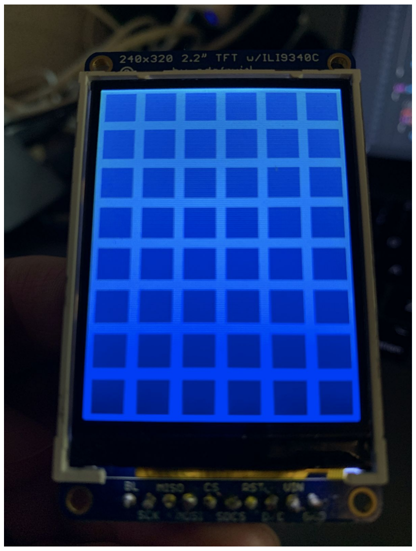
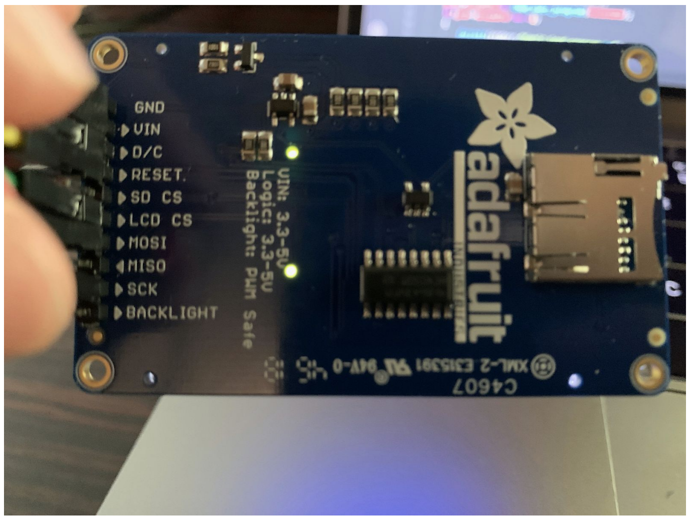
ECE 153A Lab 2B: The Display

Luke Lewis | Eduardo Olmos

Purpose

In this lab we will be connecting an LCD monitor to our FPGA board, and build a QP-nano automata to handle multiple peripherals on the board by updating the display on our connected LCD monitor. The board inputs include the board's built in push-button switches, as well as a rotary encoder. On top of a generated background, twists of the rotary encoder will move a bar either up or down the monitor, akin to a volume bar, while each button press displays a message on the monitor. Furthermore, we will implement a 2 second timeout which removes the progress bar from the monitor when idle.





LCD front with background

LCD back

Results

1. **Building our background**

   The background pattern was built by writing a for loop which divided the monitor into 8x6 squares with borders surrounding them. The drawing was done via the fillRect() function, utilizing the iteration index and modulus to make sure we are drawing squares in the right position.

2. **LCD**

   The LCD monitor was acquired from adafruit. We had no issues with the delivery or set up.

3. **Board inputs and LCD integration using QP-nano**

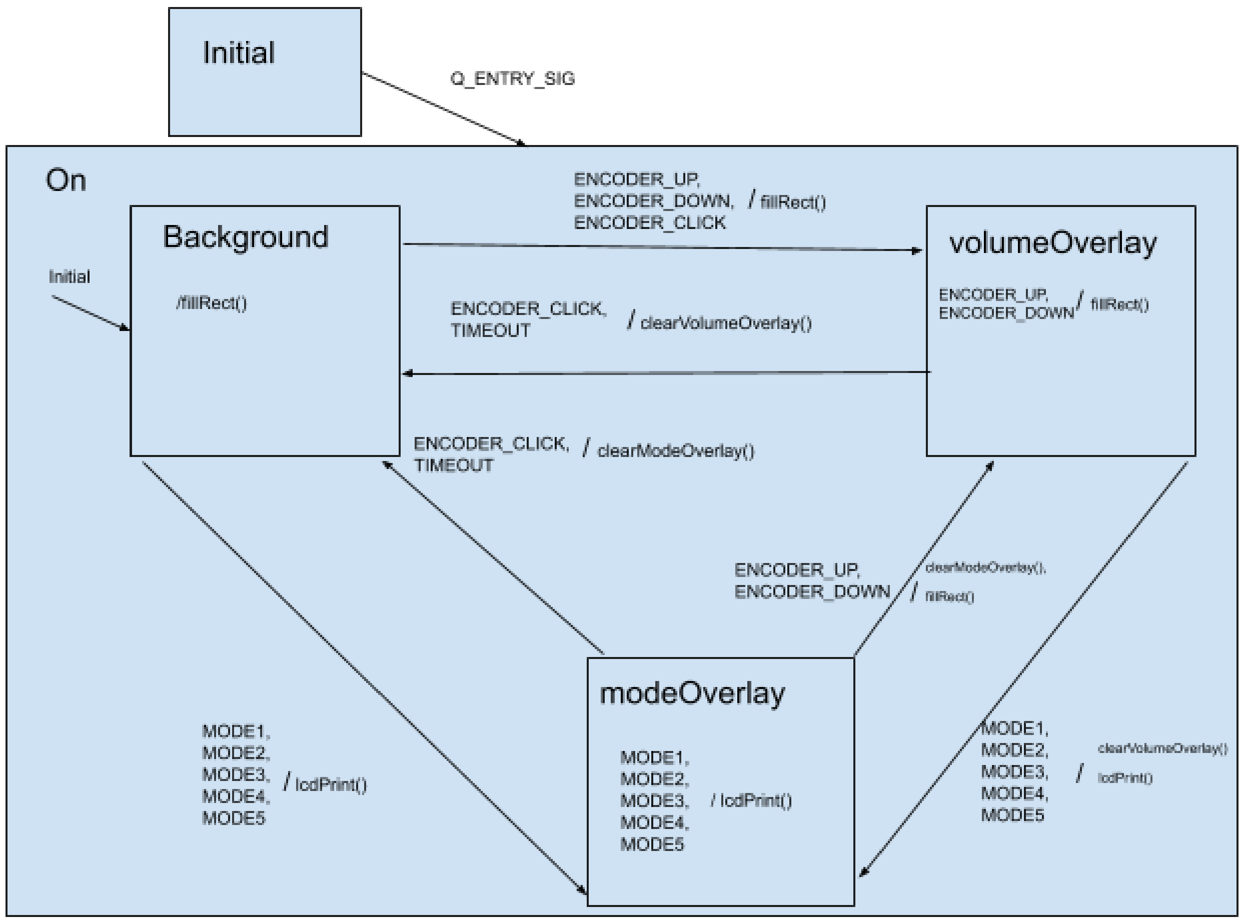
   The FPGA board inputs (buttons and rotary encoder) are detected using interrupts much like our previous labs, which are then debounced. The QP-nano integration of the LCD is done by implementing states to characterize states our machine can be in (initial, on, background, volumeOverlay, and modeOverlay), and then sending signals to those states from our interrupt handler functions, which are then dealt with accordingly, depending on the state and signal. For example, if we are in the state volumeOverlay, i.e. the volume bar is showing on the screen, and receive a signal MODE1 (button press) from our handler, the volume overlay will be cleared, and we will write text on the screen corresponding to the button press. This behaviour is slightly different depending out our states (e.g. if we are in the background state, clearVolumeOverlay() is not called, since the overlay isn't there).

4. **Inactivity - detection and overlay removal**

   We detect inactivity using the boards built in timer, which is reset to 0 after every input interaction we have with the board. If the 2 second deadline is met (i.e. no inputs in 2 seconds), a TIMEOUT signal is sent to our FSM model, which updates our LCD monitor.

   We found that the simplest solution to removing the overlaying graphic or text is simply redrawing the background image over it, since we already have that code written and available.

Statechart



Conclusion

Through this lab, we were able to create an automata with our FPGA board and LCD monitor, using QP-nano to model such a system that takes in inputs from our rotary encoder and buttons located on our board, which are then sent as state signals which update the LCD monitor and our state machine. The steps taken in this lab has given us valuable experience in building automata via QP-nano, generating displays in our LCD monitor, as well as building on previous knowledge of input and interrupt handling on our FPGA board.