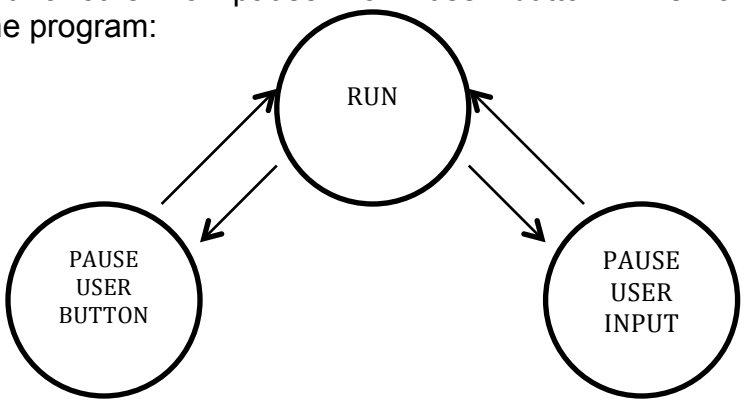
P542 Hardware System Design II

Lab Report 2

Enrique Areyan

Description of solution

The easiest solution was to represent the program as a state machine. Three states were needed: one for RUN and two for PAUSE, one for pause from keyboard and another for pause from user button. The following diagram represents the program:



From the *Run* state we can move to the *PauseUserButton* state if the button is pressed. From the *Run* state we can move to the *PauseUserInput* state if the user types a 'p'. We can only exit the *PauseUserButton* to the *Run* state if the user releases the button. Likewise, we can only exit the *PauseUserInput* state if the user types a character 'r'. Note that all the state machine is wrapped around a `while(1)` and implemented as a `switch` statement.

Description of issues

The main issue was in the delay function used for lab 1. In this function we had a while loop doing nothing and so the program was stopped in a portion of code for a long time. Somehow, and this is still not entirely clear to me, this conflicted with getting the input from the UART module. Strange behaviors occurred, e.g., the program only accepted the first input from the user and not any other.

The key to solving this is to have the user poll (non-blocking get char) right after the `while(1)` and before the switch statement. In this manner, the program always polls data from the user and maintains a clean buffer, thereby avoiding potential crashes.

The best hypothesis I have of why the above issue occurs is that there was an issue with getting the information from the buffer while using the lab 1 type of delay. Perhaps a mismatch between the running time of the main program and the UART module cause an internal exception that I was not able to capture.