Designing a Camera Module Driver Using Programmable I/O on Pi Pico RP2040

Author: Yibo Yang   Advisors: Hunter Adams, Bruce Land

Programmable I/O?

- Programmable I/O (PIO) is special feature on Raspberry Pi Pico (released in Jan. 2021), works **in parallel** with dual-core Cortex M0, providing a powerful potential for systems that highly requires parallelism
- PIO state machines are programmed by **assembly**, an independent instruction set which only has nine instructions
- Arducam designed a series of camera modules for Raspberry Pi Pico, including monochrome cameras and color cameras
- Real-time image processing can be developed, such as always-on-service for machine vision applications and IoT applications[1]

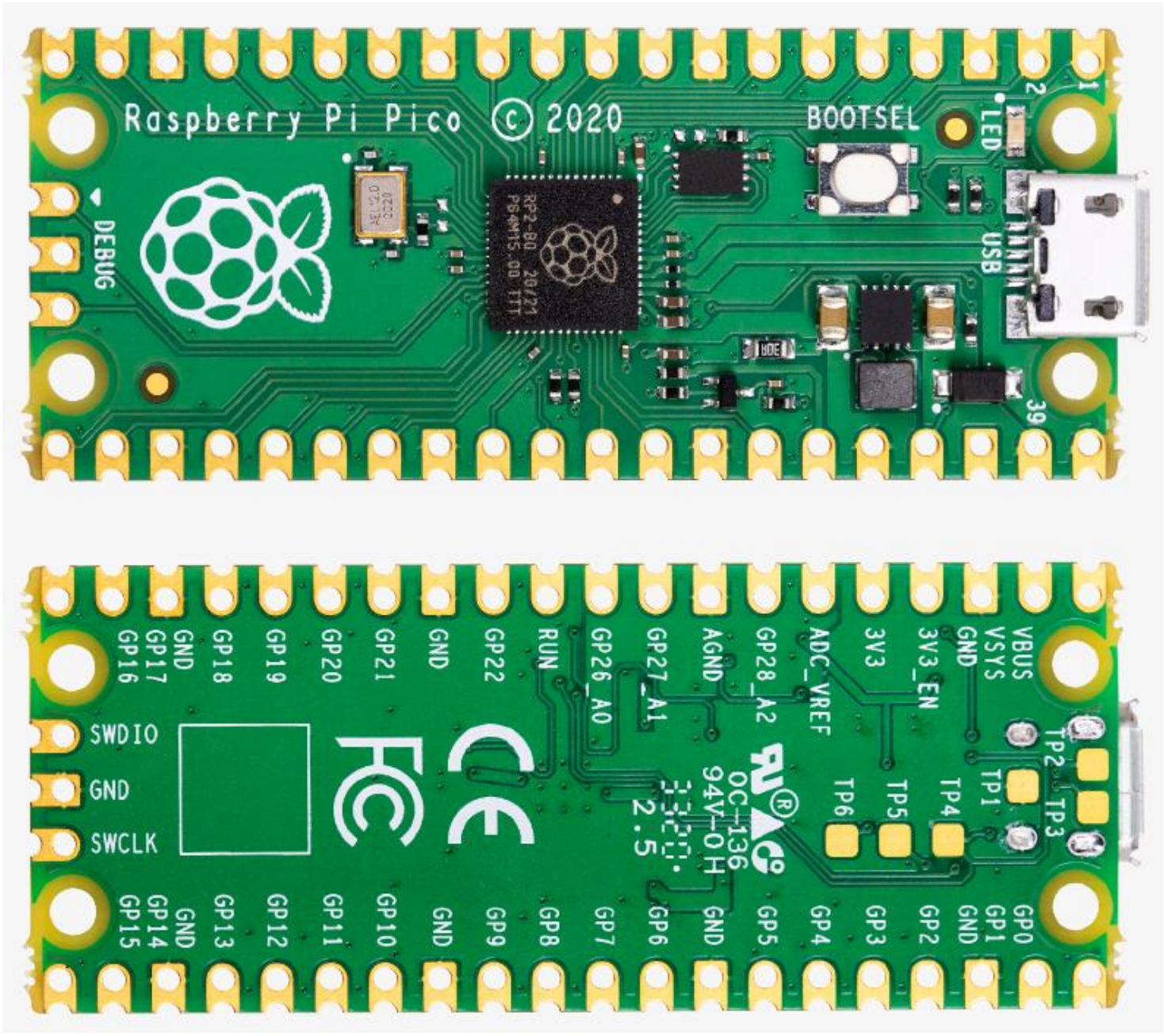


Fig.1 The Raspberry Pi Pico Rev3 Board

Fig.2 Arducam 2MP SPI Camera

Challenges

- Camera drivers usually require a **high-speed interface**, but MCUs don't have an adequate camera port for video signals—we can use SPI for this, which is much faster than I2C, UART, and I2S[2]
- RP2040's **264kB** on-chip SRAM is quite limited to hold an image—**DMA** could be exploited to stream data from video input to display
- Programmable I/O has a great potential for implementing complex peripherals, but PIO assembly is different from other popular instruction sets and provides various functions with it—luckily, some open-source simulators and debuggers for PIO state machine are available for RP2040 development

A Camera System without CPU!

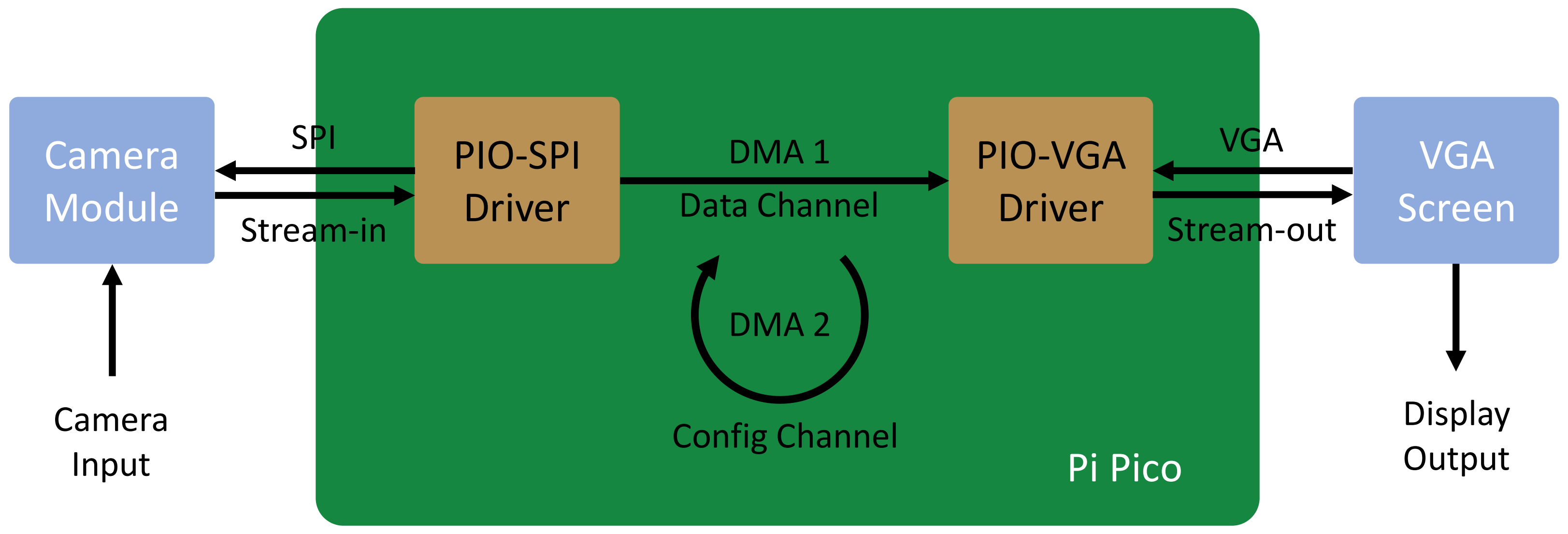


Fig.3 System Overview

Why PIO Assembly is so cool

- PIO assembly provides **9 16-bit instructions** for user development: JMP, WAIT, IN, OUT, PUSH, PULL, MOV, IRQ, SET
- RP2040 has **two PIO blocks**, each has **four PIO state machines** to execute instructions, which are read from a shared instruction memory that can hold up to **32 instructions** (one instruction memory for one PIO block)
- Various functions are supported: sideset, delay, program wrapping…

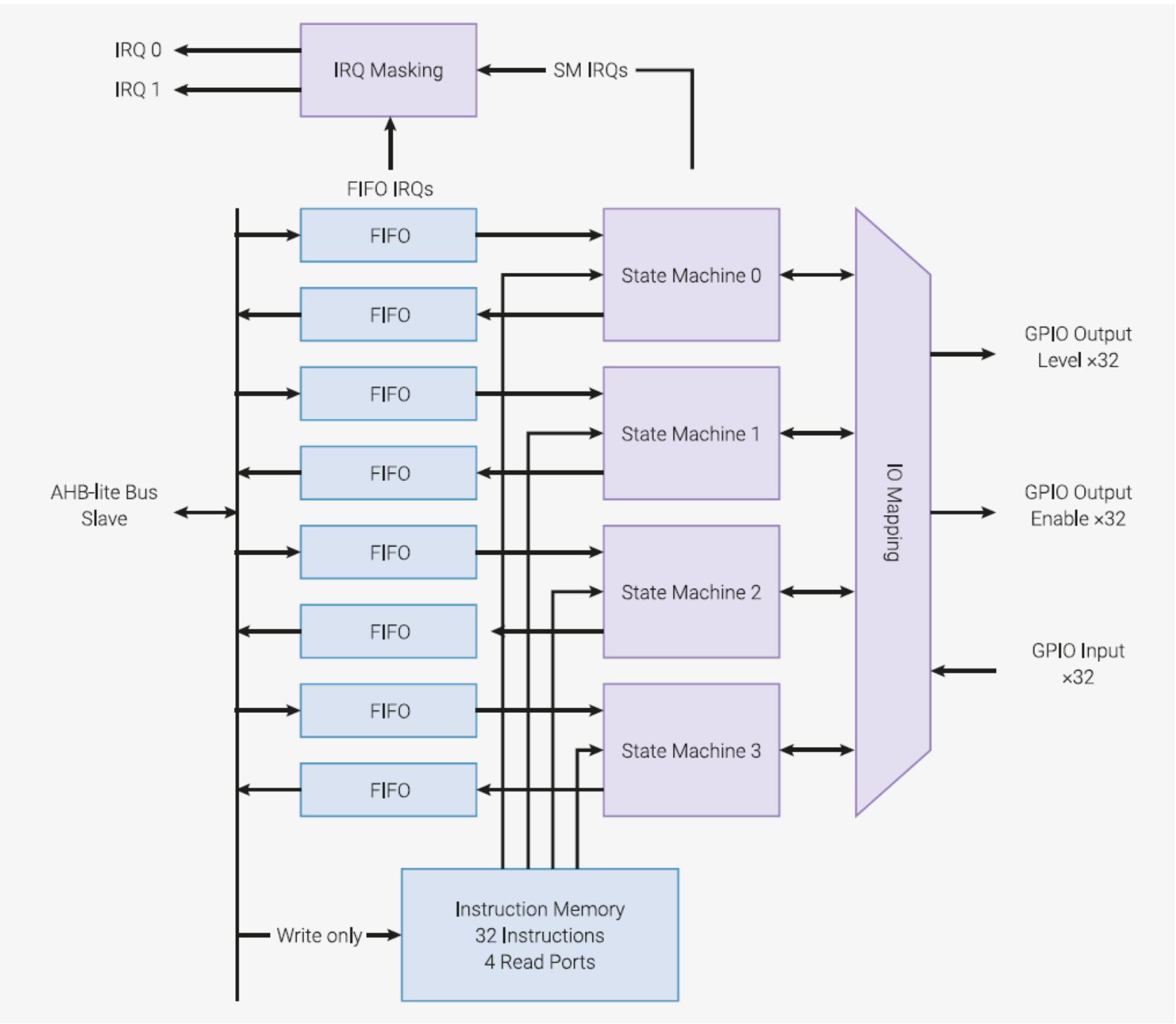


Fig.4 Programmable I/O Block

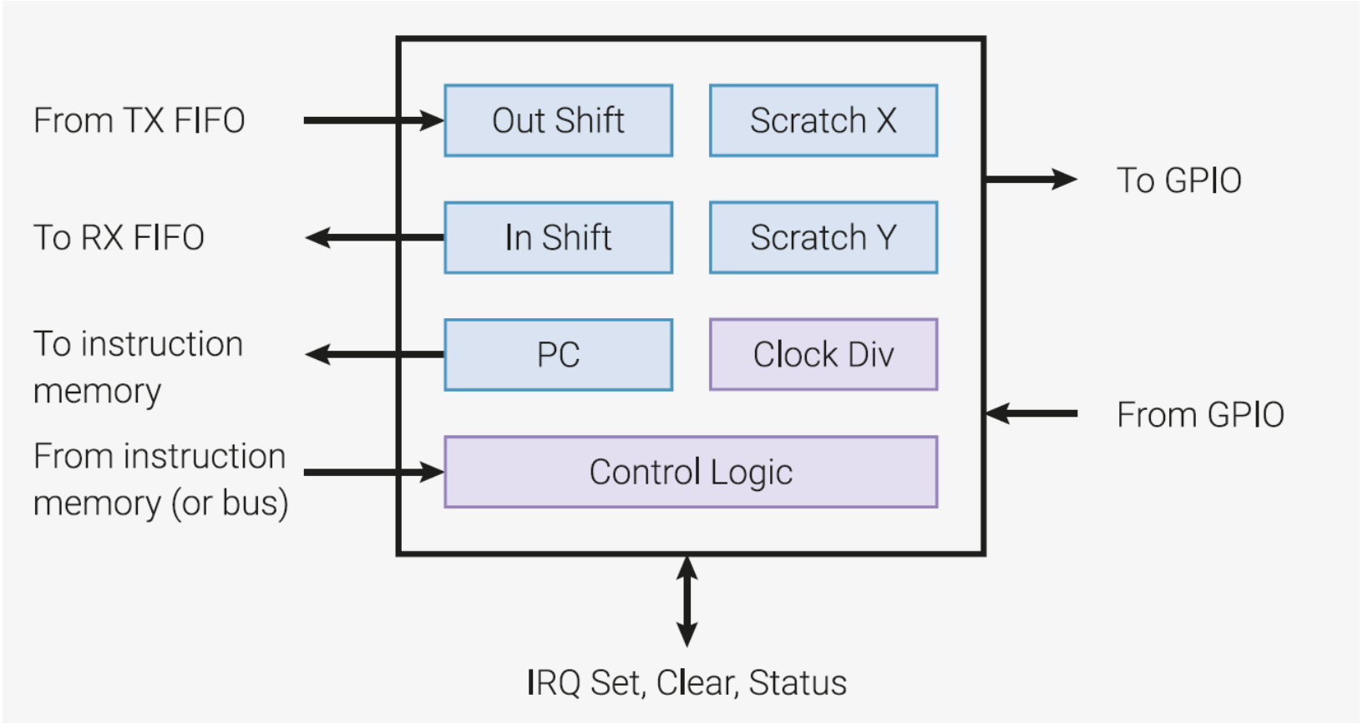
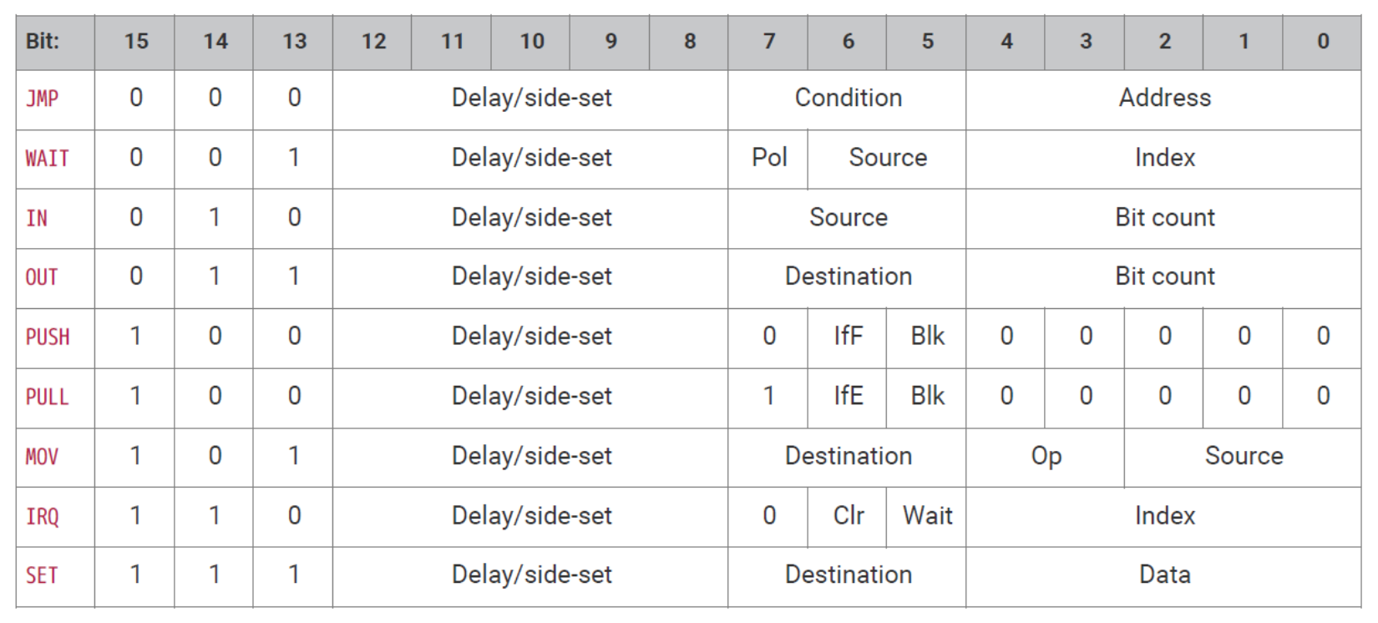


Fig.5 PIO State Machine



Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JMP	0	0	0	Delay/side-set					Condition			Address				
WAIT	0	0	1	Delay/side-set					Pol	Source		Index				
IN	0	1	0	Delay/side-set					Source			Bit count				
OUT	0	1	1	Delay/side-set					Destination			Bit count				
PUSH	1	0	0	Delay/side-set					0	IfF	Blk	0	0	0	0	0
PULL	1	0	0	Delay/side-set					1	IfE	Blk	0	0	0	0	0
MOV	1	0	1	Delay/side-set					Destination			Op		Source		
IRQ	1	1	0	Delay/side-set					0	Clr	Wait	Index				
SET	1	1	1	Delay/side-set					Destination			Data				

Fig.6 PIO Assembly Instruction Set

References

[1] https://www.arducam.com/raspberry-pi-pico-camera-modules/

[2] https://www.arducam.com/spi-arduino-camera/

[3] https://vanhunteradams.com/Pico/VGA/VGA.html

Current Work

- A 2.5MHz SPI interface is generated by PIO and tested with a 12-bit DAC module, generating an analog triangle wave with 32 levels
- Use **chained DMA** channels to free the CPU, the **data channel** is responsible for pumping a wave table to the PIO state machine to generate SPI signals, and the **configuration channel** is responsible for resetting the first channel to repeat the transaction
- A current VGA example[3] is available to be integrated with the SPI driver, which supports 8 colors and uses DMA and PIO state machine for data transfer

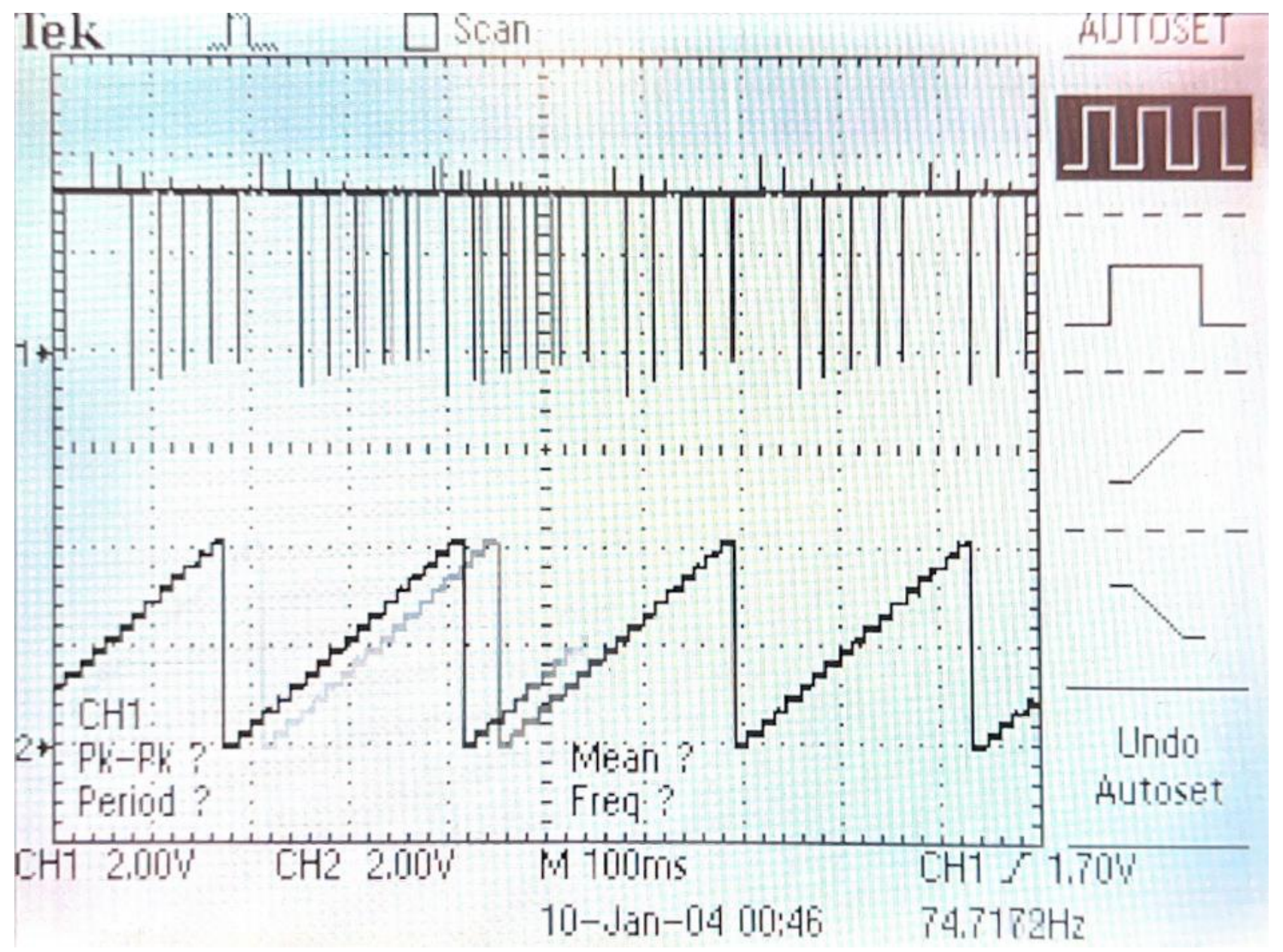


Fig.7 Triangle Wave Generated by MCP4822

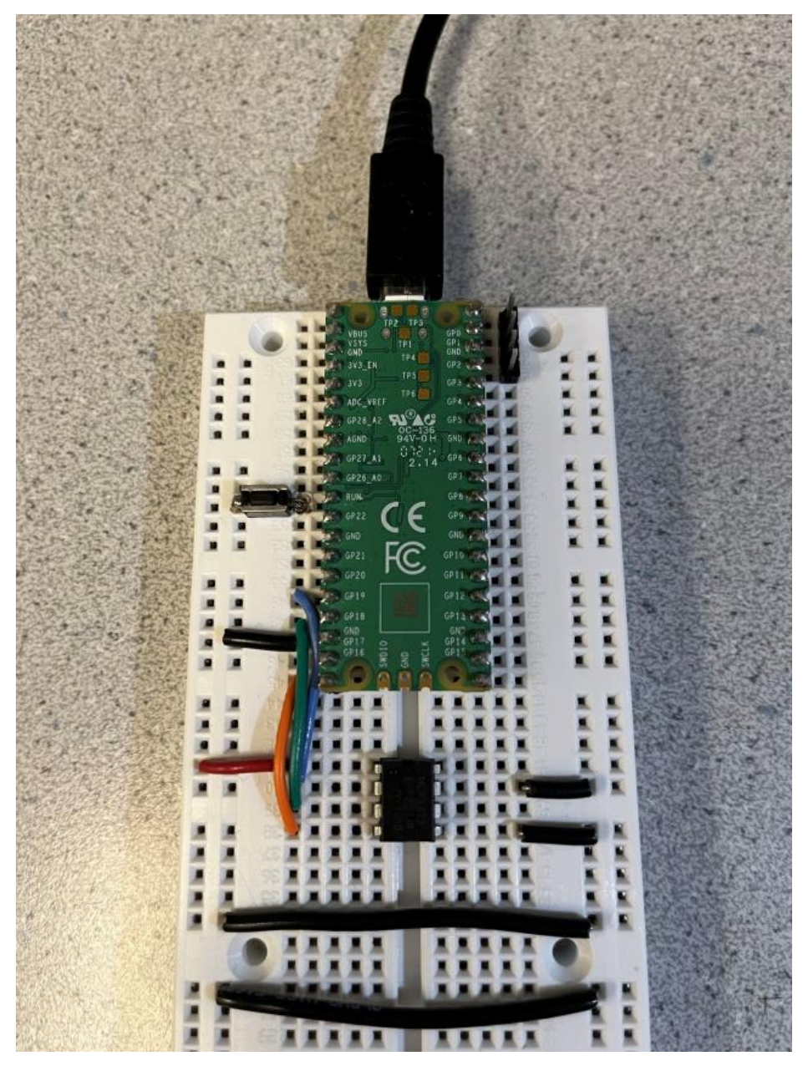


Fig.8 Whiteboard Layout

Future Work

- Use the current SPI interface to drive the Arducam 2MP SPI camera module, and then use DMA to stream the input video data to the VGA output for real-time display
- **Other high-speed camera interfaces** could be considered for implementation using programmable I/O
- Due to limited memory to hold the image, the **synchronization** between the VGA output and the video input is very important
- The dual-core system could also be utilized for some image processing algorithms

Acknowledgements

- Special thanks to my advisors, Dr. Van Hunter Adams and Prof. Bruce Robert Land for their help with this project.