Final Lab Project

Adam Shanta
CSE 121/L

December 15, 2020

1 Project Goals

The goal for the final project is to create a stand alone two channel oscilloscope with free-running, positive trigger, and negative trigger modes as well as scaling functionality using the Cypress PSoC-6 microcontroller. The oscilloscope controls will be modified through a terminal sending and receiving data connected to the USB-UART connection port. The analog input waveform data will be sampled by a multi-channeled ADC and sent through two separate DMAs to their dedicated ping-pong buffers each containing a depth of 2 x 256 16-bit values. This data will then be processed by the software to extract the signal frequency and apply the necessary x and y scaling onto a NewHaven 240 x 320 LCD Display. Additionally, two external 10KΩ potentiometers will be sampled to implement a scrolling vertical offset for each channel displayed on the screen.

2 Design

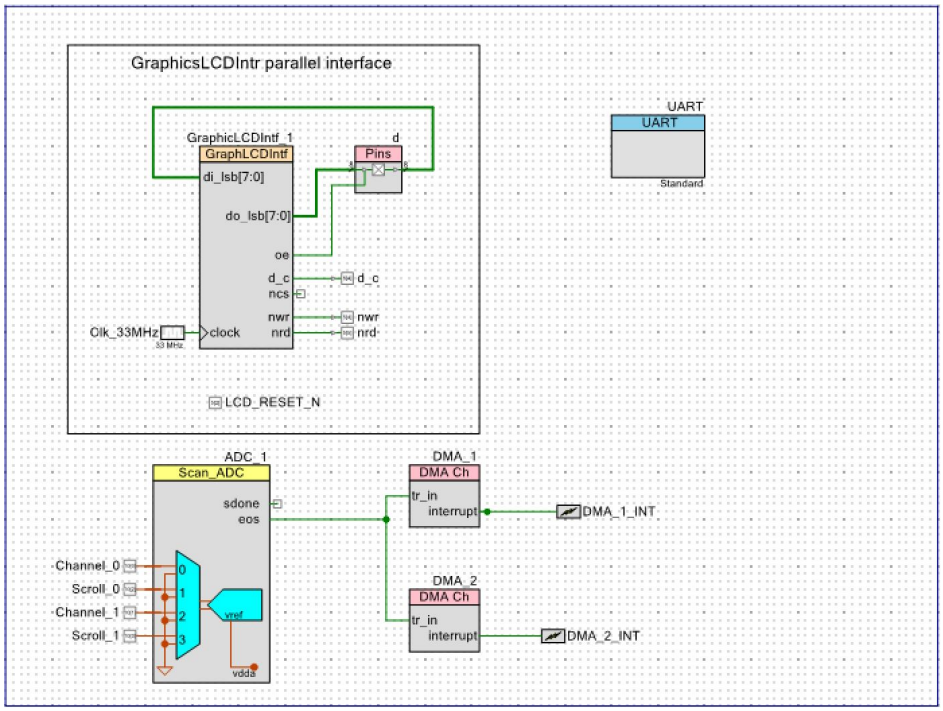

Figure 1: Top Design

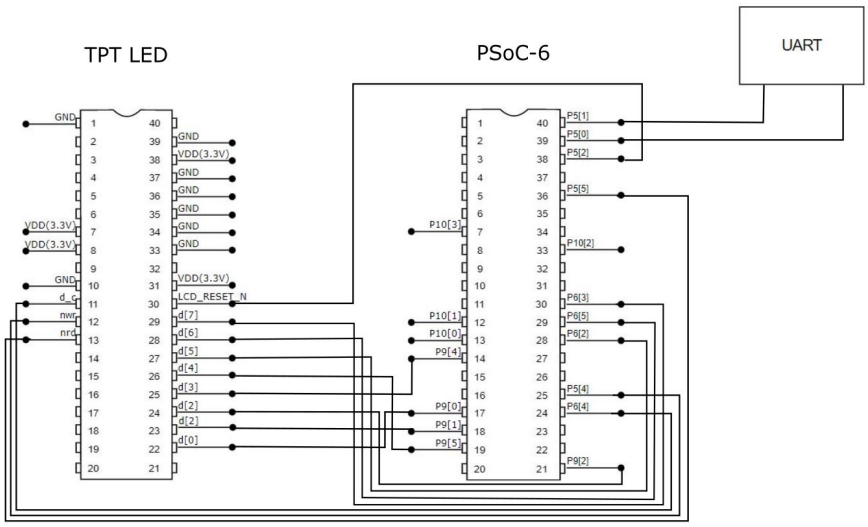

Figure 2: Pin Schematic

The design for the internal hardware allocated in the PSoC-6 was fairly straight forward. I connected a Graphics LCD Interface component to a bus of pins to manage communication between the PSoC-6 and the LCD display, instantiated a UART component for USB-UART communication, and connected two DMAs to the end of conversion signal of the 4-channel SAR ADC. Each DMA has an ISR interrupt component connected to its interrupt terminal which is triggered when one of the ping-pong buffers is full, so that the software knows when to toggle between each of the two.

In the pin schematic, you can see the LCD is wired to the PSoC-6 through pin ports P9 and P6, as well as the PSoC-6 RX and TX terminals connected to the UART through pins P5[0] and P5[1]. The channel 0 input waveform and its potentiometer input is connected to P10[0] and P10[2], and the channel 1 waveform and potentiometer inputs are connected to pins P10[1] and P10[3].

The code was structured to reduce lag time in the display refresh rate. I focused on having clean and real time data displaying on the screen at all times regardless of change in frequency and scaling. After the initialization of the components and the display, the infinite for loop drew updated and processed waveform data from both channels onto the display, then toggled between processing raw data from one of the two sets of DMA ping-pong buffers. The program checks which descriptor of the DMA is currently active, and takes the raw data from the opposing buffer. The raw data is run through a software implementation of a leaky integrator low pass filter using fixed point bit shifting arithmetic with a β of about 0.5. For each data point an edge detection function checks the slope of the waveform based on previous values, calculates its frequency value if the beginning of a cycle is recognized, and waits a specified amount of time based off of the sampling rate before processing the next data point. This is also where the negative or positive trigger value is used when the oscilloscope is not in free-running mode. When the 256 values in the buffer are processed and mapped to the 310 pixels on the display, the program erases the old waveform for that channel on the screen and assigns the newly processed values to its respective outgoing channel.

3 Testing

Testing was done as an iterative process since it is generally difficult and time consuming otherwise, especially when interfacing hardware and software components. I started with the LCD display, making sure the test code ran after connecting all the pins. I then connected the UART and the ADC to made sure I was able to get values out of both. The UART terminal was my preferred debugging method because I could easily see if a process was executing correctly or if a variable held the correct value(s) with some well placed print statements in my code. I started with a single channel input on the ADC, connected a single DMA, and allocated one looped descriptor initially. Once I had worked through a few DMA bugs I was able to finally start working with the values themselves to figure out how I would transform those values into something the LCD display would understand. Once I was able to get data on the LCD display, I removed the blocking print statements and made the rest of my debugging decisions based off of changes on the display. If I needed to retrieve specific values, I would print them onto the display or reintroduce the print statements for that specific inquiry.

4 Results

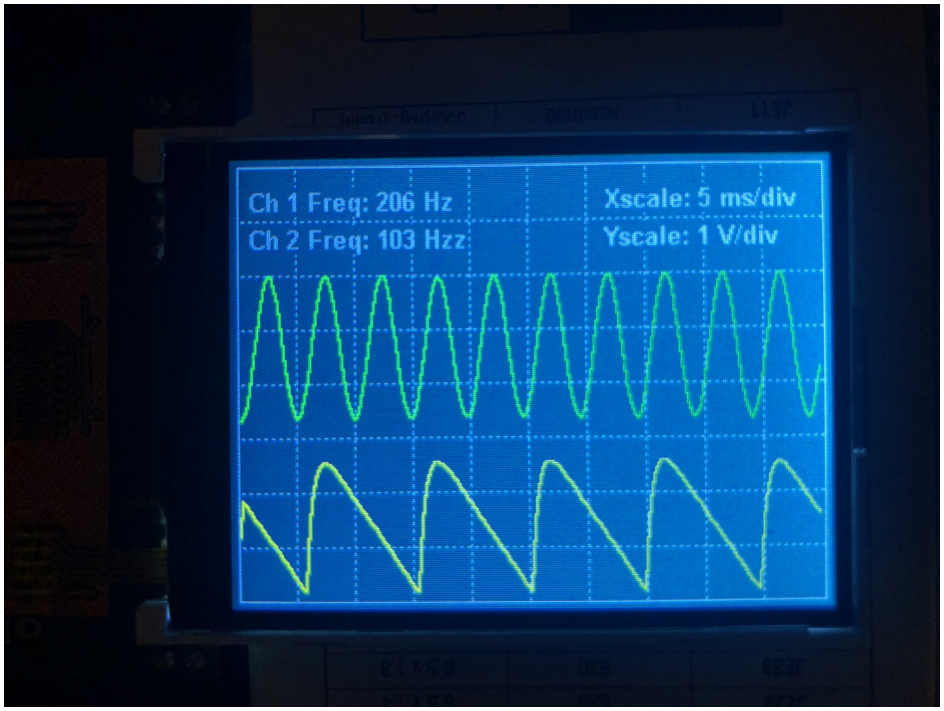

Figure 3: Channel-1 Positive Edge Triggered @ +1.0V

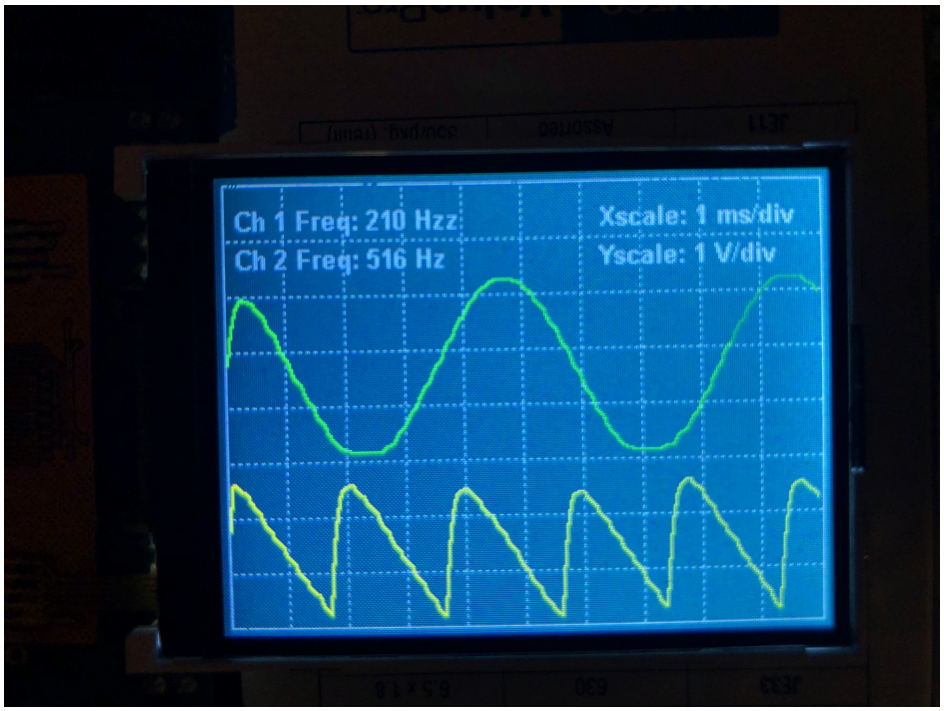

Figure 4: Channel-2 Negative Edge Triggered @ +2.0V

As you can see, the data smoothing with the software low pass filter implementation worked beautifully. The channels themselves worked with waveforms in the frequency range of 1Hz to 1KHz, although the frequency calculations were skewed when less than two cycles where displayed on the screen. This is due to the Nyquist-Shannon sampling theorem since my software calculated the frequency only once per cycle. For my practical purposes the frequency calculation worked well enough even when the x and y scales changed. The x and y scaling worked really nicely as well.

5 Conclusion

This project was very useful in understanding and interfacing microcontroller components and concepts we learned in class. Although this oscilloscope only has a 0V to +3.3V amplitude range and ¡ 1KHz frequency range, it was insightful to the fundamental process of building a more advanced oscilloscope design. I could further improve on this prototype by building a nicer user interface, better accuracy, and/or including more versatile trigger options. The most prominent limitations I would face in the future are the maximum sampling rate of the ADC for the resolution of the oscilloscope and providing stable waveform data with high frequency harmonic noise. To begin to tackle those problems I would need to use a more specialized DSP approach with hardware that would meet that level of implementation. Overall, I feel confident in being able to design and build my own microcontroller systems in the future.