Hannah Wilcox

Ask

The purpose of our project is to design a count-down alarm system in which the user can control.
Inputs include time being entered as a measure in milliseconds through a keypad. On the keypad users can enter A to start, B to stop/turn off the alarm, and D to input the time.
Outputs will be an LED and LCD display for the alarm state.
Constraints include that every time an input is entered an LED must turn on, and the LCD display must output “Time Remaining: “ followed by the current time left which is controlled by user input via button press of D followed by a time no longer than 9 minutes and 59 seconds. The LCD display is also controlled by the user pressing A or B on the keypad, to start or stop the countdown.

Research/Imagine

For this project we need to know how to hook up and program the following:

- Keypad
- LCD display
- LED

In order to properly use these peripherals I need to look over the provided documentation on my LCD display type, review the lecture code on using a keypad, and reference my notes for hooking up an LED.

As for possible solutions to the project problem, some solutions are straight forward to what the question is asking in that we must:

- Turn on an LED given user input on the keypad
- Update the LCD display every second with the proper count down time
- Update the LCD display to pause or start given user input on the keypad
- Not allow for the time remaining to be over 9 minutes 59 seconds
- If A is pressed before the user has input a valid countdown time or if the timer has run out, do nothing until there is time stored in the program

Otherwise the rest of my code could have multiple possible solutions for some undefined behavior.

- When the user is inputting a time for the alarm clock my code could either:
    - Know the time is finished being inputted by having the user hit “*” when they’re done
    - Have a timer waiting between inputs, and if the user hasn’t inputted in more than 5 seconds take the time they’ve put in so far
- When the user has inputted an improper amount of time:
    - Do nothing and just wait for the user to press D and a time once again
    - Have some text on the LCD display flash “Improper time inputted” and wait for the user to press D and a time once again

Planning

Since we are using multiple peripherals for this project it's important to plan out how the connections will work. For this we will have to consult the RM0432 to locate the proper pins on the Nucleo to use.

- We will have to make an LED output, which could mean using pin PB7 and setting the bits to 01 for output using MODER.
- Next to setup the keypad we will use pins PB8, PB9, and PB10 as InterruptIn variables. Each variable will act as a different column on the keypad being activated.
- Separately we'll have the LCD display connected to pin PB11

Once we have all of our peripherals set up we next have to worry about the code for controlling each element.

- The LED will have to be activated via the ODR on GPIOB whenever an input is detected on the keypad. This could be having a method called every time there is a rising edge on any of the InterruptIn variables associated with the keypad. Within the method we do the work for detecting what input we got on the keypad, and also turning on an LED.
- The keypad will work by our code quickly activating which rows are online on the keypad, that way we can properly read which button is being pressed by telling which row was powered and which column was activated. This will have to be a function within a rising edge call, to check the active row and column and then from there we can make assumptions on which button was pressed based on the provided diagrams in class.
- Every time we read an input from the keypad, something on the LCD display may change. We may either have to call a function to turn the display to pause, or unpause and continue counting down. The third option would be having to change the time we count down from given a user input of milliseconds no longer than 9 minutes and 59 seconds.
    - When we have the time ticking down we are then in charge of updating the display every second to count down. This could mean having a variable storing the time in milliseconds, converting that to a format of minutes and seconds, and sending that string over to print on the display in the format of "Time Remaining: " with the time remaining in the space after the phrase.
    - The LCD display updates will also need to be run every second, meaning a loop within the main function to be looped every second.

Generally the functions needed could look as follows:

- An ISR function for each column of the keypad, to be called on each rise of an InterruptIn variable to discover which button was pressed
- A function to quickly translate time in milliseconds to time in minutes and seconds for each update of the LCD display
- Three functions for each input from the user via the keypad
    - One function when the user inputs A to start the countdown
    - One function when the user inputs B to pause the countdown
    - One function when the user inputs D to read a further input of numbers in milliseconds no greater than 599000 and send that data to the variable in charge of keeping track of the remaining time.

Aside from these functions we would additionally want a loop within main to be updating the time remaining on the LCD display every second.

Possible tests include:

- Testing to make sure that the LED isn't turning on when we aren't pressing a button on the keypad
- Making sure that the count down on the LCD display doesn't go below 0 or above 9 minutes 59 seconds
- Seeing that the LCD display updates every second, and isn't skipping seconds
- Making sure the code is reading in the right buttons on the keypad
    - This can be done by printing out the button we expect within the program, and seeing if those correct lines are printed when the code is running and we are pressing buttons

The flowchart below maps out the general behavior of our program: