TinyScope

Kyle Jeffrey

Abstract—For the final project in this course, an oscilliscope was made using the PSoC 6 Microcontroller.

1 Introduction

For the final project, the PSoC-6 Microcontroller along with the NewHaven Display 240x320 TFT LCD display is used to design a dual-channel oscilloscope. With the following features:

- Sampling rate of 250 ksamples/second.
- Single or dual analog channels
- Free-running or trigger mode
- Configurable trigger channel, trigger channel and slope
- Configurable X and Y Scales
- Frequency measurement and display
- Potentiometer for scrolling Waveforms

2 Hardware Design Overview

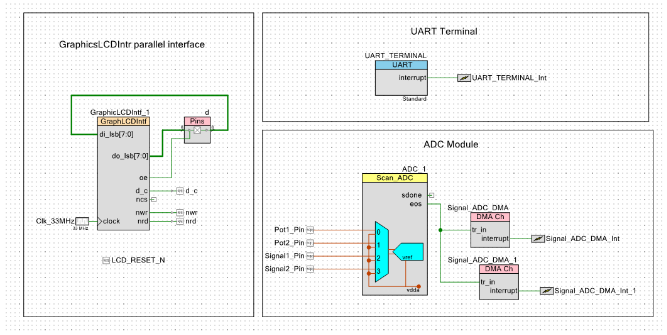

Fig. 1. Component PSoC Creator Design

The basis of the Tinyscope oscilliscope is an Analog to Digital Converter(ADC) that reads two input analog signals, and two input Potentiometer readings. The analog signals, once converted to digital, are routed to software buffers using a Direct Memory Access component (DMA). This greatly reduces the software needed, and increases performance. On top of this is the LCD Screen for displaying the ADC readings from the analog signals, and the UART Terminal, which is used to change settings of the Oscilliscope like trigger mode and trigger level, through a UART communication, with a computer using a USB connection.

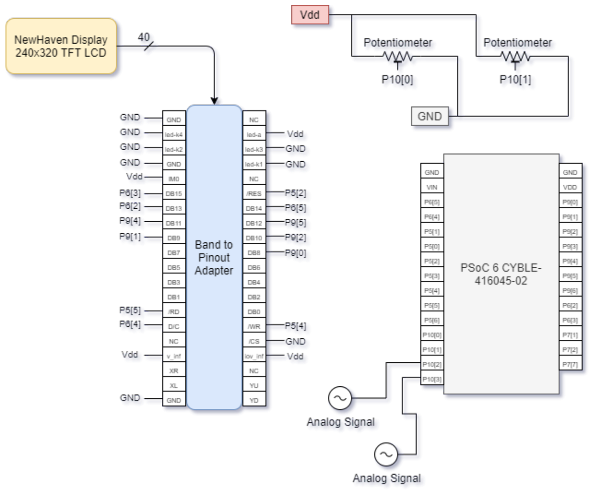

Fig. 2. Top Level Block Diagram

2.1 Design Decisions

The primary design freedom's revolved around the data flow, particularly related to application of the DMA's. The final design used two DMA's, one for each analog signal input, but no DMA for the two Potentiometer. I opted to poll for the data from Potentiometer in the main loop rather than implement hardware memory transfer, as only one value was needed from each Potentiometer while the Analog Signal's data was stored in a buffer of size 310.
Each analog signal uses a DMA, but why? Well, in order to capture 310 data points, display it

to the screen, and calculate each analog signals frequency, the data buffer needs to be processed without any data in the array changing. So, each analog signal used a ping-pong buffer system. Once a ping buffer is filled, a flag is set, and then the pong buffer will be filled, giving the system time to process the data. So each has a ping pong buffer system, requiring seperate DMA's.

3 Software Design

The software is the lionshares of labor. The system uses UART to process commands from a terminal input, draws to the LCD Screen using the emWin library, and Processes ADC Data. The primary software design goal was to modularize the system using seperate header and source files, while keeping a simple main loop.

3.1 Program Flow

3.1.1 Main Loop

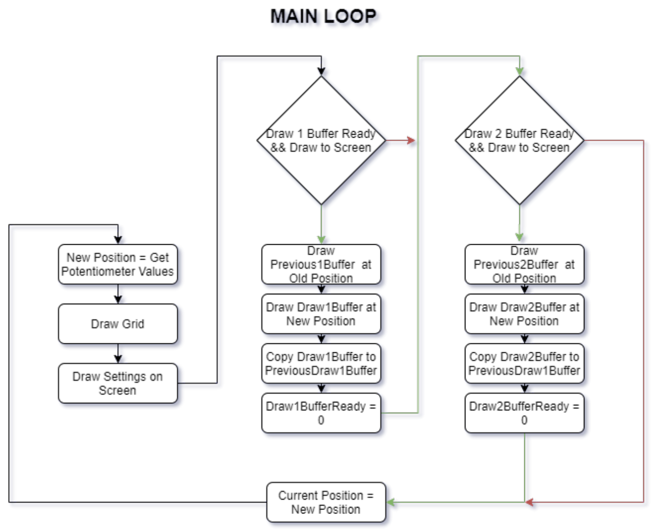
MAIN LOOP
Draw 1 Buffer Ready && Draw to Screen
Draw 2 Buffer Ready && Draw to Screen
New Position = Get Potentiometer Values
Draw Grid
Draw Settings on Screen
Draw Previous1Buffer at Old Position
Draw Draw1Buffer at New Position
Copy Draw1Buffer to PreviousDraw1Buffer
Draw1BufferReady = 0
Draw Previous2Buffer at Old Position
Draw Draw2Buffer at New Position
Copy Draw2Buffer to PreviousDraw1Buffer
Draw2BufferReady = 0
Current Position = New Position

Fig. 3. Main Loop Flowchart

The main loop is intentionally kept pretty simple. The draw buffers are filled from Interrupt Service Routines (ISRs) triggered by the DMA's. In the ISRs, manipulation occurs to fit the raw adc data buffer to a draw buffer. The system has x and y scaling to change the size and scale of the wave being drawn to the screen. All of this manipulation is done within the ISR's.

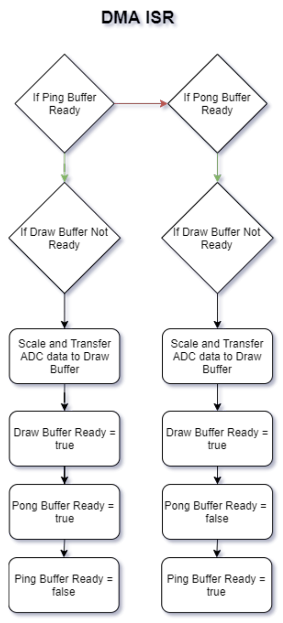

Fig. 4. DMA ISR FlowChart

3.1.2 DMA ISRs

There are two identical DMA Interrupt Service Routines that are triggered when of the analog signals fills a 310 sized array of data points. Using the ping-pong buffer system, when either the ping or pong buffer is filled, the interrupt is triggered and follows the above routine. The user of the tinyscope is able to change the xscale and yscale of the display. This manipulation is done when the data is transferred from the ADC 310 Buffer to the 610 Array Draw Buffer, which is used to draw in the main loop. Again, there are two of these ISR's, one for each analog signal input.

3.1.3 UART Terminal Command Parser

The communication with the TinyScope was done through a USB connection using the UART communication protocol and the stdio c library. The UART triggered an interrupt on every character recieve on the RX line, and would add this to character to a building string until a new line character was read. The built string was then used to determine what command had been inputted using if else and strcmp statements.

4 Testing

The system was fairly complex in the end, so the system had to be built slowly, with testing every new feature. It began with familiarizing with the emWin graphical library and wiring in the LCD display. It took some extensive work to get the LCD display in and assure that every connection was correctly made. seeing as there were approximately 15 connections to make, it was very easy to misconnect a wire.

Testing for most of the project involved placing printf in areas to check data values, or to see if the program crashed before reading the printf. Often it was the case that the program would crash due to segmentation fault.

5 Conclusion

This project introduced developing a full system from start to finish. It delved into developing for a live display system and data processing concurrently. I learned the importance in modular design to maintain seperation between the segments of code. Modular design allows for a better understanding of how the code flows, and a far easier time debugging. In conclusion, I learned a lot. I'd improve this design in the future by adding more of features like trigger levels, and the frequency calculator which I had struggled to implement.

Kyle Jeffrey is a Senior Robotics Engineering Student at 6the University of California Santa Cruz. He is the Secretary of the Engineering Fraternity Tau Beta Pi and the lead Hardware Engineer at the on campus startup Yektasonics.