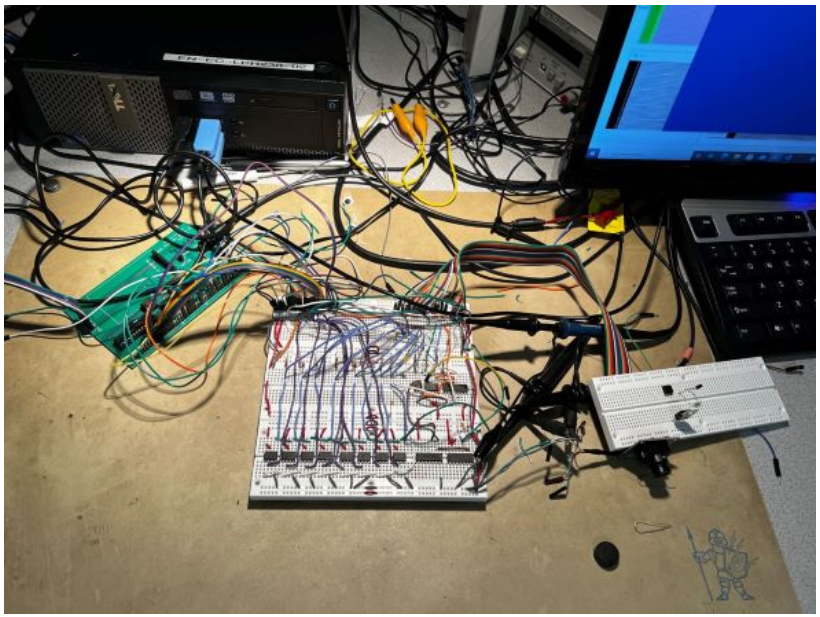
- **Initialization**: Configure pins; generate XCLK (10 MHz) using Output Compare; set SCCB (I^2C-like) for camera registers.
- **Phase-Split Transaction**:
  - *P1 (PIC → SRAM)*: Drop CS, send `WRITE` + 24-bit address via MOSI/FCLK.
  - *P2 (Camera → SRAM)*: Camera drives D[0:7]; SRAM clocks on PCLK when HREF=1; CS held by logic until VSYNC rises.



Software Features Cont.

- **Readout**: PIC32 issues `READ`; SPI-clocks bytes out of SRAM, forwards via UART to PC.
- **PC-side**: Python reconstructs image;

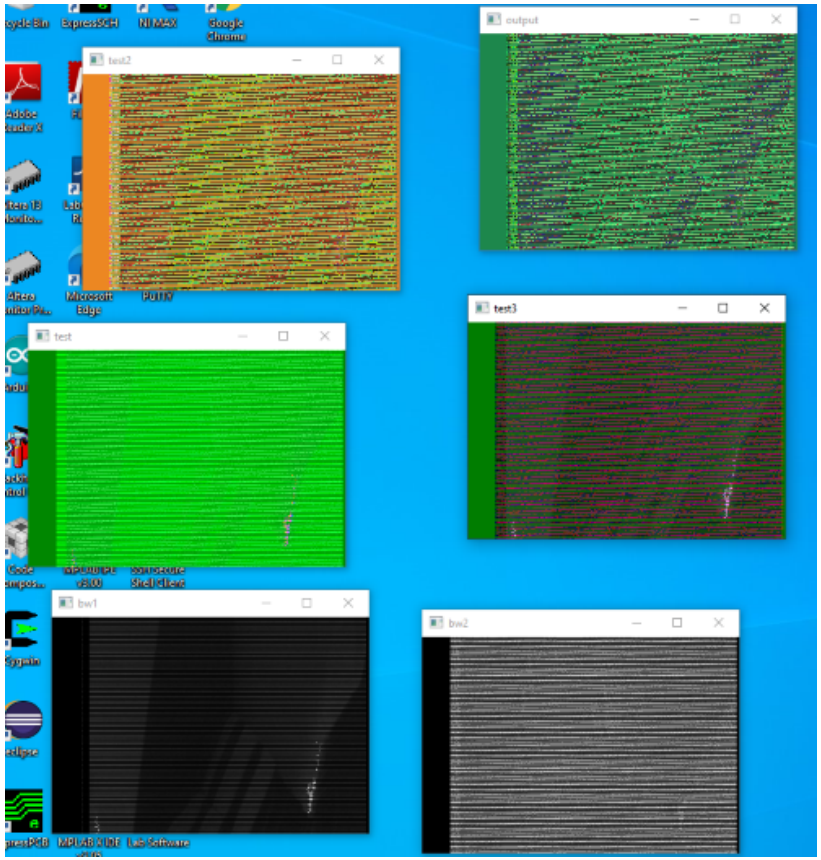
Microcontroller Peripherals



- **SPI**: Commanding SRAMs; sequential read/write.
- **UART**: Stream image bytes to host PC.
- **Output Compare (OC)**: Generate stable 10 MHz XCLK for camera.
- **GPIO**: VSYNC/HREF/PCLK sensing, SCS/CLK_SEL/FCLK control, PORTY data capture.
- **SCCB (I^2C-like)**: Camera register programming (e.g., `COM7` reset/config).

Results (Highlights)

- **Acquisition**: Received full frames; VSYNC/HREF/PCLK timing honored.
- **Integrity**: Pixel alignment correct; grayscale (Y) images render as expected.
- **Throughput**: Parallel SRAM ingestion sustains camera rate with minimal CPU load.
- **Color**: YUV→RGB conversion produced recognizable images; tuning needed for chroma interpretation.



Lessons Learned from the Project

- **Timing is king**: Edges (VSYNC/HREF/PCLK/SCK) and CS gating must be unambiguous. This just showed how the team learned about this specific interface.
- **Buffer-first architecture**: External SRAM decouples fast I/O from MCU constraints, many modern MCU's contain DRAM which can be utilized this way.
- **Tooling matters**: Logic analyzer + scope saved days of guesswork. In the field other timing tools can be used such as HDL Testbenches to verify timing constraints prior to hardware implementation.
- **Docs reality**: OV7670 documentation inconsistencies require empirical validation. This highlighted the teams inability to understand what the datasheet actually specifies.
- **Scalability**: Pattern applies to other wide, fast peripherals (ADCs, sensors), but can be improved.

Opinions

- **Reinventing the Wheel**: Other forms of buffers currently exist.
- **Difficulty with Synchronization**: Writing drivers for the timing could have been simplified using existing QSPI implementations
- **Reliance on Python**: The need to have a python script on the host windows PC greatly reduces the cool factor of this project.

Conclusion & Next Steps

- Demonstrated a practical **high-bandwidth, low-latency** bridge using **parallel SRAM + PIC32**.
- Verified frame capture correctness and robust readout path to PC.
- **Future work**: Better camera module; refine YUV→RGB; add wireless link; DMA-assisted readout.

Thank You

Questions?